Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Energy

Eline Teigland Bakke

Investigating Sum of Squares Applicability for Large Signal Stability Certificates in Electrical Systems

A new approach for optimization of stability features in wind energy conversion systems

Master's thesis in Energy and Environmental Engineering
Supervisor: Gilbert Bergna-Diaz
February 2024

NTNU
Norwegian University of
Science and Technology

Eline Teigland Bakke

Investigating Sum of Squares Applicability for Large Signal Stability Certificates in Electrical Systems

A new approach for optimization of stability features in wind energy conversion systems

Master's thesis in Energy and Environmental Engineering
Supervisor: Gilbert Bergna-Diaz
February 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Energy

NTNU
Norwegian University of
Science and Technology

Preface

I would like to express my gratitude to my supervisor, Professor Gilbert Bergna-Diaz, for all his guidance, patience and support during this thesis. Thank you for all the sparring sessions in a relatively uncharted narrow field. I have learned a lot, and gained a deeper understanding of some very complex theory and its application.

I would also like to thank my friends and family for their support. My fellow students gave me motivation, a generally good learning environment and for all the coffee breaks.

Abstact

This thesis continues the work started in the associated specialization project by further investigating the Sum of Squares(SOS) method as a tool for obtaining large signal stability certificates i.e., estimates of the region of attraction, for electrical systems of interest.

The power grid is required to operate in a safe and uninterrupted way. Mathematically, this can be viewed from the lens of system stability. If the system is subjected to a potentially unexpected disturbance, it will safely return to its nominal operating point. Most widely used stability analysis tools are able to guarantee stable operation only when the system is subjected to a *small* disturbance but fall short when the disturbance is *large*, as these analyses are based on *linearization*. This thesis aims to overcome this limitation by performing the analysis directly in the *nonlinear* system, by estimating a *region of attraction* using convex optimization and specifically the *Sum of Squares* method. The estimate of a region of attraction is a strong stability certificate, as it guarantees stability as long as the system's initial conditions stay inside the estimated area; which is estimated due to the difficulty of finding the actual area.

The Sum of Squares method is utilized for this estimation, and becomes central in our investigation, hence large parts of this thesis will focus on its corresponding background and definitions introduced to clarify the method. The method is thoroughly investigated, and all the steps are explained in detail to make the method accessible and reproducible for future work. Another key part of the method is based on a *set containment* equation based on the Positivstellensatz. This equation states that a set is guaranteed to remain inside another if the equation holds. This is exploited to ensure that the estimate of the region of attraction does not exceed the Lyapunov criteria for a stable system. Moreover, the whole optimization problem in the SOS method is based on this set containment equation. To solve this significant optimization problem, a V-s iteration is considered, which consists of five steps to enlarge the estimate of the region of attraction iteratively and its accompanying Lyapunov function.

To learn more about the application of the SOS method, and its programming in particular, the method is first applied to two complementary systems, which are less complex than the wind energy conversion system we want to apply it. This is to learn how to use the YALMIP toolbox well since surprisingly there is so little information on this topic. This application encounters several issues, such as stopping criteria for a while loops and definitions of variables and functions.

Furthermore, we turn our attention to a wind energy conversion system. We implement a leader-follower philosophy where the generator and wind turbine can operate independently from the rest of the system. This is to simplify the system, and give a better chance of the SOS method working. We then apply a PI passivity-based controller to close the loop and control the speed for maximum power extraction. The controller design is accompanied by a thorough equilibrium analysis, which is subsequently used to define the *incremental* dynamics of our system. The SOS method is then applied to this model since the original equilibrium point(zero) was uninteresting to investigate from a large signal stability point of view. The testing on the complementary systems improves our understanding of the toolbox enough to implement it on the WECS, and it is implemented with some success. Immediately, the first optimization problem is infeasible, but this issue is solved by linearizing the derivative of the Lyapunov function. After this, the method works until step four, where the complementary systems encounteres the same issue.

All this results in a product that can be utilized to fill the gap in the implementation of the SOS method on complex electrical systems.

Sammendrag

Denne masteroppgaven fortsetter arbeidet som ble startet i det tilhørende spesialiseringsprosjektet, ved å videre undersøke Sum of Squares metoden som et verktøy for å oppnå stabilitetssertifikater, dvs. Estimasjon av attraksjonsområdet, for elektriske systemer av interesse.

Kraftsystemet må kunne operere på en trygg og uavbrutt måte. Matematisk sett, kan dette bli sett på fra et systemstabilitets perspektiv. Dersom systemet er utsatt for en potensiell uforutsett forstyrrelse, vil det trygt returnere til sitt nominelle driftspunkt. Stabilitets analyse verktøy blir mest brukt for å kunne garantere stabil drift under *små* forstyrrelser men kommer til kort når forstyrrelsen er *stor*, da disse analysene baseres på *linearisering*. Denne oppgaven forsøker å overvinne denne begrensningen ved å utføre analysen direkte på det *ikke-lineære* systemet, ved å estimere et *attraksjonsområde* ved å bruke konveks optimering og mer spesifikt *Sum of Squares* metoden. Dette estimatet av attraksjonområdet vil fungere som et sterkt stabilitetssertifikat, da det garanterer stabilitet så lenge systemets startbetingelser holder seg innenfor the estimerte området, som er estimert på grunn av vanskligheten av å finne the faktiske området.

Sum of Squares metoden blir brukt til denne estimeringen og blir sentral i vår undersøkelse. Og store deler av denne masteroppgaven vil fokusere på den korrespoderende bakgrunnen og definisjoner som blir introdusert for å avklare metoden og dens bruk. Metoden blir grundig undersøkt, og alle stegene i metoden blir nøye gjennomgått for å gjøre metoden tilgjenglig og reproduserbar for fremtidig arbeid. En annen viktig del av metoden er basert på *inneslutningsligningen* som er basert på Positivstellensatz. Denne ligningen sier at et sett garantert er inni et annet dersom ligningen holder. Dette blir utnyttet for å forsikre at estimatet av attraksjonsområdet ikke overskrider Lyapunov kriteriene for et stabilit system. Dessuten er hele optimeringsproblemet i SOS metoden basert på denne inneslutningsligningen, og for å løse det store optimeringsproblemet blir V-s iterasjon vurdert. Denne består av fem steg for å forstørre estimatet av attraksjonsområdet iterativt og dens tilhørende Lyapunov funksjon.

For å lære mer om anvendelsen av SOS metoden, og dens programmering, blir metoden først anvendt på to komplementære systemer, som er mindre kompleks enn vindenergi omdannelses systemet(WECS) vi ønsker å anvende det på. Dette er for å lære hvordan YALMIP verktøykassen virker siden det overraskende nok fins lite informasjon om dette temaet. Anvendelsen fører til flere problemer, blant annet stoppkriteriet for while-løkkene og definisjoner av variabler og funksjoner.

Videre retter vi oppmerksomheten vår mot et vindenergiomdannelsessystem. Vi implementerer en leder-følger filosofi der generatoren og vindturbinen kan operere uavhegig av resten av systemet. Dette ble gjort for å gjøre systemet enklere å håndtere, og gav en bedre sjanse for SOS metoden å fungere. Deretter blir en PI passivitets basert regulator anvendt for å lukke løkken og kontrollere hastigheten for maksimal kraftutvinning. Regulatordesignet blir supplert av en grundig likevektsanalyse, som deretter ble brukt til å definere den *inkrementelle* dynamikken med hensyn til dette. SOS metoden ble så anvendt siden the originale likevektspunktet(null) var uintressant å undersøke fra et stor signalfølsomhetsperspektiv. Testingen av komplementærsystemene forbedret forståelsen vår av verktøykassen, nok til å implementere metoden på WECS og det ble implementert med noe suksess. Med en gang, ble det første optimeringsproblemet ikke løslig, men dette problemet ble løst ved å linearisere den deriverte Lyapunov funksjonen. Etter dette fungerte metoden frem til steg fire, som også stoppet komplimentærsystemene.

Alt dette resulterte i et produkt som kan bli brukt til å fylle hulllet i implementeringen av SOS metoden i komplekse elektriske system.

Table of Contents

List of Figures

List of Tables

List of Codes

List of Abbreviations

2L-VSC	Two-Level Voltage Source Converter
CPL	Constant Power Load
d-axis	Direct Axis
LCF	Lyapunov Candidate Function
LF	Lyapunov Function
LMI	Linear Matrix Inequality
LSS	Large Signal Stability
PBC	Passivity Based Controller
pH	port-Hamiltonian
PI	Proportional Integral
PMSG	Permanent Magnet Synchronous Generator
q-axis	Quadrature Axis
ROA	Region of Attraction
RoCoF	Rate of Change of Frequency
SDP	Semi Definite Programming
SOS	Sum of Squares
TSA	Transient Stability Analysis
WECS	Wind Energy Conversion System
YALMIP	Yet Another LMI Parser

Chapter 1

Introduction

This chapter gives an introduction, first with background and motivation to the work completed in this master thesis. Then, the objectives and the methodology are presented to give information on the contributions of the thesis and how these objectives are met. The scope of the thesis is then limited by making certain simplifications. Lastly, the thesis structure is presented and gives a chronological overview.

1.1 Background and Motivation

To achieve The Paris Agreement's goal to hold *"the increase in the global average temperature to well below 2°C above pre-industrial levels"* and pursue efforts *"to limit the temperature increase to 1.5°C above pre-industrial levels"*[1], serious changes needs to be made. The general secretary of the United Nations, António Guterres, said *"Science tells us that limiting global heating to 1.5 degrees will be impossible without the phase out of all fossil fuels on a timeframe consistent with this limit. This has been recognized by a growing and diverse coalition of countries*" in his closing statement at the UN Climate Change Conference COP28 in December 2023.[2] With the increasing energy demand in the world due to the growing population and this out-phasing of fossil fuels, a massive increase in renewable energy will be necessary. In this expansion of renewable energy, huge advancements need to be made in the technology. Areas that have significant expansion potential are solar and wind power, and the technology has considerable potential in these areas.

In today's changing energy system, more and more renewable energy sources are integrated. One of the most up-and-coming renewable energy sources is wind energy, and most of the expansion is located in offshore wind energy.[3] According to the Global Wind Energy Council's report from 2023[4], a total of $130GW$ offshore wind is expected to be added in 2023-2027, which will increase the share of new installations up to 23% by 2027. This thesis will focus more on offshore wind power, but the methods and stability theory that will be presented are easily transferrable to other branches of energy production. This is because the focus is on the generator that the power source is connected to more than the power source itself, which makes this stability theory interesting for the entire industry, not just wind power. For this thesis, the focus will be on wind power, but as mentioned previously, the theory readily applies to other areas of energy production.

Today's energy system is changing, and the traditional setup for the power grid has a few large energy production sites. However, this setup is changing as smaller energy production sites, especially renewable energy production, are connected to the power grid. Renewable energy production has

greatly increased in recent years, and wind energy itself is expected to reach $2TW$ by 2030[4]. The power grid is not built for large-scale injection by this many small inputs. These small inputs risk making the grid unstable and have unknown production due to variable wind and solar conditions, which causes problems with the security of supply.

When connecting renewables, in this case wind energy conversion systems(WECS), it is important to guarantee stability through a stability certificate. The certificate guarantees that the system stays stable during large disturbances and can safely be connected to the power grid without jeopardizing its stability.[5] The stability certificate often comes as a criterion where a tuning variable has to be carefully selected for the certificate to hold. In this thesis however, a more practical viewpoint is adopted where the control system has been pre-designed without being necessarily backed up by a formal stability proof. Instead, we are interested in investigating how large is the stability region of such given controller. Towards this end, the aim is to investigate the so-called *region of attraction*(ROA) for the system in question(WECS).

Furthermore, in this context of ever-growing interconnection of renewable energy sources, Plug-and-play features from a stability perspective are desirable, and facilitate their connection and power grid integration. It will be much easier if the system has such a stability certificate. The title of the associated specialization project is "Plug-and-play control of PMSG-based Wind Turbines", and the general goal in this type of research is to allow plug-and-play features that allow for easy integration of renewable energy sources into the power grid. This will again contribute to the rising energy demand and help global warming and the larger picture.

With high penetration of renewables comes high input of power electronics in the form of converters. These do little to contribute to the power grid's inertia, which is important for transient stability in the grid.[6] Inertia is used to absorb sudden changes in supply and demand in the grid and comes from large synchronous generators, which are normally used at large energy production sites. *"A direct and widely reported consequence of increasing levels of non-synchronous generation (wind and PV -variable RES generation, which is Power Electronics Interfaced to the system) is a decline in power system rotational energy or system inertia, leading to higher RoCoF values."*[6] RoCoF is defined as the Rate of Change of Frequency and refers to the speed of change of frequency in the power grid. The key to a stable power grid and supply security is stable frequency, and the higher the RoCoF, the more risk of instability in the system, which can lead to damage to the grid and the equipment.

For offshore wind, the main focus of this thesis, two types of generators are commonly used: the standard induction generator and the permanent magnet synchronous generator(PMSG). Where the induction generator is simple, large, robust, and relatively cheap[7], the PMSG is compact, relatively light, and highly efficient.[8] The PMSG also has self-excitation, which makes the normal magnetizing current absent, and the generator is able to operate with a higher efficiency than other generators.[9] Due to this higher efficiency, this thesis will investigate a wind energy conversion system that includes a PMSG.

Implementing new energy sources to the power grid requires guarantees that the system can stay stable under both minor and significant disturbances. Small signal stability investigates the system stability under steady state and is helpful in many instances, but large signal stability is crucial for the system's stability. This large signal stability is increasing in popularity, and is researched more and more, making it very relevant.

Large signal stability can be investigated in different ways. However, this thesis will focus on estimating the region of attraction, which is a reliable measure of how significant disturbances a system can handle without becoming unstable.[10] Usually, this region of attraction is estimated using Lyapunov-based methods or utilizing polynomial optimization, and this thesis will combine

these two methods by utilizing the Sum of Squares method. This method employs Lyapunov conditions and implements them into an optimization problem that optimizes a Lyapunov function with the largest estimate of the region of attraction for the system.[11]

1.2 Objectives and Methodology

The main objective of this master's thesis is to continue the work of the associated specialization project on the Sum of Squares method for optimizing the estimate of the region of attraction by using the Lyapunov stability theory. The known optimization toolbox, YALMIP, is explored and utilized for the programming to achieve this objective.

The publications used as background, [11] and [12] to name a few, apply the Sum of squares method on simple systems with very little information on the *implementation* of the technique itself, possibly due to page limitation restrictions. This, arguably hinders the popularization of the method in the electric power community where it is clearly needed to complement time-domain simulations with more general large-signal stability studies. Thus, the main objective of this thesis is to understand and explain all the theoretical and practical details related to the implementation of the application of the sum-of-squares method to an (electric) system, including a fair amount of code to make the method more accessible. We view this effort as a necessity for future related (master) projects to get familiarized with such a complex topic within their allocated time.

Objective 1: *Explain complex large-signal stability & control concepts in a clear and accessible way and apply these to a wind energy conversion system.*

Objective 2: *Provide a clear and accessible explanation of the optimization of the estimate of the region of attraction, utilizing the Sum of Squares method and V-s iteration.*

Objective 3: *Provide an accessible and reproducible explanation of how to use the YALMIP toolbox for the application of the method.*

Objective 4: *Apply the method to two less complex complementary systems, a time-reversed van der Pol system and a constant power load system, and identify and investigate challenges.*

Objective 5: *Introduce and apply the method to a wind energy conversion system, and identify challenges for further investigation.*

1.3 Limitation of Scope

Several assumptions regarding the system model under investigation limit the scope.

The external grid is modeled as a constant voltage source. This comes from the assumption that the external grid is stiff. A stiff grid is characterized as stable and does not experience significant changes in frequency or voltage. These changes will not impact the system significantly. This assumption will become irrelevant when the later introduced leader-follower philosophy is implemented since the external grid is part of the follower and does not affect the leader in any way.

Furthermore, some simplifications are made regarding the model of the synchronous generator. Primarily, the resistance and inductance are independent of frequency, magnetic saturation, and hysteresis losses, which are neglected. Another assumption is that the magnetic field is evenly

distributed in the generator, and the field created by the magnets is sinusoidally distributed in the generator. This is done to make the system easier to handle since this simplification will not significantly affect the control of the generator.

A damping coefficient, d, is introduced in the equation for the synchronous generator. This damping comes from the generator's damper windings and represents the electromagnetic friction, damping the speed of the generator. Damping caused by friction is neglected since this is insignificant in comparison. To include the flux linkages affecting the damper windings would require a more advanced model and is not studied in this specialization project.

Later, when applying a leader-follower philosophy, several assumptions are made. It is assumed that there is reliable communication between the parts of the system, and the command structure must be clear. The leader and follower are considered to be synchronized in operation and timing. There is also an assumption regarding the follower's ability to follow the leader's command.

1.4 Thesis Overview

This master thesis builds on the work of the associated specialization project[13], and some of the work is reproduced to provide a more complete view of the new contributions. It is divided into six chapters, where the first three are focused on the theory and explanation of a method. The following two are the application of both the theory and the method, and the last chapter contains the conclusion and possible further work.

Chapter 1 Introduction: contains the introduction, outlining of the thesis objectives, the limitation of the scope and an overview of the thesis.

Chapter 2 Theory: introduces several important theoretical terms, which are essential background for the work conducted in the thesis. Lyapunov stability theory, Positivstellensatz and information on the region of attraction are some of these.

Chapter 3 The Sum of Squares Method: provides a thorough investigation and explanation of the Sum of Squares method and the steps in the V-s iteration.

Chapter 4 Complementary Systems: applies the SOS method to two complementary systems to learn the method and investigate issues that arise. This chapter contains a lot of code to provide an accessible and reproducible step by step explanation of how to use the YALMIP toolbox.

Chapter 5 Wind Energy Conversion System: introduces the wind energy conversion system and implements the leader-follower philosophy and a PI Passivity Inspired Controller to the system. An incremental model is applied to move the equilibrium point away from the uninteresting point of zero. Then, the SOS method is applied, and it ends with a discussion on the application of the method on all three systems and issues that arose.

Chapter 6 Conclusion and Further Work: The thesis is concluded with a conclusion and suggests further work.

Chapter 2

Background Information

In this chapter, basic concepts and theories are introduced and examined. Most of the theory is based on or reproduced from the specialization project. The theory is essential for understanding the presented work. Some of it is improved and elaborated for a better understanding and completeness. The chapter on the Positivstellensatz is new work that contributes to a deeper understanding of the Sum of Squares method. Lastly, there is an introduction to the chosen software, the YALMIP toolbox in MATLAB, where some of the most central functions that will be utilized later in the thesis is explained.

2.1 Nonlinear Systems and Equilibrium Points

Real-world systems are often simplified to linear systems within an operating range or around an equilibrium point. This simplification makes working with the system, calculations, and simulations much easier. The real-world systems are linearized since nonlinear systems are unpredictable and potentially unstable than linear systems. Historically, control systems for electrical circuits have assumed linearity to simplify design and analysis. These simplifications work when the system's state is close to equilibrium. A general, time-invariant nonlinear system is defined in equation (2.1). Here u is the input, a system function, and y is the output.

$$\dot{x} = f(x) + g(x)u, \quad y = h(x) \tag{2.1}$$

An important topic throughout this thesis is the equilibrium point. In electrical power systems, an equilibrium point represents a stable operating condition that ensures stability in the system. Khalil's book on Nonlinear systems[14] explains it as: *"An equilibrium point is stable if all solutions starting at nearby points stay nearby; otherwise, it is unstable. It is asymptotically stable if all solutions starting at nearby points not only stay nearby, but also tend to the equilibrium point as time approaches infinity."*

The equilibrium points are generally found through a steady-state analysis where the time derivative of the state variables of x is set equal to zero. A nonlinear system can have both none and multiple equilibrium points, making it more complex and challenging to explore.

2.2 Large Signal Stability

When investigating stability, the industry has historically focused on small signal stability. Small signal stability uses linearization to examine a system's response to a minor disturbance. This topic has been extensively researched, and the industry is well-versed in this. When analyzing small signal stability, many assumptions are made. Some of the assumptions, are using linearization and only looking at minor disturbances, making the result unreliable in the general case. This thesis will explore large signal stability(LSS), where non-linear behavior will be considered. Small signal stability is easier to understand and analyze while giving valuable insight into the system's dynamic behavior. It also only considers minor disturbances close to the equilibrium point. At the same time, LSS can look closer at more significant disturbances that move the system further away from the equilibrium point.

Another essential term specific to large signal stability that analyses the system's ability to withstand severe disturbances is transient stability analysis(TSA). It looks into the dynamic behavior of an electric system after a disturbance has occurred and checks its ability to maintain synchronism.[11] TSA aims to look into a dynamic system's behavior and investigate its characteristic response as it returns to standard operation. Typical approaches to this analysis are based on energy functions and Lyapunov functions. Therefore, the optimized Lyapunov function, which gives the largest region of attraction, will be developed in the master thesis and will be a perfect application for transient stability analysis.

2.3 Lyapunov Stability

The wind energy conversion system analyzed later in this thesis is a dynamic system, which is defined as a system that consists of electrical and control components that respond to changes in operating conditions and exhibit time-varying behavior. Lyapunov stability theory is universal and can be applied to a wide range of dynamical systems, both linear and nonlinear, and is often used in the context of nonlinearity.

In this thesis, Lyapunov stability analysis is utilized in a control and power system setting but can also be used in other fields. Lyapunov stability can be used to model predictive control of economic systems.[15] It can also be utilized in flight dynamics and aircraft to design control systems for maneuverable and stable flight.[16]

Some advantages of Lyapunov stability theory include universality, rigid foundation, robustness assessment, and design.[17] Universality means that it can be utilized in many different types of systems, which can be applied in diverse fields. The rigid foundation focuses on the systematic approach to analyze and prove stability, which provides the stability certificate. If a stability certificate is obtained, i.e., the criteria hold for the Lyapunov function(LF), this contributes to stabilizing the power grid. The power grid must maintain stability to avoid blackouts and destruction of equipment. The stability of the power grid also ensures reliable power supply, grid resilience, system protection and grid efficiency. It also keeps the power grid from causing safety hazards. In this thesis, the region of attraction will be investigated. This investigation is a way of doing a robust assessment, which is an advantage of the stability theory. It can also be used when designing control systems for an electrical system and allows engineers to stabilize the system.

Mathematically, the easiest way to describe a system is to do it linearly. This makes all the calculations more effortless, making the system easier to manage. However, most real-life electrical systems are nonlinear regardless of the system type. These systems are more challenging to handle

mathematically. Therefore, many assumptions are often made to simplify the nonlinear system to a linear one when doing calculations. This makes the calculations less trustworthy and will only be used in this thesis as a starting point in the V-s iteration discussed later in chapter 3.3.

Lyapunov stability theory provides a systematic framework for studying stability in nonlinear systems. This framework is centered around a LF and some constraints. However, the Lyapunov equation in (2.2) applies to a linear system, given the presence of the known A-matrix. This equation will, however, be utilized in the later introduced Sum of Squares method for calculating a starting LF. The equation is described by theorem 3.4 in [18] and is presented in equation (2.2), where $P, Q \in \mathbb{S}^n$ and $A \in \mathbb{R}^{n \times m}$. If A is Hurwitz, which means that all the eigenvalues of the matrix have negative real parts and $Q > 0$, there exists a unique solution P for (2.2), and this solution satisfies $P > 0$ [18].

$$PA + A^T P + Q = 0 \tag{2.2}$$

The LF quantifies the energy of a system, and a stable system's energy will decrease over time, while the energy will remain the same or increase for unstable systems. Further, the stability conditions for the LF are defined below:

$$\underbrace{\begin{matrix} V(0) = 0; & V(x) > 0 \in \mathbb{D}^n \setminus (0) \\ \dot{V}(0) = 0; & \dot{V}(x) \leq 0 \in \mathbb{D}^n \setminus (0) \end{matrix}}_{\text{Global stability}} \qquad \underbrace{\begin{matrix} V(0) = 0; & V(x) > 0 \in \mathbb{D}^n \setminus (0) \\ \dot{V}(0) = 0; & \dot{V}(x) < 0 \in \mathbb{D}^n \setminus (0) \end{matrix}}_{\text{Global asymptotic stability}}$$

A LF, $V(x)$, needs to satisfy three conditions to serve as a Lyapunov function, guaranteeing stability. First, the function needs to be continuously differentiable. The second condition is specified in the first line of the conditions, where it is specified that the LF needs to be equal to zero at the equilibrium point and positive definite for all other values of x. The last criterion revolves around the derivative of the LF, which should be negative semidefinite or negative definite, depending on the type of stability, except at the equilibrium point, where it should be equal to zero. If the conditions are fulfilled, this results in $\mathcal{V}$ being an LF for $x = 0$. This means the system is asymptotically stable at equilibrium $x = 0$. In these conditions, D is defined as the set that holds for $\mathcal{V}(x) > 0$ and will be used further in this thesis.

An LF is constructed for a given system to analyze stability and behavior. If the candidate function does not meet the conditions, it means that the search for a LF is not over and must be continued. If the candidate satisfies the global stability conditions, minor disturbances will not deviate the system significantly from the equilibrium point. Global asymptotic stability ensures that the system converges to the equilibrium point over time, regardless of the starting point.

The Lyapunov function quantifies the energy of a system. Therefore, Lyapunov's theory is often used in context with port-Hamiltonian modeling, a mathematical way of representing physical systems described by first-order differential equations. This modeling is based on energy conversion and passivity, which will be discussed in chapter 5.3, and is a standard way of expressing nonlinear systems for stability analysis. However, this approach is more theoretical and requires more extensive calculations. A port-Hamiltonian model is made from the differential equations that represent the system, and this model is used to create a Lyapunov candidate function. This candidate function is then tested to see if the Lyapunov constraints hold. If this is the case, asymptotic stability can be guaranteed for the Lyapunov candidate function, i.e., the system. This is called Lyapunov's direct method.

Lyapunov stability theory is still applicable when it is desired to expand the system, in this case from a single wind turbine to a whole wind park, and investigate the stability of the expanded system. It can also be expanded by removing the leader-follower philosophy and applying Lyapunov stability to the original system which includes the back-to-back two-level voltage source converter and the voltage source that mimics the power grid.

2.4 Region of Attraction

It is one thing to determine if a certain point is asymptotically stable. However, often it is more interesting to determine how far from the equilibrium the trajectory can be and still converge to the equilibrium point as the time approaches infinity. This can be determined by introducing the term region of attraction(ROA). It is defined in Khalil's book on nonlinear systems[14]:

Let $\phi(t;x)$ be the solution of $\dot{x} = f(x)$, that starts at initial state x at time $t = 0$. Then, the region of attraction is defined as a set of all points x such that $\phi(t;x)$ is defined for all $t \geq 0$ and $\lim_{t\to\infty} \phi(t;x) = 0$

"For the large-signal stability analysis of the power system, the region of attraction (ROA) is a reliable measure which can answer the question of how large deviations from the equilibrium point under big disturbances in system."[10] The motivation for finding a region of attraction(ROA) is wanting to guarantee asymptotical stability or at least keep track of when it is stable. ROA gives an area where stability is guaranteed if the system's initial conditions stay inside this area. In this case, the initial conditions are the state variables at the starting point of the analysis, i.e., when the disturbance happens. This is useful for the industry when monitoring the power grid. Those monitoring the power grid can detect when the system acts abnormally or when an error occurs. In this case, they can monitor if the state variables are heading outside the ROA. If it is outside the ROA, the system is disconnected from the power grid to protect the power grid, the system, and the generator in particular.

The found estimation of the ROA will work as a strong stability certificate for the system. If the initial conditions stay inside the ROA, the system is guaranteed to stay stable. Stability is beneficial when the system is connected to the power grid and when the system experiences significant disturbances from the connected power grid.

The Lyapunov stability theory discussed in chapter 2.3 will be used to find the attraction region. Sum of Squares programming will be used to find the LF with the largest ROA.

When discussing stability robustness, finding a region of attraction is desired. A region of attraction for an equilibrium point, or domain of attraction as referred to in Domain of Attraction, is defined as *"The set of initial conditions from which the trajectory of the system converges to such a point."* [18]

Khalil's book on nonlinear systems[14] defines a region of attraction, R_A, in equation (2.4) and some properties of it are represented in Lemma 8.1: *"If $x = 0$ is an asymptotically stable equilibrium point for $\dot{x} = f(x)$, then its region of attraction R_A is an open, connected, invariant set. Moreover, the boundary of R_A is formed by its trajectories."*

$$R_A = \{x \in D \mid \phi(t;x) \text{ is defined } \forall\, t \geq 0 \text{ and } \phi(t;x) \to 0 \text{ as } t \to \infty\} \tag{2.4}$$

In the Lemma, f describes the system dynamics, where $x \in \mathbb{R}^n$. D contains the equilibrium point 0 and is defined as a subset: $D \subset \mathbb{R}^n$. The solution of $\dot{x} = f(x)$ is defined as $\phi(t;x)$ and starts at

x when the time is zero, $t = 0$. The equation states that x is inside the set D and is defined for all t larger than zero. It also states that the solution of the differential equation needs to go towards the equilibrium point when the time goes towards infinity. It is important to note that D is not an estimate of R_A, and there are no guarantees that an initial state in D will remain in D forever. To avoid this problem, R_A is defined as a subset of D.

2.4.1 Estimation

It can be difficult, and often impossible, to find the exact ROA in complex nonlinear systems. This would be very complicated and time-consuming. In the industry, however, it is normal to estimate a region of attraction instead. Here, the boundaries of the ROA are estimated and provide valuable information that is used to analyze the stability and performance of the system in question. Estimation can be achieved by simulations or creating Lyapunov functions and perform stability analysis on these. One example of such a method is Zubov's method[14]. In this thesis, optimization will be used to find an estimate of the ROA.

This estimate of the ROA, Ω, is defined in equation (2.5) [12]. Where Ω is a subset of R_A that again is a subset of D, making Ω bounded and contained in D. c is defined as a constant $c > 0$, and the intent is to find the largest c for which the set Ω is contained in D. This c will later be referenced as γ, and is a part of the objective for the optimization problem in chapter 3.2.1.

$$\Omega = \{x \in \mathbb{R}^n \mid V(x) \leq c\} \tag{2.5}$$

2.5 Positivstellensatz

Lyapunov stability theory and region of attraction are based on positivity theory. The systems that will be investigated in this thesis will be described using polynomials. Algebraic geometry applies here since it deals with the solution set of a system of polynomial equations.[19]

Together, this can be combined, and the Positivstellensatz can be utilized. *"The Positivstellensatz gives an algebraic characterization of functions positive on certain semi-algebraic subsets of R^n"*[20]. In other words, this means that positivity theory can be used for algebraic geometry and polynomial systems to prove the positivity need for the Lyapunov conditions described in chapter 2.3. The Positivstellensatz is described in the following theorem from [20].

Theorem 4.4.2 *Let R be a real closed field. Let $(f_j)_{j=1,...,s}$, $(g_k)_{k=1,...,t}$ and $(h_l)_{1,...,u}$ be finite families of polynomials in $R[X_1, ..., X_n]$. Denote by P the cone generated by $(f_j)_{j=1,...,s}$, M the multiplicative monoid generated by $(g_k)_{k=1,...,t}$, and I the ideal generated by $(h_l)_{1,...,u}$. Then the following properties are equivalent:*

1. *The set*

$$\{x \in R^n \mid \ f_j(x) \geq 0,\ j = 1, ..., s,\ \ g_k(x) \neq 0,\ k = 1, ..., t,\ \ h_l(x) = 0,\ l = 1, ..., u\}$$

   is empty

2. *There exist $f \in P, g \in M$ and $h \in I$ such that $f + g^2 + h = 0$*

This theorem proves that the equation in the second property holds by proving that the set in the first property is empty. This mathematically complex theorem can be applied to more general

problems, such as the set containment problem discussed later in this thesis. To reach this set containment problem, Parillo's PhD[19] starts with verifying that the following equation holds.

$$a(x) = 0 \Rightarrow b(x) \geq 0 \tag{2.6}$$

By setting $f(x) = g(x) = b(x)$ and $h(x) = a(x)$ this is true only if the following set is empty.

$$\{x \mid -b(x) \geq 0,\ b(x) \neq 0,\ a(x) = 0\}$$

Then the Positivstellensatz is applied, and the following equation appears, with the polynomials $s(x)$ and $t(x)$ being sum of squares polynomials that are multiplied with each of the polynomials in question, $b(x)$ and $a(x)$ respectfully. This is done to have some adjustable variables. These polynomials can be changed to get the Positivstellensatz to work and prove the initial condition in (2.6).

$$\begin{aligned} f + g^2 + h &= 0 \\ -s(x)b(x) + b(x)^2 + t(x)a(x) &= 0 \\ t(x) &= b(x)r(x) \end{aligned}$$

$$\begin{aligned} -s(x)b(x) + b(x)^2 + b(x)r(x)a(x) &= 0 \ \mid \frac{1}{b(x)} \\ -s(x) + b(x) + r(x)a(x) &= 0 \\ b(x) + r(x)a(x) &= s(x) \end{aligned}$$

implementing that $a(x) = 0$ and $b(x) \geq 0$ this results in the following

$$\begin{aligned} \textit{something negative} + \textit{a polynom} \cdot 0 &= \textit{something positive} \\ \textit{something negative} &= \textit{something positive} \end{aligned}$$

This does not hold and the proposed set is empty. By the Positivsellensatz, this proves that the following equation holds.

$$b(x) + r(x)a(x) = s(x) \tag{2.9}$$

This is one particular solution of the Positivstellensatz, which gives the set containment equation in (2.9), that is used throughout the whole thesis.

2.6 The YALMIP Toolbox

To perform the actual optimization of the region of attraction, YALMIP, a toolbox for MATLAB, will be utilized. YALMIP is a tool often used to model and solve optimization problems. It was initially developed to model and solve semi definite programs(SDP) by interfacing with external solvers, making developing and solving optimization problems very simply.[21] In the beginning, it was intended for SDPs and LMIs, but it has since been developed and now supports several other programming types. Some of these are linear programming, quadratic programming, second-order cone programming, mixed integer programming, and semidefinite programs with bilinear matrix inequalities. These are in addition to the Sum of Squares programming that will be utilized in this thesis.

YALMIP is a high-level modeling language used to formulate MATLAB optimization problems. From the name *"Yet another LMI parser"*, it is clear that the focus is on optimization problems with LMI constraints. It also gives an intuitive way to express semidefinite, convex, and non-convex programming problems. The high-level modeling language ensures the program's syntax is accessible and user-friendly. Complex optimization problems are easily handled by the program's wide range of solvers and advanced functionalities. Another feature that makes dealing with large-scale optimization problems is the program's ability to create hierarchies of optimization problems. This is accomplished by defining subproblems and connecting them using constraints and objectives.

The YALMIP toolbox uses different solvers to solve the actual optimization problems. LMILAB, which is a part of the robust control toolbox, is the solver that is initially used for the optimization. However, this solver is not favored by YALMIP due to its lagging on general problems,[21], and the immediate response is to *"Please do not use LMI Lab with YALMIP. Install another semidefinite programming solver such as MOSEK, SEDUMI or SDPT3"*. For this thesis, MOSEK was downloaded and utilized for all the programming, and to implement this, the code line *ops=sdpsettings('solver', 'mosek');* was needed. Later, the variable *ops* was used in the *solvesos()* function, which is explained later, to choose the solver in that particular optimization problem.

The idea is to use the YALMIP toolbox to solve the optimization problem, i.e., finding the largest estimate of the region of attraction for a system, and this is achieved by utilizing this toolbox and Sum of Squares programming. To utilize this toolbox in a good way, a few functions and their characteristics need to be known. Therefore, the most important ones are mentioned in this chapter.

The toolbox has a specific type of symbolic variable called *sdpvar*, which can be defined in many different ways. The one utilized in this master is $x = sdpvar(n, m)$, with 1x1 variables for all the states in x. For the WECS, this is the different currents, generator speed, and controller variables.

Furthermore, the function $[p, c, v] = polynomial(x, degreeMax, degreeMin)$ is utilized to create an empty polynomial p, with the variables in x from degreeMin up to degreeMax. The vector c contains all the constants in the form of the terms in the vector v and is set up in the way that $p = c' \cdot v$. In this thesis, the degrees will normally be quadratic with $degreeMax = 2$ or 4, and $degreeMin = 2$. The maximum degree could be increased to allow for higher degree polynomials as solutions, but this would come at a computational cost in the optimization problem.

The $constraint = sos(equation)$ is applied to transform equations into sum of squares constraints. This function takes in an equation that should be positive, and the output is a constraint on the form: *Sum-of-square constraint (polynomial)*. This function is used on all the equations that should be sum of squares and put in a vector to be the constraints of the optimization problem. This gathering of constraints results in a constraint defined as a $Ax1$ lmi and contains A lmi constraints.

The function used for the actual optimization is
solution = solvesos(Constraints,Objective,Options,Decisionvariables), where the input is as shown, the constraints, the objective, the options and the decision variables. In this thesis, the *constraints* are made with the *sos()* function, and the objective mostly remains empty. This empty objective is due to the fact that the *solvesos()* function is used for finding feasible solutions, and not for actual optimization because of bilinearity in the constraints and will be discussed later in the thesis. The *options* is in this thesis used to state which solver to use, MOSEK, and the decision variables are often the constants in the vector c from the previously mentioned *polynomial* function. The output of this function is quite helpful and can be studied and have several uses. It can be found in table 2.1 where it is displayed as a $1x1$ struct with six fields, where the last two fields are studied and give information on the feasibility and if there is an issue with the problem. The content is from an example and an issue that arose during the programming.

Field	Value
yalmipversion	'20230622'
matlabversion	'9.14.0.2337262 (R2023a) Update 5'
yalmiptime	0.215182799999999
solvertime	0.004817200000000
info	'Infeasible problem (¡a href="yalmip.github.io/debuggingunbounded" ¿learn to debug¡/a¿) (MOSEK-SDP)'
problem	1

Table 2.1: Content of the 6x1 solution to the *solvesos*() function

This results in the value 1 instead of 0 for the ..... *solution.problem* and can be utilized to check if the optimization problem is feasible. In the thesis, this was used as a stopping criterion for while loops as follows: *while solution.problem == 0.* The information in the *solution.info* says that the problem is unbounded. This is the result of the primal problem being infeasible. The solver immediately tries to solve the dual problem instead(normal procedure in optimization), which often is solvable if the primal is not. If the dual can not be solved as well, the result will say that the dual is unbounded, which is the information from the struct.

The $P = lyap(A, Q)$ function, which is a part of the Control System toolbox, is used to find the P matrix when the A matrix is known. This function uses the Lyapunov equation from (2.2) in chapter 2.3, and the input is the known A matrix and a Q matrix, that will be set to the identity matrix in this thesis.

Other helping functions are utilized for visibility and a good overall understanding during the programming. *value*() is used to extract the numerical value of a decision variable.[21] And *sdisplay* (symbolic display) tries to display a sdpvar object in symbolic MATLAB form.[21]

Chapter 3

The Sum of Squares Method

This chapter dives deeply into the Sum of Squares method and its application. For the application, an iterative algorithm called V-s iteration is utilized, and the steps are thoroughly explained, with information on how to utilize the software for the optimization to make the method straightforward and accessible. The background and theory are reproduced from the associated specialization project[13], and some of it is improved. It is reproduced due to its importance for understanding the method and completeness.

3.1 Sum of Squares Theory

The Sum of Squares(SOS) method is utilized to find the largest estimate of the attraction region. The SOS method is a mathematical optimization technique for solving specific optimization problems with polynomial constraints. It aims to minimize or maximize a polynomial function with constraints on the variables, and it is beneficial when the positivity of polynomials is in question. The SOS method proves non-negativity for polynomial functions, and to state that an expression or a function is SOS is equivalent to stating that it is positive definite. This comes from the simple fact that x^2 is a positive term, and a sum of positive terms must also be positive. This is very useful when discussing Lyapunov stability theory and the conditions that must be satisfied to guarantee asymptotic stability. Morten Hovd explains what the SOS method does in coherence with Lyapunov theory quite well: *"A set of arbitrary shape, which is inside a set with a Lyapunov level set as the border, is progressively enlarged until the limit of negativeness of the derivative of Lyapunov function is reached."*[12]

The Sum of Squares method can be utilized in various areas and in different ways. In this thesis, the method is applied using programming, and it is utilized in control theory by optimizing a Lyapunov function regarding power systems and their stability and robustness. Other areas include signal processing, machine learning, robotics and autonomous systems, and general polynomial optimization.[22] Regarding signal processing, SOS programming is used in signal reconstruction and analysis. Analysis and design of machine learning models can utilize SOS programming, especially to make convex relaxations on nonconvex optimization problems. SOS programming can also aid motion planning for robotic systems and be utilized for trajectory optimization.

Equation (3.1) shows a general form of an optimization problem often utilized in SOS programming, where $g(x)$ and $f_i(x)$ are polynomials. $f_i(x)$ is defined for $x = (x_1, x_2, ..., x_n)$, which are the constraints on the objective function $g(x)$.

$$max\ (or\ min) \quad g(x) \quad \text{s.t.} \tag{3.1a}$$

$$f_i(x) > 0 \ \text{ for } i = 1, 2, ..., m \tag{3.1b}$$

This type of constraint $f_i(x) > 0$, requires a specific kind of programming, which is called semi-definite programming(SDP). SDP is a convex optimization technique that optimizes an objective function with constraints, i.e., a linear matrix inequality(LMI).

The term used in reference to SOS is that a function or term is an SOS and is defined by Hovd in [12] as follows: *"For $x \in \mathbb{R}^n$, a polynomial $p(x)$ is an SOS if there exist some polynomials $f_j(x)$, $j = \{1, 2, ...m\}$, such that*

$$p(x) = \sum_{j=1}^{m} f_j(x)^2 \tag{3.2}$$

Lyapunov functions are commonly chosen as convex, but this is not a requirement as long as the function satisfies the Lyapunov conditions defined in chapter 2.3. This is easiest if the function is convex. In this thesis, the Lyapunov function is chosen to be convex, and therefore, when optimizing the region of attraction, it is natural to use convex optimization.

A convex function is defined in [23]: *"A function $f \ : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of f is a convex set and if for all $x, y \in$ the domain of f, and θ with $0 \leq \theta \leq 1$, we have*

$$f(\theta x + (1 - \theta)y) \ \leq \ \theta f(x) + (1 - \theta)f(y) \tag{3.3}$$

This definition means that the function lies below or on the line segment that connects any two points on its graph. The convexity property is used when looking for global minimums, especially when analyzing a system's Lyapunov stability. It is also utilized in optimization, since convexity makes an optimization problem significantly easier to solve. All these properties make convexity a powerful tool for stability analysis and optimization.

SOS programming utilizes SDP, which in turn utilizes LMI's. Linear matrix inequality is defined as mathematical expressions that involve matrices and is often used in optimization and control theory. An LMI is usually on the form in (3.4), where A is a linear function of the matrix variable x and B is the value that $A(x)$ needs to be smaller than or equal to.

$$A(x) \leq B \tag{3.4}$$

Semidefinite programming is an optimization technique used in accordance with the optimization of linear objective functions subject to linear and semidefinite constraints. It is an extension of linear and quadratic programming, and the decision variables are positive definite matrices. SOS programming is related to SDP in the article the following proposition in Hovd's article[12]: *A polynomial $p(x)$ of degree 2d is an SOS if and only there exists a positive semidefinite matrix Q such that*

$$p(x) = z^T Q z \tag{3.5}$$

where z is the vector of monomials of degree ut to d, i.e.,

$$z = \left[1, x_1, x_2, ..., x_n, x_1^2, x_1x_2, ..., x_n^d\right]^T \tag{3.6}$$

3.1.1 Utilizing the S-Procedure

A procedure called S-procedure combined with the previously introduced Positivstellensatz(2.5), will utilized to find the largest estimate of the region of attraction for the system. Furthermore, several different optimization tools can be utilized to execute the actual optimization in the procedure, but the ones investigated are toolboxes for MATLAB called YALMIP and SOSTOOLS. SOSTOOLS is a toolbox specifically designed to solve optimization problems with SOS and SDP techniques.[24] YALMIP is a general tool for solving a wide range of optimization problems that was useful and is therefore investigated further in this thesis. The S-procedure is used in several steps of the SOS method, and the *solvesos()* function introduced in chapter 2.6 also utilizes the S-procedure within the function itself while solving an optimization problem.

In [12] Hovd extends the S-procedure to handle rational polynomial dynamics, which some electric components are, and could be useful in future investigations. Hovd extends the procedure to include a controller design procedure in this article. This way, no controller needs to be implemented to find the estimation of the ROA. If the controller is designed in the procedure, achieving an even larger estimate of the ROA may be possible, but this will come at the cost of a simple controller. The controller could be challenging to implement in the industry since the equation for the controller would not develop a standard equation. A standard controller is utilized to make an eventual implementation in the industry more likely.

3.2 Establishing the Set Containment Problem

A system can have infinite Lyapunov functions, and finding the Lyapunov function with the largest region of attraction is desirable. To achieve this in this thesis, the S-procedure and V-s iteration are utilized to optimize the estimation of the region of attraction. The particular Positivstellensatz equation derived in chapter 2.5 is the base of the set containment problem consisting of three sets contained inside each other as portrayed in figure 3.1. The sets are further explained below the figure.

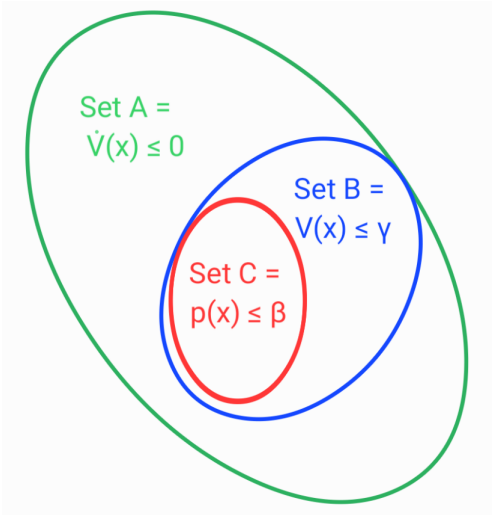


Figure 3.1: Set Containment Sets

- Set A (green): The largest set where $\dot{V}(x) \leq 0$.
- Set B (blue): The middle set is described by a function $V(x)$, which is the Lyapunov function. This Lyapunov function needs to be smaller than γ.
- Set C (red): The Innermost set is described by a function $p(x)$, which has to be smaller than β. The shape of $p(x)$ is arbitrary.

To optimize the Lyapunov function, the V-s algorithm will expand set C, which then expands set B, and then start at set C again. This will continue until the limit of set A is reached and results in the containment problem presented in equation (3.7).

$$max\ \beta,\ \gamma \quad \text{subject to} \quad Set\ C \subseteq Set\ B \subseteq Set\ A \tag{3.7}$$

3.2.1 Applying the Positivstellensatz

This set containment problem in (3.7) needs expansion to actual equations to make it solvable, and this is where the Positivstellensatz, which was introduced in 2.5, and the derived equation in (2.9) is utilized. The first step is to alter the conditions into inequalities using the this set containment equation.

First, two new sets, S_1 and S_2, and two associated polynomials, g_1 and g_2, are defined in (3.8). In the definition, g_1 and g_2 are defined as negative polynomials and will be an important detail later in the V-s iteration.

$$S_1 = \{x \in \mathbb{R}^n : g_1(x) \leq 0\} \tag{3.8a}$$
$$S_2 = \{x \in \mathbb{R}^n : g_2(x) \leq 0\} \tag{3.8b}$$

A new function $\lambda(x)$ is introduced as a helping function. $\lambda(x)$ must be a positive definite polynomial, but the value or content is irrelevant. Later, this λ will be defined as an empty polynomial, and the constants are the variables that can be changed in the optimization problem.

$$-g_1(x) + \lambda(x) \cdot g_2(x) \ > \ 0 \tag{3.9}$$

According to the spesific Positivstellensatz solution, this means that S_1 is included in S_2 if equation (3.9) holds. To further make this a solvable set of equations, the set containment problem in (3.7) is relaxed into SOS conditions since showing the positive definiteness can be quite difficult.

Using this set containment equation and the SOS relaxation on the set containment problem, the optimization problem that will produce an optimized Lyapunov candidate function is formulated in (3.10). This formulation is retrieved from Mazumder and Fuente's paper on transient stability[11].

$$max \quad \beta, \gamma \quad s.t.$$

$$V(x) - L_1(x) > 0 \tag{3.10a}$$
$$-[\nabla V(x) \cdot f(x) + L_2(x) + s_2(x)(\gamma - V(x))] > 0 \tag{3.10b}$$
$$-[(V(x) - \gamma) + s_1(x)(\beta - p(x))] > 0 \tag{3.10c}$$
$$L_1(x) = \varepsilon_1 x^T x, L_2(x) = \varepsilon_2 x^T x. \tag{3.10d}$$

From this formulation, the numbers ε_1 and ε_2 are any positive numbers, making $L_1(x)$ and $L_2(x)$ SOS by construction. $s_1(x)$ and $s_2(x)$ are defined as SOS and represents the $\lambda(x)$ in equation (3.8). $\nabla V(x) \cdot f(x)$ is the outcome of $\dot{V}(x)$. $f(x)$ is defined as $\dot{x} = f(x)$. (3.10a) is the first requirement of a Lyapunov function, that $V(x)$ must be positive definite. $L_1(x)$ is introduced as a margin to compensate for numerical errors. It also ensures that $V(x)$ cannot be zero anywhere except at the origin. Equation (3.10b) ensures that the LF decreases along the system trajectory by ensuring that $V(x) < \gamma$. $L_2(x)$ has the same task as $L_1(x)$, to compensate for numerical errors and ensuring that $\dot{V}(x) = 0$ for only $x = 0$. (3.10c) ensures that set C stays inside set B.

This formulation, however, is complex and requires a lot of knowledge on the subject to understand. Therefore, (3.10) is rewritten for easier understanding in equation (3.11), where the sets are used instead of the equations to make the set containment more prominent. Otherwise, most variables remain the same as in the original formulation. From this final formulation, it is possible to go into the V-s iteration to optimize a Lyapunov function.

$$max \quad \beta, \gamma \quad s.t.$$

$$eq1 = V(x) - L_1(x) > 0 \tag{3.11a}$$
$$eq2 = -A - L_2(x) + s_2(x) \cdot B > 0 \tag{3.11b}$$
$$eq3 = -B + s_1(x) \cdot C > 0 \tag{3.11c}$$
$$L_1(x) = \varepsilon_1 x^T x, L_2(x) = \varepsilon_2 x^T x. \tag{3.11d}$$

After this optimization problem is solved using the V-s iteration explained in the following chapter 3.3, the result will be an optimized Lyapunov function and an associated γ. In the definition of set B, Figure 3.1 states that $V(x) \leq \gamma$. Any system states that have initial conditions that satisfy this set B, and its condition will remain stable. The condition that defines the set also defines the estimate of the region of attraction, which is a reliable measure of stability in nonlinear electrical systems.

3.3 V-s Iteration

The optimization problem in equation (3.11) has several variables that needs to be determined in the optimization, whereas some of them are multiplied by each other. These terms are called bilinear, and optimization problems that contain these terms can not be solved using regular SOS optimization. This is because an SOS problem needs to be linear in the decision variables[25]. The bilinear terms are $s_2(x) \cdot \gamma$ in (3.11b) and $s_1(x) \cdot \beta$ in (3.11c).

V-s iteration is introduced and expands an initial Lyapunov functions estimation of the region of attraction to the Lyapunov function that has the largest possible estimation of the region of attraction.[11] V-s iteration can be used in many different areas, and is for example used on flight controllers.[16] A downside is that the iteration creates bilinear terms that the YALMIP toolbox can not handle directly, and additional assistance is needed. The V-s iteration is executed by following the five steps stated below, and the steps are elaborated on after.

1. **Initialization:** An estimation of the LF is calculated using linearization around the equilibrium point to have a starting LF.
2. **γ:** For the specific $V(x)$, maximise γ by solving (3.11b).
3. **β:** For the specific $V(x)$ and γ, mazimize β by solving (3.11c).
4. **V:** Update $V(x)$ by using the three first equations of (3.11), while keeping $\gamma, \beta, s_1(x)$ and $s_2(x)$ fixed.
5. **Check Convergence:** Check if the change in $p(x) - V(x)$ is significant. If this is the case, set $p(x) = V(x)$ and go to step 2. If not, terminate.

3.3.1 Step One - Initialization

The first step is to calculate and create a starting Lyapunov function to have a starting $V(x)$ for the optimization. This is executed by evaluating the system at the equilibrium point and linearizing it there.

First, calculate the equilibrium point by setting the equations equal to zero, like in (3.12), and find the values.

$$f(x) == 0 \tag{3.12}$$

$$\dot{x} = A \cdot x \tag{3.13}$$

After the equilibrium point is found, the A matrix, which is defined in (3.13), is calculated by evaluating the Jacobian of the system at this equilibrium point, as shown in equation (3.14).

$$A = \frac{\partial f}{\partial x} \mid_{x=eqpoint} \tag{3.14}$$

When all the real parts of the eigenvalues are negative, the matrix is Hurwitz. And asymptotical stability is only possible if the A matrix is Hurwitz.[14] A system that does not have a Hurwitz A matrix is not locally stable, and it would not be possible or interesting to try and increase the stability area of something that is not stable in the first place. In this thesis, it is therefore assumed that the A matrix is Hurwitz for all the systems in question. The A matrix can then be used in the Lyapunov equation, which is repeated in (3.15) to find a P matrix used in (3.16) to find the starting Lyapunov function.

$$PA + A^T P + Q = 0 \tag{3.15}$$

$$V(x) = \frac{1}{2} x^T P x \tag{3.16}$$

3.3.2 Step Two - γ Step

The γ step consists in short terms of ensuring that set B stays inside set A and is accomplished by maximizing γ by using bisection to find $s_2(x)$ while holding γ fixed. The definition of $s_2(x)$, which can be any polynomial as long as it is sum of squares, is exploited to the fullest in this step.

First, the relevant sets are defined in equation (3.17), where the content of set A is a way to portray $\dot{V}(x)$ and $f(x)$ is the equations that define the system. Set B states that the Lyapunov function needs to be smaller than γ as described in Figure 3.1.

$$A = \nabla V(x) \cdot f(x) \tag{3.17a}$$
$$B = V(x) - \gamma \tag{3.17b}$$

The two other parts are defined in equation (3.18) where $L_2(x)$ is added to compensate for numerical errors. ϵ_2 is any positive number and is, in this case, set as a very small value 10^{-6}. This makes $L_2(x)$ sum of squares by construction. $s_2(x)$ is defined as an empty polynomial containing all the combinations of x, up to a degree of $degreeMax$. The coefficients in front of each part of the polynomial remain empty until it is attempted to solve the optimization problem. v_2 contains all the monomials up to degree $degreeMax$ as set in the polynomial function. Then, the vector c_2 contains the coefficients of $s_2(x)$ and is set as the decision variables in the feasibility problem that comes later in this step.

$$L_2(x) = \epsilon_2 x^T x \tag{3.18a}$$
$$[s_2(x), c_2, v_2] = polynomial(x, degreeMax, degreeMin) \tag{3.18b}$$

Further, the first complete optimization problem is stated in (3.19), where (3.19a) and (3.19b) are the constraints, which are defined to be positive and sum of squares.

$$max \quad \gamma \quad s.t.$$

$$eq2 = -A - L_2(x) + s_2(x) \cdot B > 0 \tag{3.19a}$$
$$s_2(x) > 0 \tag{3.19b}$$

Then this is implemented into the *solvesos*() function in equation (3.21) with the constraints below, where the *sos*() function is used to transform a constraint to the correct form of sum of squares.

$$GammaConstraints = \begin{bmatrix} sos(eq2) & sos(s_2(x)) \end{bmatrix} \tag{3.20}$$

$$solution = solvesos(constraints, objective, solver, decisionvariables) \tag{3.21a}$$
$$GammaSolution = solvesos(GammaConstraints, [], mosek, c_2) \tag{3.21b}$$

This now becomes a feasibility problem since the objective remains empty because the *solvesos*() function is not used to optimize anything in this case. But to find any feasible solution by changing the constants in the c_2 vector. This is done by using bisection in the form of a while loop in MATLAB, where for each value of γ, a feasible solution is found, the γ is then increased, and a feasible solution is found again. The γ is increased until finding a solution using the *solvesos*() function is no longer possible. When it is no longer possible to find a feasible solution, the while loop is terminated, and the last γ value that had a feasible solution is retrieved, and the feasibility problem is executed one last time to find the corresponding $s_2(x)$. Then, the maximized γ and the associated $s_2(x)$ are held fixed for the rest of the steps.

3.3.3 Step Three - β Step

The β step is very similar to the γ step, where the γ is exchanged with β and $eq2$ with $eq3$. γ is kept constant at the optimized value found in the previous step. This step uses the same approach, and the optimization problem will look almost identical.

Similarly, as in the γ step, the relevant sets are defined in equation (3.22), but now it is set B and C instead. Set B is identical to the definition back in (3.17b). And set C in (3.22b) uses $p(x)$, which is defined as $p(x) = V(x)$ in the first iteration, and states that $p(x)$ needs to be smaller than β.

$$B = V(x) - \gamma \tag{3.22a}$$
$$C = p(x) - \beta \tag{3.22b}$$

In this optimization problem, $s_1(x)$ is used instead of $s_2(x)$, and the definition is stated in (3.23). $s_2(x)$ and $s_1(x)$ are defined and used in the exact same way in the optimization and feasibility problems. This means that $s_1(x)$ is also defined as an empty polynomial containing all the combinations of x, up to a degree of $degreeMax$. c_1 is also the empty vector containing the constants of $s_1(x)$ and is the decision variable in the feasibility problem later in the step.

$$[s_1(x), c_1, v_1] = polynomial(x, degreeMax, degreeMin) \tag{3.23}$$

Further, the entire second optimization problem is defined below in (3.24), where (3.24a) and (3.24b) are the constraints, which are defined as positive and sum of squares.

$$max \quad \beta \quad s.t.$$

$$-B + s_1(x) \cdot C > 0 \tag{3.24a}$$
$$s_1(x) > 0 \tag{3.24b}$$

This optimization problem is then implemented into the $solvesos()$ function in (3.26) with the constraints defined in (3.25). Similarly to the γ step, the $sos()$ function makes the content a Sum of Squares constraint.

$$BetaConstraints = \begin{bmatrix} sos(eq3) & sos(s_1(x)) \end{bmatrix} \tag{3.25}$$

$$BetaSolution = solvesos(BetaConstraints, [], mosek, c_1) \tag{3.26}$$

Similarly to the γ step, this becomes a feasibility problem because of the empty objective. β remains constant, while the content of the v_2 vector is the decision variable while the program tries to find any feasible solution. This is done using bisection as a while loop in MATLAB, where for each value of β. β is then increased after a feasible solution is found, and the feasibility problem is performed again. This while loop continues until a feasible solution is no longer possible, and the last β that gave a feasible solution is retrieved. The feasibility problem is executed again to find the corresponding $s_1(x)$, and these values are kept fixed for the rest of the steps.

3.3.4 Step Four - $V(x)$ Step

This step is quite different from the two previous steps. In this step, the Lyapunov function $V(x)$ is updated or found again. This is accomplished by first keeping γ, β, $s_1(x)$ and $s_2(x)$ constant. Secondly, $V(x)$ is defined again in the same manner as $s_1(x)$ and $s_2(x)$ in (3.27a). In this case, C_v is the vector that contains the decision variables in the optimization problem. The L_1 is defined in the same manner that L_2 was defined in the γ step. The definition of L_2 is included as well for completeness.

$$\begin{bmatrix} V(x), & C_v, & V_v \end{bmatrix} = polynomial(x, degreeMax, degreeMin) \tag{3.27a}$$

$$L_1(x) = \epsilon_1 x^T x \tag{3.27b}$$

$$L_2(x) = \epsilon_2 x^T x \tag{3.27c}$$

The new $V(x)$ defines the sets again in (3.28), impacting sets A and B. Most impact will happen in set A, where a change in $V(x)$ makes many changes in $\dot{V}(x)$ since the content of C_v is now numerous places in the whole set. Set C remains the same, and $p(x)$ equals the old $V(x)$ found in the initialization step.

$$A = \nabla V(x) \cdot f(x) \tag{3.28a}$$

$$B = V(x) - \gamma \tag{3.28b}$$

$$C = p(x) - \beta \tag{3.28c}$$

Now, the final feasibility problem is specified in equation (3.29), where this step has three constraints compared to the two constraints in the two previous steps. Correspondingly to the previous steps, all the constraints are defined as strictly positive. This final problem has an empty objective, and no function is optimized since the task is to find any feasible solution.

$$eq1 = V(x) - L_1(x) > 0 \tag{3.29a}$$

$$eq2 = -A - L_2(x) + s_2(x) \cdot B > 0 \tag{3.29b}$$

$$eq3 = -B + s_1(x) \cdot C > 0 \tag{3.29c}$$

To make the constraints able to be used in the *solvesos*() function, they need to be put into a vector and be changed into sum of squares constraints using the *sos*() function as displayed in (3.30).

$$VConstraints = \begin{bmatrix} sos(eq1) & sos(eq2) & sos(eq3) \end{bmatrix} \tag{3.30}$$

Then, the optimization problem becomes the feasibility problem in equation (3.31), with an empty objective. The program will then search for any feasible solution by changing the decision variables in C_v. Contradictory to the previous steps, the feasibility problem is only carried out once. This is because there is only one variable, $V(x)$, that needs to be found, and no bilinearity is present.

$$VSolution = solvesos(VConstraints, [], mosek, C_v) \tag{3.31}$$

This now gives a new Lyapunov function that is compared to the old one in the last step.

3.3.5 Step Five - Convergence Step

This last step checks the change in the Lyapunov function, using the equation in (3.32). If the change is insignificant, smaller than a given *tolerance* i.e. $10^{-}3$, then the iteration procedure should be terminated. If it is significant, some minor changes need to be made and then go back to step two, the γ step.

$$\Delta p(x) = p(x) - V(x) \tag{3.32}$$

The first small change to be made before going back to the γ step is setting $p(x) = V(x)$, so the next step can compare the newest Lyapunov function to the one that will be found in the next iteration. Then the variables that changes in each iteration of the while loops γ and β are reset to one.

Suppose the change in the Lyapunov function is insignificant and the iteration is terminated. In that case, this means that the Lyapunov function and the accompanying γ are optimized and set B, $V(x) \leq \gamma$ defines the estimate of the region of attraction. This meant that any system that has initial conditions that satisfy the condition in the set, will remain stable.

Chapter 4

Complementary Systems

In this chapter, the Sum of Squares method explained in the previous chapter, is applied to two complementary systems. To learn how to use the toolbox, the method was first applied to two less complex systems where many issues arose and were solved. However, there was an issue in step four(Updating $V(x)$) that was not solved due to lack of time. The first system, a time-reversed van der Pol system, was used due to a publication that had done a similar investigation, which included results underway in the coding, and the results could be reproduced. The second system, a constant power load, was chosen to proceed towards the more complex wind energy conversion system. This chapter only contains new contributions and displays the basics, but for a thorough breakdown and explanation, see chapter 5.5.

4.1 Time-Reversed Van Der Pol System

The van der Pol system is well-known in the industry and is a commonly used example in teaching. Khalil describes the Van der Pol equations as a fundamental example in nonlinear oscillation theory.[14] The fundamental example can be applied to many different fields, including chemistry, biology, physics, and electrical.[26]

During research, an article that had an example with an optimized Lyapunov function for a time-reversed van der Pol system and a good analysis and calculations along the way was found.[27] This made this system an excellent system for learning the program by checking that the answers were correct along the way. This article did not use the V-s iteration, but the Sum of Squares method was used, and many of the desired results were there. The example in the article was defined as $f(x)$ in equation (4.1).

$$\dot{x}_1 = -x_2 \tag{4.1a}$$
$$\dot{x}_2 = x_1 + (x_1^2 - 1)x_2 \tag{4.1b}$$

4.1.1 V-s Iteration

Applying the SOS method described in chapter 3.3 to try and optimize a Lyapunov function for this simple system.

Step One - Initialization

First, the equilibrium point is found by setting both the equations that describe the system equal to zero in equation (4.2). This is solved and an equilibrium point is found in $x_1 = 0$ and $x_2 = 0$.

$$\dot{x}_1 = -x_2 == 0 \tag{4.2a}$$
$$\dot{x}_2 = x_1 + (x_1^2 - 1)x_2 == 0 \tag{4.2b}$$

Then the Jacobian of the system is found, and it is evaluated at the equilibrium point in (4.3).

$$A = \frac{\partial f}{\partial x} \mid_{x=0} = \begin{bmatrix} 0 & -1 \\ 2x_1x_2 + 1 & x_1^2 - 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} \tag{4.3}$$

The A-matrix is assumed Hurwitz and inserted into the Lyapunov equation that was described in equation (3.15) in chapter 3.3 to find a P-matrix, which is described in (4.4).

$$P = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1 \end{bmatrix} \tag{4.4}$$

This P-matrix is then implemented in standard Lyapunov function (4.5), and the initialization is complete and ready to be optimized. Because the example does not multiply the Lyapunov function with $\frac{1}{2}$, it is not done here either for comparison of results.

$$V(x) = x^T P x = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1.5x_1^2 - x_1x_2 + x_2^2 \tag{4.5}$$

The following code is included for completeness and a better understanding.

```
%Definitions
syms x1 x2   %defining the variables
x = [x1; x2];

f1= -x2;                  %Defining the system
f2= x1 + (x1^2 - 1)*x2;

f11= -x2 == 0;  %setting up equations to solve for equilibrium point
f22= x1 + (x1^2 - 1)*x2 == 0;

F= [f1; f2];
f0 = [f11, f22];

equilibriumPoint = solve(f0, x);   %finding the equilibrium point
jac = jacobian(F,x);

x1bar = double(equilibriumPoint.x1);  %Making the eq-point double
x2bar = double(equilibriumPoint.x2);   %Values instead of syms values
eq_var = [x1bar, x2bar];
A = double(subs(jac,x,eq_var)); % A matrix - assuming Hurwitz

Q = eye(size(A)); %Identity matrix as Q
P = lyap(A', Q);  %Finding the P matrix
```

Listing 4.1: Time-Reversed Van Der Pol - Linearization

Step Two - γ Step

In this step, the set containment is used to optimize or maximize γ by using equation (3.11b). Several functions from the YALMIP toolbox and bisection are utilized in the following code to accomplish this.

```
ops=sdpsettings('solver', 'mosek'); %choosing to use MOSEK as the
    solver

sdpvar x1 x2   %defining the variables in sdpvariables for the
    optimization part
x = [x1; x2];

f1= -x2;   %defining the system again, with the right type of variables
f2= x1 + (x1^2 - 1)*x2;
F= [f1; f2];

V = x' * P * x; %defining the LF with the P matrix found in the first
    step

eps= 1E-6;   %very small number
L2 = eps*(x'*x);   %used to compensate for numerical errors
[s2,c2,v2]= polynomial(x,6,1);   %defining an empty polynomial
gamma = 1; %starting value for gamma

gradV = jacobian(V,x);
A = gradV*F;  %defining the sets
B = V-gamma;
eq2 = -A - L2 + s2*B;  %defining the equation

GammaConstrains= [sos(eq2), sos(s2)]   %making a vector with SOS
    constraints
GammaSolution = solvesos(GammaConstrains, [], ops,c2);  %solving a
    feasibility problem with c2 as a decision variable to have a
    starting value in the loop

while GammaSolution.problem ==0  %checking that the problem is feasible
    gamma = gamma +0.1   %gamma step
    A = gradV*F;  %defining sets and equation again in the loop
    B = V-gamma;  %to include the updated gamma value
    [s2,c2,v2]= polynomial(x,6,1);
    eq2 = -A -L2  + s2*B;
    GammaConstraints= [sos(eq2), sos(s2)];  %defining constraints with
    updated equations
    GammaSolution=solvesos(GammaConstraints,[],ops,c2); %solving
    feasibility problem
end
gamma=gamma-0.1   %go back and find the last gamma and the connected s2
    (x)
A = gradV*F;
B = V-gamma;
[s2,c2,v2]= polynomial(x,6,1);
eq2 = -A -L2  + s2*B;
GammaConstraints= [sos(eq2), sos(s2)];
GammaSolution=solvesos(GammaConstraints,[],ops,c2);
```

Listing 4.2: Time-Reversed Van Der Pol - γ Step

The objective remains empty, because the solvesos() function is not used to optimize anything in this case. However, to find any feasible solution by changing the numbers in the c_2 vector. This is

done by using bisection in form of a while loop in MATLAB, where for each value of γ a feasible solution is found, the γ is then increased and a feasible solution is found again. The γ is increased until finding a solution using the solvesos() function is no longer possible. In this example, γ is increased to 2.3, which is the optimized value in the first iteration.

Step Three - β Step

The β step is very similar to the γ step, where the γ is exchanged with β and $eq2$ with $eq3$. γ is kept constant at the optimized value that is found in the previous step. The same approach is used in the following code. The optimized value for β becomes 2.3, the same as γ.

```
beta = 1;  %starting value
[s1,c1,v1]= polynomial(x,2,1);  %empty polynomial with degree of two
p = V;  %setting this as the starting point for the function p
%note the difference between p(function) and P(matrix form Lyapunov
    equation)
B = V-gamma;  %defining the sets
C = p- beta;

eq3 = -B + s1*C;  %defining the equation

BetaConstraints= [sos(eq3), sos(s1)];  %making a vector with SOS
    constraints
BetaSolution = solvesos(BetaConstraints,[],ops,c1); %solving
    feasibility problem with c1 as the decision variable

while BetaSolution.problem == 0  %checking feasibility
    beta = beta +0.01  %beta step
    B = V-gamma;  %define everything again in the loop
    C = p- beta;  %to include updated beta
    [s1,c1,v1]= polynomial(x,2,1);
    eq3 = -B + s1*C ;
    BetaConstraints= [sos(eq3), sos(s1)];
    BetaSolution=solvesos(BetaConstraints,[],ops,c1);
end
beta= beta -0.01   %finding the last feasible beta and the connected s1
    (x)
B = V-gamma;
C= p- beta;
[s1,c1,v1]= polynomial(x,2,1);
eq3 = -B + s1*C ;
BetaConstraints= [sos(eq3), sos(s1)];
BetaSolution= solvesos(BetaConstraints,[],ops,c1);
```

Listing 4.3: Time-Reversed Van Der Pol - β Step

Step Four - $V(x)$ Step

In this step the Lyapunov function $V(x)$ is updated or established again. This is accomplished by first keeping γ, β, $s_1(x)$ and $s_2(x)$ constant. Secondly, $V(x)$ is defined again in the same manner as $s_1(x)$ and $s_2(x)$ in the following code.

```
%Updating V(x)
L1 = eps*(x'*x);  %nemrical error compensator

[Vnew,C1,V1] = polynomial(x,4,1); %defining an empty V(x) as Vnew
gradV = jacobian(Vnew,x);    %finding the jacobian of the new V(x)
A = gradV*F;      %defining the sets with the new V(x)
B = Vnew - gamma;
C= p - beta;
eq1 = Vnew - L1 ;   %defining the sets
eq2 = -A - L2 + s2*B;
eq3= -B + s1*C;
VConstraints=[sos(eq1), sos(eq2), sos(eq3)]; %Setting the three
    equations as SOS constraints
VSolution=solvesos(VConstraints,[],ops,C1); %solving the feasibility
    problem with the vector C1 as the decision variable
```

Listing 4.4: Time-Reversed Van Der Pol - Updating $V(x)$

This is where an issue arose, and there was not enough time to solve it. Unfortunately, this same issue arose for the WECS system and is discussed further in chapter 5.6.

Step Five - Checking for Convergence

Because of issues with the previous step, this step was not attempted, and the following code is a pseudocode and is not exact. Nevertheless, if the difference is significant, $p(x)$ should be changed to $p(x) = V(x)$, and the iteration should go back to step two(γ-step) and repeat. Before a possible return to step two, remember to reset γ and β. If the change is small enough, then the Lyapunov function, $V(x)$, converges, and the iteration should stop, and an optimized function is found.

```
%Checking convergence
difference = p - Vnew ;
tolerance = ???

if difference < tolerance
    jump out of loop
else
    gamma =1;  %reset gamma and beta
    beta = 1;
    go back to step 2 and go again
end
```

Listing 4.5: Time-Reversed Van Der Pol - Checking for Convergence

4.2 Constant Power Load System

The next step to implementing the method for the complex electrical system is to implement it on a simple system that still has many similarities. This system has a generator, and the speed is the variable in question. The constant power load(CPL) is presented in equation (4.6).

$$J\frac{d\omega}{dt} = -D\omega + T_m - \frac{P_{const}}{\omega} \tag{4.6}$$

4.2.1 Incremental Model

Similar to the original complex electrical system(WECS), an incremental model is required for this system. See chapter 5 for more detailed information and chapter 2.1 for theory on equilibrium points. This is also because a speed of zero is not a realistic stable operating point, since this means that the generator is turned off and will be stable by definition.

The desirable operating point is the equilibrium point, found in the same manner as it will be in chapter 5.4. Defining the equation at zero in (4.7), where $\bar{\omega}$ is defined as the speed zero.

$$0 = D\bar{\omega} + T_m - \frac{P_{const}}{\bar{\omega}} \tag{4.7}$$

Now, this zero equation is subtracted from the original equation to find the incremental model in equation (4.8).

$$\frac{d\tilde{\omega}}{dt} = -\frac{D}{J}\tilde{\omega} - \frac{P_{const}}{J}(\frac{1}{\tilde{\omega}+\bar{\omega}} - \frac{1}{\bar{\omega}}) \tag{4.8}$$

This equation is simplified to compensate for the rational dynamics in the last part. The $\tilde{\omega}+\bar{\omega}$ is replaced with $\tilde{\omega}$, which results in equation (4.9).

$$\frac{d\tilde{\omega}}{dt} = -\frac{D}{J}\tilde{\omega} - \frac{P_{const}}{J}(\frac{1}{\tilde{\omega}} - \frac{1}{\bar{\omega}}) \tag{4.9}$$

Now, the incremental model is ready for use in the V-s iteration.

4.2.2 V-s Iteration

Applying the method described in chapter 3.3 to try and optimize a Lyapunov function for this simple electrical system.

Step One - Initialization

The first step is to calculate and create a starting Lyapunov function, which is accomplished by evaluating the system at the equilibrium point and linearizing it.

First, the equilibrium point is calculated by setting the equations equal to zero, like in (4.10) and calculating the values. Two equations and two unknowns. Solving this gives an equilibrium point when $\bar{\omega} = 200$.

$$J\frac{\omega}{dt} = -D\bar{\omega} + T_m - \frac{P_{const}}{\bar{\omega}} == 0 \tag{4.10}$$

Then, the Jacobian is defined by the partial derivative of the incremental model in equation (4.9).

$$Jacobian = \frac{\partial f(\tilde{\omega})}{\partial \tilde{\omega}} = P\frac{1}{(\tilde{\omega}+200)^2} - \frac{1}{2} \tag{4.11}$$

The next thing to do is to evaluate the Jacobian at this equilibrium point to find the previously mentioned A-matrix.

$$A = \frac{\partial f(\tilde{\omega})}{\partial \tilde{\omega}} \mid_{\tilde{\omega}=200} = -\frac{1}{2} + 20000 \frac{1}{(\tilde{\omega}+200)^2} = -0.3750 \tag{4.12}$$

Then, this A is assumed Hurwitz and implemented in the Lyapunov equation defined in equation (3.15). This results in $P = 1.33$ and is implemented in the standard equation for a quadratic Lyapunov function in (4.13) and a Lyapunov candidate is ready for use in the optimization steps.

$$V(\tilde{\omega}) = \frac{1}{2}\tilde{\omega}^T P \tilde{\omega} = 0.6666\tilde{\omega}^2 \tag{4.13}$$

The following code is included for completeness and a better understanding of all the steps.

```
%definitions
syms omega omegabar omegatilde real

Tm = 200;  %values based of WECS system in chapter five
omegaref = 200;
P = 20000;
J = 7.856;
D= 0.5;

eq = -D*omegabar + Tm - P/omegabar == 0; %equation to find equlibrium
    point

equilibrium = solve(eq,omegabar); %finding the equilibrium point
equilibrium = max(equilibrium);  %line 12 gives two equilibrium points,
     and the largest is chosen

f= -D*omegatilde - P*(1/(omegatilde+omegabar)-1/omegabar); %system
    equation
jac = jacobian(f,omegatilde); %finding the jacobian
A = double( subs(jac, omegatilde, omegabar) ); %making the A matrix -
    assuming that A is Hurwitz

Q = eye(size(A)); %Identity matrix as Q
P = lyap(A', Q);  %Finding the P matrix
```

Listing 4.6: Constant Power Load - Initialization

Step Two - γ Step

The set containment is used to maximize γ, by using (3.11b) the same way as for the time-reversed van der Pol system. For a better understanding, the sets are defined in (4.14), and the rest of the step is explained in the following code.

$$A = \nabla V(\tilde{\omega}) f(\tilde{\omega}) = 2.6666\tilde{\omega} \cdot (-\frac{D}{J}\tilde{\omega} - \frac{P}{J}(\frac{1}{\tilde{\omega}} - \frac{1}{\bar{\omega}})) = \frac{2.666}{J} \cdot (-D + \frac{P}{\bar{\omega}}\tilde{\omega} - D\tilde{\omega}^2) \tag{4.14a}$$

$$B = V(\tilde{\omega}) - \gamma = 1.33333\tilde{\omega}^2 - \gamma \tag{4.14b}$$

```
%definitions
ops=sdpsettings('solver', 'mosek'); %choosing MOSEK as solver

sdpvar omegatilde  %need the variable to be sdp instead of syms

f = -(D/J)*omegatilde-(Pmax/J)*(1/omegatilde - 1/omegabar); %defining
    system equation

V= (1/2)* omegatilde' * P * omegatilde; %defining LF by using the P
    found in step one.

eps = 1E-6;
L2= eps*(omegatilde'*omegatilde); %numerical error compensator
[s2,c2,v2]= polynomial(omegatilde,2,1); %empty polynom
gamma=1; %starting gamma

gradV = jacobian(V,omegatilde);
A = gradV*f; %defining the sets
B = V-gamma;
eq2 = -A - L2 + s2*B; %defining the equation

GammaConstraints= [sos(eq2), sos(s2)];    %defining SOS constraints
GammaSolution=solvesos(GammaConstraints,[],ops,c2); %finding starting
    value for GammaSolution

while GammaSolution.problem == 0  %checking feasibility
    gamma = gamma * 10  %gamma step
    [s2,c2,v2]= polynomial(omegatilde,2,1); %defining everything inside
    A = gradV*f;                            %loop to include updated
    gamma
    B = V-gamma;
    eq2 = -A -L2  + s2*B;
    GammaConstraints= [sos(eq2), sos(s2)]; %Making SOS constraints
    GammaSolution=solvesos(GammaConstraints,[],ops,c2); %solving
    feasibility problem
end
gamma=gamma / 10  %finding the last feasible gamma and connected s2(x)
[s2,c2,v2]= polynomial(omegatilde,2,1);
A = gradV*f1;
B = V-gamma;
eq2 = -A -L2  + s2*B;
GammaConstraints= [sos(eq2), sos(s2)];
GammaSolution=solvesos(GammaConstraints,[],ops,c2);
```

Listing 4.7: Constant Power Load - γ Step

With this code, γ continues to rise to $\gamma = 10^{40}$, where the MOSEK toolbox gets an error: *"MSK RES ERR HUGE AIJ (A numerically huge value is specified for an element in A.)"* which makes γ go back to 10^{39} for a feasible result. Newertheless at $\gamma = 10^{13}$ the solver says: *"Although the solver indicates no problems, the residuals in the problem are really bad. My guess: the problem is probably infeasible. Make sure to check how well your decomposition matches your polynomial (see manual). You can also try to change the option sos.model or use another SDP solver."* This indicates that the problem became infeasible and the loop should have stopped here. However, this did not happen, indicating that the code's stopping criteria are not very good for this example. If the γ value 10^{39} is used further in the β step and the optimization step, the problem immediately becomes infeasible because of the too high value of γ. This also indicates that the while loop should have stopped before this point, and that the loop needs to be created with testing and failing instead of the standard method. This is a problem for further research.

The while loop was stopped manually by changing the stopping criteria to the last γ that gave a feasible solution. Furthermore the following while loop replaced the original loop in the previous code. This makes $\gamma = 10^{12}$ the optimized γ to use in the following steps.

```
while gamma < 10^12   %stopping the loop manually
    gamma = gamma * 10  %gamma step
    [s2,c2,v2]= polynomial(omegatilde,2,1); %defining everything inside
    A = gradV*f;                            %loop to include updated
    gamma
    B = V-gamma;
    eq2 = -A -L2  + s2*B;
    GammaConstraints= [sos(eq2), sos(s2)]; %Making SOS constraints
    GammaSolution=solvesos(GammaConstraints,[],ops,c2); %solving
    feasibility problem
end
```

Constant Power Load - Alternative

Step Three - β Step

The β step is very similar to the γ step, where the γ is exchanged with β and $lig2$ with $lig3$. γ is kept constant at the optimized value found in the previous step. The same approach is utilized in the following code.

```
%Definitions
beta = 1;
[s1,c1,v1]= polynomial(x,2,1);
p = V; %setting p equal to V as a starting point
B = V-gamma;
C = p- beta;
eq3 = -B + s1*C;

BetaConstraints= [sos(eq3), sos(s1)];  %making the SOS constraints
BetaSolution= solvesos(BetaConstraints,[],ops,c1); %making a statring
    value

while BetaSolution.problem ==0  %checking feasibility
    beta = beta *10  %beta step
    B = V-gamma;
    C = p- beta;
    [s1,c1,v1]= polynomial(x,2,1);
    eq3 = -B + s1*C ;
    BetaConstraints= [sos(eq3), sos(s1)];
    BetaSolution=solvesos(BetaConstraints,[],ops,c1); %feasibility
    problem
end
beta= beta / 10   %retrieving the last feasible beta and s1
B = V-gamma;
C= p- beta;
[s1,c1,v1]= polynomial(x,2,1);
eq3 = -B + s1*C ;
BetaConstraints= [sos(eq3), sos(s1)];
BetaSolution= solvesos(BetaConstraints,[],ops,c1);
```

Listing 4.8: Constant Power Load - β Step

With the manually procured γ from the last step, the while loop stops at $\beta = 10^8$ with a problem status: *Unknown*. Moreover, the last feasible solution 10^7 is retrieved. However, the same type

of issue as the γ step arose at $\beta = 4$, and the loop should end here. This premature stop could be because of the large γ, which is very large compared to the β. If the γ was smaller, the β could be larger. It is strange with this large difference between the two variables compared to the reversed-time van der Pol system where both γ and β were equal to 2.3. The while loop in the following code replaced the previous one to stop the β value manually, and $\beta = 4$ was used further in the steps.

```
while beta < 4  %stopping beta manually
    beta = beta +1
    B = V-gamma;
    C = p- beta;
    [s1,c1,v1]= polynomial(x,2,1);
    eq3 = -B + s1*C ;
    BetaConstraints= [sos(eq3), sos(s1)];
    BetaSolution=solvesos(BetaConstraints,[],ops,c1);
end
```

Constant Power Load - Alternative

Step Four - $V(x)$ Step

The Lyapunov function $V(x)$ is updated or recreated in this step. This is accomplished by first keeping γ, β, $s_1(x)$ and $s_2(x)$ constant. Secondly, $V(x)$ is defined again in the same manner as $s_1(x)$ and $s_2(x)$ in steps two and three.

```
%Updating V(x)
L1 = eps*(x'*x);  %numerical error compensator

[Vnew,C1,V1] = polynomial(x,4,1);  %making and empty polynomial
gradV = jacobian(Vnew,x);   %new gradV with the new V
A = gradV*f;       %defining the sets with Vnew
B = Vnew - gamma;
C= p - beta;
eq1 = Vnew - L1 ;     %defining the equations
eq2 = -A - L2 + s2*B;
eq3= -B + s1*C;
VConstraints=[sos(eq1), sos(eq2), sos(eq3)]; %defining the SOS
    constraints
Vsolution=solvesos(VConstraints,[],ops,C1);  %Solving the feasibility
    problem with C1 as decision variable
```

Listing 4.9: Constant Power Load - Updating $V(x)$

This is where the issue arises, and the feasibility problem becomes infeasible no matter what is done. More information can be found in the discussion in chapter 5.6.

Step Five - Checking for Convergence

The same issue occurred in the van der Pol example. This step was not attempted due to the stop in the previous step, and the code here is how it is imagined to be. Ideally, everything should be put in a large while loop where the condition for the if-sentence is the condition for the loop. However, it should work like this: If $\Delta p(\tilde{\omega})$ is significant, the p-function is changed to $p(\tilde{\omega}) = V(\tilde{\omega})$ and the iteration goes back to step two(γ-step) and repeat. If the change is small enough, then

the Lyapunov function, $V(\tilde{\omega})$, converges and the iteration should stop and an optimized function is found.

```
%Checking convergence
difference = p - Vnew ;
tolerance = ???

if difference < tolerance
    jump out of loop
else
    gamma =1;  %reset gamma and beta
    beta = 1;
    go back to step 2 and go again
end
```

Listing 4.10: Constant Power Load - Checking for Convergence

Chapter 5

Wind Energy Conversion System

This chapter first introduces the wind energy conversion system, which is the object of interest in this thesis. Several things are applied and implemented to prepare the system for the application of the Sum of Squares method. This part is reproduced from the associated specialization project[13], but some improvements were made, and some errors from the project were corrected. The application after this part are all new contributions. Then, the Sum of Squares method is applied and tested on the system, and the wisdom from the complementary systems is implemented on a more complex system. Lastly, the results and issues are discussed, and a alternative software is suggested.

5.1 System Information

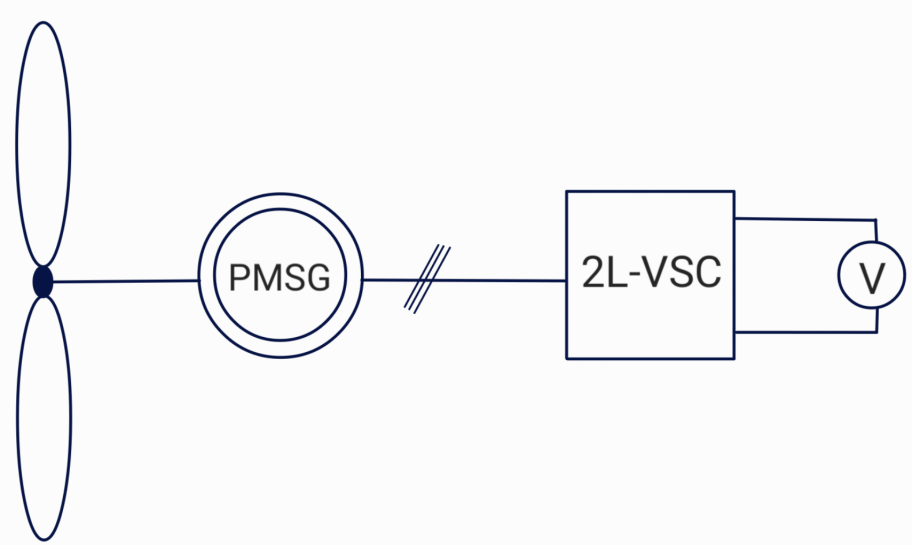


Figure 5.1: Wind Energy Conversion System

Figure 5.1 shows a model of a standard wind energy conversion system(WECS) and is the system that will be investigated in this thesis. The system consists of a wind turbine, a permanent magnet synchronous generator(PMSG), a back-to-back two-level converter(2L-VSC), and a voltage source. The voltage source represents the grid that the system is connected to. The WECS is presented

using park transformation with d- and q-axes in the whole thesis. This is utilized to simplify the complicated three-phase presentation.

The system is equipped with a PMSG, which differs from standard generators in the capacity that instead of the rotor having direct current through coils that create the magnetic field, it has magnets mounted on the outside of the rotor. These magnets have a natural magnetic field, and no outside current is needed. This is what defines a permanent magnet generator. The permanent magnet generator is superior to a typical induction generator regarding performance, torque density, and weight [28].

The WECS is inspired by both [29] and [5], and the system equation derivation is found here. This system is represented by the equations in (5.1).

$$\dot{\Psi}_d = L\dot{i}_d = -ri_d + Li_q\frac{P}{2}\omega_m - u_1V_c \tag{5.1a}$$

$$\dot{\Psi}_q = L\dot{i}_q = -ri_q - Li_d\frac{P}{2}\omega_m + \phi\frac{P}{2}\omega_m - u_2V_c \tag{5.1b}$$

$$\dot{\rho} = J\dot{\omega}_m = T_m - \frac{3}{2}\frac{P}{2}\phi i_q + d(\omega^{\text{ref}} - \omega_m) \tag{5.1c}$$

$$C\dot{V}_c = u_1i_d + u_2i_q - u_3i_d^G - u_4i_q^G - GV_c \tag{5.1d}$$

$$L_G\dot{i}_d^G = -r_Gi_d^G + \omega_GL_Gi_q^G + u_3V_c - V_d^G \tag{5.1e}$$

$$L_G\dot{i}_q^G = -r_Gi_q^G - \omega_GL_Gi_d^G + u_4V_c - V_q^G \tag{5.1f}$$

In the above equations, Ψ is the flux linkage in the d and q axes. L is the inductance in the stator windings, i_d and i_q are the currents in the respective axes. r is the resistance of the generator, while P represents the number of magnetic pole pairs in the generator. The rotor's angular velocity is represented by ω_m, and ω^{ref} is the reference for the angular speed. The electromagnetic damping effect from the damper windings is presented by d, ϕ represents the magnet's magnetic flux, while V_c is the voltage source on the DC-side which is a simplification of the grid. The u variables can be interpreted as the duty cycles for the 2L-VSC, where u_1 and u_2 are for the d, and q duty cycles and u_3 and u_4 are duty cycles for the grid side converter. ρ is the angular momentum, J is the moment of inertia for the generator. T_m is the mechanical torque while C is the capacitance and G is the conductance of the capacitor inside the 2L-VSC. Variables with the upper letter G are with reference to the grid, as i_d^G is the d-axis current from the grid side converter and r^G is the grid resistance.

Further equation (5.1c) is multiplied by a factor $\frac{2}{3}$ in equation (5.2) to make the system skew-symmetric. Where $\rho^* = \frac{2}{3}\rho$, $J^* = \frac{2}{3}J$, $T_m^* = \frac{2}{3}T_m$ and $d^* = \frac{2}{3}d$. Further ρ, J, T_m, and d are used for simpler notation.

$$\dot{\rho}^* = J^*\dot{\omega}_m = T_m^* - \frac{P}{2}\phi i_q + d^*(\omega^{\text{ref}} - \omega_m) \tag{5.2}$$

The parameters for the wind energy conversion system are listed in table 5.1 and are based on the system investigated in [29]. Note that the value for i_q^{ref}, was found in the equilibrium point calculation later in the thesis.

Item	Symbol	Nominal Value
Synchronous resistance	r	0.3676 [Ω]
Synchronous inductance	L	3.55 [mH]
Inertia	J	7.856 $[kgm^2]$
Poles	P	28
Permanent magnet flux	ϕ	0.2867 $[Wb]$
Damping	d	0.5 $[\frac{Nm}{rad/s}]$
Direct current	i_d^{ref}	0 [A]
Quadrature current	i_q^{ref}	49.82 [A]
Angular velocity	ω_m^{ref}	200 [rpm]

Table 5.1: Nominal Values For System Parameters

5.2 Leader Follower Philosophy

It is very complex and challenging to perform large mathematical calculations on this system. The leader-follower philosophy is implemented to apply a cascaded system architecture to make this tractable. Subsystems are interconnected hierarchically, where one system is the leader, and the other needs to adjust its behavior to track the leader's signal. In this case, the wind turbine and PMSG are decoupled from the rest of the system and defined as the leader. The new leader's output will be the input for the rest of the system. This way, the PMSG will be independent of the system, and the follower has to act according to how the PMSG is operated. This makes the system easier to work with. The leader is presented in equation (5.3), but simplifications need to be made in the last terms in (5.3a) and (5.3b) to decouple it from the rest of the system.

$$\dot{\Psi}_d = L\dot{i}_d = -ri_d + Li_q\frac{P}{2}\omega_m - u_1V_c \tag{5.3a}$$

$$\dot{\Psi}_q = L\dot{i}_q = -ri_q - Li_d\frac{P}{2}\omega_m + \phi\frac{P}{2}\omega_m - u_2V_c \tag{5.3b}$$

$$\dot{\rho} = J\dot{\omega}_m = T_m - \frac{P}{2}\phi i_q + d(\omega^{\text{ref}} - \omega_m) \tag{5.3c}$$

Simplifications for the controller variable u_1 and u_2 are performed in equation (5.4). This is done to make E_1 and E_2 the leader's output and input for the follower.

$$u_1 = \frac{E_1}{V_c}, \qquad u_2 = \frac{E_2}{V_c} \tag{5.4}$$

Implemented into (5.3), this results in equation (5.5) that describes the system. This leader will be referred to as the system for the remainder of the thesis.

$$\dot{\Psi}_d = L\dot{i}_d = -ri_d + Li_q\frac{P}{2}\omega_m - E_1 \tag{5.5a}$$

$$\dot{\Psi}_q = L\dot{i}_q = -ri_q - Li_d\frac{P}{2}\omega_m + \phi\frac{P}{2}\omega_m - E_2 \tag{5.5b}$$

$$\dot{\rho} = J\dot{\omega}_m = T_m - \frac{P}{2}\phi i_q + d(\omega^{\text{ref}} - \omega_m) \tag{5.5c}$$

5.3 Implementing PI Passivity Inspired Control

Stability is often studied when the system has a closed loop, meaning it has feedback that compares the system's output with its desired output. This thesis will utilize a controller inspired by the proportional-integral(PI) passivity-based controller(PBC).

Today's industry uses PI controllers as a standard because of their straightforward design and implementation. This controller has two types of correction: a proportional term that multiplies the input with a gain constant, K_p, and an integral term that integrates the input and multiplies it with a different gain constant, K_I. A PI controller uses the error between a signal and its reference as input, and the correction term is the output. The controller's output is applied to the system to reduce the system's error. This controller is represented by $u(t)$ in equation (5.6), where u is the output, K_p and K_I are adjustable positive gain constants. Meanwhile y_p and y_I are the errors between the states, making them the controller input.

$$u(t) = K_p y_p(t) - K_I \int y_I(t)dt \tag{5.6}$$

Passivity-based controllers exploit the concept of passivity and energy storage in passive systems. The concept of passivity involves energy dissipation and that a system, over time, will dissipate or consume energy, making it bounded. Passivity uses energy balance, or physical systems with passive components can not produce energy but transform it from mechanical energy to electricity with some losses. PBC minimizes these losses. The stored energy, i.e., the energy delivered to the power grid, equals the difference between supplied and dissipated energy. The PBC enforces passivity in the closed-loop system by modifying this energy balance via control. Such a system can be classified as passive, and stability can be guaranteed.

Energy control is implemented by controlling the currents of the generator since the energy is the product of the current and the voltage. In this thesis, one of these currents, i_q, is changed to control the rotation speed of the generator, ω_m in the integrator term. This allows the PMSG to follow the speed of the wind, which in turn generates maximum torque. For this specific WECS system, a PI-PBC-inspired controller is chosen. This controller combines a standard PI controller with passivity-based control. This controller is especially efficient when working with nonlinear systems since the integral part of PI compensates for nonlinearities. The PBC-inspired part regulates the system's energy and thus maintains stability more easily. Figure 5.2 shows how the controller is implemented in the system.

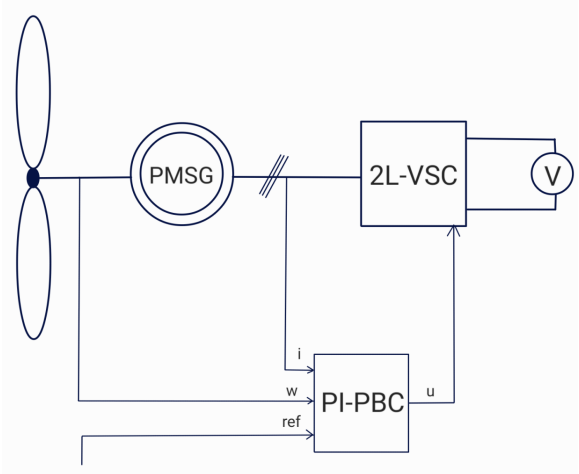


Figure 5.2: System with Implemented Control

To implement this PI-PBC into the system, the definition of the PI controller in equation (5.6) is used. First, the input terms are defined in (5.7).

$$y_p = \begin{bmatrix} i_d \\ i_q \end{bmatrix} \qquad y_I = \begin{bmatrix} i_d \\ \omega_m \end{bmatrix} \tag{5.7}$$

Furthermore, y and $\dot{x}_c$ are defined in (5.8) and (5.9), respectively, where the reference values are subtracted from the input values from the system. They are then implemented in (5.6) in equation (5.10).

$$y = (y_p - y_p^{\text{ref}}) = \begin{bmatrix} i_d - i_d^{\text{ref}} \\ i_q - i_q^{\text{ref}} \end{bmatrix} \tag{5.8}$$

$$\dot{x}_c = (y_I - y_I^{\text{ref}}) = \begin{bmatrix} i_d - i_d^{\text{ref}} \\ \omega_m - \omega^{\text{ref}} \end{bmatrix} \tag{5.9}$$

$$u = K_p y - K_I x_c \tag{5.10}$$

K_p and K_I are defined as 2×2 matrices that only has diagonal elements in (5.11).

$$K_p = \begin{bmatrix} K1 & 0 \\ 0 & K2 \end{bmatrix}, \quad K_I = \begin{bmatrix} K1 & 0 \\ 0 & K2 \end{bmatrix} \tag{5.11}$$

This results in the expanded version of the controller shown in (5.12)

$$u_{(1)} = K1_p \cdot y_{(1)} - K1_I \cdot x_{c(1)} \tag{5.12a}$$
$$u_{(2)} = K2_p \cdot y_{(2)} - K2_I \cdot x_{c(2)} \tag{5.12b}$$

Inserting this into the system, resulting in a new model for the system in equation (5.13). Here, $E1$ and $E2$ are modified since these are the control variables, as $E_1 = u_{(1)}$ and $E_2 = u_{(2)}$.

$$\dot{\Psi}_d = -ri_d + Li_q \frac{P}{2}\omega_m - (K1_p \cdot (i_d - i_d^{\text{ref}}) - K1_I \cdot x_{c(1)}) \tag{5.13a}$$
$$\dot{\Psi}_q = -ri_q - Li_d \frac{P}{2}\omega_m + \phi \frac{P}{2}\omega_m - (K2_p \cdot (i_q - i_q^{\text{ref}}) - K2_I \cdot x_{c(2)}) \tag{5.13b}$$
$$\dot{\rho} = T_m - \frac{P}{2}\phi i_q + d(\omega^{\text{ref}} - \omega_m) \tag{5.13c}$$
$$\dot{x}_{c(1)} = -(i_d - i_d^{\text{ref}}) \tag{5.13d}$$
$$\dot{x}_{c(2)} = -(\omega_m - \omega_m^{\text{ref}}) \tag{5.13e}$$

5.4 Incremental Model

$$\dot{\tilde{\Psi}}_d = -r\tilde{i}_d + L\frac{P}{2}(\tilde{i}_q\tilde{\omega}_m + \tilde{i}_q\bar{\omega}_m + \bar{i}_q\tilde{\omega}_m) - (K1_p\tilde{i}_d - K1_I\tilde{x}_{c(1)}) \tag{5.14a}$$

$$\dot{\tilde{\Psi}}_q = -r\tilde{i}_q - L\frac{P}{2}(\tilde{i}_d\tilde{\omega}_m + \tilde{i}_d\bar{\omega}_m + \bar{i}_d\tilde{\omega}_m) + \phi\frac{P}{2}\tilde{\omega}_m - (K2_p \cdot \tilde{i}_q - K2_I \cdot \tilde{x}_{c(2)}) \tag{5.14b}$$

$$\dot{\tilde{\rho}} = -\frac{P}{2}\phi\tilde{i}_q \tag{5.14c}$$

$$\dot{\tilde{x}}_{c(1)} = -\tilde{i}_d \tag{5.14d}$$

$$\dot{\tilde{x}}_{c(2)} = -\tilde{\omega}_m \tag{5.14e}$$

When operating with differential equations and stability, finding an equilibrium point expressed by the system is desirable. A natural equilibrium point is when everything is zero, but this solution concludes that the generator is turned off and is undesirable. A non-zero equilibrium point must be found. An incremental model is used to find this point. In this thesis, the natural equilibrium point, $\tilde{x}$, is the deviation of the actual operating point, x, and zero, which $\bar{x}$ is defined as. This results in the incremental model shown in equation (5.15).

$$\tilde{x} = x - 0 = x - \bar{x} \tag{5.15}$$

$\tilde{x} = 0$ when $x = \bar{x}$, ensuring the actual operating point is equal to 0, and Lyapunov theory can be applied to analyse stability. Lyapunov stability theory was investigated in chapter 2.3. This incremental model is applied to the system variables, as shown in (5.16).

$$\begin{bmatrix} \tilde{i}_d \\ \tilde{i}_q \\ \tilde{\omega}_m \\ \tilde{x}_{c(1)} \\ \tilde{x}_{c(2)} \end{bmatrix} = \begin{bmatrix} i_d - \bar{i}_d \\ i_q - \bar{i}_q \\ \omega_m - \bar{\omega}_m \\ x_{c(1)} - \bar{x}_{c(1)} \\ x_{c(2)} - \bar{x}_{c(2)} \end{bmatrix} \tag{5.16}$$

To understand the calculation better, the system is defined in the origin, $\bar{x}$, by equation (5.17). In this equation, all the system variables are replaced by 0, i.e., $\bar{x}$. The last term in (5.13c) disappears, since $\bar{\omega}_m = \omega^{\text{ref}}$ at the origin.

$$0 = -r\bar{i}_d + L\bar{i}_q\frac{P}{2}\bar{\omega}_m - (-K1_p(\bar{i}_d - i_d^{\text{ref}}) + K1_I \cdot \bar{x}_{c(1)}) \tag{5.17a}$$

$$0 = -r\bar{i}_q - L\bar{i}_d\frac{P}{2}\bar{\omega}_m + \phi\frac{P}{2}\bar{\omega}_m - (-K2_p \cdot (\bar{i}_q - i_q^{\text{ref}}) + K2_I \cdot \bar{x}_{c(2)}) \tag{5.17b}$$

$$0 = T_m - \frac{P}{2}\phi\bar{i}_q \tag{5.17c}$$

$$0 = -(\bar{i}_d - i_d^{\text{ref}}) \tag{5.17d}$$

$$0 = -(\bar{\omega}_m - \omega_m^{\text{ref}}) \tag{5.17e}$$

The new equilibrium point is then implemented into the system by subtracting (5.17) from (5.13), and the system at $\tilde{x}$ is presented in (5.18). Both i_q, i_d and ω_m will be replaced by $\tilde{i}_q + \bar{i}_q$, $\tilde{i}_d + \bar{i}_d$ and $\tilde{\omega}_m + \bar{\omega}_m$ respectively in the second term in both (5.18a) and (5.18b) for simplification. T_m is crossed out due to its lack of a state variable. This results in a system that describes the incremental model dynamics and will be used further in the thesis.

$$\dot{\tilde{\Psi}}_d = -r\tilde{i}_d + L\frac{P}{2}(\tilde{i}_q\tilde{\omega}_m + \bar{\tilde{i}}_q\bar{\omega}_m + \bar{i}_q\tilde{\omega}_m) - (-K1_p\tilde{i}_d + K1_I\tilde{x}_{c(1)}) \tag{5.18a}$$

$$\dot{\tilde{\Psi}}_q = -r\tilde{i}_q - L\frac{P}{2}(\tilde{i}_d\tilde{\omega}_m + \tilde{i}_d\bar{\omega}_m + \bar{i}_d\tilde{\omega}_m) + \phi\frac{P}{2}\tilde{\omega}_m - (-K2_p \cdot \tilde{i}_q + K2_I \cdot \tilde{x}_{c(2)}) \tag{5.18b}$$

$$\dot{\tilde{\rho}} = -\frac{P}{2}\phi\tilde{i}_q \tag{5.18c}$$

$$\dot{\tilde{x}}_{c(1)} = -\tilde{i}_d \tag{5.18d}$$

$$\dot{\tilde{x}}_{c(2)} = -\tilde{\omega}_m \tag{5.18e}$$

5.5 Applying the Sum of Squares Method

After the preliminary system configuration, the system is ready for application of the Sum of Squares method described in detail in chapter 3.3. The application can happen directly, but the overall optimization problem is repeated in equation (5.19) to refresh the memory. After this, the V-s iteration can be directly applied and the steps are executed in the following steps.

$$max \quad \beta, \gamma \quad s.t.$$

$$eq1 = V(x) - L_1(x) > 0 \tag{5.19a}$$

$$eq2 = -A - L_2(x) + s_2(x) \cdot B > 0 \tag{5.19b}$$

$$eq3 = -B + s_1(x) \cdot C > 0 \tag{5.19c}$$

$$L_1(x) = \varepsilon_1 x^T x, L_2(x) = \varepsilon_2 x^T x. \tag{5.19d}$$

The first thing to do is define the known constants from the previous chapter 5.1.

```
% define constants
r = 0.3676;
L = 3.55*10^(-3);
p = 28;
Tm = 200;
iqref = 49.82;
omegaref = 200;
idref = 0;
flux= 0.2867;
K1p = 70;
K1i = 100;
K2p = 70;
K2i = -100;
d= 0.5;
```

Listing 5.1: WECS - Initialization

5.5.1 Step One - Initialization

Using the method described in chapter 3.3.1, an initial Lyapunov function is created using linearizing at the equilibrium point. The original equations are used for the equilibrium calculation before introducing the incremental model. In the code, the first thing to do is to define the different

variables, and using syms which is the standard variable when the goal is to calculate equilibrium points. For easier writing of the code, $x_{c(1)}, x_{c(2)}$ is written as xc1 and xc2 respectfully. Instead of writing *idtilde* on everything, the $\tilde{x}$ is neglected for the variables in x for easier writing.

```
syms iq id omega xc1 xc2 idbar iqbar omegabar xc1bar xc2bar real

x = [id, iq, omega, xc1, xc2];
xbar = [idbar, iqbar, omegabar, xc1bar, xc2bar]
```

WECS - Initialization

Then, the equilibrium point is calculated by setting the original system equations equal to zero, $F(x) = 0$, and using the *solve*() function that was introduced in chapter 2.6. This solves five equations with five unknowns. The original variables are exchanged with bar variables for the calculations so that the result will show the bar values. After the equilibrium point is located, the values at this point are extracted and made into double values.

```
%Finding equilibrium point using original system equations
equation1 = -r*idbar + L*iqbar*(P/2)*omegabar - (K1p*(idbar-idref) -
    K1i*xc1bar) == 0;
equation2 = -r*iqbar - L*idbar*(P/2)*omegabar + flux*(P/2)*omegabar - (
    K2p*(iqbar-iqref) - K2i*xc2bar) == 0;
equation3 = Tm - (P/2) * flux * iqbar == 0;
equation4 = - (idbar - idref) == 0;
equation5 = - (omegabar - omegaref) == 0;

EquilibriumEquations = [equation1, equation2, equation3, equation4,
    equation5];

Equilibrium = solve(EquilibriumEquations, xbar);

%Extracting the equilibrium values since they are syms values after the
     equilibrium calculation
idbar = double(Equilibrium.idbar);
iqbar = double(Equilibrium.iqbar);
omegabar= double(Equilibrium.omegabar);
xc1bar = double(Equilibrium.xc1bar);
xc2bar = double(Equilibrium.xc2bar);

barValues = [idbar, iqbar, omegabar, xc1bar, xc2bar];
```

WECS - Initialization

The results are displayed below, and the values are implemented in table 5.2.

$$\bar{i}_d = i_d^{\text{ref}}, \qquad \bar{i}_q = \frac{2T_m}{P\phi}, \qquad \bar{\omega}_m = \omega_m^{\text{ref}}$$

$$\bar{x}_{c(1)} = \frac{r}{K1_I} i_d^{\text{ref}} - L\frac{T_m}{\phi K1_I}\omega_m^{\text{ref}}$$

$$\bar{x}_{c(2)} = \frac{2rT_m}{K2_I P\phi} + \frac{P}{2K1_I}\omega_m^{\text{ref}}(Li_d^{\text{ref}} - \phi) + \frac{K2_p}{K2_I}(\frac{2T_m}{P\phi} - i_q^{\text{ref}})$$

$\bar{i}_d$	$\bar{i}_q$	$\bar{\omega}_m$	$\bar{x}_{c(1)}$	$\bar{x}_{c(2)}$
0	49.82809308 ≈ 49.83	200	-4.952912452 ≈ -4.95	7.838766775 ≈ 7.84

Table 5.2: Equilibrium point

Next, the Jacobian is calculated using partial derivation and the *jacobian*() function. The system equations used here are the incremental model and evaluated at the found equilibrium point in the following code. Then, the A matrix from $\dot{x} = Ax$ is established by changing the values from *syms* to *double*. For a valid A matrix for stability analysis, it is assumed that the matrix is Hurwitz, as mentioned in chapter 3.3.1.

```
% Define your system equations in the incremental model
f1 = -r*id + L*(P/2)*(iq*omega + iq*omegabar + iqbar*omega) - (K1p*id -
     K1i * xc1);
f2 = -r*iq - L*(P/2)*(id*omega + id*omegabar + id_bar*omega) + flux*(P
    /2)*omega - (K2p*iq - K2i * xc2);
f3 = -(P/2)*flux*iq + d*(omegaref-omega);
f4 = -id;
f5 = -omega;

%System Vector
F = [f1; f2; f3; f4; f5];

jacobianMatrix = jacobian(F,x);
jacobianAtEquilibrium = subs(jacobianMatrix, x, barValues);  %is a syms
     matrix

A = double(jacobianAtEquilibrium)  %making it a double matrix
%assuming that the A matrix is Hurwitz
```

WECS - Initialization

The last thing to do in the initialization is to find the starting Lyapunov function in equation (5.21) by using the Lyapunov equation in equation (3.15). This is coded by using the *lyap*() function from the Control System Toolbox in MATLAB and setting the Q matrix as the identity matrix.

$$V = \frac{1}{2}x^T Px \tag{5.21}$$

```
%making the identity matrix
Q = eye(size(A));

P=lyap(A', Q); %finding the P matrix

V = (1/2) x' * P * x;  %making the initial Lyapunov function
```

WECS - Initialization

5.5.2 Step Two - γ Step

Step two in the V-s iteration involves maximizing γ while keeping $V(x)$ constant by using equation (5.26b). The optimization problem to solve in this step is presented in (5.22). This problem was previously stated in 3.3.2 but is repeated for completeness.

$$max \quad \gamma \quad s.t.$$

$$eq2 = -A - L_2(x) + s_2(x) \cdot B > 0 \tag{5.22a}$$

$$s_2(x) > 0 \tag{5.22b}$$

Regarding the coding, the first thing to do is redefine the variables. To use the YALMIP toolbox in MATLAB the variables must be redefined from *syms* to *sdpvar* introduced in chapter 2.6. Tee system equations must also be reformulated so the program implements these changed variables into the equations. In the following code, the *tilde* variables are still simplified in the code to just $id, iq, omega, xc1$ and $xc2$, for easier code. The code still uses the defined values that were defined before the first step for all the known values. The first line changes or chooses the solver that will be utilized in the *solvesos*() function later in this step. The chosen solver for use in the YALMIP toolbox is MOSEK, which was discussed in chapter 2.6.

```
1  % definitions
2  ops=sdpsettings('solver', 'mosek'); % Choose your solver
3
4  id=sdpvar(1,1);
5  iq=sdpvar(1,1);
6  omega=sdpvar(1,1);
7  xc1=sdpvar(1,1);
8  xc2=sdpvar(1,1);
9
10 %combine all states into a vector
11 x= [id; iq; omega; xc1; xc2];
12
13 %incremental model
14 f1 = -r*id + L*(p/2)*(iq*omega + iq*omegabar + iqbar*omega) - (K1p*id -
       K1i * xc1);
15 f2 = -r*iq - L*(p/2)*(id*omega + id*omegabar + idbar*omega) + flux*(p
      /2)*omega - (K2p*iq - K2i * xc2);
16 f3 = -(p/2)*flux*iq + d*(omegaref-omega);
17 f4 = -id;
18 f5 = -omega;
19
20 % Combine all equations into a vector
21 F = [f1; f2; f3; f4; f5];
```

Listing 5.2: WECS - γ Step

The problem mainly consists of ensuring that set B always stays inside set A, i.e., that the Lyapunov function is smaller than γ in set B, and that $\dot{V}(x)$ in set A, always stays negative for in the given set B. The sets are defined again in (5.23) for a clearer view of the relevant sets and variables. The polynomials $L_2(x)$ and $s_2(x)$ are also defined in (5.23), where ϵ_2 is any small positive number since this polynomial is there to help with numerical errors. In this problem, it is set to 10^{-6}. The content of (5.23) is also implemented in the code below, where the Lyapunov function is also defined again with *sdpvariables* in the x vector. In the following code, line 10 is included because if the original nonlinear system is used, the problem is infeasible from the start. Therefore, the linearized version is included to make the problem easier, and perhaps it is solvable. This is the case for the initialization, at least, and it is discussed later in chapter 5.6.

$$A = \dot{V}(x) = \nabla V(x) \cdot f(x) \tag{5.23a}$$
$$B = V(x) - \gamma \tag{5.23b}$$
$$L_2(x) = \epsilon_2 x^T x \tag{5.23c}$$
$$[s_2(x), c_2, v_2] = polynomial(x, 2, 2) \tag{5.23d}$$

```
V=(1/2)* x' * P * x;  %The P is the P matrix that was calculated in the
    initialization step

eps= 1E-6;
L2 = eps*(x'*x);
[s2,c2,v2] = polynomial(x,2,2);
gamma = 1;

gradV= jacobian(V,x);
A = gradV*F; %has to be negative
A = (1/2)*x' * (P*A + (P*A)')*x  %the linearized version
B = V-gamma; %has to be negative

eq2 = -A - L2 + s2*B; needs to be positive and sum of squares
```

WECS - γ Step

In this step, both γ and $s_2(x)$ are determined, but since these variables are multiplied in the expression, this makes it bilinear, and the usual way of optimizing is no longer an option. To overcome this, multiple solutions can be utilized. Bisection is one. Before initializing the bisection, a starting value fir the variable *solution* is needed and procured in the following code, where the constraints are established.

```
GammaConstraints = [ sos(eq2), sos(s2) ];  %making the constraints

GammaSolution = solvesos(GammaConstraints, [], ops, c2);
```

WECS - γ Step

The solution that will be explored in this thesis is somewhat *brute force* programming. In formulation the problem, the value of $s_2(x)$ is irrelevant; it just needs to be positive or the sum of squares. Therefore, an iterative approach is used in the following code, where γ is gradually increased. In each iteration, YALMIP(or solvesos()) is utilized to check that there exists a $s_2(x)$ that is sum of squares. if this is the case, γ is increased and so on until an $s_2(x)$ no longer exists. Then the last γ and $s_2(x)$ are used further in the procedure. It becomes a feasibility problem, where YALMIP is used to check if all the constraints hold while a while-loop is used to increase γ gradually.

```
%step 1  gamma step
while GammaSolution.problem ==0
    gamma = gamma * 10
    A = gradV*F;
    B = V-gamma;
    [s2,c2,v2]= polynomial(x,4,1);
    eq2 = -A -L2  + s2*B;
    GammaConstraints = [ sos(eq2), sos(s2) ];
    GammaSolution=solvesos(GammaConstraints, [], ops, c2);
end
gamma = gamma / 10
A = gradV*F;
B = V - gamma;
[s2,c2,v2] = polynomial(x,4,1);
eq2 = - A - L2  + s2*B;
GammaConstraints = [ sos(eq2), sos(s2) ];
GammaSolution=solvesos(GammaConstraints, [], ops, c2);
```

WECS - γ Step

5.5.3 Step Three - β Step

Use the YALMIP toolbox to maximize β, while keeping $V(x)$ and the γ found in step two constant. The procedure is very similar to the γ step, but now there is a different equation and variable that are used. The optimization problem is presented in (5.24).

$$max \quad \beta \quad s.t.$$

$$-B + s_1(x) \cdot C > 0 \tag{5.24a}$$

$$s_1(x) > 0 \tag{5.24b}$$

The contents of the optimization problem are defined in (5.25) and implemented in the code below.

$$B = V(x) - \gamma \tag{5.25a}$$

$$C = p(x) - \beta \tag{5.25b}$$

$$[s_1(x), c_1, v_1] = polynomial(x, 2, 2) \tag{5.25c}$$

```
%definitions
beta = 1;
[s1,c1,v1] = polynomial(x,2,2);

gradV= jacobian(V,x);
B = V - gamma; %has to be negative
C = p - beta;  %has to be negative

eq3 = -B + s2*C; needs to be positive and sum of squares
```

Listing 5.3: WECS - β STep

This optimization problem also encounters bilinearity in the term $s_1(x) \cdot C$ which is, in fact, $s_1(x) \cdot \beta$, and bisection is utilized in this step as well. Before initializing the bisection, a starting *solution*, and constraints in the following code need to be computed.

```
1 BetaConstraints = [ sos(eq3), sos(s1) ];  %making the constraints
2
3 BetaSolution = solvesos(BetaConstraints, [], ops, c1);
```

WECS - β Step

Then, the bisection can start as a while loop where the stopping criteria is the *BetaSolution.problem* being equal to zero. When this is not the case, the while loop stops, and the last β and $s_1(x)$ from the last feasible solution are retrieved in line 11 and 17 in the following code.

```
1  %Step 2   beta step
2  while BetaSolution.problem ==0
3      beta = beta * 10
4      B = V-gamma;
5      C = p- beta;
6      [s1,c1,v1]= polynomial(x,2,1);
7      eq3 = -B + s1*C ;
8      BetaConstraints = [sos(eq3), sos(s1)];
9      BetaSolution=solvesos(sos1,[],ops,c1);
10 end
11 beta= beta / 10
12 B = V-gamma;
13 C= p- beta;
14 [s1,c1,v1]= polynomial(x,2,1);
15 eq3 = -B + s1*C ;
16 BetaConstraints = [sos(eq3), sos(s1)];
17 BetaSolution= solvesos(sos1,[],ops,c1);
```

WECS - β Step

5.5.4 Step Four - Update $V(x)$

This step updates the Lyapunov function $V(x)$ by redefining it and solving a feasibility problem with an empty objective where the redefined $V(x)$ constants are the decision variables. The feasibility problem is defined again in (5.26) where the optimized values for $\gamma, \beta, s1$ and $s2$ are implemented. This feasibility problem has no bilineareties, and bisection in the form of a while loop is unnecessary.

$$V(x) - L_1(x) > 0 \tag{5.26a}$$
$$-A - L_2(x) + s_2(x)B > 0 \tag{5.26b}$$
$$-B + s_1(x)C > 0 \tag{5.26c}$$

Almost everything that depends on $V(x)$, both set A, and B, needs to be redefined to update the equations now that $V(x)$ is defined as an empty polynomial in (5.27a). These definitions are also implemented in the code below, but the equations are defined for clarity.

$$\begin{bmatrix} V(x), & C1, & V1 \end{bmatrix} = polynomial(x, 2, 2) \tag{5.27a}$$

$$L_1(x) = \epsilon_1 x^T x \tag{5.27b}$$

$$A = \nabla V(x) \cdot f(x) \tag{5.27c}$$

$$B = V(x) - \gamma \tag{5.27d}$$

$$C = p(x) - \beta \tag{5.27e}$$

```
%Updating V(x)
L1 = eps*(x'*x);

[V,C1,V1] = polynomial(x,4,1);
gradV = jacobian(V,x);
%defining the sets
A = gradV*F;
B = V - gamma;
C= p - beta;
```

Listing 5.4: WECS - Updating $V(x)$

Then the equations are defined again in the code, to include the latest sets in the optimization problem in the following code.

```
eq1 = V - L1 ;
eq2 = -A - L2 + s2*B;
eq3= -B + s1*C;
```

WECS - Updating $V(x)$

Lastly, the constraints are set, and the consequent code solves the feasibility problem.

```
VConstraints=[sos(eq1), sos(eq2), sos(eq3)];

VSolution = solvesos(VConstraints,[],ops,C1);
```

WECS - Updating $V(x)$

5.5.5 Step Five - Check for Convergence

The last step checks if the change in $p(x) - V(x)$ is insignificant, the optimized Lyapunov function is found, and the Lyapunov function has converged. If not, go back to step 2 and start again.

Because of the issues that was encountered in step 4, updating $V(x)$, this step was not implemented but it should look similar to the following code.

```
%Checking convergence
difference = p - V
tolerance = ???

if difference < tolerance
    jump out of loop
else
    go back to step 2 and go again
end
```

Listing 5.5: WECS - Checking for Convergence

This code is not exact, more like pseudocode and several changes need to be made. All the steps from two to five should be inside a large while loop that has the convergence check as a condition. What it would look like is presented in the following pseudocode.

```
%define all the Sets and equations
%make starting conditions for all the solvesos() functions
%define difference and tolerance

%whileloop
while difference > tolerance
    %do step two - the gamma step
    find/optimize gamma and s2

    %do step three - the beta step
    find/optimize beta and s1

    %do step four - the update V step
    Update V

    %do step five - check for convergence
    difference = p - V
    Reset gamma and beta
    Set p equal to the new V
end
```

Listing 5.6: WECS - Pseudocode for the Whole Setup

5.6 Results and Discussion

The theory and background found in the publications were not very detailed due to page limitation restrictions. This hinders information sharing and leaves many questions, especially regarding the coding.

A significant problem was encountered in step four, which updates the Lyapunov function established in the first step. More time would be needed to solve this problem; therefore, the largest while loop and the last step were not attempted. The coding for the step itself needs to be completed, and new issues can arise in this step. However, along the coding journey, many issues arose and were solved; this chapter will discuss these.

5.6.1 Smaller Issues that were Solved

The first step, initialization, went relatively fine after finding out which equations to use where. The original equations without an incremental model while finding the equilibrium point, and the

incremental model while finding and evaluating the Jacobian at the calculated equilibrium point. It also took some testing and changing the control variables to obtain an A matrix that satisfied the Hurwitz condition. Nevertheless, it was accomplished using a negative gain on the term containing the speed. This part of the controller differed from the typical controller, and the information on the gain for this part was unknown. It ended up working with a negative value, and the issue was solved. These values for all the different K_p and K_I's were not tuned due to lack of time, and the testing of values stopped as soon as an accepted A matrix was found. Further testing and tuning would be a good topic for further investigation. Finding and using the proper format for the $lyap()$ function also took some time, and it was settled on $P = lyap(A', Q);$, which got the correct result.

Utilization of the YALMIP toolbox

Step two, the γ step, introduced the YALMIP toolbox and its many new functions that cause many issues. Even downloading and implementing the toolbox and the needed extra solver was tricky. While running the *yalmiptest* file that tested that everything worked, there was an issue with the type of solver used to solve optimization problems, especially the Sum of Squares kind. The solver used was the original solver in MATLAB called LMILAB, which is a part of the Robust Control Toolbox, and the YALMIP toolbox was incompatible with this one. It was necessary to download an additional solver, and some of the options were MOSEK, GUROBI, SEDUMI, and SDPT3. MOSEK was chosen. There were many issues with implementing MOSEK into MATLAB and getting the YALMIP toolbox to choose MOSEK as a solver instead of LMILAB. The important thing here was to remember where the files were saved and the path that needed to be added for it to work. So, implementing YALMIP and MOSEK took a long time before it could even be used.

In the publications, [11] and [12], there was very little information on the variables $s_1(x)$ and $s_2(x)$, and they were referred to as positive definite polynomials[11]. It said the content was irrelevant if they were positive polynomials, i.e., SOS. This little information made it difficult to determine how to define the variables in the code. After much research and testing, the function $[p, c, v] = polynomial(x, degreeMax, degreeMin)$ was used. This creates an empty parametrized polynomial with p as the function, c as a vector containing the constants and v as a vector containing the variables. More information on this function can be found in chapter 2.6 on the YALMIP toolbox. The empty c vectors that this function created were used as decision variables in all of the $solvesos()$ functions. This function was also applied in step four, updating the Lyapunov function $V(x)$. In this step, $V(x)$ was defined once more as an empty polynomial with the $polynomial()$ function.

Before entering the while loop for the first time, an initial $solution = solvesos()$ needs to be carried out to have an initial value for the while condition. This worked okay for the two less complex systems, but an issue arose when the method reached this point in the code for the desired system. The solution for the β step solution worked okay, but the solution for the γ step resulted in an infeasible problem. This issue was further investigated, but the issue was found with set A, which contains $\dot{V}(x) = \nabla V \cdot F(x)$. A conclusion was drawn that the system was too complex when a solution was found when the $\dot{V}(x)$ was linearized. The issue was gone after the linearization, and the next steps worked perfectly. A minor issue arose in the fourth step when finding set A with a completely new $V(x)$ and linearized system functions, $F(x)$, was required. $F(x)$ was replaced with $A \cdot x$ to accommodate this.

The next issue to solve was the bilinearity in steps two and three, where γ and β were multiplied with $s_2(x)$ and $s_1(x)$, respectively. This thesis uses bisection to solve this problem, but other options could be explored. Although it could have been implemented better, our while loop is an easy way to implement the bisection. An issue that arose when using a while loop was how big

each step in the iteration should be, i.e., how much to change the increasing values, γ and β in steps two and three, respectively. In this thesis, the size of steps was decided by testing and failing, and when the last step that gave a feasible solution is retried, it is not a specific optimized value. The step length should be reduced, and a new while loop should be entered. This type of iteration should be repeated until the steps are tiny. However, this would take a lot more coding and would depend very much on the system and how significant the steps should be in the initial while loop. The step length could be an interesting subject for further investigation.

Finding a while condition that would cause the loop to stop when the problem is no longer feasible was difficult regarding the bisection and while loop. For explanation, the γ step is used as a reference. See chapter 5.5.2 for more details. Firstly, the state of the decision variable vector c_2 was used as a condition, where $any(isnan(value(c2)))$ was the condition. This condition checked if any content of c_2 was Nan, i.e., not a number. This was not an optimal condition, and after some research the content of the result of the *solvesos*() function, see chapter 2.6 for more information on this, a better condition was found in *solution.problem*. The condition was changed to $whilesolution.problem == 0$, i.e., while there was not a problem with the optimization, this stayed zero, and the while loop continued. This was the solution utilized in the thesis, but this also made some issues when the problem was zero. However, the optimization printed: *"The problem is probably unsolvable"*, which caused the while loop to continue when the problem was infeasible. The way to solve this problem was to go in manually in the result which is printed and find at what value the problem stopped being feasible and changing the condition to stop at this value. For example, γ step in the CPL had to be changed from: $while\ solution.problem == 0$ to $while\ gamma <= 10E5$. There was not enough time to fix this issue, and it should be explored more.

Another minor issue that was easy to solve was remembering to define the sets and equations in each iteration in the while loop. This is to implement the newest γ and β values in the equations. If this is not implemented, the optimization will only use the original γ and β values, and the problem will always be feasible, which will cause the while loop never to stop.

5.6.2 The Unsolved Issue

Step four was the unsolved issue that needed more time to be solved, which updated the Lyapunov function found in the initialization step. The publications lacked detailed information on this step. Mazumder said: *"In this step, $\gamma, \beta, s_1(x)$ and $s_2(x)$ are hold fixed while $V(x)$ is determined,"*[11] and the full optimization problem defined in equation 3.11 in chapter 3.2.1. This information gives little information on how to execute the step, and the solution that was utilized was to define $V(x)$ once more in the same way $s_1(x)$ and $s_2(x)$ were. By using the *polynomial*() function. Then, the new empty $V(x)$ was implemented in the sets and equations. The equations were set as constraints, and the feasibility problem was attempted to be solved with an empty objective. The vector containing the empty $V(x)$ constants was set as decision variables. This immediately gave an infeasible problem, and the issues started.

The initially established Lyapunov function would be an obvious solution to the feasibility problem. This was not the case, and the issue was not solved, but it was investigated and possible problems could be:

- **The Solver:** It could be that the problem is with the YALMIP toolbox, and changing this to, say, SOSTOOLS could solve the issue easily. This would be a good subject for more investigation.
- **Too many constraints:** that lead to more complex feasibility problems. The new and empty $V(x)$ is used in two of the three sets and, therefore, in all of the equations that make the constraints. This makes most of the optimization problem dependent on this new $V(x)$ instead of the previously relatively easy variables $s_1(x)$ and $s_2(x)$, which was not as involved in the equations. This could be an issue.
- **Wrong use of the** *polynomial*() **function:** It was used to redefine $V(x)$, another solution should be chosen. This is the most likely issue, but no better option was found due to lack of time. This is something to be explored further. One solution could be to keep the original function but manage to go in and change the constants in the already established polynomial.
- **Could be wrong values for γ and β:** There are no references on what these values should be for the WECS.
- **Scaling of values:** It attempted to scale the values for the time-reversed van der Pol system, with no luck.

Chapter 6

Conclusion and Future Work

In this last chapter, the thesis concluded that the Sum of Squares method could apply to complex electrical systems with more investigation and testing. Lastly, there are suggestions for future work to investigate the method further, as well as possible investigations on the limitations of the scope made in the first chapter.

6.1 Conclusion

In this master thesis, a comprehensive study was conducted on large signal stability analysis through a systematic analysis of complex control concepts, answering the first objective of the thesis.

Chapter 3 did a deep dive into the Sum of Squares method, both the theory that is utilized and of the execution of the method itself. All the steps are explained thoroughly, and an idea of the optimization and how to utilize the YALMIP toolbox for the programming. The thesis objective two was to present and make the method accessible and clear for future related projects, which was completed.

The Positivstellensatz, which is the background for the set containment equation, is established in the theory, and the set containment equation is derived from the Positivstellensatz theorem. Then, the equation is used as a framework for the whole method and in each of the steps of the V-s iteration to establish an optimized Lyapunov function with the largest estimate of the region of attraction. According to thesis objective four, the method was then applied and tested on a time-reversed van der Pol system and a constant power load to learn the method and identify issues. During both chapter 4 and especially chapter 5, thesis objective three is executed by including a lot of code to make the programming accessible and provide a clear explanation of how to apply the method using MATLAB and the YALMIP toolbox. This resulted in much information, and numerous challenges were successfully addressed to get the method ready to apply to the wind energy conversion system, which was the fifth objective.

To be able to apply the method to the wind energy conversion system, several preliminary steps were made, first by applying the leader-follower philosophy to acquire a simpler system that would make the system easier to work with and had a more significant chance of the SOS method working. Then a PI Passivity Inspired Controller was implemented to be able to control the current in the proportional term and the speed in the integral term. Choosing the speed as the controlled variable in the integral part, differs from the norm in the industry but was utilized to have maximum

power extraction from the generator. An incremental model was applied to the system to move the equilibrium point from zero, which wouldn't be an interesting point to investigate stability for.

The Sum of Squares method was then applied, and it was immediately discovered that the system is too complex for the method to work using this software. The derived Lyapunov function $\dot{V}(x)$ was linearized for the initial optimization problem to be feasible. The first steps worked okay, but there was an issue with the stopping criteria, and a manual criterion was found by investigating where the loop should have stopped. The issue that could not be solved due to lack of time, appeared at step four, the $V(x)$ step, where the Lyapunov function was redefined as an empty polynomial and implemented into the original optimization problem with an empty objective, and the constants in the new $V(x)$ as decision variables. This feasibility problem was infeasible after a lot of problem-solving and there was no time left to solve the issue.

However, this thesis has made the Sum of Squares method accessible and shown a lot of coding to make it reproducible. With a few "tweaks," the issue should be solvable with more investigation into the YALMIP toolbox or perhaps with another solver such as SOSTOOLS.

6.2 Future Work

An obvious future work would be to try and solve problems that were encountered in this thesis, and there was not enough time to solve them. It would be very valuable to manage to get the Sum of Squares method to work, because it could save many calculations and a lot of time if it were possible to use optimization instead of extensive mathematical proofs. This method, if perfected, would be very easily transferred to other systems, both other wind energy systems and others, such as solar power and hydropower, and could make a lot of things a lot easier.

As mentioned in the discussion, a change of toolbox in MATLAB, or even a change in software, could solve the issue, and a good alternative is the SOSTOOLS toolbox for MATLAB and could be investigated. This would provide an estimate of the region of attraction and an accompanying stability certificate that could guarantee large signal stability for smaller connections to the power grid.

Other aspects to investigate further could be the simplifications or limitations of the scope made in chapter 1.3, such as simplification on the generator model and the damping coefficient, d. Due to their relative magnitude, the generator model was simplified by neglecting hysteresis losses and magnetic saturation effects.

Moreover, tuning the different gains of the controller is left for future work due to lack of time. This would also require an extensive investigation and much testing and could be a topic for further investigation.

Further, the wind energy conversion system was simplified by implementing a leader-follower philosophy, and removing this simplification would be a good place to start when the SOS method is operational, making the system even more complex. The system was linearized in the second step, and a logical step would be to remove this and get the method working for a nonlinear system. It could also be expanded to include an entire wind park in the future.

Bibliography

[1] UNFCCC. *The Paris Agreement*. URL: https://unfccc.int/process-and-meetings/the-paris-agreement (visited on 29/06/2023).

[2] António Guterres. *Secretary-General's statement at the closing of the UN Climate Change Conference COP28*. URL: https://www.un.org/sg/en/content/sg/statement/2023-12-13/secretary-generals-statement-the-closing-of-the-un-climate-change-conference-cop28 (visited on 20/02/2024).

[3] Mehmet Bilgili, Abdulkadir Yasar and Erdogan Simsek. 'Offshore wind power development in Europe and its comparison with onshore counterpart'. In: *Renewable and Sustainable Energy Reviews* 15.2 (2011), pp. 905–915. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2010.11.006. URL: https://www.sciencedirect.com/science/article/pii/S1364032110003758.

[4] Feng Zhao Mark Hutchinson. 'GLOBAL WIND REPORT 2023'. In: *Global Wind Energy Council* (2023). URL: https://gwec.net/globalwindreport2023/.

[5] Tutturen T.M. Nyhus-Solli J. 'Towards Plug-and-Play Control of Wind Power Systems: Scalable stability certificate guaranteeing large signal stability for entire wind parks'. MA thesis. NTNU, 2022.

[6] 'Technical Shortfalls for Pan European Power System with High Levels of Renewable Generation'. In: 2020.

[7] Adolfo Dannier et al. 'Doubly-Fed Induction Generator (DFIG) in Connected or Weak Grids for Turbine-Based Wind Energy Conversion System'. In: *Energies* 15.17 (2022). ISSN: 1996-1073. DOI: 10.3390/en15176402. URL: https://www.mdpi.com/1996-1073/15/17/6402.

[8] Marwa A. Abd El Hamied and Noha H. El. Amary. 'Permanent Magnet Synchronous Generator Stability Analysis and Control'. In: *Complex Adaptive Systems,* 6 (2016).

[9] Md. Enamul Haque, Michael Negnevitsky and Kashem M. Muttaqi. 'A Novel Control Strategy for a Variable-Speed Wind Turbine With a Permanent-Magnet Synchronous Generator'. In: *IEEE Transactions on Industry Applications* 46.1 (2010), pp. 331–339. DOI: 10.1109/TIA.2009.2036550.

[10] Zhenxi Wu et al. 'A Novel Method for Estimating the Region of Attraction for DC Microgrids via Brayton-Moser's Mixed Potential Theory'. In: *IEEE Transactions on Smart Grid* 14.4 (2023), pp. 3313–3316. DOI: 10.1109/TSG.2023.3262166.

[11] Sudip K. Mazumder and Eduardo Pilo de la Fuente. 'Transient stability analysis of power system using polynomial Lyapunov function based approach'. In: *2014 IEEE PES General Meeting — Conference Exposition*. 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939524.

[12] Mohsen Vatani and Morten Hovd. 'Lyapunov stability analysis and controller design for rational polynomial systems using sum of squares programming'. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 4266–4271. DOI: 10.1109/CDC.2017.8264288.

[13] Eline T. Bakke. *Plug-and-play control of PMSG-based Wind Turbines*. Specialisation Project in TET4510. Department of Electric Power Engineering, NTNU – Norwegian University of Science and Technology, June 2023.

[14] Hassan K. Khalil. *Nonlinear Systems*. Vol. 3. Pearson Education Limited, 2015.

[15] Moritz Diehl, Rishi Amrit and James B. Rawlings. 'A Lyapunov Function for Economic Optimizing Model Predictive Control'. In: *IEEE Transactions on Automatic Control* 56.3 (2011), pp. 703–707. DOI: 10.1109/TAC.2010.2101291.

[16] Peter Seiler Abhijit Chakraborty and Gary J. Balas. 'Susceptibility of F/A-18 Flight Controllers to the Falling-Leaf Mode: Nonlinear Analysis'. In: *JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS* (2011).

[17] Aristide Halanay and Vladimir Rasvan. *Applications of Lyapunov Methods in Stability*. Feb. 1993. ISBN: 978-94-010-4697-8. DOI: 10.1007/978-94-011-1600-8.

[18] G. Chesi. *Domain of Attraction Analysis and Control via SOS Programming*. Springer, 2011.

[19] Pablo A. Parrilo. 'Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization'. PhD thesis. California Institute of Technology, 2000.

[20] Marie-Françoise Roy Jacek Bochnak Michel Coste. *Real Algebraic Geometry*. Vol. 36. Springer Berlin, Heidelberg, 1998, p. 430. DOI: https://doi.org/10.1007/978-3-662-03718-8.

[21] J. Lofberg. 'YALMIP: a toolbox for modeling and optimization in MATLAB'. In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. 2004, pp. 284–289. DOI: 10.1109/CACSD.2004.1393890.

[22] Zachary Jarvis-Wloszek et al. 'Control Applications of Sum of Squares Programming'. In: *Positive Polynomials in Control*. Ed. by Didier Henrion and Andrea Garulli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 3–22. ISBN: 978-3-540-31594-0. DOI: 10.1007/10997703_1. URL: https://doi.org/10.1007/10997703_1.

[23] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[24] Chen Zhang et al. 'Synchronizing Stability Analysis and Region of Attraction Estimation of Grid-Feeding VSCs Using Sum-of-Squares Programming'. In: *Frontiers in Energy Research* 8 (Apr. 2020), p. 56. DOI: 10.3389/fenrg.2020.00056.

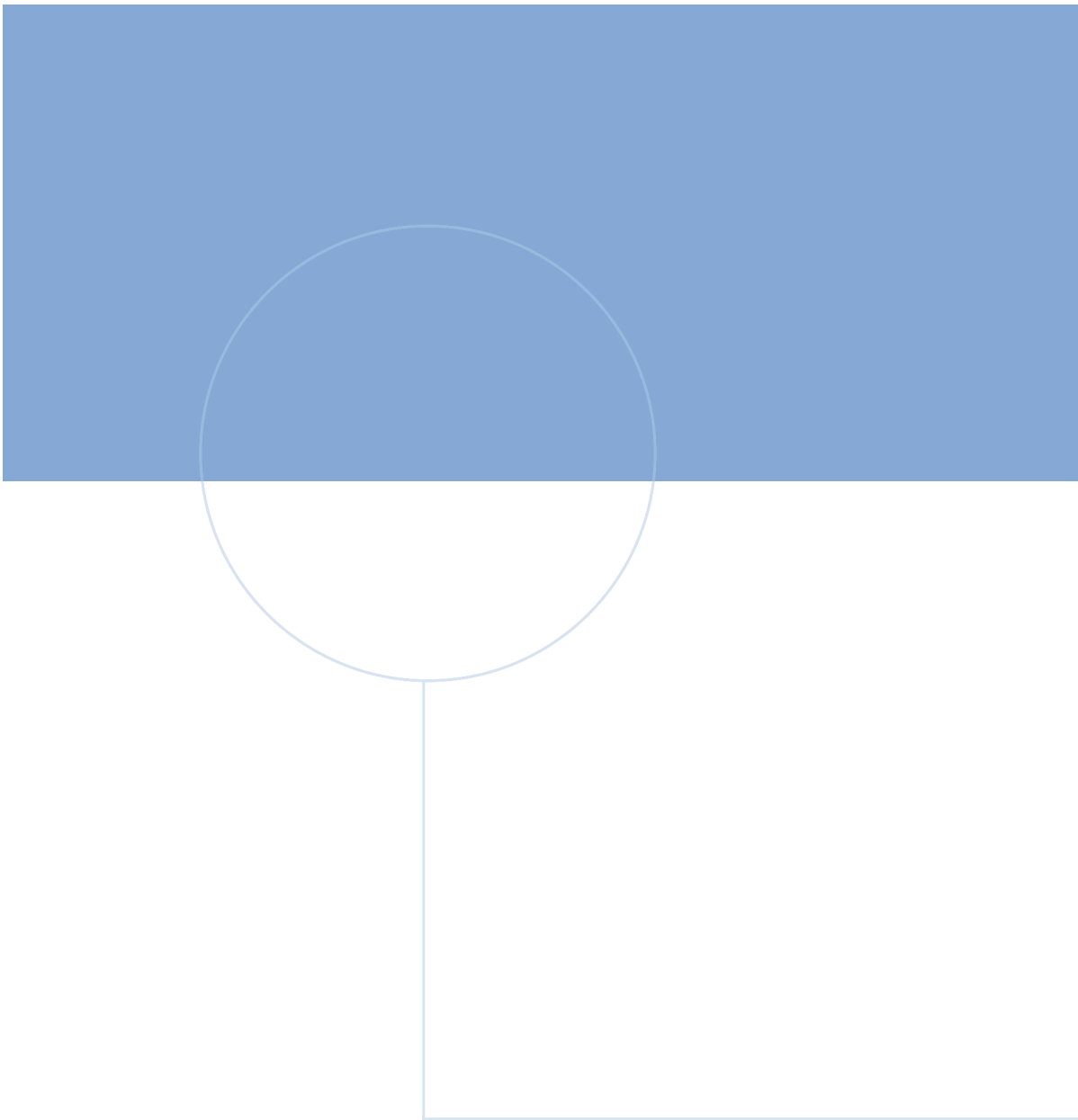
[25] Sudip K. Mazumder and Eduardo Pilo de la Fuente. 'Stability Analysis of Micropower Network'. In: *IEEE Journal of Emerging and Selected Topics in Power Electronics* 4.4 (2016), pp. 1299–1309. DOI: 10.1109/JESTPE.2016.2592938.

[26] U.L.P. Nguyen et al. 'Efficient Implementation of Mixing Sequence- Based Van der Pol Duffing System on the Modulated Wideband Converter Compressed Sensing Scheme'. In: *Arabian Journal for Science and Engineering* 48 (2022), pp. 6717–6727. URL: https://doi.org/10.1007/s13369-022-07529-3.

[27] Fanwei Meng et al. 'Application of Sum-of-Squares Method in Estimation of Region of Attraction for Nonlinear Polynomial Systems'. In: *IEEE Access* 8 (2020), pp. 14234–14243. DOI: 10.1109/ACCESS.2020.2966566.

[28] Hanejko F. *INDUCTION VS. PERMANENT MAGNET MOTOR EFFICIENCY — AUTO ELECTRIFICATION*. URL: https://www.horizontechnology.biz/blog/induction-vs-permanent-magnet-motor-efficiency-auto-electrification (visited on 02/02/2024).

[29] Rafael Cisneros et al. 'An adaptive passivity-based controller for a wind energy conversion system'. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 4852–4857. DOI: 10.1109/CDC40024.2019.9030090.



NTNU

Norwegian University of
Science and Technology