TECHNISCHE UNIVERSITÄT CHEMNITZ

Faculty of Electrical Engineering and Information Technology

Master's Thesis

Extension of a Novel Set-Based Method for Transient Stability Analysis of Power Grids

Gyula Molnár

Supervisors: Dr. Willem Esterhuizen
M. Sc. Tim Aschenbruck

Examinor: Prof. Dr.-Ing. habil. Stefan Streif

Date: January 20th 2022

Automatic Control and System Dynamics
Prof. Dr.-Ing. habil. Stefan Streif

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Prüfungsausschuss des Studiengangs
Master Systems Engineering
Prüfungsausschussvorsitzender

Technische Universität Chemnitz · ZPA · D-09107 Chemnitz

Herrn
Gyula Molnar

Aktenzeichen:

Bearbeiter:
Raum:
Telefon:
Fax:
E-Mail: @verwaltung.tu-chemnitz.de
Internet: www.tu-chemnitz.de/studentenservice/
Ort, Datum: Chemnitz, 04.10.2021

Aufgabenstellung

Sehr geehrter Herr Molnar,

Ihr Antrag zur Abschlussarbeit wurde vom Prüfungsausschuss genehmigt.

Thema:
Extension of a novel set-based method for transient stability analysis of power grids

Ausgabedatum:	25.08.2021
Abgabedatum:	01.02.2022
Prüfer:	Prof. Dr. -Ing. habil. Stefan Streif
Prüfer:	Willem Esterhuizen

Die Abschlussarbeit ist in zwei Exemplaren in maschinenschriftlicher und gebundener Ausfertigung sowie zusätzlich als elektronische Datei im Zentralen Prüfungsamt abzugeben. Eine Selbstständigkeitserklärung ist einzubinden.

Mit freundlichen Grüßen
Im Auftrag

Zentrales Prüfungsamt

Dienst- u. Paketanschrift: Technische Universität Chemnitz ·ZPA
Reichenhainer Str. 70 · 09126 Chemnitz
Postanschrift: Technische Universität Chemnitz · 09107 Chemnitz · GERMANY

Kurzfassung

Der Letztere Paradigmenwechsel zu erneuerbaren Energieerzeugung stellt eine neue Situation dar, in der die Überwachung von Stromnetzstabilität immer mehr zu einem Hauptanliegen wird. Vorige Forschung ließ uns folgern, dass der Aufbau eines mengenbasierten Ansatzes zur automatisierten Optimierung der transienten Stromnetzstabilität möglich sein soll.

Die vorliegende Masterarbeit befasst sich zunächst mit theoretischen Vorkenntnissen zur mengenbasierten transienten Stabilitätsanalyse von Stromnetzen: oszillatorische Modelle, Viability Theory, Theorie der Barrieren, Netzentkopplung und Validität von MRPI-Mengen. Anschließend befassen wir uns mit der Erstellung eines Optimierungsproblems für Bestimmung von MRPI-Mengen einschließlich mathematischer Formulierung und Definition von Nutzenfunktionen. Übliche Optimierungsalgorithmen und entsprechende Löser werden vorgestellt, und auf das zuvor definierte Problem zur Stabilitätsoptimierung des IEEE-9-Bus-Testsystems angewendet. Ergebnisse werden ausgewertet und eine Schätzung der Steigerung von Rechenkomplexität bei komplexeren Systemen wird abgeleitet. Folgend wird die Kategorisierung der Netzstabilität gegeben und Ansätze zur Stabilitätsbewertung werden zusammengefasst. Abschließend wird die mengenbasierte Bestimmung der Critical Clearing Time erklärt, und eine Heuristik für mengenbasierte CCT-Optimierung vorgestellt.

Abstract

The recent paradigm shift towards renewable means of energy generation presents a new situation in which power grid stability governance is more and more of a principal concern. Preceding research let us conclude that building a set-based approach for automated power grid transient stability optimization should be feasible.

The thesis at hand initially overviews theoretical preliminaries of set-based transient stability analysis of power grids: oscillatory models, viability theory, the theory of barriers, grid decoupling, and MRPI set validity. The work then engages in outlining an optimization problem for MRPI set determination including mathematical formulation and definition of objection functions. Common optimization algorithms and corresponding solvers are surveyed, and applied to the problem defined beforehand for stability-optimizing the IEEE nine-bus test system. Results are evaluated, and an estimation for increase in computational complexity for more complex systems is deducted. Subsequently, a categorization of power system stability is given, and stability assessment approaches are reviewed. In conclusion set-based critical clearing time determination is presented, and a heuristic for set-based CCT optimization is showcased.

Contents

Notation

A	Node MRPI area
A_i	Power output / power demand of the ith node
$\mathcal{A}$	Admissible set
$[\partial\mathcal{A}]_-$	Admissible set barrier
$\boldsymbol{c}$	Combined node dynamics vector
D_i	Damping constant of the ith node
$\mathcal{D}$	Disturbance space
$\boldsymbol{d}$	Disturbance input
$\bar{\boldsymbol{d}}$	Input corresponding to the barrier trajectory
δ	Torque angle or angular deviation
$\delta_{\min}, \delta_{\max}$	Angular constraints boundaries
δ_{sat}	Saturated torque angle
E_i^*	Steady-state internal voltage of the ith node
$\boldsymbol{f}(\cdot)$	System dynamics
$f_{G,i,j}$	The ith component of the dynamics of the jth generator node
$f_{L,i,j}$	The ith component of the dynamics of the jth load node
$\boldsymbol{\Phi}(\cdot)$	Continuous feedback control law
Γ	Set of all generator nodes
$\mathcal{G}$	Constraint set
$\mathcal{G}_0$	Constraint boundary set
$\mathcal{G}_-$	Constraint internal set
γ_{ij}	Phase shift involved in the coupling between nodes i and j
$g_i(\cdot)$	Constraint function of the ith constraint
H_i	Inertia constant of the ith node
$\boldsymbol{h}(\cdot)$	System output function
$\boldsymbol{I}$	Current injection vector
i_{ref}	Index of the reference node
K_{ij}	Strength of dynamical coupling between nodes i and j
Λ	Set of all load nodes
$\bar{\boldsymbol{\lambda}}$	Adjoint vector
$\boldsymbol{\lambda}_{\smile,i}$	Lower adjoint evolution

$\boldsymbol{\lambda}_{\frown,i}$	Upper adjoint evolution
M	Angular momentum
$\mathcal{M}$	Maximal robust positively invariant (MRPI) set
$[\partial\mathcal{M}]_{\_}$	MRPI set barrier
N	Number of machines/nodes/oscillators in the system
$\mathcal{N}_i$	Set of the ith node's neighboring nodes
ω, ω_R	Frequency or rotation speed, reference frequency
P_i^*	Steady-state active power injection of the ith node
P_s, P_e, P_a	Shaft-, electrical-, and accelerating power
p	Number of state-space constraints
Θ_i	Oscillator phase of the ith oscillator
$\bar{t}$	Time of angular constraint boundary intersection
$\bar{t}_{\smile,i}$	Time of the lower barrier candidate's tangential intersection of the higher constraint boundary.
$\bar{t}_{\frown,i}$	Time of the upper barrier candidate's tangential intersection of the lower constraint boundary.
$\tilde{t}$	Time of the first $\omega = 0$ axis intersection
$\mathcal{U}$	State-dependent control space
$\boldsymbol{u}$	Control input
$\mathcal{X}$	State space
$X(\cdot)$	Phase-dependent influence
$\boldsymbol{x}$	State vector
$\bar{\boldsymbol{x}}$	Initial state on the admissible-, or MRPI barrier
$\boldsymbol{x}_{\text{eq}}$	State vector at equilibrium
$\boldsymbol{x}_{\bar{\boldsymbol{d}},\bar{\boldsymbol{x}}}, \boldsymbol{x}_{\bar{\boldsymbol{d}}}$	Barrier trajectory with initial conditions $\bar{\boldsymbol{x}}$ and $bar\boldsymbol{d}$
$\boldsymbol{x}_{\smile,i}$	Lower candidate trajectory
$\boldsymbol{x}_{\frown,i}$	Upper candidate trajectory
$\mathcal{Y}$	Output space
$\boldsymbol{Y}$	Nodal admittance matrix
$\boldsymbol{y}$	Output vector
y_{ij}	Admittance between nodes i and j
y_{ij}	Admittance between nodes i and j
$\boldsymbol{V}$	Voltage injection vector
V_i^*	Steady-state terminal voltage of the ith node
$\mathcal{Z}$	Control space
$Z(\cdot)$	Phase-dependent response
$\bar{\boldsymbol{z}}$	State of angular constraint boundary intersection

List of Abbreviations

CCA	Critical Clearing Angle
CCT	Critical Clearing Time
EN	Effective Network (oscillatory model)
MRPI	Maximal Robust Positively Invariant
RPI	Robustly Positively Invariant
SM	Synchronous Motor (oscillatory model)
SP	Structure Preserving (oscillatory model)

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Problem Setup

During the last century, electricity has certainly become an essential necessity of our lives. With that, the power grid came to be one of the most extensive, most complex, and most critical systems humankind relies on. Nevertheless, the very fact that these systems interconnect an extensive amount of power devices constitutes an inherent vulnerability against contingencies.

All the more as we observe an ever-growing urge for a shift towards renewable sources of energy: new forms of power generation not only are prone to environmental uncertainties, but recent technological developments counteract power system stability in general. This is because regenerative energy production means are usually controlled by power electronics in place of mechanical variables such as moment of inertia or friction. These are associated with less inertia and therefore less damping, hence falling behind in stability. Further research on transient power system stability analysis is thus much desirable in order to have a grasp on aforementioned challenges.

In the preceding project work [46] we have already examined set based transient stability analysis and the validity of safety sets, with a focus on finding such sets through a systematic approach. Building on this, the problem setup that we consider is an oscillatory model of a power grid, decomposed into individual power elements, and a research framework in which conventional and set-based power analysis tools have been implemented and are readily available.

1.2 Task Description

Power grid's stability is classified into voltage stability, frequency stability, and rotor angle stability, with further specifications related to short and long term effects, as well as the consideration of small and big disturbances which are affecting the grid. The focus of this work is on the short term rotor angle stability where the power grid is subjected to a large disturbance. This form of stability is also known as *transient stability*. There exist several approaches for the transient stability analysis of power grids. Well known methods are based on time simulations, energy functions, and set-based methods. A novel set-based approach for the transient stability problem via the theory of barriers is presented in [4]. The goal of this thesis is to consider the transient stability problem via invariant sets constructed by the theory of barriers and investigate the problem of invalid sets. The specific tasks for this thesis are:

1. Literature review and understanding of the basic principles of the considered topics.
   a) Reading basic literature about power grid stability and understanding the stability problems as well as approaches considering the rotor angle stability and particularly the transient stability problem.
   b) Reading and understanding the principle of modeling a power grid as a network of coupled oscillators.
   c) Summarizing the principles of common oscillatory models (EN/SM/SP), and the construction thereof.
   d) Overview the process of decoupling a power grid for the purpose of set-based analysis.
   e) Studying and summarizing the theory of barriers [14] [18] and particularly the work related to the transient stability analysis of power grids via theory of barriers including the idea of considering a decoupled power grid [4].
2. Summarizing of related theories and preliminary research.
   a) Overviewing how the research framework is set up, including theoretical and technical aspects thereof.
   b) Describing MRPI set validity, and possible reasons for invalidities.
   c) Introducing the conjecture on barrier phases and the connection between phases and set validity. Show how such a connection might serve as a basis for an iterative algorithm.
   d) Formulate a theoretical optimization problem making use of the conjecture.
3. Implementation an optimization problem for the sake of obtaining and optimizing valid MRPI sets in an automated manner.
   a) Implementation the formulated optimization problem in a parametrizable way.
   b) Defining and implementing suitable objective functions.
   c) Overviewing and comparing common optimization algorithms.
   d) Comparing different combinations of optimization algorithms, goal functions, and oscillatory models as applied to the nine-bus test system.
   e) Analyzing the applicability of the most promising approaches to more complex power grids.
   f) Illustrating a practical use-case of set-based analysis on the example of the critical clearing time problem, including a comparison with other approaches.
4. Finalizing the thesis and summarizing the results as well as discussing further research directions.

1.3 Overview

The principal objective of this thesis is to further unfold the findings and questions that arose in [46]. In particular, we set out to investigate the automatability of selecting the generators' angular deviation constraints, such that these result in MRPI sets of optimal size.

Chapter 2 serves as an overview of preliminaries necessary for subsequent exploratory efforts. In it, we first recapitulate the coupled oscillator representation of power grids, and deduce the system dynamics from Kuramoto's formulation of the swing equation. Then, the Kron reduction method —which provides for the size reduction of the nodal admittance matrix, and eventually a complete oscillatory graph— is briefly characterized, followed by the electric circuit representation of transmission lines and the equivalent circuit of synchronous generators. Next, three oscillatory models (Effective Network, Structure Preserving, and Synchronous Motor) and their construction methods are surveyed, leading up to a tabular comparison of these approaches. What follows, is again a reiteration of viability theory fundamentals, including the formulation of a general constrained, uncertain dynamical system, and of the constraint-, admissible-, and MRPI set definitions. We define set barriers for aforementioned sets, and show that these can be obtained through numerical backwards integration. Finally, the two topologically distinct types of valid MRPI sets are presented, as well as the two sorts of invalidities that an MRPI barrier may be subject to.

Chapter 3 starts off by giving the mathematical formulation of an optimization problem and a goal function structure in general. We formulate inequality constraints —relative to each machines' equilibrium point— for our special case of power elements. Next, we summarize desirable properties of an ideal objective function to be minimized, and contemplate over the computational advantage the constraint violation might enable, as well as possible discontinuity of the objective function due to the jumping phenomenon. Having considered the above, we propose two optimization methods: the MRPI set area method, and the span method. As hereby the equilibrium point is much relied on (remaining static throughout optimization it serves as an origin), we continue by showing how it can be obtained. At last, an overview of six common optimization algorithms follows.

With Chapter 4 theory gets put into practice. We show the software landscape on which we rely in our examinations: in what steps an optimization task is generally carried out in MATLAB, what solvers does one have access to through what toolboxes, and how we utilize Nishikawa's `pg_sync_models` library for MATPOWER cases and oscillatory model generation. We present the IEEE nine-bus three-generator test system along with two initialization vectors on which we examine optimization performance. We compare six MATLAB-provided solvers' performance with both the area method based-, as well as the span method based objective function by evaluating their speed and —where present— end results. To conclude this chapter, we explore the applicability of MRPI set area optimization to power grids more complicated than the nine-bus test system. Hereby, the EN and SP oscillatory models —each with the span method

based goal function— are observed, especially how the CPU time required for objective function evaluation grows as the complexity of the inspected power grid increases.

Having elaborated on the feasibility of automated angular constraint adjustment and MRPI set area optimization, in Chapter 5 we wish to exhibit a practical engineering use-case. For this, we aim at drawing a connection between MRPI set area size, and critical clearing time. Hereby, a classification of power system stability is provided, showing the whereabouts of rotor angle transient stability in this hierarchy. We further elaborate on transient stability, outlining Dy-Liacco's diagram of power grid operating states, and show what operating states the pre-fault, fault-on, and post-fault scenarios might correspond to. We proceed to describe the classical model of synchronous electrical machines, and the power-angle relationship. Building on this, we deduct the equal areas criterion, and review further approaches for stability analysis, including time domain simulation, direct methods, model-free approaches, and set-based methods. Subsequently, considerations behind set-based CCT determination are outlined, and a test framework —based on the nine-bus power grid's EN oscillatory model, and two distinct fault-on scenarios— is set up. At the very end we compare MRPI set areas and fault-on trajectory lengths, as well as MRPI set areas and CCT estimates in search of correlations.

Theoretical background

2

The generation of electrical energy and the delivery thereof is an area of electrical engineering that brings along problems that excite many experts, as in our current times, most households and businesses depend on a safe and reliable power source to operate.

However, such a power source presupposes a reliable transmission-, and distribution grid of interoperating electrical equipment. When it comes to reliability, it is inevitable to examine such an electrical network as a whole. Thus, power grid models for simulation and analysis were developed to enable and support power systems management.

2.1 Coupled Oscillator Representation of Power Systems

Let us consider a grid of interconnected electrical components. As our work is mainly concerned with transient rotor angle stability, a convenient enough model for our investigations is one that considers the rotor angle deviation, as well as its rate of change as state variables. Said model designates the node with index i_{ref} as a reference bus with a constant reference angle taken as 0, while all the other nodes' state vectors are described by

$$\boldsymbol{x}_i = \begin{bmatrix} x_{1,i} \\ x_{2,i} \end{bmatrix} = \begin{bmatrix} \delta_i \\ \omega_i \end{bmatrix} \tag{2.1}$$

Research around synchronization phenomena is all but new. As we have already provided in [46, sec. 2.2.1] a brief overview of the breakthroughs of Winfree [59], Huygens [27], Kuramoto [38], and Filatrella [19], we omit any further historical survey. Instead we introduce Nishikawa's formulation of a power grid coupling model that *"can be regarded as a second-order analog of the Kuramoto model with arbitrary coupling structure"* [49]. In said model, a power grid is modeled as a web of interconnected oscillators described through a set of swing equations of form:

$$\left(\frac{2H_i}{\omega_R} \ddot{\delta}_i + \frac{D_i}{\omega_R} \dot{\delta}_i = A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_i - \delta_j - \gamma_{ij}\right) \right) \forall\, i \in \Gamma \cup \Lambda \tag{2.2}$$

where

- δ_i is the ith node's angle deviation as compared to a reference node common to the whole system.
- H_i is the ith node inertia constant: $H_i \neq 0 \,\forall\, i \in \Gamma,\ H_i = 0 \,\forall\, i \in \Lambda$

- ω_R is a reference angular frequency for the system
- D_i is the ith node damping constant
- A_i is the ith node power output / power demand.
- K_{ij} is a constant that represents the strength of the dynamical coupling between the ith and jth node
- γ_{ij} represents a phase shift involved in the coupling between the ith and jth node
- $\mathcal{N}_i$ is the set of all neighboring nodes of the ith node
- Γ is the set of all generator nodes: nodes modeled as second order oscillators, e.g. nodes with $H_i \neq 0$
- Λ is the set of all load nodes: nodes modeled as first order oscillators, e.g. nodes with $H_i = 0$

Using (2.2), the system dynamics can be described by the state's time derivatiave.:

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} \dot{\delta}_i \\ \dot{\omega}_i \end{bmatrix} = \begin{bmatrix} \omega_i \\ \frac{\omega_R}{2H_i}\left(A_i - \frac{D_i}{\omega_R}\omega_i - \sum_{j\in\mathcal{N}_i} K_{ij}\cdot\sin\left(\delta_i - \delta_j - \gamma_{ij}\right)\right) \end{bmatrix} \forall\, i \in \Gamma \tag{2.3}$$

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} \dot{\delta}_i \\ \dot{\omega}_i \end{bmatrix} = \begin{bmatrix} \frac{\omega_R}{D_i}\left(A_i - \sum_{j\in\mathcal{N}_i} K_{ij}\cdot\sin\left(\delta_i - \delta_j - \gamma_{ij}\right)\right) \\ 0 \end{bmatrix} \forall\, i \in \Lambda \tag{2.4}$$

With that, the groundwork for an oscillatory model is concluded. However, [49] further on conducted a comparative analysis of three leading power network structure models on the basis of (2.2). Given that this thesis relies heavily on said models, a brief introduction will be provided in coming subsections.

2.1.1 Kron Reduction

Two of the three aforementioned models in [49] utilize a method called the Kron-reduction, which is a method for node elimination in power systems through a process most similar to Gauss-elimination.

In simplified terms, the method first considers the nodal admittance equation of the unreduced, N-node network:

$$\boldsymbol{YV} = \boldsymbol{I} \tag{2.5}$$

where $\boldsymbol{Y} \in \mathbb{C}^{N\times N}$ is the nodal admittance matrix with elements

$$Y_{ij} = \begin{cases} y_i + \sum y_{ik} \quad \forall\, k \in \{1\ldots N\}\setminus\{i\} & \text{if} \quad i = j \\ -y_{ij} & \text{if} \quad i \neq j \end{cases} \tag{2.6}$$

where y_{ij} is the complex valued admittance between nodes i and j, and y_i is the ith node admittance to ground. Furthermore, $\boldsymbol{V} \in \mathbb{C}^N$ and $\boldsymbol{I} \in \mathbb{C}^N$ are the nodes' voltage and current injection vectors respectively.

Assuming that the current injection at the pth node is zero, (2.5) becomes

$$\boldsymbol{Y}\boldsymbol{V} = \begin{bmatrix} \boldsymbol{I}_{1..(p-1)} \\ 0 \\ \boldsymbol{I}_{(p+1)..N} \end{bmatrix} \tag{2.7}$$

Then, the pth bus can be removed if voltage at that node is not of particular interest. This is done by constructing an effective admittance matrix $\mathbf{Y}^{\text{red}} \in \mathbb{C}^{(N-1)\times(N-1)}$ of the reduced network, so that voltages and currents at remaining nodes remain as before the reduction:

$$\boldsymbol{Y}^{\text{red}} \begin{bmatrix} \boldsymbol{V}_{1..(p-1)} \\ \boldsymbol{V}_{(p+1)..N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{1..(p-1)} \\ \boldsymbol{I}_{(p+1)..N} \end{bmatrix} \tag{2.8}$$

According to [24, Eq. 7.67], elements of the reduced admittance matrix can be calculated as

$$Y_{jk}^{\text{red}} = Y_{jk} - \frac{Y_{jp}Y_{pk}}{Y_{pp}} \quad \forall\, j,k \in \{1\ldots N\} \setminus \{p\} \tag{2.9}$$

Then, if necessary, the Kron reduction process (2.5)-(2.9) may be carried out recursively until the desired number of zero-injection nodes are removed.

An important property of recursive Kron reduction is that it leads to a fully connected reduced network (a compete graph). This is due to [15, Theorem 3.4/1)], stating that two nodes are connected in the reduced network graph if and only if the two nodes were connected in the original graph by an edge or a path in which all nodes have been eliminated. That is, $Y_{ij}^{\text{red}} \neq 0\ \forall\ i,j$.

This is in contrast to the original, unreduced graph that often represents the physical network structure of a power grid, and so usually contains numerous pairs of nodes (generators or buses) i and j that are not connected, and thus $Y_{ij} = 0$. An Illustration of this general change in topology is illustrated in Figure 2.1.

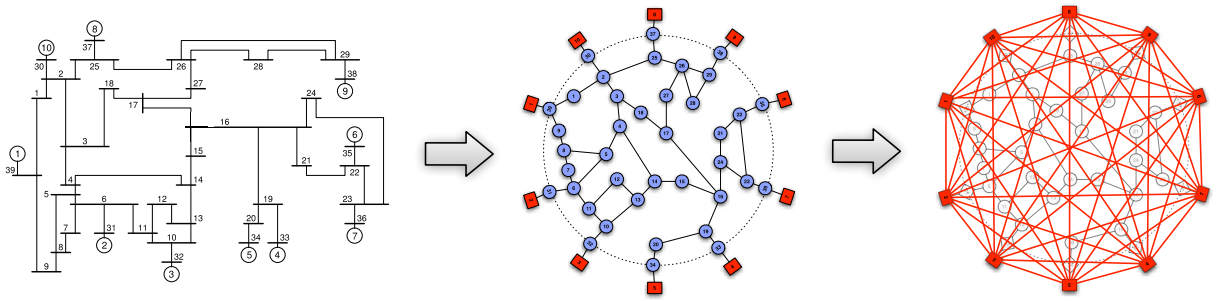


Figure 2.1: Illustration of the Kron-reduction. Left: the single line diagram of a power grid. Center: equivalent representation with generators (denoted by red rectangles), and buses (denoted by blue dots). Right: the corresponding Kron-reduced network. From [15, Fig. 2.4.].

2.1.2 Electric Circuit Representation

Simple network representation of power grids are most concerned with representing the topological structure of a power grid. Such representations consist of elements such as generators, consumers, and transformers, with links between them representing transmission lines. They often reflect the topology of a real-world power network, and are essential for analyzing power flow. The leftmost graph of Figure 2.1 illustrates such a representation (notice how generated/consumed power units at each node are shown), albeit fictitious ones.

However, simple network representation is incapable of describing dynamical behavior of the interplay of power system components. For that, the most versatile electrical circuit representation must be derived from the simple representation by the means of substituting equivalent electrical circuits for components as well as transmission lines.

For example, transmission lines are traditionally represented in the so-called π-model as shown in Figure 2.2.

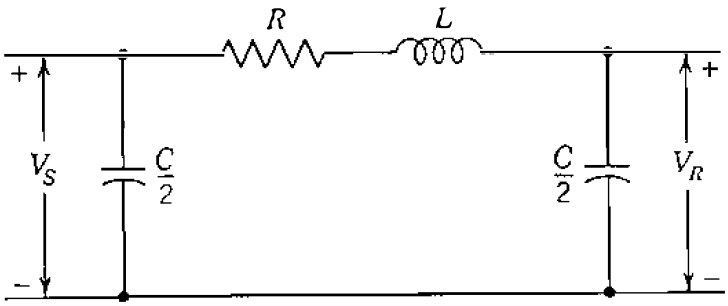


Figure 2.2: Equivalent electrical circuit (π-model) of a medium-length transmission line. From [24, Fig. 6.2].

while the synchronous generator is modeled by a constant voltage source behind a transient reactance as in Figure 2.3.

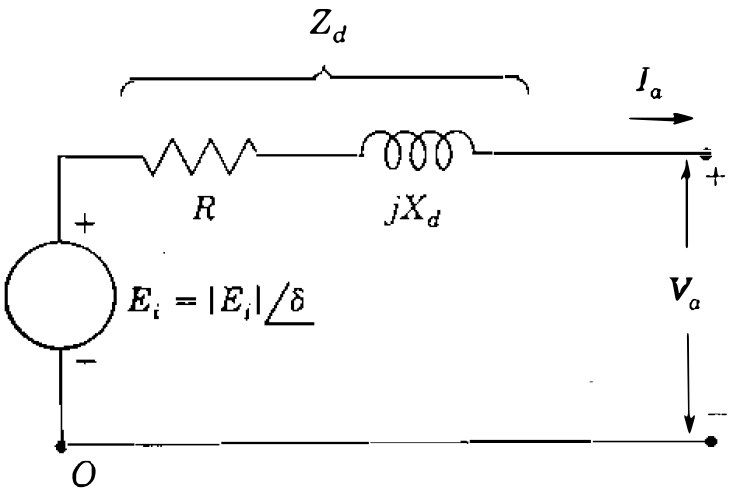


Figure 2.3: Equivalent electrical circuit (voltage-behind-reactance, VBR-model) for a synchronous generator. From [24, Fig. 3.8].

For the representation of loads in the circuit diagram multiple approaches exist. In fact, the difference between these load representations is a major difference between the three network structure models whose introductions briefly follow.

To show how the simple network representation is translated into the electric circuit representation, [49] called for a small illustratory network consisting of two generators and a load in between as shown in Figure 2.4.

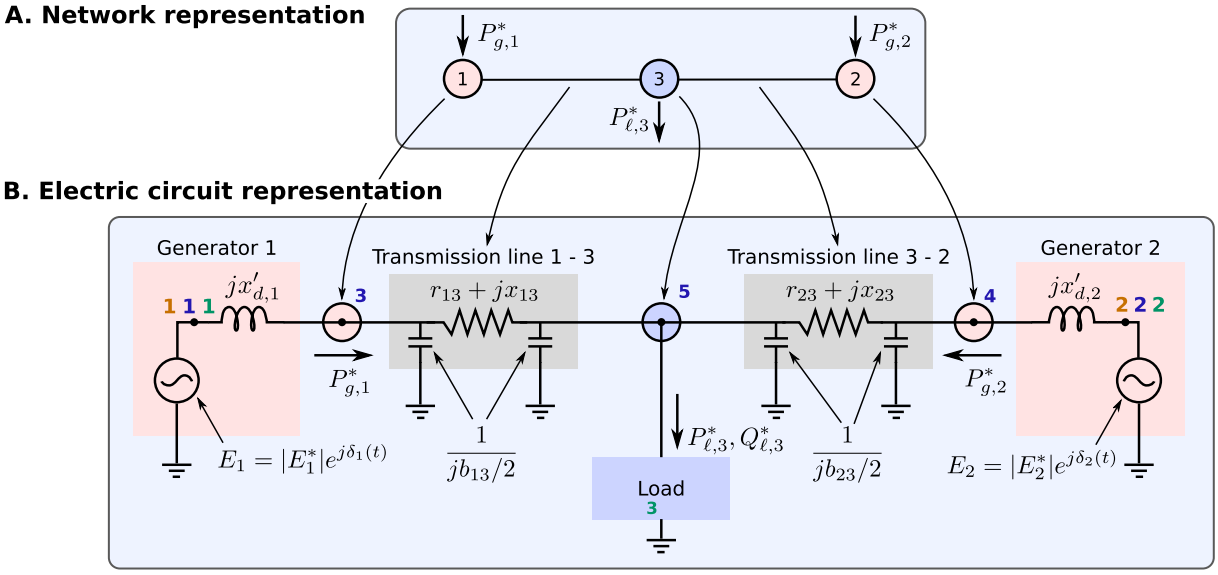


Figure 2.4: Modeling of power-grid network dynamics. Cropped from [49, Fig. 1].

Indeed, equivalent circuits from Figure 2.2 and Figure 2.3 were substituted for generators and transmission line, even though in the circuit equivalent for transmission line, a transient impedance is shown instead of a resistance and a reactance. In the circuit for synchronous generators, the resistance is omitted altogether.

2.1.3 The Effective Network (EN) Model

The effective network model (EN model for short) was initially proposed by [47], and is also referred to as the network-reduction model or network-reduced model.

The essential consideration of the model is that the nature of coupling between a pair of generators connected to the same power grid is primarily determined by the structure of the transmission lines and loads in between the generator nodes [49, S. 4.1], while the dynamical interaction between said generators is described by applying the classical model's power-angle equation (5.3) for both machines.

Effective network representation is concerned with grasping the effect of aforementioned intermediate network structure on the dynamic interaction of each pair of *generators* through a single term that depends only on the generators' state variables. This can indeed be done through applying *network reduction* to a network model that considers loads as *constant impedances* tied to the ground potential. Building such a model involves the following steps:

First, the assumption is made, that network structure is fixed, and that power demand is constant. This is deemed viable for the purposes of transient stability analysis, as only very short time scales are considered. With these assumptions, the solution to the optimal power flow problem is calculated.

Secondly, the physical network structure described by (2.5) is extended with the transient reactances connecting the generators' internal and terminal nodes, and equivalent impedances for the loads. Let us denote the admittance matrix of this extended structure Y'.

Third, this extended circuitry undergoes Kron-reduction as discussed in Section 2.1.1: assuming that the nodal admittance equation is in the form:

$$\boldsymbol{Y}' \begin{bmatrix} \boldsymbol{V}^g \\ \boldsymbol{V}^t \\ \boldsymbol{V}^\ell \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I}^t \\ \boldsymbol{I}^\ell \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}^g \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{2.10}$$

where $\boldsymbol{V}^g$, $\boldsymbol{V}^t$ and $\boldsymbol{V}^\ell$ are voltages, and $\boldsymbol{I}^g$, $\boldsymbol{I}^t$ and $\boldsymbol{I}^\ell$ are current injections at -in order- generator internal nodes (between the reactance and the voltage source in the generator model), generator terminal nodes (on the other side of the generator reactance), and load nodes. However, since there is no current injection at transmission lines (thus neither at terminal nodes), nor at load nodes (since they are modeled as constant impedance to the ground) $\boldsymbol{I}^t$ and $\boldsymbol{I}^\ell$ are always zero, and so (2.10) can be Kron-reduced to

$$\boldsymbol{Y}^{\mathrm{EN}} \boldsymbol{V}^g = \boldsymbol{I}^g \tag{2.11}$$

where $\boldsymbol{Y}^{\mathrm{EN}} \in \mathbb{C}^{N_g \times N_g}$ is the effective admittance matrix where N_g is the number of generators in the power grid. In it, every pair of generator is connected by an effective admittance Y_{ij}^{EN} [49, S. 4.1].

Finally, this reduced admittance matrix is used for defining parameters for (2.2) as follows:

$$\begin{aligned} A_i^{\mathrm{EN}} &= P_{g,i}^* - |E_i^*|^2 \,\mathrm{Re}(Y_{ii}^{\mathrm{EN}}) \\ K_{ij}^{\mathrm{EN}} &= |E_i^* E_j^* Y_{ij}^{\mathrm{EN}}| \\ \gamma_{ij}^{\mathrm{EN}} &= \mathrm{Arg}(Y_{ij}^{\mathrm{EN}}) - \frac{\pi}{2} \end{aligned} \tag{2.12}$$

The star symbol in (2.12) denotes values of the steady-state operation: $P_{g,i}^*$ is the ith generator's steady-state active power injection, and E_i^* is the ith node steady-state internal voltage. Indeed, these are the values static to the network's operation in it's equilibrium point, the ones that are calculated in the first step of building the EN model by solving the power flow problem.

In practice, the open-source power system simulation software toolkit MATPOWER's power flow solver `runpf` [61, S. 2.4.2, S. 4.4] was generally applied to power system models from [50] to obtain aforementioned steady-state data in all cases.

In the EN model, every node of the reduced network (that is, every generator) is modeled as a second order oscillator, and so inertia constants H_i and damping constants D_i in (2.2) are both positive. These mechanical parameters most depend on the physical design of the generators in question, and how they are estimated [49, S. 5] once again lies outside the scope of this paper. In practice, the pre-defined values in [50] have been used for our purposes.

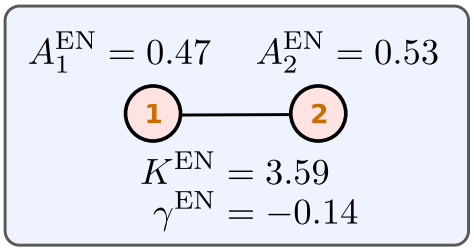


Figure 2.5: EN model oscillator representation of the illustratory network in Figure 2.4. Second order oscillator nodes 1 and 2 correspond to points marked with a brownish 1 and 2 in the electric circuit representation in Figure 2.4. Cropped from [49, Fig. 1].

2.1.4 The Structure Preserving (SP) Model

The structure preserving model (SP model for short) was originally proposed by [6]. It seeks to describe the dynamic behavior of loads by considering them as *first-order oscillators*, by the means of which building an oscillator model that preserves physical network structure is possible. Building said oscillator model involves the following steps. First, similarly to the EN model, assumptions about the fixedness of network structure and constant power demand are made, and the solution to the optimal power flow problem is calculated. Secondly, much like with the EN model, an extended circuitry is built with generators' terminal reactances-, and load nodes added. Then, dynamic models for each load's power consumption is derived in the form:

$$P_{\ell,i}(t) = P^*_{\ell,i} + \frac{D_i}{\omega_R}\dot{\delta}_i(t) \tag{2.13}$$

where $P_{\ell,i}(t)$ is the ith load active power demand, $P^*_{\ell,i}$ is the ith load steady-state active power demand, D_i is a linear constant, ω_R is the reference frequency, and $\dot{\delta}_i(t)$ is the ith node voltage phase shift as compared to a common reference node. Notice that (2.13) is but a linearization of (5.3) around the steady state power demand $P^*_{\ell,i}$, which is acceptable for the purposes of transient stability analysis, since only small deviations from the steady-state frequency are considered [49, S. 4.2].

Substituting $H_i = 0$ into (2.2), and reinterpreting D_i as a linear constant of a load's active power demand's dependence on the load's voltage frequency leads to an equation much like (2.13): a *first order oscillator*. Indeed, (2.2) will be used for building the dynamics for the SP model's loads (modeled as first order oscillators with $H_i = 0$) and generators (modeled as second order oscillators with $H_i \neq 0$) as well, even though the constants from (2.12) need to be redefined.

Unlike in (2.10), no assumptions about zero power injections at load- and generator terminal nodes (again, both modeled as first order oscillators) can be made so the nodal admittance equation goes

$$\boldsymbol{Y'}\begin{bmatrix}\boldsymbol{V}^{\mathcal{g}}\\ \boldsymbol{V}^{t}\\ \boldsymbol{V}^{\ell}\end{bmatrix}=\begin{bmatrix}\boldsymbol{I}^{\mathcal{g}}\\ \boldsymbol{I}^{t}\\ \boldsymbol{I}^{\ell}\end{bmatrix} \tag{2.14}$$

Also, since the SP model foresees no network reduction (preserving the physical network structure in the oscillator model is a major advantage with this model after all), above admittance matrix is indeed the one that is used for the redefinition of the parameters of (2.2), thus $\boldsymbol{Y'}=\boldsymbol{Y}^{\mathrm{SP}}$

Then, aforementioned parameters are defined as follows:

$$\begin{aligned}A_i^{\mathrm{SP}}&=\begin{cases}P^*_{\mathcal{g},i} & \text{for generator internal nodes}\\ -P^*_{\ell,i}-|V^*_{i-N_{\mathcal{g}}}|^2\,\mathrm{Re}(Y_{ii}^{\mathrm{SP}}) & \text{for load nodes}\end{cases}\\ K_{ij}^{\mathrm{SP}}&=\begin{cases}|E_i^*V_i^*\,\mathrm{Im}(Y_{ij}^{\mathrm{SP}})| & \text{for generator internal nodes}\\ |V^*_{i-N_{\mathcal{g}}}V^*_{j-N_{\mathcal{g}}}\,\mathrm{Im}(Y_{ij}^{\mathrm{SP}})| & \text{for load nodes}\end{cases}\\ \gamma_{ij}^{\mathrm{SP}}&=\begin{cases}0 & \text{for generator internal nodes}\\ \mathrm{Arg}(Y_{ij}^{\mathrm{SP}})-\dfrac{\pi}{2} & \text{for load nodes}\end{cases}\end{aligned} \tag{2.15}$$

where $P^*_{\{\mathcal{g},\ell\},i}$ is the ith node steady-state active power injection/consumption ($\mathcal{g}$ or ℓ in the subscript denote that the node in question is modeled as a generator (second order oscillator) or a load (first order oscillator) in the oscillator model respectively, and only serve clarity), E_i^* is the ith node (given that it is a generator) steady-state internal voltage, V_i^* denotes the ith node's steady-state (terminal) voltage, and $N_{\mathcal{g}}$ the number of generators. For the indices i and j it is assumed that that nodes are numbered as shown in Figure 2.6, starting with all generator internal nodes, followed by all terminal nodes in the same order, so that the generator internal node with index i is only connected to the generator terminal node $i+N_{\mathcal{g}}$.

As with the EN model, values of (2.15) denoted by a star symbol are values generated by solving the optimal power flow problem, in our case by utilizing [61] and [50]. Likewise, mechanical parameters of (2.2) $H_i>0$ and $D_i>0$ have to be estimated based on the machine's design parameters in case of generators. Since loads are modeled as first order oscillators in the SP model, $H_i=0$ is true for them, whereas D_i has to be chosen so that (2.13) holds well. Again, we have been relying on [50] for these values.

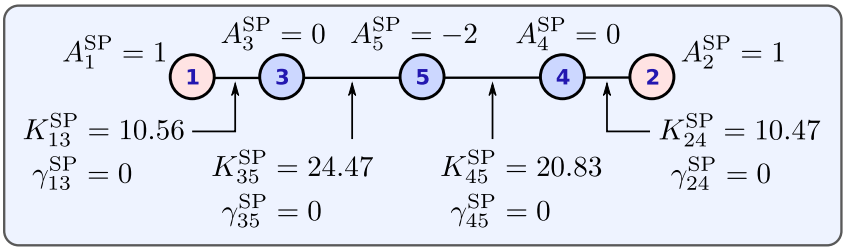


Figure 2.6: SP model oscillator representation of the illustratory network in Figure 2.4. Second order oscillator nodes (generators marked by a light red filling) 1 and 2 correspond to points marked with the same bluish numbers in the electric circuit representation in Figure 2.4. Nodes 3, 4, and 5 are first order oscillator nodes (loads marked by a light blue filling) corresponding to the two generator terminal nodes, and the actual load node in the electrical circuit representation marked by the same blue numerals in Figure 2.4. Cropped from [49, Fig. 1].

2.1.5 The Synchronous Motor (SM) Model

Applying the Kuramoto model to an utility power grid can be attributed to [19]. In it, both generators and loads were modeled as synchronous machines; indeed [49, S. 4.3] refers to an oscillator model of such topology as the synchronous motor model, or SM model for short.

Just like in the EN model, all nodes are modeled as second order oscillators, and thus building the SM model is analogous to how the EN model is built. However, because in this case loads are also considered as synchronous generators (of negative power generation: motors, that is), the extended admittance matrix Y' has to be further extended to Y'' in that transient reactances for the generators modeling loads are considered as well. Then, much like in (2.10):

$$\boldsymbol{Y''}\begin{bmatrix}\boldsymbol{V}^{g}\\ \boldsymbol{V}^{t}\\ \boldsymbol{V}^{\ell}\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}^{g}\\ \boldsymbol{I}^{t}\\ \boldsymbol{I}^{\ell}\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}^{g}\\ \boldsymbol{0}\\ \boldsymbol{0}\end{bmatrix} \tag{2.16}$$

which gets Kron-reduced similar to as in (2.11):

$$\boldsymbol{Y}^{\mathrm{SM}}\boldsymbol{V}^{g} = \boldsymbol{I}^{g} \tag{2.17}$$

Then, this reduced admittance matrix is used for defining parameters for (2.2) analogously to (2.12)

$$\begin{aligned} A_i^{\mathrm{SM}} &= \begin{cases} P_{g,i}^{*} - |E_i^{*}|^2\,\mathrm{Re}(Y_{ii}^{\mathrm{SM}}) & \text{for nodes modeling generators} \\ -P_{\ell,i}^{*} - |E_i^{*}|^2\,\mathrm{Re}(Y_{ii}^{\mathrm{SM}}) & \text{for nodes modeling loads} \end{cases} \\ K_{ij}^{\mathrm{SM}} &= |E_i^{*}E_j^{*}Y_{ij}^{\mathrm{SM}}| \\ \gamma_{ij}^{\mathrm{SM}} &= \mathrm{Arg}(Y_{ij}^{\mathrm{SM}}) - \frac{\pi}{2} \end{aligned} \tag{2.18}$$

where $P_{\{g,\ell\},i}^{*}$ is the ith node steady-state active power injection/consumption (g or ℓ in the subscript only denote whether the node in question is modeling a generator or

a load of the original physical network), and E_i^* is the ith node steady-state internal voltage.

Steady state values and mechanical constants are obtained by means of [61] and [50] just as in Section 2.1.3.

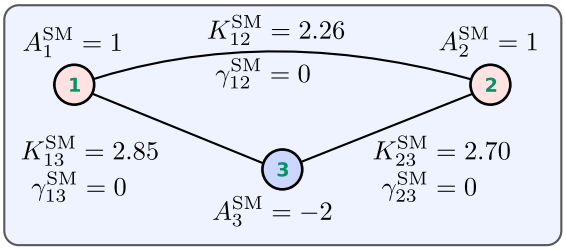


Figure 2.7: SM model oscillator representation of the illustratory network in Figure 2.4. Second order oscillator nodes 1, 2 (representing generators) and 3 (representing a load) correspond to the same numbers written in a greenish color in the electric circuit representation in Figure 2.4. Cropped from [49, Fig. 1].

2.1.6 Comparison of Oscillatory Models

Features of the oscillator models overviewed in Sections 2.1.3, 2.1.4, and 2.1.5 are collected into Table 2.1.

Features	Oscillator models		
	EN	**SP**	**SM**
Oscillator model structure	complete graph	preserved	complete graph
Generators modeled as (electrical model)	constant voltage behind purely transient reactance		
Loads modeled as (electrical model)	const. volt. & trans. react.	(2.13)	const. volt. & trans. react.
Generators modeled as (oscillator model)	second order oscillator		
Loads modeled as (oscillator model)	set of effective admittances	first order oscillator	second order oscillator
Number of first order oscillators	0	$N_g + N_\ell$	0
Number of second order oscillators	N_g	N_g	$N_g + N_\ell$
Total number of oscillator nodes	N_g	$2N_g + N_\ell$	$N_g + N_\ell$
Number of edges (oscillator model)	$\frac{N_g(N_g-1)}{2}$	structure dependent	$\frac{(N_g+N_\ell)(N_g+N_\ell-1)}{2}$
Steps involved in model building	OPF, Y', Kron, A_i^{EN}, K_{ij}^{EN}, $\gamma_{ij}^{\mathrm{EN}}$	OPF, Y', A_i^{SP}, K_{ij}^{SP}, $\gamma_{ij}^{\mathrm{SP}}$	OPF, Y'', Kron A_i^{SM}, K_{ij}^{SM}, $\gamma_{ij}^{\mathrm{SM}}$

Table 2.1: Comparison of the EN, SM, and SP models. "OPF" refers to having to solve the optimal power flow problem.

2.2 Viability Theory and Theory of Barriers

In [46, Sec. 2.3] we have given a brief overview of the origins and underlying considerations of viability theory. In it, we have demonstrated that an uncertain, dynamical system subjected to engineering constraints —think of equipment tolerances— can be formulated as

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{d}) \\ \boldsymbol{x}_{t=0} = \boldsymbol{x}_0 \\ \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \\ g_i(\boldsymbol{x}(t)) \leq 0 \ \forall\, t \in [0, T],\ \forall\, i \in \{1, \ldots, p\} \end{cases} \tag{2.19}$$

where

- $\boldsymbol{x} = \boldsymbol{x}(t)$ denotes a state vector over state space $\mathcal{X}$

- $\boldsymbol{y} = \boldsymbol{y}(t)$ denotes an output vector over the output space $\mathcal{Y}$.
- $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{d})$ is a single valued map from some $(\Omega \subset \mathcal{X}) \rightarrow \mathcal{X}$
- $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}(t))$ denotes the output function of the system.
- $\boldsymbol{d} = \boldsymbol{d}(t)$ is an (external) disturbance input vector over the disturbance space $\mathcal{D}$
- $g_i = g_i(\boldsymbol{x}) = g_i(\boldsymbol{x}(t))$ is the ith real-valued constraint function.
- p is the number of space-state constraints on the examined system
- $T > 0 \in \mathbb{R}$ is some arbitrarily chosen point in time

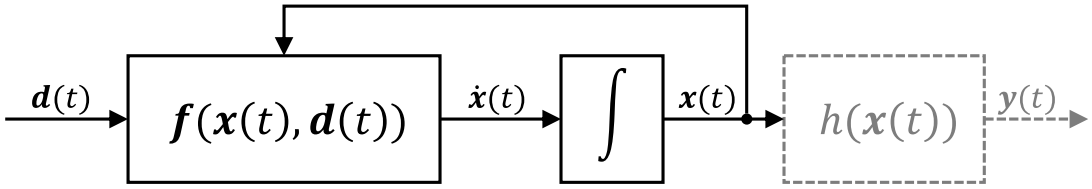


Figure 2.8: Block diagram of a system described in (2.19)

.

Definition 1 (Constraint set)**.** *The constraint set is the set of all points in $\mathcal{X}$ so that all constraints are met [14, Eq. 3.5]:*

$$\mathcal{G} = \{\boldsymbol{x} \in \mathcal{X} : g_i(\boldsymbol{x}) \leq 0 \; \forall \; i \in \{1, \ldots, p\}\}$$

Definition 2 (Admissible set)**.** *The set of admissible states (or the admissible set) is the set of all initial states in $\mathcal{X}$ for which a disturbance $\boldsymbol{d}$ exists so that none of the system constraints are violated [14, Eq. 4.1] [4, Def. 1] for all future evolution:*

$$\mathcal{A} = \{\boldsymbol{x}_0 \in \mathcal{G} : \exists \; \boldsymbol{d} \in \mathcal{D} \; s.t. \; \boldsymbol{x} \in \mathcal{G} \; \forall \; t \in [0, \infty)\}$$

The admissible set is also referred to as the viability kernel in some sources.

Definition 3 (Maximal robust positively invariant set)**.** *The maximal robust positively invariant set (MRPI) is the set of all initial states in $\mathcal{X}$ for which no disturbance evolutions violate any system constraints:*

$$\mathcal{M} = \{\boldsymbol{x}_0 \in \mathcal{G} : \boldsymbol{x} \in \mathcal{G} \; \forall \; \boldsymbol{d} \in \mathcal{D} \, , \forall \; t \in [0, \infty)\}$$

The main motive of what follows revolves around applying the considerations outlined in this subsection to the dynamical system modeling a power grid, specifically finding admissible and invariant sets as a means of making a statement about power grid stability.

2.2.1 Theory of Barriers

Having defined the Admissible set in Def. 2 and the MRPI set in Def. 3, the question remains of how one determines what exact points of the state-space fall inside or outside the sets defined in Def. 2 and Def. 3, given specific system dynamics. In answering this question , we will be relying on the theory of barriers, and the works of [4, S. 3.1], [18, S. 4], and [14].

Let us consider an uncertain dynamical system $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{d}(t))$ from (2.19). We supplement the constraint set $\mathcal{G}$ from Def. 1 with the following sets:

Definition 4 (Constraint boundary set)**.**

$$\mathcal{G}_0 = \{\boldsymbol{x} \in \mathcal{X} : g_i(\boldsymbol{x}) = 0 \; \forall \; i \in \{1, 2, \ldots, p\}\}$$

Definition 5 (Constraint internal set)**.**

$$\mathcal{G}_{\_} = \{\boldsymbol{x} \in \mathcal{X} : g_i(\boldsymbol{x}) < 0 \; \forall \; i \in \{1, 2, \ldots, p\}\}$$

with $\mathcal{G} = \mathcal{G}_0 \cup \mathcal{G}_{\_}$.

In [46, S. 2.3.1] we have imposed a set of assumptions from [4, S. 3.1], [18, S. 4], and [14], stating that if they hold, the admissible set $\mathcal{A}$ and the MRPI set $\mathcal{M}$ are coherent. Given this case, we might go on denoting their boundaries as $\partial\mathcal{A}$ and $\partial\mathcal{M}$, based on which we further define the following barriers:

Definition 6 (Admissible set barrier)**.**

$$[\partial\mathcal{A}]_{\_} = \partial\mathcal{A} \cap \mathcal{G}_{\_}$$

Definition 7 (MRPI set barrier)**.**

$$[\partial\mathcal{M}]_{\_} = \partial\mathcal{M} \cap \mathcal{G}_{\_}$$

Under the referenced set of assumptions, for every initial condition $\bar{\boldsymbol{x}} \in [\partial\mathcal{M}]_{\_}$ (or $\bar{\boldsymbol{x}} \in [\partial\mathcal{A}]_{\_}$), there exists an input $\bar{\boldsymbol{d}} \in \mathcal{D}$ such that the resulting trajectory $\boldsymbol{x}_{\bar{\boldsymbol{d}},\bar{\boldsymbol{x}}}(t) \in [\partial\mathcal{M}]_{\_}$ (or $\boldsymbol{x}_{\bar{\boldsymbol{d}},\bar{\boldsymbol{x}}}(t) \in [\partial\mathcal{A}]_{\_}$) remains on the MRPI barrier (or admissible set barrier) until the integral curve intersects $\mathcal{G}_0$.

2.3 Power Grid Decoupling for Set-Based Analysis

Although several models of power grids exist [47][2][6][19], physical arrangement dictates that a common trait of these is that they generally assign system components (modeled as oscillators) to nodes-, and connections to edges of a mathematical graph.

This usually results in a large scale system where each node's state is governed by the swing equation (2.2). However, because of reasons already discussed in [46, S. 2.4.], relying directly on a power grid model for simulation involves solving the swing equations for all the state vectors for every time step. In other words, if all n nodes' state vectors

are of dimension $\mathbb{R}^m$, then the state vector of the complete grid model —examined as a whole— would be of dimension $\mathbb{R}^{(n \cdot m)}$.

Other than the computationally challenging task of analyzing nonlinear systems of high-dimensional state spaces, simulation-based approaches also prove to be insufficient due to the possible environmental disturbances characterizing power systems. Furthermore, preliminary safety analysis of a power system requires that all possible states of the system be accounted for [11]. Latter consideration brings us to the sets introduced in Section 2.2. Admissible set $\mathcal{A}$ of Def. 1 and the MRPI set $\mathcal{M}$ of Def. 2 together provide information about a given point $\boldsymbol{x}$ in state-space as follows [4, S. 3.2]:

$$\boldsymbol{x} : \begin{cases} \textit{safe} \text{ if } \boldsymbol{x} \in \mathcal{M} \\ \textit{potentially safe} \text{ if } \boldsymbol{x} \in \mathcal{A} \setminus \mathcal{M} \\ \textit{unsafe} \text{ if } \boldsymbol{x} \in \overline{\mathcal{A}} \end{cases} \tag{2.20}$$

However, obtaining $\mathcal{A}$ and $\mathcal{M}$ for high-dimensional nonlinear systems is often difficult, or possible only with certain trade-offs [11].

Thus, decomposing high-dimensional systems into smaller subsystems is desirable, like how [4] applies the decomposition principles introduced in [12], [40] to a power system. Decomposition in aforementioned works follows the principle of considering subsets of the complete system's state vector as a state vector of a smaller system. However, some state variables in these subsystems' state vectors might depend on other state variables of the *big* composed system that might have been decomposed into another subsystem. Such interdependencies between subsystems are accounted for by assigning each state variable to exactly one subsystem as as state variable, and considering the variable in question as a disturbance input in other, interdependent subsystems. Said disturbance inputs are also called *decoupling variables* [4] in this context.

Although in [46, S. 2.4.], we have shown the decoupling process on a small exemplary grid, here we confine ourselves to outlining the generalized case. Hereby, the undecoupled system dynamic can be formulated as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dots & \dot{\boldsymbol{x}}_i & \dots \end{bmatrix}^T =$$

$$= \begin{bmatrix} \vdots \\ \begin{cases} (2.3) \text{ if } i \in \Gamma \\ (2.4) \text{ if } i \in \Lambda \end{cases} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \begin{cases} \begin{bmatrix} f_{G,1,i}(\omega_i) \\ f_{G,2,i}(\delta_i, \omega_i, \{\delta_j \, \forall j \in \mathcal{N}_i \setminus \{i_{\text{ref}}\}\}) \end{bmatrix} & \text{if } i \in \Gamma \\ \begin{bmatrix} f_{L,1,i}(\delta_i, \{\delta_j \, \forall j \in \mathcal{N}_i \setminus \{i_{\text{ref}}\}) \\ 0 \end{bmatrix} & \text{if } i \in \Lambda \end{cases} \\ \vdots \end{bmatrix} \tag{2.21}$$

$$\forall \, i \in (\Gamma \cup \Lambda) \setminus \{i_{\text{ref}}\}$$

where

- Γ is the set of all generator nodes in the system
- Λ is the set of all load nodes in the system
- $\boldsymbol{f}_{G,i} = (2.3) = [f_{G,1,i} \quad f_{G,2,i}]^T$ is the generator dynamics for nodes $i \in \Gamma$
- $\boldsymbol{f}_{L,i} = (2.4) = [f_{L,1,i} \quad f_{L,2,i}]^T$ is the load dynamics for nodes $i \in \Lambda$
- $\mathcal{N}_i$ is the set of all neighboring nodes of the ith node
- i_{ref} is the index of the reference node.

The aforementioned state-interdependence can now be represented in a State Dependency Graph, as it was introduced in [40] as in Figure 2.9(a).

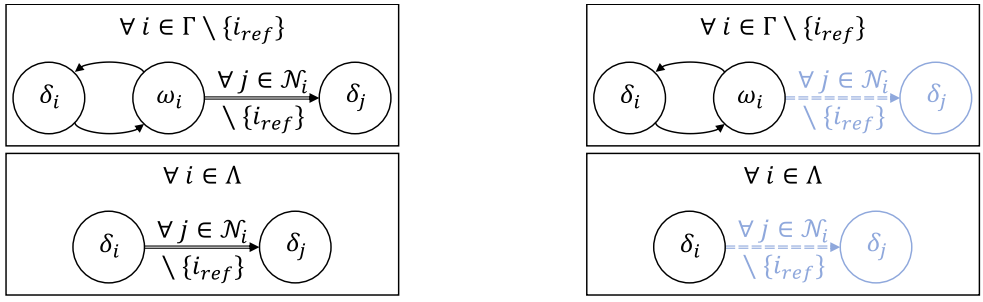


Figure 2.9: Dependency graph for an arbitrary un-decomposed power grid (a), and for the decoupled case. (b). Decomposition variables taken as disturbance inputs are denoted by light color.

A convenient decoupling might be treating each node as an individual subsystem, considering neighboring nodes' angle deviations as decoupling variables. The dependency graph of the decomposition can be seen in Figure 2.9(b).

Although through such a decomposition the analyzed systems' state space dimension could technically be reduced to 2 and 1 for generator-, and load nodes respectively, the reduced subsystems are still interdependent on each other through the decomposition variables. Indeed, the advantages of decomposition is less obvious until it comes to set-based analysis introduced in Section 2.2. Instead of finding admissible set of Def. 2, and MRPI set of Def. 3 for the original, high-dimensional system (that would not be computationally plausible for large systems), lower dimensional $\mathcal{A}$ and $\mathcal{M}$ can be found for each decoupled subsystem.

Still, admissible-, and MRPI sets depend on the decoupling variables (a node's neighbors' angular deviations), since they —even though treated as disturbance inputs— do affect the subsystems' dynamic behavior. Fortunately, we are considering an engineering application after all, and can rely on inequality constraints introduced in Section 5.1.1 that are constraints on loads' and generators' angular deviations (and possibly on generators' frequencies, although this thesis only considers the former). In other words, it is

possible to define angular constraints $\delta_{\min,i}$ and $\delta_{\max,i}$ for each node, thus limiting the interval of values each decoupling variable can take, based on which the admissible-, and MRPI sets can be determined as described in Section 2.4.

One advantage of set-based decomposition is that admissible-, and MRPI sets for the subsystems only depend on nodes directly connected to the node in question (the number of which in a real-life power grid is typically much smaller than in a full graph), easing computational complexity. Furthermore, applying the outlined approach to determine safe-, potentially safe-, and unsafe operating state areas (2.20) for each node, power grid operators need not analyze nodes whose post-fault state lies in the corresponding MRPI set, as those -per definition- are guaranteed to ceteris paribus stay inside the MRPI set, thus between angular inequality constraints as well.

As a consequence of aforementioned benefits, a single node's or connection's fault is less likely to trigger a constraint violation in a node further away, and so power system supervision can concentrate their resources on topologically (and possibly also geographically) well-bounded areas following power system malfunction.

2.4 Determining Set Barriers for Power Grid Analysis

For a system to fulfill rotor angle stability, it must be able to return to its equilibrium point even if a single machine's state is changed between some finite constraints. The barrier trajectory candidates of the admissible, and MRPI set candidates ($\partial\mathcal{A}$ and $\partial\mathcal{M}$ of Section 2.2.1) are obtained by solving the initial value problem of [14, S. 7] and [18, S. 6] that we have also formulated in [46, S. 2.5]. There, we defined a function $\boldsymbol{c}$ by combining the system dynamic $\boldsymbol{x}$ and the adjoint equation $\boldsymbol{\lambda}$:

$$\boldsymbol{c}(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \cdots & \boldsymbol{x}_i & \cdots \end{bmatrix}^T \\ \begin{bmatrix} \cdots & \boldsymbol{\lambda}_i & \cdots \end{bmatrix}^T \end{bmatrix} \quad \forall i \in \{1 \dots n\} \setminus \{i_{\text{ref}}\} \tag{2.22}$$

where n is the number of machines in the oscillatory model, $\boldsymbol{x}_i$ is the ith machine's barrier candidate trajectory, consisting of the trajectories lower barrier candidate $\boldsymbol{x}_{\smile,i}$ and/or the upper barrier candidate $\boldsymbol{x}_{\frown,i}$ so that $\boldsymbol{x}_i = \boldsymbol{x}_{\smile,i} \cup \boldsymbol{x}_{\frown,i}$, whereas $\boldsymbol{\lambda}_i \neq 0$ is the nonzero adjoint evolution of the ith machine (a vector that at all times stays perpendicular to $\dot{\boldsymbol{x}}_i$), for which $\boldsymbol{\delta}_i = \boldsymbol{\delta}_{\smile,i} \cup \boldsymbol{\delta}_{\frown,i}$ similarly holds. The differential equation describing the barrier trajectories is a function of each constraint in the whole system:

$$\dot{\boldsymbol{c}} = \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{\lambda}} \end{bmatrix} = \dot{\boldsymbol{c}}(\boldsymbol{c}, \boldsymbol{\delta}_{\min}, \boldsymbol{\delta}_{\max}, \boldsymbol{\delta}_{\text{sat}}) \tag{2.23}$$

$$\dot{\boldsymbol{c}}_i\Big|_{t=\bar{t}_{\smile,i}} = [0 \quad 1 \quad -1 \quad 0]^T \quad \forall i \in \{1 \dots n\} \setminus \{i_{\text{ref}}\} \tag{2.24}$$

$$\dot{\boldsymbol{c}}_i\Big|_{t=\bar{t}_{\frown,i}} = [0 \quad -1 \quad 1 \quad 0]^T \quad \forall i \in \{1 \dots n\} \setminus \{i_{\text{ref}}\} \tag{2.25}$$

where $t_{\smile,i}$ and $t_{\frown,i}$ are the points in time when the lower and upper barrier candidates tangentially intersect the angular constraints $\delta_{\min}$ and $\delta_{\max}$ respectively, and $\boldsymbol{\delta}_{\text{sat}}$ denotes a saturation function as described in [4, Eq. 10]:

$$\begin{aligned} &\delta_{\text{sat},ij}(\delta_i, \lambda_{2,i}, \delta_{\min,j}, \delta_{\max,j}) = \\ &= \begin{cases} \min(\delta_{\max,j}, \max(\delta_{\min,j}, \delta_i - \frac{\pi}{2}\text{sign}(\lambda_{2,i}))) \text{ for set } \mathcal{A} \\ \min(\delta_{\max,j}, \max(\delta_{\min,j}, \delta_i + \frac{\pi}{2}\text{sign}(\lambda_{2,i}))) \text{ for set } \mathcal{M} \end{cases} \end{aligned} \tag{2.26}$$

Accordingly, (2.24) and (2.25) —or in a narrower sense $\boldsymbol{x}_i\big|_{t=\bar{t}_{\smile,i}} = [\delta_{\min,i} \quad 0]$ and $\boldsymbol{x}_i\big|_{t=\bar{t}_{\frown,i}} = [\delta_{\max,i} \quad 0]$— are referred to as the points of ultimate tangentiality. Given this initial value problem, the admissible-, and MRPI set barrier candidate trajectories are obtained by first fixing endpoints thereof in accordance with the criterion in (2.24) and (2.25), then performing numeric backwards integration of (2.23).

2.5 Validity of MRPI Sets

A valid MRPI set is one that is coherent [18, S. 3.1], and whose envelope is defined by the barrier trajectory $\boldsymbol{x}_i$ defined in (2.23), as well as the angular constraints boundaries $\delta_{\min,i}$ and $\delta_{\max,i}$. In case of our interconnected second order oscillators (i.e. generators, as discussed in Section 2.1), two topologically distinct variants of a valid MRPI set may exist, as shown in Figure 2.10 (please note that from this point on, the machine index will generally be omitted for the sake of simplicity. Thus, when the reader encounters notations such as δ, $\delta_{\min}$, $\delta_{\max}$, ω, $\boldsymbol{x}$, $\boldsymbol{\lambda}$, D, H, . . ., they should consider these as variables related to the ith machine: δ_i, $\delta_{\min,i}$, $\delta_{\max,i}$, ω_i, $\boldsymbol{x}_i$, $\boldsymbol{\lambda}_i$, D_i, H_i, etc.)

- On one hand, a valid MRPI set may look similar to that in Figure 2.10(a), both the lower-, and upper MRPI barriers intersecting the opposite angular constraint. In this case, the MRPI set's envelope is defined by both barriers and both angular constraint boundaries. This will be referenced to as a **type A** MRPI set later in this work.
- On the other hand, a valid MRPI set may look like that in Figure 2.10(b), whose envelope is defined by one angular constraint boundary, as well as by one MRPI barrier intersecting said angular constraint boundary twice. This will be referenced to as a **type B** MRPI set later in this work.

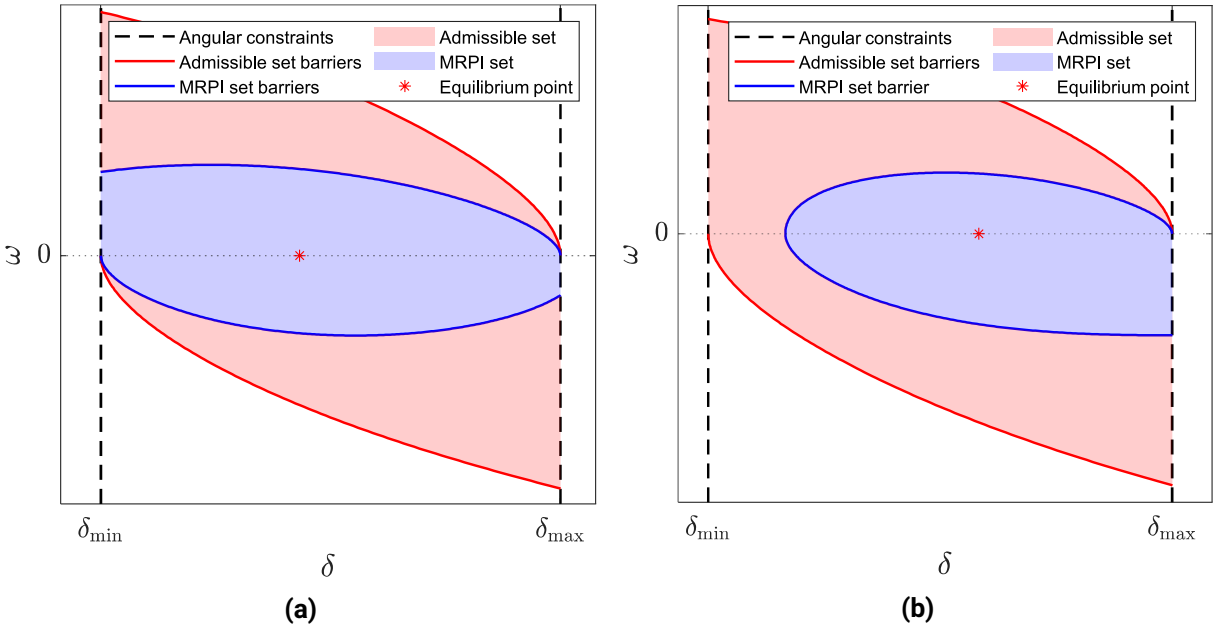


Figure 2.10: Examples of valid MRPI sets. (a): Type A. (b): Type B.

However, as we have summarized in [46, S. 3.1], not all barrier candidates fulfill the criterion of coherence, and may resemble those in Figure 2.11.

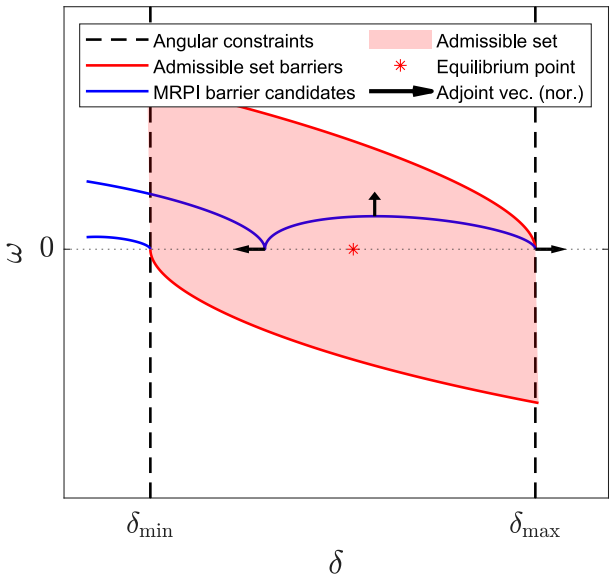


Figure 2.11: Phase diagram of a generator, displaying a valid admissible set with MRPI barrier candidates that do not build a valid MRPI set. The upper barrier candidate represents the jumping phenomenon, while the lower barrier candidate represents constraint violation.

These invalidities can be attributed to two phenomena: the constraint violation, and the jumping phenomenon. The former happens if and only if [4, Proposition 1]:

$$A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_{\min,i} - \delta_j - \gamma_{ij}\right) > 0 \tag{2.27}$$

$$A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_{\max,i} - \delta_j - \gamma_{ij}\right) < 0 \tag{2.28}$$

is unfulfilled, whereas the jumping phenomenon is due to $\dot{\omega}$ undergoing a sign change as the barrier candidate trajectory intersects the $\omega = 0$ axis. Unlike the constraint violation, the jumping phenomenon cannot be foreseen by simple algebraic inequalities. This is because their incidence can be attributed to that $\boldsymbol{x}$ and $\boldsymbol{\delta}$ of (2.23) utilize the signum function in their definitions [46, Eq. 2.31], allowing for an abrupt change. The ultimate consequence is that finding out whether a set of angular constraints lead to an MRPI barrier candidate undergoing the jumping phenomenon is actually only possible by the means of differential equation (2.23). In other words, we cannot make a certain statement about neither the existence nor the whereabouts of a jumping point (the point on the $\omega = 0$ axis in the phase diagram where the phenomenon takes place) unless we conduct the numerical integration of (2.23) as far backwards as either a jump or an angular constraint intersection. A more detailed description of the circumstances under which the jumping phenomenon takes place can be found in [46, S. 3.1.2].

Optimization Problem to Determine MPRI and Admissible Sets

3

3.1 Formulation of the Optimization Problem

As outlined in [39, S. 5.1] and [3, S. 1.2], an optimization problem can generally be formulated as

$$\begin{aligned} \boldsymbol{z}^* = \underset{\boldsymbol{z}\in\mathbb{R}^n}{\operatorname{argmin}}\, & f(\boldsymbol{z}) \\ \text{s.t. } & g_i(\boldsymbol{z}) \leq 0, \quad i \in \{1, 2, \ldots, p\} \\ & h_j(\boldsymbol{z}) = 0, \quad j \in \{1, 2, \ldots, q\} \end{aligned} \tag{3.1}$$

where

- $\boldsymbol{z} \in \mathbb{R}^n$ is a vector of n design variables.
- $\boldsymbol{z}^*$ is the optimal design vector.
- $f(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}$ is the goal function (also objective function, or cost function) to be minimized.
- $g_i(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R},\ i \in \{1, 2, \ldots, p\}$ is a set of $p \geq 0$ inequality constraints.
- $h_j(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R},\ j \in \{1, 2, \ldots, q\}$ is a set of $q \geq 0$ equality constraints.

In our case, we will choose $\boldsymbol{z} := [\boldsymbol{\delta}_{\min}\, \boldsymbol{\delta}_{\max}]^T$ as the design variable vector, and define the goal function $f(\boldsymbol{z})$ so that it returns

$$f(\boldsymbol{z}) \in \begin{cases} (-\infty, -1] & \text{if } \boldsymbol{z} \text{ is feasible} \\ (-1, 0] & \text{otherwise} \end{cases} \tag{3.2}$$

where $\boldsymbol{z}$ is considered feasible if z_i constitutes valid MRPI barriers $\forall\, i \in \Gamma \cup \Lambda \setminus \{i_{ref}\}$.

Because δ_i only ever appears composed with a periodic (sine or cosine) function in [46, Eq. 2.29-30], there is no point looking for solutions outside of the $[\delta_{eq,i} - \frac{\pi}{2}, \delta_{eq,i} + \frac{\pi}{2}]$ interval:

$$\delta_{eq,i} - \frac{\pi}{2} \leq \{\delta_{\min,i}, \delta_{\max,i}\} \leq \delta_{eq,i} + \frac{\pi}{2} \quad \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.3}$$

When a valid MRPI set exists for a machine, it necessarily encloses the equilibrium point:

$$\delta_{\min,i} \leq \delta_{eq,i} \leq \delta_{\max,i} \quad \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.4}$$

By reformulating (3.3) and (3.4), we arrive at inequality constraints

$$\left.\begin{aligned} g_1(\boldsymbol{z}) &:= \delta_{eq,i} - \frac{\pi}{2} - \delta_{\min,i} \leq 0 \\ g_2(\boldsymbol{z}) &:= \delta_{eq,i} - \frac{\pi}{2} - \delta_{\max,i} \leq 0 \\ g_3(\boldsymbol{z}) &:= -\delta_{eq,i} - \frac{\pi}{2} + \delta_{\min,i} \leq 0 \\ g_4(\boldsymbol{z}) &:= -\delta_{eq,i} - \frac{\pi}{2} + \delta_{\max,i} \leq 0 \\ g_5(\boldsymbol{z}) &:= \delta_{\min,i} - \delta_{eq,i} \leq 0 \\ g_6(\boldsymbol{z}) &:= \delta_{eq,i} - \delta_{\max,i} \leq 0 \end{aligned}\right\} \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.5}$$

Since changing the constraints doesn't affect the whereabouts of the equilibrium point, we might as well take the distances between the angular error constraints, and the equilibrium angular error as the optimization parameters:

$$\begin{aligned} \boldsymbol{s}_{\min} &= \boldsymbol{\delta}_{eq} - \boldsymbol{\delta}_{\min} \\ \boldsymbol{s}_{\max} &= \boldsymbol{\delta}_{\max} - \boldsymbol{\delta}_{eq} \\ \boldsymbol{s} &= [\boldsymbol{s}_{\min}\ \boldsymbol{s}_{\max}]^T \end{aligned} \tag{3.6}$$

and redefine the goal function, as well as the validity intervals from (3.3) and (3.4):

$$f_s(\boldsymbol{s}) = f(\boldsymbol{z}) \tag{3.7}$$

$$-\frac{\pi}{2} \leq \{s_{\min,i}, s_{\max,i}\} \leq \frac{\pi}{2} \quad \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.8}$$

$$\{s_{\min,i}, s_{\max,i}\} \geq 0 \quad \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.9}$$

Reformulating these we arrive at the inequality constraints:

$$\left.\begin{aligned} g_{s,1}(\boldsymbol{s}) &:= -s_{\min,i} - \frac{\pi}{2} \leq 0 \\ g_{s,2}(\boldsymbol{s}) &:= -s_{\max,i} - \frac{\pi}{2} \leq 0 \\ g_{s,3}(\boldsymbol{s}) &:= s_{\min,i} - \frac{\pi}{2} \leq 0 \\ g_{s,3}(\boldsymbol{s}) &:= s_{\max,i} - \frac{\pi}{2} \leq 0 \\ g_{s,5}(\boldsymbol{s}) &:= -s_{\min,i} \leq 0 \\ g_{s,6}(\boldsymbol{s}) &:= -s_{\max,i} \leq 0 \end{aligned}\right\} \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{\text{ref}}\} \tag{3.10}$$

3.2 Defining the Objective Functions

The initial step towards setting up an optimization problem is finding a suitable objective function (also referred to as a goal function or cost function). What constitutes a

good objective function partly depends on the sort of optimization problem at hand. As for us, the following properties of the goal function are desired: $f_s(\boldsymbol{s})$ must

1. be $\mathbb{R}^{2(N-1)} \rightarrow \mathbb{R}$, with N being the number of nodes in the oscillatory model (nodes and generators). The goal function's domain is of dimension $2(N-1)$ because for each machine, the angular constraints of both sides $\boldsymbol{s}_i = [s_{min,i}, s_{max,i}]^T \in \mathbb{R}^2$ (see (3.6)) are considered as optimization variables, and because one generator node serving as the reference node is ignored $i \in \Gamma \cup \Lambda \setminus \{i_{ref}\}$.

2. prefer maximal MRPI sets, that contain the maximal number of points of the state space.

3. privilege global validity to local optimality. Global validity hereby denotes that for all nodes of the oscillatory model, optimization parameters —and thereby angular constraints as in (3.6)— are so, that valid MRPI sets —as in Section 2.5— exist for all nodes but the reference node. Local optimality means that aforementioned optimization parameters are instead so, that a subset of nodes possess MRPI sets that are maximal as in Point 2.

4. preferably be able to indicate when global validity —as in Point 3— is reached. This is important so that it can be determined whether a set of valid MRPI sets has been found, in which case optimization parameters might be readjusted for optimization for MRPI set size. In fact, $f(\boldsymbol{s})$ has been defined in (3.2) so that this point is fulfilled.

5. preferably is applicable to MRPI sets of both type A and B as in Figure 2.10.

6. preferably be computationally inexpensive and easily parallelizable.

7. preferably be continuous, so that continuous optimization methods can be applied.

3.2.1 Further Considerations

The property of continuity in Section 3.2 Point 7, however, can not be generally fulfilled with our problem setup.

This is due to Point 4 and especially (3.2), because of which no matter what metric we choose as a measurement of barrier candidates' goodness, a change in the validity of a single MRPI barrier candidate might result in a sudden step of the goal function's value.

Let us consider the following short example for better comprehension. Given that (3.7) currently evaluates to some value $f_s(\boldsymbol{s}) < -1$, we can assume —by the definition of the goal function (3.2)— that the current set of optimization variables (angular constraint deviances as in (3.6)) build valid MRPI sets for all oscillator nodes, with the possible exception of the reference node. Then, let us consider that the vector of optimization variables $\boldsymbol{s}$ is being continuously modified. If during this modification one

of the MRPI barrier candidates suddenly turns invalid, then —again by (3.2)— the goal function must now suddenly evaluate to a value $f_s(\boldsymbol{s}) > -1$.

In above example it was merely considered that continuous change of the optimization variables $\boldsymbol{s}$ might result in the sudden (in)validity of an MRPI barrier candidate. In the following subsections it will be shown in what circumstances is this indeed a possibility.

Constraint violation

Valid MRPI barriers should tangentially intersect with one of the angular constraints and cross either that same constraint, or the opposite one. However, for some $\boldsymbol{s}$ values, backwards integration of (2.23) results in an MRPI barrier candidate that violates the angular constraint by spreading outwards of the $[\delta_{\min,i}, \delta_{\max,i}]$ interval as discussed in [46, Sec. 3.1.1].

The described phenomenon only ever occurs if inequalities (2.27) and (2.28) are not fulfilled [4, Proposition 1]. However, since in practice numerical backwards integration of (2.23) is to be carried out anyway once constraint violation is excluded, it does not constitute great computational effort to -instead of relying on conditions (2.27) and (2.28)- actually perform said integration, while checking whether the resulting δ component in $\boldsymbol{c}$ is outside of the constraint bounds after a single time step.

That being said, latter method does require more computations, as all four components in $\dot{\boldsymbol{c}}$ in (2.23) have to be evaluated, so it might worth checking whether the above necessary conditions are met for a given $\boldsymbol{s}$ vector first after all.

Jumping phenomenon

In [46, Sec. 3.1.2] we have described how the jumping phenomenon can solely be attributed to a signum function undergoing sudden value change due to the sign change of λ_2. It is to be noted that discontinuity was understood in time dimension, whereas here we are discussing how the objective function might undergo a sudden or abrupt change as it's parameters are changed continuously. In other terms, it is the objective function's continuity that is assessed.

That being said, the statement can be made that continuous adjustment of the optimization vector $\boldsymbol{s}$ can lead to a sudden change in the validity of an MRPI barrier candidate, and with that, an abrupt change in the objective function's value as well. The argumentation is as follows:

Each barrier candidates' evolution is ultimately described by the angular constraints, and by the differential equation (2.24). Would δ_{sat} not be present in these equations, would state variables $\boldsymbol{x}(t-\bar{t})$ and $\boldsymbol{\lambda}(t-\bar{t})$ change continuously (for some fixed t) as $\boldsymbol{s}$ (and through (3.6), the angular constraints) undergoes continuous change.

However, we described in [46, Sec. 3.1.2] that because of the sign function in δ_{sat} in (2.26), $\boldsymbol{c}$ is prone to undergoing sudden change in the time domain at some time $\tilde{t}$.

Let us consider a generator in the oscillatory model with a valid type B (see Figure 2.10) MRPI set, and observe that it's barrier candidate intersects the $\omega = 0$ axis without changing direction.

Because δ_{sat} is ultimately also a function of the angular constraints, it is possible to continuously adjust $\boldsymbol{s}$ until the λ_2 sign change at $\omega = 0$ is introduced to the aforementioned barrier candidate, resulting in invalidity due to the jumping phenomenon.

3.2.2 Set Area Optimization Method

From Section 3.2 Point 2, a convenient approach would be defining the goal function as

$$f_s(\boldsymbol{s}) = \begin{cases} 0 \quad \text{if} \quad \exists \quad i \in \Gamma \cup \Lambda \setminus \{i_{ref}\} \quad s.t. \quad \mathcal{M}_i = \emptyset \\ \frac{1}{(N-1)} \frac{1}{A_{\text{avg}}} \sum_{i \neq i_{ref}} -A(\mathcal{M}_i(\boldsymbol{s})) \quad \text{otherwise} \end{cases} \tag{3.11}$$

where $A(\mathcal{M}_i)$ returns the area enclosed between the MRPI barrier(s) and the angular constraint(s) for the ith machine, that is, the area marked with blue in Figure 2.10. The total number of nodes in the system is denoted by N, and A_{avg} is an experimental value considered as the average MRPI set area, used to norm the function output approximately to the range defined in (3.2).

Once valid MRPI sets exist for all (but the reference) machine, and we know the corresponding MRPI barriers, summing up the area of said sets is as straightforward as calling MATLAB's `polyarea` function on a set of barrier points that are the result of numerically solving (2.23) with the right $\{\boldsymbol{\delta}_{\min}, \boldsymbol{\delta}_{\max}\}$ values.

Indeed, when researching the applicability of the set area method for finding valid MRPI sets, an implementation in MATLAB has been realized, supporting the method's capability to optimize for maximal MRPI set size. That is, if the initial optimization variable vector $\boldsymbol{s}_0$ builds valid MRPI sets for all nodes (with the possible exception of the reference node), the optimized vector $\boldsymbol{s}_{\text{sol}}$ generally brought bigger MRPI sets.

However, this method's applicability proves to be insufficient when starting with optimization variables $\boldsymbol{s}_0$ that do not result in valid MRPI sets for all the nodes. Before discussing why this might be, let us take a look at the flowchart of the proposed implementation in Figure 3.1.

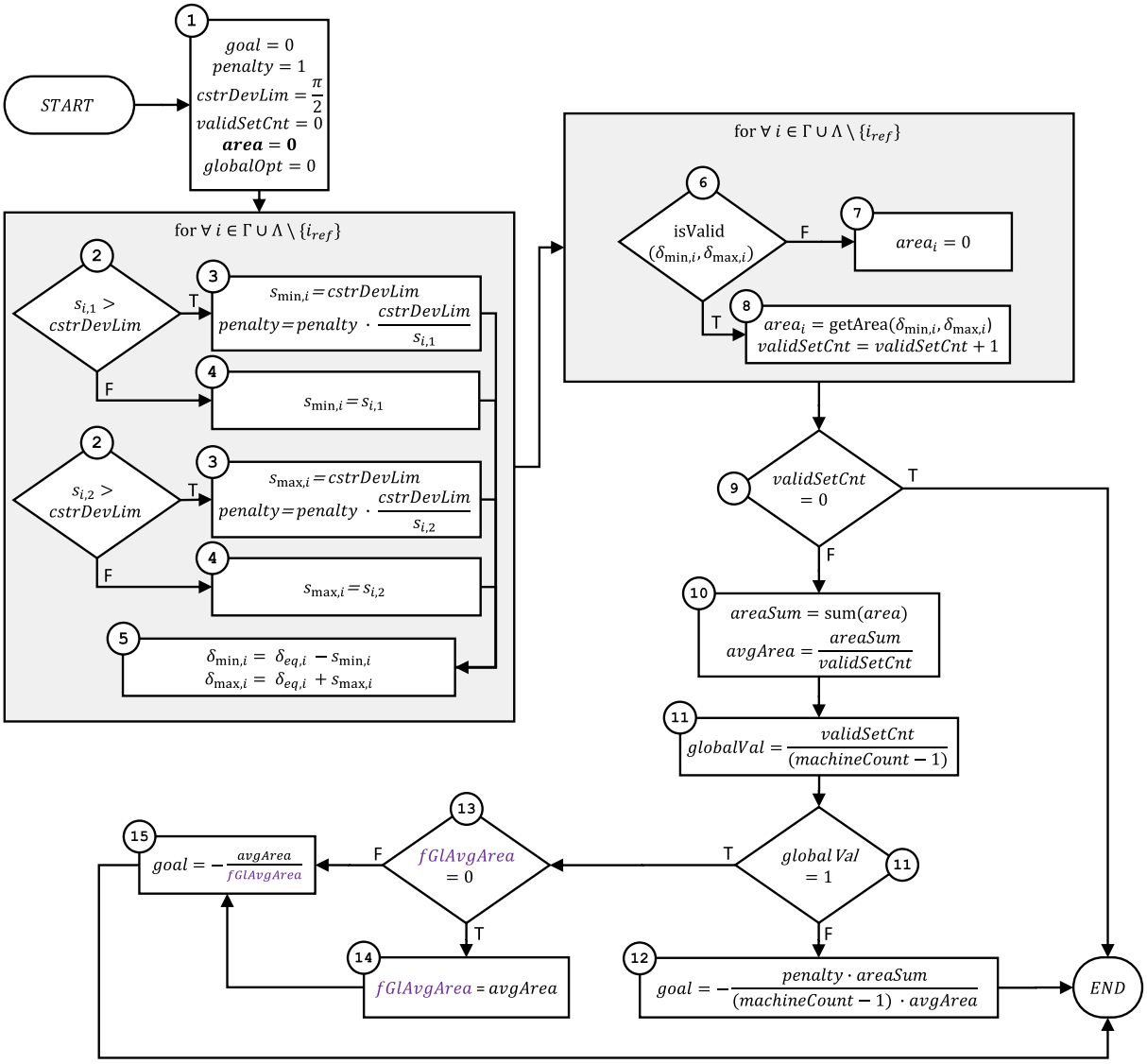


Figure 3.1: The area method for searching for a set of valid MRPI sets. The purple variable name denotes a global variable that is persistent between subsequent calls to the objective function.

Aforementioned figure shows numbered blocks, explanations to which are as follows:

1. First, initialization of variables take place.
2. Then, we enter a for loop in which all the machines will be examined, except for the one that represents the infinite bus. Optimization variables (that is, the lower-, and upper constraint candidates) are read, and checked whether they fulfill (3.8).
3. If not, then their value will be saturated to $cstrDevLim$, that is, $\frac{\pi}{2}$, and $penalty$ will be changed by an amount proportional to the amount by which the deviation limit was violated by the constraint candidate on hand. This is to discourage the superimposed optimization algorithm searching in that invalid direction, while maintaining the monotonicity of the objective function.

4. If (3.8) was met, then the constraint deviation $s_{\min,i}$ or $s_{\max,i}$ is set to the corresponding optimization variable.

5. Then, angular constraints are calculated. This has to be done for all machines at once, because subsequent function calls take in all the constraints at once. This is due to that the barrier dynamics (2.24) depends on each and every angular constraint $\{\boldsymbol{\delta}_{\min}, \boldsymbol{\delta}_{\max}\}$. In other words, subsequent steps cannot be executed sequentially with steps up to this point.

6. That being said, we enter a new loop in which for each machine (with the exception of the reference machine) we determine whether the supplied angular constraints build a valid MRPI set.

7. If this is not the case, we set the *i*th element of the *area* array to zero, indicating that the current optimization variables (or rather the angular constraints derived from them in the previous point) do not build a valid MRPI set for the *i*th machine.

8. If the current value of the optimization variables do build a valid MRPI set for the *i*th machine, then the area of the *i*th node's MRPI set is appended to the *area* array. Furthermore, the *validSetCnt* variable is incremented by one, so that after the loop, it includes the number of nodes at which the current optimization variables built a valid MRPI set.

9. Then, it is checked whether any valid MRPI sets have been found at all, and if not, the current iteration of the optimization returns with the initial *goal* still being zero.

10. Otherwise, the areas of all found valid MRPI sets are summed up, and their average is taken.

11. It is examined whether global validity is reached with the current set of optimization parameters, that is, whether they build valid MRPI sets for all but the reference machine. This is to ensure Section 3.2 Point 3.

12. If not, the current value of the goal function is calculated as follows: the average MRPI set area is multiplied by the number of non-reference machines, in a way estimating a total MRPI area sum if all nodes would be subject to a valid MRPI set. Then, the real total MRPI area sum gets divided by this value, resulting in a value that is basically the percentage of non-reference machines that posses a valid MRPI barrier. This gets further multiplied by *penalty* so that the return value of the objective function gets gradually more and more spoiled as optimization variables move further away from the set defined in (3.8). Finally, the negative of this value is returned as the value of the current iteration of the optimization, as the optimization problem is generally formulated as a minimization task.

13. If global validity has been reached, it is checked whether global (as in persistent between multiple calls to the objective function) variable *fGlAvgArea* —for "first global average area"— has already been set to any value other than it's initialization value 0.

14. If not, then this variable is set to the current MRPI area average. From this point onwards, *fGlAvgArea* will never again be modified, and thus will always contain the average area of MRPI sets when global validity was initially reached.

15. Finally, the objective function returns the negative of the ratio between the average MRPI area in the current iteration of the optimization and the average MRPI area when global validity was reached. Intuitively, this is a measure of how much further (in terms of average MRPI area) the optimization has come since reaching global validity. In other words, this is where Section 3.2 Point 2 is ensured.

As previously mentioned, this method does not work too well for cases in which the initialization vector of optimization variables $\boldsymbol{s}_0$ do not build MRPI sets already. This can be attributed to that the metric chosen for the definition of the objective function (MRPI set area) is one that is nonexistent for nodes that do not posses a valid MRPI set. Although a bit of a tautology, what this means in practice, is that the optimization algorithm —to rely on an anthropomorphism— can not iterate but only guess towards better $\boldsymbol{s}$ values.

In other words, set area optimization does not fulfill Section 3.2 Point 7, because $f_s(\boldsymbol{s})$ would only take on $N-1$ discrete values between $[-1, 0)$, (N being the number of nodes in the system) if it wasn't for the penalty variable.

Nonetheless, once global validity —as in Section 3.2 Point 3— is reached, aforementioned disadvantage ceases to exist and so $f_s(\boldsymbol{s})$ becomes at continuous at least on subsets of the $[-\infty, -1)$ interval.

A big advantage of the set area method is that it is able to work with both type A and type B MRPI sets as described in Section 3.2 Point 5 — a property that, as we will see in Section 3.2.3 can not always be fulfilled.

On the other hand, when it comes to computational complexity and parallelization as in Section 3.2 Point 6, the set area method may not be the most efficient one. Even though some parts of the process —such as the actual area calculation— are well parallelizable, a prerequisite is knowledge of the barrier candidates, which must be obtained through computationally rather expensive numerical integration anyway.

Another drawback of the set area method is that once global validity is reached, the definition of the objective function becomes rather ambiguous in that it is hard to precisely define what exactly A_{avg} in (3.11) must be. In the above implementation it has been considered as the average of the actual MRPI set areas in the system at the time when global validity has been reached, even though it could be defined otherwise. For example, some estimation based on the current angular constraint boundaries, or other experimental constant could be used.

3.2.3 Span Optimization Method

Although the area method described in 3.2.2 proved to be working for maximizing the area of the MRPI set when the initial set of design variables (angular constraints) already define a valid, coherent MRPI set for all the nodes other than the reference node,

it was less successful at finding such values when the initial design variables do not build valid MRPI sets for most nodes. It has been shown that this was due to the objective function's discontinuity in the globally invalid region. Besides, it has been mentioned that the area method is computationally rather intensive.

Thus, an attempt was made at eliminating aforementioned issues by constructing a method that relies on a different metric to the MRPI set size, and which will be referred to as the span method.

The span method makes a simplification in that it disregards Section 3.2 Point 5, and only considers type A MRPI sets as valid. The objective function thus makes the (indeed inaccurate) assumption that a valid MRPI barrier —going backwards in time from the point of ultimate tengentiality— must first intersect the opposite angular constraint boundary, followed by the $\omega = 0$ axis.

Hence, the span method considers the first position where the MRPI candidate trajectory intersects the $\omega = 0$ axis when integrating backwards in time. This would be $\delta_{j,U,i}$ and $\delta_{j,L,i}$ for the ith node's upper-, and lower trajectory candidate respectively as shown in Figure 3.2. The fundamental assumption hereby is that if the first ever crossing of the $\omega = 0$ axis falls outside the region of permitted angular constraints, then the MRPI barrier candidate must have crossed the closest angular constraint boundary before (again, integrating backwards in time) that.

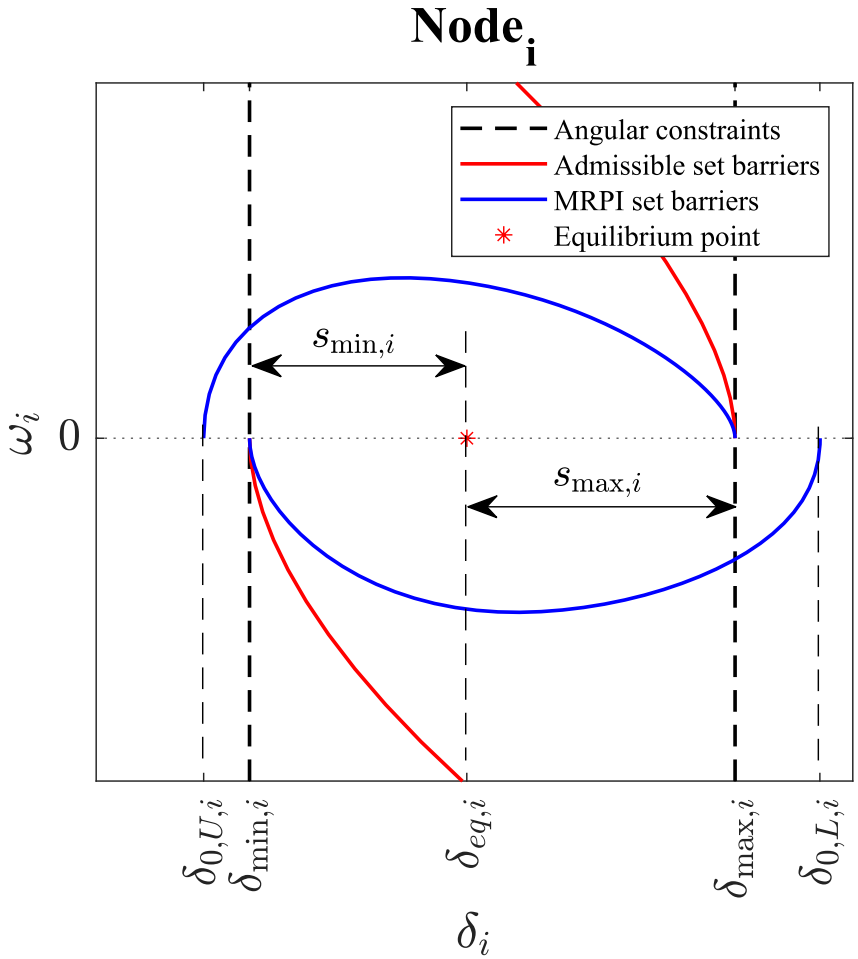


Figure 3.2: Graphical representation of the span method's workings.

To put it another way, it is assumed that if the candidate trajectory does not touch the $\omega = 0$ axis between $\delta_{\min,i}$ and $\delta_{\max,i}$, then it must be valid as

- it must reach the opposite constraint, that is, $\delta_{\min,i}$ in case of the upper barrier candidate, and $\delta_{\max,i}$ in case of the lower one.
- no jumping can occur, because the jumping phenomena is unique to $\omega = 0$.

As with the set area method, a MATLAB implementation of the span method has been realized. The inner workings of the corresponding objective function is shown in Figure 3.3.

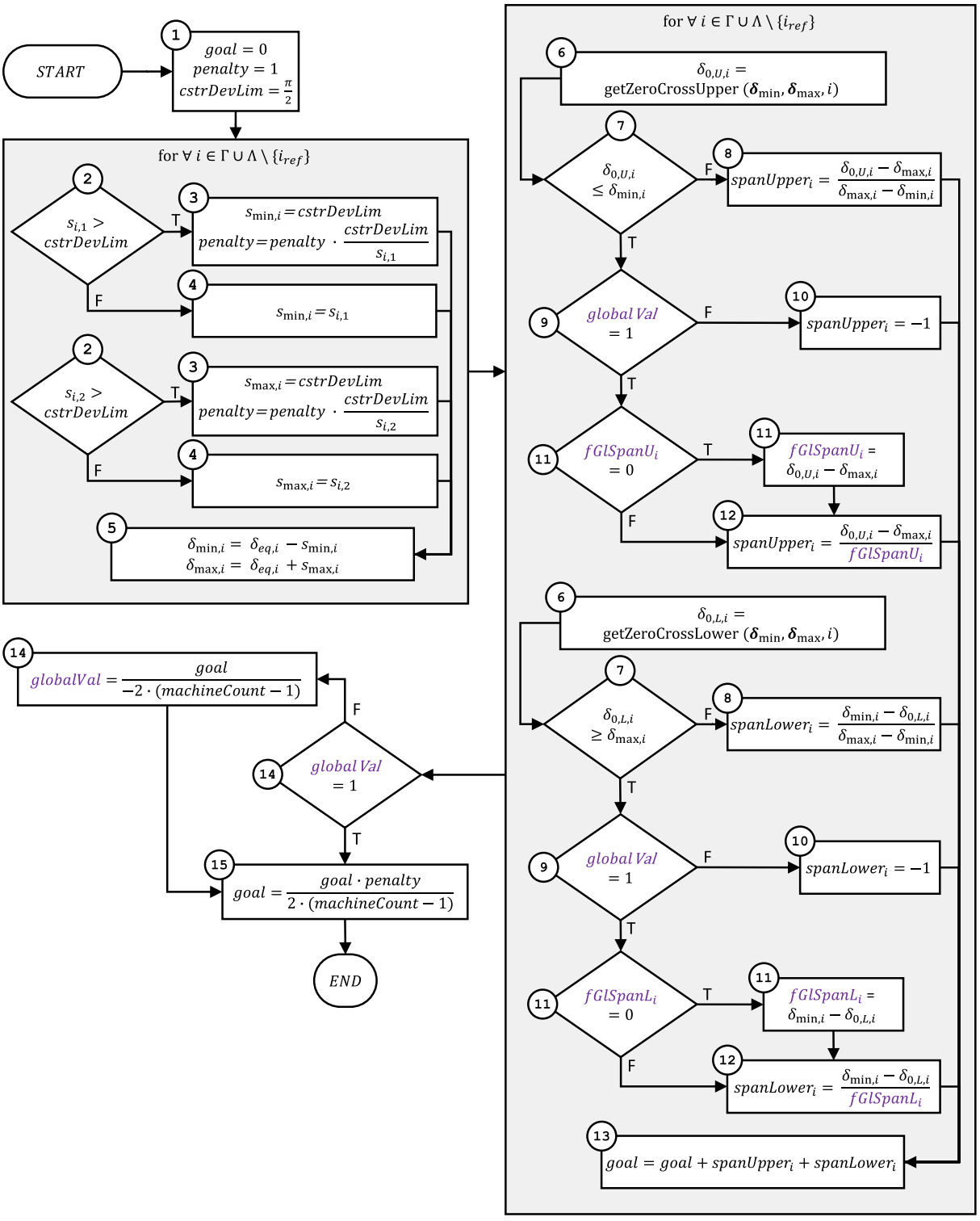


Figure 3.3: The span method for searching for a set of valid MRPI sets. The purple variable name denotes a global variable that is persistent between subsequent calls to the objective function.

Further explanations for the numbered blocks in above figure are hereby given, serving as a kind of walk-through:

1.-5. These steps are identical to steps 1.-5. in the set area method in Figure 3.1.

6. In the next for loop , $\delta_{0,U,i}$, and $\delta_{0,L,i}$, that is, the upper and lower $\omega = 0$ axis intersections are determined.

7. Then, it is checked whether the points of $\omega = 0$ intersection are between the lower-, and upper angular constraints that have been determined at the end of the previous for loop.

8. Is this not the case, then the negative of the procentual distance between the barrier candidates' point of ultimate tangentiality ($s_{\min,i}$ or $s_{\max,i}$) and the barrier candidates' point of the first $omega = 0$ axis intersection ($\delta_{0,L,i}$ or $\delta_{0,U,i}$) as related to the whole distance between the constraints ($s_{\min,i}$ and $s_{\max,i}$) are calculated. The absolute value of expression will be further referred to as the **span** of the type A MRPI barrier candidate in question. By using procentual values, all distances are normalized to the same interval, preventing bigger absolute distances to overshadow smaller ones of other nodes.

9. Otherwise, it is checked whether global validity (as in Section 3.2 Point 3) has already been reached.

10. If not, then $distLower_i$ or $distUpper_i$ is set to -1, that is, negative 100%, denoting that the corresponding barrier trajectory crosses 100% of the distance between $s_{\min,i}$ and $s_{\max,i}$. This is to ensure Section 3.2 Point 3, that is, to privilegize global validity. In other terms, the algorithm will not be seeking better MRPI barriers until all barriers are valid type A barriers, which means that all but the reference node possesses valid MRPI sets.

11. If global validity has been reached, then another check is carried out to determine whether global variable $fGlSpanU_i$ ($fGlSpanL_i$) (for "first global span upper (lower)", denoting the absolute value of the span of the *i*th node upper (lower) barrier candidate's span right after global validity has been reached) is still initialized to zero. If so, the current span is being assigned to the variable in question.

12. Then, or if $fGlSpanU_i$ ($fGlSpanL_i$) was already set before, the procentual value is calculated much like in Point 8, but instead of the distance between the current constraints, the distance of when global validity has been reached is taken as a basis. This is necessary, because otherwise a given barrier candidate's contribution to the objective function value would not change if the barrier candidate crosses the $\omega = 0$ axis at the same procentual span value, regardless if the span distance got bigger or smaller in absolute terms. For example, if a barrier intersected the opposite constraint boundary exactly in $\omega = 0$, then that barrier's contribution to the objective function value (see Point 8 above, and *spanUpper* and *spanLower* in Figure 3.3) would be -1. However, if the span of the examined barrier (and possibly the MRPI set area) got bigger, but the barrier's $\omega = 0$ axis intersection point would still lie exactly on the opposite constraint boundary, the barrier's contribution to the objective function value would still remain -1.

13. These procentual values are then being appended to the goal function's return value *goal*. The reason for relying on these procentual values instead of absolute ones is that we are mostly interested in finding *any* valid MRPI sets, and are not interested in further optimization with this span method.

14. If global validity has been just reached in the current iteration of the optimization algorithm, the global (as in persistent between multiple calls to the objective function) variable *globalVal* is set to 1, and won't be further modified in subsequent iterations. Otherwise it is set to some value other than 1.

15. Finally, the *goal* value is multiplied by the penalty multiplier, and is averaged to the number of non-reference nodes' two (minimum and maximum) constraints per machine.

Once objective function returns -1, then a set of constraints has been found that correspond to a set of valid MRPI barriers for each machine, and the optimization halts.

Formally, the span method's objective function could be formulated as

$$f_s(\boldsymbol{s}) = \begin{cases} \frac{1}{N-1}\sum_{i\neq i_{ref}} \mathrm{span}_i(\boldsymbol{s}) \text{ if } \nexists\ i \in \Gamma \cup \Lambda \setminus \{i_{ref}\}\ \ s.t.\ \ \mathcal{M}_i^A = \emptyset \\ \frac{1}{N-1}\sum_{i\neq i_{ref}} (\max(\mathrm{span}_i(\boldsymbol{s}), -1)) \text{ otherwise} \end{cases} \tag{3.12}$$

where span_i is as described in Point 8 of the description of Figure 3.3. $\mathcal{M}_i^A$ denotes the ith node's type A MRPI set, and N is the number of nodes in the system.

Even though the span method does not set out to fulfill Section 3.2 Point 2 directly like the set area method does, it still fulfills it to some extent, albeit with restrictions:

- Because of how MRPI barrier candidate dynamics tend to work, it is theorized that barrier candidates that evolve further away —going backwards in time— from the point of ultimate tangentiality tend to intersect the opposite angular constraint boundary at a higher $|\omega|$ value, resulting in a greater MRPI set area (given that the node in question has another valid type A barrier). If this assumption holds, then the span can indeed be a viable metric.

- However, aforementioned assumption definitely does not work the other way around, because we know from experience that perfectly valid type A MRPI sets exist that do not cross the $\omega = 0$ axis at all. These, although being perfectly valid type A barriers, would not be considered by the proposed algorithm.

- Another drawback is that the span method disregards Section 3.2 Point 3 and type B MRPI sets altogether, some of the MRPI sets cannot even be examined for optimality.

Advantages to the span method mostly have to do with Points 6 and 7 of Section 3.2:

- As for the computational costs, the span method is indeed cheaper than the set area method, because no costly area calculation has to be done. In fact, backwards integration gets broken as the barrier candidate intersects the $\omega = 0$ axis, and so the span can be directly calculated from the last δ value of the integration.
- Also, the objective function is —although not continuous in the mathematical sense because of the reasons described in Section 3.2.1— it at least takes on *many* values between $[-1, 0)$.

3.3 Obtaining the Equilibrium Point

We set out to finding a frequency-synchronous state of all generators of the examined system, so that their angular velocities are equal.

$$\dot{\delta}_i = \ldots = \dot{\delta}_n \quad \forall\, i \in \Gamma \cup \Lambda \tag{3.13}$$

Furthermore, since —as we have discussed in Section 2.1 — we are considering the node with index i_{ref} as a reference node with a constant angular deviation of 0

$$\omega_{ref} = \dot{\delta}_{ref} = \ldots = \dot{\delta}_i = \dot{\delta}_{i+1} = \ldots = 0 \tag{3.14}$$

Considering said state variables of (2.1), the equilibrium point is a set of points (generator states)

$$\begin{gathered} \boldsymbol{x}_{eq,i} = \begin{bmatrix} x_{eq,1,i} \\ x_{eq,2,i} \end{bmatrix} = \begin{bmatrix} \delta_{eq,i} \\ \omega_{eq_i} \end{bmatrix} = \begin{bmatrix} \delta_{eq,i} \\ \dot{\delta}_{eq,i} \end{bmatrix} = \begin{bmatrix} \delta_{eq,i} \\ 0 \end{bmatrix} \\ \boldsymbol{x}_{eq} = \begin{bmatrix} \boldsymbol{x}_{eq,1} \\ \boldsymbol{x}_{eq,2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_{eq} \\ \mathbf{0} \end{bmatrix} \end{gathered} \tag{3.15}$$

so that everything else left unchanged, each and every generator's angular deviation also remains unchanged. By substituting (3.15) into (2.3) and considering that the very definition of an equilibrium point is that the system dynamic remains zero when evaluated at that point we get:

$$\dot{\boldsymbol{x}}_{eq,i}|_{\boldsymbol{x}=\boldsymbol{x}_{eq}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.16}$$

Since the MRPI sets -if exist- must enclose the equilibrium points by definition, the vector of equilibrium points $\boldsymbol{\delta_{eq}}$ will serve as a starting point for finding valid MRPI sets in our examinations.

For a given system, the equilibrium point is found by solving equation (3.16) for all machines:

$$\dot{\boldsymbol{x}}_{eq} = \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix} = \mathbf{0} \tag{3.17}$$

Because $\dot{\delta}_i = \omega_i$ in (2.3), and because $\dot{\delta}_i = 0$ in (3.15), it follows that $\omega_i = 0$ can be substituted into (2.3) and (2.4).

Then, for second order oscillators (generators):

$$(3.16) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = (2.3)|_{\omega_i=0} = \begin{bmatrix} 0 \\ A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_i - \delta_j - \gamma_{ij}\right) \end{bmatrix} \quad \forall\, i \in \Gamma \tag{3.18}$$

whereas for first order oscillators (loads):

$$(3.16) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = (2.4) = \begin{bmatrix} A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_i - \delta_j - \gamma_{ij}\right) \\ 0 \end{bmatrix} \quad \forall\, i \in \Lambda \tag{3.19}$$

Merging (3.18) and (3.19) results in a system of nonlinear equations

$$f_{eq,i}(\boldsymbol{\delta}) = A_i - \sum_{j \in \mathcal{N}_i} K_{ij} \cdot \sin\left(\delta_i - \delta_j - \gamma_{ij}\right) = 0 \quad \forall\, i \in \Gamma \cup \Lambda \setminus \{i_{ref}\} \tag{3.20}$$

where $\boldsymbol{f}_{eq}(\boldsymbol{\delta}) = [\cdots f_{eq,i}(\boldsymbol{\delta}) \cdots]^T : \mathbb{R}^(N-1) \to \mathbb{R}^(N-1)$ is the function describing equilibrium dynamics for all oscillator nodes. In practice, (3.20) is solved by utilizing MATLAB's numerical solver for a system of nonlinear equations, `fsolve` [63].

3.4 Overview of Common Optimization Algorithms

The general form of an optimization problem has already been expressed in (3.1). However simple this general mathematical formulation may be, conducting the actual optimization process in practice is less profoundly outlined. Nonetheless, it always involves feeding the objective function to an optimization algorithm, often referred to as an optimizer or solver.

To avert future confusion, at this point we would like to clarify optimization-related terminologies, and their usage throughout this thesis. In the context of optimization

- an optimization **method** refers to the structure of the objective function. More specifically, to (3.11) of Section 3.2.2, and (3.12) of Section 3.2.3.
- an optimization **algorithm** refers to one of the algorithms in the coming subsections. These take an objective function (the implementation of an optimization *method*; see above point) as an argument, and utilize (make calls to) it as dictated by their inner logic.

- the **optimizer** or **solver** refers to a specific implementation of an optimization *algorithm*.

The generalized working of an optimizer is shown in Figure 3.4.

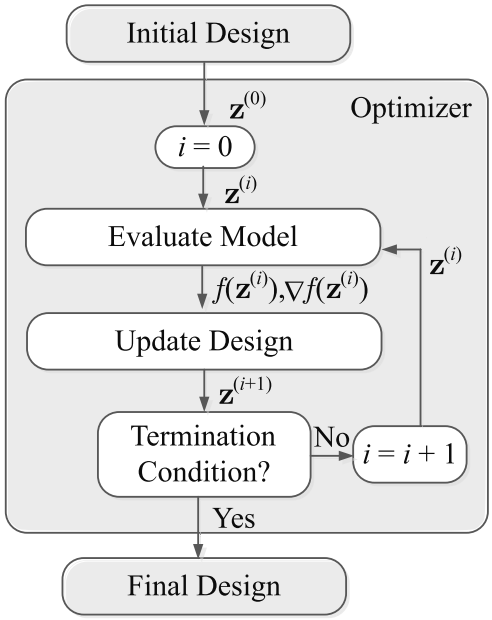


Figure 3.4: Optimization flow. Adapted from [35, Fig. 3.1].

The algorithm basically evaluates the objective function (referred to as the "model" on Figure 3.4) at a number of points in the design space, retrieving the goal function values, and —with some algorithms— its derivatives in said points. Based on these, the optimizer makes a proposition about at what points the objective function shall better be evaluated, so that the minimum of the objective function values evaluated at those points would be lower than any objective function value seen so far. Accordingly, the objective function gets evaluated in these propose points, forming a loop that goes on until some termination condition (maximum iteration count, minimum resolution, time constraint, etc.) is met. By the time said termination criterion is met, a set of design parameters are produced, so that the evaluation of the objective function at that point approximates the function's actual minimum value.

Now, in *what* points the objective function gets evaluated, and *how* the optimizer makes a proposition about the adjustment of the design parameters for the next iteration, depends on the optimizer itself. Indeed, optimizers come in many varieties: some work best on a specific set of problems, some are more general, some rely on heuristics, some are more rigorous, etc. Available approaches towards optimization may vary based on the design vector $\boldsymbol{z}$, the objective function $f(\boldsymbol{z})$, and the equality and inequality constraints $\boldsymbol{g}(\boldsymbol{z})$ and $\boldsymbol{h}(\boldsymbol{z})$ respectively [3][10][33][39]. Important considerations therefore include the following:

1. **Univariate vs. multivariate** optimization, as determined by the size of the design vector.

2. **Global vs. local** optimization. Global optimization attempts to find the global minimum or minima, whereas local optimization aims to find one or some of the local optima over all feasible $\boldsymbol{z}$. Most optimization algorithms are only capable of determining a local minimum without making a statement about global optimality. Those which do are either custom-tailored to the problem in question or are exponentially difficult.

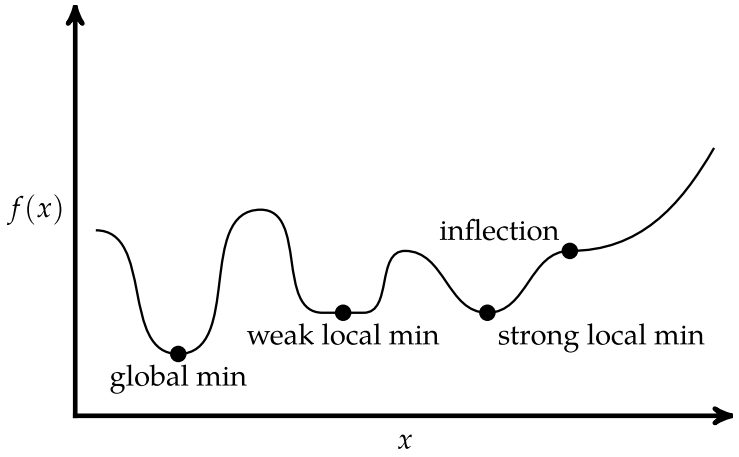


Figure 3.5: Global minimum, local minima, and inflection point of the goal function of an univariate optimization problem. From [33, Figure 1.6].

3. **Convexity** of the feasible the objective function, and the constraint functions: for a convex problem, any local optimum is also a global optimum. Furthermore, for convex problems, efficient solution algorithms are generally available. As [8] puts it: *"With only a bit of exaggeration, we can say that, if you formulate a practical problem as a convex optimization problem, then you have solved the original problem."*

4. **Constraint types.** An optimization problem may either be unconstrained, or subject to one or more constraints as outlined in (3.1). Constraints might be simple bounds (also colled box constraints), although linear or quadratic constraints are also possible, as well as any constraint that can be expressed as a set of equalities and/or inequalities. Generally speaking, unconstrained and box-constrained problems are the most easy to work with.

5. **Differentiability** of the objective function. Many of the most efficient algorithms rely on the gradient of the objective function, although derivative-free approaches also exist [29], the latter also being known as *pattern search* or direct algorithms [33, Chapter 7].

6. **Discrete vs. continuous** optimization. In general, discrete optimization such as integer programming or combinatorial optimization is much harder harder than a similar but continuous problem [29].

7. **Static vs. dynamic** optimization problems. While the former is one that is invariant of time and can be defined in the form outlined in (3.1), the latter *"involve dynamic variables whose values change in time"* [20, Sec. 15.1].

8. **Deterministic vs. stochastic** optimization. In the broadest sense, the latter refers to any optimization problem that involves randomness and/or probability. In fact, this definition is —to some extent— ambiguous in that according to [54, Sec. 1.1.3], stochastic optimization may either refer to
   a) an optimization problem where either the objective function and/or the constraints are subject to random noise and/or
   b) an optimization algorithm which involves randomness in its workings to find optima of a static or stochastic optimization problem.

Before we can proceed with an overview of optimization algorithms that are relevant for the purposes of our optimization problem set up in Section 3.1, we ought to classify it in accordance with the considerations listed above. Our problem at hand is...

- ...multivariate, because the vector of design variables $\boldsymbol{z}$ consists of the minimum and maximum allowable angular constraints $\delta_{\min,i}$ and $\delta_{\max,i}$ for every generator other than the reference generator as defined in Section 3.1.
- ...global, because we are interested in finding the global minimum of the objective function.
- ...non-convex, because we are unable to make a statement about the convexity of neither of the two objective functions that we have defined in (3.11) and in (3.12).
- ...box-constrained, because no equality constraints has been set, and the inequality constraints defined in (3.1) are simple bounds.
- ...not differentiable neither for the set area method of Section 3.2.2 nor for the span method outlined in Section 3.2.3. With the former, the non-differentiability can be attributed to the possible discontinuity in (3.11). Although the latter approach eliminates such an abrupt change in goal function value, the gradient of the function still cannot be explicitly defined. This is due to that the goal function is dependent on the span in (3.12), while the span is dependent on the MRPI barrier candidate trajectory's first (going backwards in time) $\omega = 0$ axis crossing position. However, this position cannot be determined other than through conducting backwards numerical integration until the crossing itself. The resulting trajectory is dependent not only on angular constraints of machines that are adjacent to the observed generator in the oscillatory model, but also of the angular constraint which it tangentially intersects. Since this applies to each and every MRPI barrier candidate, a statement about what effect changing a single angular constraint has on the goal function is analytically not foreseeable, let alone the direction of greatest change when interacting with multiple constraints.
- ...continuous, because the optimization variables and the goal function are real valued.
- ...static, because the optimization variables and the goal function value are not a function of time (as in time elapsing during the optimization process) nor does it

involve optimizing for a set time period. Indeed, the MRPI set in Def. 3 considers *all future evolutions*.

- ...deterministic, because neither randomness nor noise is involved at any point.

Based on the above criteria, common optimization algorithms that might be applied to the optimization problem at hand include those, that will be introduced in the following subsections.

3.4.1 Genetic Algorithms

A genetic algorithm is a gradient-free metaheuristic that is inspired by mechanics of natural selection and genetics as in Charles Darwin's theory of natural evolution. It combines a "survival of the fittest" approach with structured yet randomized information exchange to form a search algorithm [22]. More precisely, the genetic algorithm draws analogy between structured information and a chromosome, considering elementary building blocks of said information structure as one would consider genes, whereas their positions as loci of the chromosome. In its most basic implementation, a fixed-length byte array is referred to as a *chromosome*, and a bit or a byte position (or really the position of any fitting data structure) might be referred to as a *locus*. In the context of numerical optimization, each chromosome can be thought about as the binary representation of a design vector $\boldsymbol{z}_i$ from (3.1). With that, each chromosome can be assigned a fitness level, which is just the objective function value $f(\boldsymbol{z}_i)$. Having defined the fundamental concepts, the basic task flow of a genetic optimization algorithm involves the following steps [44, Sec. 1.6]:

1. A set of N chromosomes $\boldsymbol{z}_1 \ldots \boldsymbol{z}_i$ each of length l are randomly generated. A set of N chromosomes is further referred to as a *population*, whereas the population generated in this step is specifically the *parent population*.

2. The fitness of each chromosome $f(\boldsymbol{z}_i)$ is calculated.

3. The following substeps are repeated over and over until N chromosomes, i.e. a new population —members of which are called *children* or *offspring*— is created.

   a) **Selection.** Selects a pair of chromosomes from the parent population in such a way, that the probability of choosing any specific chromosome shall be proportional to its fitness, i.e. better performing chromosomes have a greater chance to be chosen. It is worth noting that selection takes place with replacement so that the same chromosome might be chosen multiple times.

   b) **Crossover.** Crosses over the previously selected two chromosomes. This constitutes an operation in which two offspring are created by splitting both chromosomes at the random locus $1 \leq k \leq l$ selected from a uniform probability, and switching up the genes (bits) up to that point between the two chromosomes as shown in Figure 3.6. However, a crossover event only occurs with a $0 < p_c < 1$ crossover probability (typically $p_c \approx 0,7$[52]); in

other cases the crossover is considered to be undertaken at the 0th locus, i.e. the selected chromosomes are left unmodified.

c) **Mutation.** Mutates the two children chromosomes at each and every locus with the mutation probability $0 < p_m < 1$ (generally being very small, with $p_m \approx 0,001$ being a rather typical value [52]). The resulting chromosomes are placed in the children population. If N is odd, any single offspring can be discarded at random.

4. The current population gets replaced with the new population, or rather the new *generation* of the population.

5. Finally, a pre-set termination condition is examined to determine whether the algorithm is to be halted. Typical halting criteria include...
   - ...having produced a chromosome with a sufficient fitness level, or...
   - ...having reached a pre-defined number of generations, or...
   - ...not having been able to come up with an improvement for a pre-defined number of generations.

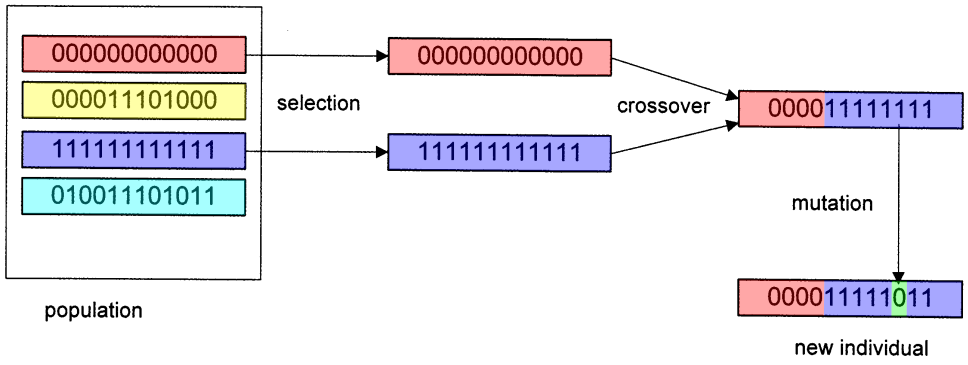


Figure 3.6: Genetic operations. Adapted from [52, Fig. 2.].

3.4.2 Particle Swarm

The Particle Swarm optimization algorithm is a gradient-free metaheuristic, which introduces a flock-like dynamic that might —depending on the optimization problem at hand— help in finding the global minimum over the local minima.

The inner workings of the algorithm are shown in Figure 3.7. The algorithm initially configures a few constants (whose role we will shortly see), and sets up a population of candidate solutions, referred to as a swarm of particles. The initial placement of particles is usually chosen to be uniform throughout the design space. From this point onward, each particle behaves more or less as an *autonomous object* in that the ith particle encapsulates its current position $\boldsymbol{z}_i^{(k)}$, current speed $\boldsymbol{v}_i^{(k)}$, its best position so far $\boldsymbol{p}_{\text{best},i}^{(k)}$ (called the personal best). Furthermore, it has access to the best position among the whole swarm so far $\boldsymbol{g}_{\text{best}}^{(k)}$ (called the global best), and —in some implementations—

to the best position among a pre-defined subset (neighborhood) of particles so far $\boldsymbol{l}_{\text{best}}^{(k)}$ (referred to as the local best). With that, once the initial swarm has been placed, the particles' internal state will be redetermined in each simulation step according to

$$\begin{aligned}\boldsymbol{v}_i^{(k+1)} = r\Bigg(w\boldsymbol{v}_i^{(k)} + u_1 c_p \left(\boldsymbol{p}_{\text{best},i}^{(k)} - \boldsymbol{z}_i^{(k)}\right) \\ + u_2 c_l \left(\boldsymbol{l}_{\text{best},i}^{(k)} - \boldsymbol{z}_i^{(k)}\right) + u_3 c_g \left(\boldsymbol{g}_{\text{best}}^{(k)} - \boldsymbol{z}_i^{(k)}\right) \Bigg)\end{aligned} \tag{3.21}$$

Where $0 < w \leq 1$ is the inertia constant, u_1, u_2, and u_3 are each different random numbers sampled from a uniform distribution between 0 and 1, whereas c_p, c_g, and c_l are user-defined positive coefficients for the personal , global, and local vectors (the latter is omitted from, while the latter two are referred to as the "individual" and "social" vectors in Figure 3.7). Furthermore, $0 < r \leq 1$ is a scale factor that might be constant, or be decreasing with each subsequent iteration, limiting the movement of particles more and more as time progresses.

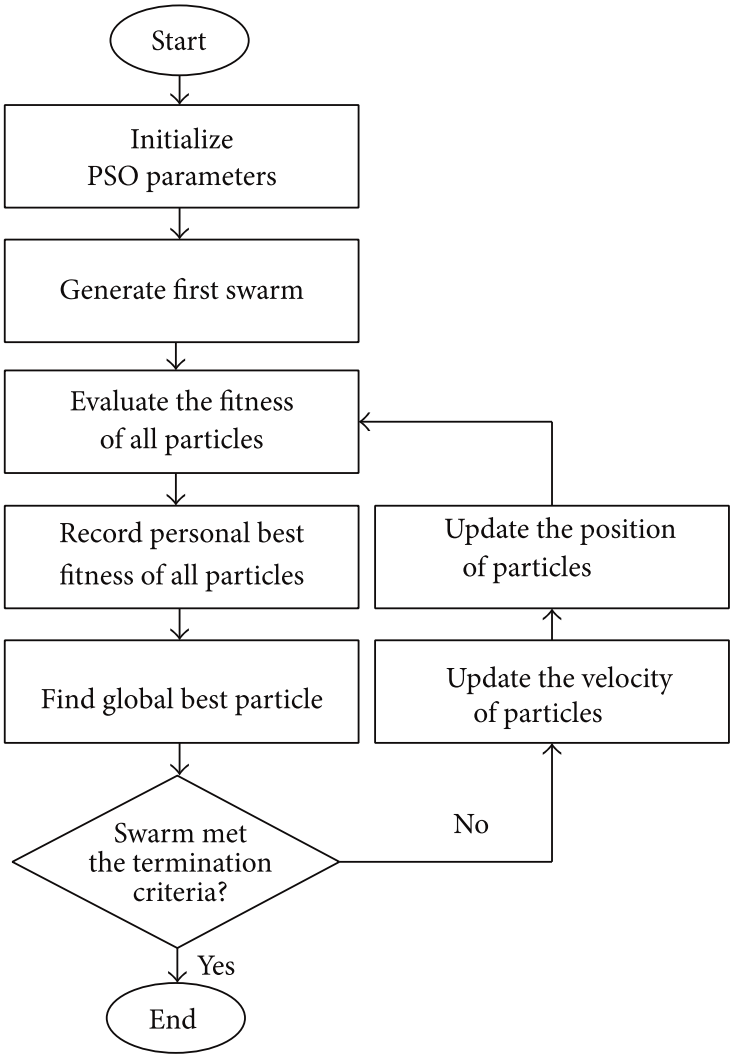


Figure 3.7: Descriptive flow diagram for pattern search. Note that this flow chart does not account for the case when the optional local best positions are also considered. From [30, Figure 1.].

As time goes by, the particles all get *pulled* generally towards the best solution so far (although affected by the random parameters), and so after enough time, all all particles tend to *gather* around a minimum. The bigger the initial swarm size and resolution (i.e. how close to each other individual particles are), the bigger the likelihood that one ore more particles would *gravitate* into the global minimum, *pulling in* the rest of the swarm.

Finally, the whole algorithm is halted if a termination criteria is reached. These commonly include the maximum number of iterations, finding a satisfactory solution, or not being able to better the best result so far for a predefined span of time [30].

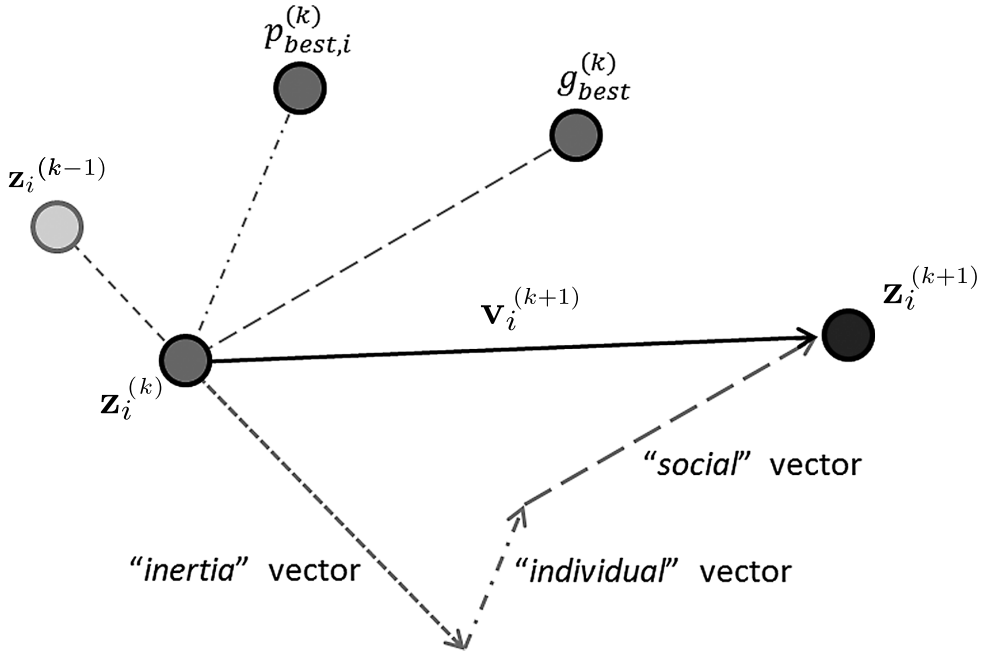


Figure 3.8: Displacement of a particle during particle swarm optimization. Note that the local best position, and the resulting vector is not taken into consideration on this figure, as it is optional. From [56, Fig. 1.].

3.4.3 Pattern Search

Pattern search is a class of derivative-free optimization heuristics. The algorithm considers a base point and a pattern of exploratory points around the base point. The algorithm's workings are outlined in figure Figure 3.9.

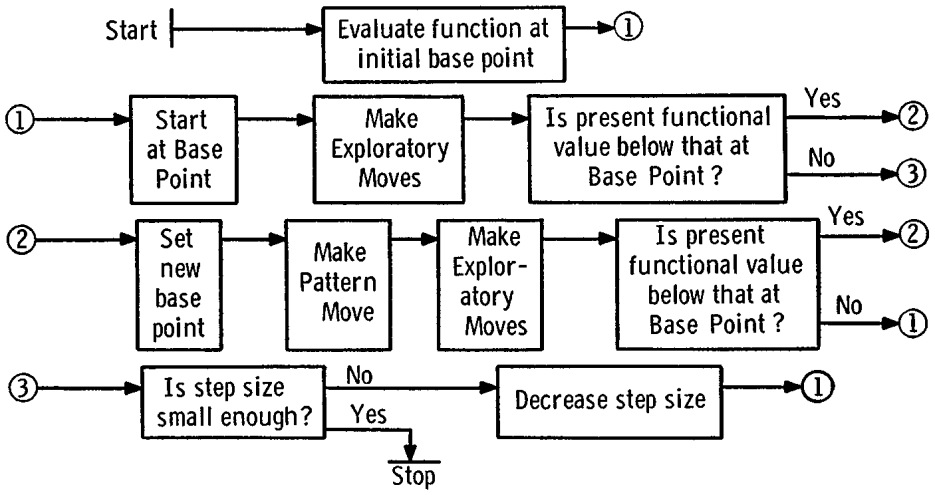
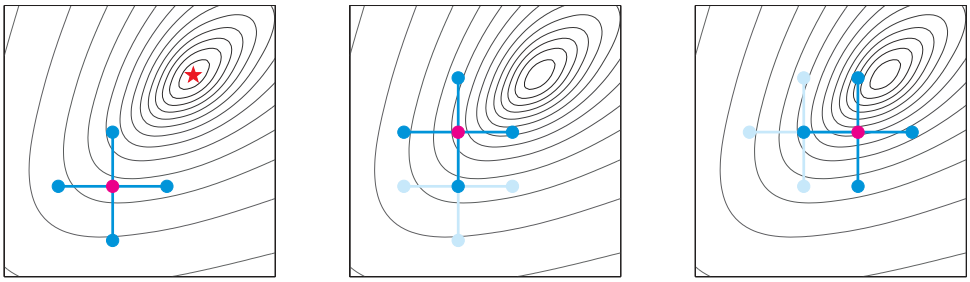


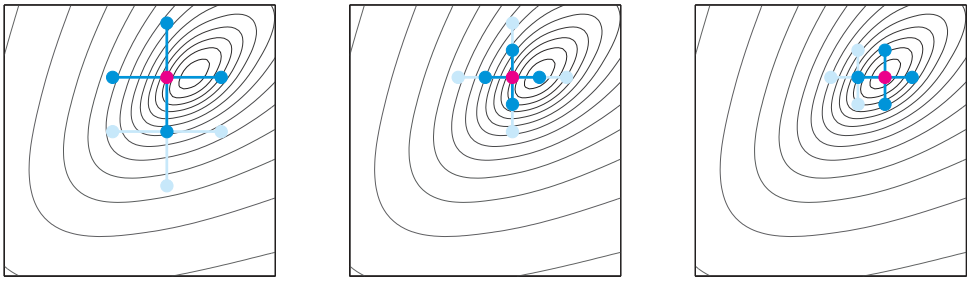
Figure 3.9: Descriptive flow diagram for pattern search. From [26, Chart 1.].

The initial set of design parameters are chosen either randomly or through some problem-specific consideration. These define the base point, and with that, the exploratory points around the base point. The goal function value is evaluated in all of these points (unless they fall outside of optimization constraints). Then, the exploratory

point with the lowest goal function value is compared with that of the base point. In case the former is lower than the latter, the whole pattern is shifted, so that this point now becomes the new base point as shown in Figure 3.10(b), and the process repeats. However, if the goal function does not evaluate lower in any of the exploratory points than in the base point, the pattern gets contracted (typically) to half its size as shown in Figure 3.10(e) Once a pre-defined minimal size of the pattern (accuracy) is reached, the algorithm stops and the last base point is considered as the proposed optimum solution.



(a) Initial pattern (b) Move North (c) Move West



(d) Move North (e) Contract (f) Move West

Figure 3.10: Pattern search applied to a bivariate function. The red star in (a) represents the minimum of the goal function. From [34, Fig. 1.1].

3.4.4 Simulated Annealing

Simulated Annealing is a probabilistic optimization algorithm that attempts to avoid local minima for the global optimum through a metaheuristic that is derived from metallurgy [60]. The general idea is to simulate a cooling process similar to that of annealing in metallurgy. Hereby a slow, controlled cooling process ensures that atoms have enough time for arranging themselves into a low-energy state that is close to the optimum. A particularity of the algorithm in question is that under some —probabilistic— circumstances the best result at any time might be disposed of for a worse set of design parameters in order to escape local minima. The inner workings of the algorithm is shown in Figure 3.11.

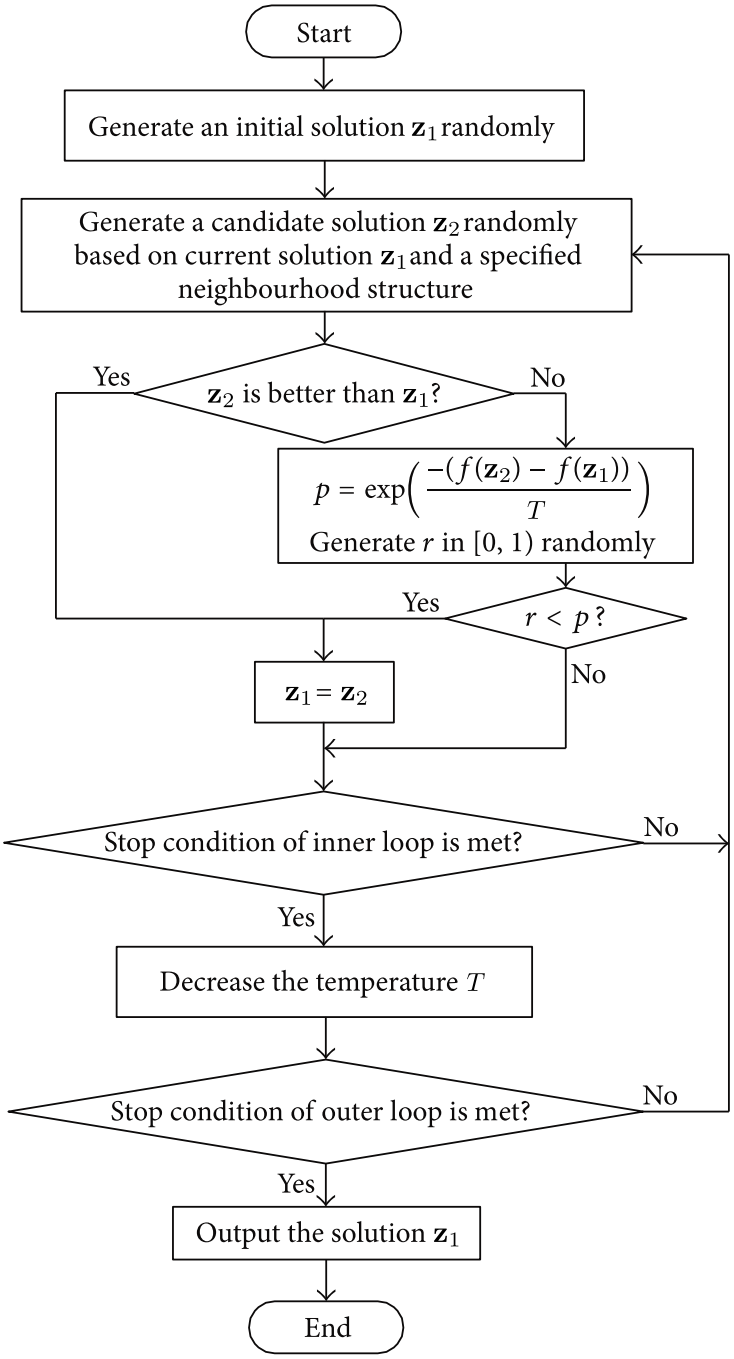


Figure 3.11: Flowchart of the simulated annealing algorithm. Adapted from [60, Figure 1.].

Initially, a base point $\boldsymbol{z}_1$ is chosen either randomly, or based on some engineering consideration. Then, another design vector $\boldsymbol{z}_2$ is randomly selected from the neighborhood of $\boldsymbol{z}_1$. What constitute neighbors of a state is defined by the means of a neighborhood structure, which —for continuous problems like ours— this often means sampling a Gaussian distribution around $\boldsymbol{z}_1$. Next, it is examined, whether the newly selected point of the design space is better than the current one, i.e. whether $f(\boldsymbol{z}_2) \leq f(\boldsymbol{z}_1)$. If so, $\boldsymbol{z}_2$) is chosen right away as the current base point. However, even if it is worse than the current one $f(\boldsymbol{z}_2) > f(\boldsymbol{z}_1)$, the same might just happen, if acceptance probability p happens to be higher than a random variable r sampled from a uniform distribution between 0 and 1. The acceptance probability is calculated based on the Boltzmann prob-

ability distribution [53, Sec. 10.9] and the amount by which the worst position is worse than the base point:

$$p = p(\boldsymbol{z}_1, \boldsymbol{z}_2, t) \begin{cases} 1 \text{ if } f(\boldsymbol{z}_2) \leq f(\boldsymbol{z}_1) \\ \exp\left(-\frac{f(\boldsymbol{z}_2) - f(\boldsymbol{z}_1)}{T}\right) \text{ else} \end{cases} \tag{3.22}$$

With this, the bigger the difference between the proposed position and the base position, the closer p gets to 0. Furthermore, p also gets lower and lower as the temperature T gets lowered. Consequently, the probability of $r < p$ is highest when the temperature is high, and the trade-off between the positions is small, leaving a fair amount of opportunity for the algorithm to escape local minima. The sampling of neighboring continues until a condition —typically an upper limit on sample count— is reached, whereby the temperature is decreased. Next, the algorithm would examine whether an outer stopping criteria has been met. This might be the temperature reaching zero, however, many implementations incorporate a preconfigured number of temperature-resets to facilitate better avoidance of local minima. In this latter case, the outer stopping criterion would be the number of temperature-resets so far. In either case, if the condition is fulfilled, the best solution candidate $\boldsymbol{z}_1$ so far is returned, and the process halts. Otherwise a new random candidate is chosen from the neighborhood structure, and the loop continues.

3.4.5 The Nelder-Mead Simplex Algorithm

The simplex algorithm proposed by [48] is a derivative-free optimization heuristic, just like pattern search introduced in Section 3.4.3. However, instead of a fixed pattern around a base point, it considers a simplex (which is the simplest polytope in any dimension: a line in 1D, a triangle in 2D, a tetrahedron in 3D, etc.).

Initially, the points are set down either randomly, or according to some problem-specific consideration. Then, the algorithm goes through (some of the) following stages, assuming $n - 1$ design variables and the goal function $f(\boldsymbol{x})$

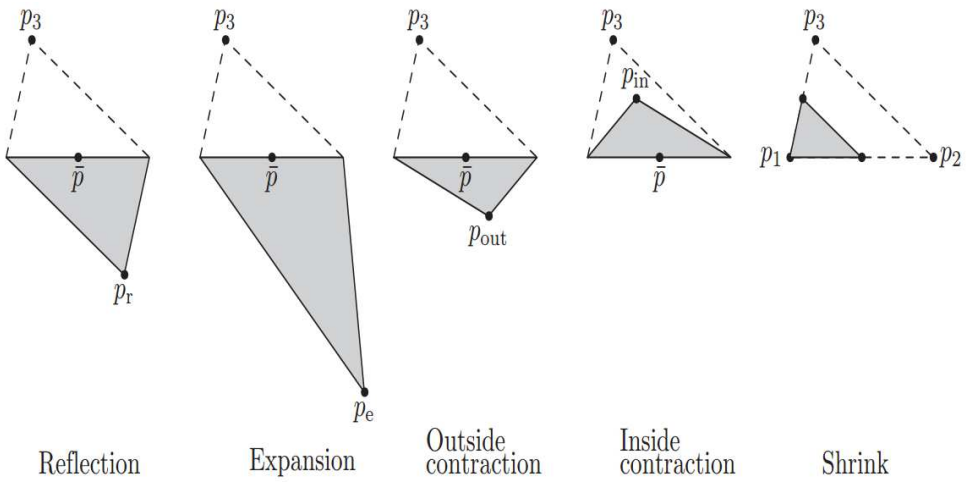


Figure 3.12: Stages of the The Nelder-Mead simplex algorithm for a bivariate goal function. From [23, Figure 1.].

1. **Sort.** The goal function value is evaluated in all n points of the n-simplex. The point with the best (lowest) value will be further referred to as $\boldsymbol{p}_1$ and the one with the worst (highest) value as $\boldsymbol{p}_n$ ($\boldsymbol{p}_3$ in the bivariate case). Accordingly, the rest of the points are assigned the denominations $\boldsymbol{p}_2 \ldots \boldsymbol{p}_{n-1}$ such that $f(\boldsymbol{p}_2) \leq f(\boldsymbol{p}_3) \leq \cdots \leq f(\boldsymbol{p}_{(n-1)})$. Furthermore, the position of the centroid of the $(n-1)$-polytope specified by points $\boldsymbol{p}_1 \ldots \boldsymbol{p}_{n-1}$ is calculated. In the bivariate case, this would be the midpoint of the $\boldsymbol{p}_1\boldsymbol{p}_2$ line segment, while in the trivariate case, the centroid of the $\boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3$ triangle, and so on. This point will be further referred to as $\bar{\boldsymbol{p}}$.

2. **Reflection.** The worst-performing point $\boldsymbol{p}_n$ gets reflected over the centroid $\bar{\boldsymbol{p}}$. Let us denote the resulting point $\boldsymbol{p}_r$ as in the leftmost picture in Figure 3.12. The goal function is then evaluated at this point, and if it is better than the second worst point, but not better than the best $f(\boldsymbol{p}_1) \leq f(\boldsymbol{p}_r) < f(\boldsymbol{p}_n)$, then the worst point $\boldsymbol{p}_n$ is replaced by $\boldsymbol{p}_r$, and the whole process gets restarted from step 1. with this updated simplex.

3. **Expansion.** If the reflected point is even better than the best point so far $\leq f(\boldsymbol{p}_r) < f(\boldsymbol{p}_1)$, then let us obtain an expanded point $\boldsymbol{p}_e = \bar{\boldsymbol{p}} + \gamma(\boldsymbol{p}_r - \bar{\boldsymbol{p}})$ with the expansion factor $\gamma > 1$ as in the second picture of Figure 3.12 (typically $\gamma = 2$, which would be the same as further reflecting $\bar{\boldsymbol{p}}$ over $\boldsymbol{p}_r$). If the expanded point is even better than the reflected point $f(\boldsymbol{p}_e) < f(\boldsymbol{p}_r)$, then the worst point $\boldsymbol{p}_n$ gets replaced with the expanded point $\boldsymbol{p}_e$. Else, the worst point gets replaced with the original reflected point $\boldsymbol{p}_r$. Anyhow, the whole process gets restarted from step 1. with this updated simplex.

4. **Contraction.** Now, since the algorithm has not found any better points than $\boldsymbol{p}_n$ so far, an intermediate point on the $\bar{\boldsymbol{p}}\boldsymbol{p}_r$ line segment $\boldsymbol{p}_{out} = \bar{\boldsymbol{p}} + \rho(\boldsymbol{p}_r - \bar{\boldsymbol{p}})$ will be inspected, with the contraction factor $0 < \rho < 1$ (typically $\rho = 0.5$). This point is called the outside contraction point. Similarly, the inside contraction point

is determined by $\boldsymbol{p}_{in} = \bar{\boldsymbol{p}} + \rho(\boldsymbol{p}_n - \bar{\boldsymbol{p}})$. Let us designate the better of the two contraction points by $\boldsymbol{p}_c = \operatorname{argmin}_{\boldsymbol{x} \in \{\boldsymbol{p}_{out}, \boldsymbol{p}_{in}\}} f(\boldsymbol{x})$. In case $\boldsymbol{p}_c$ is better than the worst point $\boldsymbol{p}_n$, then the latter gets replaced by the former, and the whole process gets restarted from step 1. with this updated simplex. This step is illustrated by third and fourth pictures in Figure 3.12.

5. **Shrink.** If still no better point than $\boldsymbol{p}_n$ has been found, then the simplex gets shrunk towards the best point $\boldsymbol{p}_n$. That is, every point of the simplex $\boldsymbol{p}_m$ gets replaced by $\boldsymbol{p}'_m = \boldsymbol{p}_1 + \sigma(\boldsymbol{p}_m - \boldsymbol{p}_1)$, with the shrinkage coefficient $0 < \sigma < 1$ (typically $\sigma = 0.5$), as illustrated on the rightmost picture in Figure 3.12. The following step is then carried out on this updated simplex.

6. **Check stop criteria.** In this step, it is checked whether a pre-defined stop criterion (or criteria) of the optimization process has been reached. This can either be some metric of the simplex, some statistical indicator (often the standard deviation of the goal function values in all points of the simplex), or the number of iteration steps so far.

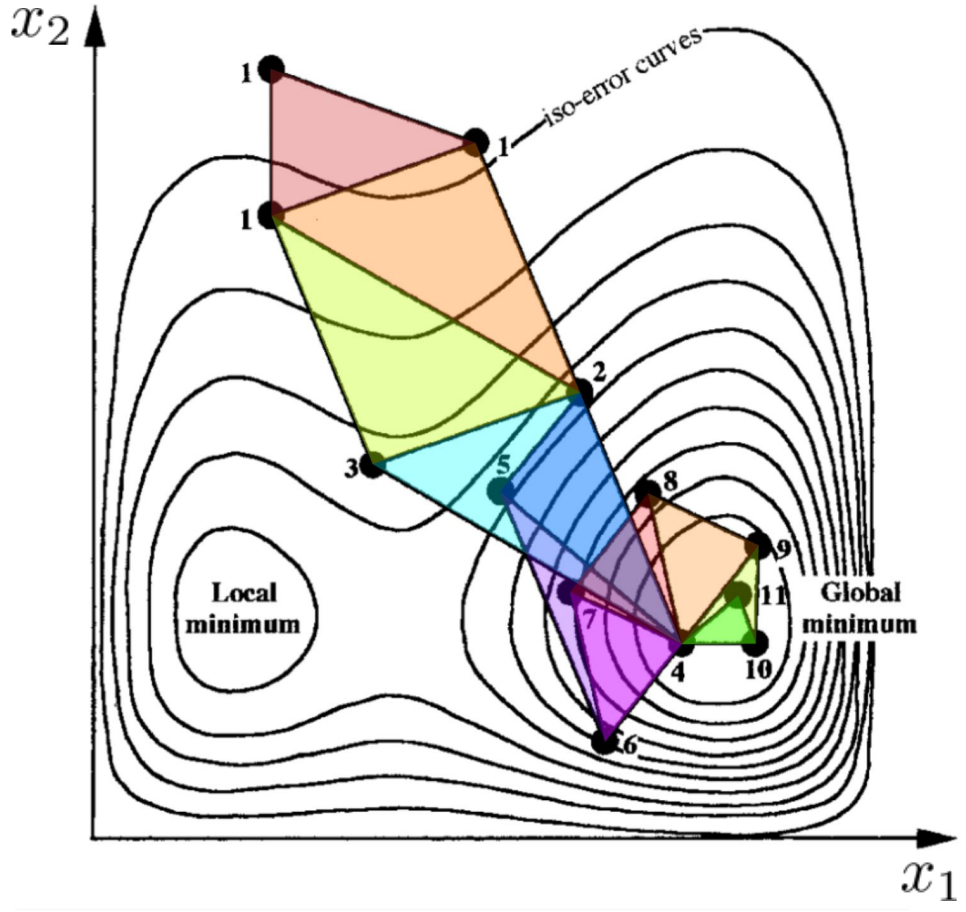


Figure 3.13: Progression of the Nelder-Mead simplex algorithm in the 2D case. The colored simplices were —in order of progression— the result of: initial condition; expansion; reflection; reflection; inside contraction; reflection; inside contraction; reflection; reflection; outside contraction; inside contraction. Notice that the same algorithm could very well iterate towards the local minimum instead of the global one if different initial points were chosen. Adapted from [16, Fig 1.].

3.4.6 Surrogate Optimization

Surrogate Optimization is a class of optimization algorithms whose workings is somewhat different from the general structure shown in Figure 3.4. This is because Surrogate Optimization (also referred to as SBO - Surrogate-Based Optimization) considers an objective function, the evaluation of which is computationally challenging, and attempts to construct a surrogate that is easier to evaluate and hence easier to optimize. For this to work however, the surrogate function must be a good approximation of the objective function. Since it is hard to find a good enough model at once, an outer iterative loop for refining the surrogate model is made use of. This results in the general optimization problem (3.1) being reframed. As [35, eq. 4.1] formulates:

$$\boldsymbol{z}^{(i+1)} = \operatorname*{argmin}_{\boldsymbol{z}} U\left(\boldsymbol{f}_s^{(i)}(\boldsymbol{z})\right) \tag{3.23}$$

where $\boldsymbol{z}^{(i)}$, $i = 0, 1, \ldots$ is a sequence of approximate solutions to the original problem in (3.1), $\boldsymbol{f}_s^{(i)}(\cdot)$ is the surrogate model in the ith iteration, and $U(\cdot)$ is a real-valued utility function that aggregates elements of $\boldsymbol{f}_s^{(i)}(\cdot)$ into a scalar, would the latter be vector-valued [36].

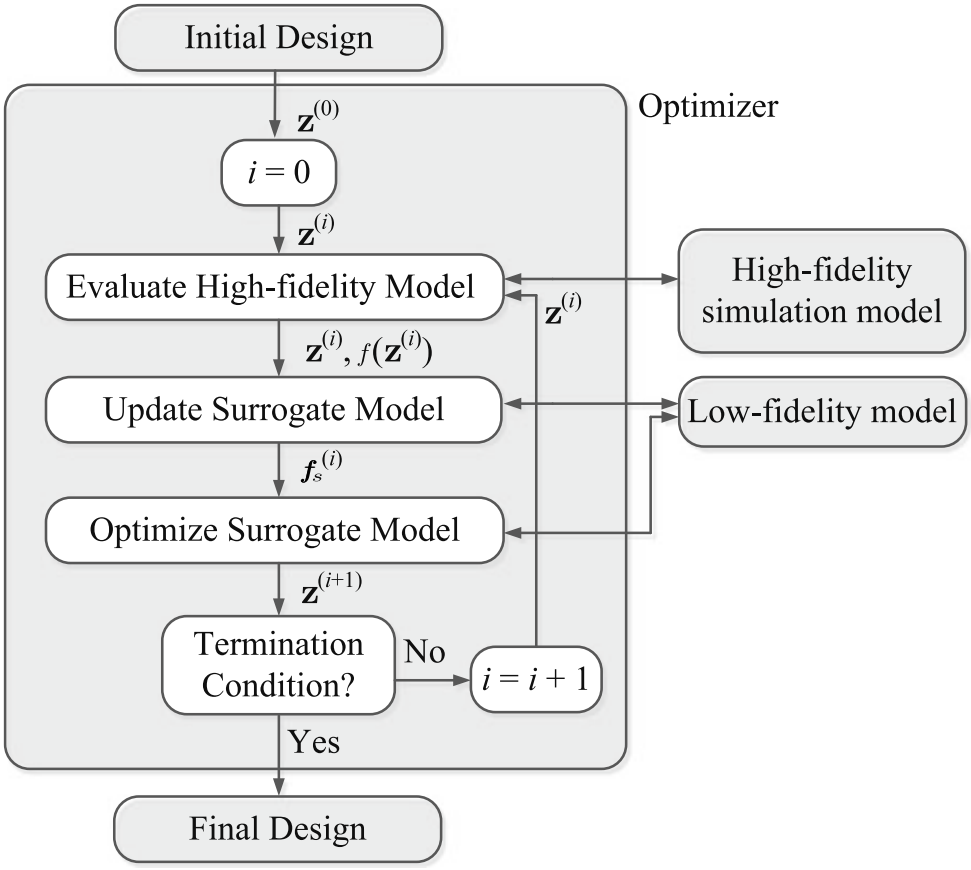


Figure 3.14: Conceptual process of the surrogate-based optimization. From [35, Fig. 4.1].

Figure 3.14 shows the general flowchart of surrogate-based optimization. This involves sampling the high fidelity, computationally hard objective function at the current design points in each outer iteration, and then creating or updating a surrogate model based on the sample points (also referred to as training points) collected so far. The theoretical construction process of a surrogate model is shown in Figure 3.15. Said process generally implies model identification followed by the validation of the identified surrogate, possibly involving further iterations, would the training points collected so far not grant sufficient accuracy. What model identification method is exactly utilized is not predetermined by surrogate optimization per se. Common realizations tend to utilize radial basis function interpolation (for its good performance), although polynomial regression, Gaussian process regression (Kriging), and neural network based regression -among others- are also possibilities [35, Sec. 4.2.3]. Physics-based surrogate construction is a further option with some optimization problems, in which instead of pure mathematical regression, a coarser, less accurate, but also simpler objective function is chosen as the surrogate model. Hereby the low-fidelity model is either derived from

the high-fidelity one through neglection of parameters, or a less accurate, but computationally more advantageous representation of physical reality is knowingly chosen.

Having obtained a surrogate model for the ith iteration $\boldsymbol{f}_s^{(i)}$, it can be handed over to a conventional optimization algorithm [41], such as those discussed so far in previous subsections.

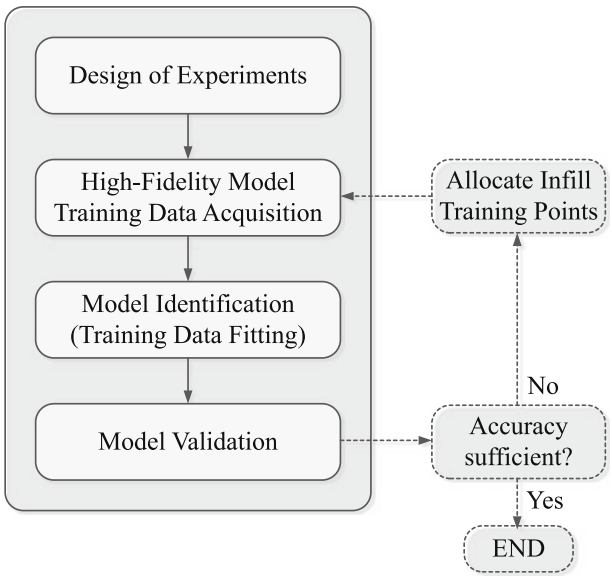


Figure 3.15: General surrogate model construction flowchart. From [35, Fig. 4.2].

Once a good enough (as determined by a termination condition of the inner optimization algorithm inside "Optimize Surrogate Model" in figure Figure 3.14) approximation of a minimum of the surrogate function is found, the high-fidelity function is once again evaluated in the corresponding design points $\boldsymbol{z}^{(i+1)}$. Then, based on the termination condition of the outer loop either the optimization process comes to a halt, or a new round of iteration takes place, incorporating these new point-value pairs into building a more refined surrogate model. In some implementations, further training points are gathered around those returned by surrogate optimization, often by sampling a Gaussian distribution around them.

4 Optimization Results

A practical methodology towards finding valid MRPI sets would be adjusting the optimization variable vector $\boldsymbol{s}_{sol}$, while observing the output of the goal function $f_s(\boldsymbol{s}_{sol})$. Other than pure heuristics, such a black-box approach might prove effective because of the conjecture on barrier phases proposed in [46, S. 3.4], where it was hypothesized that an MRPI barrier candidate's stage tends to transition towards a preferred direction as a single angular constraint boundary is increased or decreased in the oscillatory model. Aforementioned approach can be implemented in practice by relying on the obejctive function prototypes outlined in Section 3.2 supplied to nonlinear numerical optimization algorithms, such as those in MATLAB's Optimization Toolbox [63] or Global Optimization Toolbox [62].

4.1 Implementation of the Optimization Problem

Having found the points of equilibrium, minimization of the goal function $f_s(\boldsymbol{s}_{sol})$ can take place. In MATLAB, setup and solution of optimization problems generally involve the following steps:

Listing 4.1: Pseudocode of carrying out an optimization task in MATLAB.

```
%1. Defining the objective function.
objectiveFunction = @(s) fs(s, .. );

%2. Setting optimization parameters.
options = optimoptions(@solver, .. )

%3. Executing the optimization.
[s_sol,fs_end] =
        solver(objectiveFunction [,s_init], options);
```

where `fs` is the goal function, `s` is the vector of optimization variables, `s_init` is the optimization vector's ($\boldsymbol{s}$ of (3.6)) initial value (the square brackets denote that it is not always required), `s_sol` is the vector of optimized optimization variables, `solver` is a placeholder for a solver function, and `fs_end` is minimized the end value of `fs` after the optimization, that is, `fs_end = fs(s_sol, .. )`.

First, a suitable objective function is defined, the handle of which is assigned to `objectiveFunction`. Then, a call to `optimset( .. )` is made in which options to the optimization problem are supplied. These include the verbosity of the optimization algorithm, the maximum number of iterations and goal function evaluations, as well as

the termination tolerance of the solution value [63]. Finally, the objective function and the optimization options are supplied to a solver function along with an initialization vector for the optimization variables.

How these steps have been applied to the optimization problem at hand —that is, finding valid MRPI sets— will be further elaborated in the following subsections.

4.2 Setting Optimization Parameters

What optimization parameters are available for adjustment depends on the optimization algorithm chosen, thus it is worth considering the type of optimization algorithms before anything else.

Because neither of the approaches discussed in Section 3.2.2 and Section 3.2.3 fulfill Section 3.2 Point 7, and thus not result in a continuous objective function, it is consequent that these objective function candidates are not smooth either.

Smoothness and differentiability is an important property of objective functions, because a number of optimization algorithms rely on calculating its gradient or Hessian matrix [7]. However, algorithms applicable to nonsmooth objective functions exist as well on which we might better rely.

MATLAB's Optimization Toolbox [63, pp. 2-4] provides us with two options —solvers— for optimization of nonsmooth functions: `fminsearch` implementing the simplex algorithm of Section 3.4.5, as well as `fminbnd`. The latter, however, does not work with multidimensional optimization variables, and so is unsuitable for us.

The Global Optimization Toolbox presents more solvers, and [62, pp. 1-34] recommends a particular order in which they worth trying. According to this list

1. `patternsearch` implementing the algorithm outlined in Section 3.4.3,
2. `surrogateopt` implementing the surrogate-function-based simulation of Section 3.4.6,
3. aforementioned `fminsearch` of the Optimization Toolbox,
4. `particleswarm` implementing the particle swarm algorithm of Section 3.4.2,
5. `ga` implementing a genetic algorithm as described in Section 3.4.1, and finally
6. `simulannealbnd` implementing the simulated annealing algorithm of Section 3.4.4

should generally bring the best results in this descending order.

Many optimization parameters are common for all aforementioned solvers, including the display verbosity, the number of maximum iterations (or maximum generations in the case of the genetic algorithm), maximum run time the function-, and step tolerance. Also, all of the above solvers but `fminsearch` can utilize parallel processing on suitable hardware if the corresponding optimization parameter is set.

Furtermore, a number of optimization parameters are unique to one specific solver, such as the initial particle population to `particleswarm`, or the mutation function to `ga`.

As for our purposes, no adjustment to any of these parameters proved to bring significant advantage as compared to the default values, considering that we are solely interested in assessing the viability of utilizing optimization algorithms in finding and optimizing MRPI sets. That being said, parameter `Display` had to be set to `'iter'` in order to obtain detailed information about the number of function evaluations in each iteration on which Section 4.3 relies a lot.

4.3 Executing Optimization

Once optimization parameters have been set, executing the actual optimization involves making a call to the solver, supplying Listing 4.1 with the initial set of optimization parameters (the objective function, the optimization options, and —where applicable— the initial design vector). Unless stated otherwise, we have executed all optimization on the IEEE nine-bus power grid as shown in Figure 4.1 and as defined in the `test_system_3gen.m` MATPOWER case file supplied by [50]. We further relied on [50] for oscillatory model generation, attributes of which have been passed to the objective function.

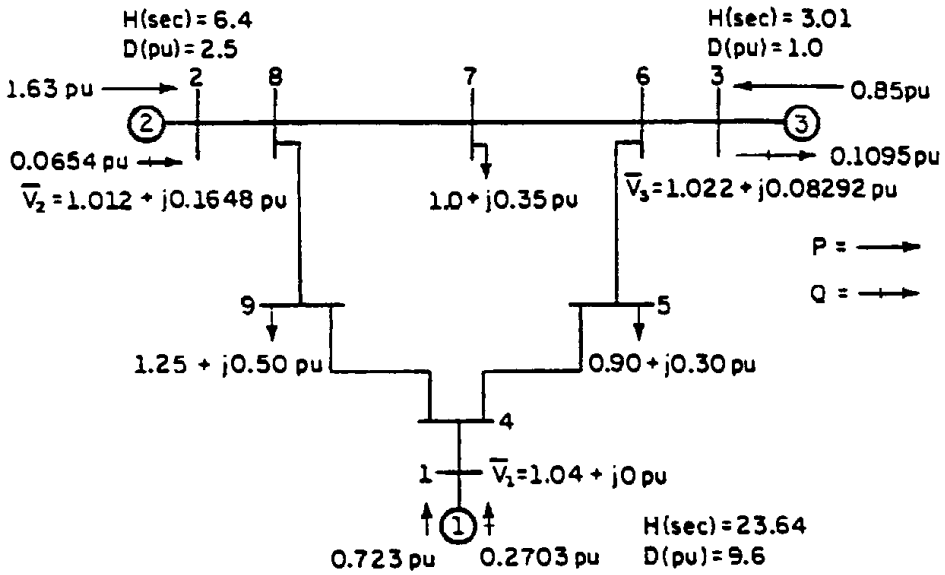


Figure 4.1: The IEEE three-machine, nine-bus test system. During experimentation, machine 1 served as the infinite bus throughout. From [13, Fig. 4.3.1].

For the oscillatory representation, the EN model of Section 2.1.3 has been considered. During these tests, two set of values have been used for the initial optimization vector `s_init`, these being

$$[1,36 \quad 0,46 \quad 1,30 \quad 0,83] \tag{4.1}$$

and

$$[1,00 \quad 0,46 \quad 1,30 \quad 1,00] \tag{4.2}$$

The two different initial design vectors are introduced to demonstrate that solvers relying on it, `patternsearch`, `surrogateopt`, `fminsearch` and `simulannealbnd` may bring significantly different results for different start conditions. The other two solvers, `particleswarm` and `ga` —instead of a vector of initial optimization variables— only take the number of optimization variables as a parameter.

First, the set area method of Section 3.2.2 has been supplied as the objective function to all four solvers mentioned in Section 4.2. The results are summarized in Figure 4.2 and Table 4.1.

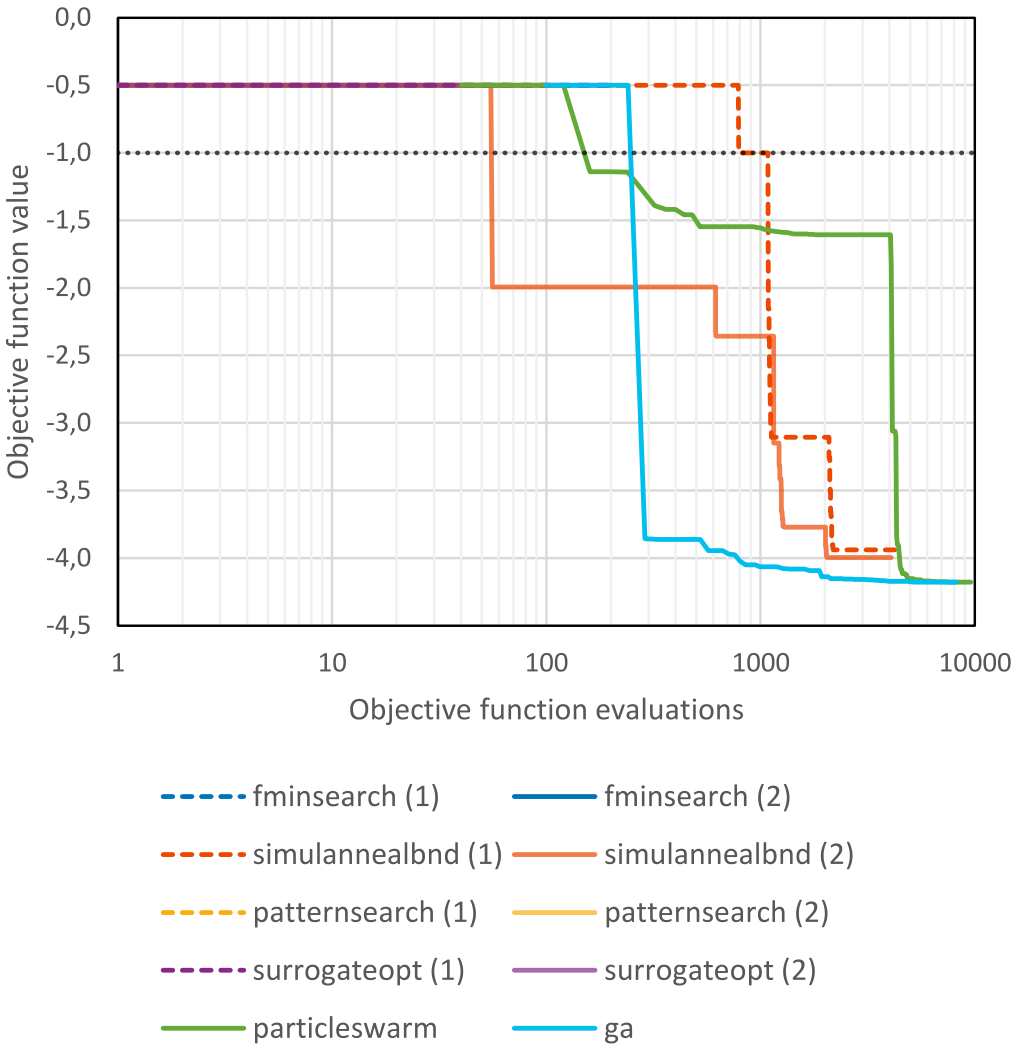


Figure 4.2: Optimization performance of different solvers with the set area objective function, the EN model, and the 9-bus-system model supplied by [50].

Solver	s_init	fd_end	#f (gl.)	#f	Exit event	A_2	A_3
fminsearch	(4.1)	-0,50	-	65	TolX&TolFun	-	64,3
fminsearch	(4.2)	-0,50	-	65	TolX&TolFun	-	80,9
simulannealbnd	(4.1)	-3,94	789	4232	Fun.Tol.	27,3	58,8
simulannealbnd	(4.2)	-4,00	56	4083	Fun.Tol.	28,3	56,2
patternsearch	(4.1)	-0,50	-	99	MeshTol.	-	23,4
patternsearch	(4.2)	-0,50	-	99	MeshTol.	-	30,0
surrogateopt	(4.1)	-0,50	-	200	MaxFun.Eval.	-	29,6
surrogateopt	(4.2)	-0,50	-	200	MaxFun.Eval.	-	29,6
particleswarm	-	-4,18	160	9600	Fun.Tol.	27,0	61,4
ga	-	-4,18	288	8184	Fun.Tol.	27,0	61,4

Table 4.1: Overview of start-, and end parameters of the optimization sessions from Figure 4.2; **s_init** refers to the initialization value of the optimization vector, **fd_end** to the goal function value at the end of the optimization, **#f (gl.)** to the number of objective function evaluations when global validity was first reached (i. e.when a goal function value of -1 was first encountered), **#f** to the total number of objective function evaluation count at the end of the optimization, whereas A_2 and A_3 the MRPI set area of the second and third —non-reference— machines after the optimization. The red and green color of the rows represent whether the optimization session was successful in reaching global validity or not respectively.

Indeed, as described in Section 3.2.2, the set area method does not prove to be particularly successful at finding valid MRPI sets with most solvers. All solvers —except for simulannealbnd, particleswarm and ga— resulted in a -0,5 final objective function value indicating that a valid MRPI set has been found for only one of the two non-reference machines.

Results of the successful optimization sessions were very similar to each other as in MRPI area. In fact, particleswarm and ga came up with the exact same results. The average MRPI size for all successful optimization sessions were around 43, with the average of the maximal MRPI area sizes just below 60.

Next, the span method of Section 3.2.3 has been tried with the same solvers and start conditions as before. The results can be seen in Figure 4.3 and Table 4.2.

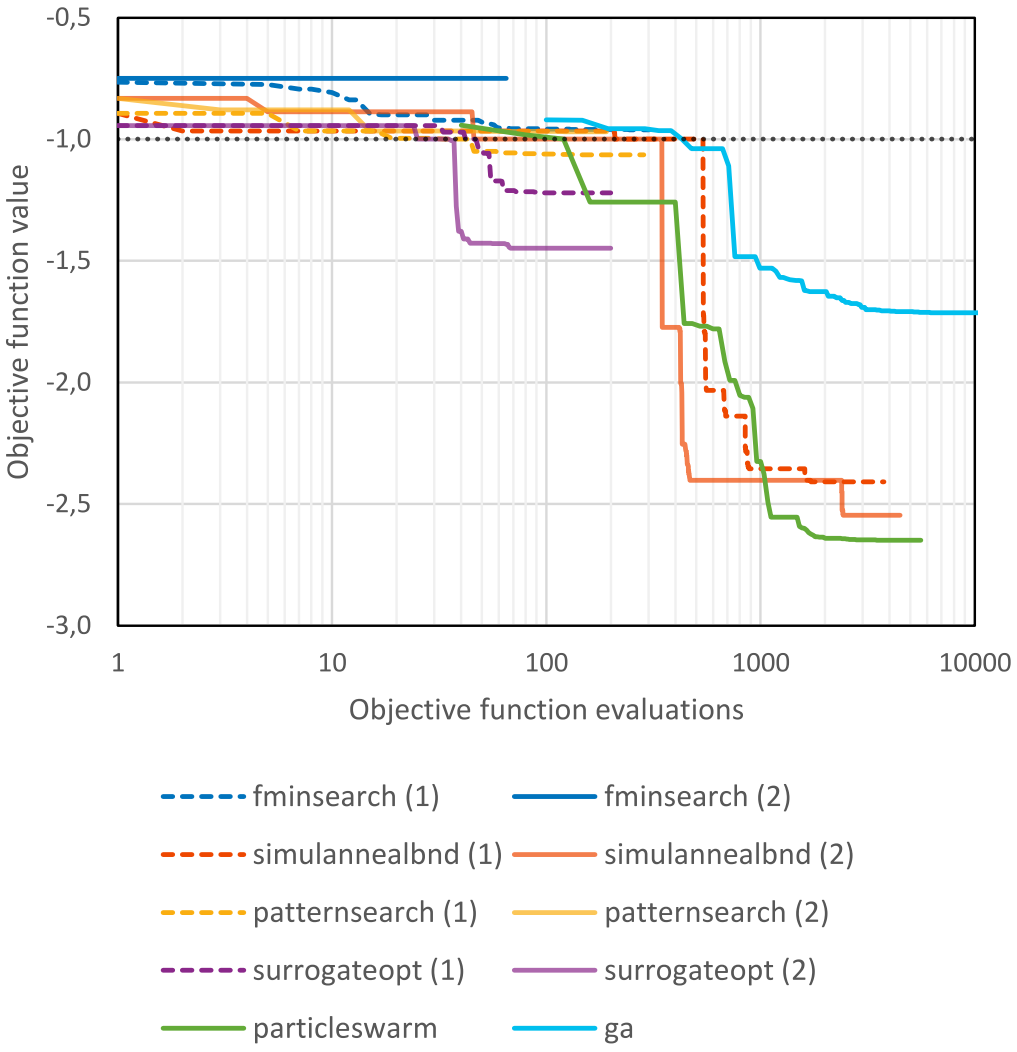


Figure 4.3: Optimization performance of different solvers with the span objective function, the EN model, and the 9-bus-system model supplied by [50].

Solver	s_init	fd_end	#f (gl.)	#f	Exit event	A_2	A_3
fminsearch	(4.1)	-0,96	-	302	TolX&TolFun	54,1	-
fminsearch	(4.2)	-0,75	-	65	TolX&TolFun	80,9	-
simulannealbnd	(4.1)	-2,41	209	3769	Fun.Tol.	17,7	64,8
simulannealbnd	(4.2)	-2,55	46	4486	Fun.Tol.	32,5	38,7
patternsearch	(4.1)	-1,06	33	285	MeshTol.	13,2	20,8
patternsearch	(4.2)	-0,97	-	191	MeshTol.	46,1	-
surrogateopt	(4.1)	-1,22	42	200	MaxFun.Eval.	32,7	42,6
surrogateopt	(4.2)	-1,45	25	200	MaxFun.Eval.	33,6	39,0
particleswarm	-	-2,65	120	5600	Fun.Tol.	28,7	54,6
ga	-	-1,71	429	10346	Fun.Tol.	13,3	20,9

Table 4.2: Overview of start-, and end parameters of the optimization sessions from Figure 4.3; **s_init** refers to the initialization value of the optimization vector, **fd_end** to the goal function value at the end of the optimization, **#f (gl.)** to the number of objective function evaluations when global validity was first reached (i. e.when a goal function value of -1 was first encountered), **#f** to the total number of objective function evaluation count at the end of the optimization, whereas A_2 and A_3 the MRPI set area of the second and third —non-reference— machines after the optimization. The red and green color of the rows represent whether the optimization session was successful in reaching global validity or not respectively.

With the span method, results are in all cases better than with the set area method; in 7 out of the 10 total scenarios, global validity has been reached. The failing ones are `fminsearch`, and `patternsearch` with start condition (4.2).

From a performance aspect, most successful optimization sessions reached global validity in under 60 objective function evaluations. Exceptions were the 209 evaluations of `simulannealbnd` with start condition (4.2), and the 429 evaluations of `ga` it took to reach global validity.

As for the goodness of the results, successful optimization scenarios brought similar results with an average MRPI area of around about 32,4 and an average maximal MRPI area of 40,2. Outstanding values were brought by the —in this regard— worst performing `patternsearch` and `ga`, as well as `particleswarm` and `simulannealbnd` with start condition (4.1), both of which came up with a high MRPI area for machine 3, but at the price of a lower value for machine 2. Still, these latter two brought the highest average MRPI areas, as well as the highest maximal MRPI area values, even though also the highest variance among the machines MRPI areas.

Another important concern is influence of the start conditions on the end result. That is, `fminsearch` failed regardless of the initial states, whereas with `patternsearch`, one of the start conditions resulted in global validity, with initialization vector (4.2) the solver did not cross —although came very close to— the -1 mark — just like `fminsearch` with(4.1).

Ultimately, the span method is —to some extent— capable of optimizing the MRPI set area (living with the assumption that generally the greater span a type A MRPI set has, the bigger area it encloses). However, utilizing the span method for optimization is quite problematic in practice, as described in Section 3.2.3. Because a whole range of valid type A MRPI sets evolve without ever crossing the $\omega = 0$ axis, the span is simply undefined for these barriers.

On the other hand, because the set area method does not consider such an assumption but optimizes directly for the MRPI set area, we would generally expect better results from the set area method than the span method, given that global validity has already been reached. However, when comparing end results of set area optimization scenarios with those of the span method, we find —somewhat unexpectedly— that the span method's results either come really close to, or even outperform those of the set area method.

4.3.1 Results Overview

Having consolidated area data for each successful optimization session, now it is possible to see how the sessions' performances compare to each other based on average MRPI area at the end of the optimization and the number of objective function evaluations needed, as it has been done in Figure 4.4.
In it, the following patterns emerge:

- Most sessions' results cluster between 4000 and 10000 objective function evaluations, with average MRPI sizes between 35 and 45.
- However, generally sessions of this cluster delivered the most varying results.
- All results outside of the cluster were produced by utilizing the span method.
- Among these, `surrogateopt` produced arguably the best results, with an average MRPI area size comparable to those in the cluster, but being an order of magnitude faster.
- The sole successful run of `patternsearch` was second to `surrogateopt` in speed, but was among the worst performers.
- The absolute worst results were delivered by the genetic algorithm `ga` when applied to the span method, being one of the slowest sessions, while having the worst result as measured in average MRPI area.

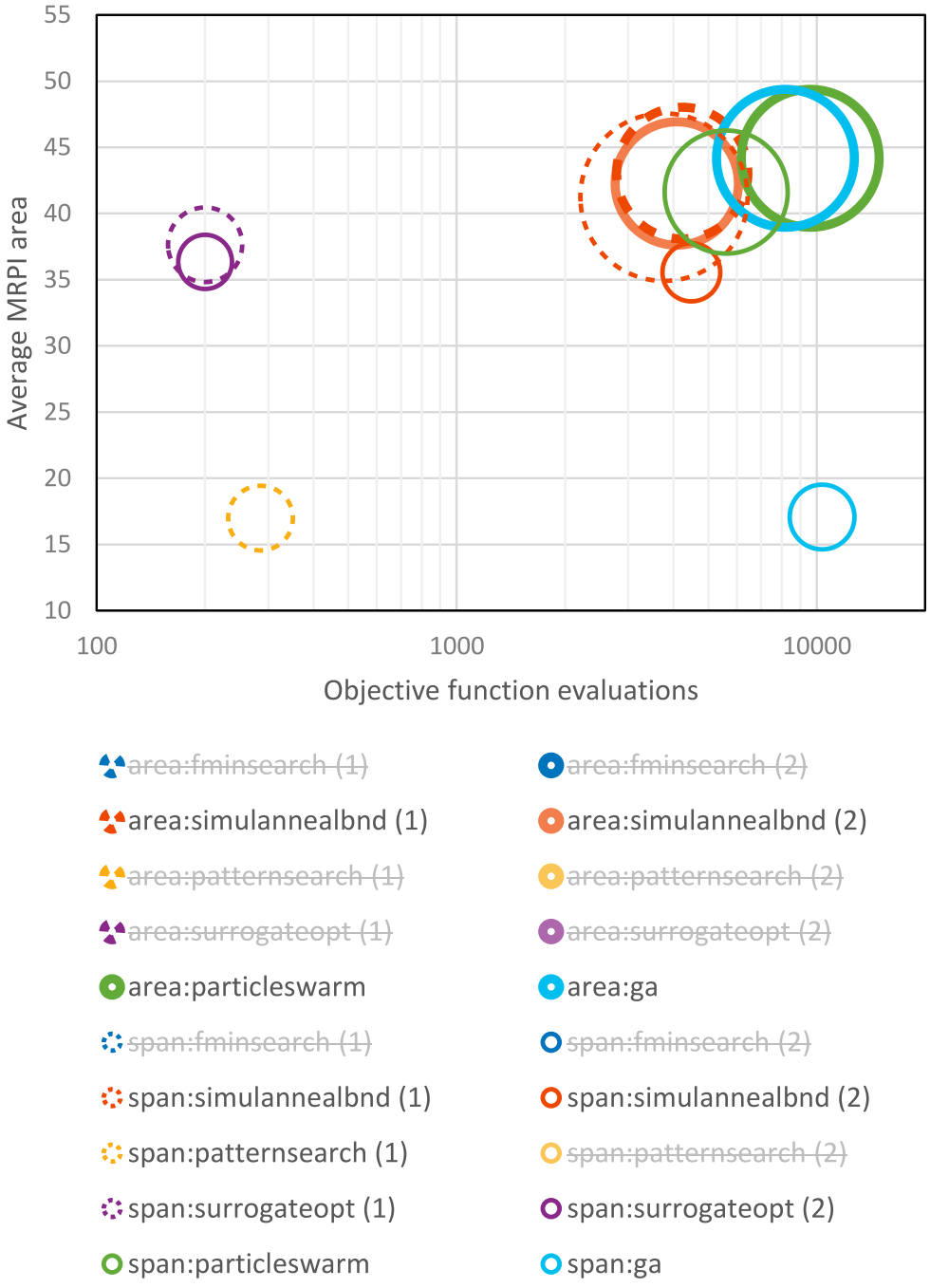


Figure 4.4: Comparison of successful optimization sections based on the objective function evaluations needed and the final average MRPI set area. The area enclosed by the circles represent the variance in MRPI set areas among non-reference machines.

4.4 Applicability to More Complex Systems

In Section 4.3 we have examined the applicability of optimization methods outlined in Section 3.2.2 and Section 3.2.3 as well as optimization algorithms overviewed in Section 3.4 in finding and size-optimizing sets of angular constraints granting global MRPI set validity. Nevertheless, in our experiments we only ever relied on the effective network (EN) model of Section 2.1.3, and the IEEE nine bus test system of Figure 4.1. Although such a small-scale system is fairly easy to comprehend (consider its oscillatory graph in Figure 5.11(a)), having encountered several inadequate optimization algorithms leaves us with the question whether systems of greater extent can at all be considered plausible subjects for optimization of these sorts.

A bottleneck worthy of attention is the increase in computational complexity as the number of generators in the system increase. We find that because the EN oscillatory model constitutes a full graph, the number of edges E increases with the square of the generator count N_g, specifically $E(N_g) = 0{,}5{N_g}^2 - 0{,}5N_g$ as also marked in Table 2.1. Consider the example brought along by [47] seen in Figure 4.5, and how 822 links in the network representation (see Figure 2.4) resulted in 14365 edges (and a comparatively slight decrease in node count from 678 to 170) in the EN oscillatory model. The issue with having a high number of edges in the oscillatory model is that $K_{ij} \cdot \sin\left(\delta_i - \delta_j - \gamma_{ij}\right)$ has to be calculated twice for each edge (once for each node it connects) in (2.2), with $\sin(\cdot)$ being not among the computationally cheapest mathematical operations. In [9, Theorem 7.2] the computation of $\sin(\cdot)$ to the nth digit is shown not to be worse than $\mathcal{O}(M(n) \log n)$, where $M(n)$ represents the number of operations it takes to multiply two integers in the range $[0, 2^{|n|})$. In in a small test —involving computing the sine of a million random numbers, as well as the product of a million pairs of random values with 16 digits of precision, each selected from the range of $(0, 1)$— we saw a 2,5-fold increase in computation time of a sine evaluation as compared with the multiplication. All in all, less sine evaluations, and consequently, less edges in the oscillatory graph is more desirable.

Taking a look at Table 2.1, we may conclude that the SM model (see Section 2.1.5) would be a step in the wrong direction as that would only increase the number of nodes in the oscillatory graph by the number of physical loads, applying which to the example in Figure 4.5 would result in —the SM model constituting a full graph much like the EN model— $0{,}5 \cdot 678 \cdot (678 - 1) = 14365$ in 229503 edges - an almost 16-fold increase from the EN model.

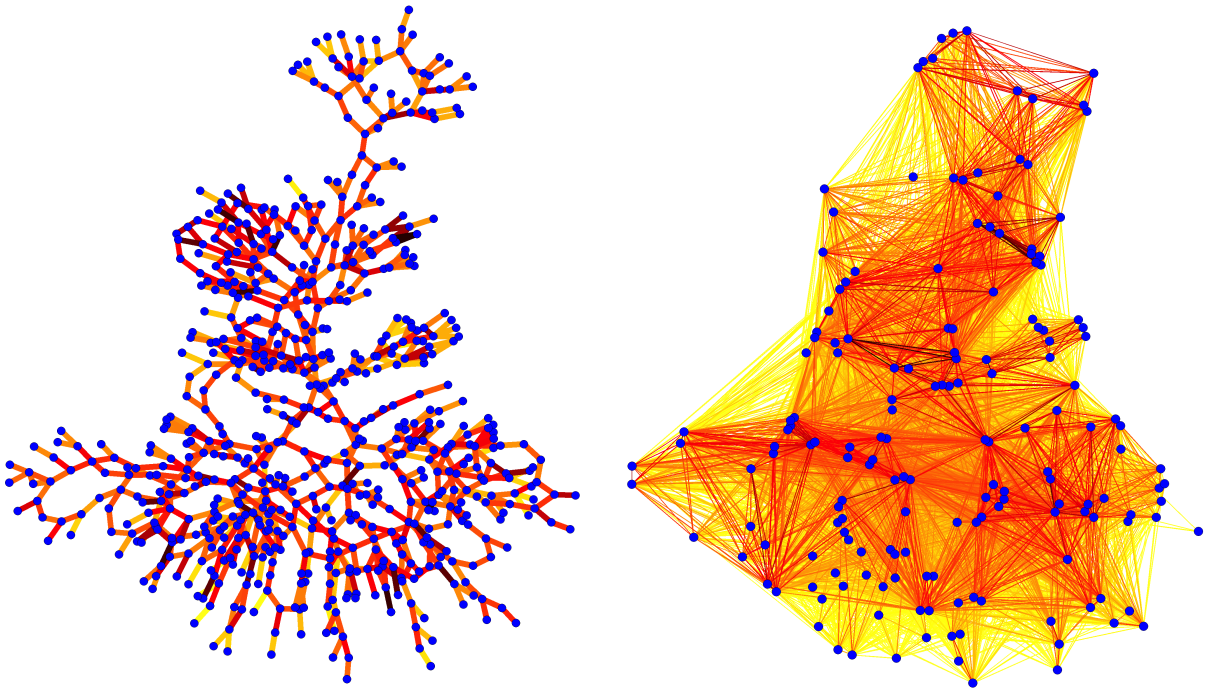


Figure 4.5: Power grid of Northern Italy. (a) The physical network of transmission lines, with 678 nodes and 822 links. (b) The EN oscillatory model with all power elements that are not a generator omitted, leaving 170 nodes. Note that for visibility reasons, only half of all $0{,}5 \cdot 170 \cdot (170 - 1) = 14365$ edges are shown, specifically those with the highest dynamic coupling coefficients. From [47, Figure 1].

For the structure preserving (SP) model (see Section 2.1.5) one cannot determine the node and edge counts in the oscillatory graph. This is exactly because of the persistence of the structure of the examined power grid's network representation. Nonetheless, one might assume —based on real-world examples such as the one in Figure 4.5— that power grids tend to be *rather loosely* connected, with node degrees seldom exceeding four. Still, it would be desirable to be able to compare the different oscillatory models regarding computational complexity.

Hence, we conducted another test, in which we loaded case files (supplied with the MATPOWER library by default) of increasing generator counts, measuring the average CPU time it takes to evaluate a single call of the objective function. We decided to rely on measurements of the span method of Section 3.2.3 only because of two reasons. First, in Section 4.3 the span method proved to be generally more reliable than the area method of Section 3.2.2. Second, it is not the optimization performance of these methods that we are interested in at this point, but the growth of computational complexity as the number of generators increases. We assume that the span method is more consistent in its evaluation time throughout an optimization session, because it involves —more or less— the same general steps before and after global validity has been reached: obtaining the MRPI set barrier candidates through numerical backwards integration, and span determination. With the area method on the other hand, the MRPI area of a machine may or may not be calculated based on whether the determined

MRPI barrier candidates constitute a valid MRPI set or not. (Hereby we conducted an improvised test running the `simulannealbnd` solver with starting conditions as in (4.1) with both the area and span methods. We indeed found that the the variance in CPU time required for the area based objective function execution of $\sigma^2 \approx 0{,}00167$ was about 1,57-fold higher than the $\sigma^2 \approx 0{,}00106$ of the span method.) That being said, the results of our measurements are collected in Table 4.3.

Case	N_g	EN model			SP model			
		ΣE	$\overline{\deg}$	$\times t_{CPU}$	N_1	ΣE	$\overline{\deg}$	$\times t_{CPU}$
`9`	3	3	2	1	9	36	3,00	1
`14`	5	10	4	2,33E0	14	69	3,63	1,41E0
`57`	7	21	6	2,45E0	57	234	3,66	3,53E0
`39`	10	45	9	3,86E0	39	161	3,29	3,00E0
`89pegase`	12	66	11	4,94E0	89	537	5,32	4,85E0
`145`	50	1225	49	1,33E1	145	1139	5,84	9,64E0
`300`	69	2346	68	1,35E1	300	1325	3,59	1,85E1
`1354pegase`	260	33670	259	1,17E2	1354	5554	3,44	5,62E1
`2383wp`	327	53301	326	1,13E2	2383	9136	3,37	1,02E2
`9241pegase`	1445	1043290	1444	1,59E3	9241	41990	3,93	1,10E3
`13659pegase`	4092	8370186	4091	1,10E4	13659	63185	3,56	2,69E3

Table 4.3: Computational complexity with two oscillatory models. **Case** refers to the MATPOWER case file loaded. All values in this column are to be prefixed with "`case`", and postfixed with "`.m`'', thus the first value would be `case9.m`. Furthermore, N_g is the number of generators in the network representation of the loaded power grid case. For the compared models EN and SP, this happens to be equal to the number of second order oscillators in the oscillatory representation as in Table 2.1, and for each model ΣE is the number of edges in the oscillatory graph, $\overline{\deg}$ is the average number of edges connecting to a node in the oscillatory graph, and $\times t_{CPU}$ is the average CPU time it took to evaluate the span objective function, expressed as a multiple of the average CPU time used for objective function evaluation when `case9.m` was loaded for the given oscillatory model. E.g. when working with the SP model, a single call to the span objective function took —on average— about $2{,}69 \cdot 10^3$ times as much CPU time when working with `case13659pegase.m` than when working with `case9.m`. Furthermore, N_1 represents the number of first-order oscillatory nodes in the oscillatory graph.

In Figure 4.6 we plotted out the objective function's average CPU time increase $\times \boldsymbol{t}_{\mathbf{CPU}}$ as a function of the generator count $\boldsymbol{N}_g$. In it we can observe that the CPU time consumption indeed grows with some square function of the generator count for the EN model, whereas this growth is rather linear with the SP model.

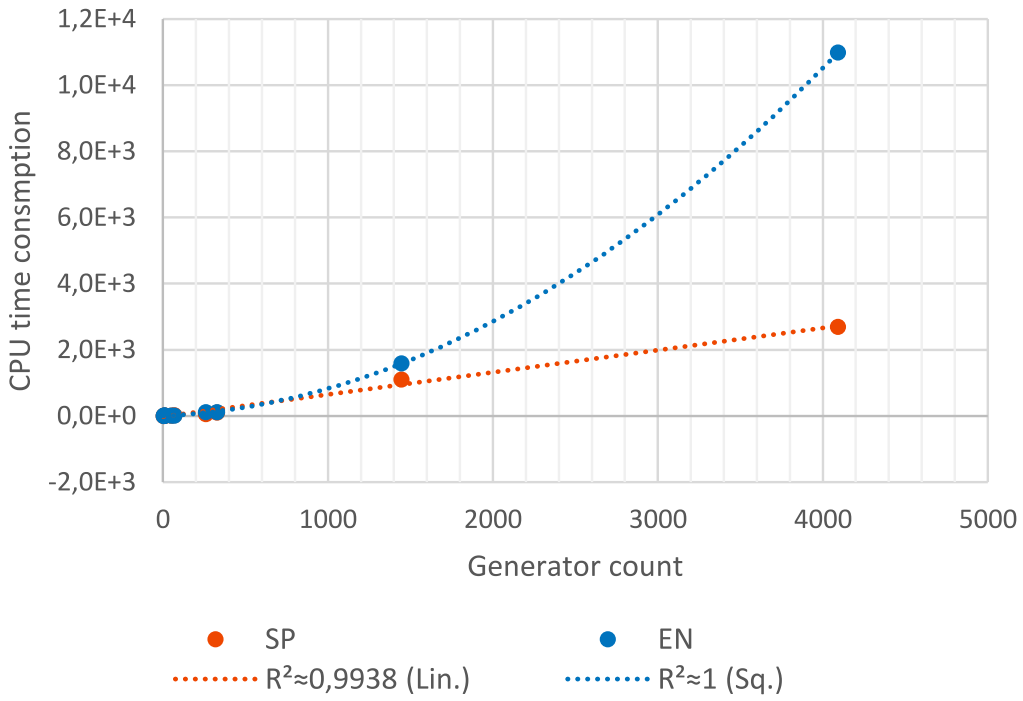


Figure 4.6: CPU time consumption of a single span-method objective function evaluation for the EN and SP models as a function of generator count. The horizontal axis corresponds to $\boldsymbol{N}_g$, whereas the vertical axis to $\times \boldsymbol{t}_{\mathbf{CPU}}$ in Table 4.3.

We find similar results in Figure 4.7, where we have compared the number of edges in the oscillatory model as compared to the number of generators. Here, the SP model shows an *almost* linear interdependence, whereas the EN model's data series can be best fitted with a square function.

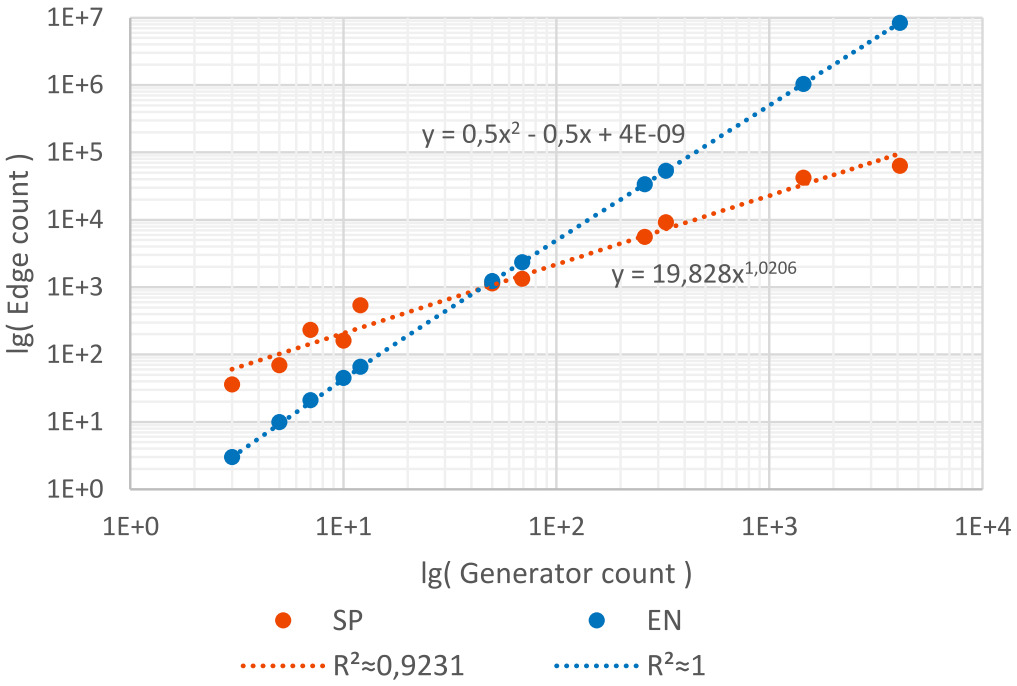


Figure 4.7: Edge count in the SP and EN models as a function of generator count. The horizontal axis corresponds to $\boldsymbol{N}_g$, whereas the vertical axis to $\boldsymbol{\Sigma E}$ in Table 4.3.

In this section we have already presumed that the quadratic growth in computing time with the EN model can —for the most part— be attributed to the ever-growing number of sine evaluations due to the increasing node degrees in the oscillatory model. Data presented in Figure 4.8 appears verifying, in that we observe a linear growth in average node degree when compared with generator count for the EN model, whereas for the SP model, the average node degree stays relatively close to a constant value of around 3,87.

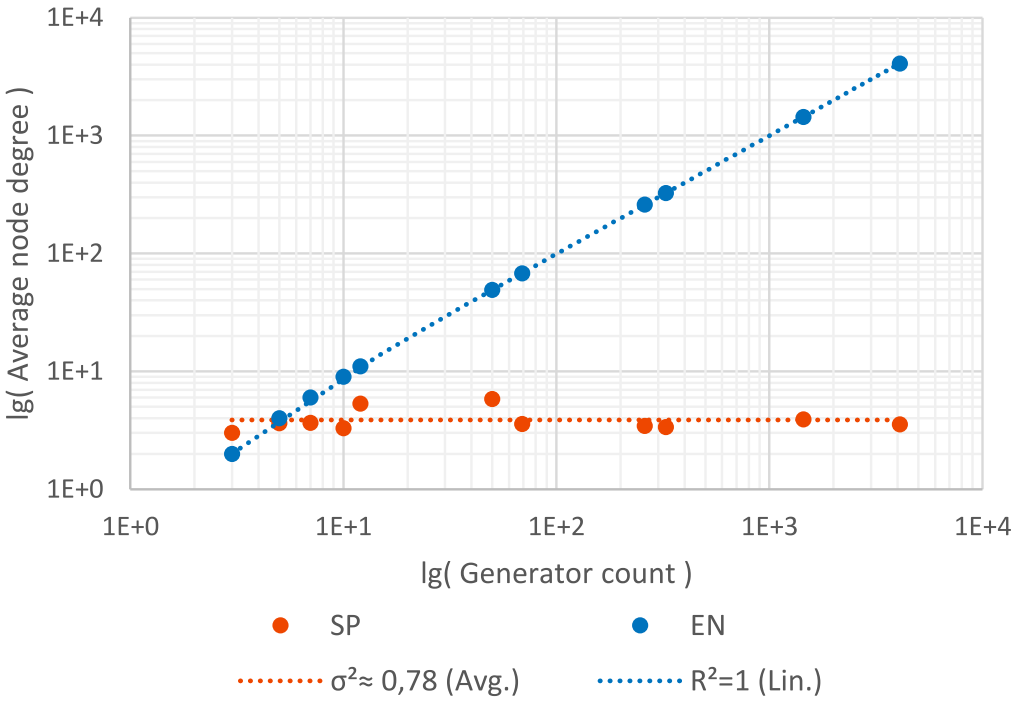


Figure 4.8: Average node degree in the SP and EN models as a function of generator count. The horizontal axis corresponds to $\boldsymbol{N}_g$, whereas the vertical axis to $\overline{\mathbf{deg}}$ in Table 4.3.

Having thoroughly examined the data in Table 4.3 we arrive at the conclusion that the EN model (or the SM model) is generally not suitable for MRPI set search/optimization tasks due to the quadratic complexity curve shown in Figure 4.6. It might however worth calling attention to that it is not the whole optimization process that we pronounce these statements about, but only a single evaluation of the (span method) objective function. However, an increase in power grid complexity most probably introduces further computational complexity through...

- ...an increase in the execution time of the outer optimization algorithms. Although most illustrations in Section 3.4 we have exemplified the algorithms' inner workings in two dimensional state-space, we must not forget that the dimensionality of the design vector of (3.6) is growing linearly with $\boldsymbol{s} \in \mathbb{R}^{(2N_g)}$.
- ...an increase in the number of objective function evaluations. Because of the nonlinearities in (2.23) as well as in the optimization metods of Section 3.2.2 and Section 3.2.3, one would expect that the function evaluation count would rise *fairly* fast with a growing number of generators — even though we are unable to make an exact statement about the rate of growth at this point of research.

Because of the above reasons, we suppose that the EN and SM oscillatory models are —although might perform good in other applications— impractical for the optimization of large-scale, real-world power grids; the structure preserving model, on the other hand, appears to be more promising. Whether the applicability of the latter for power grid optimization remains plausible even with the added computational overhead due to the above listed factors, shall be the subject of further exploratory research.

Applying Optimization in Critical Clearing Time Determination

5

5.1 Power Systems Stability

Power systems management is concerned with a safe and reliable power flow in three aspects as described in [25]:

- Power system protection deals with the safe operation of power system equipment, such as transformers, generators, and transmission lines. [25, Part I]
- Power system control has to do with the (economically) optimal operation of the whole power grid, thus involves topics such as Energy Management, and the unit commitment- and optimal power flow problem. [25, Part III]
- Power system stability is focusing on preserving the integrity of the power grid in that the power system as a whole is able to regain a state of equilibrium after a physical disturbance [25, Part II], and is the part we will be most concerned with in this work.

As described in [37, S. 2.2], it makes practical sense to break the problem of power system stability up into (sometimes overlapping, but more or less distinct) classes and subclasses that are similar in physical nature, the magnitude of the disturbance, in affected devices, processes, or applicable calculation methods. Figure 5.1 serves as a brief overview of this classification.

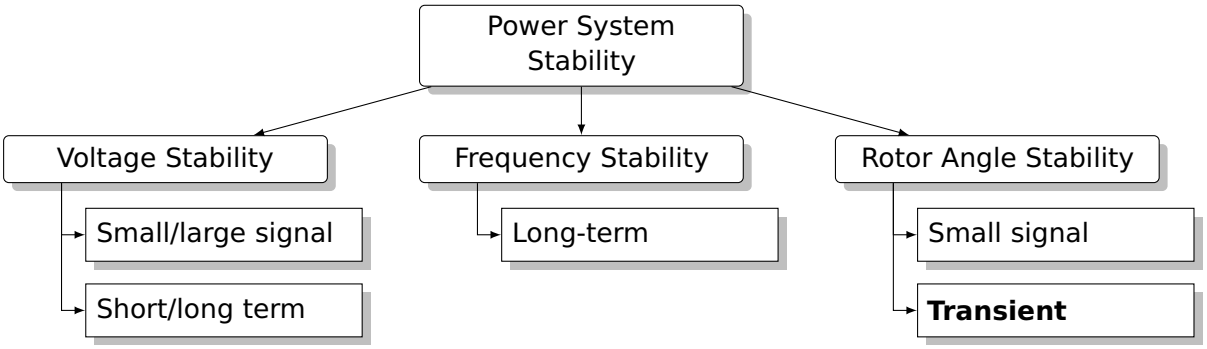


Figure 5.1: A simplified classification of power system stability. Adapted from [37].

- Voltage stability is concerned with being able to maintain an acceptable voltage level on all buses after a disturbance has occurred [37, S. 2.1.2]. Voltage stability might be further divided into overlapping subcategories of small- or large disturbance stability as well as short-, and long term stability [25, S. 8.2.3].

- Frequency stability is concerned with the power system's ability to maintain its steady frequency after a major disturbance, resulting in a significant imbalance between power generation and consumption. [25, S. 8.2.4] Such major disturbances cause significant deviation of voltages, frequencies, and power flows that require adjusting slow processes and controls (such as boiler-, or conduit dynamics), and so are not included in other models for Power system stability. [37, S. 2.1.3] *"Generally, frequency stability problems are associated with inadequacies in equipment responses, poor coordination of control and protection equipment, or insufficient generation reserve."* [25, S. 8.2.4]

- Rotor angle stability is concerned with the ability of a power system of interconnected synchronous machines to maintain synchronism, and has to do with studying electromechanical oscillations present in the system. Rotor angle stability may further be subdivided into two subcategories: small signal stability, and transient stability. The former is concerned with the ability of the power system to maintain phase synchronism under continuous disturbances (such as ever-present changes in power generation and consumption) considered small enough for system equations to be linearized, while with the latter the focus is on whether the power system is able to maintain synchronism following a disturbance so severe, that the pre-, and post disturbence steady state of the power system differ significantly. [37, S. 2.1.1] [25, S. 8.2.2]

Of all power system stability classes, this work will be considered with *transient rotor angle stability* exclusively.

5.1.1 Transient Stability

As described above, transient stability analysis deals with fault scenarios in which the pre-, and post-disturbance steady states differ significantly. Speaking of changes in power system steady state it is worth touching on the subject of Operating States as conceptualized in [17] and further studied in [21], according to which, three sets of equations supervise power system operation:

- A set of differential equations like those in Chapter 1 describing the physical laws governing the **dynamical behavior** of the system components.

- A set of algebraic equations comprising **equality constraints**, that is, the relationship between power generation and consumption.

- Another set of algebraic equations describing **inequality constraints**, representing constraints on system variables that should not be exceeded (e.g. those that we defined in Section 3.1).

Since power system stability is most concerned with whether and how a power system regains a state of equilibrium after a physical disturbance, it is inequality constraints (again, describing constraints on system variables) that transient stability anal-

ysis is most concerned with. However, it is discussed in Section 2.1 that system parameters static to the equilibrium state (such as steady-state power consumption or steady-state voltages) are obtained by optimization of equality constraints, as in solving the optimal power flow problem. These steady-state system parameters are taken as constant for the purposes of transient stability analysis (since only very small time scales are considered), and thus are viewed as static input parameters to dynamical models. These dynamical models are then used to describe the evolution of system variables, and to examine if inequality constraints are violated.

Based on whether equality-, and inequality constraints are met, [21] describes five operating states of power system operation as summarized in Figure 5.2.

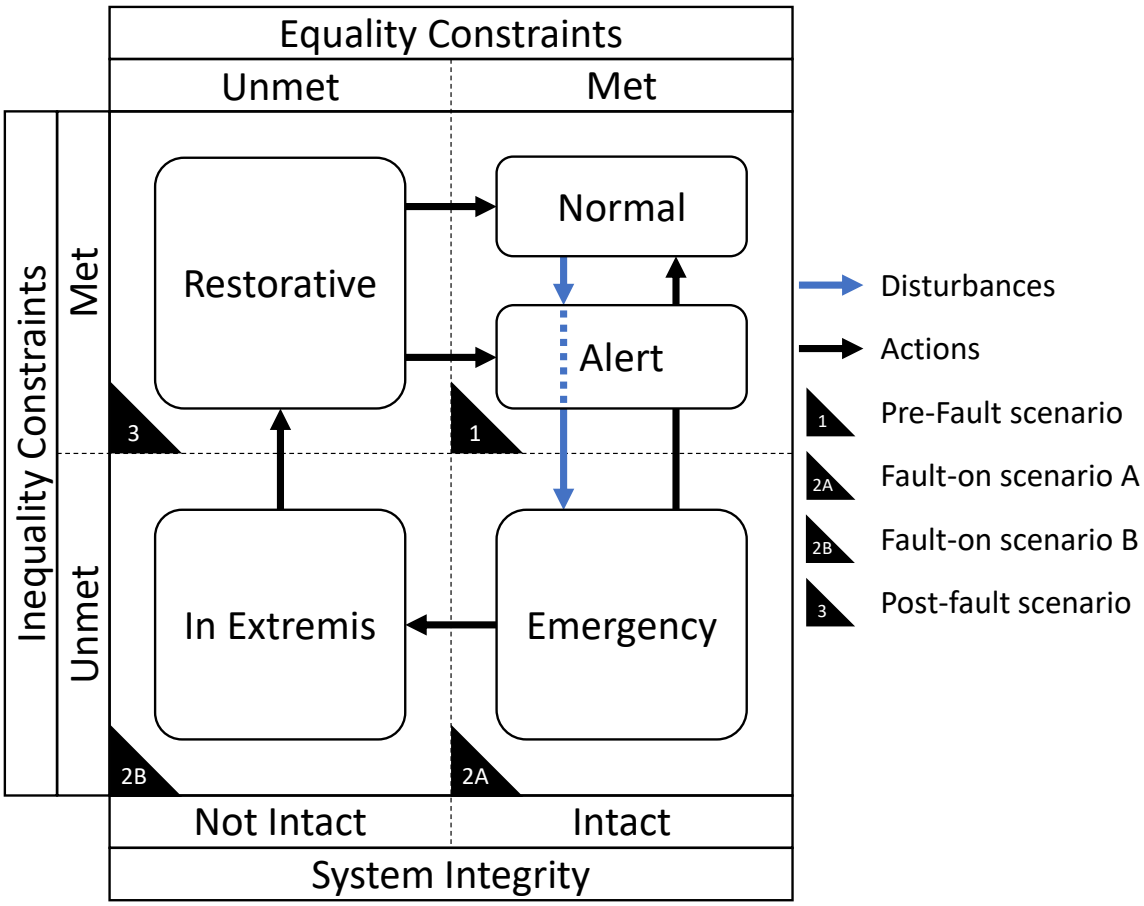


Figure 5.2: Dy-Liacco's diagram extended with possible pre-fault-, fault-on, and post-fault states. Adapted from [21].

During normal operation, all system constraints are met. Once a disturbance causes some system attributes to reach a level of inadequacy, the power system might enter the alert state in which preventive action must be taken to return the system to the normal state of system attributes between adequate thresholds, while constraints are still met all along. However, a large-scale disturbance might cause the power system to violate one or more inequality (voltage, rotor angle, frequency, ...) constraints, and thus enter the emergency state. Once here, it might be possible to —through fast corrective action— bring the system back to a state of met constraints. This might be achievable

due to the possibility to overload an electric component for a short amount of time, that hasn't been exceeded before corrective action was taken. If components have been overloaded more, or for longer than allowed, built-in physical protection might disable components causing the system to lose integrity, resulting in partial-, or total service interruption, entering the in extremis operating state. Once inequality constraints are satisfied, the power system enters the restorative state, where resynchronization and load pickup takes place, returning the system to a steady state in which all conditions are met once again.

With transient stability, the scope of the analysis is whether angular-, and frequency constraints are met in a given steady state of power system operation, and how rotor angles and frequencies evolve over time in case of a state transition. Transient security analysis may be divided into a static and a dynamic part, as summarized in [51, S. 1.3.2]. The static part aims to examine the post-fault equilibrium state of the power system, checking whether it leads to acceptable operating conditions, that is, whether inequality constraints are met. However, a post-fault equilibrium state might not be stable; dynamic transient security assessment considers how the system will reach its post-fault operating conditions. Thus, in the scope of transient security analysis, three scenarios are always considered:

- The *pre-fault-scenario* considers the pre-fault equilibrium point (the pre-fault steady state), which serves as a starting point for transient analysis. As shown in Figure 5.2, the pre-fault-scenario may correspond to the normal-, or alert operating states, in which inequality constraints remain fulfilled.

- The *fault-on-scenario* considers the system dynamics that have —due to the appearance of a fault— changed. The fault-on-scenario may correspond to the Emergency operating state in which some inequality constraints are being violated.

- The *post-fault-scenario* considers the post-fault equilibrium point, and post-fault inequality constraints after the system dynamics once again underwent a change due to fault isolation or clearance. The post-fault-scenario may either correspond to the in extremis operating state in which —although a number of power system protection measures might took place to avert an emergency— inequality constraints remain unmet, or the restorative operating state in which through additional measures the power system has regained inequality constraint compliance.

Mathematically formulated, the system dynamics in above scenarios can be described through a set of differential equations as in [4, S. 2.1]

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}_I(\boldsymbol{x}), \quad t \in \,]-\infty, t_F[\\ \dot{\boldsymbol{x}} &= \boldsymbol{f}_F(\boldsymbol{x}), \quad t \in [\,t_F, t_C[\\ \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}), \quad t \in [\,t_C, \infty[\end{aligned} \tag{5.1}$$

where $\boldsymbol{x}$ is the vector of system state variables, and $f_I(\boldsymbol{x})$ is the initial system dynamics before a fault occurs at time t_F, referred to as the *pre-fault dynamics* of the system. The *fault-on* dynamics of the system is denoted by $f_F(\boldsymbol{x})$, prevalent between the fault

time t_F and the point in time the fault was cleared, that is, the *clearance time*. The system dynamics after fault clearance is described by $f(\boldsymbol{x})$. Transient security analysis' main focus is then to investigate, whether inequality constraints are met in and during transitions between each scenario, as well as to make a statement about the post-fault steady state.

A prototypical use-case for transient stability analysis is critical clearing time (CCT) assessment. Hereby it is assumed that the pre-fault, fault-on, and post-fault dynamics are already assumed as known. Then, one is interested in finding the critical clearing time t_{CC}: the duration up to which the fault dynamics may prevail $(t_C - t_F) < t_{CC}$ so that the post-fault dynamics still brings the system to an equilibrium, and so that no inequality constraints are violated during the whole process. The fault-on scenario may denote a system that is identical to the pre-fault scenario except for the fault, say, a power transmission line being cut, or a short circuit triggering protection equipment. The post-fault dynamics may or may not be identical to the pre-fault one. In the former case, the fault can be considered as resolved, whereas in the latter as detached.

There are multiple methods for computing the critical clearing time. Basically all models discussed in Section 5.1.3 are capable to facilitate CCT calculation. Say, one might rely on the classical model to determine the critical clearing angle using the equal areas criterion, from which the CCT can be obtained if the system frequency is known and taken as constant. Another way would be just running time-domain simulations, while changing the duration of the fault-on dynamics, and examining the results.

5.1.2 The classical model

A synchronous electrical machine's dynamic behavior is well described by the swing equation [57, Eq. 15.20]

$$M\ddot{\delta} = P_s - P_e = P_a \tag{5.2}$$

where M is the angular momentum, δ is the torque angle or angular deviation (angular difference between the rotor as compared to a reference bus), P_s is the rotor (shaft) power, and P_e is the electrical power. The difference of the latter two, $P\alpha$, is called the accelerating power. At this point one might notice that (2.2) was also referenced to as the swing equation in Section 2.1. Indeed, as [47] deduces, (2.2) is actually derived from (5.2) through linearizing the latter around a synchronous equilibrium point and introducing some constants. Said linearization is feasible, since [49]'s main concern is power grid synchronization, and thus considers very short time intervals, and rather small perturbations.

The most rudimentary model for synchronous electrical machines used for transient stability analysis considers a generator as a constant voltage behind a transient reactance for electrical parameters, and a mechanical parameter such as the moment of inertia such as in (5.2) (or the angular momentum, taken as constant for near-constant angular velocities) [25, 9.3.1]. This is known as the classical model of synchronous electrical machines. As we will soon see in (2.2), the coefficient of the angular acceleration

of the machine rotor $\ddot{\delta}$ is often expressed as the function of some inertia constant H and the rated frequency of the machine ω_R. The main correlation governing the classical model is

$$P_e = P_{\max} \sin \delta \tag{5.3}$$

known as the power-angle relationship, where P_e denotes electrical power, $P_{\max}$ is the maximal electrical power, and δ is the torque angle.

5.1.3 Approaches for Stability Assessment

Transient stability analysis is most concerned with the evolution of rotor angles and frequencies of power system components modeled as electrical motors or generators.

Most approaches that are going to be discussed in this subsection build on the classical model of synchronous electrical machines that has been discussed previously in Section 5.1.2.

Equal areas criterion

With the help of (5.2), the equal area criterion can be deducted, which states that

$$\int_{\delta_0}^{\delta_m} P_a \, d\delta = \int_{\delta_0}^{\delta_m} (P_s - P_e) \, d\delta \Rightarrow \int_{\delta_0}^{\delta_m} P_s \, d\delta = \int_{\delta_0}^{\delta_m} P_e \, d\delta \tag{5.4}$$

where P_s is the shaft-, or mechanical power, P_e is the electrical power, δ_0 is the steady-state torque angle before a fault in the system, and δ_m is the maximal torque angle that the machine reaches after the fault. Figure 5.3 shows (5.3) and the implication of (5.4) in a power-angle diagram, the latter known as the equal areas criterion of transient power system stability.

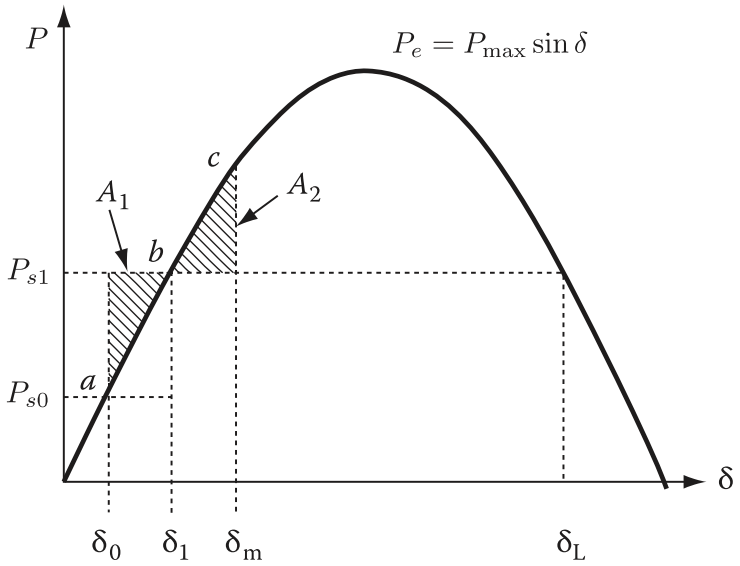


Figure 5.3: Power-angle curve showing a step change in shaft (mechanical) power with areas relevant to (5.4). P_{s0} and P_{s1} are shaft power values from before-, and after the step change. δ_0 and δ_1 are torque angles associated with steady states corresponding to the aforementioned shaft powers. P_e is the electrical power from (5.3). From [25, Figure 9.5].

The criterion states that since the area under the power-angle curve is (assuming synchronicity) proportional to energy consumed, and since for a machine to maintain it's stability, the total (mechanical and electrical) energy generated (or dissipated) during transition from the old equilibrium state δ_0 to the new equilibrium state δ_1 must be equal to the energy dissipated (or generated) during transition from the new equilibrium state δ_1 to the state with the maximum angle deviation δ_m, and so the corresponding areas A_1 and A_2 in the power-angle diagram must also be equal.

This holds intuitively, because assuming that the P_e curve would just be a straight, infinitely long line, say, tangential to the P_e curve in Figure 5.3 at the axis origins, then any step change in mechanical power output would would cause the machine to first reach, then *shoot over* the new equilibrium point, oscillating around δ_1 between δ_0 and δ_m until -due to power losses- the oscillation amplitude would settle more and more, and eventually would die out.

However, since the $P_e(\delta)$ function is not actually an infinite straight line (hence P_{max}: neither a machine, nor a bus can generate or consume power above some finite value), thus it can very well be, that A_2 simply *can not get* as big as A_1.

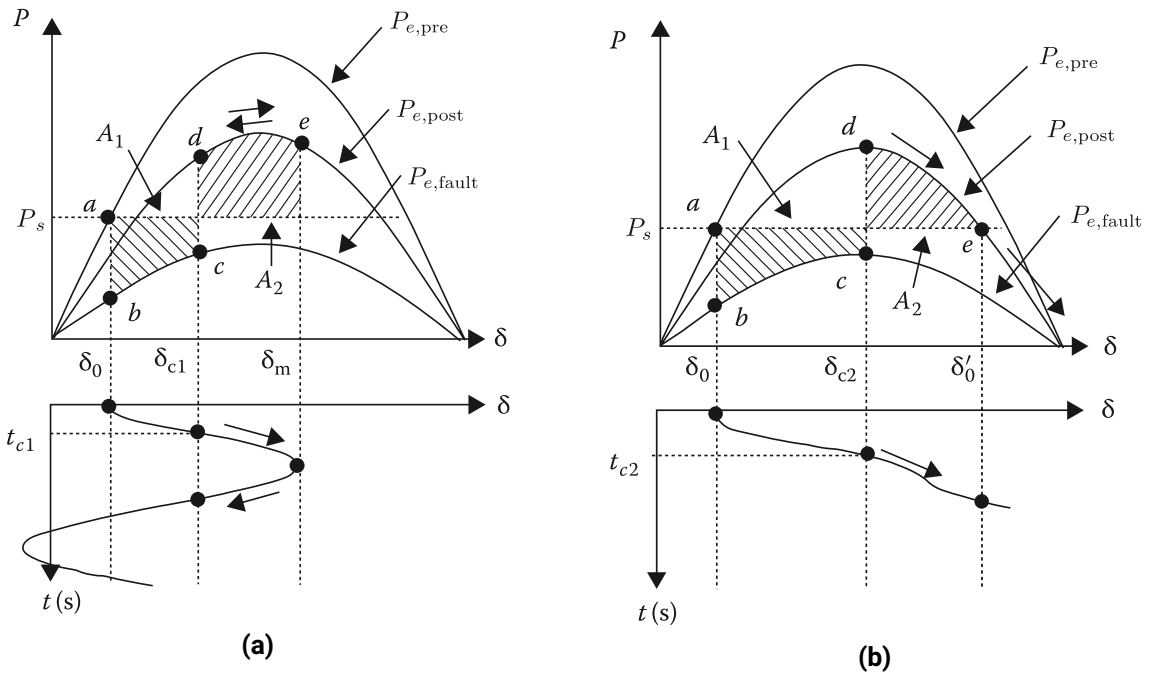


Figure 5.4: Evaluating critical clearing time problems with the help of the power-angle relationship. P_s denotes shaft power, δ_0 the pre-fault steady state torque angle, δ_{c1} the torque angle at the time of fault clearance t_{c1}, and δ_m the maximum torque angle deviance. From [25, Figure 9.6b, Figure 9.7b].

Figure 5.4 shows another use-case for transient stability analysis based on the equal areas criterion, in which power-angle curves corresponding to the fault states (as introduced in Section 5.1.1) are shown. This time it is not the shaft power that is changing from scenario to scenario , but the shape of electrical power-angle curve.

In Figure 5.4(b), the above criterion is unmet, because the fault was cleared too late for the machine to be able to recover, because there is just *not enough area* between points *d* and *e* and curves P_s, and the post-fault P_e for it to match the area proportional to the mechanical energy generated during the fault-on scenario (that is, the area between P_s and P_e during fault and between points *b* and *c*.) In Figure 5.4(a) however, the equal areas criterion is met, and the post-fault system will approach it's new equilibrium power angle where the line of shaft power P_s intersects the curve of post-fault electrical power P_e. Given this latter scenario, we might determine the critical clearing angle δ_c analytically by first considering its physical meaning: it is the rotor's biggest possible angular deviation, above which the machine is no longer able to return to its equilibrium point. Intuitively, this angle must lay right of δ_{c1} in Figure 5.4(a), since we still have *unused area* left right of $A2$. We also know, that it must be let of δ_{c2} in Figure 5.4(b), because the equal areas criterion is not fulfilled in that scenario, and so the machine turns unstable. In other words we do not have *enough* area under the post-fault power-angle curve to counterbalance $A1$. Indeed, the critical clearing angle is the angular deviation at which the two areas are equal $A1 = A2$. Hence

$$\int_{\delta_0}^{\delta_c} \left(P_s - P_{e,\text{fault,max}} \cdot \sin(\delta)\right) d\delta = \int_{\delta_c}^{\delta_0'} \left(P_{e,\text{post,max}} \cdot \sin(\delta) - P_s\right) d\delta \tag{5.5}$$

which yields for the critical clearing angle

$$\delta_c = \arccos\left(\frac{P_s(\delta_0' - \delta_0) - P_{e,\text{fault,max}} \cdot \cos(\delta_0) + P_{e,\text{post,max}} \cdot \cos(\delta_0')}{P_{e,\text{post,max}} - P_{e,\text{fault,max}}}\right) \tag{5.6}$$

as [55, eq. 2.14] states, assuming that the fault-on electrical power is given by $P_{e,\text{fault}}(\delta) = P_{e,\text{fault,max}} \cdot \sin\delta$, whereas the post-fault electrical power by $P_{e,\text{post}}(\delta) = P_{e,\text{post,max}} \cdot \sin\delta$. Furthermore, since in point e in Figure 5.4(b) $P_s = P_{e,\text{post}}$

$$\delta_0' = \pi - \arcsin\left(\frac{P_s}{P_{e,\text{post,max}}}\right) \tag{5.7}$$

Having expressed the critical clearing angle in (5.6) and (5.7), we might derive the critical clearing time by integrating (5.2) [24, Sec. 16.6] utilizing the fact that $\frac{d\delta}{dt} = 0$ when $t = 0$

$$\begin{gathered}\frac{d^2\delta}{dt^2} = \frac{P_a(t)}{M} \rightarrow \frac{d\delta}{dt} = \int_0^t \frac{P_a(t)}{M} dt = \\ \frac{1}{M}\int_0^t \left(P_s - P_{e,\text{post,max}} \cdot \sin(\delta(t))\right) dt\end{gathered} \tag{5.8}$$

now, at this point one cannot help but take issue with integrating $\sin\delta(t)$ with respect to time, as the formal interdependence between power angle and time $\delta(t)$ is not explicitly known (take as examples the time-domain plots in Figure 5.4). Indeed, symbolically expressing the critical clearing time is in most scenarios not possible. In [24, Sec. 16.6] a deduction of the critical clearing time is given for the special case when $P_{e,\text{fault}}(\delta) = 0$. In this special case, the $P_a(t) = P_s$, with which

$$\frac{d\delta}{dt} = \int_0^t \frac{P_s}{M} dt = \frac{P_s}{M} t \rightarrow \int_{\delta_0}^{\delta_c} d\delta = \frac{P_s}{M} \int_0^{t_c} dt \tag{5.9}$$

going forth with the integrations

$$\delta_c - \delta_0 = \frac{P_s}{M} \frac{t_c^2}{2} \tag{5.10}$$

with which we arrive at

$$t_c = \sqrt{2\frac{M}{P_s}(\delta_c - \delta_0)} \tag{5.11}$$

Because we have assumed in (5.9) that in the fault-on scenario all electrical power consumption falls out, while the generated mechanical power P_s remains constant, we

implicitly also made the assumption that all of the generated power can only be utilized to increase the rotor's kinetic energy under constant accelerating power. In other words, if $P_{e,\text{fault}}(\delta(t))$ would be greater than 0 in some time interval, the total accelerating power in that time window would be less than when the fault-on power consumption is zero. Consequently, the critical clearing angle δ_c would be reached in a longer time, that is, the critical clearing time t_c would be higher. Thus, (5.11) constitutes an underestimation for the critical clearing time for the case that we initially proposed. Although this is is a useful metric from an engineering point of view, as for explicit results, we might as well paraphrase [24, Sec. 16.6]: "This example serves to establsh the concept of critical clearing time which is essential to teh design of proper relaying schemes for fault clearing. In more general cases the critical clearing time cannot be explicitly found without solving the swing equations by computer simulation."

Furthermore, the equal areas method is clearly best applicable in a Single Machine Infinite Bus (SMIB) system. In fact, [25, S. 9.2.3] states that although *this method is not generally applicable to multimachine systems, it is a valuable learning aid*.

Other sources, such as [2, S. 2.9] give an example of reducing a two-machine system to a SMIB one, stating that *the same assumptions used for a system of one machine connected to an infinite bus are often assumed valid for a multimachine system*, and that this model is*useful for stability analysis, but is limited to the study of transients for only the "first swing" or for periods on the order of one second*.

In section 4.2 of [43] "Dynamic Equivalent of the Critical Machines", the author proposes that *when a fault occurs in a large power system, only a few machines actively response to the fault and tend to lose synchronism. [...] Therefore it is enough to study the behaviour of the critical machines with respect to the rest of the power system in order to evaluate the transient stability of the system for a specific fault*.

However, these works do not anymore count among the newest ones, and as it will shortly be discussed, new or refined models and methods rendered a number of the classical model's concerns obsolete.

Time-domain simulation

The classical representation of synchronous machines (introduced in Section 2.1) was historically used to reduce the computational burden of more detailed models as in neglecting a number of machine characteristics, such as the effects of damper windings, core saturation, excitation systems, mechanical load dynamics, etc [25, 9.3.1].

However, with today's cheaply accessible computational capacity, neglecting above properties became unnecessary, as even with more complex, more detailed models, computational methods for time-domain simulation that are efficient enough to produce quasi-instantaneous results (on the short time scale that transient stability analysis is concerned about) are pretty much available.

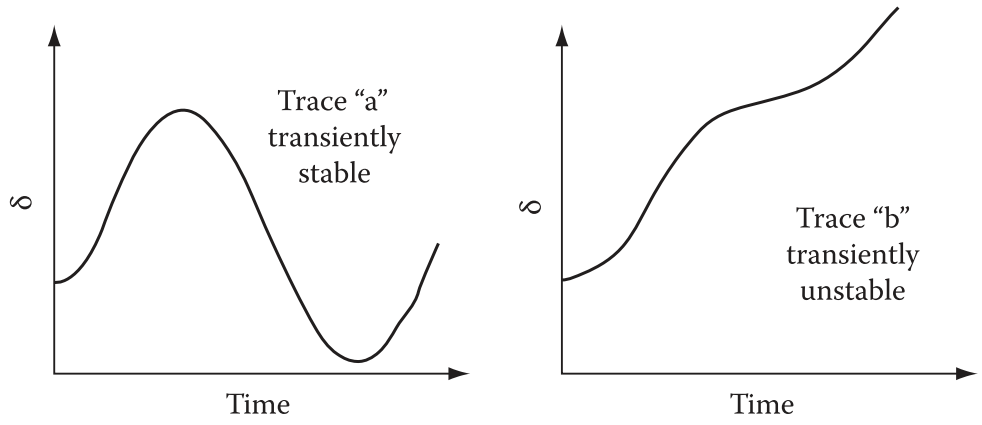


Figure 5.5: Time domain plots of the power angle of a transiently stable-, and unstable machine. From [25, Figure 9.1].

However, classical modeling is still of great use when it comes to analyzing the angular stability of a single machine connected to an infinite bus or, practically, a large network, or when detailed model data is unavailable [25, 9.3.1].

Direct methods

Direct methods of stability analysis seek to provide a statement about transient stability without explicitly solving the set of differential equations describing the dynamical behavior of a system. Indeed, the equal areas criterion was one of the direct methods for transient stability analysis, even though one that is hardly if at all scalable.

The historical overview of direct methods in [25, S. 12.1] emphasizes the importance of the work of Aleksandr Mikhailovich Lyapunov, especially for laying out his second method for stability (also called the direct method) in [42] way back in 1892. This states that a dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ having an equilibrium point at $\boldsymbol{x} = \boldsymbol{0}$ is locally stable around this equilibrium point if there exists a function (a Lyapunov function) $V(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ such that

$$\begin{aligned} V(\boldsymbol{x}) &= 0 \quad \text{if} \quad \boldsymbol{x} = \boldsymbol{0} \\ V(\boldsymbol{x}) &> 0 \quad \forall \quad \boldsymbol{x} \neq \boldsymbol{0} \\ \dot{V}(\boldsymbol{x}) &\leq 0 \quad \forall \quad \boldsymbol{x} \neq \boldsymbol{0} \quad \text{for Lyapunov stability} \\ \dot{V}(\boldsymbol{x}) &< 0 \quad \forall \quad \boldsymbol{x} \neq \boldsymbol{0} \quad \text{for asimptotic stability} \\ \dot{V}(\boldsymbol{x}) &> 0 \quad \forall \quad \boldsymbol{x} \neq \boldsymbol{0} \quad \text{for instability} \end{aligned} \tag{5.12}$$

Lyapunov stability intuitively means that solutions to the differential equation representing the dynamical system starting *close enough* to (that is, in some finite vicinity around) the equilibrium state, will remain *close enough* to (that is, in some other finite vicinity around) the equilibrium point for all future points in time.

Asymptotic stability intuitively means, that solutions starting *close enough* to (that is, in some finite vicinity around) the equilibrium state will eventually converge towards the equilibrium as time progresses.

Finally, instability means, that the state $\boldsymbol{x} = \mathbf{0}$ is an unstable equilibrium state, and so that solutions starting anywhere else will eventually leave any vicinity of the equilibrium point.

If a function V satisfies the first two criteria of (5.12), but none of the last three, then Lyapunov's direct method cannot deliver a statement about the systems (in)stability.

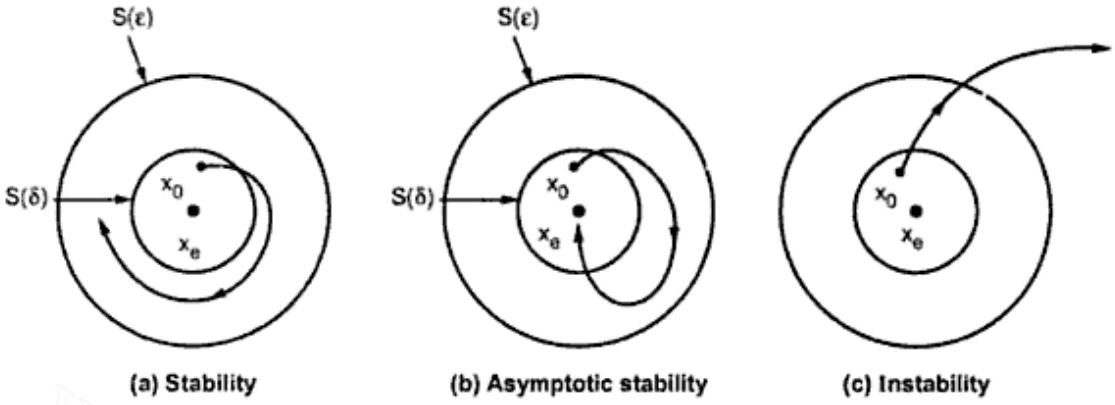


Figure 5.6: Visualization of state (in)stability in the sense of Lyapunov. From [5, Fig. 9.1].

Now, if one starts to wonder what physical meaning could V be assigned to, something like the energy-rotation function of a pendulum might come to mind first. Indeed, an energy functions is generally a valid Lyapunov candidate. However, the ingenuity of Lyapunov's method lies in that V can be any $\mathbb{R}^n \rightarrow \mathbb{R}$ function that fulfills (5.12).

Lyapunov's work did not caught much attention until the 1930s, when Soviet mathematicians revived his investigations, showing that his findings can be applied to a range of physics-, and engineering problems. However, it was not until the 1960s that academic discussion of engineering applications really took off on the western hemisphere [31, S. 3]. Starting in the late 1960s, the application of this method to power system analysis generated lively scientific discourse [25, S. 12.1].

The historical literature overview in [25, S. 12.1] reveals that much of said discourse actually revolved around finding and improving suitable Lyapunov-functions that would bring applicable results for more complex multimachine power systems.

However, the major issue with applying Lyapunov's direct method to power systems is that the solutions it delivers are approximated. Literature review in [58] divides scientific analysis of the subject matter in two major parts, with one concentrating on defining the exact Lyapunov function regardless of the size and complexity of the electric system in question, and the other focusing on defining an equivalent OMIB (one machine, infinite bus) system for a larger power system, as the Lyapunov function for the former is known. Still, both methods rely on approximation: the former because the Lyapunov function can only be expressed in a form of a series that has to be truncated at some point, and the latter because of the simplification that the OMIB model introduces.

Furthermore, [58] cites [51] in proposing that the most successful application of the OMIB-based approach is a hybrid method called SIME (single machine equivalent), which basically applies the OMIB approach and the Lyapunov stability method in each

step of a time domain integration, in order to overcome the approximation introduced by the OMIB model. This method is however still way faster than simple time domain integration, as based on the Lyapunov-criteria, a statement about the stability of a trajectory can be made in shorter time.

Model-free approaches

Methods of transient stability analysis discussed so far all rely on white-box models, that is, models that are deterministic, fairly detailed, and rooted in physical knowledge.

Other methods of transient stability analysis are based on gray-, or black-box models, and so rely on raw data more; some sources, such as [1] choose to refer to these methods as model-free instead.

An overview of methods for transient security analysis for power systems was given in [45] in the mid-1990s, dedicating a section to

- Pattern recognition methods, focusing on establishing a functional relationship between selected features of the fault-on trajectory, and the system state.
- Neural net methods, focusing on building and training neural network, that is capable of identifying system state based on a set of selected features.
- Probabilistic methods, focusing on determination of probability distributions for power system stability, e.g. Monte Carlo simulation.
- Expert system methods, combining the knowledge of human experts with other method of transient stability analysis in an "if-then-else" rule set.

Set-based approaches

Since after the millennium, publications discussing the topic of power system transient stability based on reachability analysis started emerging more and more. In [28], reachability analysis is being characterized as focusing on finding reachable sets. These are basically subsets of the state-space of a dynamical system that capture the behavior of entire groups of trajectories at once.

Reachable sets can further be subcategorized into forward-, and backward reachable sets. The former being defined as the set of all states that can be reached along trajectories that start in a specified initial set [28].

A backward reachable set, on the other hand, is the set of states where trajectories can reach a specified target set [28].

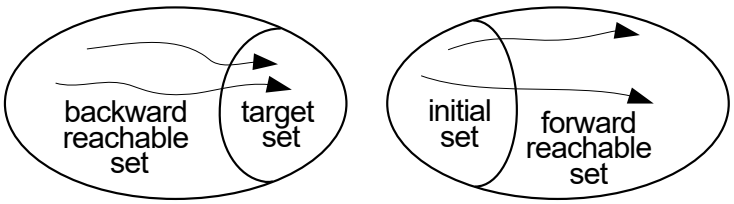


Figure 5.7: Visualization of backward-, and forward reachable sets. From [28, Fig. 1].

The main motivation for the recent academic activity around set-based stability analysis is that it combines advantages of numerical time-domain integration and direct methods. Given a set of initial-, or final states, and sets of disturbances and parameters, all possible trajectories of a system can be computed at once. The result is rigorous, as opposed to those based on Lyapunov's method that are generally -for power systems more complex than OMIB- approximate [1].

However, a significant disadvantage of set-based methods —at least in the case of power system applications— is considerable computational complexity.

Set-based power system transient stability analysis being the main focus of this work, Section 5.2 will provide an overview of how to go an about set-based critical clearing time determination.

5.2 Set-based CCT determination

With set-based CCT determination we are first considering the three operational scenarios from Section 5.1.1. We build the oscillatory models for each of the scenarios, which we will be using to extract data as follows:

- **Pre-fault scenario:** the pre-fault equilibrium point. This is the state in which machines reside up until the occurrence of a fault. Hence, this is the starting point for the fault trajectory. This is represented by the red marker in Figure 5.8.
- **Post-fault scenario:** the MRPI and admissible set barriers of the system as left after fault clearance. These would be the thick blue and red set barriers in Figure 5.8.
- **Fault-on scenario:** the fault-trajectory evolution from the re-fault equilibrium point until its intersection with the admissible set barrier (crossing the MRPI barrier) This is represented as a yellow curve in Figure 5.8.

Having obtained above data, we can determine the fault trajectory's crossing time of the MRPI set barreir t_M, and of the admissible set barrier t_A. Now, let us assume, that the fault has occurred some t time ago, when it gets cleared at once. If $t \leq t_M$, the post-fault state is somewhere along the fault-trajectory (along the yellow curve in Figure 5.2), but inside the MPRI set. Given the post-fault dynamics, Def. 2 and Def. 3, the post-fault trajectory will evolve back to the post-fault equilibrium point. If $t > t_A$, the post-fault state falls outside of the admissible set, and so the post-fault dynamics will definitely not bring the state back into the MRPI set (in fact, not even into the admissible set), and a post-fault steady-state will never reached. Consequently the critical clearing time for a specific triplet of pre-fault, fault-on, and post-fault scenarios must be somewhere between $t_M \leq t_C \leq t_A$.

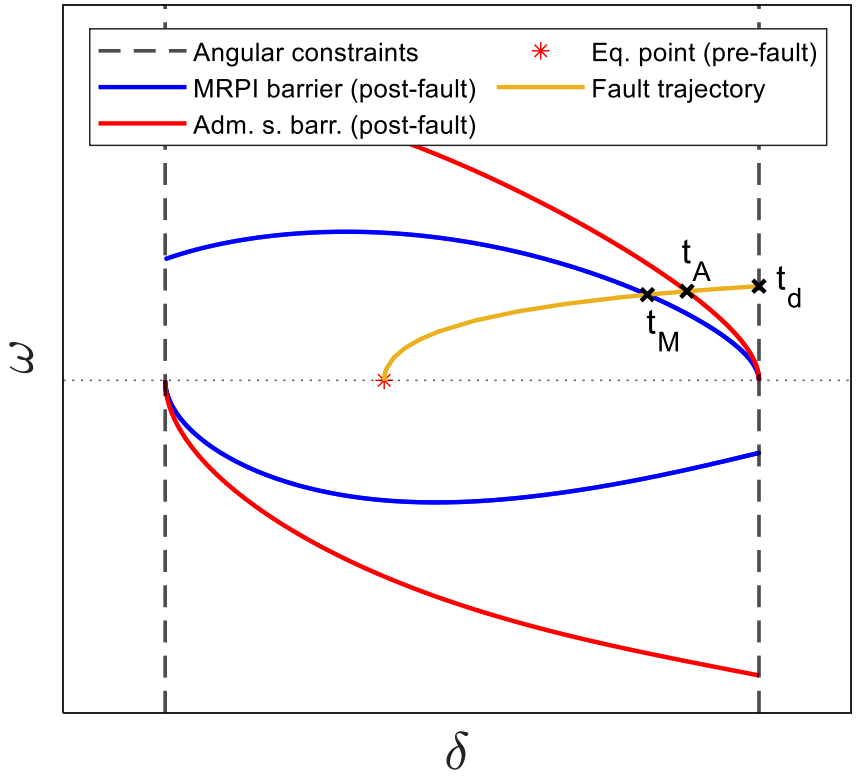


Figure 5.8: A phase diagram with a fault trajectory.

A special case is when in the post-fault scenario represents a full recovery, that is, when the pre-, and post-fault scenarios are exactly the same. In this case, $t_C = t_A$, as illustrated in Figure 5.9. This means, that if the fault is cleared exactly when the fault trajectory intersects the admissible set barrier, then the post-fault trajectory will evolve exactly along the intersected admissible set barrier, crossing the $\omega = 0$ axis in its point of ultimate tangentiality $[d_{max}\ 0]$. For a clearing time just above t_A, the post-fault trajectory would evolve outside of the admissible set, and would intersect the angular constraint just above (as in Figure 5.9) or below the point of ultimate tangentiality.

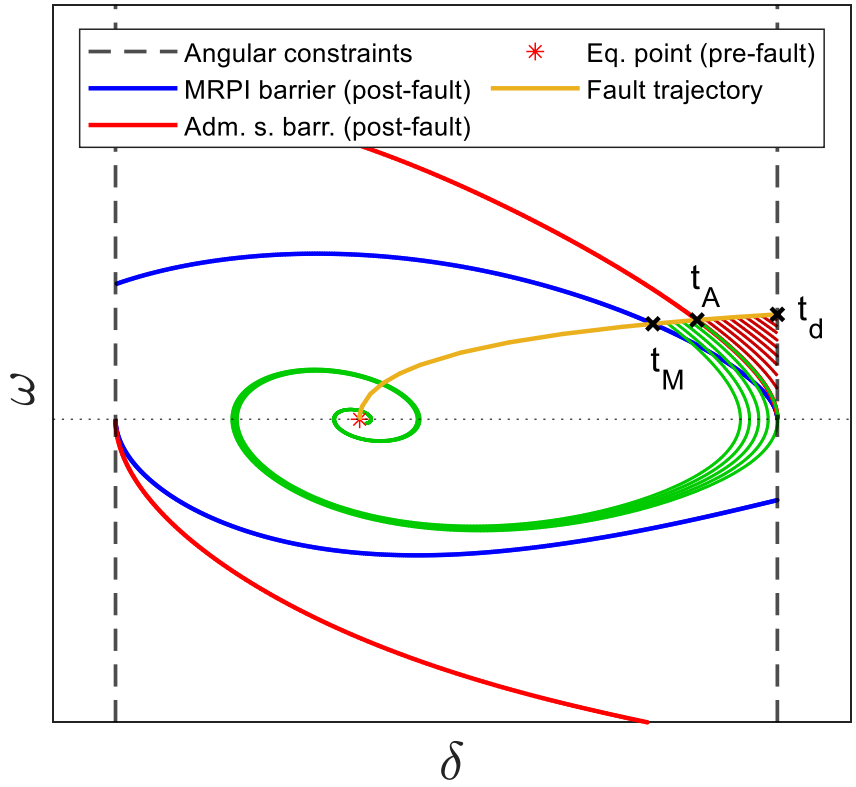


Figure 5.9: Phase diagram with identical pre-, and post fault scenarios.

In the general case —when the post-fault scenario differs from the pre-fault one—, the state corresponding to the critical clearing time lies on the fault trajectory segment between its MRPI and admissible set barrier crossing. This would be the point marked at t_C in Figure 5.10. Following the post-fault trajectory originating at this critical point, we find that this trajectory is the one among all stable trajectories that intersects the angular constraint in $\omega = 0$, that is, exactly in the point of ultimate tangentiality.

Indeed, one way of obtaining the critical clearing time would be conducting numerical backwards integration from the point of ultimate tangentiality in the post-fault system — basically redetermining the admissible set barrier — up until the intersection with the fault trajectory.

This can either be done by first obtaining the two integral curves through numerical integration, and then finding their intersection in the state space. Once an approximation (the goodness of which is dependent on the integral curves' chosen resolution) for the whereabouts of the intersection is given, the corresponding time value of the fault trajectory can be regarded as an approximation for the critical clearing time.

Another approach would be conducting forwards integration of the fault dynamics parallel to backwards integration of the critical post-fault trajectory. The difficulty hereby would be that the curves in question do not intersect each other at any given time as in there is always some positive distance between the actual states of the two evolutions. In other words, once the fault trajectory reaches the critical clearing angle, the critical post-fault trajectory might not reached the same state yet, or has already passed it. A possible solution to this problem might be dynamically adjusting the post-fault dynamics in each numerical integration step, so that it would evolve slower/faster as needed to

catch the fault trajectory. This approach would seem plausible due to that the generator dynamics (2.3) is time-invariant. Such an iterative algorithm could determine the critical clearing time for an arbitrary exactness, with the post-fault trajectory's time-domain data ignored and/or discarded.

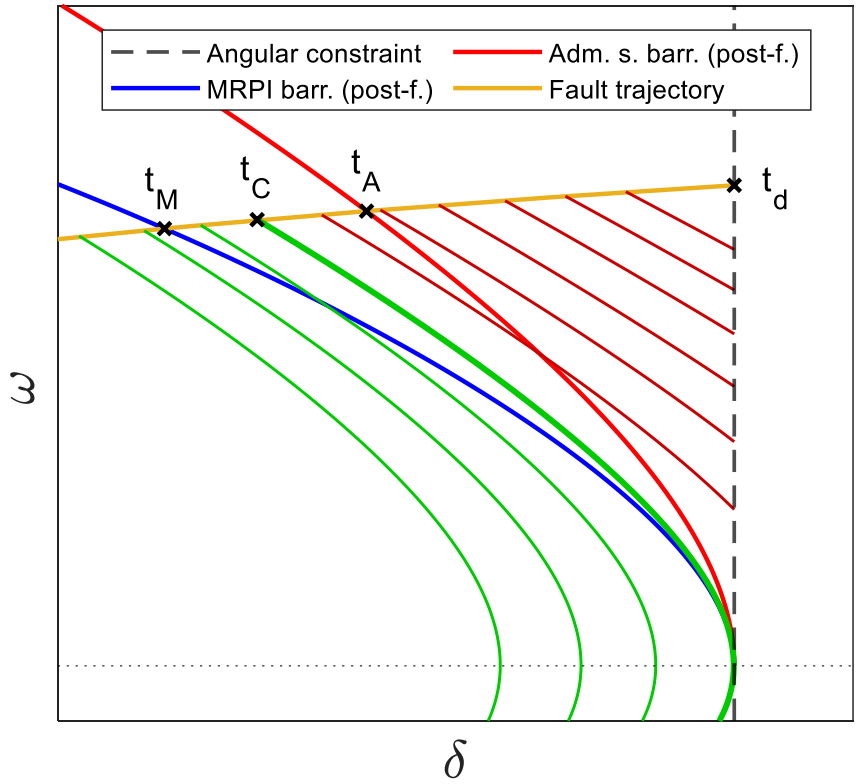


Figure 5.10: Set-based CCT determination. The green lines represent post-fault trajectories that stay inside or eventually evolve back into the MRPI set without breaking any angular constraints. Among these, the thicker green curve is the one corresponding to the critical clearing angle, and the CCT. Thin red lines are post-fault trajectories that violate the upper angular constraint.

5.2.1 Optimizing for Critical Clearing Time

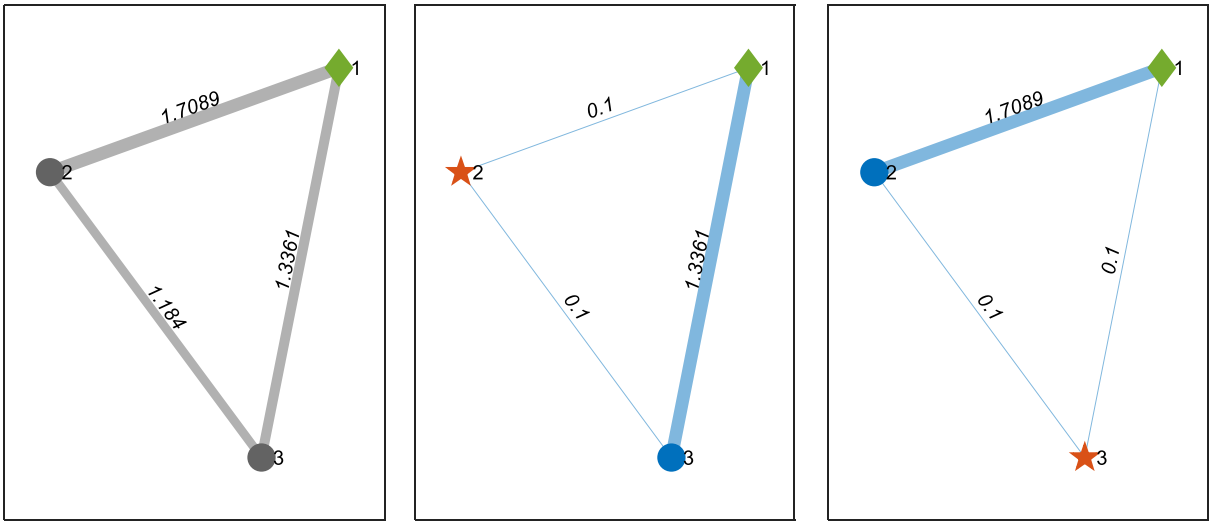
We can concede that the greater the critical clearing time value, the more desirable it is, as it represents an upper time constraint under which a fault must be cleared so that rotor angle stability can be preserved. As we have have discussed in Section 5.2, determining the CCT for a given triplet of scenarios, we have to determine the pre-fault equilibrium point, the fault-on trajectory, and the critical post-fault trajectory. Now, even with the computational complexity ignored, optimizing directly for the critical clearing time would only be plausible if two of the three scenarios (pre-fault, post-fault, and fault-on) would be pre-assumed. We would typically have the pre-fault steady state and the fault trajectory as a given, and would go on to find angular permissible angular constraints for the post-fault system so that the CCT is as big as possible.

Yet, an approach like this would disregard a cardinal advantage of the set-based method: the fact that with the set barriers —as overviewed in Section 5.1.3 under "Set-

based approaches"— all possible trajectories of a dynamical system is determined at once. Indeed, having determined the fault trajectory, and the post-fault barriers, we have already obtained —as discussed before— upper- and lower estimates for the critical clearing angle and critical clearing time as in Figure 5.10.

Of particular interest from an engineering-dimensioning point of view is the lower estimate t_M, as if the fault-on scenario remains in effect for no longer than the lowest $t_{M,i}$ of all $i \in \{1 \dots n\} \setminus \{i_{\text{ref}}\}$ machines, then stability of the whole interconnected system is retained.

We hypothesize that the greater the MRPI set in the phase diagram of a machine is, the longer (as in time and as in length) it generally takes for the fault trajectory to intersect the MRPI set barrier. We base this on the consideration that the bigger the MRPI set is, the more *interim states* must the fault dynamics go through until the critical clearing angle is reached. In order to review this hypothesis, we have conducted a test comparing MRPI area size and estimated critical clearing time values. In it, we examined the the IEEE nine-bus test system of Figure 4.1 and the EN model as in Section 4.3, with the resulting oscillatory representation is shown in Figure 5.11(a).



(a) **(b)** **(c)**

Figure 5.11: EN model oscillatory graphs. The green diamond represents the reference node, the orange star the examined node, while other nodes are marked by a dot. The numbers on each edge is the strength of the dynamical coupling (K_{ij} from (2.2)) between the nodes it interconnects. (a): Pre-fault and post-fault system. (b): Fault-on scenario for when we are examining machine 2. (c): Fault-on scenario for when we are examining machine 3.

Then, we considered the results of all the successful optimization sessions from Section 4.3 as post-fault constraints. Since we were unable to identify a fault-no scenario in which the fault trajectory would intersect the resulting post-fault barriers, we proposed two separate fault-on test cases. Both of these were constructed by direct modification of the oscillatory model by lowering the dynamical coupling Indeed a more realistic approach would have been modifying the network representation —simulating failing transmission lines, or a change in power generation/consumption or in machines' dy-

namic parameters—, based on which the fault-on oscillatory model could have been generated. Still, we find such a direct modification justifiable for the purposes of researching the correlation between MRPI set size and critical clearing time. The resulting fault-on oscillatory graphs are shown in Figure 5.11(b) and Figure 5.11(c) for when machine 2 or 3 is examined respectively. Both of these fault-on scenarios then result in a fault trajectory evolution similar to that in Figure 5.8, crossing the MRPI and admissible set barriers in the examined machine's phase diagram. Next, we recorded the fault trajectory's MRPI barrier intersection time t_M and admissible set barrier intersection time t_A, as well as the corresponding fault-on trajectory lengths s_M and s_A for all successful optimization scenarios from Section 4.3.

For the extraction of aforementioned data, we have first obtained the post-fault barriers in their entirety through numerical backwards integration, then conducted numerical —forwards— integration of the appropriate fault-on dynamics until the MRPI barrier crossing took place. This method incorporated a reasonable tradeoff between having to implement a highly customized integration algorithm, and the less accurate method of finding (interpolating) the point of barrier crossing post-integration. The numerical ODE solver `ode45` of MATLAB could be supplemented with a custom exit function that ensured that integration stops once a post-fault barrier has been reached. Having set up numerical integration this way, the solver makes sure to dynamically decrease integration step size as the stopping condition is approached, so that absolute and relative solution tolerances are met. Once the MRPI intersection point has been found and relevant data has been saved, a second integration session took place with the point of intersection as initial state, thereby obtaining the point of admissible set barrier intersection. Given the time-invariant nature of the swing dynamics (2.2), the upper estimate for the CCT t_A could then be expressed as the sum of the previously obtained lower estimate t_M, plus the time at which the second integration session has been terminated $t_A - t_M$.

Method	Solver	s_init	M	A	t_M	t_A	s_M	s_A
area	ga	-	2	26,96	0,314	0,371	10,478	11,464
area	ga	-	3	61,38	0,588	0,614	6,451	6,551
area	particleswarm	-	2	26,96	0,314	0,371	10,478	11,464
area	particleswarm	-	3	61,39	0,588	0,614	6,451	6,551
area	simulannealbnd	(4.1)	2	27,30	0,316	0,382	10,515	11,653
area	simulannealbnd	(4.1)	3	58,84	0,579	0,612	6,417	6,543
area	simulannealbnd	(4.2)	2	28,32	0,315	0,369	10,507	11,442
area	simulannealbnd	(4.2)	3	56,20	0,579	0,605	6,415	6,516
span	ga	-	2	13,27	0,222	0,262	8,517	9,448
span	ga	-	3	20,87	0,371	0,391	5,651	5,725
span	particleswarm	-	2	28,66	0,314	0,371	10,478	11,464
span	particleswarm	-	3	54,58	0,588	0,614	6,451	6,551
span	patternsearch	(4.1)	2	13,19	0,221	0,261	8,494	9,423
span	patternsearch	(4.1)	3	20,80	0,367	0,387	5,639	5,714
span	simulannealbnd	(4.1)	2	17,66	0,290	0,339	10,021	10,928
span	simulannealbnd	(4.1)	3	64,83	0,528	0,549	6,225	6,303
span	simulannealbnd	(4.2)	2	32,49	0,331	0,362	10,786	11,331
span	simulannealbnd	(4.2)	3	38,66	0,478	0,516	6,042	6,183
span	surrogateopt	(4.2)	2	33,63	0,340	0,371	10,950	11,464
span	surrogateopt	(4.2)	3	39,04	0,476	0,521	6,036	6,199

Table 5.1: Data relevant for critical clearing time estimation for each successful optimization session, and for each non-reference machine. **M** represents the number of the machine being examined, A is the MRPI area of the examined machine, t_M is the time in which the fault trajectory intersects the MRPI barrier from the pre-fault steady state, whereas t_A is the fault trajectory's evolution time between the pre-fault steady state, and the admissible set barrier intersection. Furthermore, s_M and s_A are the corresponding trajectory lengths respectively.

First we proceeded by examining the interdependence between MRPI area size, and fault-on trajectory lengths, as in our hypothesis. The corresponding results are shown in Figure 5.12.

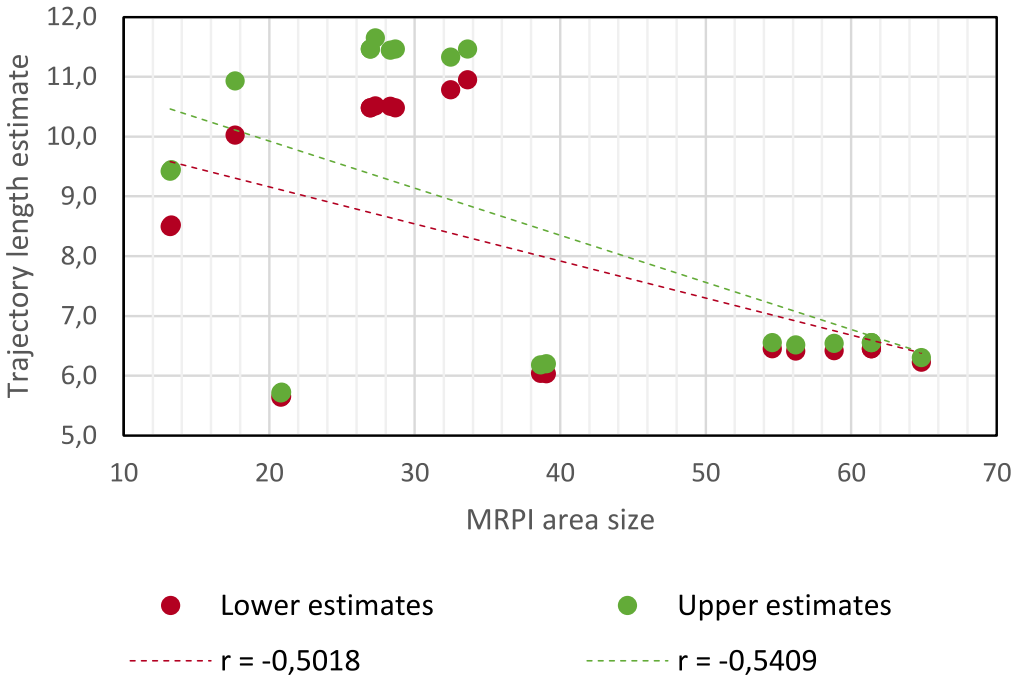


Figure 5.12: MRPI area vs. trajectory length until MRPI barrier intersection. The dashed line represents an LS linear regression.

Although there is a significant connection between the two datasets (with p values in the 10^{-6} order of magnitude at a 99,99% confidence level when running a two-tailed t-test), the Pearson correlation coefficients of around -0,5 do not reflect the expected strong linear correlation. Hence, we disregarded the initial hypothesis, and went on to undertake a comparison of MRPI area size and CCT estimates (that is, post-fault barrier crossing times of the fault-on trajectory). The results are shown in Figure 5.13.

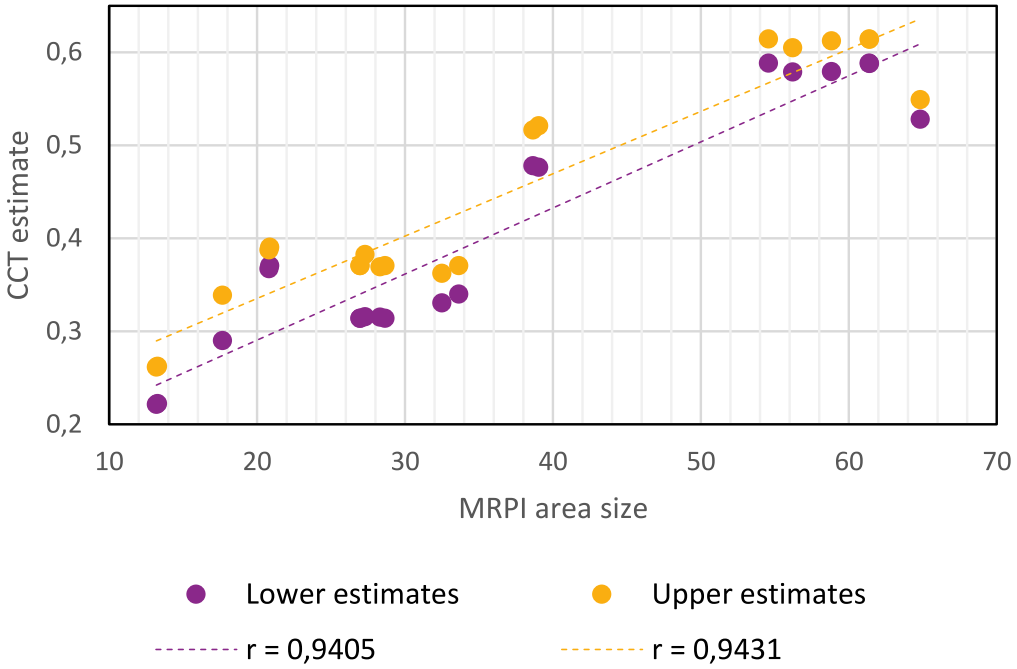


Figure 5.13: MRPI area vs lower CCT estimate. The dashed line represents an LS linear regression.

Here we observe a much stronger positive linear correlation with Pearson correlation coefficients above 0,94 and p values in the 10^{-8} range at a at a 99,99% confidence level. Although such results were not directly expected, they can be accounted for by the ever-changing evolution speed of the fault trajectory. This is illustrated in Figure 5.14.

Even though the strong correlation between MRPI area size and CCT estimates can be considered promising, it might worth deliberating over that the speed distribution shown in Figure 5.14 is unique to the examined fault-on scenario. Hence, to state whether a *good enough* correlation between MRPI set area and critical clearing time exists in a more general sense (that is, for any scenario triplet), further investigation would be propitious.

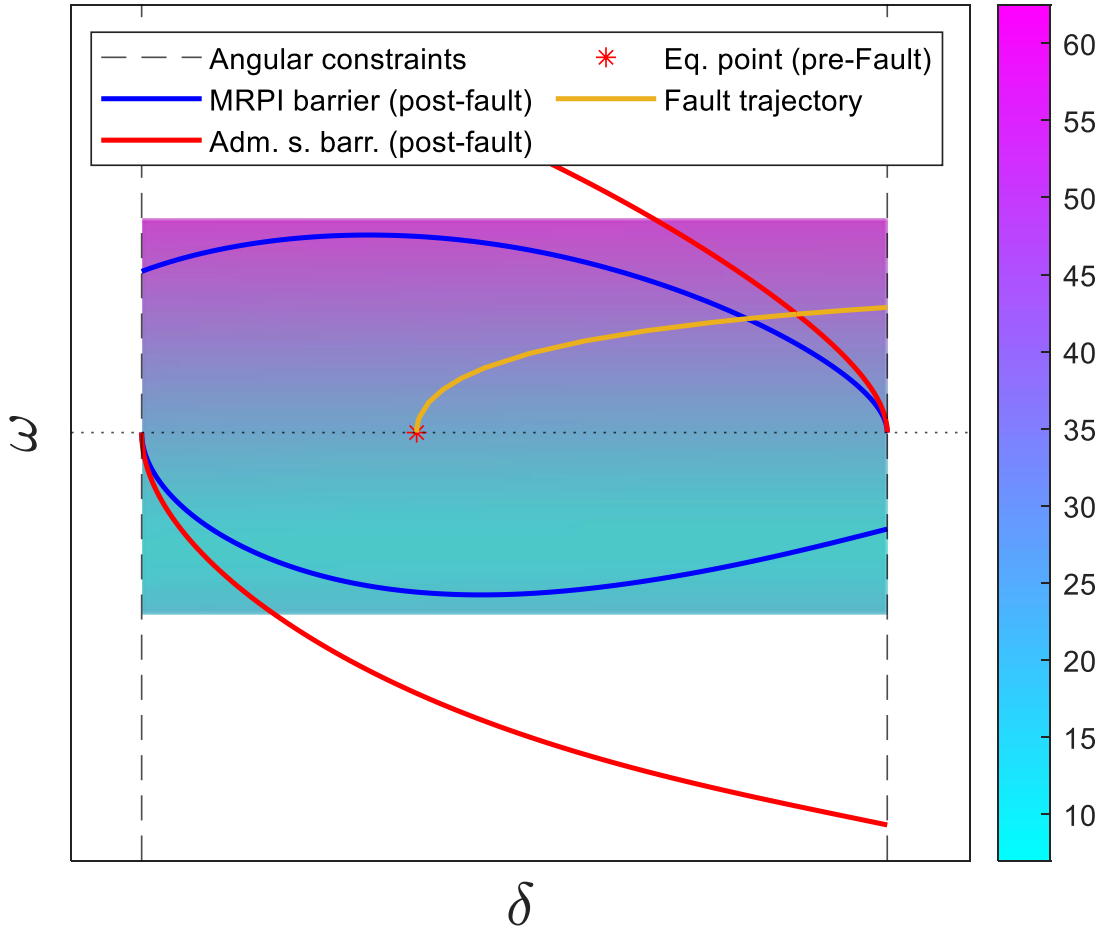


Figure 5.14: Change in evolution speed (Euclidean norm of the derivative vector) of the fault dynamics as represented by a color map. The more magenta a point is, the faster the fault dynamic evolution in that state gets.

Conclusion and Outlook

6

6.1 Summary

In this thesis we have furthered our findings in [46] by building and evaluating a framework for set-based power grid stability optimization. Having surveyed the theoretical preliminaries and preceding research, we defined and implemented an optimization problem for MRPI set size optimization. Executing and evaluating various optimization scenarios we concluded that —in accordance with [46]— an automated, methodical approach is indeed plausible, with the span-based objective function generally outperforming the area-based one. Next, we pursued to make a statement about the optimization problem's increase in computational complexity. According to our findings, the SP model's performance remains close to linear in this regard, whereas that of the EN model decreases quadratically. Towards the very end, we compared MRPI set areas and fault-on trajectory lengths, as well as MRPI set areas and CCT estimates, and have found that although we were expecting a stronger connection at the former comparison, it was actually the latter where we observed a very strong correlation. We have furthermore shown that this is likely due to that speed distribution is heterogenous over the state space.

6.2 Outlook

Even though for most part our investigations bore fruit, numerous questions also arose that we did not come to examine. These include:

- Section 4.3 has shown that the span method based objective function was much more successful at finding MRPI sets than the one based on the area method. However, we have also mentioned in Section 3.2.2 that this is in most part due to that the latter does not have the means to efficiently iterate towards valid sets when the initial design vector is not of global validity. Could we have better optimization results by building a compound goal function that would rely on the span method for arriving at global MRPI set validity, but then would conduct further optimization using the area method?
- Similarly, whether specific combinations of lower performing solvers could bring better results than a higher performing solver on its own is unknown.
- In Section 4.4 we have examined the computational complexity that arises as the examined power grid gets more complicated, yet we did not actually showcase any optimization results for power grids other than the most rudimentary IEEE

nine-bus three-generator test system. Wold our optimization processes be reasonably applicable to complex, real-life grids?

- Even though a method for set based CCT determination in Section 5.2.1 was considered, we did not make it subject to comparative analysis. How would the performance of set-based CCT determination differ from that of common methods in accuracy, reliability, and speed?
- Furthermore, while we have found a strong linear correlation between MRPI set size and critical clearing time, we did not dispute over the characteristics of this correlation. Is this correlation always linear? What determines its steepness, displacement and shape?
- In Section 5.2.1 we have been relying solely on the EN model. Would a correlation have persisted if we were to base our examinations on the SM or SP oscillatory models?
- The numerical examples for set-based CCT determination in Table 5.1 were calculated by first obtaining a numerical approximation of the fault-on trajectory, and the intersecting post-fault MRPI set barrier trajectory, both between the angular constraints. Then, the intersection of these trajectories were obtained through linear interpolation. Would it make practical sense to create an algorithm that would somehow simultaneously conduct forwards integration of the fault-on trajectory, and —on a distinct time scale— backwards integration of the MRPI barrier until both reach the intersection point, so that an arbitrarily accurate approximation of the CCT can be obtained?
- Can we fine-tune optimization so that it is not aiming for MRPI sets of great sizes in general, but so that certain critical pieces of power equipment can be granted more stability than others?

Bibliography

[1] M. Althoff, M. Cvetkovic, M. Ilic, "Transient Stability Analysis by Reachable Set Computation", presented at the IEEE Power and Energy Society International Conference and Exhibition on Innovative Smart Grid Technologies, Berlin, Germany, 14-17 October 2012.

[2] P. M. Anderson, A. A. Fouad, *Power System Control and Stability*, Wiley (2003).

[3] R. K. Arora, *Optimization: algorithms and applications*, CRC Press (2015).

[4] T. Aschenbruck, W. Esterhuizen, S. Streif, "Transient Stability Analysis of Power Grids with Admissible and Maximal Robust Positively Invariant Sets", *Automatisierungstechnik*, vol. 68, no. 12 (2020).

[5] U. A. Bakshi, M. V. Bakshi, *Modern Control theory*, Technical Publications (2008).

[6] A. R. Bergen, D. J. Hill, "A Structure Preserving Model for Power System Stability Analysis", *IEEE Transactions on Power Apparatus and Systems*, vol. 100 no. 1 (1981).

[7] N. Boumal, *An introduction to optimization on smooth manifolds* (2020). Available: http://www.nicolasboumal.net/book

[8] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press (2004).

[9] R.P. Brent, "Fast Multiple-Precision Evaluation of Elementary Functions", *Journal of the ACM*, vol. 23 no. 2 (1976).

[10] C. L. Byrne, *A First Course in Optimization*, CRC Press (2014).

[11] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, C. J. Tomlin, "Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems", *IEEE Transactions on Automatic Control*, vol. 63 no. 11 (2018).

[12] M. Chen, S. L. Herbert, C. J. Tomlin, "Fast reachable set approximations via state decoupling disturbances", presented at the IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA (2016).

[13] J. H. Chow, *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems*, Springer (1982).

[14] J. A. Doná, J. Lévine, "On barriers in state and input constrained nonlinear systems", *SIAM Journal on Control and Optimization*, vol. 51 no. 4 (2013).

[15] F. Dörfler, F. Bullo, “Kron Reduction of Graphs With Applications to Electrical Networks”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60 no. 1 (2013).

[16] M. Dornier, B. Heyd, M. Danzart, “Evaluation of the Simplex method for training simple multilayer neural networks”, *Neural Computing & Applications*, vol. 7 no. 2 (1998).

[17] T. E. Dy-Liacco, “Control of Power Systems via the Multi-Level Concept”, Ph.D. thesis, Case Western Reserve University (1968).

[18] W. Esterhuizen, T. Aschenbruck, S. Streif, “On maximal robust positively invariant sets in constrained nonlinear systems”, *Automatica*, vol. 119 (2020).

[19] G. Filatrella, A. H. Nielsen, N. F. Pedersen, “Analysis of a power grid using a Kuramoto-like model”, *The European Physical Journal B*, vol. 61 no. 4 (2008).

[20] C. A. Floudas et al., *Handbook of Test Problems in Local and Global Optimization*, Kluwer (1999).

[21] L. H. Fink, K. Carlsen, “Operating Under Stress and Strain”, *IEEE Spectrum*, vol. 15 no. 3 (1978).

[22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1988).

[23] K. Gonçalves-e-Silva, D. Aloise, S. Xavier-de-Souza, N. Mladenovic, “Less is more: Simplified Nelder-Mead method for large unconstrained optimization”, *Yugoslav Journal of Operations Research*, vol. 28 no. 2 (2018).

[24] J. Grainger, W. D. Stevenson, *Power System Analysis*, McGraw-Hill (1994).

[25] L. L. Grigsby, J. H. Harlow, J. D. McDonald, *Power System Stability and Control*, CRC Press (2012).

[26] R. Hooke, T. A. Jeeves, “"Direct Search" Solution of Numerical and Statistical Problems”, *Journal of the ACM*, vol. 8 no. 2 (1961).

[27] Ch. Huygens, Letters to R. F. de Sluse, Co. Huygens, and R. Moray, no. 1333, 1335, and 1345, The Hague, Dutch Republic (1665). Available: http://ckcc.huygens.knaw.nl/epistolarium/letter.html?id=huyg003/{1333,1335,1345}

[28] L. Jin, H. Liu, R. Kumar, J.D. McCalley, N. Elia, V. Ajjarapu, “Power system transient stability design using reachability based stability-region computation”, presented at the 37th Annual North American Power Symposium, Ames, IA, USA, 23-25 October 2005.

[29] S. G. Johnson, 18.335J Introduction to Numerical Methods, *A Brief Overview of Optimization Problems* (Spring 2019) Massachusetts Institute of Technology: MIT OpenCourseWare.

[30] R. Kalatehjari, A. Safuan, A. Rashid, N. Ali, M. Hajihassani, “The Contribution of Particle Swarm Optimization to Three-Dimensional Slope Stability Analysis”, *The Scientific World Journal*, vol. 2014, 973093 (2014).

[31] R. E. Kálmán, J. E. Bertram, “Control System Analysis and Design Via the "Second Method" of Lyapunov: I—Continuous-Time Systems”, *Journal of Basic Engineering*, vol. 82 no. 2 (1960).

[32] J. Kennedy, R. Eberhart, “Particle Swarm Optimization”, presented at ICNN’95 - International Conference on Neural Networks, Perth, WA, Australia (1995).

[33] M. J. Kochenderfer, T. A. Wheeler, *Algorithms for Optimization*, MIT Press (2019).

[34] T. G. Kolda, R. M. Lewis, V. Torczon, “Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods”, *SIAM Review* vol 45. no. 3 (2003).

[35] S. Koziel, L. Leifsson, *Simulation-Driven Design by Knowledge-Based Response Correction Techniques*, Springer (2016).

[36] S. Koziel, X. Yang, *Computational Optimization, Methods and Algorithms*, Springer (2011).

[37] P. Kundur, N. J. Balu, M. G. Lauby, *Power System Stability and Control*, MCGraw-Hill (1994).

[38] Y. Kuramoto, “Self-entrainment of a population of coupled non-linear oscillators”, presented at the International Symposium on Mathematical Problems in Theoretical Physics, Kyoto, Japan, 23–29 January 1975.

[39] K. Lange, *Optimization*, Springer (2013).

[40] A. Li, M. Chen, “Guaranteed-Safe Approximate Reachability via State Dependency-Based Decomposition”, presented at the American Control Conference (ACC), Denver, CO, USA, pp. 1-3 July 2020.

[41] Y. H. Lee, *Surrogate Model Based Optimization Teaching Tool*, https://github.com/yonghoonlee/SMBO_TeachingTool (state of Apr. 13, 2020)

[42] A. M. Lyapunov, “The general problem of the stability of motion”, Doctoral dissertation, Kharkiv University (1892).

[43] A. M. Mihirig, “Transient stability analysis of multimachine power systems by catastrophe theory”, Doctoral dissertation, University of British Columbia (1987).

[44] M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, The MIT Press (1996).

[45] M. Moechtar, T. C. Cheng, L. Hu, “Transient stability of power system-a survey”, presented at WESCON’95, San Francisco, CA, USA, 7-9 November 1995.

[46] Gy. Molnár, *Validity Analysis of Safety Sets Applied in Transient Stability of Power Grids*, Project work, Chemnitz University of Technology (2021).

[47] A. E. Motter, S. A. Myers, M. Anghel, T. Nishikawa, "Spontaneous synchrony in power-grid networks", *Nature Physics*, vol. 9 no. 3 (2013).

[48] J. A. Nelder, R. Mead, "A Simplex Method for Function Minimization", *The Computer Journal*, vol. 7 no. 4 (1965).

[49] T. Nishikawa, A. E. Motter, "Comparative analysis of existing models for power-grid synchronization", *New Journal of Physics*. vol. 17 (2015).

[50] T. Nishikawa, *Power-grid synchronization models* https://sourceforge.net/projects/pg-sync-models/ (state of Jan. 1, 2015).

[51] M. Pavella, D. Ernst, D. Ruiz-Vega, *Transient Stability of Power Systems: A Unified Approach to Assessment and Control*, Kluwer (2000).

[52] V. Podgorelec, J. Brest, P. Kokol, "Power of heterogeneous computing as a vehicle for implementing E^3 medical decision support systems", *Studies in Health Technology and Informatics*, vol. 58-59 (2000).

[53] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*, Cambridge University Press (1988).

[54] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley (2003).

[55] Y. Sun, J. Ma, J. Kurths, M. Zhan, "Equal-area criterion in power systems revisited", *Proceedings of the Royal Society A*, vol. 474 no. 2210 (2018).

[56] D. Spiller, L. Ansalone, F. Curti, "Particle Swarm Optimization for Time-Optimal Spacecraft Reorientation with Keep-Out Cones", *Journal of Guidance, Control, and Dynamics*, vol. 39 no. 2 (2016).

[57] W. D. Stevenson, *Elements of Power System Analysis*, McGraw-Hill (1955).

[58] F. Torelli, F. Milano, A. De Bonis, "A general power system control technique based on Lyapunov's function", presented at the IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cesme, Turkey, 17-20 June 2012.

[59] A. T. Winfree, "Biological rhythms and the behavior of populations of coupled oscillators", *Journal of Theoretical Biology*, vol. 16 no. 1 (1967).

[60] S. Zhan, J. Lin, Z. Zhang, Y. Zhong, "List-Based Simulated Annealing Algorithm for Traveling Salesman Problem", *Computational Intelligence and Neuroscience*, vol. 2016, 1712630 (2016).

[61] R. D. Zimmerman, C. E. Murillo-Sánchez, *MATPOWER User's Manual - Version 7.1*, Power Systems Engineering Research Center (2020).

[62] *Global Optimization Toolbox User's Guide*, Natick, MA, USA: The MathWorks, Inc., Rev. March 2021.

[63] *Optimization Toolbox™ User's Guide*, Natick, MA, USA: The MathWorks, Inc., Rev. March 2021.

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Studentenservice – Zentrales Prüfungsamt
Selbstständigkeitserklärung

Name: Molnár Vorname: Gyula geb. am: Matr.-Nr.:	Bitte beachten: 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 20.01.2022 Unterschrift:

---

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.