NTNU
Norwegian University of
Science and Technology

A Scheme for Creating an Small-Signal On-line Dynamic Security Assessment Tool

Using PSS/E and PacDyn

Knut Bjørsvik

Master of Energy and Environmental Engineering
Submission date: June 2016
Supervisor: Kjetil Uhlen, ELKRAFT

Norwegian University of Science and Technology
Department of Electric Power Engineering

Preface

Before you lies the thesis "A scheme for creating small-signal on-line dynamic security assessment tool", and is a contribution to the field of small-signal stability and Electrical power grid security. It has been written to fulfil the graduation requirements of the Energy and Environmental Engineering program at the Norwegian University of Science and Technology (NTNU), in the field of Electric Power Engineering and Smart Grids. Most of the literature research was conducted during the fall of 2015, background and insight I used to perform the final research and writing of this theses during the spring semester from January to June 2016. Most of my work have gone to program and implement features into the on-line assessment tool, and even though many hours have been used to write code, I have mostly loved every part of the work.

I would like to thank my supervisor, Professor Kjetil Uhlen, for proposing the problem and his excellent guidance and support during this process. I also wish to thank Professor Sergio Gomes Jr. and D. Sc. student Thiago Masseran, both had been flying in from Brazil, to work alongside my work and implement features in PacDyn. Their insight into modal analysis, control theory and their work in the field have made this research possible.

To all my teachers, especially Dr. Vijay Venu Vadlamudi for his teaching on the field of Electrical Power Systems, and fellow students at NTNU, I would like to thank you for your contributions during my time in Trondheim. Lectures and discussion with you have helped me understand complex subjects and motivated me for future research. Lastly I want to show my gratitude towards my girlfriend and roommates. You made it easy to maintain focus on my work. I hope that you enjoy your reading, and that the work presented in this thesis could be of interest.

Trondheim, June 10, 2016

Knut Bjørsvik

Abstract

The trend in today's electrical power system is that consumption and power peaks are increasing, which will push the grid to its limits. The large deployment of new renewable energy connected to the grid has also shifted the operational scene from only a few large power plants towards a multitude of medium and small generators, changing the natural flow of the power. The power flow is becoming less predictable, and more dynamic approaches for defining the limits are needed.

All synchronous machines that are connected to a steady-state power grid is working in synchronism, where all machines have found their power equilibrium point. The difference between the rotor angle voltage and the infinite bus are defining the power output of the generators. When machines that are connected through the grid experience changes such as changed consumption demand or other small disturbances, the equilibrium point is perturbed. As a result all the generators need to find back to a new equilibrium point, and while this occur, some power oscillations could arise between the machines. For security reasons the oscillations should damp out fast enough to maintain security in the grid.

By monitoring the eigenvalues of the system the damping ratio could be measured, and through this master thesis a on-line dynamic security assessment tool is created in Python. The tool is collecting (almost) real-time measurements from NordPoolSpot, updating a system model, load flow simulation is perform in PSS/E, small signal analysis is perform using Dominant Pole Spectrum Eigensolver in PacDyn and the security state of the system is presented to the operator. If the power system should move towards insecurity, alerts will arise, and the operator could remedy the threads. The results shows that the damping of power oscillation between Norway and Finland is worst during the winter, when the consumption is high. The scheme for assessing small-signal stability proved to be working, being able to track multiple eigenvalues and perform contingencies analysis and recommend remedy actions.

Oppsummering

Trenden i dagens energisystemer er at både effekttoppene og forbruket av elektrisk energi øker, noe som gjør at sikkerhetsmarginene i strømnettet blir mindre og mindre. Den store utrullingen av fornybar energi som er koblet til nettet har introdusert et skifte i hvilken retning strømmen blir sendt, fra kun noen store kraftverk i retning mot flere mellomstore og små kraftverk, ofte langt ute i nettet. Som resultat av dette blir flyten i strømnettet mindre forutsigbart, og nye stabilitetsproblemer kan oppstå.

Alle synkronmaskiner som er koblet til et stasjonært strømnett jobber synkront, og de har funnet seg en felles likevektpunkt som maskinene opererer på. Forskjellen mellom den indre rotorvinkelen til strømnettet er med på å bestemme hvor mye effekt som går ut fra hver generator. Når maskiner som er sammenkoblet gjennom et strømnett opplever endringer i form av endret last, eller andre mindre forstyrrelser, må alle synkronmaskinene finne et nytt likevektspunkt. Som regel oppstår det små effektsvingninger mellom maskinene inntil det nye likevektspunktet er oppnådd, og for sikkerhetsmessige grunner er det viktig at de dempes ut fortest mulig.

Ved å overvåke egenverdiene til systemet kan dempehastigheten måles, og det er gjennom masteroppgaven programmert et program som i (nesten) sanntid henter oppdaterte dataverdier, oppdaterer en model, gjør analyser, og gir en tilstandsvurdering av strømnettet med tanke på dempehastighet tilbake til operatøren. Eksisterende analyseprogram og metoder har blitt koblet sammen for å lage det dynamiske tilstandsprogrammet. Om kraftsystemet er på vei til å bli ustabilt, vil alarmer gå, hvorhen operatøren kan gjøre motvirkende handlinger. Data hentes fra NordPoolSpot, lastflytanalyser gjøres i PSS/E, småsignalanalyser i PacDyn ved bruk av Dominant Pole Spectrum Eigensolver (DPSE), og er automatisert ved bruk av Pythonkode. Resultatene viser at dempingen av effektsvingninger mellom Norge og Finland er verst på vinteren, ved høy last. Et annet resultat er at det lar seg gjøre å overvåke flere svingninger på samme tid, med utfallsanalyser og forbedrings forslag.

Contents

Chapter 1

List of Abbreviations

AMS Automatic measurement systems

CEPEL Centro de Pesquisas de Energia Elétrica (Electrical Energy Research Center), located in Rio de Janeiro

CIGRÉ Conseil International des Grands Réseaux Electriques (Council on Large Electric Systems), international assosiation to promote collaboration between power system experts.

D. Sc. Doctor of Science

DELT Power, (or rotor) angle with respect to an infinite busbar.

DPSE Dominant Pole Spectrum Eigensolver, algortithm developed by Nelson Martin.

Enaml Enaml is a programming language and framework for creating professional quality user interfaces with minimal effort, a superset of Python programming

FI Finland, Area in NordPoolSpot or Nordic44-model

FTP File Transfer Protocol, a standard network protocol used to transfer computer files between a client and server on a computer network

GENROU Round rotor synchronous generator

GENSAL Salient pole synchronous generator

GUI Graphical user interface, makes it possible for the user to interact with programming script

HVDC High Voltage Direct Current

IEEE Institute of Electrical and Electronics Engineers, a professional association

MVA Megavolt amperes, a unit used for measuring apparent power.

MW Megawatt, a unit used for measuring real power

NO1-5/8 Area of Norway, suffix 1-5 correspond to NordPoolSpot areas, if suffix is 1-8, then it correspond to Nordic44 model.

P/Q Real power/reactive power ratio

PMU Phasor measurement unit

PSSE Power System Simulator for Engineering, software from Siemens.

SCADA Supervisory Control And Data Acquisition

SE1-4 Area of Sweden.

SSSA Small-signal Stability Assessment

TF Tranfer function

WAMS Wide Area Monitoring System

WW Angular speed

Chapter 2

Introduction

2.1 Motivation

The task of maintaining stable and secure operation of the power system is expected to become more and more difficult in the future power system with more HVDC interconnections and an increasingly integration of renewable energy sources. Today there exist powerful off-line analysis tools for stability analysis that can be utilized to provide operators with critical information. However, a grid operator needs to be able to quickly respond to security threats in a dynamic system, and the demand of going from off-line to on-line analysis tool increases. Extensive work implementing on-line assessment tools for transient, voltage and frequency stability have been conducted , whereas the field on on-line small-signal stability assessment tools are still under research. [1].

2.2 Scope

The goal of this thesis is to apply existing methods and models for small-signal stability analysis and develop a scheme for "On-line Dynamic Security Assessment" for small-signal stability. This implies exploring the possibility of linking operational data with off-line analytical tools, and implement a solution that demonstrates (almost) on-line stability analysis. Continually result of the operational state, possible outcomes of contingencies and feasible remedial actions should provide the operator with necessary information to maintain stability security.

It is not a goal of this thesis to obtain a true and accurate model of the Nordic grid, but rather show the possibilities an on-line dynamic security assessment tool could provide. For this reason, the Nordic model used in this thesis could give results deviating from the actual system. Both the model and the operational data has been highly aggregated, henceforth reducing the details in the power flow and introducing many assumptions. Nevertheless the accuracy of the model, the scheme for "On-line Dynamic Security assessment" will still be the same.

2.3 Theoretical and practical relevance

This thesis is not trying to bring forth any new theoretical methods on the field of small-signal stability analysis, but rather rely on existing methods from modal analysis and load flow analysis. The theoretical and mathematical background for the methods used will briefly be explained, however the off-line analysis tools used are still under development, thereupon some methods and algorithm presented may change before final releases. PSS/E is used to perform load flow solutions, and PacDyn is used to perform small-signal stability analysis.

The small-signal on-line DSA scheme presented in this thesis is based on the steps presented in the "requirements for on-line DSA" given in CIGRE's review of on-line DSA tools [1], recited in Section 3.6. The steps are used as a general building blocks for the DSA, however the specific implementation is tailored towards a small-signal DSA scheme. The methods presented in the proposal scheme is a result of discussion with Professor Kjetil Uhlen, Professor Sergio Gomes Jr. and D. Sc. student Thiago Masseran.

As for the a **practical relevance** of the work presented in this thesis, it could be used:

1. As a reference scheme for creating an on-line DSA tool to determine the small-signal stability of a grid.

2. As an introduction to modal analysis and the relationship between eigenvalues, eigenvectors and the behaviour of the system.

3. To monitor the small-signal stability of a system, given a good system model and real operational points.

4. In future research of on-line DSA-tools and methods. Measurement data from SCADA systems, PMU systems, or state estimators could be used to get more accurate and detailed information about the operational points, and new algorithms and methods could be tested towards the result provided by the program.

5. To generate load flow solutions of the Nordic grid to be studied off-line.
6. As teaching material, showing how the modes/eigenvalues of a system can change during different operational points, and seasonal correlations.
7. As example code for programming in python and GUI-programming using Enaml and PyQt.

2.4 Objective and problem statements

2.4.1 Problem statement

Given an operational point, the grid operator should be able to get real-time insight about the modes of interest, while being confident that the track of the modes have been ensured. The DSA-tool should be able to report on power oscillations inherit in the system and provide information to the operator to help maintain the security of the system.

2.4.2 Objectives

The objectives of this research is:

- To develop a scheme for performing on-line small signal stability assessment
- Implement a working example DSA tool using Python by combining existing methods and analysis software, such as PSS/E and PacDyn.
- To evaluate strengths and weaknesses related to the example tool.
- Report on results of analysis
- Describe operational point in the Nordic grid more likely of poor damped modes.

2.5 Thesis outline

The outline for this thesis is as follow; in chapter 3 a theoretical introduction to power system stability and on-line DSA tools are provided, before a mathematical background for the methods used in this thesis is presented. In chapter 4, the methodology is presented, showing the steps of the proposed DSA tool and how the simulations are performed. Chapter 5 will present the results of running the DSA tool. Discussion of the results and the DSA scheme is found in Chapter 6. Conclusion and further work explored in Chapter 7. In the Appendix the source code for the proposed DSA tool is added and further system information, such as dynamic files, system files, base files and contingencies.

Chapter 3

Theory

3.1 Power System Stability background

Power system stability, as defined by IEEE/CIGRE Joint Task Force, "is the ability of an electric power system, for a given initial operating condition, to regain a state of operating equilibrium after being subjected to a physical disturbance, with most system variables bounded so that practically the entire system remains intact" [2]

The following part was covered more in detail in the author's pre-master project[3], and only a small recap will be presented here. Most of the background information in this Theory Chapter is found in the CIGRE's Review of On-line Dynamic Security Assessment Tools and Techniques or Machowski's excellent book about Power System Stability[1][4].

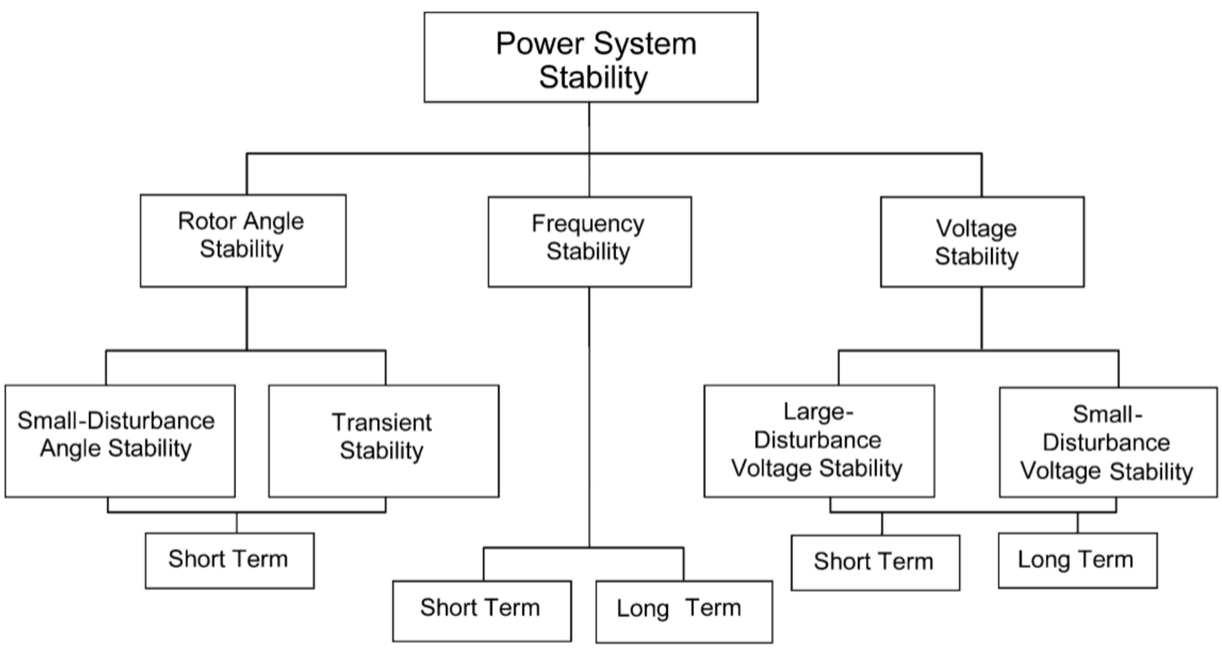

Figure 3.1: Classification of stability. [2]

Usually when dealing with power system stability, the overall system is divided into different stability categories. By maintaining the stability of each category, the total system stability will be maintained. According to well-known textbooks dealing with the subject of power system stability, the categories consist of rotor angle stability, frequency stability and voltage stability [4][5]. The categories refers to three quantities that is important to power system operation, generators voltage angle δ, system frequency f, and bus voltage magnitude $|V|$. The classification map can be shown in Figure 3.1

The real power P flowing in the system is highly dependent on the voltage angle δ between two points of the system, and the reactive power Q is highly dependent on the voltage magnitude $|V|$ between two points of the system. All machines are connected synchronous together and operates at the same frequency, which is highly dependent on the load/production ratio.

Rotor angle stability could be further divided into a transient subcategory and small-signal subcategory, determined by the size of the physical disturbance. Large disturbances affect the transient course of generators, while small disturbances are considered to be sufficiently small such that linearization of system equations is permissible during response analysis. Dynamic simulations are performed on transient runs, while steady-state analysis are performed on the small-signal analysis. In this thesis, the small-signal stability will be investigated.

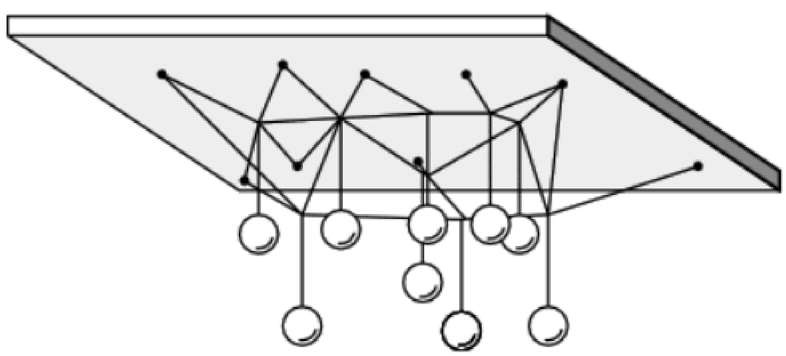

Figure 3.2: A mechanical analogue of swings in a power grid. Several masses are interconnected through elastic strings and fluctuations will ripple through the system if any mass is perturbed. Figure from [4], based on [6]

3.2 Small-signal stability

Definition: "Small-disturbance (or small-signal) rotor angle stability is concerned with the ability of the power system to maintain synchronism under small disturbances. The disturbances are considered to be sufficiently small that linearization of system equations is permissible for purposes of analysis." [2]

When all power generators in a steady-state power system is operating synchronous in a connected system, each generator has it's own equilibrium point. At the equilibrium point, the internal rotor angle δ and the output power P is constant. When a system is perturbed, the equilibrium point is lost, and based on physical laws each machine starts to accelerate or decelerate because of change in the torque. Some power oscillations or electromechanical oscillations could arise in the power system until all generators have found back to their new equilibrium point. A visual representation is shown in Figure 3.2, where a set of masses have found their equilibrium point related to each other. Should one of the strings be cut or a mass is perturbed, the system needs to find a new equilibrium point where the potential energy is at its minimum. Through the course of obtaining the new equilibrium point, some oscillations may occur, either between a group of masses towards another group of masses, a single mass to its neighbor mass, or a combination of all. The same oscillations could be found in a power system, often referred to as inter-area oscillations or local-area oscillations.

Small-signal analysis is the study of the electromechanical oscillations inherent in power systems. The synchronous torque of a machine could be divided into the two components synchronizing torque and damping torque. The damping torque correlates with the speed deviation $\Delta\omega$ and the synchronizing torque correlates to the rotor angle deviation $\Delta\delta$.

The system stability is dependent on the electromechanical torque components of all the synchronous machines in the system. Having insufficient synchronous torque could result in **aperiodic** or **non-oscillatory instability**, but for the most part, this problem is eliminated in power systems by use of voltage regulators on the generators. However, the damping torque is still often a problem, where the lack of damping torques results in oscillatory instability, giving that the amplitude of the oscillation increase for each cycle, or are not reduced fast enough to become stable. In such situations, some generators will perturb the system back as a response of the first perturbation, increasing the power oscillation for each oscillation swing.

Electromechanical modes should be reduced by a damping ratio of minimum 3-5 % to maintain system security, meaning the oscillation amplitude is reduces by 3-5 % each swing [7]. The oscillations should be reduced until the new equilibrium point is reached, or fluctuate around the equilibrium point with a small deviation, fulfilling the first or second Lyapunov stability criteria's.

In this thesis the small-signal stability will be evaluated by observing the eigenvalues of the system. While measurements from PMUs or Disturbance recorders could be used to monitor oscillations from large disturbances in the system, methods for analysing contingencies and security margins are of interest. Observing the small-signal stability through eigenvalues will enable this. The mathematics and the properties of eigenvalues are discussed in Section 3.7.2.

The eigenvalues are often referred to as modes in the system, and throughout the thesis, both will be used interchanged. The modes are classified into local modes and inter-area modes depending on where the oscillation is occurring, and the numbers of generators that is included in the oscillation. There are reports on the existence of two inter-area power oscillations on 0.58 Hz and 0.30 Hz in the Nordic system, oscillating between Norway-Sweden and Sweden-Finland respectively [8]. Later in this thesis, a scheme for monitoring such oscillations and eigenvalues are to be presented, giving continually insight of the modes of the system.

A power system should be able to deliver expected customer service all the time. The system should be built to be able to withstand plausible contingencies or disturbances that could arise. For all larger grid systems, this implies creating a system that could handle N-1 or sometimes N-2 contingencies, i. e. short-circuits, line trips, change in load demand etc.

3.3 Security states of the grid

When the power system is able to deliver expected service, for all plausible contingencies, the system is operating in **normal state**. Here all customers are delivered, no technical constraints are violated and the system can remain intact after being subject to plausible contingencies.

A power system is not constant, and the security state could change. Following the convention given in stability literature [1][4], the system could be in five security states. **Normal**

state, **Alert state**, **Emergency state**, **In extremis state** and **Restorative state**. By defining equality constrains and inequality contrains of a power system enables the classification of each state. Equality constraints is power match, total production should be equal to total consumption demand and total loss in a system. Inequality constraints is security limits such as voltage limits, frequency limits, damping ratio limits, heat limits, line current limits etc. In the *Normal state*, all equality and inequality constraints are satisfied and system margins are sufficient to withstand any of the plausible contingencies. In *alert state*, the system is weakened, whereas the equality and inequality constraints are still satisfied, while the system margins are not sufficient to withstand all plausible contingencies, which could violate the inequality constraints. In *emergency state*, the system operates such that some of the security (inequality) constraints are violated, making the system vulnerable. Going from *emergency state* towards *in extreme state* will make the system not intact. The system is then not able to deliver sufficient power to customers, meaning the equality constraints are violated, because lines, generator or major portions of the system has shut down. When corrective actions are performed to reconnect all facilities, the state is in *restorative state*. As a result, the system should be made intact again, moving the state back to alert or normal state. Figure 3.3 visually represents how the states could change.

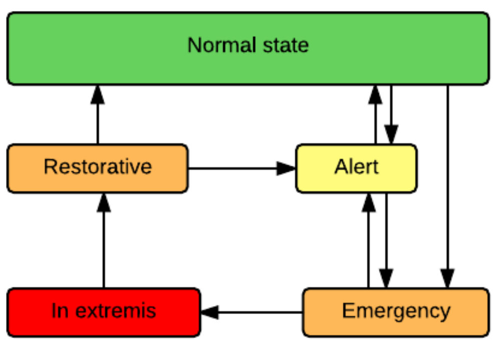

Figure 3.3: Operation states as presented by Dy Liacco (1968). An on-line DSA constantly check what state the system operates in. In alert and emergency state the system is still in intact, but remedies need to be conducted to keep the security intact. [1][4]

3.4 Level of assessment

Performing a security assessment of a power grid, there exist different levels of assessment. First, the current state need to be **monitored**, checking if the current situation is inside limits. Second level is **analysis** whether the system will stay intact after a disturbance, raising alert or emergency state if the contingencies will move the system such that violations are met. Third and lastly, **determine the security margin**, telling how much more the system could handle before it becomes unstable.

3.5 Moving towards on-line assessment

In today's deregulated power system, where the power demand is increasing more rapidly than grid investments, it is of economic interest for the grid operator to be able to operate the grid closer to the limits. Earlier, when power production mainly came from large power plant, it was feasible to perform off-line studies to determine stability limits. Uncertain predictions about the future had to be used, and the grid could therefore be oversized to ensure security for a long period, often up to 50 years, or the limits was set to conservative, to ensure secure operation for all situation. As the increase of decentralized renewable energy sources enters the system, the system is changing in a rapid manner. The act of pre-defining security limits is not that easy anymore, and the need of moving toward a continually assessment check is raising. Additional, the decrease of limits in the system often need rapid response from an operator to overcome emerging situations. As a consequence the need for real-time simulations increase, often called on-line assessments tools. These on-line dynamic security assessment tools (DSA) operate like a radar, performing a system scan and calculates the security of the grid. The on-line DSA could get updated data about the power system from measurements such as PMU, WAMS or SCADA systems. Through the on-line DSA tool the measurement data is connected to a network model, analysed, and any security results are reported back.

3.6 Requirements for On-line DSA

When implementing a DSA tool, CIGRE's Review paper made a list of requirements to be fulfilled;

"The basic functions of an On-line DSA system includes the following basic steps:

1. Take a snapshot of the power system condition
2. Develop a suitable network model
3. Combine any additional dynamic data and contingency data required to perform the assessment
4. Perform the analysis
5. Report on results of analysis
6. Raise alarms when security issues are deteced.
7. Identify secuirty issues and make recommendations on how to alleviate them. (This function is not always achievable. "[1]

3.7 Mathematical background

To be able to describe the dynamic behaviour of system, i.e a electrical power system, a set of n first-order nonlinear ordinary differential equations is needed when the system is of order n. The differential equations are n functions f of n state variables x, r control variables u and time t. Writing the equations in a vector-matrix notation gives [5]:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{3.1}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

and $\dot{\mathbf{x}}$ represents the derivative of the state variables $\mathbf{x}$ in respect to time. Often the time t is included into the state variables or control variables, reducing Equation 3.1 into:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3.2}$$

In the same manner m numbers of outputs y from the system can also be describes as m nonlinear functions g of the state variables $\mathbf{x}$ and control variables $\mathbf{u}$ giving the equations:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \tag{3.3}$$

Where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix}$$

When a system is represented in a state-space approach, all futuristic behaviour of the system is independent on past behaviour, but rather on internal states and future inputs. The state-space consist of a minimal representation of n linearly independent system variables needed to describe the state of the system. [5]

State variables could be physical quantities such as voltage, angle, speed etc, or mathematical quantities that describe the behaviour of the system. As long as the set of state variables are linear independent, the choice of states variables is just a matter of preferred coordinate system.

3.7.1 Linearize around an operation point

At equilibrium point $\mathbf{x_0}$ the system is in steady-state, giving:

$$\dot{\mathbf{x}}_0 = \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) = 0 \tag{3.4}$$

The steady-state, or small-signal, stability of a power system is the ability of the system to maintain synchronism when subjected to a small disturbance [4]. When operating the system at a equilibrium point $\mathbf{x_0}$, the system is steady-state stable if the system returns to the equilibrium point $\mathbf{x_0}$ after a small disturbance in the system.

Perturbing around the equilibrium point gives:

$$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x} \tag{3.5}$$

$$\mathbf{u} = \mathbf{u}_0 + \Delta\mathbf{u} \tag{3.6}$$

Inserting Equation 3.5 and Equation 3.6 into Equation 3.2 gives:

$$\dot{\mathbf{x}}_0 + \Delta\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{u}_0 + \Delta\mathbf{u}) \tag{3.7}$$

As long as the disturbances are small, the systems nonlinear equations f can be linearized around the equilibrium point $\mathbf{x_0}$. Expressing Equation 3.7 in terms of Taylor's series expansion yields:

$$\begin{aligned} \dot{x}_{i0} + \Delta\dot{x}_i = f_i(\mathbf{x}_0, \mathbf{u}_0) &+ \Delta x_1 \left.\frac{\partial f_i}{\partial x_1}\right|_{x^0,u^0} + \Delta x_2 \left.\frac{\partial f_i}{\partial x_2}\right|_{x^0,u^0} + \ldots + \Delta x_n \left.\frac{\partial f_i}{\partial x_n}\right|_{x^0,u^0} \\ &+ \Delta u_1 \left.\frac{\partial f_i}{\partial u_1}\right|_{x^0,u^0} + \Delta u_2 \left.\frac{\partial f_i}{\partial u_2}\right|_{x^0,u^0} + \ldots + \Delta u_r \left.\frac{\partial f_i}{\partial u_r}\right|_{x^0,u^0} + \mathcal{O} \end{aligned} \tag{3.8}$$

Knowing that $\dot{x}_{i0}$ and $f_i(\mathbf{x}_0, \mathbf{u}_0)$ is both equal to zero from Equation 3.4 and neglecting higher-order terms $\mathcal{O}$ reduces the system. Doing the same for Equation 3.3, the system could be represented in normal form:

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \tag{3.9}$$

$$\Delta\dot{\mathbf{y}} = \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{u} \tag{3.10}$$

where:

$\Delta\mathbf{x}$	- State vector of state deviation, $\in \mathbb{R}^n$
$\Delta\mathbf{y}$	- Small deviation of output variables, $\in \mathbb{R}^m$
$\Delta\mathbf{u}$	- Small deviation of input variables, $\in \mathbb{R}^r$
$\mathbf{A}$	- State matrix of the system, $n \times n$
$\mathbf{B}$	- System input matrix, $n \times r$
$\mathbf{C}$	- System output matrix, $m \times n$
$\mathbf{D}$	- System direct transfer matrix, $m \times r$

and:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_r} \end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_r} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial u_1} & \cdots & \frac{\partial g_m}{\partial u_r} \end{bmatrix} \tag{3.11}$$

The system matrix, or state matrix **A**, is often also called the Jacobian matrix, after the mathematician Carl Gustav Jacob Jacobi (1804-1851) [9]. The behaviour of the system around a equilibrium point is related to the eigenvalues of **A**, and the system could therefore be analysed by eigenvalue analysis. If all the eigenvalues of the system have negative real parts, then the system is stable around the equilibrium point.

3.7.2 Eigenvalues

When the system is formulated in state space form, this is a great base for doing more sophisticated analysis on the system. Because the total system is described in the state space, characteristics of the system could be described. Obtaining the eigenvalues and the eigenvectors of a system is the basis for subject such as observability, controllability, sensitivity and participation factors.

The scalar parameter λ is called an eigenvalue of matrix **A** if there exist non-trivial solutions to:

$$\mathbf{A}\mathbf{w} = \mathbf{w}\lambda \tag{3.12}$$

where $\mathbf{w} \neq 0$ and is a column vector of $n \times 1$. The eigenvalues can be found by rearrange Equation 3.12 in the form

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{w} = \mathbf{0} \tag{3.13}$$

where **I** is the $n \times n$ identity matrix and **0** is a $n \times 1$ column vector of zeros. Since $\mathbf{w} \neq 0$, the equations is true only when

$$det(\mathbf{A} - \lambda\mathbf{I}) = 0 \tag{3.14}$$

From the expanded version of Equation results the *characteristic equation* that has n solutions of λ satisfying Equation . Each eigenvalue λ_i could be a real or complex number. Complex number of a real **A** comes in conjugate pairs. The eigenvalues can be written in the form:

$$\lambda = \alpha + i\Omega \tag{3.15}$$

The real part correlates to the time constant of the exponential function, whereas the imaginary part correspond to the frequency of oscillation.

$$\Omega\left[\frac{rad}{s}\right] = 2\pi f\ [Hz] \tag{3.16}$$

The negative real line correlates to the damping of the system.

The damping of the amplitude for each oscillation is denoted as relative damping and is calculated as [4]

$$\xi = \frac{-\alpha}{\sqrt{\alpha^2 + \Omega^2}} \tag{3.17}$$

All electromechanical eigenvalues should have at least 5 % relative damping to be operating in secure state. An on-line DSA on small-signal stability should therefore monitor the damping of such electromechanical oscillations.

Understanding the results

During monitor the eigenvalues are presented visually as scatter plot in a two-axis system, the real-axis correspond to damping time constant and the imaginary axis correlates to the frequency of the oscillation, as shown in Figure 3.4. The relative damping ratio ξ is governed by its angle from the imaginary line, and calculated as stated in Equation 3.17. Electromechanical modes of higher oscillation speed gives information about local modes, while slower oscillation modes are related to inter-area modes, where the frequency of inter-area modes usually are in the range of 0.1-1.0 Hz [8].

The position of the modes changes during different operation points, and it is of importance to keep the modes in the secure region, where the damping ratio is higher than 5 %. Should a mode related to any of the contingencies monitored move inside the insecure area of 5 % damping ratio, an alert should arise. If a non-contingency mode enters the insecure area, an emergency alert should arise. A contingency mode in the unstable area sets the system in emergency state, while an non-contingency mode will make the system unable to operate, and the unwanted in-extreme state is obtained.

The modes could also be monitored along a time line, plotting the damping ratio ξ versus the time. Information about the modes, such as the frequency and position are then lost, and in these situation an extra time line containing information about the frequency should be added. The time line plot is good for showing how the damping of different modes changes relative over time, while a two-axis plot is good for showing the area that each mode is operating within.

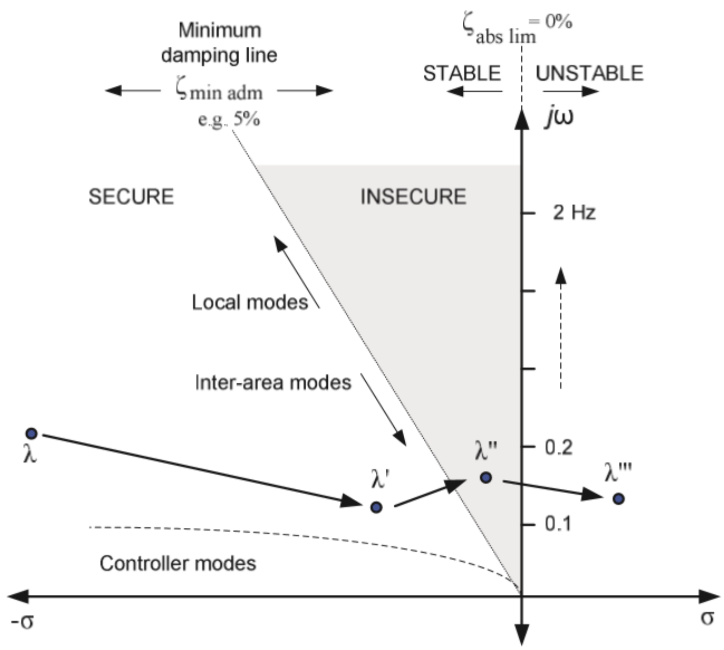

Figure 3.4: The eigenvalues are plotted visually into a two-axis plot, showing the secure and insecure areas, figure from [7]

3.7.3 Eigenvectors

For each λ_i a related eigenvector w_i exist of the form

$$\mathbf{w_i} = \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{ni} \end{bmatrix} \tag{3.18}$$

In fact, an infinite numbers of eigenvectors exist for each λ because $k\mathbf{w_i}$ is also a eigenvector related to λ_i, where k is a scalar. The length of the vector space is not of importance, only the direction.

Summing up all eigenvectors $\mathbf{w_i}$ corresponding to each eigenvalue λ_i in a $n \times n$ matrix $\mathbf{W}$ yields:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_n \end{bmatrix} \tag{3.19}$$

W is called the right eigenvectors of the state matrix **A**.

In a similar way, the left eigenvectors **U** could be obtained by solving

$$\mathbf{uA} = \mathbf{u}\lambda \tag{3.20}$$

where $\mathbf{u} \neq 0$ and is a column vector of $n \times 1$. The left eigenvectors could also be obtained by the inverse of matrix **W**. Because of this, multiplying the left eigenvectors **U** with the right eigenvectors **W** results in the the identity matrix I.

$$\mathbf{UW} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.21}$$

3.7.4 Diagonalizing the state matrix

Working with the state matrix is often tedious work, and therefore a similarity transformation is often useful to simplify computations. By combining the left and right eigenvectors together with the state matrix **A** the state matrix could be rewritten in a more simplistic way, where the eigenvalues are found on the diagonal, and the non-diagonal elements equals to zero. The diagonalized matrix Λ is therefore:

$$\Lambda = \mathbf{W}^{-1}\mathbf{AW} = \mathbf{UAW} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{3.22}$$

The time response for the homogeneous linear equations of the form

$$\dot{\mathbf{x}} = \mathbf{Ax} \tag{3.23}$$

would be of the form

$$x(t) = e^{At}x_0 \tag{3.24}$$

where x_0 is the initial values for the system. The same time response could be obtained from a diagonalized matrix, giving a linear combination of aperiodic and oscillatory time responses such as

$$\mathbf{x}(t) = \mathbf{W}e^{\Lambda t}\mathbf{U}\mathbf{x}_0 \tag{3.25}$$

where

$$e^{\Lambda t} = \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}$$

3.7.5 Modal form

In order to utilize matrix diagonalization for the solution of the state equation the state vector $\mathbf{x}$ is transformed into a new state vector $\mathbf{z}$ using the linear transformation

$$\mathbf{x} = \mathbf{W}\mathbf{z} \tag{3.26}$$

Solving Equation 3.26 in respect to $\mathbf{z}$ gives

$$\mathbf{z} = \mathbf{W}^{-1}\mathbf{x} = \mathbf{U}\mathbf{x} \tag{3.27}$$

Substituting Equation 3.26 into Equation 3.23 and using the diagonal matrix λ from Equation 3.22 yields

$$\dot{\mathbf{z}} = \mathbf{W}^{-1}\mathbf{A}\mathbf{W}\mathbf{z} = \Lambda\mathbf{z} \tag{3.28}$$

The elements of the modal matrix $\mathbf{z}$ is often referred to as modes. Each mode contains information about aperiodic and oscillatory behaviours found in the system of the state matrix $\mathbf{A}$. There exist no physical meaning to the modes of a system, but it represents oscillations found in the system.

3.7.6 Controllability

The modal variable $z_i(t)$ can be represented as a linear combination of the state variables, such as

$$z_i(t) = \sum_{j=1}^{n} u_{ij} x_j(t) \tag{3.29}$$

where u_{ij} is the i, j element of the left eigenvectors $\mathbf{U}$. Equation 3.29 shows that the left eigenvector $\mathbf{U}$ contains information about the controllability of the modes z_i, based on changing state variable x_j. Only when the element U_{ij} is great, the state variable x_j has a great impact of mode variable z_i.

3.7.7 Observability

In the same manner as the modal variable $z_i(t)$ is a linear combinations of the state variables x, the state variables x_k could be represented as a linear combination of the modal variables $z_i(t)$, such as

$$x_k(t) = \sum_{i=1}^{n} w_{ki} z_i(t) \tag{3.30}$$

where w_{ki} is the k, i element of the right eigenvectors $\mathbf{W}$. Equation 3.30 shows that the right eigenvector $\mathbf{W}$ contains information about the observability of the state variables x_k, based on the mode variable z_i. Only when the element w_{ki} is great, the mode variable z_i can easily be seen in state variable x_k.

The normalized eigenvector $\mathbf{w}_i$ corresponding to the eigenvalue λ_i is often referred to as the mode shape for that particular eigenvalue. Plotting the mode shapes into a complex plane gives a visual representation of how the mode will be observed in the system, when the mode is excited by a disturbance. The size of the disturbance will not change the shape of the mode.

3.7.8 Participation factor

Sometimes it is interesting to find out which parts of the system that is responsible for the various modes. By combining the observability with the controllability, a relationship could be made between the state variable x_k to the mode z_i. The relationship factor is called participation factor p_{ki}, and is defined as

$$p_{ki} = u_{ik} w_{ki} \tag{3.31}$$

The participation factor gives a good correlation between the ith modal variable and the kth state variable. When placing remedial actions against oscillations in the system, such as stabilizers and damping controllers, finding the place with the highest participation factor to the given mode is a good choice.

3.7.9 QR-method

When the system dimension n is getting to large, a problem of finding the eigenvalues based on the *characteristic equation* starts to be difficult. A solution to this is to create a similarity transformed matrix, keeping the eigenvalues intact. Rutishauser (1958) developed a transformation algorithm called LR-transformation, by forming a lower and upper triangular matrices, but this method often meet convergence problems. Francis build on the work of Rutishauser to develop the QR-transformation [10]. Where the LR-transformation could only find the eigenvalues for a symmetrical matrix, the QR-transformation could also find solutions to unsymmetrical matrices. The similarity transformed matrix is found by iteration, where

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{Q}_k\mathbf{R}_k\mathbf{Q}_k = \mathbf{Q}_k^{-1}\mathbf{A}_k\mathbf{Q}_k = \mathbf{Q}_k^T\mathbf{A}_k\mathbf{Q}_k \tag{3.32}$$

$\mathbf{A}_{k+1}$ and $\mathbf{A}_k$ are similarly, meaning they have the same eigenvalues. $\mathbf{Q}_k$ is an orthogonal matrix, meaning $\mathbf{Q}_k^{-1} = \mathbf{Q}_k^T$.

The QR-transformation to make the eigenvalues appear along the diagonal in a diagonal matrix is rather slow, and requires heavy computation. For a system with rank N, the computation needed to perform the QR-transformation is then N^3. Even though the QR-method is a robust method to obtain the eigenvalues of a system $\mathbf{A}$, due to the slow computation it is not feasible to run real-time application using the QR-transformation for larger systems. Other more fast approaches need to find place. One of these methods is the DPSE-method.

3.7.10 Dominant Pole Spectrum Eigensolver-method (DPSE)

The Dominant Pole Spectrum Eigensolver is a algorithm developed by Nelson Martins, working at CEPEL in Brazil [11]. The algorithm was developed from a combination of two other algorithms, the *Refactored Bi-Iteration*[12] and *The Dominant Pole Algorithm*[13]. The first is a subspace iteration method derived from the the Bi-Iteration algorithm, but using multiple moving shifts. The latter is a one-eigenvalue-at-a-time method which computes the dominant closed-loop poles in a given transfer function $F(S)$. The DPSE-method computes efficiently and simultaneously the closed-loop poles in any high-order scalar transfer function.

A transfer function can be obtained using the normal state space notation given in Equation 3.10, so that

$$F(s) = \mathbf{C}^T \cdot (s\mathbf{I} - \mathbf{A})^{-1} \cdot \mathbf{B} + \mathbf{D} \tag{3.33}$$

The m most dominant poles of a transfer function will approximately reproduce the behaviour of the full transfer function. The reduced system is therefore a good measurement to find the eigenvalues. The similarity between the full transfer function and the reduced model is shown in Nelson Martins paper [11].

The algorithm requires the user to specify m initial shift that the person wants to converge to eigenvalues. A **LDU** factorization matrix is used to guess the right and left eigenvectors for each specified shift. The right and left guessed eigenvectors are obtain by solving

$$\mathbf{C}^T \cdot \mathbf{w}(s_i^{(k)}) = 1 \tag{3.34}$$

$$-\mathbf{B}^T \cdot \mathbf{u}(s_i^{(k)}) = 1 \tag{3.35}$$

where $s_i^{(k)}$ is the i shift of the k iteration, $w(s)$ and $u(s)$ is the right and left eigenvector. A reduced subspace is obtained through diagonalization and similarity transformation and the respective eigenvalues are found. A reorthogonalization process is then conducted for every iteration to prevent repeated eigensolutions. For every iteration, the shift moves towards a new position. The speed benefits of using the DPSE algorithm is that instead of inversing a $n \times n$ matrix **A**, a subspace of size $m \times m$, $m << n$, is inverted to get the eigenvalues. These new eigen-

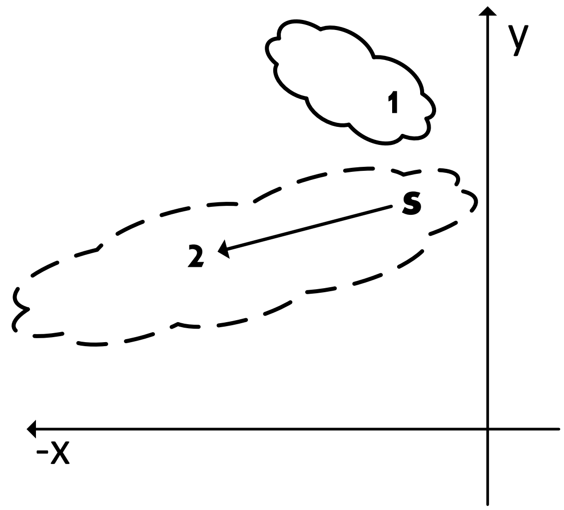

Figure 3.5: The DPSE-algorithm illustrated by using dominant regions. The shift will move towards eigenvalue 2 because of the attraction of its dominant region, even though eigenvalue 1 is closer. The arbitrary dominant region is found through a specified transfer function $F(s)$ where the eigenvalue is observed.

values of the the reduced matrix is then used as the new set of shifts $s_i^{(k+1)}$ for the next iteration. The algorithm stops when the system mismatch vector have converged below some small tolerance.

Figure 3.5 demonstrates the use of DPSE-method. In a Reyleigh method, the solution is trying to converge to the closest eigenvalue, in this example eigenvalue 1, but in the DPSE-method the dominance poles of eigenvalue 2 will attract the shift towards itself.

In the latest, yet unreleased, version 9.8.0 of PacDyn they included a method for having multiple DPSE running in parallell. Even though a single DPSE-algorithm does not converge to the same eigenvalues for different shift, this could happen when running multiple DPSE-algorithm on different transfer functions. The problem is under investigation, and may have been fixed before the new version is released. In some operation point, a chosen transfer function could be dominated by a specified eigenvalue, but in a different operation point, the same eigenvalue is not highly dominating in the same transfer function. In a situation like this, the DPSE-method could converge to a different eigenvalue than is of interest, meaning loose of track.

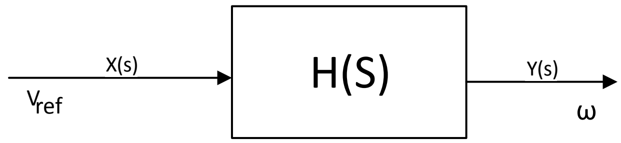

Figure 3.6: A transfer function is a black box that manipulates the input to a output through a function $H(s)$. For instance V_{ref} to ω.

3.7.11 Transfer function

A transfer function is a *black box-representation* **H** of the behaviour of the system from an input variable x to an output variable y. It is a representation of how any step in variable x will affect the output y.

$$y = \mathbf{H} \cdot x \tag{3.36}$$

Dependent on the complexity of the system, the transfer function **H** can vary in order and complexity, from a constant to a time-delayed function $f(x)$. Monitoring electromagnetic oscillation can usually be seen by tracking the transfer function over reference voltage V_{ref} input to a rotor speed ω as seen in Figure 3.6.

3.7.12 Generation Hopf Bifurcation

Generation Hopf Bifurcation method is a corrective measure method under development of the Brazilian D. Sc. student Thiago Masseran [14]. The method is a modification of the Hopf Bifurcation method[15][16][17][18]. Usually the Hopf Bifurcation method is used to find values of control systems parameters that is making eigenvalues cross security boundaries. The method finds the minimum parameters variation of control system variables to obtain a desired damping factor.

Instead of changing the control system parameters, the Generation Hopf Bifurcation method will try to find the minimum generation variation needed to obtain a desired damping factor. The method is therefore an optimization problem, solved by using Lagrange method, where the objective functions and its constrains are:

$$\min \quad f_{\text{obj}}(P) = \sum_{i=1}^{n} (P_i - P_{i0})^2 \tag{3.37}$$

$S.t.$

$$\alpha(P) + \frac{\xi}{\sqrt{1-\xi^2}}\Omega(P) = 0 \tag{3.38}$$

$$\sum_{i=1}^{m} P_i - \sum_{i=1}^{m} P_{i0} = 0 \tag{3.39}$$

where P_i is the new generation output of generator i, P_{i0} is the original generation of generator i, $\alpha(P)$ is the real value of mode λ, $\Omega(P)$ is the imaginary value of mode λ and ξ is the desired damping ratio of mode λ.

The Generation Hopf Bifurcation method uses a generation sensitivity to represent the sensitivities of mode λ in relationship to active power of an generator, and is important when finding the most optimal change in generation dispatch.

At present time, some limitations to the method have been noted, such as the changed losses of the system have not been considered, and that only the MVA base is used as the upper generation limit and not the specific maximum production. In the on-line scheme presented later, the generators are turned on according to the needs, resulting in changes in the system. However the DPSE method will not have access to perform these changes, but will try to present a solution based on the situation and parameters given.

3.7.13 Load flow analysis

Full Newton-Raphson iteration method in PSS/E version 33 is used to converge the load flow solutions, and the updated system file is used in Pacdyn to perform small-signal stability analysis. The load flow method is iterating until

$$I_i = \sum_{j=1}^{n} Y_{ij} U_j \tag{3.40}$$

is satisfied, and all load and generation condition is satisfied within a specified error limits. [19]

Chapter 4

Method

The main objective of this thesis is to create a scheme for small-signal stability "on-line dynamic security assessment tool". Throughout this chapter the methods and models used for an example tool is presented, linking operational data with analytical tools to perform on-line stability analysis.

4.1 Software

The following software's was used throughout the scheme:

PSS/E version 33 A commercial software created by Siemens to perform load flow simulations. Full Newton Raphson-algorithm was used to converge the load flows. [19]

PacDyn version 9.8.0 Unreleased version of a commercial software created by CEPEL. The program is used to perform small-signal analysis. In the current unreleased version of PacDyn a module for real-time monitoring is under development. The QR-method was used during initiation of the real-time monitoring, DPSE method was used to track the eigenvalues during simulation, and the Generation Hopf Bifurcation was used to give remedial actions. [20]

Python version 2.7 Python is a programming language. The Anaconda2.7 collection is used as a compiler. [21]

PyQt4 PyQt4 is a GUI (graphical user interface) language.[22]

Enaml Enaml is a python package for easy building block of PyQt4 code. [23]

Plot.ly A python package to create interactive visualizations of the results.[24]

4.2 System description

A remodeled Nordic44-model is used to perform simulation, and information about the original system model is retrieved from Hamre's master thesis [25]. Originally the model consisted of 44 buses, 43 loads and 61 generators, whereas 20 generators in Norway, 29 in Sweden and 12 in Finland. All generator dynamic models are "GENSAL" and "GENROU", being standardized salient pole generator and round generator models respectively, and depends on the dominating hydro or nuclear/other thermal of each bus. The swing bus is located in east part of Sweden, at bus 3300. The Nordic44 model consist of 14 areas, whereas eight in Norway, four in Sweden and two in Finland.

The model is trying to replicate characteristics found in the Nordic grid, and real production and consumption data is therefore fed into the model.

A few minor changes to the Nordic44-model was done, to make the model ready for real-time simulations. The generators that was connected to the same bus was aggregated into one equivalent generator, and data describing one original generator was saved into a spreadsheet, shown in Table B.1. This spreadsheet is then used to recalculate the aggregated generator data depending on how many generators that should be turn on. This is done because it makes the model easier to interact with, and will be explained further in Section 4.5.

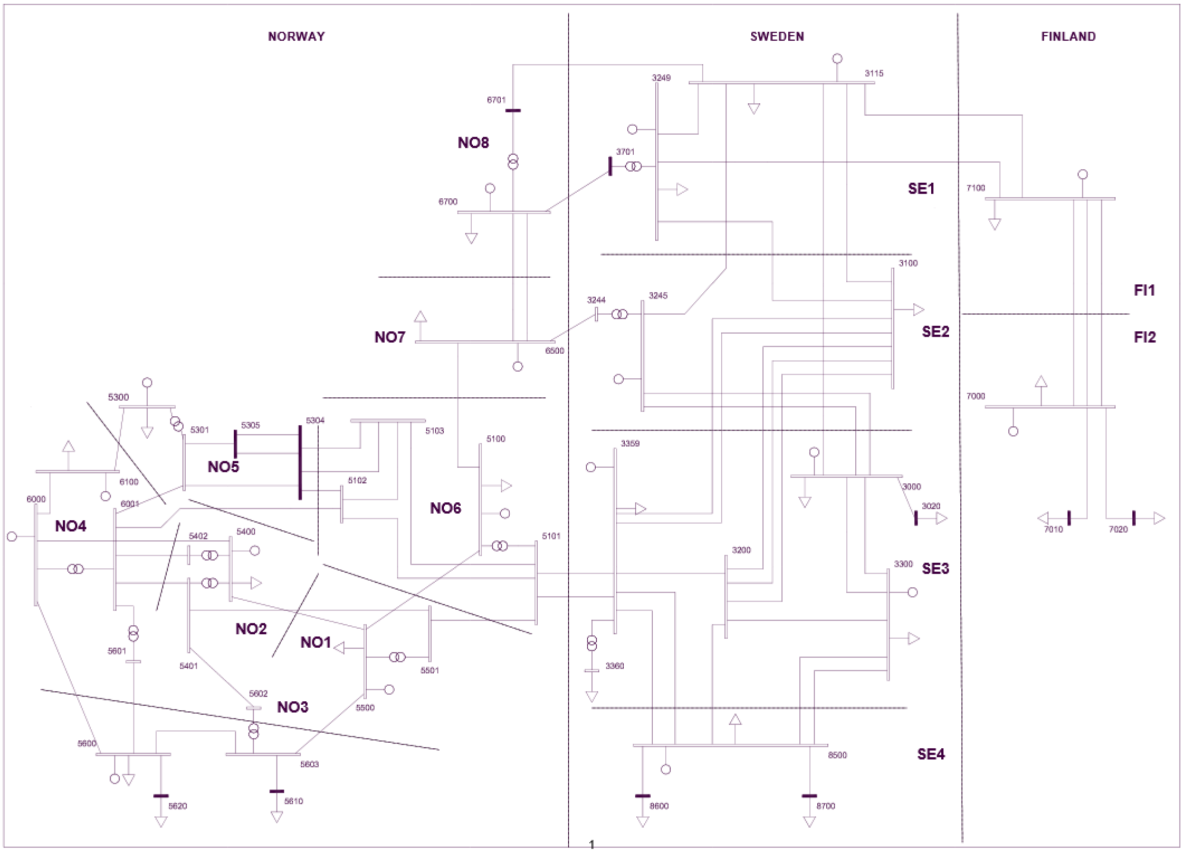

Figure 4.1: A single line diagram showing the aggregated Nordic44 model.

4.2.1 Area mapping from NordPoolSpot to Nordic44-model

The model is populated with data from NordPoolSpot[26], and since the areas in NordPoolSpot/Elspot is not exactly matching with the Nordic44-model, a remapping was performed following the conventions given in Hamre's thesis[25] and shown in Table 4.1.

A single line diagram of the model is shown in Figure 4.1. Hamre mentioned in her thesis that some errors had been found in the model, but where not improved. They have not been altered anything in this work either, since the exact result is not that much of importance, compared to showing that the real-time monitoring system could work satisfactory. Some of the errors are the mapping of generators and loads into the areas, and also some line ratings have been noted could be incorrect. In some situation the production from a generator exceeded the production limit, and this was solved by adding more generators than originally stated.

Elspot / NordPoolSpot area	Nordic44 model
NO1	NO1+NO6
NO2+NO5	NO2+NO3+NO4+NO5
NO3	NO7
NO4	NO8
SE1	SE1
SE2	SE2
SE3	SE3
SE4	SE4
FI	FI1+FI2

Table 4.1: Mapping of NordPoolSpot data into Nordic 44 [25]

4.2.2 Mapping of HVDC-cables

The Nordic grid also have some HVDC-connections connected to the system, and these are modelled simply as loads in the Nordic44-model. The cable are connected to the system as shown in Table 4.2. It could be noted that the cables Finland-Estland and Finland-Russia is aggregated together, and also Sweden-Poland and Sweden-Germany. Since the HVDC-cables are modelled as loads, their characteristics will not contribute to any oscillations in the system. Passing large power through HVDC-converters could introduce harmonic ripples in the system. While this could affect the stability of the system, it has not been included in this study. For later references, good HVDC-models should be considered used during simulations.

Cable/connection	Between	Capacity [MW]	Corresponds to bus
Nordned	NO-NL	700	5620
Skagerak 1-4	NO-DK1	1500	5610
KontiSkan	SE-DK1	550	3360
To Zealand	SE-DK2	—	8600
SwedPol + Baltic	SE-POL + SE-DE	600+600	8700
FennoScan	SE-FI	1300	3020
FennoScan	FI-SE	1300	7010
EstLink 1 & 2 + Russia	FI-EE+FI-RU	1000	7020

Table 4.2: Mapping of HVDC-cables into Nordic 44 model. [25]

4.3 Implementation of the on-line DSA

The source code for the on-line DSA tool is provided in Appendix A.1 and the Enaml-code to create the GUI is provided in Appendix A.1.1. The tool was implemented by following the guideline steps described in the requirements for on-line DSA tools presented in section 3.6. For each iteration throughout the monitoring process, multiple steps needs to be performed. The steps are presented in Figure 4.2, and will be explained in more detail in the following sections. In the beginning of a simulation the system needs to be initialized, applying the transfer functions the DPSE-method are to track during simulation. PacDyn is ready to monitor the system when the transfer functions, starting guess shift, dynamic files and contingencies files are provided. The python program is initiated by providing a starting and ending point, the time steps to be simulated, the system base files and a folder to save the results. For each iteration a time stamp is sent to the program to initiate the multi-step process. As long as the final ending point have not been reached, the program will continually simulate, pulling new data into the process.

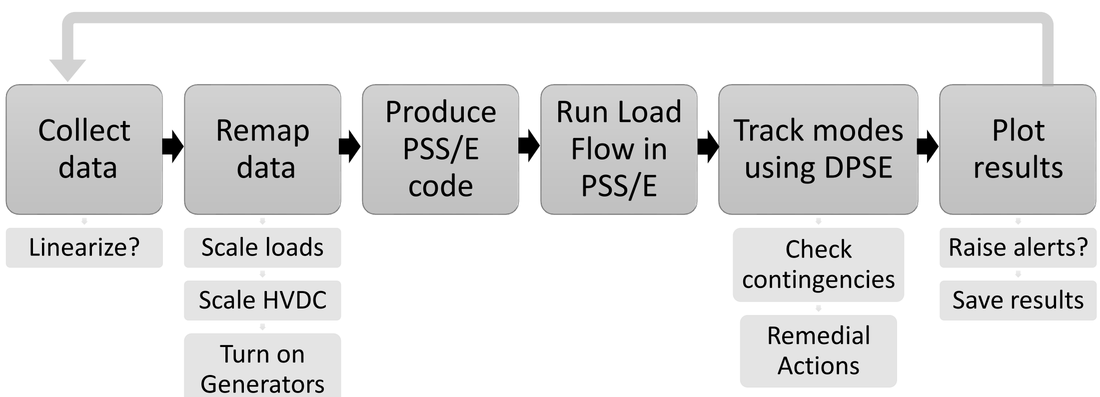

Figure 4.2: The building blocks of the small-signal DSA tool. The figure present the processes performed for each iteration.

4.4 Collecting the data

In a real-time monitoring system data input is essential to make the system dynamic. To simulate change of the operation point in the model, the model is updated by data from NordPoolSpot. The process of collecting data is shown in Figure 4.3, and the source code is found in Appendix A.2. The program receives a specified date and time, and it returns the operation point of the Nordic grid for that specified time.

Lack of real-time data was eluded by pulling historical data from NordPoolSpot's server and interpolate the data to simulate real-time behaviour. If the specified time is between two full hours, a interpolation is done by pulling the last full hour and the next full hour, conducting a interpolation as shown in Figure 4.4. This reduces the step interval and improves PacDyn's ability of not losing track of a mode, since the last known mode is used as a starting guess for the next one.

In the FTP-server, the data is saved in different folders and files. The different countries has their own folder, and inside the folder, each week has a summary file containing all production, consumption and flow data for that country and week. The program therefore need to collect data from multiple files, skim through the files and pull out the relevant information to be pushed to the model.

A filter scheme is reducing the weekly report by; first checking if the data contains information about the production, consumption or flow, secondly, checking if the data has the same date stamp as specified, and then thirdly finding the correct hour, and lastly neglect all irrelevant areas.

The resolution of the data from NordPoolSpot has only one hour interval, and this could result in problems for the eigenvalues to converge to the correct mode when using the DPSE-algorithm because of large steps in the operation points. As stated, an interpolation scheme is performed to simulate less steps. For every call to the process of collecting data, the program determines if an interpolation is needed. Should the specified time consist of a full hour, meaning minutes and seconds is equal to zero, normal collection is performed. But should the specified time have minutes and seconds not equal to zero, the data is collected for the last and the next full hour. This results in two calls to the FTP-server, and could slow the system down. A buffer could be implemented for faster loading time, but was not done at this point.

The returned operation point Y is calculated from

$$Y = a + (b - a)x \tag{4.1}$$

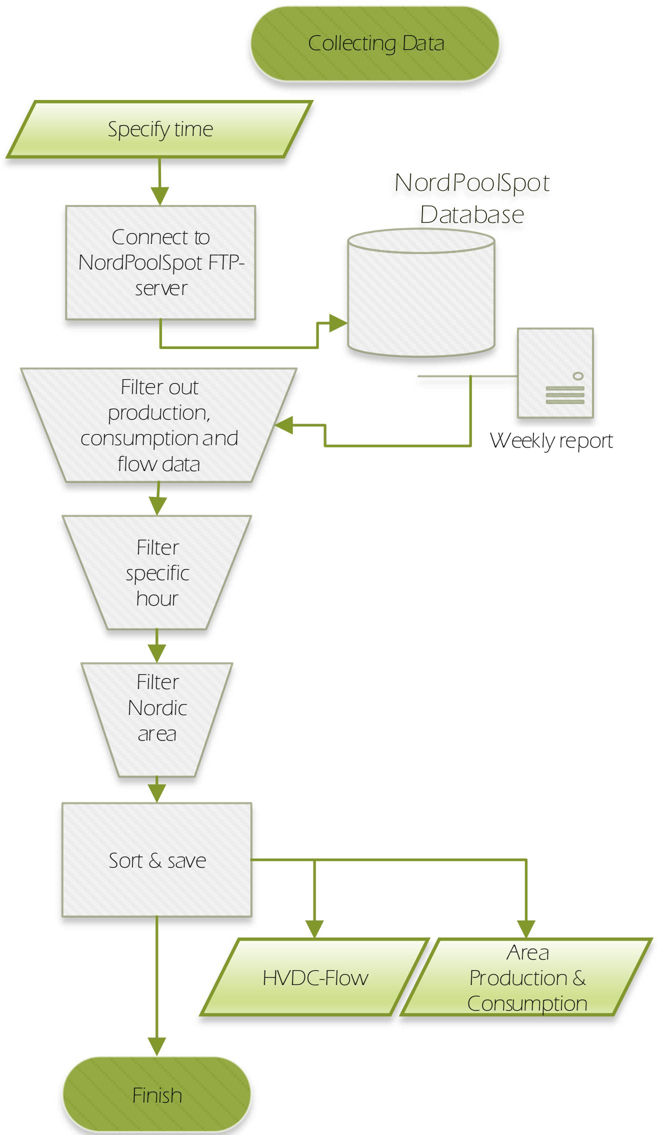

Figure 4.3: Flow chart of the process of collecting data

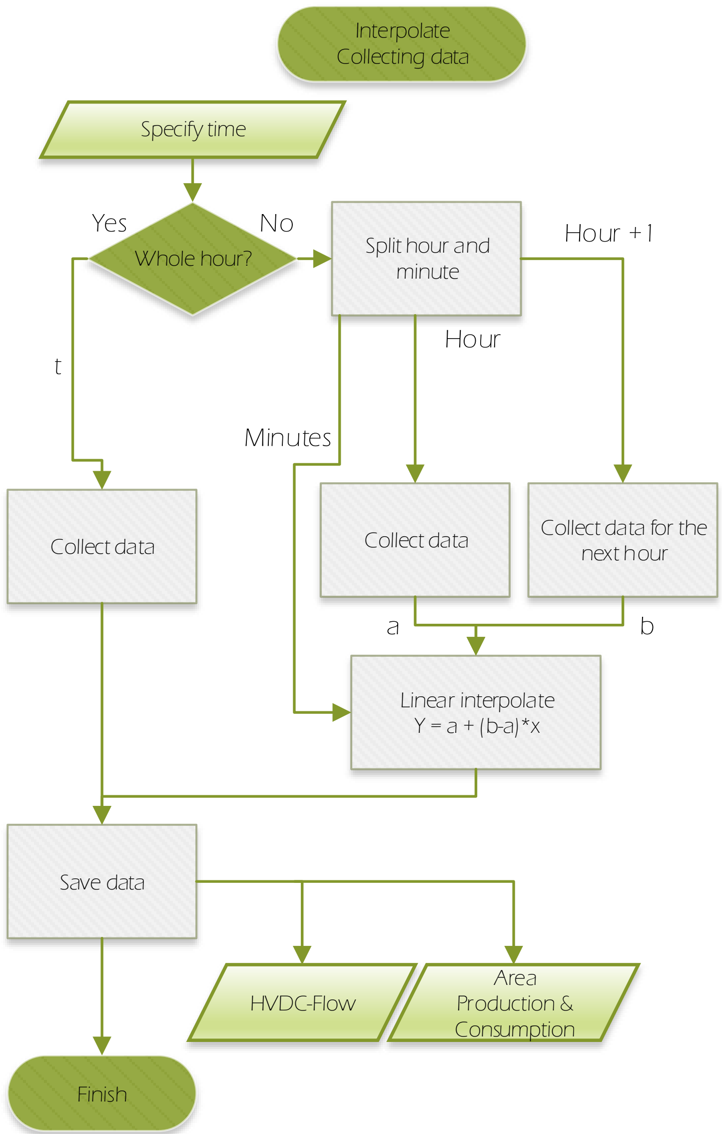

Figure 4.4: Flow chart showing how the interpolation is done to get more smooth transitions between hours.

where a is data from the last full hour, b is the next full hour, and x is the fraction of hour the minutes and seconds correspond to at the specified time. These variables are local variables, and should not be confused with the state space variables presented earlier.

The output from the process of collecting data is printed to two csv-files, one containing the HVDC-flow found in the system at their representative bus number, and one containing the different areas and their total production and consumption. Table 4.1 is used to map the collected data into correct areas according to the Nordic44-model. Table 4.2 maps the HVDC-connections to the correct bus number.

4.5 Producing the PSS/E code

The next step is to populate the model with data. A process of generating the code to be run in PSS/E is therefore needed. The data flow is presented in Figure 4.5 and the source code is found in Appendix A.3. As input to the process, the two files generated in the previous step containing information about the areas production and consumption and the HVDC-flow is needed, also a set of base cases that defines how the collected data should be divided needs to be provided.

The HVDC-flow data points are already divided correctly into corresponding bus and are directly sent to the PSS/E code generator. The production and consumption data are compared to two base cases, one containing information about the loads, and one containing information about the generators.

The production data is compared to a generation base case, which contains generator data such as available generators, real and reactive power limits and short circuit information for each bus. The program determines how many generators needs to be turned on for each bus, and will change the data in the aggregated generator of the model according to needs. The number is chosen such as the power factor of the generators are as high as possible, but less than a specified number, here 1.0, of Pmax. Example: Say a bus has originally four generators at 100 MW max production each. If the bus production is 280 MW, only three generators need to fulfill the production, and one is left unconnected. Should the bus production raise to 310 MW the next hour, the last generator will also be connected, and the parameter for the aggregated generator is changed. The base case and the number of available generators are shown in Table

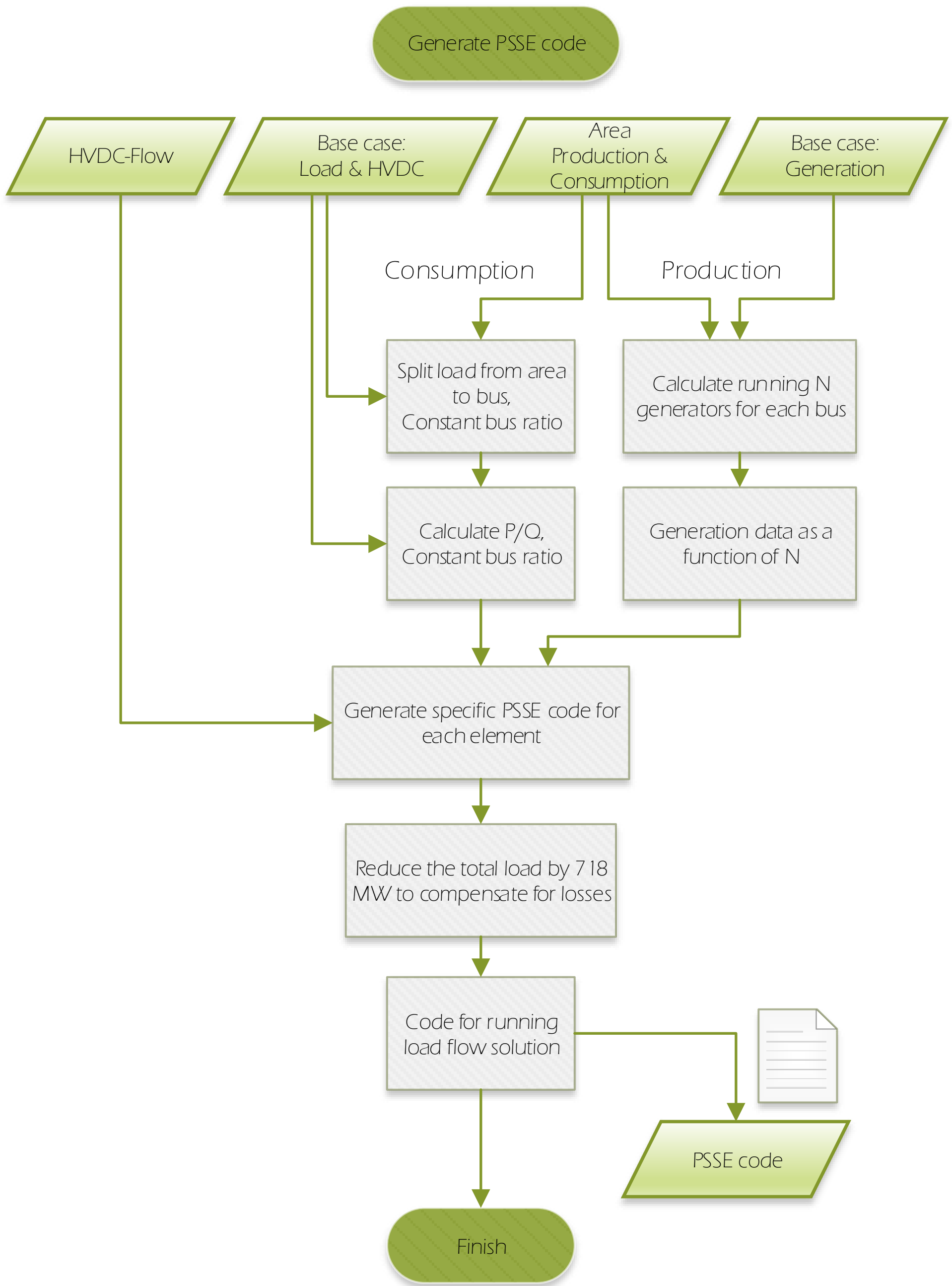

Figure 4.5: Schematics for producing the PSSE code to update the model with new data.

B.1.

The load data is compared to a base case, and the total area consumption is divided and scaled according to the base case. The same P/Q ratio is kept for each load, and the same ratio between loads of the same area is kept constant, as seen in Figure 4.6. This could be looked

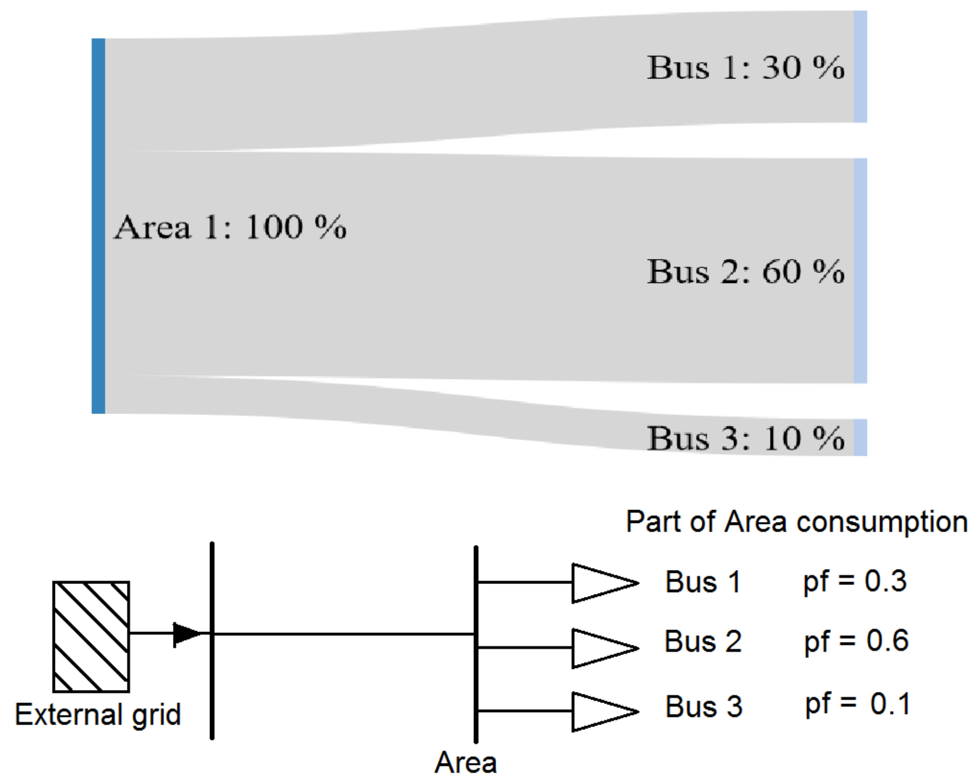

Figure 4.6: The area consumption is divided proportionally into the loads, keeping the P/Q ratio and the bus/area ratio constant.

upon as a simple simplification, however, the data provided does not give more details, guesses have to be made.

When the data for all generators, loads and HVDC-cables have been found, they are sent to a PSS/E code generator shown in Figure 4.7. The generator iterates through each element and produce a PSS/E code to apply the changes to the model. Due to mismatch in the production, consumption and exchange data from NordPoolSpot, all the loads are scaled down with a constant factor of 718 MW. The number is found experimental after tuning the model after 25. march 2015 at 11.00. This may not be correct for all cases, such as high demand and low demand situation, but as for a start this solves the mismatch problem.

4.6 Running the PSS/E code

To initiate PSS/E from Python the following code snippet needs to be run from a Python 2.7 compiler:

```
# Printing code for updating the generators
psspy.machine_chng_2(3000,r"""1""",[_i,_i,_i,_i,_i,_i],[ 2000, _f,...
        1934,-1934,2625.75,0,2600,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f])
psspy.machine_chng_2(3115,r"""1""",[_i,_i,_i,_i,_i,_i],[ 1700, _f,...
        1866,-1866,2000,0,2200,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f])
:
:
#Printing code for updating the loads
psspy.load_chng_4(3100,r"""1""",[_i,_i,_i,_i,_i,_i],[ 621, 110,_f,_f,_f,_f])
psspy.load_chng_4(3115,r"""1""",[_i,_i,_i,_i,_i,_i],[ 621, 650,_f,_f,_f,_f])
:
:
#Printing HVDC, same as normal loads
psspy.load_chng_4(3020,r"""1""",[_i,_i,_i,_i,_i,_i],[ 1219, 616,_f,_f,_f,_f])
psspy.load_chng_4(3360,r"""1""",[_i,_i,_i,_i,_i,_i],[ -330, 262,_f,_f,_f,_f])
:
:
#Printing code for scaling down all loads, in total 718 MW
psspy.scal(0,1,1,[0,0,0,0],[0.0,0.0,0.0,0.0,0.0,0.0,0.0])
psspy.scal(0,1,2,[3,0,5,0],[-718.0,0.0,0.0,0.0,0.0,0.0,0.0])
# Solving power-flow
psspy.fdns([0,0,0,1,1,0,99,0])
```

Figure 4.7: The program will generate PSS/E code that will update the model each time new data is collected.

```
import psspy
psspy.psseinit(1000)
psspy.read(0,r"""Nordic44.raw""")
execfile(r'generated_PSSE_code.py')
```

Here the PSS/E Python package is imported and a blank system is initiated. The nordic44-model is read into memory and the generated code is performed to update the Nordic44-model with new values. The new converged system is saved to a file, that is being continually inspected for updates by PacDyn. The source code is found in Appendix A.4.

4.7 Compute eigenvalues using PacDyn

A unreleased version 9.8.0 of PacDyn is used to perform real-time monitoring of a system. The monitor uses the Dominant Pole Spectrum Eigensolver (DPSE) to track the modes in the system.

The DPSE-method uses spesified transfer functions to find the most dominant poles of each function. Initiating the system is performed to find the needed transfer functions.

4.7.1 Initiate PacDyn

PacDyn requires a system data file (.raw) to be monitored (Appendix C.1), a system dynamic file (.dyn) (Appendix B.3) and a contingency list (.ctg) (Appendix B.4), where the contingency list is voluntary.

To be able to run the monitoring feature of PacDyn, the system need to be set up to track a set of one or more transfer functions. To determine which tranfer functions that should be tracked, some initial setup needs to be conducted. One way of initiating the system is to perform a QR-method to find the eigenvalues of interest. Observability factor and controllability factor could be used to determine the transfer functions to be observed. For each mode the input value V_{ref} should have a high controllability factor, and the output value ω should have a high observability factor. Using this method could result in a transfer function having different generators as input and output. In theoretical cases this could be easily implemented.

On the other hand, often it is preferred to have the input and output variables from the same generator. This could be done by checking the residues, a multiplication of the observability factor and the controllability factor, to find the generator where the mode is easily observed. The latter was used here, using the residues. The DPSE method does not converge to the same mode if only one transfer function is chosed, so for this simulation the input and output is from the generator at bus 7000 as shown in Table 4.3.

Mode	Start shift	TF Input V_{ref}	TF Output ω	TF
1	-0.3750+j3.7519	7000	7000	1
2	-0.1021+j2.040	7000	7000	1
3	-0.7004+j5.9419	7000	7000	1
4	-0.4955+j5.3423	7000	7000	1

Table 4.3: Start shift at 05.March 2015 11.00.00, four modes and two transfer functions

4.8 Perform simulations

A set of simulations is perform to present the on-line DSA tool. The following simulations are performed:

4.8.1 A week simulation, 4 modes

Start time: March 5. 2015 11:00:00

End time: March 12. 2015 11:00:00

Increment time: 0:15 hour

Contingencies: None

Description: The four modes shown in Table 4.3 are being monitored for one week. **DPSE-method** is used for tracking.

4.8.2 A week simulation, 4 modes with contingencies

Start time: March 5. 2015 11:00:00

End time: March 12. 2015 11:00:00

Increment time: 0:15 hour

Contingencies: 4 contingencies; Line trip between Bus 5100-6500, line trip between Bus 5301-5304, line trip between Area SE1-FI, line trip between Area NO1-SE3

Description: The four modes shown in Table 4.3 on the last page are being monitored for one week. **DPSE-method** is used for tracking.

4.8.3 High damped situation of mode 2

Time: March 6. 2015 03:00:00

Contingencies: None

Description: During the week simulation above, Mode 2 appeared to be the most critical mode. A high damped situation will be compared towards a low damped situation.**QR-method**, **mode shape**, **Generation Hopf Bifurcation** and **Generation sensitivity** is performed. In Generation Hopf Bifurcation, mode two is moved from 15.49 % to 10 %. Operational data found in Table 4.4, Table 4.5 and Table 4.6. The simulation is done to get some qualitative insight into the mode behaviour of mode 2.

System data: Specific data produced by the on-line DSA, containing information about production units data, loads data and flow data is shown in Table C.4, Table C.6 and Table C.5 in the Appendix.

4.8.4 Low damped situation, mode 2

Time March 6. 2015 03:00:00

Contingencies: None

Description: During the week simulation above, Mode 2 appeared to be the most critical mode. A high damped situation will be compared towards a low damped situation. **QR-method**, **mode shape**, **Generation Hopf Bifurcation** and **Generation sensitivity** is performed. Operational data found in Table 4.4, Table 4.5 and Table 4.6. The simulation is done to get some qualitative insight into the mode behaviour of mode 2.

System data: Specific data produced by the on-line DSA, containing information about production units data, loads data and flow data is shown in Table C.1, Table C.3 and Table C.2 in the Appendix.

4.8.5 Year simulation, 2 modes

Start time: January 1. 2015 00:00:00

End time: January 1. 2016 00:00:00

Increment time: 1:00 hour

Contingencies: None

Description: Mode 1 and Mode 2 presented in Table 4.3 is tracked. Even though only Mode 2 is of interest, Mode 1 is tracked to prevent Mode 2 in converging into Mode 1 and losing track. Simulation is performed to get seasonal variations and see if the sofware will be able to stay intack and not loose track of mode 2.

4.9 Data

Production data, consumption data, and HVDC-flow for the first day of the week simulation is shown in Table 4.4, 4.5 and 4.6, and collected from NordPoolSpot [26].

Time \Area →	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4	FI
2015-03-05 11:00:00	2104	8884	2443	3506	6151	3748	6599	9622	754	8659
2015-03-05 12:00:00	2110	8732	2251	3260	6129	3723	6695	9213	744	8359
2015-03-05 13:00:00	2119	8689	2197	3092	6170	3724	6660	9180	722	8226
2015-03-05 14:00:00	2109	8699	2303	3173	6059	3721	6792	9216	729	8479
2015-03-05 15:00:00	2099	8644	2307	3121	6070	3802	7001	9265	726	8573
2015-03-05 16:00:00	2099	8674	2282	3013	6085	3921	7062	9303	716	8652
2015-03-05 17:00:00	2102	8721	2319	3071	6157	3816	7046	9289	711	8593
2015-03-05 18:00:00	2119	8921	2338	3078	6255	3882	7281	9622	737	8766
2015-03-05 19:00:00	2086	8852	2205	2949	6160	3794	7090	9628	731	8606
2015-03-05 20:00:00	2064	8370	1995	2871	6173	3684	6789	9670	776	8155
2015-03-05 21:00:00	2067	7701	1879	2804	5874	3094	6600	9769	800	8023
2015-03-05 22:00:00	2049	7027	1751	2608	5633	2422	6445	9810	872	7888
2015-03-05 23:00:00	1981	6258	1622	2362	4635	1962	6135	9894	918	7788
2015-03-06 00:00:00	1858	5734	1541	2353	3924	1614	5460	10135	960	7731
2015-03-06 01:00:00	1792	5383	1399	2224	3604	1502	5164	10176	1012	7717
2015-03-06 02:00:00	1774	5290	1407	2151	3342	1436	4971	10130	1057	7701
2015-03-06 03:00:00	1781	5343	1404	2125	3329	1407	4858	10101	1096	7731
2015-03-06 04:00:00	1815	5637	1462	2334	3623	1557	4887	10169	1116	7871
2015-03-06 05:00:00	1840	5958	1552	2612	4333	2136	5422	10123	1147	8060
2015-03-06 06:00:00	2005	7079	1851	2810	5290	2868	6434	10049	1168	8659
2015-03-06 07:00:00	2106	8672	2123	3027	6044	3555	6939	9991	1186	9000
2015-03-06 08:00:00	2117	8996	2147	3119	6320	3791	6976	9997	1193	9041
2015-03-06 09:00:00	2124	8914	2206	3101	6339	3694	7000	9854	1186	9004
2015-03-06 10:00:00	2107	8746	2142	3062	6204	3722	7106	9773	1210	8906
2015-03-06 11:00:00	2109	8541	2105	3037	6156	3692	7120	9668	1200	8867

Table 4.4: Production data for the first day of a week simulation [MW]

Time \Area →	NO1	NO2	NO3	NO4	NO5	SE1	SE2	SE3	SE4	FI
2015-03-05 11:00:00	5561	4899	3039	2489	2651	1242	2265	12540	3720	10831
2015-03-05 12:00:00	5452	4918	3024	2406	2588	1210	2305	12310	3578	10671
2015-03-05 13:00:00	5430	4857	3036	2416	2615	1192	2138	11927	3488	10647
2015-03-05 14:00:00	5462	4949	3067	2459	2604	1151	2059	11829	3408	10689
2015-03-05 15:00:00	5515	4900	3044	2460	2644	1142	2042	11760	3391	10614
2015-03-05 16:00:00	5525	4908	3053	2426	2634	1129	2000	11792	3394	10602
2015-03-05 17:00:00	5532	4878	3058	2447	2611	1139	1906	12225	3506	11036
2015-03-05 18:00:00	5661	4985	3076	2434	2599	1117	2032	12749	3616	11198
2015-03-05 19:00:00	5611	4997	3060	2433	2596	1044	2029	12420	3547	11148
2015-03-05 20:00:00	5458	4836	2981	2408	2542	1032	2015	11987	3434	10657
2015-03-05 21:00:00	5245	4729	2939	2416	2469	973	2026	11385	3221	10652
2015-03-05 22:00:00	4934	4536	2872	2348	2544	1064	1992	10576	3063	10403
2015-03-05 23:00:00	4568	4262	2736	2253	2335	1021	1907	9889	2845	9903
2015-03-06 00:00:00	4328	4092	2626	2208	2143	1055	1959	9809	2629	9552
2015-03-06 01:00:00	4141	3965	2569	2224	2106	1043	1922	9606	2543	9320
2015-03-06 02:00:00	4073	4005	2608	2182	2064	996	1934	9481	2543	9240
2015-03-06 03:00:00	4070	4020	2617	2163	2061	948	1930	9574	2569	9303
2015-03-06 04:00:00	4134	4033	2602	2182	2097	1043	2040	9829	2568	9688
2015-03-06 05:00:00	4320	4143	2650	2244	2150	1055	2044	10348	2781	10581
2015-03-06 06:00:00	4914	4541	2794	2298	2277	1076	2177	11670	3166	10849
2015-03-06 07:00:00	5561	4998	3013	2467	2498	1177	2311	12377	3501	11040
2015-03-06 08:00:00	5654	5054	3036	2581	2549	1174	2319	12663	3620	11134
2015-03-06 09:00:00	5510	4993	3100	2546	2569	1156	2389	12572	3674	11107
2015-03-06 10:00:00	5243	4907	2938	2531	2532	1182	2459	12635	3678	11025
2015-03-06 11:00:00	5065	4866	2883	2555	2525	1114	2450	12574	3635	11104

Table 4.5: Consumption data for the first day of a week simulation [MW]

Time \Bus nr →	3360	5610	5620	7010	7020	8600	8700
2015-03-05 11:00:00	-330	1412	414	-1219	343	546	628
2015-03-05 12:00:00	-336	1457	414	-1220	25	467	601
2015-03-05 13:00:00	-204	1581	413	-1220	-23	410	717
2015-03-05 14:00:00	12	1443	414	-1220	377	511	720
2015-03-05 15:00:00	66	1461	413	-1220	546	480	895
2015-03-05 16:00:00	147	1434	413	-1220	455	573	1080
2015-03-05 17:00:00	-95	1465	412	-1220	-35	858	641
2015-03-05 18:00:00	213	1395	413	-1030	-115	969	515
2015-03-05 19:00:00	139	1476	413	-960	-490	703	699
2015-03-05 20:00:00	125	1385	414	-960	-406	605	977
2015-03-05 21:00:00	-317	1250	414	-960	-585	615	1173
2015-03-05 22:00:00	-335	1017	413	-940	-583	520	1131
2015-03-05 23:00:00	-331	820	413	-507	-525	314	1016
2015-03-06 00:00:00	-254	483	413	-507	-447	427	295
2015-03-06 01:00:00	-111	187	413	-508	-471	381	137
2015-03-06 02:00:00	-123	82	412	-507	-511	232	66
2015-03-06 03:00:00	-195	151	413	-507	-563	179	-3
2015-03-06 04:00:00	-310	241	412	-522	-250	156	85
2015-03-06 05:00:00	-265	379	412	-1214	-118	53	381
2015-03-06 06:00:00	-337	828	413	-1220	263	485	757
2015-03-06 07:00:00	-336	1269	414	-1220	527	642	1082
2015-03-06 08:00:00	-208	1535	413	-1221	483	643	1014
2015-03-06 09:00:00	-218	1571	413	-1220	473	584	944
2015-03-06 10:00:00	-72	1566	413	-1220	401	575	894
2015-03-06 11:00:00	-82	1467	415	-1220	420	573	926

Table 4.6: HVDC-flow for the first day of a week simulation [MW]

Chapter 5

Results

5.1 A week simulation, 4 modes

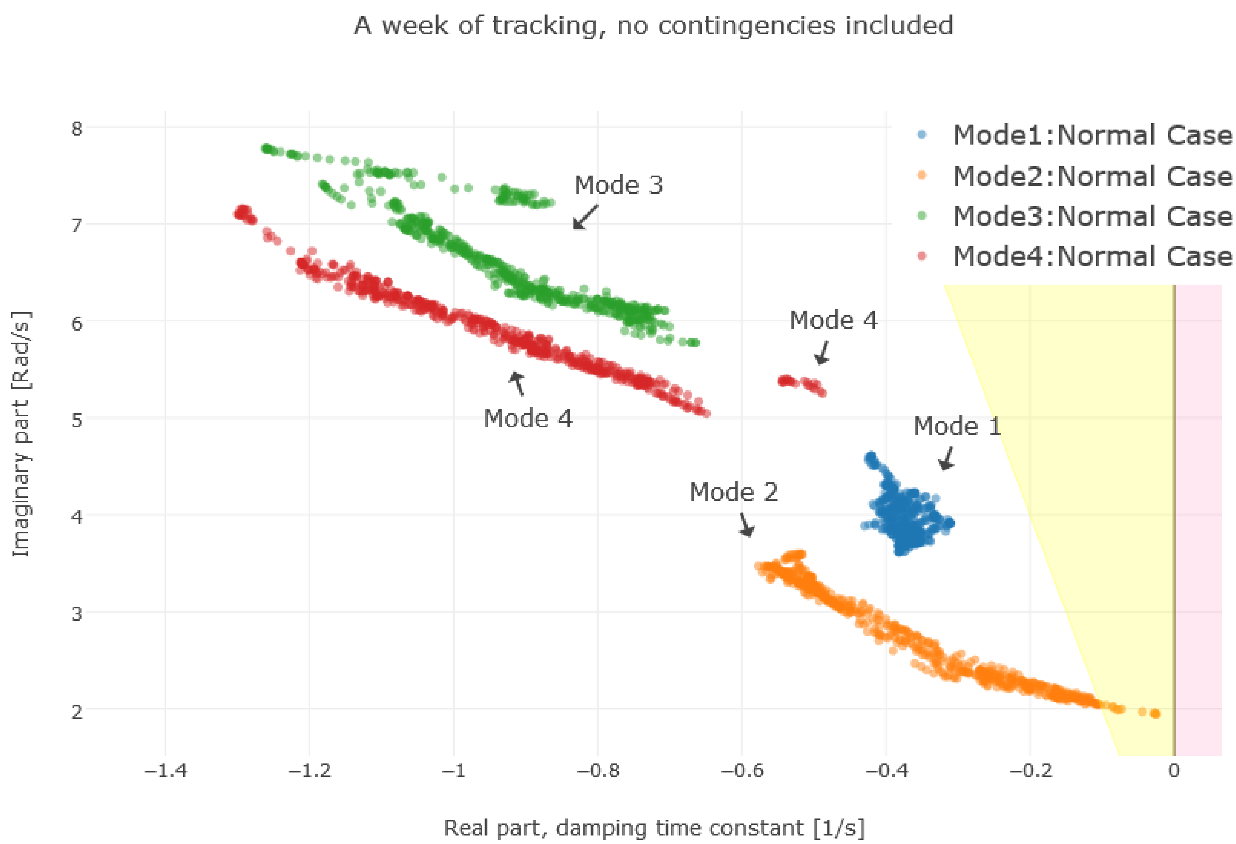

Figure 5.1: Results of tracking 4 modes for a week in two-axis plot. Eigenvalues with positive imaginary axis plotted. The yellow area is the insecure area, and the red area relates to unstable area.

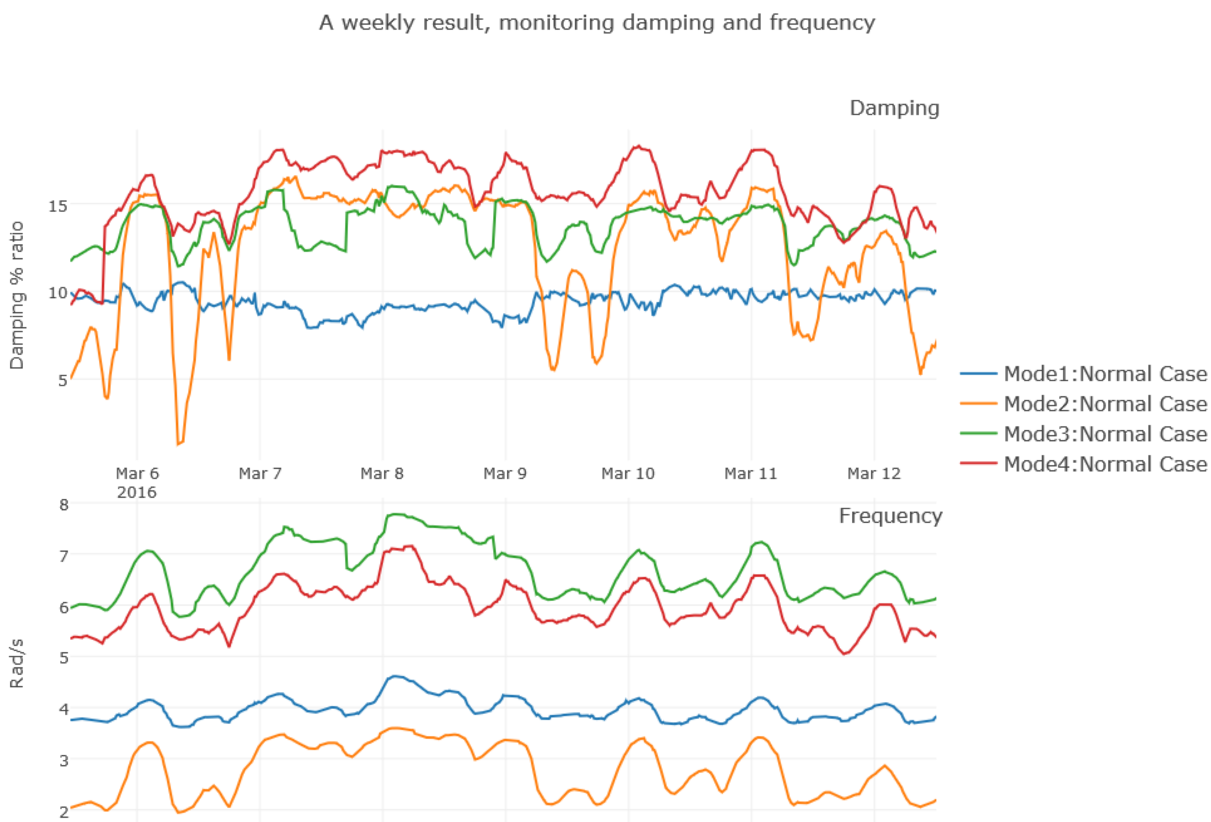

Figure 5.2: Frequency and damping ratio of 4 modes monitored for a week. All modes should have a damping factor higher than 5 % to remain in the secure state.

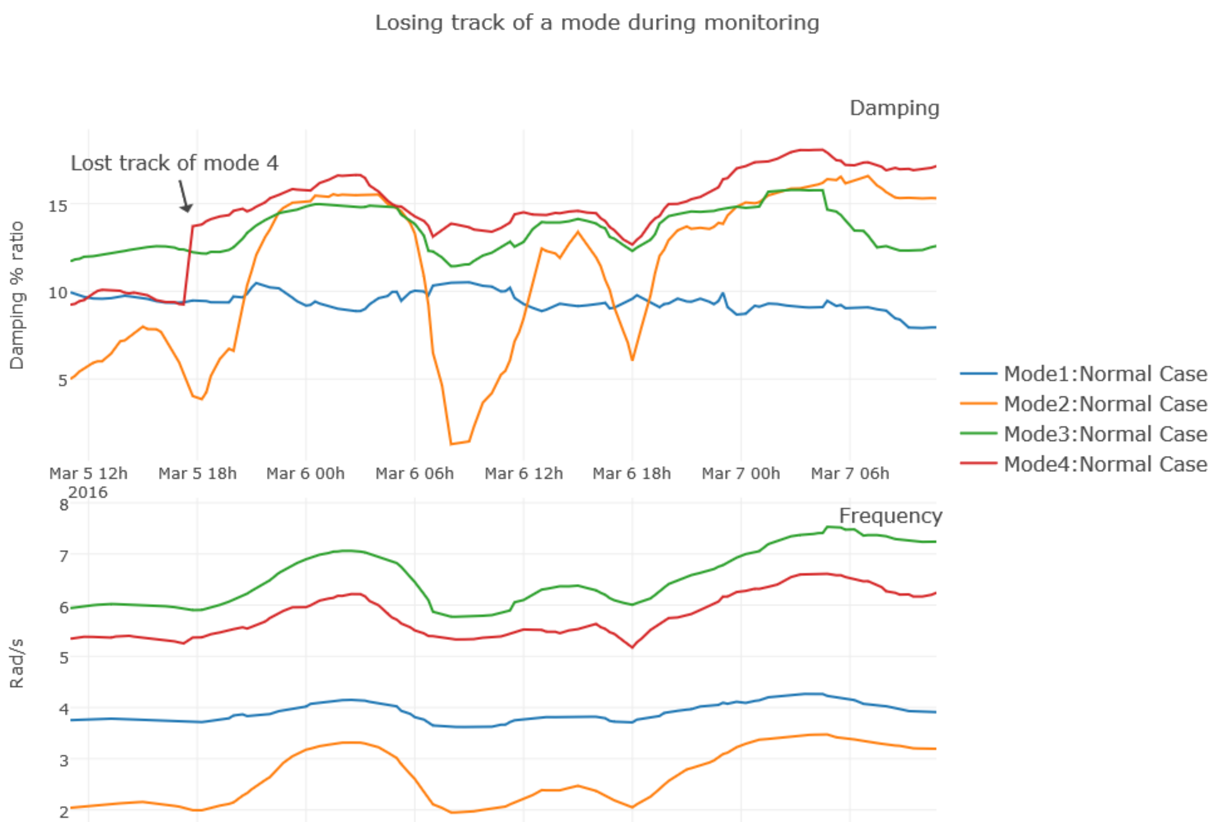

Figure 5.3: Zooming into the first two days of the simulations reveals that Mode 4, starting at (-0.4955+j5.3423), loses track at March 5, 18:00.

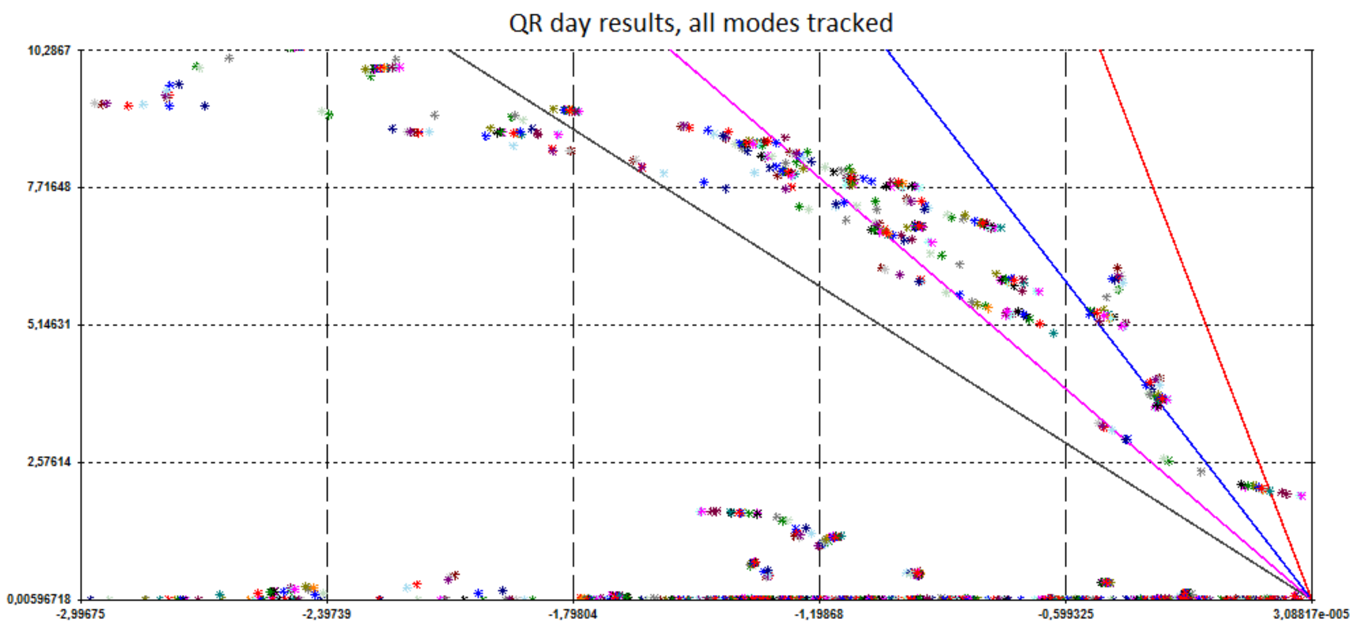

Figure 5.4: QR-method for the first day of the weekly simulation, showing all eigenvalues, even those not being tracked.

5.2 A week simulation, 4 modes with contingencies

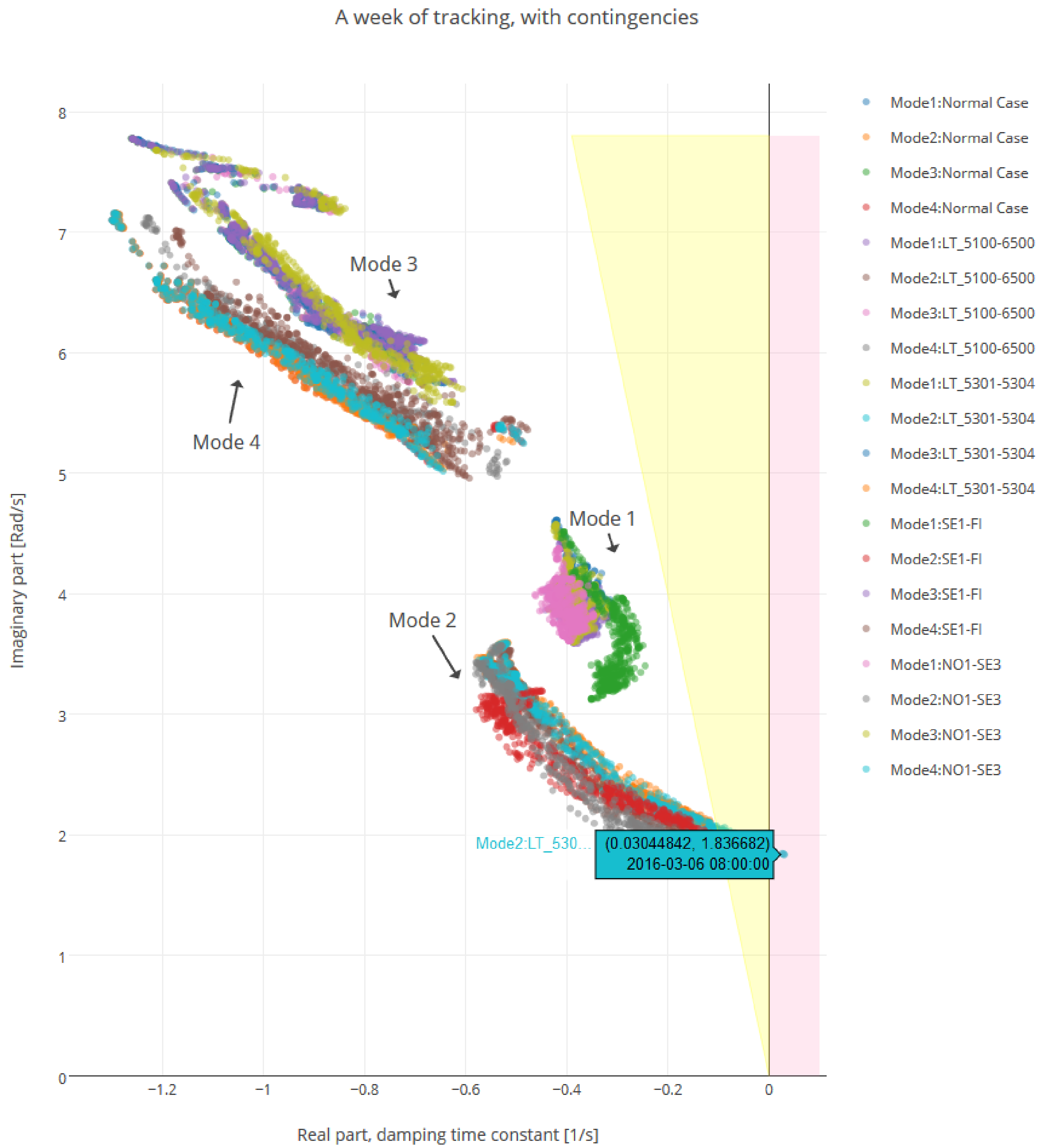

Figure 5.5: Two-axis plot of 4 modes and 4 contingencies applied.

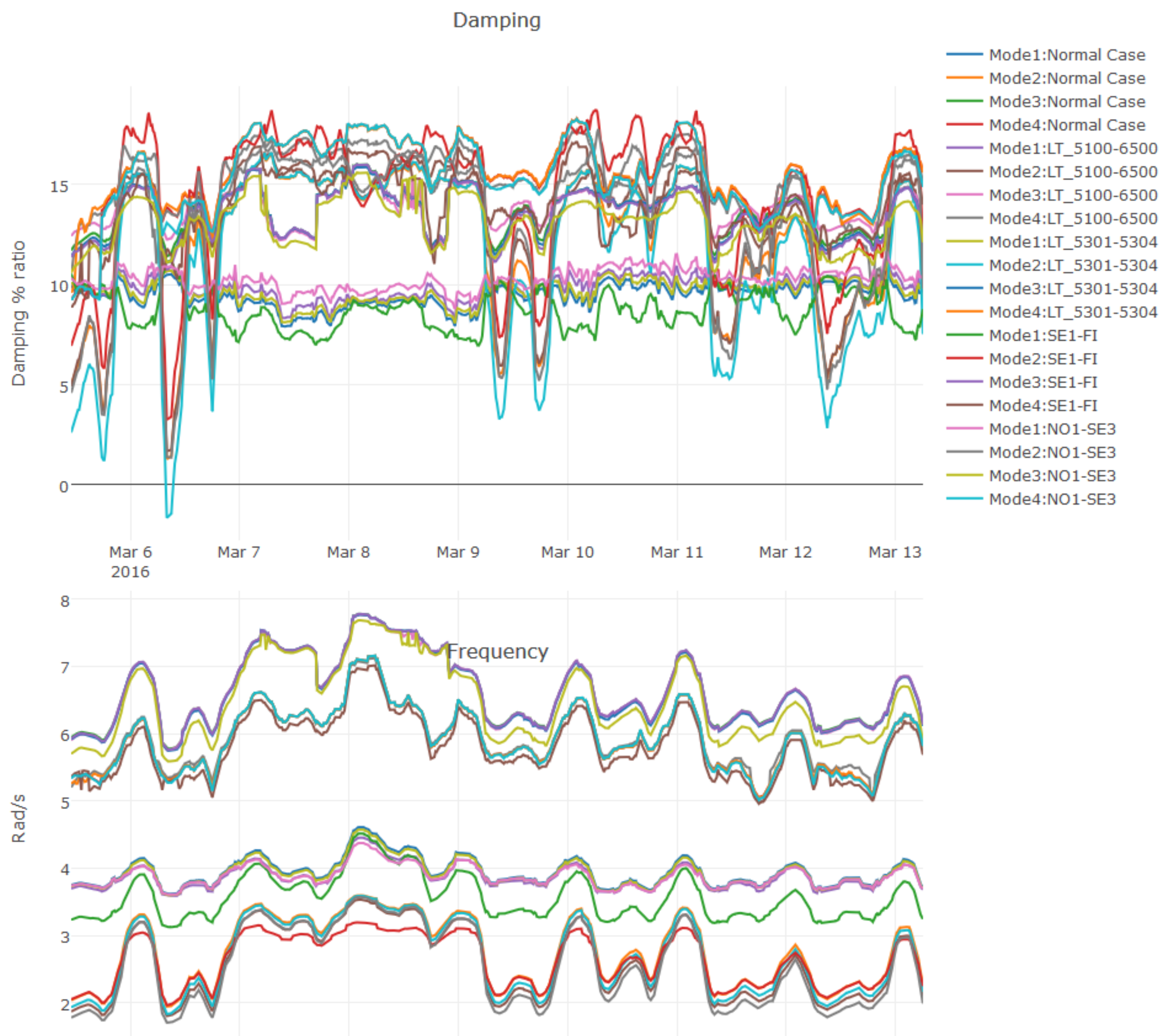

Figure 5.6: Time line plot of 4 modes and 4 contingencies applied.

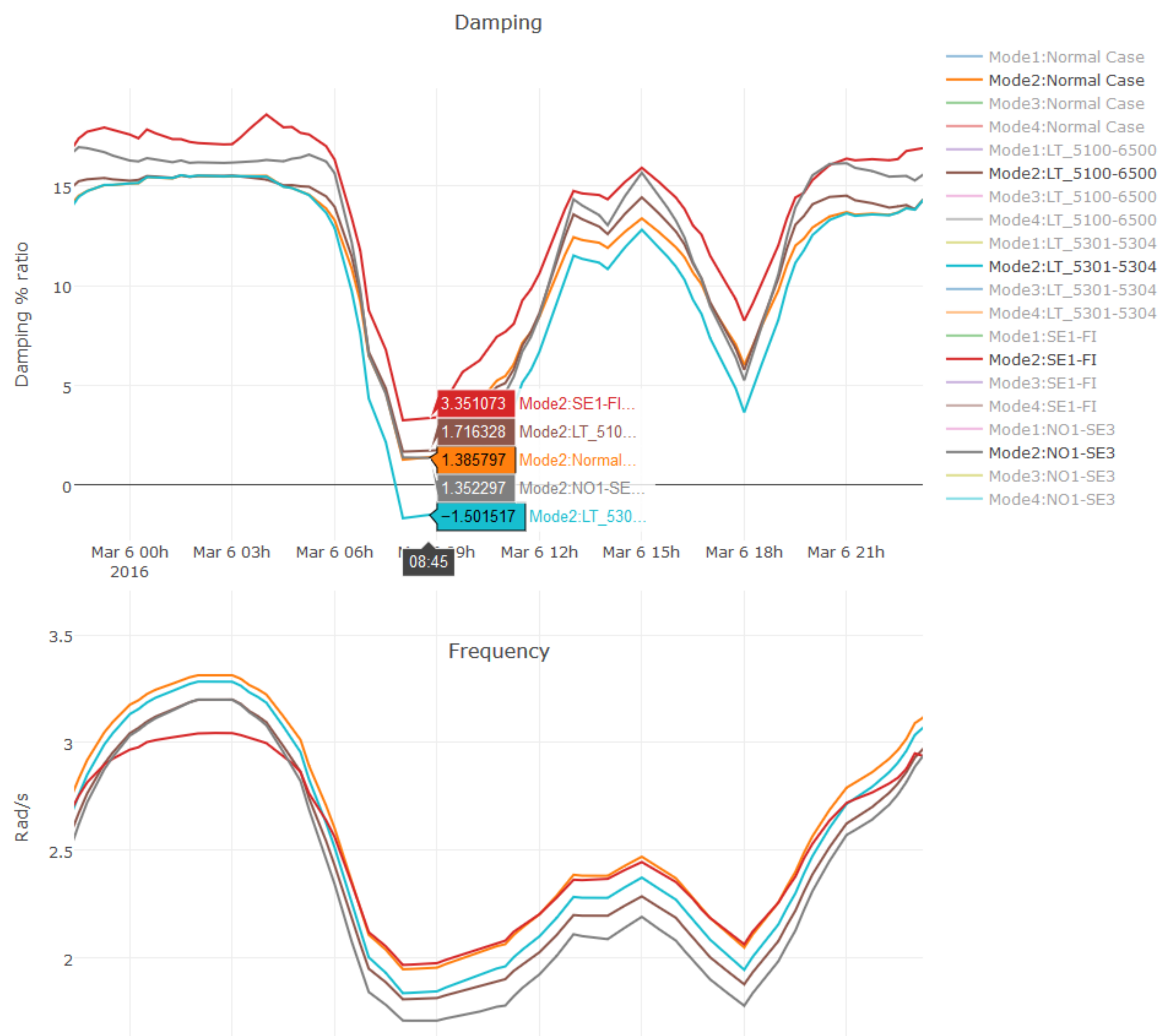

Figure 5.7: Time line plot of 4 modes and 4 contingencies applied. The contingency lines shows where the system would be, if the line trips.

5.3 High damped situation of mode 2

5.3.1 QR-results

#	Real	Imaginary	Module	Freq. (Hz)	Damp(%)	Part. Factor	Mode
1	-0,4738	6,2121	6,2301	0,9887	7,6047	WW Generator # 3359 1	
2	-0,3697	4,1494	4,1658	0,6604	8,8737	DELT Generator # 7000 1	1
3	-0,9857	7,5090	7,5734	1,1951	13,015	WW Generator # 3300 1	
4	-1,0539	7,0454	7,1238	1,1213	14,794	DELT Generator # 6100 1	3
5	-1,2724	8,3831	8,4792	1,3342	15,007	WW Generator # 3000 1	
6	-1,3338	8,5808	8,6838	1,3657	15,360	WW Generator # 8500 1	
7	-0,5193	3,3125	3,3530	0,5272	15,489	DELT Generator # 7000 1	2
8	-1,2998	7,9451	8,0507	1,2645	16,145	DELT Generator # 3115 1	
9	-1,4266	8,6475	8,7644	1,3763	16,277	DELT Generator # 3249 1	
10	-1,0476	6,2147	6,3024	0,9891	16,623	DELT Generator # 6700 1	4
11	-1,5358	8,8741	9,0061	1,4124	17,053	DELT Generator # 6500 1	
12	-1,6519	8,2347	8,3988	1,3106	19,669	WW Generator # 5300 1	
13	-1,8036	8,4101	8,6013	1,3385	20,969	DELT Generator # 6000 1	
14	-2,1848	8,7526	9,0212	1,3930	24,218	DELT Generator # 5500 1	
15	-2,8084	10,815	11,174	1,7213	25,134	DELT Generator # 7100 1	
16	-2,7871	9,4082	9,8124	1,4974	28,404	DELT Generator # 5100 1	
17	-2,9450	9,2803	9,7363	1,4770	30,247	DELT Generator # 5400 1	

Table 5.1: QR-results with most dominant participation factor for the electromechanical modes at March 6. 2015 03:00:00

5.3.2 Mode-shapes

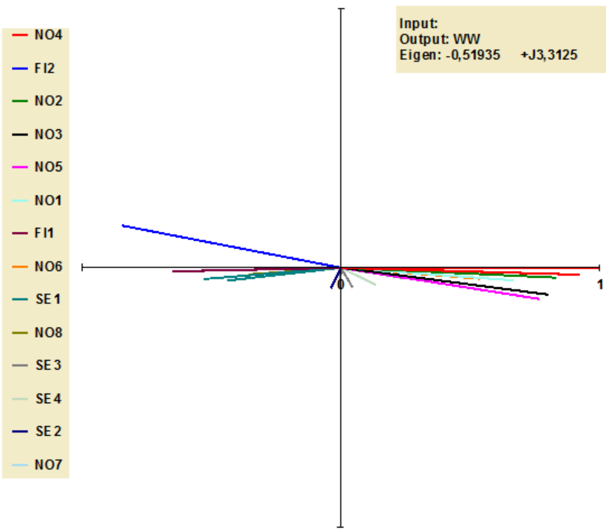

Figure 5.8: Mode shape of mode 2 in high damped situation, Southern Norway oscillates against Finland

Module	Phase	Bus Name	Area
1,0000	0	Generator # 6100 1	NO4
0,9169	-1,6724	Generator # 6000 1	NO4
0,8585	169,13	Generator # 7000 1	FI2
0,8275	-2,6385	Generator # 5400 1	NO2
0,8030	-7,3889	Generator # 5600 1	NO3
0,7709	-8,9500	Generator # 5300 1	NO5
0,6660	-4,1891	Generator # 5500 1	NO1
0,6489	181,25	Generator # 7100 1	FI1
0,5370	-4,4879	Generator # 5100 1	NO6
0,5319	184,73	Generator # 3249 1	SE1
0,4383	186,45	Generator # 3115 1	SE1
0,3540	184,14	Generator # 6700 1	NO8
0,2557	-9,4543	Generator # 3359 1	SE3
0,1452	-26,252	Generator # 8500 1	SE4
0,08884	-59,631	Generator # 3300 1	SE3
0,08462	-115,94	Generator # 3245 1	SE2
0,07959	-91,685	Generator # 3000 1	SE3
0,07106	-94,238	Generator # 6500 1	NO7

Table 5.2: Eigenvalue -0.51935 +j 3.3125

5.3.3 Generation Hopf Bifurcation

Generator	Original	Changed	New
# 7000	6283.7	876.47	7160.2
# 6100	2844.7	875.29	3720
# 5300	3329	847.68	4176.7
# 7100	1446.4	220.29	1666.7
# 6000	314.54	195.46	510
# 5500	958.22	185.46	1143.7
# 3249	768.82	116.54	885.36
# 3115	638.18	95.15	733.33
# 5600	1066.9	72.476	1139.4
# 5400	1117.4	57.997	1175.4
# 5100	822.78	52.719	875.5
# 6700	2125	-184.18	1940.8
# 3359	5668.8	-269.27	5399.5
# 3300	1904.5	-387.93	1516.5
# 6500	1403.4	-529.51	873.92
# 8500	1096	-602.54	493.46
# 3000	2099.6	-677.06	1422.5
# 3245	4858	-686.51	4171.5

Table 5.3: Proposed solution from Generation Hopf Bifurcation to move mode 2 into a more insecure condition

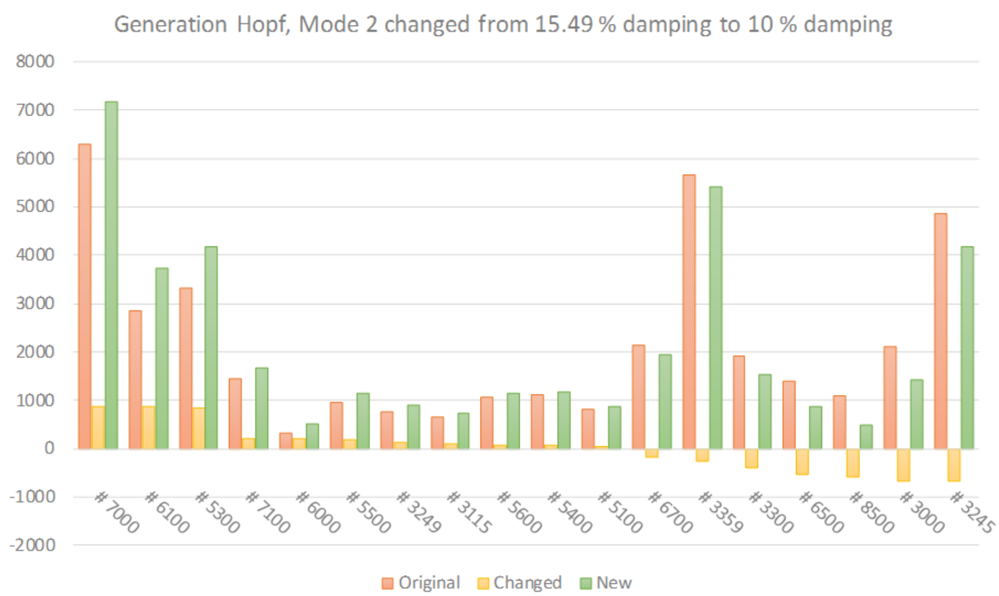

Figure 5.9: Results of moving mode 2 into a less damped situation

5.4 Low damped situation of mode 2

5.4.1 QR-results

#	Real	Imaginary	Module	Freq. (Hz)	Damp(%)	Part. Factor	Mode
1	-0.02495	1.9476	1.9478	0.31	1.2808	EQ' Generator # 5300 1	2
2	-0.4615	5.1195	5.1403	0.8148	8.9787	DELT Generator # 5600 1	
3	-0.3819	3.6212	3.6413	0.5763	10.487	DELT Generator # 7000 1	1
4	-0.772	6.9695	7.0121	1.1092	11.009	WW Generator # 3359 1	
5	-0.6641	5.7736	5.8117	0.9189	11.427	DELT Generator # 3245 1	3
6	-1.0246	7.7386	7.8062	1.2316	13.126	DELT Generator # 6500 1	
7	-0.9495	6.98	7.0443	1.1109	13.479	DELT Generator # 3249 1	
8	-0.9252	6.6871	6.7507	1.0643	13.705	WW Generator # 5300 1	
9	-0.7459	5.3308	5.3827	0.8484	13.858	DELT Generator # 6700 1	4
10	-1.1261	7.8991	7.979	1.2572	14.114	DELT Generator # 3115 1	
11	-1.1174	7.7856	7.8654	1.2391	14.206	WW Generator # 8500 1	
12	-1.3246	8.3122	8.4171	1.3229	15.738	WW Generator # 3000 1	
13	-1.7845	9.1449	9.3174	1.4555	19.152	DELT Generator # 5100 1	
14	-1.836	8.7075	8.899	1.3858	20.632	DELT Generator # 5600 1	
15	-2.3283	10.561	10.814	1.6808	21.53	DELT Generator # 5500 1	
16	-2.2205	9.9688	10.213	1.5866	21.742	DELT Generator # 6000 1	
17	-2.77	10.832	11.181	1.724	24.775	DELT Generator # 7100 1	

Table 5.4: QR-results with most dominant participation factor for the electro-mechanical modes at March 6. 2015 08:00:00

5.4.2 Mode-shapes

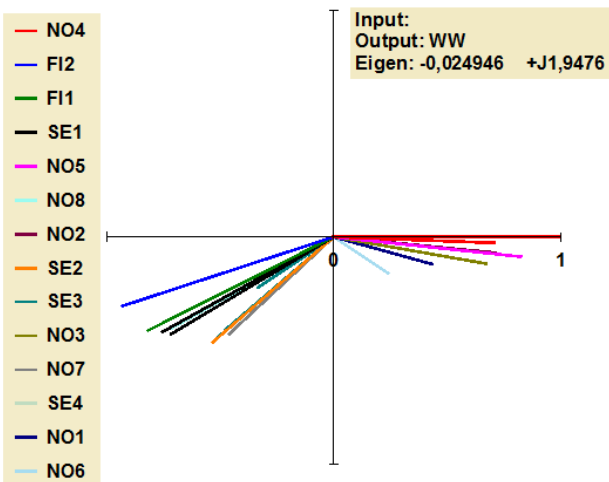

Figure 5.10: Mode shape of mode 2 in low damped situation, Southern Norway oscillates against Finland

Module	Phase	Bus Name	Area
1,0000	0	Generator # 6100 1	NO4
0,9788	-161,87	Generator # 7000 1	FI2
0,9147	-153,25	Generator # 7100 1	FI1
0,8690	-150,98	Generator # 3249 1	SE1
0,8331	-149,11	Generator # 3115 1	SE1
0,8273	-6,0464	Generator # 5300 1	NO5
0,8220	-150,59	Generator # 6700 1	NO8
0,7171	-5,6217	Generator # 5400 1	NO2
0,7081	-2,1300	Generator # 6000 1	NO4
0,7076	-138,93	Generator # 3245 1	SE2
0,6961	-139,11	Generator # 3000 1	SE3
0,6811	-9,9517	Generator # 5600 1	NO3
0,6380	-139,37	Generator # 3300 1	SE3
0,6285	-137,12	Generator # 6500 1	NO7
0,5546	-137,14	Generator # 8500 1	SE4
0,4489	-15,501	Generator # 5500 1	NO1
0,4004	-146,26	Generator # 3359 1	SE3
0,2865	-33,753	Generator # 5100 1	NO6

Table 5.5: Eigenvalue -0.024946 +j 1.9476

5.4.3 Generation Hopf Bifurcation

Generator	Original	Changed	New
# 7000	7348.5	25.24	7373.7
# 3249	2071.5	25.026	2096.5
# 7100	1691.5	24.988	1716.5
# 6700	3119	24.897	3143.9
# 3115	1719.5	24.834	1744.3
# 6500	2146.1	24.658	2170.8
# 3359	5610.5	24.521	5635
# 8500	1193	24.499	1217.5
# 3000	2077.9	23.586	2101.5
# 3245	6976	23.319	6999.3
# 5100	978.01	11.447	989.46
# 3300	2526.4	4.3164	2530.8
# 5500	1139	-0.9123	1138.1
# 5600	1796.4	-33.452	1762.9
# 5400	1881.4	-36.782	1844.6
# 6000	529.59	-38.527	491.06
# 6100	4789.6	-99.915	4689.7
# 5300	6320	-101.25	6218.8

Table 5.6: Proposed solution from Generation Hopf Bifurcation to move mode 2 into a more insecure condition

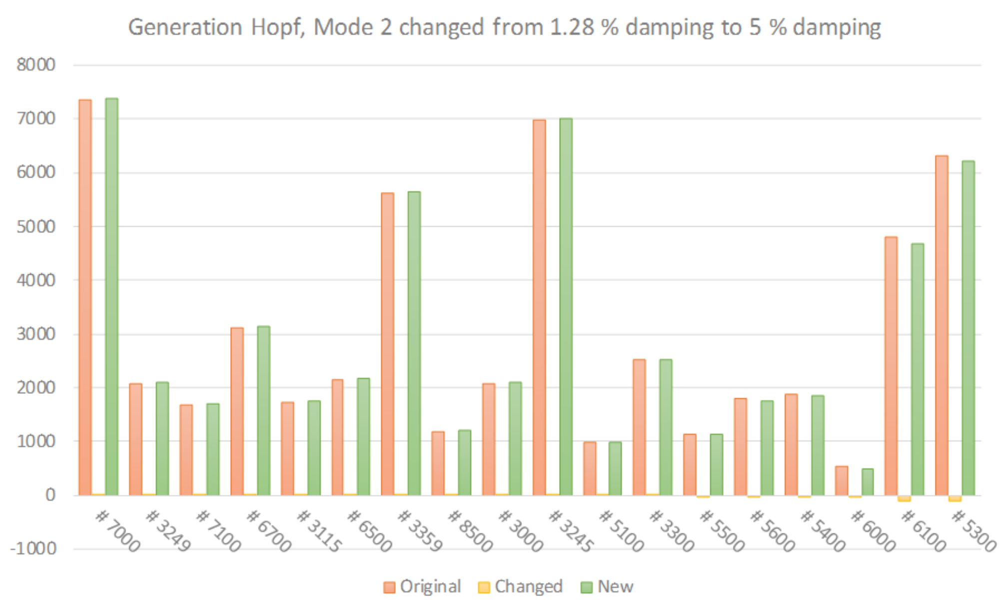

Figure 5.11: Results of moving mode 2 from a bad damped situation into a well damped situation

5.5 Year simulation, 2 modes

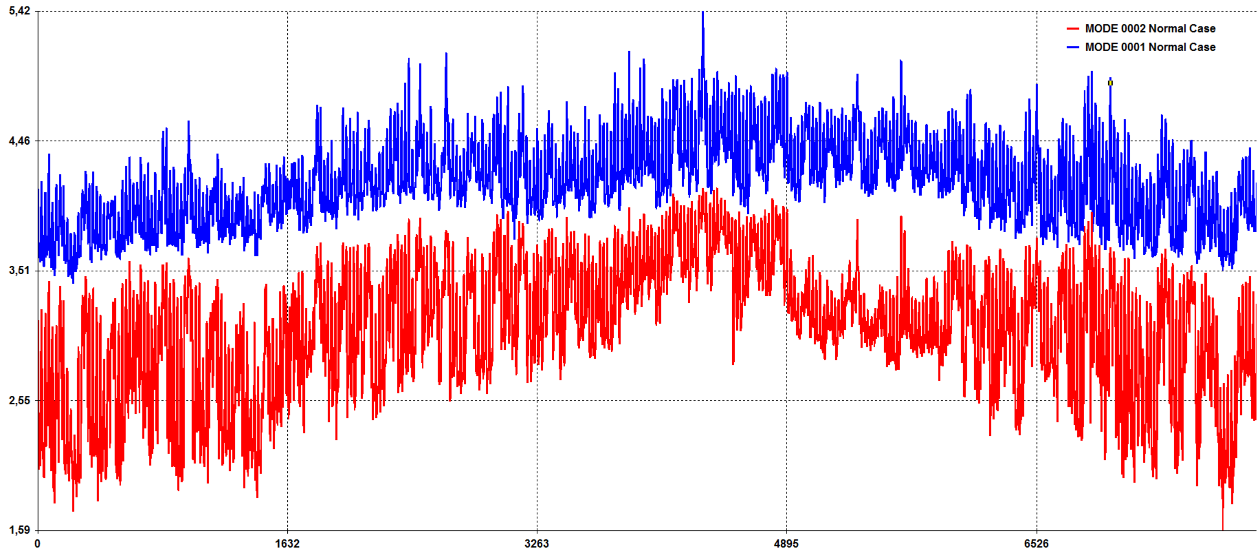

Figure 5.12: Frequency of mode 1 and 2 for a year.

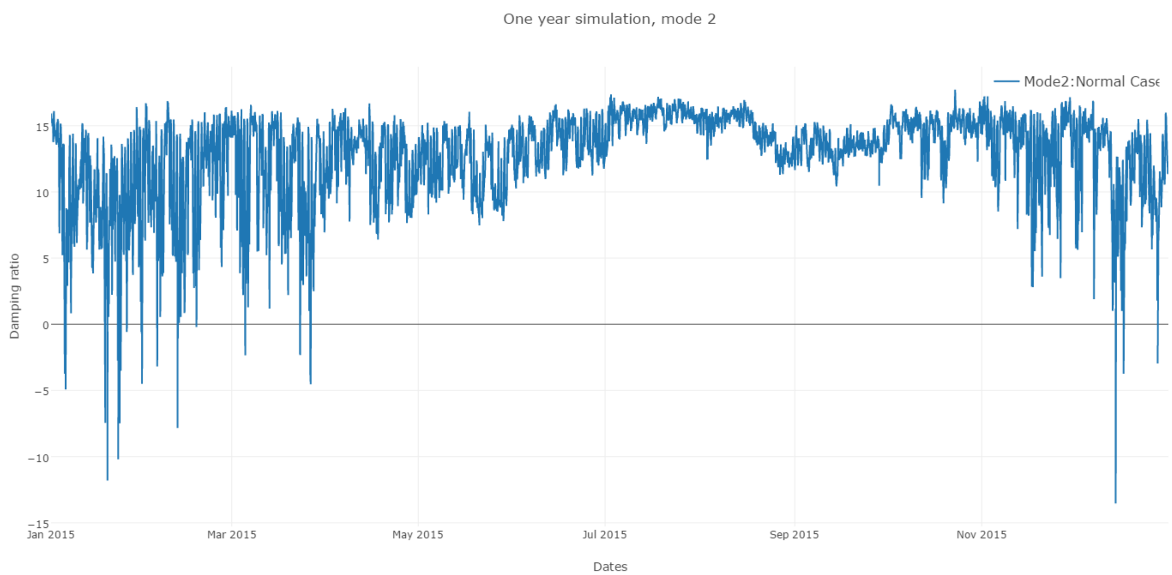

Figure 5.13: Damping ratio of mode 2 is plotted simulated for a year

Worst damped			Best damped		
Nr	Dates	Damp %	Nr	Dates	Damp %
1	2015-12-14 18:00	-13.536	30	2015-08-15 11:00	16.936
2	2015-02-11 08:00	-7.831	29	2015-07-23 18:00	16.942
3	2015-12-14 19:00	-5.758	28	2015-07-23 17:00	16.945
4	2015-03-26 19:00	-4.534	27	2015-07-04 00:00	16.957
5	2015-01-30 17:00	-4.507	26	2015-07-25 12:00	16.969
6	2015-12-14 20:00	-3.876	25	2015-11-01 13:00	16.969
7	2015-12-17 08:00	-3.733	24	2015-11-01 14:00	16.995
8	2015-03-26 16:00	-3.646	23	2015-07-25 00:00	17.003
9	2015-02-04 16:00	-3.195	22	2015-07-18 23:00	17.009
10	2015-12-28 11:00	-2.961	21	2015-07-25 17:00	17.013
11	2015-02-04 17:00	-2.506	20	2015-07-25 10:00	17.023
12	2015-03-05 10:00	-2.354	19	2015-07-25 01:00	17.027
13	2015-03-23 08:00	-2.333	18	2015-07-24 23:00	17.045
14	2015-03-26 17:00	-1.793	17	2015-07-19 07:00	17.053
15	2015-01-30 16:00	-1.604	16	2015-07-18 13:00	17.081
16	2015-02-11 11:00	-1.067	15	2015-07-02 21:00	17.090
17	2015-03-05 09:00	-1.037	14	2015-10-23 10:00	17.101
18	2015-01-30 12:00	-0.579	13	2015-07-03 23:00	17.126
19	2015-03-26 18:00	-0.271	12	2015-07-02 16:00	17.132
20	2015-02-17 12:00	-0.209	11	2015-11-29 20:00	17.159
21	2015-02-11 19:00	0.098	10	2015-11-01 21:00	17.182
22	2015-12-17 10:00	0.440	9	2015-07-18 08:00	17.188
23	2015-02-12 09:00	0.551	8	2015-07-02 22:00	17.201
24	2015-02-05 16:00	0.555	7	2015-11-01 20:00	17.204
25	2015-03-05 07:00	0.595	6	2015-11-02 22:00	17.211
26	2015-02-17 13:00	0.721	5	2015-10-23 06:00	17.348
27	2015-02-13 14:00	0.851	4	2015-07-02 23:00	17.361
28	2015-03-05 08:00	0.903	3	2015-10-23 09:00	17.490
29	2015-12-16 18:00	1.010	2	2015-10-23 07:00	17.557
30	2015-03-26 08:00	1.011	1	2015-10-23 08:00	17.717

Table 5.7: The 30 best and worst damping situation through the year, 196 points had less than 5 % damping.

Chapter 6

Discussion

6.1 Discuss of the results

Normal simulation Figure 5.1 shows the results of a week of simulation. Not all the modes behave in the same manner, showing that while Mode 2, 3 and 4 moves in a direction of down to the right, Mode 1 stays mostly at the same place. The movement of the eigenvalues are clearly seen, even though any traces have not been included to show the actual movement. Further on, it looks like the simulation lost track of Mode 4, making a clear jump from -0.5 to -0.7 on the real axis. By comparing Figure 5.1 with the given points from the QR-simulations in Figure 5.4, it's clearly that some points are left unattended in the area around x-axis = -0.5 and a jump may have indeed occurred. The QR-results shows that the movement of the mode that started as Mode 4 with a frequency of 5.3 rad/s does barely move. Inspecting the time line in Figure 5.2 and Figure 5.3 shows that even though Mode 4 lost track, the frequency stayed almost identical. As the DPSE-method uses a shift from the previous point to converge to the next point, it looks like the imaginary part plays a larger role than the real part when choosing the next shift. The new track may therefore be a result of a more dominant pole with the same frequency, making the tracking jump.

Multiple transfer functions was tried to keep track of Mode 4, but unfortunately the dominance from this mode made it difficult to track. Consequently, the information about this mode is lost during simulation, however this may not be dangerous, since the movement is limited, but all together it is unfortunate for the integrity of the on-line DSA when operating with blind spots. Maybe, in a less aggregated system model, there exist a transfer function where this mode is dominant, and could more easily be tracked.

By visual inspection of Mode 3 in Figure 5.1, the shape could indicate that two different modes are operating in this area, one moving almost horizontal, and one moving in a diagonal direction. This has not been further investigated, leaving the possibility that in fact, both directions noted belongs to Mode 3. Nevertheless, the mode of most interest is Mode 2, clearly crossing the yellow 5 % damping line, multiple times. This would raise the Alert/Emergency state, because one of the modes are violating the grid standards.

From the weekly timeline graph in Figure 5.2 a clear weekly pattern arise. Knowing that March 7. 2015 was a Saturday, it could be noted that the frequency has different behaviour for the weekend than during weekdays. The system is better damped during low load, at night and weekends. While the damping of Mode 2, 3 and 4 moves almost in the same direction for different operation points, it looks like Mode 1 is negative related, becoming worse dampened in low load situation, when Mode 2, 3 and 4 is getting better damping.

Including the contingencies further increase the details of the monitor screen. As shown in the two-axis Figure 5.5 the visuals are becoming crowded. As for Mode 2, 3 and 4 the contingencies barely shifts the behaviour, while in Mode 1 a totally different behaviour have risen. A line trip between SE1 and FI1 reduces the damping of Mode 1 and introduce a large movement of the mode, moving in a crook form.

Looking at the timeline Figure 5.6 and Figure 5.7 the simulation shows that if the contingencies was applied, where the new behaviour of the system would be. The Alert state would arise multiple times, as for each time any contingencies would push the damping below 5 %. Through this week, the Alert state would have risen 7 times. At March 6. 08:00, the normal condition of Mode 2 was less than 5 %, being as low damped as 1.38 % raising the Emergency state. If a line trip between bus 5301 and bus 5304 would have occurred at this time period, the system would have moved towards an unstable situation, putting the system in Extreme state, where the possibility for black-out arise. The simulations also shows an interesting fact, that some of the contingencies will increase the system damping instead of making it worse.

The on-line DSA tools shows that it can track multiple modes and multiple contingencies. Having the possibility to turn on and off the traces does help the operator to inspect a specific mode or situation.

From simulations Mode 2 is noted as being the most critical mode related to damping. A simulation on a high damped situation and a low damped situation could provide insight about the mode. $\lambda_{2,high}$ = -0.5193 + j3.3125 and $\lambda_{2,low}$ = -0.02495 + j1.9476. To check if these modes are in fact the same, the mode shapes in Figure 5.8 (Table 5.3.2) and Figure 5.10 (Table 5.4.2) are compared, showing that the same machines are oscillating against each other. Finland and Northern Sweden against Southern Norway. Checking the QR-results in Table 5.1 and Table 5.4 shows that Mode 2 has the lowest frequency in both situation, making it likely they are the same mode.

The results of Generation Hopf Bifurcation at the high damped situation at March 6, 03:00, shown in Figure 5.9 and Table 5.3, shows the margin of moving the mode from 15.49 % damping to 10 % damping. Reducing the damping ratio is easiest done by increasing the production at bus 7000, 6100 and 5300, corresponding to South Finland and West-Norway, while reducing the production in mid-Sweden. Thereupon it looks like moving the production further out to the edges and reducing in the middle will decrease the damping factor of this mode. Turning from net export from SE3 to net import.

On the other hand, checking the Generation Hopf Bifurcation at the low damped situation at March 6, 08:00, shows how the production should change to make the system more secure, moving Mode 2 from 1.28 % to 5 % damping. Compared to the margin calculation above in the high damped situation, which could handle a large deviation in the production, a much smaller deviation is needed to secure the grid for the low load situation. As the result of increasing production at bus 5300 and 6100 in the latter example made the system less secure, the opposite is shown in Figure 5.11 and Table 5.6, in addition the method wants to increase the production in mid-Sweden. However, strangely the production at bus 7000 is moving in the same direction for both situations.

The results from the one year simulation, shown in Figure 5.12 and Figure 5.13 shows that the software is able to track the two modes of interest throughout the whole year. From Figure 5.13 the yearly results of the damping ratio of Mode 2 shows clearly seasonal variations. During the low load summer the damping ratio of Mode 2 is more secure, and the daily variations are less observed, resulting in a more constant damping ratio. Notably the system is more insecure during winter. Even though the damping ratio at some points are as good in the winter as during the summer, the high load situations introduces large damping fluctuations through the day, making some unstable situations. In Table 5.7 the best and worst damping situation are presented. The worst situation recorded is at December 14, 2015 with -13.536 % damping, meaning that oscillations would have increased with an amplitude of 13.5 % for each oscillation. All the 30 worst situations are during the winter months from December to March. Most of the best damping situations are during the summer, in July. October 23th and November 1th was days with particular good damping.

6.2 Discuss of the on-line DSA

Testing and simulations have shown that the proposed scheme for an on-line dynamic security assessment tool works. Some parts still need to be pointed out; first, even though the DPSE-method ensure the system does not converge to the same eigenvalue, there is no guarantee that the mode of interest, actually is being tracked. Secondly, both the collected data from the SCADA system and the system model is highly aggregated, introducing many assumptions to the monitoring. Thirdly, the generator base case used to tune the model turned out to contain some errors about bus 3000, where the Max Production limit (Pmax) was higher than Mva Base (Mbase). Changing this would have changed the behaviour of the modes. And lastly, the Generation Hopf Bifurcation method proved to solve the problem in this simulations, while for larger systems the methods could become slow, and may not converge, however, the method is still under development and could be more robust at a later stage.

When running the simulations, multiple times the DSA-tool reported that the production of some generators was exceeding the production limits. To remedy this problem either the generator parameters should be changed, or shifting the partial productions from the generators in the same area.

Some technical problems concerning the graphical user interface came to being, where the simulation plots and the alerts did not update after the initialization of the program. For this reason, during simulation, CEPEL's plot software was used to visualize the results, and a stand-alone python script was executed in Jupyter Notebook with the plotting package from Plot.ly to perform the post-simulation visualizations. In this plots, hovering over a time line could give full insight into all the modes at a glimpse, as shown in Figure 5.7 or from a particular situation as shown in Figure 5.5. The modes and contingencies could be turned on/off by the user, making it easier to compare the wanted information. The same script was providing the tabular output of all the worst damped situations.

During the simulation, some elements of the DSA-tool was slowing down the process, such as connecting and receiving data from the FTP server, connecting to the PSS/E licence server or performing the Generation Hopf Bifurcation method. Solutions could be to implement a local buffer from the FTP-server or have direct access to the data, secondly, own a PSS/E licence installed on the computer running the monitoring, and lastly, keeping the model as small as possible. Because of the unpredictable simulation time, the system will wait until PacDyn has made a signal telling it is finished processing the data before proceeding.

Chapter 7

Conclusion and further work

7.1 Conclusion

Throughout this master thesis a fully functional on-line dynamic security assessment (On-line DSA) has been developed, though with some limitations. After a short introduction to the field of stability assessment, the mathematical background was covered, then the method used to develop the software was described. The tool proved to be able to track multiple eigenvalues, with and without contingencies applied, and report results almost instantly. The simulations showed a critical mode fluctuating between Finland and Southern Norway, which was worst during the high load winter situations, and well damped during the summer. Use of Generation Hopf Bifurcation could in some cases be used to improve the damping situation, by shifting the production from one site to another. The Dominant Pole Spectrum Eigensolver (DPSE) did lose track of some modes during simulations, revealing weaknesses in the method.

7.2 Further Work

The on-line DSA presented does show a fully working scheme for assessing the small-signal stability in close to real-time, but some few improvements should be considered before implementing the system into a real grid, such as;

Connect a PMU

For on-line DSA systems, it is crucial to have updated data from the system, and the development in the field of PMU's are opening up for better system monitoring. Having synchronous measurements and detailed information about the grid makes it possible to perform advanced analysis on the grid.

Utilizing AMS

The data that will be produced by all the AMS's systems that are under deployment could give detailed information to an on-line DSA tool for updating the model.

Tune the model

Having multiple points where the damping is less than 5 % and 0 % in the simulation clearly shows that the model is not completely well tuned. Using PMU's could be useful when tuning the model.

Database

As for now, the results are saved locally to a folder, however setting up a database could make the information more open for further analysis. The on-line DSA will generate a vast amount of data, that could provide good insight to the grid by utilizing machine learning techniques, big data and behaviour classifications.

Predictions

Implementing a spline extrapolation on the graphs could predict the future behaviour of the grid. Using forecasted data in the simulations could also give hint about future situations. For instance, two simulations could be run at the same time, one for the current situation, and one using the forecasted data.

Visualization

Some problems occurred when trying to implement the visual into the on-line DSA, and CEPEL's plotting system was used during the simulations. The same information was available, but some limitations in their plotting software makes it harder to compare the modes, and the timeline is only showing the hours since start of monitoring instead of date and time. A working python Jupyter Notebook script to fix this is added in the Appendix A.5 for off-line visualization an need to be implemented to the on-line DSA tool.

Automatic tuning of control systems

Shifting the generation from one area to another may increase the stability, however the raise of on-line DSA-tools could in the future give suggestions to controller systems, which could affect the stability more effectively, by automatically update the parameters of the controller to always ensure the best damping available. This task should be investigated more, and could be made as a master or doctor thesis, or tackled by a research team.

Overall performance

The script used to create the on-line DSA was programmed in Python. Other languages as C++ or Fortran are known to be faster. The software is a working minimal of what an on-line DSA tool could look like, and the algorithms may not follow modern coding standards or costumes.

Appendix A

Python Source Code

A.1 The GUI

```
from __future__ import print_function
import enaml
from enaml.qt.qt_application import QtApplication

# Importing the SimulationCase class for storing the case data
#from main_model import SimulationCase



import datetime
import os

from atom.api import Atom, Unicode, Range, Typed, observe, Value, Bool, Property
from enaml.application import deferred_call, timed_call         ###

from threading import Thread

import collectingData
import generatePSSEcode
from time import sleep
import numpy as np

# A Class for containing all the information of the setup-file
class SimulationCase(Atom):

    busy = Bool(False)

    saveToPath = Unicode(default=os.getcwd())
    baseGeneratorPath = Unicode((unicode(os.getcwd())+u'\AggregatedGenerator.csv').replace('\\','/'))
    baseLoadPath = Unicode((unicode(os.getcwd())+u'\LoadDataforScaling.csv').replace('\\','/'))
    baseRawPath = Unicode((unicode(os.getcwd())+u'\\Nordic44xlsagr.raw').replace('\\','/'))
    outputRawPath = Unicode((unicode(os.getcwd())+u'\\Nordic44xlsagr_rtime.raw').replace('\\','/'))

    dontStopSimulation = Bool(default=False)   # For running realtime, set to True, not implemented
    pauseSimulation = Bool(default=False)
```

```

    startDateTime = Property()
    _startDateTime = Typed(datetime.datetime)

    currentDateTime = Property()
    _currentDateTime = Typed(datetime.datetime)

    endDateTime = Property()
    _endDateTime = Typed(datetime.datetime)

    incrementTime = Property()
    _incrementTime = Typed(datetime.time)

    incrementTimeDelta = Property()
    _incrementTimeDelta = Typed(datetime.timedelta)

    #Saving for figuring
    totalHVDC_flow = Property()
    _totalHVDC_flow = Typed(np.ndarray)

    sumOfProduction = Property()
    _sumOfProduction = Typed(np.ndarray)

    sumOfConsumption = Property()
    _sumOfConsumption = Typed(np.ndarray)

    timeSerie = Property()
    _timeSerie = Typed(np.ndarray)

    # Getter and setter for startDateTime
    def _set_startDateTime(self, startDateTime):
        self._startDateTime = startDateTime
        self._currentDateTime = startDateTime

    def _get_startDateTime(self):
        return self._startDateTime

    # Getter and setter for currentDateTime
    def _set_currentDateTime(self, currentDateTime):
        self._currentDateTime = currentDateTime

    def _get_currentDateTime(self):
        return self._currentDateTime

    # Getter and setter for endDateTime
```

```
    def _set_endDateTime(self,endDateTime):
        self._endDateTime = endDateTime

    def _get_endDateTime(self):
        return self._endDateTime

    # Getter and setter for incrementTime
    def _set_incrementTime(self,incrementTime):
        self._incrementTime = incrementTime
        self._incrementTimeDelta = datetime.timedelta(hours=incrementTime.hour,minutes=incrementTime.minute,
 seconds=incrementTime.second)


    def _get_incrementTime(self):
        return self._incrementTime

    ## Getter and setter for incrementTimeDelta
    #def _set_incrementTimeDelta(self,incrementTimeDelta):
    #    self._incrementTimeDelta = incrementTimeDelta

    def _get_incrementTimeDelta(self):
        return self._incrementTimeDelta

    def _get_totalHVDC_flow(self):
        return self._totalHVDC_flow

    def _set_totalHVDC_flow(self, totalHVDC_flow):
        if str(self._totalHVDC_flow)=='[]':
            self._totalHVDC_flow = np.array(totalHVDC_flow)
        else:
            #Populating a matrix of HVDC_flow
            self._totalHVDC_flow = np.hstack([self._totalHVDC_flow,np.array(totalHVDC_flow)[:,[2]]])
            #print(self._totalHVDC_flow)


    def _get_sumOfProduction(self):
        return self._sumOfProduction

    def _set_sumOfProduction(self, sumOfProduction):
        #print('Production')
        if str(self._sumOfProduction)=='[]':
            placeholder = np.array(sumOfProduction)
            self._sumOfProduction = np.hstack([placeholder[:,[0]],placeholder[:,[1]]])
            #print(self._sumOfProduction)
        else:
```

```
            #Populating a matrix of productions, the [] around the number 2 makes a row vector
            self._sumOfProduction = np.hstack([self._sumOfProduction,np.array(sumOfProduction)[:,[1]]])
            #print(self._sumOfProduction)



    def _get_sumOfConsumption(self):
        return self._sumOfConsumption

    def _set_sumOfConsumption(self, sumOfConsumption):
        #print('Consumption')
        if str(self._sumOfConsumption)=='[]':
            placeholder = np.array(sumOfConsumption)
            self._sumOfConsumption = np.hstack([placeholder[:,[0]],placeholder[:,[2]]])
            #print(self._sumOfConsumption)
        else:
            #Populating a matrix of HVDC_flow, the [] around the number 2 makes a row vector
            self._sumOfConsumption = np.hstack([self._sumOfConsumption,np.array(sumOfConsumption)[:,[2]]])
            #print(self._sumOfConsumption)

    def _get_timeSerie(self):
        return self._timeSerie
#
    def _set_timeSerie(self,DateTime):
        #print('Time serie')
        if str(self._timeSerie)=='[]':
            self._timeSerie = np.array(self.currentDateTime)
            #print(self._timeSerie)
        else:
            #Populating a matrix of HVDC_flow, the [] around the number 2 makes a row vector
            self._timeSerie = np.hstack([self._timeSerie,self.currentDateTime])
            #print(self._timeSerie)


    # Functions
    def collectData2(self):
        threadedCollection(self)

    def SaveOperationPoints(self):
        np.savez('operationPoint_file',HVDC=self.totalHVDC_flow, Production=self.sumOfProduction, Consumption=
     self.sumOfConsumption, Time=self.timeSerie)

        # Step 1: Data is collected, for only getting the production and consumption data for reporting.
        # Does not run the simulation
    def collectData(self):
```

```
        if self.currentDateTime < self.endDateTime:
            print('Time processed: %s' % self.currentDateTime)
            #Collecting the data and making the system ready for the next step.
            try:
                HVDC_flow,operationPoint= collectingData.calculateCollectingData(self.currentDateTime)

                #Printing out the current time for using in PacDyn
                currentDateAndHourPATH = 'currentDateandTime2.txt'
                with open(currentDateAndHourPATH, 'w') as f:
                    dateAsString = self.currentDateTime.strftime('%d.%m.%Y-%H:%M:%S')
                    f.write(dateAsString)

                self.addHVDC_flow(HVDC_flow)
                self.addProdCon(operationPoint)
                self.addTime()
            except:
                print('Failed to collect data at: %s' % self.currentDateTime)

            self.currentDateTime += datetime.timedelta(hours=self.incrementTime.hour,minutes=self.incrementTime.minute,seconds=self.incrementTime.second)

            #sleep(0.01)
            self.collectData()
        else:
            print('Finished collecting data')
            print(self.currentDateTime)

        #self.startGeneratePSSEcode(operationPoint)

        # For writing history



    # Step 2: PSSE-code is generated
    def startGeneratePSSEcode(self,productiondata):
        generatePSSEcode.runScript(baseGeneratorPATH=self.baseGeneratorPath, baseLoadPATH=self.baseLoadPath, productionData=productiondata)
        print('Generated PSSE code')

    # Step 3: Run PSS/E-code
    def startRunPsseCode(self):
            #runpssepy.run()
                #Preparing for recognizing psspy library from PSS/E
        os.system('c:\python27\python runpssepy.py')

```

```
        #Printing out the current time for using in PacDyn
        currentDateAndHourPATH = self.saveToPath + '/' + 'currentDateandTime.txt'
        with open(currentDateAndHourPATH, 'w') as f:
            dateAsString = self.currentDateTime.strftime('%d.%m.%Y-%H:%M:%S')
            f.write(dateAsString)


    # Step 4: Save production to graph
    def addHVDC_flow(self,HVDC_flow):
        #print('HVDC')
        self.totalHVDC_flow = HVDC_flow
        #print(self._totalHVDC_flow)


    def addProdCon(self,operationPoint):
        #print('Production and Consumption')
        self.sumOfProduction = operationPoint
        self.sumOfConsumption = operationPoint
        #print(operationPoint)

    def addTime(self):
        self.timeSerie = self.currentDateTime





    def timedCollectData(self):
        while self.currentDateTime < self.endDateTime:
            thread = Thread(target=worker,args=(self,))
            thread.daemon = True
            thread.start()




    # Populating the data, this function is ran when the program is started
    # To have some initial values inside the class.
    def __init__(self):
        self._startDateTime = datetime.datetime.strptime('05.03.2015-11','%d.%m.%Y-%H')
        self._currentDateTime = self._startDateTime
        self._endDateTime = self._startDateTime + datetime.timedelta(days=int(7))
        self._incrementTime = datetime.time(1,0,0)
        self._totalHVDC_flow = np.array([])
        self._sumOfProduction = np.array([])
```

```
        self._sumOfConsumption = np.array([])
        self._timeSerie = np.array([])

    # For restaring all variables back to the first initial setting
    # So the program don't need to be restarted to start a simulation again.
    def restart(self):
        self._startDateTime = datetime.datetime.strptime('05.03.2015-11','%d.%m.%Y-%H')
        self._currentDateTime = self._startDateTime
        self._endDateTime = self._startDateTime + datetime.timedelta(days=int(7))
        self._incrementTime = datetime.time(1,0,0)
        self._totalHVDC_flow = np.array([])
        self._sumOfProduction = np.array([])
        self._sumOfConsumption = np.array([])
        self._timeSerie = np.array([])

# The software is programmed to have one master on several slaves or workers
# This is implemented so that the program will not freeze during simulation when waiting
# for the next step.
def worker(simulationcase):

    while simulationcase.pauseSimulation == True:
        print('On-line DSA is busy')
        sleep(0.2)

    # It PacDyn is busy, it will wait till it has released the
    while os.path.isfile('fdm_busy.txt'):
        print('PacDyn is busy')
        sleep(0.3)

    # If the end has not been reached, it will simulate a new step.
    if simulationcase.currentDateTime < simulationcase.endDateTime:
        deferred_call(setattr, simulationcase, 'pauseSimulation', True)
        simulationcase.addTime()
        # Step 1: Data is collected
        print('Time processed: %s' % simulationcase.currentDateTime)
        #Collecting the data and making the system ready for the next step.
        # The try functio is to handle any errors that should occur.
        try:
            HVDC_flow,operationPoint= collectingData.calculateCollectingData(simulationcase.currentDateTime)

            simulationcase.addHVDC_flow(HVDC_flow)
            simulationcase.addProdCon(operationPoint)

            #Step 2
            simulationcase.startGeneratePSSEcode(operationPoint)
```

```python

            # For writing history
            #Printing out the current time for using in PacDyn
            currentDateAndHourPATH ='currentDateandTime2.txt'
            with open(currentDateAndHourPATH, 'w') as f:
                dateAsString = simulationcase.currentDateTime.strftime('%d.%m.%Y-%H:%M:%S')
                f.write(dateAsString)


            #Step 3:
            os.system('c:\python27\python runpssepy.py')
        except:
            deferred_call(setattr, simulationcase, 'pauseSimulation', False)

        #Printing out the current time for using in PacDyn
        currentDateAndHourPATH = simulationcase.saveToPath + '/' + 'currentDateandTime.txt'
        with open(currentDateAndHourPATH, 'w') as f:
            dateAsString = simulationcase.currentDateTime.strftime('%d.%m.%Y-%H:%M:%S')
            f.write(dateAsString)

        #Step 4:
        p = simulationcase.currentDateTime+datetime.timedelta(hours=simulationcase.incrementTime.hour,minutes=simulationcase.incrementTime.minute,seconds=simulationcase.incrementTime.second)
        deferred_call(setattr, simulationcase, 'currentDateTime', p)

        #Step 5: Updating the GUI
        #deferred_call(setattr,simulationcase,'currentDateTime',)
    deferred_call(setattr, simulationcase, 'busy', False)
    deferred_call(setattr, simulationcase, 'pauseSimulation', False)


def threadedCollection(simulationcase):
    if not simulationcase.busy:
        simulationcase.busy = True
        thread = Thread(target=worker,args=(simulationcase,))
        thread.daemon = True
        thread.start()


# Starting the GUI loop
if __name__ == '__main__':

    # Getting the GUI file from main.enaml
    with enaml.imports():
        from main import *
```

```

    #Setting up the initial base case
    initialCase = SimulationCase()
    #print(initialCase.currentDateTime)

    # Defining the backend as PyQt application
    app = QtApplication()


    view = Main(case=initialCase)
    view.show()

    app.start()
```

: files/mainGUI2.py

A.1.1 Enaml-file

```
# This is the GUI written in PyQt by using ENAML-markup language
from enaml.layout.api import (vbox, hbox, spacer)
from enaml.widgets.api import (
    MainWindow, ToolBar, DockPane, MenuBar, Menu, Action, ActionGroup,
    StatusBar, StatusItem, Container, Html, PushButton, Label, Field,
    MPLCanvas, CheckBox, ComboBox, Window, TimeSelector, DatetimeSelector, FileDialogEx, Calendar, Timer,
     WebView
)

from matplotlib.figure import Figure
import datetime
import os
import collectingData

enamldef MyStatusBar(StatusBar):
    attr current
    StatusItem:
        Label:
            text << current

def testCollecting(datetimeCase, incTime):
    collectingData.calculateCollectingData(datetimeCase)
    incrementBy = datetime.timedelta(hours=incTime.hour,minutes=incTime.minute,seconds=incTime.second)
    datetimeCase += incrementBy
    print datetimeCase

enamldef MyToolBar(ToolBar):
    Action:
        text = 'Restart'
        tool_tip = 'Initialize the system'
        triggered :: case.restart()
    Action:
        checkable = True
        text = 'Run'
        triggered :: timer.start()
        tool_tip = 'Start simulation'
    Action:
        text = 'Collect data'
        triggered :: case.collectData()
    Action:
        text = 'Next'
        triggered :: case.collectData2()
        tool_tip = 'Simulate a single operating point'
```

```
    Timer: timer:
        interval = 500
        single_shot = False
        timeout :: case.collectData2()
    Action:
        text = 'Save all operation points to pickle-file'
        triggered :: case.SaveOperationPoints()

enamldef MyDatePicker(DockPane):
    title = 'Select dates'
    attr case

    Container:
        Label:
            text = 'Start time (dd.mm.yyyy hh.mm.ss)'
        DatetimeSelector: startTime:
            calendar_popup = True
            datetime := case.startDateTime
        Label:
            text = 'End time (dd.mm.yyyy hh.mm.ss)'
        DatetimeSelector: endTime:
            calendar_popup = True
            datetime := case.endDateTime
        Label: selectedTime:
            text = 'Increment time (hh.mm.ss)'
        TimeSelector: incrementTime:
            time := case.incrementTime

        Container:
            constraints = [vbox(saveTo,
                                hbox(saveToFld,saveToPb))
                          ]
            Label: saveTo:
                text = 'Save to folder: '
            Field: saveToFld:
                placeholder = 'Save to folder...'
                read_only = True
                text := case.saveToPath
            PushButton: saveToPb:
                text = "..."
                clicked ::
                    path = FileDialogEx.get_existing_directory()
                    if path:
                        case.saveToPath = path

```

```
        Container:
            constraints = [vbox(lbl2,
                                hbox(fld2,pb2))
                            ]
            Label: lbl2:
                text = 'Generator base case (CSV)'
            Field: fld2:
                placeholder = 'Generator base case...'
                text := case.baseGeneratorPath
            PushButton: pb2:
                text = "..."
                clicked ::
                    path = FileDialogEx.get_open_file_name()
                    if path:
                        case.baseGeneratorPath = path

        Container:
            constraints = [vbox(lbl3,
                                hbox(fld3,pb3))
                            ]
            Label: lbl3:
                text = 'Load base case (CSV)'
            Field: fld3:
                placeholder = 'Generator base case...'
                text := case.baseLoadPath
            PushButton: pb3:
                text = "..."
                clicked ::
                    path = FileDialogEx.get_open_file_name()
                    if path:
                        case.baseLoadPath = path

        Container:
            constraints = [vbox(lbl4,
                                hbox(fld4,pb4))
                            ]
            Label: lbl4:
                text = 'PSSE system case (Raw)'
            Field: fld4:
                placeholder = 'Raw file...'
                text := case.baseRawPath
            PushButton: pb4:
                text = "..."
                clicked ::
                    path = FileDialogEx.get_open_file_name()
```

```
                if path:
                    case.baseRawPath = path


enamldef Main(MainWindow):
    attr case = SimulationCase()
    MyStatusBar:
        current << str(case.currentDateTime)
    MyToolBar:
        pass
    MyDatePicker:
        case := parent.case
        dock_area = 'left'
        movable = False
    Container:
        constraints = [vbox(
                            hbox(check, spacer)
                            )]
        CheckBox: check:
            text = case.saveToPath
```

: files/main.enaml

A.2 Collecting data

```
import datetime
import numpy as np
import csv
from ftplib import FTP
from IPython.lib.security import passwd # for use in ftp server connection
import urllib

FTPstudentlogin = 'student'
FTPstudentpassword = '********'
collectFromFTPserver = True

def calculateCollectingData(caseDateAndHour,saveProductionToPath=0,saveHVDCtoPath=0,saveFolderPath=0):

    # r1,r2 = Collect data 1:
    H0,P0 = collectData(caseDateAndHour)

    if caseDateAndHour.minute is not 0:
        # r3,r4 = Collect data next hour
        # r1,r2 = Interpolate(r1,r2,r3,r4,minutes+seconds)
        nextHour = caseDateAndHour + datetime.timedelta(hours=1)
        H1,P1 = collectData(nextHour)
        H0,P0 = interpolateCollectedData(H0,P0,H1,P1,caseDateAndHour)

    saveData(H0,P0)

    return(H0,P0)

def interpolateCollectedData(H0,P0,H1,P1,caseDateAndHour):
    # Finding the HVDC interpolation
    H = np.hstack([H0,H1])
    Ha = H[:,2][:,np.newaxis]     # Copy the first points
    Hb = H[:,5][:,np.newaxis]     # Copy the last points

    # Finding the Production and consumption interpolation
    P = np.hstack([P0,P1])
    Pa = P[:,1][:,np.newaxis]    # Copy the first Productions
    Pb = P[:,4][:,np.newaxis]    # Copy the last Productions
    Ca = P[:,2][:,np.newaxis]    # Copy the first Consumption
    Cb = P[:,5][:,np.newaxis]    # Copy the last Consumption

    #fractionOfHour
    x = caseDateAndHour.minute*1.0/60 + caseDateAndHour.second*1.0/3600

```

```
    H = np.delete(H,[2,3,4,5],axis=1) # Deleting old values
    P = np.delete(P,[1,2,3,4,5],axis=1) # Deleting old values

    H = np.hstack([H,np.around(Ha + (Hb-Ha)*x,decimals=0).astype(int)])  # Linear interpolation of HVDC
    P = np.hstack([P,np.around(Pa + (Pb-Pa)*x,decimals=0).astype(int)])  # Linear interpolation of Production
    P = np.hstack([P,np.around(Ca + (Cb-Ca)*x,decimals=0).astype(int)])  # Linear interpolation of Consumption

    return (H.tolist(),P.tolist())

def saveData(listOfHVDC, outputToCurrentProductionAndLoadForScaling, saveHVDCtoPath='currentHVDCData.csv',
     saveProductionToPath='currentProductionAndLoadForScaling.csv'):
    with open(saveHVDCtoPath, 'w') as f:

        writer = csv.writer(f,delimiter=';')
        writer.writerow(['Bus','ID','Flow'])
        for row in listOfHVDC:
            writer.writerow(row)

    # Save the currentProductionAndLoadForScaling.csv
    with open(saveProductionToPath, 'w') as f:
        # using outputToCurrentProductionAndLoadForScaling

        writer = csv.writer(f,delimiter=';')
        writer.writerow(['Area','Production','Consumption'])
        for row in outputToCurrentProductionAndLoadForScaling:
            writer.writerow(row)

def collectData(caseDateAndHour):

    class StudyCase(object):
        def __init__(self,dateAndHour):
            self.P = {}
            self.C = {}
            self.FLOW = {}
            self.caseTime = dateAndHour

    # Alternative paths if before 2016
    pathNorway = '/Operating_data/Norway/'
    pathSweden = '/Operating_data/Sweden/'
    pathFinland = '/Operating_data/Finland/'
    yearNumber = datetime.datetime.strftime(caseDateAndHour,'%y')
    weekNumber = "%02d" % caseDateAndHour.isocalendar()[1]
    pathEnding = yearNumber + weekNumber + '.sdv'

    if caseDateAndHour.year < 1998:
```

```python
    print('No data before 1997 exist, program is quitting')
    exit()
elif caseDateAndHour.year < 2016:
    pathNorway = pathNorway + str(caseDateAndHour.year) + '/pono' + pathEnding
    pathSweden = pathSweden + str(caseDateAndHour.year) + '/pose' + pathEnding
    pathFinland = pathFinland + str(caseDateAndHour.year) + '/pofi' + pathEnding
elif caseDateAndHour.year == 2016:                          # This should be changed to update to todays year
    pathNorway = pathNorway + '/pono' + pathEnding
    pathSweden = pathSweden + '/pose' + pathEnding
    pathFinland = pathFinland + '/pofi' + pathEnding

#print(caseDateAndHour)
#print(pathNorway)
#print(pathSweden)
#print(pathFinland)

# List of lines, new lines will be appended here
lines = []
def collectFromFTP():
    # Connecting to the FTP-server
    ftp = FTP('ftp.nordpoolspot.com')
    ftp.login(user=FTPstudentlogin, passwd =FTPstudentpassword)
    ftp.retrlines('RETR '+ pathNorway, lines.append)
    ftp.retrlines('RETR '+ pathSweden, lines.append)
    ftp.retrlines('RETR '+ pathFinland, lines.append)

def collectFromLocal():
    with open(pathNorway, 'r') as f:
        for row in f:
            lines.append(row[:-1])
    with open(pathSweden, 'r') as f:
        for row in f:
            lines.append(row[:-1])
    with open(pathFinland, 'r') as f:
        for row in f:
            lines.append(row[:-1])

if collectFromFTPserver == True:
    collectFromFTP()
else:
    pathNorway = pathNorway[1:]
    pathSweden = pathSweden[1:]
    pathFinland = pathFinland[1:]
    collectFromLocal()
```

```

    # Making a string out of the list of lines recieved from the ftp-server
    csvfile = ''
    for line in lines:
        csvfile = csvfile + line + '\n'

    # For deleting the last '\n' to prevent errors while reading an empty line.
    csvfile = csvfile[:-2]

    testCase = StudyCase(caseDateAndHour)
    #testCase.caseDate = dateOfInterest
    #testCase.caseHour = hourOfInterest

    # Initializing all flow keys
    testCase.FLOW['FI_EE'] = 0
    testCase.FLOW['FI_RU'] = 0
    testCase.FLOW['FI_SE3'] = 0
    testCase.FLOW['NO_DK'] = 0
    testCase.FLOW['NO_NL'] = 0
    testCase.FLOW['SE3_DK1'] = 0
    testCase.FLOW['SE3_FI'] = 0
    testCase.FLOW['SE4_DK2'] = 0
    testCase.FLOW['SE4_DE'] = 0
    testCase.FLOW['SE4_PL'] = 0

    # Because of summer time, a shift is needed to collect the time correctly.
    hourOfInterest = caseDateAndHour.hour
    if caseDateAndHour.year >=2014:
        if hourOfInterest > 2:
            hourOfInterest +=1

    dateOfInterest = datetime.datetime.strftime(caseDateAndHour,'%d.%m.%Y')

    # For splitting the data into cells
    reader = csv.reader(csvfile.split('\n'), delimiter=';')
    for row in reader:
        if row[0] =='PS':
            if row[1] =='P':
                if row[5] == dateOfInterest:
                    #print(row)
                    try:
                        testCase.P[row[6]]=int(row[hourOfInterest+7])
                    except:
                        pass
        if row[0] =='FB':
```

```
            if row[1] =='F':
                if row[5] == dateOfInterest:
                    #print(row)
                    try:
                        testCase.C[row[6]]=int(row[hourOfInterest+7])
                    except:
                        pass
        if row[0] =='UT':
            if row[1] =='U':
                if row[5] == dateOfInterest:
                    #print(row)
                    try:
                        testCase.FLOW[row[6]]=int(row[hourOfInterest+7])
                    except:
                        pass
    # Using "With open" deletes a variable after use, her the function "del" is used instead.
    del reader

    # Giving name to the areas, in NordPoolSpot the areas are given by numbers.
    def lookUpAreaNumberFromName(AreaName):
        areanumber=-1
        if AreaName == 'NO1':
            areanumber=11
        elif AreaName == 'NO2':
            areanumber=12
        elif AreaName == 'NO3':
            areanumber=13
        elif AreaName == 'NO4':
            areanumber=14
        elif AreaName == 'NO5':
            areanumber=15
        elif AreaName == 'SE1':
            areanumber=21
        elif AreaName == 'SE2':
            areanumber=22
        elif AreaName == 'SE3':
            areanumber=23
        elif AreaName == 'SE4':
            areanumber=24
        elif AreaName == 'FI':
            areanumber=31
        return areanumber

    # Sort the production keys
    outputToCurrentProductionAndLoadForScaling = []
```

```
    for key in sorted(testCase.P.keys()):
        if lookUpAreaNumberFromName(key) == -1:
            pass
        else:
            #print('%s:%1.f' % (key,testCase.P[key]))
            #print('%s:P %1.f C %1.f' % (lookUpAreaNumberFromName(key),testCase.P[key],testCase.C[key]))
            outputToCurrentProductionAndLoadForScaling.append([lookUpAreaNumberFromName(key),testCase.P[key],
 testCase.C[key]])

    outputToCurrentProductionAndLoadForScaling = sorted(outputToCurrentProductionAndLoadForScaling)

    # The mapping of the HVDC-cables into a bus load.
    HVDCflowBasedOnBus = {7020: testCase.FLOW['FI_EE']+testCase.FLOW['FI_RU'],
                          7010: testCase.FLOW['FI_SE3'],
                          5610: testCase.FLOW['NO_DK'],
                          5620: testCase.FLOW['NO_NL'],
                          3360: testCase.FLOW['SE3_DK1'],
                          3020: testCase.FLOW['SE3_FI'],
                          8600: testCase.FLOW['SE4_DK2'],
                          8700: testCase.FLOW['SE4_DE']+testCase.FLOW['SE4_PL']
    }

    # Save the currentHVDCData.csv
    listOfHVDC = []
    for bus in sorted(HVDCflowBasedOnBus):
        listOfHVDC.append([bus,1,(-1)*HVDCflowBasedOnBus[bus]])
    return (listOfHVDC,outputToCurrentProductionAndLoadForScaling)

#For testing the module
#caseDateAndHour = datetime.datetime.strptime('11.03.2015-08:30','%d.%m.%Y-%H:%M')
#calculateCollectingData(caseDateAndHour)
```

: files/collectingData.py

A.3 Generating PSS/E code

```
import csv
import math

# To use this function two files are needed
# 1. A basecase file for scaling the buses
# 2. Production and consumption numbers based on areas

# 1.1 Basecase should have the format:  The postfix[1] = _oneGenerator
#    [Busnumber,ID, AreaNR, BusParticipationFactor, Pmax1, Pmin1, Qmax1, Qmin1, Mbase1, R_source1, X_source1,
     Rtran1, Xtran1, MinimumRunningGen, MaxRunningGen]

# 2.1 Production and consumption file should be on the format
#    [AreaID, AreaProduction, AreaConsumption]

baseGeneratorPATH = 'AggregatedGenerator.csv'
saveProductionToPath = 'currentProductionAndLoadForScaling.csv'
baseLoadPATH = 'LoadDataforScaling.csv'
saveHVDCtoPath = 'currentHVDCData.csv'


#For calculating how many generators should be running
def calculateGenData(AreaID, AreaProduction, listOfProductionData, baseGeneratorPATH):
    #For calculating how many generators should be running
    # The upper limit is maximum allowed production before a new generator is turn on as percentage
    # (Should be between 0-1. If the maximum operation point should be 0.95 of Pmax, change the upperLimmit to
     0.95)
    upperLimit = 1

    # Open up the csvfile in the path
    with open(baseGeneratorPATH) as csvfile:
        #outfile = []
        reader = csv.reader(csvfile,delimiter=';')
        # Running through each line of the file
        for row in reader:
            if row[2]==str(AreaID):
                BusProduction = round(AreaProduction*float(row[3]),2)
                MaxProduction = round(float(row[4])*float(row[14])*upperLimit,2)
                if BusProduction <= MaxProduction:
                    busNr = int(row[0])
                    N = math.ceil(BusProduction/(float(row[4])))
                    ID = int(row[1])
                    Pgen = BusProduction
                    Pmax = round(float(row[4])*N,2)
```

```
                    Pmin = round(float(row[5])*N,2)
                    Qmax = round(float(row[6])*N,2)
                    Qmin = round(float(row[7])*N,2)
                    Mbase = round(float(row[8])*N,2)
                    Rsource = round(float(row[9])/N,5)
                    Xsource = round(float(row[10])/N,5)
                    RTran = round(float(row[11])/N,5)
                    XTran = round(float(row[12])/N,5)
                    #print(row)
                    #print('Bus,Pgen,Pmax,Pmin,Qmax,Qmin,Mbase,Rsource,XSource,RTran,Xtran,N')
                    #print((busNr,BusProduction,Pmax,Pmin,Qmax,Qmin,Mbase,Rsource,Xsource,RTran,XTran,N))
                    #print('ok production')
                    #print(BusProduction)
                    #print(MaxProduction)

                    #outfile.append((busNr,BusProduction,Pmax,Pmin,Qmax,Qmin,Mbase,Rsource,Xsource,RTran,XTran,N))
                    listOfProductionData.append(GenUpdateData(busNr,ID,Pgen,Pmax,Pmin,Qmax,Qmin,Mbase,Rsource,Xsource,RTran,XTran,N))
                    #.append(GenUpdateData(busNr,BusProduction,Pmax,Pmin,Qmax,Qmin,Mbase,Rsource,Xsource,RTran,XTran,N))

                else:
                    print('Production: %0.1f \nMax Production: %0.1f' % (BusProduction,MaxProduction))
                    print('Production at bus %s is higher than its production limit by %0.1f MW.' % (row[0],BusProduction-MaxProduction))

                    with open('ProductionOverLimit.log','a') as f:
                        f.write('Production: %0.1f \nMax Production: %0.1f' % (BusProduction,MaxProduction))
                        f.write('Production at bus %s is higher than its production limit by %0.1f MW.' % (row[0],BusProduction-MaxProduction))

def calculateLoadData(AreaID, AreaConsumption, listOfConsumptionData, baseLoadPATH,sumOfActivePowerInArea):
    with open(baseLoadPATH) as csvfile:
        reader = csv.reader(csvfile,delimiter=';')
        pTotal = sumOfActivePowerInArea[AreaID]

        # Running through each line of the file, format:
        # ['#BusNR', 'ID', 'Scalable', 'Area', 'Pload', 'Qload']
        for row in reader:
            if row[2]=='1':
                if row[3]==str(AreaID):
                    bus = int(row[0])
                    ID = int(row[1])
```

```
                    pload = round((float(row[4])/float(pTotal))*AreaConsumption,2)      #The proportion of the area consumption
                    qload = round((float(row[5])/float(pTotal))*AreaConsumption,2)      #The proportion of P and Q
                    listOfConsumptionData.append(LoadData(bus,ID,pload,qload))

def calculateHVDCData(saveHVDCtoPath,listOfHVDCData,HVDCDataForScaling):
    with open(saveHVDCtoPath) as csvfile:
        reader = csv.DictReader(csvfile,delimiter=';')

        # Running through each line of the file
        for row in reader:

            bus = int(row['Bus'])
            ID = int(row['ID'])
            pload = round(float(row['Flow']),2)      #The proportion of the area consumption
            qload = 0.0

            # busdata = [int(row['#BusNR']),int(row['ID']),int(row['Scalable']),int(row['Area']),float(row['Pload']),float(row['Qload'])
            for busdata in HVDCDataForScaling:
                if busdata[0] == bus:
                    if busdata[1] == ID:
                        if busdata[2] == 0:
                            #qload = round(abs(float(pload*busdata[5]/busdata[4])),2)
                            qload = round(float(pload*busdata[5]/busdata[4]),2)
                            #print('Bus %d: Pload: %g, Qload: %g' % (bus,pload,qload))
                            break
            listOfHVDCData.append(LoadData(bus,ID,pload,qload))



# For getting the production data:
def readingOperatingPointFromFile(saveProductionToPath):
    outfile=[]

    with open(saveProductionToPath) as csvfile:

        reader = csv.DictReader(csvfile,delimiter=';')
        for row in reader:
            outfile.append([int(row['Area']),float(row['Production']),float(row['Consumption'])])
    return outfile

# For getting the consumption data, will not copy non-scalable
def readingConsumptionOperatingPointFromFile(baseLoadPATH):
```

```
    outfile=[]

    with open(baseLoadPATH) as csvfile:

        #reader = csv.DictReader(csvfile,delimiter=';')
        #for row in reader:
        #    outfile.append([int(row['#BusNR']),int(row['ID']),int(row['Scalable']),int(row['Area']),float(row['Pload']),float(row['Qload'])])

        reader = csv.DictReader(csvfile,delimiter=';')
        for row in reader:
            if row['Scalable']=='1':
                outfile.append([int(row['#BusNR']),int(row['ID']),int(row['Scalable']),int(row['Area']),float(row['Pload']),float(row['Qload'])])


    return outfile

# For getting the HVDC data, will not copy scalable loads.
# The HVDC loads should be modelled as "Non-scalable" in the raw.file
def readingHVDCOperatingPointFromFile(baseLoadPATH):
    outfile=[]

    with open(baseLoadPATH) as csvfile:

        reader = csv.DictReader(csvfile,delimiter=';')
        for row in reader:
            if row['Scalable']=='0':
                outfile.append([int(row['#BusNR']),int(row['ID']),int(row['Scalable']),int(row['Area']),float(row['Pload']),float(row['Qload'])])
    return outfile


class GenUpdateData():
    def __init__(self, bus, ID, pgen, pmax,pmin,qmax,qmin,mbase,rsource,xsource, rtran, xtran,N):
        self.bus = bus
        self.ID = ID
        self.pgen = pgen
        self.pmax = pmax
        self.pmin = pmin
        self.qmax = qmax
        self.qmin = qmin
        self.mbase = mbase
        self.rsource = rsource
        self.xsource = xsource
```

```
        self.rtran = rtran
        self.xtran = xtran
        self.N = N

    #For writing the PSSE-code to filename f
    def pssewrite(self, f):
        #Command Python for changing machine:
        #   psspy.machine_chng_2(busNr,r"""ID""",[_i,_i,_i,_i,_i,_i],[ BusProduction, _f, Qmax,Qmin,Pmax,Pmin, Mbase, Rsource, Xsource, Rtran, Xtran,_f,_f,_f,_f,_f,_f])
        f.write('psspy.machine_chng_2(%d,r"""%d""",[_i,_i,_i,_i,_i,_i],[ %g, _f, %g,%g,%g,%g,%g,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f])\n' % (self.bus,self.ID,self.pgen,self.qmax,self.qmin,self.pmax,self.pmin,self.mbase))


    # The representation of the object if called
    def __repr__(self):
        return('(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)' % (self.bus, self.ID, self.pgen, self.pmax, self.pmin, self.qmax, self.qmin, self.mbase, self.rsource, self.xsource, self.rtran, self.xtran, self.N))

    # The string-output if print(object) is used, possibility to print PSSE-code here
    def __str__(self):
        #return str(self.__class__) + ": " + str(self.__dict__)
        return str(self.__dict__)

class LoadData():
    def __init__(self, bus, ID, Pload, Qload):
        self.bus = bus
        self.ID = ID
        self.pload = Pload
        self.qload = Qload

    def pssewrite(self, f):
        f.write('psspy.load_chng_4(%g,r"""%d""",[_i,_i,_i,_i,_i,_i],[ %g, %g,_f,_f,_f,_f])\n' % (self.bus,self.ID,self.pload,self.qload))

    def __repr__(self):
        return('(%s,%s,%s,%s)' % (self.bus, self.ID, self.pload, self.qload))

    def __str__(self):
        return str(self.__dict__)

def runScript(baseGeneratorPATH='AggregatedGenerator.csv', baseLoadPATH='LoadDataforScaling.csv',
           saveProductionToPath = 'currentProductionAndLoadForScaling.csv', saveHVDCtoPath = 'currentHVDCData.csv',productionData=0,consumptionData=0):
    # THE PROGRAM STARTS HERE     #
    ##############################
```

```

    #Getting production and consumption data
    if productionData == 0:
        productionData = readingOperatingPointFromFile(saveProductionToPath)
    #The basecase is used for scaling
    consumptionData = readingConsumptionOperatingPointFromFile(baseLoadPATH)

    # For scaling the P and Q
    HVDCDataForScaling = readingHVDCOperatingPointFromFile(baseLoadPATH)

    # Getting the base total consumption of an area
    sumOfActivePowerInArea = dict()
    for line in consumptionData:
        #Checking if the key is already in the dictionary
        if line[3] in sumOfActivePowerInArea:
            sumOfActivePowerInArea[line[3]] = round(sumOfActivePowerInArea[line[3]]+ float(line[4]),2)
        else:
            sumOfActivePowerInArea[line[3]] = float(line[4])

    # Saving all the updated generators in a list
    listOfProductionData = []
    listOfConsumptionData = []
    listOfHVDCData = []
    for row in productionData:
        # calculateGenData is using (AreaID, Production, listOfProductionData, generationDataPath)
        # Adds a GenUpdateData()-object into listOfProductionData
        calculateGenData(row[0],row[1],listOfProductionData,baseGeneratorPATH)
        calculateLoadData(row[0],row[2],listOfConsumptionData,baseLoadPATH,sumOfActivePowerInArea)

    calculateHVDCData(saveHVDCtoPath,listOfHVDCData,HVDCDataForScaling)

    # Generating the PSSE code, after everything is in order
    with open('psse.py', 'w') as f:
        #f.write('psspy.read(0,r"""C:\\NTNU\\python\\Nordic44xlsagr.raw""")'+'\n')

        # May be changed for PSSE-code
        #print('production elements:')
        for element in sorted(listOfProductionData, key=lambda obj: obj.bus):
            #print('Turned on %d generator at bus %d' % (element.N,element.bus))
            #print(element)
            element.pssewrite(f)

        #print('consumption elements:')
        for element in sorted(listOfConsumptionData, key=lambda obj: obj.bus):
            #print(element)
```

```
            element.pssewrite(f)

        #print('HVDC elements')
        for element in sorted(listOfHVDCData, key=lambda obj: obj.bus):
            #print(element)
            element.pssewrite(f)

        #Solving power flo
        f.write('psspy.scal(0,1,1,[0,0,0,0],[0.0,0.0,0.0,0.0,0.0,0.0,0.0])\n')

        #The system is scaled down with a constant of 718 MW in all of the system
        #to remedy for losses in the system.
        f.write('psspy.scal(0,1,2,[3,0,5,0],[-718.0,0.0,0.0,0.0,0.0,0.0,0.0])\n')
        f.write('psspy.fdns([0,0,0,1,1,0,99,0])\n')


    #print('End of Writing PSSE Files for Simulation')

    #Generation file format:
    # Bus    Pgen    Pmax    Pmin    Qmax    Qmin    Mbase   R_source    X_source    Rtran   Xtran   N
```

: files/generatePSSEcode.py

A.4 Running PSS/E code

```
def run(inputRAWfile='Nordic44xlsagr.raw', outputRAWfile = 'Nordic44xlsagr_rtime.raw'):

    #Printing out the current time for using in PacDyn
    currentDateAndHourPATH = 'currentDateandTime2.txt'
    historicalRawFile = outputRAWfile
    with open(currentDateAndHourPATH, 'r') as f:
        for row in f:
            placeholder = row
            historicalRawFile = historicalRawFile[:-4] + placeholder.replace('.','_').replace(':','_').replace('-','_') + '.raw'
        #dateAsString = self.currentDateTime.strftime('%d.%m.%Y-%H:%M:%S')
        #f.write(dateAsString)
    print(historicalRawFile)


    #Preparing for recognizing psspy library from PSS/E
    PSSEVERSION=33
    if PSSEVERSION==34:
       import psse34              # it sets new path for psspy
    else:
        import sys, os
        PSSE_LOCATION = r"C:\PTI\PSSE33\PSSBIN"
        sys.path.append(PSSE_LOCATION)
        os.environ['PATH'] = os.environ['PATH'] + ';' + PSSE_LOCATION

    import psspy
    import redirect
    redirect.psse2py()
    psspy.psseinit(1000)

    #For supressing output of PSS/E
    psspy.report_output(6,'',[])
    psspy.progress_output(6,'',[])
    psspy.alert_output(6,'',[])
    psspy.prompt_output(6,'',[])



    psspy.read(0,inputRAWfile)
    from psspy import _i,_f
    execfile(r'psse.py')
    if psspy.solved() == 2:
        print("Blownup case")
```

```
    else:
        import datetime
        now=datetime.datetime.now()
        psspy.rawd_2(0,1,[1,1,1,0,0,0,0],0,outputRAWfile)
        # Creating history file
        #psspy.rawd_2(0,1,[1,1,1,0,0,0,0],0,historicalRawFile)

        t=datetime.datetime.now()-now
        print('====>Time for writting: %g' % t.total_seconds())

run()
```

: files/runpssepy.py

A.5 Post-simulation visualizatioin script

```

# coding: utf-8

# In[2]:

import ipywidgets as widgets
from IPython.display import display

modes=['Mode1', 'Mode2', 'Mode3', 'Mode4']
contingencies=['Normal Case ', 'LT_5100-6500', 'LT_5301-5304', 'SE1-FI        ', 'NO1-SE3       ']

selected_modes = widgets.SelectMultiple(
                    description="Modes",
                    options=modes,
                    selected_labels=modes
)
selected_contingencies = widgets.SelectMultiple(
      description="Contingencies",
      options=contingencies,
      selected_labels=contingencies
)
display(selected_modes)
display(selected_contingencies)


# In[3]:

import plotly
from plotly.offline import download_plotlyjs, init_notebook_mode, iplot # For plotting offline
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
from plotly.graph_objs import Scatter, Layout, Stream, Figure
import pandas as pd
import numpy as np
import time
import datetime
plotly.offline.init_notebook_mode()

# Setting the folder of the case ex: 'Case#01/'
case = 'Case#01/'

# Getting my stream-tokens to connect to the server
```

```
stream_ids = tls.get_credentials_file()['stream_ids']

# Get stream id from stream id list
stream_id = stream_ids[0]

# Make instance of stream id object
stream = Stream(
    token=stream_id,  # (!) link stream id to 'token' key
    maxpoints=24      # (!) keep a max of 24 pts on screen
)
# Getting the data
class pacdynplt:
    def __init__(self,filename):
        f=open(filename)
        self.n=int(f.readline())
        self.curve=''
        #print(n)
        self.curves=[]
        self.modes=[]
        self.contingencies=[]
        line = f.readline()
        self.curves.append('Time')
        for i in range(1,self.n):
            line = f.readline()
            self.curves.append(line[:-1])
            self.modes.append(int(line[5:9]))
            self.contingencies.append(line[10:-1])
        #print(curves)
        self.__m=[]
        #m.append(curve)
        for line in f.readlines():
            self.__m.append(list(map(float,line.split())))
        #print(m)
        f.close()
        self.x=[]
        for row in self.__m:
            self.x.append(row[0])
        self.makeheader()
    # For selecting a spesific row
    def select(self,i):
            self.y=[]
            for row in self.__m:
                self.y.append(row[i])
            self.curve=self.curves[i]
    def makeheader(self):
```

```
        self.header= ['Time']
        for i in range(0,len(self.contingencies)):
            self.header.append('Mode%d:%s' % (self.modes[i],self.contingencies[i]))
    def makePandaFrame(self):
        return pd.DataFrame(self.__m,columns=self.header)

dampplt = pacdynplt(case+r'fdm_monit_damp.plt')
freqplt=pacdynplt(case+r'fdm_monit_freq.plt')
ddamp = dampplt.makePandaFrame()
dfreq = freqplt.makePandaFrame()
## The Time sampling
#dftime = pd.read_csv('timeSampling.txt',delimiter=';')
#dftime['Dates']=pd.to_datetime(dftime['Historical'], format='%d.%m.%Y-%H:%M:%S')

numbersOfModes = len(ddamp.columns)-1
df = pd.merge(ddamp,dfreq,on='Time',suffixes=('_D','_I'))

#Finding minimum and maximum of the x-axis and y-axis
min_xaxis = float()
max_xaxis = float()

#Frequency columns:
# Read as: return col if it contains '_I', iterate throug every coloumn headers
freq_cols = [col for col in df.columns if '_I' in col]
print('Freq: min/max')
min_freq = df[freq_cols].min().min()
max_freq = df[freq_cols].max().max()
print(min_freq)
print(max_freq)

#Imaginary columns:
damp_cols = [col for col in df.columns if '_D' in col]
print('Damping: min/max')
min_damp = df[damp_cols].min().min()
max_damp = df[damp_cols].max().max()
print(min_damp)
print(max_damp)

#List for saving the various contingencies and modes
contingencies = []
modes = []
allContingencies=[]

#Calculating the x-coordinates based on damping and frequency
for contingency in dampplt.header:
```

```
    if 'Mode' not in contingency:
        pass
    else:
        columnString = contingency+'_x'
        damping = contingency+'_D'
        frequency = contingency+'_I'
        serie = -(0.01*df[damping]*df[frequency])/(np.sqrt(1-0.0001*df[damping]**2))
        if serie.min() < min_xaxis:
            min_xaxis = serie.min()
        if serie.max() > max_xaxis:
            max_xaxis = serie.max()
        df[columnString] = serie

        #Making a list of modes and contingencies
        a,b = contingency.split(':')
        if a not in modes:
            modes.append(a)
        if b not in contingencies:
            contingencies.append(b)
        if contingency not in allContingencies:
            allContingencies.append(contingency)

# Real-part columns:
real_cols = [col for col in df.columns if '_x' in col[-2:]] #Checking the suffix
print('Real: min/max')
min_real = df[real_cols].min().min()
max_real = df[real_cols].max().max()
print(min_real)
print(max_real)

## Appending the Dates into the df
#df['Dates']=dftime['Dates']
##### Getting the correct date and time
#### NEED TO BE UPDATED TO FIND THE CORRECT FILE IF MORE THAN THE ORIGINAL
def createTimeSamplefromMonitTime(case='Case#01/'):
    dates = getStartAndEndTimeFromMonitTime(case)
    # Adding the offset and the startdate into a new columns
    df['Dates']= dates[0] + pd.to_timedelta(df['Time'],unit='h')
    return dates

def getStartAndEndTimeFromMonitTime(case='Case#01/'):
    f = case + 'fdm_monit_time.txt'
    #f = pd.read_csv('Case#01/fdm_monit_time.txt',delimiter='\n')
    with open(f, 'r') as f:
        dates = []
```

```
        for row in f:
            dates.append(datetime.datetime.strptime(row[:-1],'%Y_%m_%d_%H_%M_%S'))
    return dates

startDateTime,endDateTime = createTimeSamplefromMonitTime()

import ipywidgets as widgets
from IPython.display import display

selected_modes = widgets.SelectMultiple(
                description="Modes",
                options=modes,
                selected_labels=modes
)
selected_contingencies = widgets.SelectMultiple(
    description="Contingencies",
    options=contingencies,
    selected_labels=contingencies
)
display(selected_modes)
display(selected_contingencies)


# In[4]:

print(selected_modes.value)
print(selected_contingencies.value)
listOfInterestingColumns = []
listOfInterestingColumns.append('Time')
listOfInterestingColumns.append('Dates')
for col in df.columns:
    for mode in selected_modes.value:
        if mode in col:
            for cont in selected_contingencies.value:
                if cont in col:
                    print(col)
                    listOfInterestingColumns.append(col)


# In[5]:

# Creating a reduced DataFrame based on the selected Modes and Contingencies
subdf = df[listOfInterestingColumns]

#Finding the reduced contingency list
```

```
reducedList=[]
for col in subdf.columns:
    if 'Mode' in col:
        if col[:-2] not in reducedList:
            reducedList.append(col[:-2])

#Scaling to the closes 0.1
scale_xaxis_max = np.ceil(max_real*10)/10
scale_xaxis_min = np.floor(min_real*10)/10
scale_yaxis_max = np.ceil(max_freq*10)/10
scale_yaxis_min = np.floor(min_freq*10)/10

# 5% damping line
dampingPercent = 0.05
x_dampingline = -scale_yaxis_max*dampingPercent/np.sqrt(1-dampingPercent**2)

# Creating the various plots from data
traces=[]
tracesfig2=[]
tracesfig3=[]
for contingency in reducedList:
    if 'Mode' in contingency:
        xcoord = contingency+'_x'
        frequency=contingency+'_I'
        damping=contingency+'_D'
        traces.append(Scatter(x=subdf[xcoord],
                        y=df[frequency],
                        text=subdf.Dates,
                        mode='markers',
                        marker=go.Marker(opacity=0.5),
                        name=contingency,
                        legendgroup=contingency
                        ))
        tracesfig2.append(Scatter(x=subdf['Dates'],
                                    y=df[frequency],
                                    name=contingency,
                                    legendgroup=contingency
                                   ))
        tracesfig3.append(Scatter(x=subdf['Dates'],
                                    y=subdf[damping],
                                    name=contingency,
                                    legendgroup=contingency,
                                    showlegend=False
                                   ))

```

```
layout = {

    'xaxis': {
        #'range': [0, 9],
        #'zeroline': False,,
        'title': 'Real Part'
    },
    'yaxis': {
        #'range': [0, 11],
        #'showgrid': False,
        'title': 'Imaginary part'
    },
    'width': 900,
    'height': 1000,
    'shapes': [
         # filled Yellow Triangle
            {
                'type': 'path',
                'path': ' M 0 0 L 0 %f L %f %f Z' % (scale_yaxis_max,x_dampingline,scale_yaxis_max),
                'fillcolor': 'rgba(255, 255, 0, 0.2)',
                'line': {
                    'color': 'rgba(255, 255, 0,0.3)',
                    'width': 1
                },
            },
        # filled Red Squared
        {
            'type': 'path',
            'path': ' M 0,0 L %f,0 L %f,%f L 0,%f Z' % (np.max(scale_xaxis_max,0),np.max(scale_xaxis_max,0),
    scale_yaxis_max,scale_yaxis_max),
            'fillcolor': 'rgba(255, 140, 184, 0.2)',
            'line': {
                'color': 'rgba(255, 140, 184,0.0)',
            },
        },

    ],
    'hovermode': 'closest'
}

fig= Figure(data=traces, layout=layout)
fig2= Figure(data=tracesfig2,layout={'title':'Damping', 'yaxis':{'title':'Damping'}})
fig3= Figure(data=tracesfig3,layout={'title':'Frequency'})
xaxis = go.XAxis(title='Real part, damping time constant [1/s]')
yaxis = go.YAxis(title='Imaginary part [Rad/s]')
```

```
fig['layout'].update(xaxis=xaxis,
                     yaxis=yaxis)

#fig2yaxis = go.YAxis(title='Damping %')
#fig2['layout'].update(yaxis=fig2yaxis)



subplotFig = tls.make_subplots(rows=2,cols=1,shared_xaxes=True,subplot_titles=('Damping','Frequency'))
for i in range (0,len(tracesfig2)):
    subplotFig.append_trace(tracesfig2[i],1,1)
for i in range(0,len(tracesfig3)):
    subplotFig.append_trace(tracesfig3[i],2,1)
subplotFig['layout']['xaxis1'].update(mirror='all')
subplotFig['layout']['yaxis1'].update(domain=[0,0.45], title='Rad/s')
subplotFig['layout']['yaxis2'].update(domain=[0.5,0.95], title='Damping % ratio')
subplotFig['layout'].update(legend=dict(traceorder='normal'))
subplotFig['layout'].update(height=1000)

url = plotly.offline.iplot(fig)
url = plotly.offline.iplot(subplotFig)


# In[6]:

df['Dates']


# In[ ]:
```

: files/NotebookRevisual.py

A.6 Python dependencies

Multiple packages have been used to develop the DSA tool presented in this thesis, and to run the code not only does one need Python 2.7, but multiple packages need also to be installed. The following packages should therefore be installed. Most of them could be installed through the package control "Pip".

Anaconda

Appendix B

Base Cases

B.1 Aggregated generators

#BusNR	ID	Area Name	BusPF	Pmax1	Pmin1	Qmax1	Qmin1	Mbase1	R_source_1	X_source_1	Rtran_1	Xtran_1	MinN	MaxN
3000	1	23	0.207856994	**875.25**	0	644.6666667	-644.6666667	**866.6666667**	0	0.225	0	0	1	4
3115	1	21	0.45357524	333.3333333	0	311	-311	366.6666667	0	0.23	0	0	1	9
3245	1	22	1	907.25	0	83.75	-83.75	1008	0	0.15385	0	0	1	8
3249	1	21	0.54642476	861	0	657.3333333	-657.3333333	904.6666667	0	0.21	0	0	1	10
3300	1	23	0.231033049	750	0	767	-767	1100	0	0.16	0	0	1	4
3359	1	23	0.561213885	811.3333333	0	702.1433673	-702.1433673	964.286415	0	0.19375	0	0	1	9
5100	1	11	0.461977186	15.49295775	0	13.49198035	-13.49198035	19.04750167	0	0.15135	0	0	1	133
5300	1	15	1	138.0816327	0	37.77788641	-37.77788641	167.067147	0	0.26	0	0	1	60
5400	1	12	0.209140027	17.1875	0	16.5137884	-16.5137884	22.47710088	0	0.16	0	0	1	204
5500	1	11	0.538022814	13.76344086	0	14.45780084	-14.45780084	17.46984268	0	0.22825	0	0	1	117
5600	1	12	0.199684827	16.40625	0	15.59628988	-15.59628988	30.27515094	0	0.28	0	0	1	235
6000	1	12	0.058869878	155	0	125	-125	170	0	0.28	0	0	1	8
6100	1	12	0.53241783	25	0	23.68423711	-23.68423711	32.63161558	0	0.18	0	0	1	318
6500	1	13	0.999590667	15.15151515	0	14.81478738	-14.81478738	20.37033265	0	0.15802	0	0	1	445
6700	1	14	1	38.98989899	0	20.00004443	-20.00004443	47.66677257	0	0.17062	0	0	1	120
7000	1	31	0.812795935	552.7894737	0	490.5386455	-490.5386455	688.1541042	0	0.225	0	0	1	19
7100	1	31	0.187088578	300	0	233.3333333	-233.3333333	333.3333333	0	0.15385	0	0	1	10
8500	1	24	1	1183	0	917	-917	1300	0	0.17062	0	0	1	6

Table B.1: The base case for scaling aggregated generators

B.2 Load and HVDC points

BusNR	ID	Scalable	Area	Pload	Qload
3000	1	1	23	4261.98	1701
3020	1	0	23	1219	616
3100	1	1	21	621	110
3115	1	1	21	621	650
3249	1	1	22	2265	650
3300	1	1	23	2434.72	800
3359	1	1	23	5843.32	2400
3360	1	0	23	-330	262
5100	1	1	11	1154.17	70
5300	1	1	15	2651	-70
5400	1	1	12	1149.77	100
5500	1	1	11	4406.84	400
5600	1	1	12	1349.72	250
5610	1	0	12	1412	363
5620	1	0	12	414	175
6100	1	1	12	2399.52	800
6500	1	1	13	3039	999
6700	1	1	14	2489	150
7000	1	1	31	7967.65	350
7010	1	0	31	-1219	600
7020	1	0	31	343	-4
7100	1	1	31	2863.36	400
8500	1	1	24	3720	1299
8600	1	0	24	546	10
8700	1	0	24	628	0

Table B.2: Base case, HVDC and load points. HVDC-points are marked with yellow, and modelled with putting "Scalable" to 0.

B.3 Dynamic file

```
3000 'GENROU' 1     5.0000      0.50000E-01   1.0000       0.50000E-01
       5.9700       0.0000       2.2200       2.1300       0.36000
      0.46800      0.22500      0.16875      0.10890       0.37795    /
/3000 'STAB2A' 1     1.0000       2.0000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
3000 'IEEET2' 1     0.0000       729.00      0.40000E-01   5.3200
      -4.0500       1.0000      0.44000      0.66700E-01   2.0000
      0.44000       6.5000      0.54000E-01   8.0000       0.20200    /
3000 'IEESGO' 1    0.10000E-01   0.0000      0.15000       0.30000
       8.0000      0.40000       0.0000      0.70000       0.43000
       1.0000       0.0000    /
3115 'GENSAL' 1     7.5700      0.45000E-01  0.10000        4.7410
       0.0000      0.94600      0.56500      0.29000       0.23000
      0.11077      0.10239      0.27420    /
/3115 'STAB2A' 1     1.0000       4.5000      0.87000        2.0000
/      0.87000E-01   1.0000      0.10000E-01  0.40000E-01/
3115 'SCRX'   1    0.25385       13.000       31.000       0.50000E-01
       0.0000       4.0000       0.0000       0.0000     /
3115 'HYGOV'  1    0.60000E-01  0.40000       5.0000       0.50000E-01
      0.20000      0.10000       1.0000       0.0000        1.0000
       1.0577      0.50000      0.10000    /
3245 'GENSAL' 1     5.0000      0.60000E-01  0.10000        3.3000
       0.0000      0.75000      0.50000      0.25000       0.15385
      0.11538      0.10239      0.27420    /
3245 'SCRX'   1    0.25385       13.000       31.000       0.50000E-01
       0.0000       4.0000       0.0000       0.0000     /
3245 'HYGOV'  1    0.60000E-01  0.40000       5.0000       0.50000E-01
      0.20000      0.10000       1.0000       0.0000        1.0000
```

```
       1.0100      0.50000      0.10000     /
3249 'GENSAL' 1     10.130      0.60000E-01  0.10000       4.5430
       0.0000      1.0360      0.63000      0.28000      0.21000
      0.11538      0.10239      0.27420     /
3249 'SCRX'   1    0.25385      13.000       31.000      0.50000E-01
       0.0000      4.0000       0.0000       0.0000     /
3249 'HYGOV'  1    0.60000E-01  0.40000      5.0000      0.50000E-01
      0.20000      0.10000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000     /
3300 'GENROU' 1     10.800      0.50000E-01   1.0000      0.50000E-01
       6.0000       0.0000       2.4200       2.0000      0.23000
      0.41080      0.16000      0.14812      0.10890      0.37795     /
/3300 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
3300 'SCRX'   1     0.0000      0.40000E-01   10.000      0.40000E-01
       0.0000       5.0000       0.0000       0.0000     /
3300 'IEESGO' 1    0.10000E-01   0.0000      0.15000      0.30000
       8.0000      0.40000       0.0000      0.70000      0.43000
       1.0000       0.0000     /
3359 'GENROU' 1     4.7500      0.50000E-01   1.0000      0.50000E-01
       4.8200       0.0000       2.1300       2.0300      0.31000
      0.40300      0.19370      0.14531      0.10890      0.37795     /
/3359 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.68000       1.0000      0.10000E-01  0.30000E-01/
3359 'SCRX'   1    0.20000       10.000       165.00      0.40000E-01
       0.0000       5.0000       0.0000       0.0000     /
3359 'IEESGO' 1    0.10000E-01   0.0000      0.15000      0.30000
       8.0000      0.40000       0.0000      0.70000      0.43000
       1.0000       0.0000     /
5100 'GENSAL' 1     4.9629      0.50000E-01  0.15000       3.9871
```

```
        0.0000      1.1332      0.68315     0.24302     0.15135
       0.13405     0.10000     0.30000    /
5100 'SEXS'   1    0.50000E-01   100.00      200.00      0.50000
        0.0000      4.0000    /
5100 'HYGOV'  1    0.60000E-01  0.40000      5.0000      0.50000E-01
       0.20000     0.20000      1.0000      0.0000       1.0000
        1.1000     0.50000     0.10000    /
5300 'GENSAL' 1     6.4000     0.50000E-01  0.15000      3.5000
        0.0000      1.1400     0.84000     0.34000     0.26000
       0.20000     0.10000     0.30000    /
/5300 'STAB2A' 1     1.0000      4.5000      0.0000      2.0000
/      0.55000      1.0000     0.10000E-01  0.30000E-01/
5300 'STAB1' 1     25.00      5.0000      4.200      0.03
       4.200      0.03      0.10000E-00/
5300 'SCRX'   1    0.25385      13.000      61.000      0.50000E-01
        0.0000      4.0000      0.0000      0.0000    /
5300 'HYGOV'  1    0.60000E-01  0.40000      5.0000      0.50000E-01
       0.20000     0.20000      1.0000      0.0000       1.0000
        1.1000     0.50000     0.10000    /
5400 'GENSAL' 1     6.5000     0.50000E-01  0.15000      4.1000
        0.0000      1.0200     0.63000     0.25000     0.16000
       0.13000     0.10000     0.30000    /
5400 'SEXS'   1    0.50000E-01   100.00      200.00      0.50000
        0.0000      4.0000    /
5400 'HYGOV'  1    0.60000E-01  0.40000      5.0000      0.50000E-01
       0.20000     0.20000      1.0000      0.0000       1.0000
        1.1000     0.50000     0.10000    /
5500 'GENSAL' 1     7.1980     0.50000E-01  0.15000      3.0000
        0.0000      1.2364     0.65567     0.37415     0.22825
       0.16194     0.10000     0.30000    /
```

```
5500 'SEXS'   1    0.50000E-01   100.00       200.00       0.50000
       0.0000       4.0000    /
5500 'HYGOV'  1    0.60000E-01  0.40000       5.0000       0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
5600 'GENSAL' 1     7.8500      0.50000E-01  0.15000       3.5000
       0.0000       1.0000      0.51325      0.38000      0.28000
      0.21000      0.10000      0.30000    /
5600 'STAB1' 1     26.97       3.0000      7.881      0.03
      7.881      0.03      0.10000E-00/
5600 'SCRX'   1    0.25385       13.000       61.000       0.50000E-01
       0.0000       4.0000       0.0000       0.0000    /
5600 'HYGOV'  1    0.60000E-01  0.30000       5.0000       0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
6000 'GENSAL' 1     9.7000      0.50000E-01  0.15000       3.5000
       0.0000       1.2800      0.94000      0.37000      0.28000
      0.20000      0.10000      0.30000    /
6000 'SEXS'   1     1.0000      0.10000       20.000       0.10000
      -4.0000       4.0000    /
6000 'HYGOV'  1    0.60000E-01  0.30000       5.0000       0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
6100 'GENSAL' 1     9.9000      0.50000E-01  0.15000       3.0000
       0.0000       1.2000      0.73000      0.37000      0.18000
      0.15000      0.10000      0.30000    /
/6100 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
6100 'SCRX'   1    0.25385       13.000       61.000       0.50000E-01
       0.0000       4.0000       0.0000       0.0000    /
```

```
6100 'HYGOV'  1    0.60000E-01  0.40000       5.0000      0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
6500 'GENSAL' 1     5.4855      0.50000E-01  0.15000       3.5580
       0.0000       1.0679      0.64200      0.23865      0.15802
      0.13514      0.10000      0.30000    /
6500 'SEXS'   1    0.50000E-01   100.00       200.00      0.50000
       0.0000       4.0000    /
6500 'HYGOV'  1    0.60000E-01  0.40000       5.0000      0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
6700 'GENSAL' 1     5.2400      0.50000E-01  0.15000       3.5920
       0.0000       1.1044      0.66186      0.25484      0.17062
      0.14737      0.10000      0.30000    /
/6700 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
6700 'STAB1' 1     5.442       3.0000      6.962      0.03
       6.962      0.03      0.10000E-00/
6700 'SCRX'   1    0.25385       13.000       61.000      0.50000E-01
       0.0000       4.0000       0.0000       0.0000    /
6700 'HYGOV'  1    0.60000E-01  0.40000       5.0000      0.50000E-01
      0.20000      0.20000       1.0000       0.0000       1.0000
       1.1000      0.50000      0.10000    /
7000 'GENROU' 1     10.000      0.50000E-01   1.0000      0.50000E-01
       5.5000       0.0000       2.2200       2.1300      0.36000
      0.46800      0.22500      0.16875      0.10890      0.37795    /
/7000 'STAB2A' 1     1.0000       1.0000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
7000 'STAB1' 1     5.192       5.0000     13.50      0.03
       13.50      0.03      0.10000E-00/
```

```
7000 'IEEET2' 1     0.0000       800.00      0.40000E-01   5.3200
      -4.0500       1.0000      0.44000      0.66700E-01   2.0000
      0.44000       6.5000      0.54000E-01   8.0000      0.20200    /
7000 'IEESGO' 1    0.10000E-01   0.0000      0.15000      0.30000
       8.0000      0.40000       0.0000      0.70000      0.43000
       1.0000       0.0000    /
7100 'GENSAL' 1     5.0000      0.60000E-01  0.10000       3.2000
       0.0000      0.75000      0.50000      0.25000      0.15385
      0.11538      0.10239      0.27420    /
/7100 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
7100 'SCRX'   1    0.25385       13.000       61.000      0.50000E-01
       0.0000       4.0000       0.0000       0.0000    /
7100 'HYGOV'  1    0.60000E-01  0.40000       5.0000      0.50000E-01
      0.20000      0.10000       1.0000       0.0000       1.0000
       1.0100      0.50000      0.10000    /
8500 'GENROU' 1     10.000      0.50000E-01   1.0000      0.50000E-01
       7.0000       0.0000       2.4200       2.0000      0.23000
      0.41080      0.17062      0.14812      0.10890      0.37795    /
/8500 'STAB2A' 1     1.0000       4.5000       0.0000       2.0000
/      0.55000       1.0000      0.10000E-01  0.30000E-01/
8500 'SCRX'   1     0.0000      0.40000E-01   10.000      0.40000E-01
       0.0000       5.0000       0.0000       0.0000    /
8500 'IEESGO' 1    0.10000E-01   0.0000      0.15000      0.30000
       8.0000      0.40000       0.0000      0.70000      0.43000
       1.0000       0.0000    /
```

B.4 Contingency file

```
(
( Detalhamento dos Campos
( Ident  = Identificacao da contingencia
(          Utilizado pelo PacDyn para determinar os arquivos associados aos pontos de operacao
(          que serao analisados na SSA (metodo dos nomogramas)
( TP     = Tipo de contingencia a ser simulada no ANATEM
(          No PacDyn apenas o tipo "ABCI" e utilizado para a identificacao dos dados da contingencia
( EL     = Barra DE da linha de transmissao cuja contingencia sera avaliada
( PA     = Barra PARA da linha de transmissao cuja contingencia sera avaliada
( NC     = Numero do circuito da linha de transmissao cuja contingencia sera avaliada
( FIMCTG = Indica o fim dos dados da contingencia
(
CONTINGENCIA
(    Ident   ) (-----------------------------------------------------------------)
LT_5100-6500  F berg - vre  Vinstra
(Tp) ( Tempo)( El )( Pa)Nc( Ex) ( % ) (ABS ) Gr Und        (Bl)P ( Rc ) ( Xc )
APCL 1.000000  5100 6500 1       0.50                                    0.0001
ABCI 1.100000  5100 6500 1
FIMCTG
((    Ident   ) (-----------------------------------------------------------------)
(LT_5304-5305 LT_5304-5305
((Tp) ( Tempo)( El )( Pa)Nc( Ex) ( % ) (ABS ) Gr Und        (Bl)P ( Rc ) ( Xc )
(APCL 1.000000  5304 5305 1       0.50                                    0.0001
(ABCI 1.100000  5304 5305 1
(FIMCTG
(    Ident   ) (-----------------------------------------------------------------)
LT_5301-5304 LT_5301-5304
(Tp) ( Tempo)( El )( Pa)Nc( Ex) ( % ) (ABS ) Gr Und        (Bl)P ( Rc ) ( Xc )
APCL 1.000000  5301 5304 1       0.50                                    0.0001
ABCI 1.100000  5301 5304 1
FIMCTG
(    Ident   ) (-----------------------------------------------------------------)
SE1-FI         SE1-FI
(Tp) ( Tempo)( El )( Pa)Nc( Ex) ( % ) (ABS ) Gr Und        (Bl)P ( Rc ) ( Xc )
APCL 1.000000  3115 7100 1       0.50                                    0.0001
ABCI 1.100000  3115 7100 1
FIMCTG
(    Ident   ) (-----------------------------------------------------------------)
NO1-SE3        NO1-SE3
(Tp) ( Tempo)( El )( Pa)Nc( Ex) ( % ) (ABS ) Gr Und        (Bl)P ( Rc ) ( Xc )
APCL 1.000000  5101 3359 1       0.50                                    0.0001
ABCI 1.100000  5101 3359 1
FIMCTG
```

```
((     Ident   ) (------------------------------------------------------------)
(LT_5301-6001  LT_5301-6001
((Tp)  ( Tempo)(  El  )(  Pa)Nc(  Ex)  ( % )  (ABS  )  Gr  Und            (Bl)P  (  Rc  )  (  Xc  )
(APCL  1.000000   5301  6001  1          0.50                                              0.0001
(ABCI  1.100000   5301  6001  1
(FIMCTG
FIM
```

: files/nordic44.ctg

Appendix C

System Files

C.1 Raw-file

```
0,   1000.00, 33, 0, 1, 50.00     / PSS(R)E-33.6    MON, MAY 09 2016  10:54
REDUCED NORDEL POWER SYSTEM MODEL
V 3.2, 12.08.97
  3000,'            ', 420.0000,2,  23,   1,   1,0.92761,   5.8273,1.10000,0.90000,1.10000,0.90000
  3020,'            ', 420.0000,1,  23,   1,   1,0.92083,   5.0096,1.10000,0.90000,1.10000,0.90000
  3100,'            ', 420.0000,1,  22,   1,   1,1.02520,   0.0345,1.10000,0.90000,1.10000,0.90000
  3115,'            ', 420.0000,2,  21,   1,   1,1.00000,  14.7999,1.10000,0.90000,1.10000,0.90000
  3200,'            ', 420.0000,1,  23,   1,   1,1.01081,  -2.0829,1.10000,0.90000,1.10000,0.90000
  3244,'            ', 300.0000,1,  22,   1,   1,0.94693,  32.3216,1.10000,0.90000,1.10000,0.90000
  3245,'            ', 420.0000,2,  22,   1,   1,0.94430,  33.1359,1.10000,0.90000,1.10000,0.90000
  3249,'            ', 420.0000,2,  21,   1,   1,1.00000,   8.1348,1.10000,0.90000,1.10000,0.90000
  3300,'            ', 420.0000,3,  23,   1,   1,1.00000,   0.0000,1.10000,0.90000,1.10000,0.90000
  3359,'            ', 420.0000,2,  23,   1,   1,1.00000,   1.5693,1.10000,0.90000,1.10000,0.90000
  3360,'            ', 135.0000,1,  23,   1,   1,0.99657,   2.0240,1.10000,0.90000,1.10000,0.90000
  3701,'            ', 300.0000,1,  21,   1,   1,0.99020,  12.8634,1.10000,0.90000,1.10000,0.90000
  5100,'            ', 300.0000,2,  16,   1,   1,0.96635,  16.2245,1.10000,0.90000,1.10000,0.90000
  5101,'            ', 420.0000,1,  16,   1,   1,0.94132,  17.0334,1.10000,0.90000,1.10000,0.90000
  5102,'            ', 420.0000,1,  16,   1,   1,0.90977,  28.7651,1.10000,0.90000,1.10000,0.90000
  5103,'            ', 420.0000,1,  16,   1,   1,0.89892,  30.5380,1.10000,0.90000,1.10000,0.90000
  5300,'            ', 300.0000,2,  15,   1,   1,0.99978,  76.5239,1.10000,0.90000,1.10000,0.90000
  5301,'            ', 420.0000,1,  15,   1,   1,0.92864,  60.4580,1.10000,0.90000,1.10000,0.90000
  5304,'            ', 420.0000,1,  15,   1,   1,0.88780,  44.8952,1.10000,0.90000,1.10000,0.90000
  5305,'            ', 420.0000,1,  15,   1,   1,0.90520,  46.8155,1.10000,0.90000,1.10000,0.90000
  5400,'            ', 300.0000,2,  12,   1,   1,1.00700,  41.1229,1.10000,0.90000,1.10000,0.90000
  5401,'            ', 420.0000,1,  12,   1,   1,0.97114,  34.7164,1.10000,0.90000,1.10000,0.90000
  5402,'            ', 420.0000,1,  12,   1,   1,0.99435,  41.6981,1.10000,0.90000,1.10000,0.90000
  5500,'            ', 300.0000,2,  11,   1,   1,1.00400,   9.8673,1.10000,0.90000,1.10000,0.90000
  5501,'            ', 420.0000,1,  11,   1,   1,0.99740,  11.5690,1.10000,0.90000,1.10000,0.90000
  5600,'            ', 300.0000,2,  13,   1,   1,1.01000,  35.4190,1.10000,0.90000,1.10000,0.90000
  5601,'            ', 300.0000,1,  13,   1,   1,1.00428,  41.9598,1.10000,0.90000,1.10000,0.90000
  5602,'            ', 420.0000,1,  13,   1,   1,0.94069,  23.2151,1.10000,0.90000,1.10000,0.90000
  5603,'            ', 300.0000,1,  13,   1,   1,0.90940,  21.8044,1.10000,0.90000,1.10000,0.90000
  5610,'            ', 300.0000,1,  13,   1,   1,0.90525,  20.8216,1.10000,0.90000,1.10000,0.90000
  5620,'            ', 300.0000,1,  13,   1,   1,1.00826,  35.1860,1.10000,0.90000,1.10000,0.90000
```

```
  6000,'            ', 300.0000,2,  14,   1,   1,0.99256,  43.2509,1.10000,0.90000,1.10000,0.90000
  6001,'            ', 420.0000,1,  14,   1,   1,0.98625,  42.1113,1.10000,0.90000,1.10000,0.90000
  6100,'            ', 300.0000,2,  14,   1,   1,1.00000,  85.7523,1.10000,0.90000,1.10000,0.90000
  6500,'            ', 300.0000,2,  17,   1,   1,1.00000,  25.2866,1.10000,0.90000,1.10000,0.90000
  6700,'            ', 300.0000,2,  18,   1,   1,1.02000,  32.1411,1.10000,0.90000,1.10000,0.90000
  6701,'            ', 420.0000,1,  18,   1,   1,1.00390,  31.3250,1.10000,0.90000,1.10000,0.90000
  7000,'            ', 420.0000,2,  32,   1,   1,1.00000,   7.2874,1.10000,0.90000,1.10000,0.90000
  7010,'            ', 420.0000,1,  32,   1,   1,0.99389,   7.9901,1.10000,0.90000,1.10000,0.90000
  7020,'            ', 420.0000,1,  32,   1,   1,1.00003,   7.0908,1.10000,0.90000,1.10000,0.90000
  7100,'            ', 420.0000,2,  31,   1,   1,1.00000,   7.1215,1.10000,0.90000,1.10000,0.90000
  8500,'            ', 420.0000,2,  24,   1,   1,0.95815, -11.8643,1.10000,0.90000,1.10000,0.90000
  8600,'            ', 420.0000,1,  24,   1,   1,0.95803, -12.2052,1.10000,0.90000,1.10000,0.90000
  8700,'            ', 420.0000,1,  24,   1,   1,0.95813, -12.2563,1.10000,0.90000,1.10000,0.90000
0 / END OF BUS DATA, BEGIN LOAD DATA
  3000,'1 ',1,  23,   1,  4199.819,  1701.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3020,'1 ',1,  23,   1,  1219.000,   616.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  3100,'1 ',1,  22,   1,   611.944,   110.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3115,'1 ',1,  21,   1,   611.944,   650.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3249,'1 ',1,  21,   1,  2231.971,   650.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3300,'1 ',1,  23,   1,  2399.216,   800.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3359,'1 ',1,  23,   1,  5758.100,  2400.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  3360,'1 ',1,  23,   1,  -330.000,   262.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  5100,'1 ',1,  16,   1,  1137.339,    70.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  5300,'1 ',1,  15,   1,  2612.342,   -70.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  5400,'1 ',1,  12,   1,  1133.003,   100.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  5500,'1 ',1,  11,   1,  4342.567,   400.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  5600,'1 ',1,  13,   1,  1330.038,   250.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  5610,'1 ',1,  13,   1,  1412.000,   363.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  5620,'1 ',1,  13,   1,   414.000,   175.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  6100,'1 ',1,  14,   1,  2364.529,   800.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  6500,'1 ',1,  17,   1,  2994.684,   999.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  6700,'1 ',1,  18,   1,  2452.704,   150.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  7000,'1 ',1,  32,   1,  7851.452,   350.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  7010,'1 ',1,  32,   1, -1219.000,   600.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  7020,'1 ',1,  32,   1,   343.000,    -4.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  7100,'1 ',1,  31,   1,  2821.605,   400.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  8500,'1 ',1,  24,   1,  3665.753,  1299.000,     0.000,     0.000,     0.000,     0.000,   1,1,0
  8600,'1 ',1,  24,   1,   546.000,    10.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
  8700,'1 ',1,  24,   1,   628.000,     0.000,     0.000,     0.000,     0.000,     0.000,   1,0,0
0 / END OF LOAD DATA, BEGIN FIXED SHUNT DATA
0 / END OF FIXED SHUNT DATA, BEGIN GENERATOR DATA
  3000,'1 ',  2000.000,  1934.000,  1934.000, -1934.000,1.00000,     0,  2900.000, 0.00000E+0, 2.25000E-1,
     0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  2625.750,     0.000,   1,1.0000
  3115,'1 ',  1700.000,   358.659,  1866.000, -1866.000,1.00000,     0,  2200.000, 0.00000E+0, 2.30000E-1,
     0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  2000.000,     0.000,   1,1.0000
```

```
3245,'1 ',  6599.000,   670.000,   670.000,  -670.000,1.00000,     0,  8064.000, 0.00000E+0, 1.53850E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  7258.000,     0.000,   1,1.0000
3249,'1 ',  2048.000,   485.973,  1972.000, -1972.000,1.00000,     0,  2714.000, 0.00000E+0, 2.10000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  2583.000,     0.000,   1,1.0000
3300,'1 ',  2223.852,  2883.988,  2301.000, -2301.000,1.00000,     0,  3300.000, 0.00000E+0, 1.60000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  2250.000,     0.000,   1,1.0000
3359,'1 ',  5400.000,  3281.928,  4915.000, -4915.000,1.00000,     0,  6750.000, 0.00000E+0, 1.93750E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  5679.330,     0.000,   1,1.0000
5100,'1 ',   972.000,   849.990,   849.990,  -849.990,1.00000,     0,  1199.990, 0.00000E+0, 1.51350E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   976.060,     0.000,   1,1.0000
5300,'1 ',  6151.000,  1700.000,  1700.000, -1700.000,1.00000,     0,  7518.020, 0.00000E+0, 2.60000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  6213.670,     0.000,   1,1.0000
5400,'1 ',  1858.000,  1409.123,  1800.000, -1800.000,1.00700,     0,  2450.000, 0.00000E+0, 1.60000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  1873.440,     0.000,   1,1.0000
5500,'1 ',  1132.000,   601.322,  1200.000, -1200.000,1.00400,     0,  1450.000, 0.00000E+0, 2.28250E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  1142.370,     0.000,   1,1.0000
5600,'1 ',  1774.000,  1123.709,  1700.000, -1700.000,1.01000,     0,  3299.990, 0.00000E+0, 2.80000E-1,
   0.00000E+0, 0.00000E+0,0.99900,1,  100.0,  2970.000,     0.000,   1,1.0000
6000,'1 ',   523.000,   500.000,   500.000,  -500.000,1.00500,     0,   680.000, 0.00000E+0, 2.80000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,   620.000,     0.000,   1,1.0000
6100,'1 ',  4730.000,  1276.413,  4500.010, -4500.010,1.00000,     0,  6200.010, 0.00000E+0, 1.80000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  4750.000,     0.000,   1,1.0000
6500,'1 ',  2442.000,  1190.436,  2400.000, -2400.000,1.00000,     0,  3299.990, 0.00000E+0, 1.58020E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  2454.550,     0.000,   1,1.0000
6700,'1 ',  3506.000,    66.699,  1800.000, -1800.000,1.02000,     0,  4290.010, 0.00000E+0, 1.70620E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  3509.090,     0.000,   1,1.0000
7000,'1 ',  7038.000,   751.025,  6377.000, -6377.000,1.00000,     0,  8946.000, 0.00000E+0, 2.25000E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  7186.260,     0.000,   1,1.0000
7100,'1 ',  1620.000,   533.564,  1400.000, -1400.000,1.00000,     0,  2000.000, 0.00000E+0, 1.53850E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  1800.000,     0.000,   1,1.0000
8500,'1 ',   754.000,   917.000,   917.000,  -917.000,1.02000,     0,  1300.000, 0.00000E+0, 1.70620E-1,
   0.00000E+0, 0.00000E+0,1.00000,1,  100.0,  1183.000,     0.000,   1,1.0000
0 / END OF GENERATOR DATA, BEGIN BRANCH DATA
3000,  3020,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
3000,  3115,'1 ', 7.50000E-2, 9.00000E-1,   0.50000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
3000,  3245,'1 ', 8.00000E-3, 1.20000E-1,   0.05000, 1200.00, 1600.00, 1800.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
3000,  3245,'2 ', 1.80000E-2, 2.00000E-1,   0.05000,  800.00, 1300.00, 1600.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
3000,  3300,'1 ', 6.00000E-3, 8.00000E-2,   0.03000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
3000,  3300,'2 ', 9.00000E-3, 1.00000E-1,   0.02500, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
```

```
3100, 3115,'1 ', 3.00000E-2, 4.00000E-1,  0.11000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,2,   0.00,   1,1.0000
3100, 3200,'1 ', 4.00000E-2, 2.40000E-1,  0.20000, 1200.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3100, 3200,'2 ', 4.00000E-2, 2.40000E-1,  0.20000, 1200.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3100, 3200,'3 ', 4.00000E-2, 2.40000E-1,  0.20000, 1200.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3100, 3249,'1 ', 3.00000E-2, 4.30000E-1,  0.16000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,2,   0.00,   1,1.0000
3100, 3359,'1 ', 8.00000E-2, 5.00000E-1,  0.25000,  900.00, 1300.00, 1600.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3100, 3359,'2 ', 4.00000E-2, 2.30000E-1,  0.24000, 1200.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3115, 3245,'1 ', 4.50000E-2, 5.00000E-1,  0.14000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3115, 3249,'1 ', 1.50000E-2, 2.00000E-1,  0.08000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3115, 6701,'1 ', 4.00000E-2, 4.00000E-1,  0.10000,  850.00, 1000.00, 1100.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,2,   0.00,   1,1.0000
3115, 7100,'1 ', 4.00000E-2, 1.30000E-1,  0.13000, 1300.00, 1500.00, 1700.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3200, 3300,'1 ', 2.00000E-2, 2.00000E-1,  0.06000,  800.00, 1100.00, 1300.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3200, 3359,'1 ', 1.00000E-2, 2.00000E-1,  0.07000, 1300.00, 1800.00, 2000.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3200, 8500,'1 ', 1.00000E-2, 1.70000E-1,  0.06000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3244, 6500,'1 ', 1.00000E-2, 2.00000E-1,  0.06000, 1800.00, 2300.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,2,   0.00,   1,1.0000
3249, 7100,'1 ', 2.00000E-2, 7.50000E-2,  0.07800, 1300.00, 1500.00, 1700.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3300, 8500,'1 ', 2.00000E-2, 2.30000E-1,  0.06000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3300, 8500,'2 ', 1.20000E-2, 2.70000E-1,  0.10000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3359, 5101,'1 ', 1.60000E-2, 2.60000E-1,  0.09000, 1900.00, 2200.00, 2600.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,2,   0.00,   1,1.0000
3359, 5101,'2 ', 2.00000E-2, 2.20000E-1,  0.06000, 1600.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3359, 8500,'1 ', 1.20000E-2, 2.70000E-1,  0.10000, 1500.00, 2000.00, 2500.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
3359, 8500,'2 ', 2.50000E-2, 3.20000E-1,  0.09000, 1100.00, 1300.00, 1400.00,  0.00000,  0.00000,  0.00000,
    0.00000,1,1,   0.00,   1,1.0000
```

```
3701, 6700,'1 ', 2.50000E-1, 2.00000E+0, 0.03000, 300.00, 400.00, 500.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5100, 5500,'1 ', 2.70000E-2, 2.60000E-1, 0.04400, 700.00, 800.00, 900.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5100, 6500,'1 ', 8.00000E-2, 9.00000E-1, 0.06000, 800.00, 900.00, 950.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5101, 5102,'1 ', 8.00000E-3, 1.00000E-1, 0.09000, 1700.00, 1800.00, 1900.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5101, 5103,'1 ', 1.00000E-2, 1.40000E-1, 0.04000, 1350.00, 1600.00, 1800.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5101, 5501,'1 ', 1.00000E-2, 1.50000E-1, 0.55000, 2000.00, 2200.00, 2500.00, 0.02230, -0.97440, -0.02160, 0.97440,1,2, 0.00, 1,1.0000
5102, 5103,'1 ', 4.00000E-3, 7.00000E-2, 0.03000, 2000.00, 2200.00, 2400.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5102, 5304,'1 ', 1.70000E-2, 2.40000E-1, 0.07000, 1500.00, 1800.00, 2000.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5102, 6001,'1 ', 3.00000E-2, 4.60000E-1, 0.13000, 1450.00, 1700.00, 2000.00, 0.00020, 0.00010, 0.00020, -0.00010,1,2, 0.00, 1,1.0000
5103, 5304,'1 ', 2.00000E-2, 2.50000E-1, 0.07000, 1000.00, 1200.00, 1400.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5103, 5304,'2 ', 1.30000E-2, 2.00000E-1, 0.06000, 1500.00, 1800.00, 2000.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5300, 6100,'1 ', 2.10000E-2, 2.20000E-1, 0.01000, 800.00, 900.00, 950.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5301, 5304,'1 ', 1.00000E-2, 2.00000E-1, 0.06000, 1250.00, 1500.00, 1700.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5301, 5305,'1 ', 7.00000E-3, 1.20000E-1, 0.03100, 1250.00, 1500.00, 1700.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5301, 6001,'1 ', 1.30000E-2, 2.00000E-1, 0.05000, 1800.00, 2000.00, 2150.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5304, 5305,'1 ', 1.00000E-2, 1.50000E-1, 0.05000, 1950.00, 2200.00, 2400.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5304, 5305,'2 ', 1.30000E-2, 1.70000E-2, 0.04000, 1350.00, 1400.00, 1500.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5400, 5500,'1 ', 9.00000E-3, 9.40000E-1, 0.05000, 400.00, 550.00, 650.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5400, 6000,'1 ', 3.30000E-2, 3.60000E-1, 0.02500, 800.00, 900.00, 950.00, 0.00000, 0.00000, 0.00000, 0.00000,1,2, 0.00, 1,1.0000
5401, 5501,'1 ', 1.75000E-2, 2.70000E-1, 0.08000, 1500.00, 1800.00, 2000.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5401, 5602,'1 ', 1.60000E-2, 2.55000E-1, 0.09000, 2200.00, 2400.00, 2600.00, 0.00000, 0.00000, 0.00000, 0.00000,1,1, 0.00, 1,1.0000
5401, 6001,'1 ', 6.40000E-3, 1.00000E-1, 0.02800, 1250.00, 1500.00, 1700.00, -0.00020, -0.00050, 0.00020, 0.00050,1,2, 0.00, 1,1.0000
```

```
  5402,  6001,'1 ', 7.00000E-4, 1.00000E-2,   0.00300, 1500.00, 1700.00, 1900.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
  5500,  5603,'1 ', 5.00000E-2, 6.00000E-1,   0.05000,  800.00,  900.00,  950.00,  0.00030,  0.00130, -0.00030,
    -0.00130,1,1,   0.00,   1,1.0000
  5600,  5601,'1 ', 3.00000E-2, 3.40000E-1,   0.02000,  800.00,  900.00,  950.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
  5600,  5603,'1 ', 2.00000E-2, 2.20000E-1,   0.02000,  900.00, 1050.00, 1200.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  5600,  5620,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  5600,  6000,'1 ', 2.00000E-2, 2.00000E-1,   0.07000, 1350.00, 1500.00, 1650.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  5603,  5610,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  6000,  6100,'1 ', 3.40000E-2, 4.20000E-1,   0.03000,  800.00,  900.00,  950.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  6500,  6700,'1 ', 1.70000E-1, 1.80000E+0,   0.10000,  800.00,  900.00,  950.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  6500,  6700,'2 ', 1.00000E-1, 1.30000E+0,   0.12000, 1000.00, 1200.00, 1300.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  7000,  7010,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  7000,  7020,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  7000,  7100,'1 ', 4.00000E-2, 1.20000E-1,   0.13000, 1040.00, 1200.00, 1500.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
  7000,  7100,'2 ', 4.00000E-2, 1.20000E-1,   0.13000, 1040.00, 1200.00, 1500.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
  7000,  7100,'3 ', 4.00000E-2, 1.40000E-1,   0.13000, 1200.00, 1500.00, 1700.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,2,   0.00,   1,1.0000
  8500,  8600,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
  8500,  8700,'1 ', 0.00000E+0, 1.00000E-2,   0.00000,    0.00,    0.00,    0.00,  0.00000,  0.00000,  0.00000,
     0.00000,1,1,   0.00,   1,1.0000
0 / END OF BRANCH DATA, BEGIN TRANSFORMER DATA
  3244,  3245,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,
      0,1.0000,'            '
 5.00000E-3, 2.00000E-2,  1000.00
1.00000,   0.000,   0.000,   500.00,   500.00,     0.00, 1,   3245, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0,
    0.00000, 0.00000,  0.000
1.00000,   0.000
  3701,  3249,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,
      0,1.0000,'            '
 2.00000E-2, 5.00000E-1,  1000.00
```

```
1.00000,    0.000,    0.000,   300.00,   350.00,     0.00, 1,   3701, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  3359,  3360,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 5.00000E-3, 2.00000E-2,  1000.00
0.99980,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   3360, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5101,  5100,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 8.00000E-4, 3.05000E-2,  1000.00
1.00635,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   5101, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5300,  5301,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 1.60000E-3, 6.10000E-2,  1000.00
1.00000,    0.000,    0.000,  2000.00,  9000.00,  9000.00, 1,   5301, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5400,  5401,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 3.20000E-3, 1.20000E-1,  1000.00
1.00635,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   5401, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5400,  5402,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 4.00000E-4, 1.50000E-2,  1000.00
1.00000,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   5402, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5500,  5501,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 4.00000E-4, 1.50000E-2,  1000.00
1.01260,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   5501, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
  5601,  6001,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 2.00000E-4, 7.60000E-3,  1000.00
1.01806,    0.000,    0.000,  1000.00,  9000.00,  9000.00, 1,   5601, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,    0.000
```

```
  5603,  5602,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 8.00000E-4, 3.05000E-2,  1000.00
0.96825,   0.000,   0.000,  1000.00,  9000.00,  9000.00, 1,   5602, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
  6000,  6001,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 4.00000E-4, 1.50000E-2,  1000.00
1.00625,   0.000,   0.000,  1000.00,  9000.00,  9000.00, 1,   6001, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
  6700,  6701,     0,'1 ',1,1,1, 0.00000E+0, 0.00000E+0,2,'            ',1,   1,1.0000,   0,1.0000,   0,1.0000,   0,1.0000,'            '
 5.00000E-3, 2.00000E-2,  1000.00
1.01250,   0.000,   0.000,  1000.00,  9000.00,  9000.00, 1,   6701, 1.40000, 0.60000, 1.01000, 0.99000, 127, 0, 0.00000, 0.00000,  0.000
1.00000,   0.000
0 / END OF TRANSFORMER DATA, BEGIN AREA DATA
  11,     0,     0.000,    10.000,'NO1         '
  12,     0,     0.000,    10.000,'NO2         '
  13,     0,     0.000,    10.000,'NO3         '
  14,     0,     0.000,    10.000,'NO4         '
  15,     0,     0.000,    10.000,'NO5         '
  16,     0,     0.000,    10.000,'NO6         '
  17,     0,     0.000,    10.000,'NO7         '
  18,     0,     0.000,    10.000,'NO8         '
  21,     0,     0.000,    10.000,'SE1         '
  22,     0,     0.000,    10.000,'SE2         '
  23,     0,     0.000,    10.000,'SE3         '
  24,     0,     0.000,    10.000,'SE4         '
  31,     0,     0.000,    10.000,'FI1         '
  32,     0,     0.000,    10.000,'FI2         '
0 / END OF AREA DATA, BEGIN TWO-TERMINAL DC DATA
0 / END OF TWO-TERMINAL DC DATA, BEGIN VSC DC LINE DATA
0 / END OF VSC DC LINE DATA, BEGIN IMPEDANCE CORRECTION DATA
0 / END OF IMPEDANCE CORRECTION DATA, BEGIN MULTI-TERMINAL DC DATA
0 / END OF MULTI-TERMINAL DC DATA, BEGIN MULTI-SECTION LINE DATA
0 / END OF MULTI-SECTION LINE DATA, BEGIN ZONE DATA
0 / END OF ZONE DATA, BEGIN INTER-AREA TRANSFER DATA
0 / END OF INTER-AREA TRANSFER DATA, BEGIN OWNER DATA
0 / END OF OWNER DATA, BEGIN FACTS DEVICE DATA
0 / END OF FACTS DEVICE DATA, BEGIN SWITCHED SHUNT DATA
0 / END OF SWITCHED SHUNT DATA, BEGIN GNE DATA
0 / END OF GNE DATA, BEGIN INDUCTION MACHINE DATA
```

```
0 / END OF INDUCTION MACHINE DATA
Q
```

: files/Nordic44xlsagr.raw

C.2 High load situation, PSS/E files

Bus Number	Bus Name	Id	Area Num	Area Name	PGen (MW)	PMax (MW)	Mbase (MVA)	QGen (Mvar)	QMax (Mvar)	QMin (Mvar)	X Source (pu)
3000	420,00	1	23	SE3	2099,5600	2625,7500	2600,00	1684,7050	1934,0000	-1934,0000	0,225000
3115	420,00	1	21	SE1	638,1800	666,6700	733,33	-84,9730	622,0000	-622,0000	0,230000
3245	420,00	1	22	SE2	4858,0000	5443,5000	6048,00	109,5060	502,5000	-502,5000	0,153850
3249	420,00	1	21	SE1	768,8200	861,0000	904,67	472,2700	657,3300	-657,3300	0,210000
3300	420,00	1	23	SE3	1904,4620	3000,0000	4400,00	280,1120	3068,0000	-3068,0000	0,160000
3359	420,00	1	23	SE3	5668,8200	5679,3300	6750,00	996,4260	4915,0000	-4915,0000	0,193750
5100	300,00	1	16	NO6	822,7800	836,6200	1028,57	309,5990	728,5700	-728,5700	0,151350
5300	300,00	1	15	NO5	3329,0000	3452,0400	4176,68	-84,5270	944,4500	-944,4500	0,260000
5400	300,00	1	12	NO2	1117,4400	1134,3800	1483,49	189,5180	1089,9100	-1089,9100	0,160000
5500	300,00	1	11	NO1	958,2200	963,4400	1222,89	-646,0630	1012,0500	-1012,0500	0,228250
5600	300,00	1	13	NO3	1066,9200	1082,8100	1998,16	444,5840	1029,3600	-1029,3600	0,280000
6000	300,00	1	14	NO4	314,5400	465,0000	510,00	-268,4790	375,0000	-375,0000	0,280000
6100	300,00	1	14	NO4	2844,7100	2850,0000	3720,00	640,5250	2700,0000	-2700,0000	0,180000
6500	300,00	1	17	NO7	1403,4300	1409,0900	1894,44	927,2660	1377,7800	-1377,7800	0,158020
6700	300,00	1	18	NO8	2125,0000	2144,4400	2621,67	5,9160	1100,0000	-1100,0000	0,170620
7000	420,00	1	32	FI2	6283,7300	6633,4700	8257,85	175,7980	5886,4600	-5886,4600	0,225000
7100	420,00	1	31	FI1	1446,3800	1500,0000	1666,67	400,8600	1166,6700	-1166,6700	0,153850
8500	420,00	1	24	SE4	1096,0000	1183,0000	1300,00	917,0000	917,0000	-917,0000	0,170620

Table C.1: Machine parameters during high load situation,2015 March 3th 08:00

Area to from	NO1	NO2	NO3	NO4	NO5	NO6	NO7	NO8	SE1	SE2	SE3	SE4	FI1	FI2	Total From
NO1		-832	-202			-1182									-2216
NO2	832		198	-841											189
NO3	202	-198		-592											-588
NO4		841	592		-313	88									1208
NO5				313		975									1288
NO6	1182			-88	-975		-14				-133				-28
NO7						14		87		-1267					-1166
NO8							-87		88						1
SE1								-88		-1057	-169		356		-958
SE2							1267		1057		2017				4341
SE3						133			169	-2017		1611			-104
SE4											-1611				-1611
FI1									-356					-629	-985
FI2													629		629

Table C.2: PSS/E area flow during the high load situation at, 2015 March 3th 08:00

Bus Number	Bus Name	Area Num	Area Name	Scalable	Pload (MW)	Qload (Mvar)
3000	420,00	23	SE3	1	3194,4040	1298,6700
3020	420,00	23	SE3	0	507,0000	256,2000
3100	420,00	22	SE2	1	465,3300	83,9600
3115	420,00	21	SE1	1	465,3300	496,1400
3249	420,00	21	SE1	1	1894,6990	553,8600
3300	420,00	23	SE3	1	1824,8500	610,7800
3359	420,00	23	SE3	1	4379,6310	1832,3400
3360	135,00	23	SE3	0	-195,0000	154,8200
5100	300,00	16	NO6	1	829,2690	51,2300
5300	300,00	15	NO5	1	2023,3030	-54,4200
5400	300,00	12	NO2	1	926,2130	82,0600
5500	300,00	11	NO1	1	3166,2880	292,7500
5600	300,00	13	NO3	1	1087,2920	205,1400
5610	300,00	13	NO3	0	151,0000	38,8200
5620	300,00	13	NO3	0	413,0000	174,5800
6100	300,00	14	NO4	1	1932,9660	656,4600
6500	300,00	17	NO7	1	2569,1340	860,2800
6700	300,00	18	NO8	1	2123,4380	130,3500
7000	420,00	32	FI2	1	6718,4260	300,6200
7010	420,00	32	FI2	0	-507,0000	249,5500
7020	420,00	32	FI2	0	-563,0000	6,5700
7100	420,00	31	FI1	1	2414,4160	343,5700
8500	420,00	24	SE4	1	2522,0110	897,0800
8600	420,00	24	SE4	0	179,0000	3,2800
8700	420,00	24	SE4	0	-3,0000	0,0000

Table C.3: PSS/E HVDC and load parameters during high load situation, 2015 March 3th 08:00

C.3 Low load situation

Bus Number	Bus Name	Id	Area Num	Area Name	PGen (MW)	PMax (MW)	Mbase (MVA)	QGen (Mvar)	QMax (Mvar)	QMin (Mvar)	X Source (pu)
3000	420,00	1	23	SE3	2077,9500	2625,7500	2600,00	1934,0000	1934,0000	-1934,0000	0,225000
3115	420,00	1	21	SE1	1719,5000	2000,0000	2200,00	216,5130	1866,0000	-1866,0000	0,230000
3245	420,00	1	22	SE2	6976,0000	7258,0000	8064,00	670,0000	670,0000	-670,0000	0,153850
3249	420,00	1	21	SE1	2071,5000	2583,0000	2714,00	442,2360	1972,0000	-1972,0000	0,210000
3300	420,00	1	23	SE3	2526,4400	3000,0000	4400,00	2324,6320	3068,0000	-3068,0000	0,160000
3359	420,00	1	23	SE3	5610,4600	5679,3300	6750,00	2711,3880	4915,0000	-4915,0000	0,193750
5100	300,00	1	16	NO6	978,0100	991,5500	1219,04	863,4900	863,4900	-863,4900	0,151350
5300	300,00	1	15	NO5	6320,0000	6351,7600	7685,09	1737,7800	1737,7800	-1737,7800	0,260000
5400	300,00	1	12	NO2	1881,4200	1890,6300	2472,48	1737,1890	1816,5200	-1816,5200	0,160000
5500	300,00	1	11	NO1	1138,9900	1142,3700	1450,00	764,6360	1200,0000	-1200,0000	0,228250
5600	300,00	1	13	NO3	1796,3600	1804,6900	3330,27	1262,6800	1715,5900	-1715,5900	0,280000
6000	300,00	1	14	NO4	529,5900	620,0000	680,00	500,0000	500,0000	-500,0000	0,280000
6100	300,00	1	14	NO4	4789,6300	4800,0000	6265,27	1494,8140	4547,3700	-4547,3700	0,180000
6500	300,00	1	17	NO7	2146,1200	2151,5200	2892,59	1268,9810	2103,7000	-2103,7000	0,158020
6700	300,00	1	18	NO8	3119,0000	3119,1900	3813,34	21,5100	1600,0000	-1600,0000	0,170620
7000	420,00	1	32	FI2	7348,4900	7739,0500	9634,16	776,7400	6867,5400	-6867,5400	0,225000
7100	420,00	1	31	FI1	1691,4700	1800,0000	2000,00	549,8800	1400,0000	-1400,0000	0,153850
8500	420,00	1	24	SE4	1193,0000	2366,0000	2600,00	1834,0000	1834,0000	-1834,0000	0,170620

Table C.4: Machine parameters during low load sitation, 2015 March 3th 03:00

Area to from	NO1	NO2	NO3	NO4	NO5	NO6	NO7	NO8	SE1	SE2	SE3	SE4	FI1	FI2	Total From
NO1		-2024	-280			-1018									-3322
NO2	2024		776	-2104											696
NO3	280	-776		-1079											-1575
NO4		2104	1079		-887	459									2755
NO5				887		2753									3640
NO6	1018			-459	-2753		-103				1955				-342
NO7						103		-32		-918					-847
NO8							32		542						574
SE1								-542		115	100		1238		911
SE2							918		-115		-5544				-4741
SE3						-1955			-100	-5544		4083			-3516
SE4											-4083				-4083
FI1									-1238					-14	-1252
FI2													14		14

Table C.5: PSS/E area flow during the low load situation at, 2015 March 3th 03:00

Bus Number	Bus Name	Area Num	Area Name	Scalable	Pload (MW)	Qload (Mvar)
3000	420,00	23	SE3	1	4241,7090	1717,6800
3020	420,00	23	SE3	0	1221,0000	617,0100
3100	420,00	22	SE2	1	578,5340	103,9800
3115	420,00	21	SE1	1	578,5340	614,4100
3249	420,00	21	SE1	1	2285,5550	665,5000
3300	420,00	23	SE3	1	2423,1420	807,8500
3359	420,00	23	SE3	1	5815,5290	2423,5400
3360	135,00	23	SE3	0	-208,0000	165,1400
5100	300,00	16	NO6	1	1156,5460	71,1700
5300	300,00	15	NO5	1	2512,2380	-67,3100
5400	300,00	12	NO2	1	1169,0430	103,1600
5500	300,00	11	NO1	1	4415,9100	406,6900
5600	300,00	13	NO3	1	1372,3380	257,9100
5610	300,00	13	NO3	0	1535,0000	394,6200
5620	300,00	13	NO3	0	413,0000	174,5800
6100	300,00	14	NO4	1	2439,7290	825,3100
6500	300,00	17	NO7	1	2992,2140	998,0100
6700	300,00	18	NO8	1	2543,7760	155,5400
7000	420,00	32	FI2	1	8072,4150	359,7900
7010	420,00	32	FI2	0	-1221,0000	600,9800
7020	420,00	32	FI2	0	483,0000	-5,6300
7100	420,00	31	FI1	1	2901,0090	411,1900
8500	420,00	24	SE4	1	3567,7910	1264,0800
8600	420,00	24	SE4	0	643,0000	11,7800
8700	420,00	24	SE4	0	1014,0000	0,0000

Table C.6: PSS/E HVDC and load parameters during low load situation, 2015 March 3th 03:00

Bibliography

[1] CIGRE Technical Brochure No. 325. Review of on-line dynamic security assessment tools and techniques. e-cigre.org, 2007.

[2] Prabha Kundur, John Paserba, Venkat Ajjarapu, Göran Andersson, Anjan Bose, Claudio Canizares, Nikos Hatziargyriou, David Hill, Alex Stankovic, Carson Taylor, Thierry Van Cutsem, Vijay Vittal, and Fonds De La Recherche Scientifique. Definition and classification of power system stability IEEE/CIGRE joint task force on stability terms and definitions. *Power Systems, IEEE Transactions on*, 19(3):1387–1401, 2004.

[3] Knut Bjørsvik. *Monitoring the Stability of the Nordic Power Grid using On-line Dynamic Security Assessment Tools*. Unpublished pre-master's project, 2015.

[4] Jan Machowski, Janusz Bialek, and Jim Bumby. *Power System Dynamics : Stability and Control (2nd Edition)*. John Wiley & Sons, Chichester, GBR, 2009.

[5] P. Kundur. Power System Stability and Control. *Power System Stability and Control*, 1994. Cited By :8737 Export Date: 10 September 2015.

[6] Olle I. Elgerd. *Electric energy systems theory : an introduction*. McGraw-Hill series in electrical engineering. Power and energy. McGraw-Hill, New York, 2nd ed. edition, 1982.

[7] U. Kerin, T. N. Tuan, E. Lerch, and G. Bizjak. Small signal security index for contingency classification in dynamic security assessment. In *PowerTech, 2011 IEEE Trondheim*, pages 1–6, 2011.

[8] Espen Hagstrøm, Ian Norheim, and Kjetil Uhlen. Large-scale wind power integration in Norway and impact on damping in the Nordic grid. *Wind Energy*, 8(3):375–384, 2005.

[9] *The Princeton Companion to Mathematics*. Princeton University Press, Princeton, US, 2010.

[10] J. G. F. Francis. The QR Transformation A Unitary Analogue to the LR Transformation—Part 1. *The Computer Journal*, 4(3):265–271, 1961.

[11] N. Martins. The dominant pole spectrum eigensolver [for power system stability analysis]. *IEEE Transactions on Power Systems*, 12(1):245–254, 1997.

[12] J. M. Campagnolo and D. M. Falcao. Refactored bi-iteration: A high performance eigensolution method for large power system matrices. *IEEE Transactions on Power Systems*, 11(3):1228–1235, 1996.

[13] N. Martins and H. J. C. P. Pinto. Computing dominant poles of power system transfer functions. *IEEE Transactions on Power Systems*, 11(1):162–170, 1996.

[14] Thiago Jose Masseran Antunes Parreiras. *Metodologias para Avaliações de Segurança a Pequenos Sinais de Sistemas Elétricos de Potência,*. Unpublished phd thesis.

[15] S. Gomes, N. Martins, and C. Portela. Computing small-signal stability boundaries for large-scale power systems, 2003.

[16] I. Dobson and L. Lu. Computing an optimum direction in control space to avoid stable node bifurcation and voltage collapse in electric power systems. *Automatic Control, IEEE Transactions on*, 37(10):1616–1620, 1992.

[17] F. Alvarado, I. Dobson, and Y. Hu. Computation of closest bifurcations in power systems. *IEEE Transactions on Power Systems*, 9(2), 1994.

[18] D. J. Hill and D. J. Zhao-Yang Dong. Nonlinear computation and control for small disturbance stability, 2000.

[19] Siemens. Psse v.33 documentation, program application guide volume 1.

[20] CEPEL. Pacdyn v.9.8.0 documentation.

[21] Python version 2.7.0. https://www.python.org/downloads/. Accessed: 2016-06-09.

[22] PyQt4. https://www.riverbankcomputing.com/software/pyqt/download. Accessed: 2016-06-09.

[23] Enaml. https://github.com/nucleic/enaml. Accessed: 2016-06-09.

[24] Plot.ly. https://plot.ly/. Accessed: 2016-06-09.

[25] Silje Mork Hamre. *Inertia and FCR in the Present and Future Nordic Power System - Inertia Compensation*. Unpublished master's thesis, 2015.

[26] Market data | nord pool spot. ftp://ftp.nordpoolspot.com/. Accessed: 09 Jun, 2016.