COMPUTATION OF LARGE-DISPLACEMENT STABILITY METRICS IN DC POWER SYSTEMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Carl J. Olthoff

In Partial Fulfillment of the

Requirements for the Degree

of

Masters of Science in Electrical and Computer Engineering

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF THESIS APPROVAL

ACKNOWLEDGMENTS

I would like to first give special recognition to my advisor, Oleg Wasynczuk. From helping me gain entry to the Purdue ECE graduate program, introducing me to PC Krause and Associates, and supporting my research endeavors, nobody has provided me with more opportunities to achieve success and for this I am forever grateful. I would also like to extend my gratitude to Steven Pekarek, Dionysios Aliprantis, and Scott Sudhoff for pushing me beyond my limits and mentoring me throughout my time as a graduate student. Lastly, I'd like to thank all the friends that I've made in the power and energy area. Nobody accomplishes anything solely by themselves and without the support of these outstanding people I would not have been able to come this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

ABBREVIATIONS

RAS	Region of Asymptotic Stability
RSSE	Region of Steady-State Equilibria
RSS	Reachable State-Space
RLDS	Region of Large-Displacement Stability
LDSM	Large-Disturbance Stability Margin

ABSTRACT

Senn, Mark D. M.S., Purdue University, August 2019. Computation of Large-Displacement Stability Metrics in DC Power Systems. Major Professor: Oleg Wasynczuk.

Due to the instabilities that may occur in dc power systems with regulated power-electronic loads such as those used in aircraft, ships, as well as terrestrial vehicles, many analysis techniques and design methodologies have been developed to ensure stable operation following small disturbances starting from normal operating conditions. However, these techniques do not necessarily guarantee large-displacement stability following major disturbances such as faults, regenerative operation, pulsed loads, and/or loss of generating capacity. In this thesis, a formal mathematical definition of large-displacement stability is described and the analytical conditions needed to guarantee large-displacement stability are investigated for a notional dc power system. It is shown possible to guarantee large-displacement stability for any piecewise continuous value of load power provided it is bounded by the peak rating of the dc source.

1. INTRODUCTION

The presence of regulated power-electronic loads in aircraft, ships, and terrestrial vehicles can lead to instabilities in their power systems. For example, ac-dc and dc-dc converters are often designed such that the output power is not affected by small perturbations in the input voltage. In other words, the output power can be modeled as a constant-power load. It can be shown that small disturbances in constant-power loads can appear as negative impedances, which can lead to destabilizing effects within the power system. As a result, much effort has been dedicated to mitigating the instabilities associated with constant-power loads.

The conventional approach to this problem is to design for small-signal stability by satisfying the Nyquist criterion. For a single source and load configuration, this criterion is satisfied if the Nyquist contour of the source impedance, Z_S, and load admittance, Y_L, does not encircle the point $(-1, 0)$ on the complex plane. The Nyquist criterion is satisfied if the product $|Z_S||Y_L|$ remains within the open unit-circle for all frequencies, leading to the Middlebrook criterion [9]. This type of approach is appealing because the analysis is well-understood and has been practiced for some time. However, with any approach based on small-signal analysis, stability is not guaranteed if the power system is subject to major disturbances such as faults, regenerative operation, and/or pulsed loads. Additionally, small-signal analysis is difficult to apply to more complicated systems where there are multiple source/load configurations.

An investigation was performed in [2] that established the minimum design requirements to ensure large-displacement stability for an ideal dc source and constant-power load. The results from this study were used to design a practicable dc source consisting of a 3-phase, permanent-magnet generator connected to an active rectifier. It was shown that the practical dc source could remain stable when subject to individual load steps between ± 1 per unit of its rated power. Very recently, the research in [1] extended the results and design paradigm from [2] to guarantee large-displacement stability when the load power varies arbitrarily within ± 1 per unit. It was concluded that the dc source in [2] is large-displacement stable for arbitrary bounded variations in load power if the system's reachable state-space is a subset of its so called region of large-displacement stability.

1.1 Thesis Contributions

The criterion for large-displacement stability set forth in [1] necessitates the calculation of the system's reachable state-space. Knowledge of a system's reachable state-space lends itself to a more comprehensive analysis of the system's controllability, so it is a natural pursuit by many authors. However, obtaining this unique set is no trivial task as it often requires reachability analysis, an approach that is both mathematically intense and not universally applicable. Reachability analysis has been shown in [4] to produce exact reachable sets of linear systems using zonotopes or ellipses. Furthermore, these techniques are readily implementable in publicly available software packages, such as the Continuous Reachability Analyzer (CORA), developed by Mathias Althoff. In the case of nonlinear dynamics however, it is very difficult to produce a solution with the well-established approach because every system contains unique and subtle nuances that may lead to numerical instabilities. Furthermore, reachability analysis for nonlinear systems yields a very conservative approximation of reachable sets [3]. In this thesis, a very tight approximation or exact calculation is extremely desirable, so the need for an alternative approach is heavily motivated for all of these reasons.

Presented in this thesis is an approach for calculating a system's reachable state-space that employs an iterative algorithm coded in MATLAB. The algorithm, appropriately named "Advancing Fronts", proceeds by expanding the boundary of the set of allowable initial conditions into the boundary of the system's reachable state-space. The result is a process that has a similar duration to that of existing reachability analysis software packages and, more importantly, supplants the need for reachability analysis entirely. Additionally, the algorithm is in close agreement to reachable state-spaces calculated by other algorithms presented herein and exceeds their capability in terms of accuracy. The algorithm is then used to show how the parameters of an average-value model of a dc source and constant-power load affect the conditions for large-displacement stability set forth in [1]. Virtually all system parameters are candidates for variation because the degree by which the design meets the conditions in [1] can be quickly reassessed with the algorithm. Ultimately, the algorithm allows an engineer to quickly choose the best physical parameters such as switching frequency, bus capacitance, etc. for the targeted application while guaranteeing large-displacement stability.

1.2 Thesis Organization

In Chapter 2, definitions are provided for large-displacement stability, reachable state-space, and all other accompanying concepts that are necessary to fully articulate the design paradigm in [1] and advancing front algorithm. In Chapter 3, a simplified average-value model of a dc power system is introduced to give physical significance to the notions of stability given in Chapter 2. There are many methods associated with computing the various mathematical sets needed to characterize large-displacement stability. The method for computing the region of large-displacement stability is contained in Chapter 3 and three methods for computing the reachable state-space are described in Chapter 4. Also in Chapter 4, several candidate designs are considered to demonstrate how large-displacement stability can be achieved for the simplified dc power system. In Chapter 5, the claims used on the simplified average-value model are then applied to a detailed model of a practicable dc power system as a proof of concept. Through repeated simulations under rigorous conditions, it is shown that the chosen physical parameters confirm large-displacement stability for the dc system studied.

2. STABILITY DEFINITIONS

Designing converter-based power systems to be stable under small disturbances is well-defined by the Nyquist and Middlebrook criteria [9]. Moreover, it is desirable to broaden the capabilities of such systems so that they remain stable following large disturbances such as pulsed loads, faults, or component failures. Before describing how large-displacement stability can be guaranteed, a brief introduction to the technical definitions contained in this thesis is in order. Since some of the definitions are rather conceptual, this chapter has been dedicated not only to defining key concepts, but providing explanation through generalized mathematical examples. This chapter concludes by defining a design paradigm that guarantees large-displacement stability for a practicable dc power system.

2.1 Lyapunov Stability

Average-value models of dc power systems can typically be expressed as a system of first-order ordinary differential equations.

$$\frac{dx}{dt} = f(x(t), u(t)); \qquad x(t_0) = x_0 \tag{2.1}$$

In the previous equation, $x(t) \in \Re^n$ is an n-dimensional vector consisting of the system's states. Examples of dc power system states include inductor currents, capacitor voltages, and control system variables. Also in (2.1), $u(t) \in \Re^m$ is an m-dimensional vector of inputs. An example of an input is the power consumed by an individual load. An equilibrium state of the solution to (2.1) is denoted x_e, where $f(x_e, u) = 0$. In typical dc power systems, both u and x_e are constant in the steady-state.

An equilibrium state for a given u is said to be stable in the sense of Lyapunov (SISL) if for every $\epsilon > 0$ there exists a $\delta(\epsilon, t_0)$ such that $|x(t_0) - x_e| < \delta(t_0)$ implies $|x(t) - x_e| < \epsilon$. Here, $|x|$ is used to denote the Euclidean norm of x. If, in addition, there exists a $\delta(t_0)$ where $|x(t_0) - x_e| < \delta(t_0)$ implies $|x(t) - x_e| \to 0$ as $t \to \infty$, then the equilibrium state is said to be asymptotically stable in the sense of Lyapunov (ASISL). If δ is independent of t_0, then the equilibrium state is uniformly ASISL (UASISL). Since (2.1) is time-invariant, an equilibrium state of (2.1) that is ASISL is also UASISL. Lastly, an equilibrium state is globally UASISL if $\delta = \infty$ [11].

In a practicable dc power system, UASISL is an essential feature for all equilibrium states associated with its operating regime. *Global* UASISL, however, is usually not feasible due to constraints on voltage, current, and/or load power. Therefore, it is understood that δ will be finite for most, if not all, equilibria of the power system. The notion of a finite δ implies the existence of a set of all $x(t_0)$ where the solution of (2.1) satisfies $|x(t) - x_e| \to 0$ as $t \to \infty$. The "region of asymptotic stability" (RAS) is the name given to such a set. By definition, the RAS wholly contains all stable trajectories, and all unstable trajectories never enter the RAS. The RAS for an equilibrium state of an arbitrary 2nd-order system is depicted in Figure 2.1.

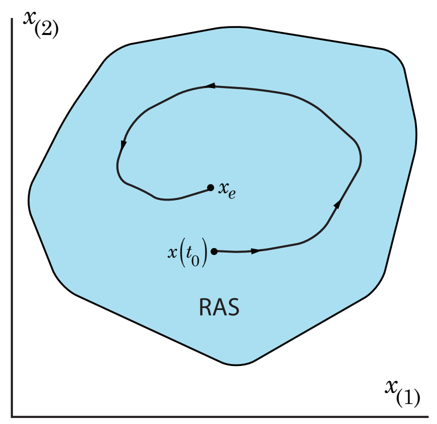

Fig. 2.1. The region of asymptotic stability for some equilibrium state, x_e. $x_{(1)}$ and $x_{(2)}$ denote the two components of $x(t) \in \Re^2$

.

In general, there will be multiple states that satisfy $f(x_e, u) = 0$ in (2.1) because the equilibrium state is dependent on the input, u, which may vary. It is assumed that there is a unique equilibrium state, $x_{e,i}$, for every input $u_i \in U$. It is important to define the region of steady-state equilibria (RSSE) as

$$\text{RSSE} = \{x_{e,i} \, \forall \, u_i \in U\} \tag{2.2}$$

Since the concepts presented in this discussion will eventually be applied to a physical power system, some additional assumptions are made regarding (2.1), inputs to (2.1), and the RSSE. First, U is assumed to be a path-connected topological space that may be, but is not necessarily convex. Second, it is assumed that a controller can be designed such that $x_{e,i}$ is at least locally UASISL for every u_i, so there is a continuous mapping from u_i to $x_{e,i}$. The first two assumptions imply that the RSSE is also a path-connected topological space. Lastly, (2.1) is assumed to be globally Lipschitz continuous on x and u, thus guaranteeing a unique solution [8]. Although the existence and uniqueness of a solution to (2.1) cannot be guaranteed in the most general case, the existence of a single preferred solution for any allowable input is a desirable feature of any practicable dc power system.

As an artifact of the controller design assumption, each $x_{e,i}$ will have a corresponding RAS, denoted RAS_i. Now, it is possible that an intersection exists between two or more RAS's. If this is the case, a trajectory that originates in the region of intersection can converge to either of the corresponding equilibrium states. For example, if there is an overlap between RAS_i and RAS_{i+1}, any $x(t_0)$ within the intersection will converge asymptotically for an input step change from u_i to u_{i+1}, and similarly for a step change from u_{i+1} to u_i, regardless of the distance between equilibrium points, $|x_{e,i} - x_{e,i+1}|$. Such a scenario would allow convergence to be achieved even under the largest allowable input step change. Therefore, it is useful to define the region of large-displacement stability (RLDS) as the intersection of each RAS_i for all i.

$$\text{RLDS} = \bigcap_{x_{e,i} \in \text{RSSE}} \text{RAS}_i \tag{2.3}$$

2.2 Single-Step Stability

The knowledge of a system's RLDS can be very useful for the analysis of its trajectories with respect to their stability. By definition of the RLDS, an $x(t_0)$ originating in the RLDS is guaranteed to also originate in RAS$_i$ for any $x_{e,i}$. Therefore, if the RLDS is a nonempty set, a trajectory within the RLDS at time $t = t_0$ subject to a constant input $u \in U$ is guaranteed to be stable. Although these conditions set $x(t_0)$ on a stable course, the trajectory can still destabilize if the input causes $x(t)$ to exit the RLDS. To understand an unstable case where $x(t_0) \in$ RLDS, it is useful to consider the arbitrary system in Figure 2.2 subject to step changes among inputs $u_{i=1,2,3}$. In Figure 2.2, only the intersection of RAS$_{i=1,2,3}$ is shown, but it is useful to assume that this intersection is the RLDS of the system.

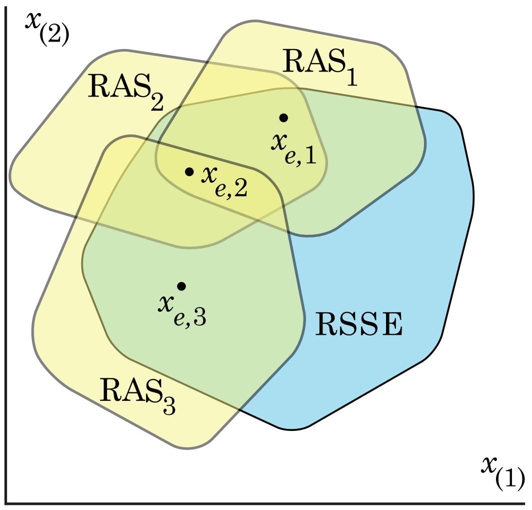

Fig. 2.2. In this system, the RLDS is assumed to be RAS$_1 \cap$RAS$_2 \cap$RAS$_3$

.

If in the system in Figure 2.2 $x(t)$ starts at equilibrium with $u(t_0) = u_2$ and $x(t_0) = x_{e,2} \in$ RLDS, the system will remain stable if u_2 is stepped to u_1 because $x_{e,2} \in$ RAS$_1$. As $x(t)$ converges to $x_{e,1}$, it will leave the RLDS. If u_1 is stepped to u_3 while $x(t) =$

$x_{e,1} \notin$ RLDS, the trajectory will be unstable because $x_{e,1} \notin \mathrm{RAS}_3$. It is possible to reach $x_{e,3}$ from $x_{e,1}$ however, so long as $x_{e,1}$ is driven to $x_{e,2}$ first.

A system is defined herein to be single-step stable if the input can be stepped from any $u_i \in U$ directly to any $u_j \in U$ at time $t = t_0$ and the resulting trajectory will converge to $x_{e,j}$. For this to be true, every $x_{e,i}$ would have to be inside the RAS of every $x_{e,j}$. Such is the case if and only if the RLDS for the system is both nonempty and

$$\mathrm{RSSE} \subset \mathrm{RLDS}. \tag{2.4}$$

Therefore, the arbitrary system shown in Figure 2.3 is single-step stable because its RSSE is a subset of its RLDS.

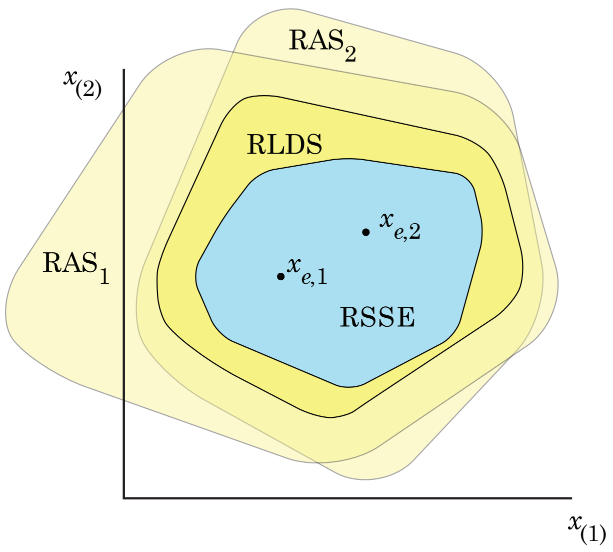

Fig. 2.3. Graphical representation of an arbitrary system with an RLDS that contains the RSSE. Thus, the condition for single-step stability is satisfied.

In contrast, the system in Figure 2.2 does not satisfy the condition for single-step stability. This was further evident in the previous example as well, where $x_{e,1}$ could not be stepped to $x_{e,3}$ directly, which is not in agreement with the definition of single-step stability.

An important point to make is that single-step stability only guarantees that a trajectory can be set on a stable path between two arbitrary equilibrium states in the RSSE for a *single* input step change at time $t = t_0$. If there is a sequence of input step changes, the possibility for an unstable response exists. As an example, it is helpful to consider the system in Figure 2.3 with $x(t_0) = x_{e,1}$. If the system in Figure 2.3 experiences an input step $u(t_0) = u_2$, the ensuing trajectory is guaranteed to remain within RAS_2, but not necessarily within the RLDS. Therefore, the trajectory could enter the region where $x(t) \in \text{RAS}_2 \notin \text{RAS}_1$. If the input is stepped from u_2 back to u_1 while $x(t)$ is in this region, there will be an unstable response. In the general case, a trajectory of a system that is single-step stable could leave the RLDS, which in turn implies that it will be outside at least 1 RAS, such as RAS_k. If, in this instance, the input is stepped to u_k, convergence to $x_{e,k}$ will not be possible.

2.3 Large-Displacement Stability

Single-step stability guarantees convergence for a single input step change within $u \in U$ at time $t = t_0$, but does not guarantee convergence for a pulsed or continuously varying input. In order to guarantee that any $x_{e,i}$ can be guided stably to any $x_{e,j}$ when the input varies arbitrarily, $x(t)$ must be within the RLDS for all t. If all trajectories are bounded by the RLDS, a new notion of stability arises that exceeds single-step stability. A system is defined herein to be large-displacement stable (LDS) if the set containing all trajectories originating from the RSSE is a subset of the RLDS.

It is useful to name the set that contains all trajectories originating from the RSSE. A reachable set at time $t = T$ is defined in literature as the set that contains all trajectories originating from a bounded initial condition set and driven by a bounded input set through the interval $[0, T]$ [3]. In this thesis, a special case of the reachable set at time $t = T$ is considered, called the "reachable state-space" (RSS). The RSS is defined in [1] as the set of all $x(t)$ originating from the RSSE driven by the set of allowable inputs U through an infinite time horizon.

$$\mathrm{RSS} = \{x(t, x_0, u(.))|x_0 \in \mathrm{RSSE}, u(t) \in U \forall t\} \tag{2.5}$$

The RSS can intuitively be thought of as the set of all states that are reachable from the RSSE, driven by all possible inputs within U. The infinite time horizon associated with the RSS implies that the RSS is equal to the reachable set at time $t = T$ when T is sufficiently large (i.e. large enough for the system to reach steady-state). The RSS then, is both time-invariant and a unique set in this regard.

Defining a unique reachable set for the purposes of [1] and this thesis allows the condition for large-displacement stability to be reformulated in a more elegant way. A system is LDS if its reachable state-space is a subset of its region of large-displacement stability.

$$\mathrm{RSS} \subset \mathrm{RLDS} \tag{2.6}$$

To further understand why the RSS must be a subset of the RLDS for large-displacement stability, it is helpful to consider $u(t)$ as a piecewise constant function of time with a countable number of step changes within U. As the number of step changes approaches infinity, the piecewise constant $u(t)$ can be made to closely approximate a continuous or piecewise continuous function of time. Modeling $u(t)$ in this way is reasonable because in practice, loads can be switched on, off, or varied continuously. If this is the case, the corresponding equilibrium state $x_e(t)$ will be a piecewise continuous function of time constrained to the RSSE. If the RSS is a subest of the RLDS, then the resultant trajectory will always remain within the RLDS while following the moving $x_e(t)$. As a result of remaining in the RLDS for all t, $x(t)$ remains stable for all time subject to any input variation within the allowable set. Thus, if (2.5) is satisfied, a system is said to be large-displacement stable.

Finally, it is important to define the large-disturbance stability margin (LDSM), which is the minimum distance from the boundary of the RSS to the boundary of the RLDS.

$$\text{LDSM} = \min |x_1 - x_2| : x_1 \in \text{RSS}, x_2 \notin \text{RLDS} \tag{2.7}$$

The LDSM is motivated by concerns with modeling error and system noise. Technically, the RSS could share a boundary with the RLDS and no notion of large-displacement stability would be violated. Unfortunately, there are inevitable uncertainties associated with the computation of each of these spaces, which means that any difference whatsoever between the model and a real system could mean that states in the RSS that are not within the RLDS exist, thus sabotaging efforts to guarantee large-displacement stability. As a way of knowing how much room for error is possible, the worst-case difference between the two spaces was calculated, thus giving rise to the formal definition of the LDSM. In Figure 2.4, an example system is shown where condition (2.6) is satisfied and the LDSM is illustrated.

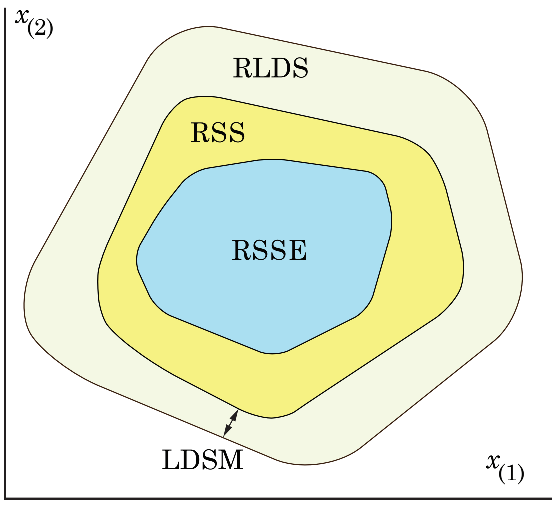

Fig. 2.4. Graphical illustration of a system that is large-displacement stable with a finite large-disturbance stability margin.

Designing systems that are large-displacement stable is simply a matter of choosing parameter values such that the RSS for the system is a subset of the RLDS. Though simple conceptually, there is one more condition that must be adhered to. It is desirable to design systems with a wide LDSM so that modeling uncertainties in the calculated boundaries cannot possibly result in reachable states existing outside of the RLDS. In addition, real systems have noise and unpredictable excitations, which can also cause the boundary of the RSS to extend outside of the RLDS. To be absolutely certain that condition (2.6) is not compromised, the LDSM must be sufficiently wide.

3. SIMPLIFIED DC POWER SYSTEM

To give context to the mathematical sets and definitions provided in Chapter 2 in terms of dc power systems, an average-value model of a simplified dc power system is presented in this chapter. The model, originally set forth in [2] as a simplified "study system", is extremely tractable and allows all of the concepts in this thesis to be presented in an abstract sense while still retaining physical significance. Also in this chapter, the impact of the study system's parameters on the stability of its trajectories is analyzed. A method for calculating the system's RSSE and RLDS is also contained in this chapter. These spaces are then used to make further implications about the stability of its trajectories.

3.1 Study System

A simplified model of a power electronic system is illustrated in Figure 3.1. The model consists of a controllable current source, an output capacitor, and a constant-power load. The current source controller exhibits first-order filter characteristics, meaning the commanded current and source current are related by

$$i_{dc} = \frac{i_{dc}^*}{\tau s + 1} \tag{3.1}$$

In (3.1) and subsequent equations, starred variables represent commanded values, while un-starred variables represent actual values. The controller sets hard limits on the source current, which are generalized as $i_{\lim}$ and $-i_{\lim}$, so (3.1) is only valid within these limits.

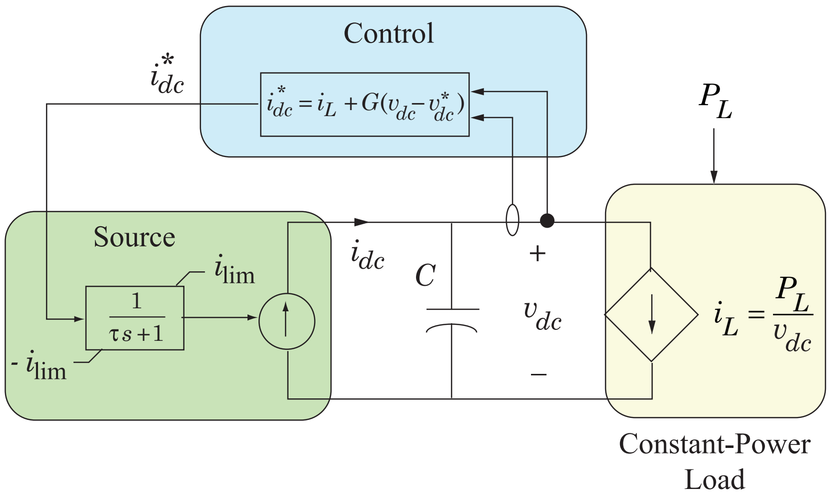

Fig. 3.1. Simplified model of a dc power system.

The load is represented as an idealized constant-power load, where the load current is given by

$$i_L = \frac{P_L}{v_{dc}} \tag{3.2}$$

In (3.2), the power P_L is considered to be an input to the system. P_L is allowed to be a constant parameter or a bounded piecewise continuous function of time, much like how $u(t)$ is used in the context of (2.1). The voltage on the capacitor is governed by

$$\frac{dv_{dc}}{dt} = \frac{i_{dc} - i_L}{C} \tag{3.3}$$

The equation governing source current can be obtained by converting (3.1) to the time-domain.

$$\frac{di_{dc}}{dt} = \frac{i_{dc}^* - i_{dc}}{\tau} \tag{3.4}$$

For the remainder of this thesis, all values associated with (3.3) and (3.4) will be normalized relative to their base ratings so that the results from the study system can be applied to arbitrary voltage, current, and power levels. The subscript b will be used to denote base rated values, while the subscript pu will be used to denote per-unit values. For example, the rated dc voltage is expressed as $V_{b,dc}$, while the rated impedance is expressed as Z_b. Per-unit values are typically expressed as the ratio of their value to their rated value.

$$v_{dc,pu} = v_{dc}/V_{b,dc} \tag{3.5}$$

$$i_{dc,pu} = i_{dc}/I_{b,dc} \tag{3.6}$$

$$C_{pu} = CZ_b = C\frac{V_{b,dc}}{I_{b,dc}} \tag{3.7}$$

The subscript pu will be omitted in subsequent equations unless specifically needed for clarity. Note that the per-unit voltage and current are unitless, while the per-unit capacitance has units of seconds. Using the definition of C_{pu}, it is possible to eliminate capacitance from (3.1) by defining scaled time and the scaled time constant as

$$t' = t/C_{pu} \tag{3.8}$$

$$\tau' = \tau/C_{pu} \tag{3.9}$$

Substituting (3.8) and (3.9) into (3.3) and (3.4) yields the scaled time equations which are convenient because the sytem behavior now applies to arbitrary values of capacitance.

$$\frac{dv_{dc}}{dt'} = i_{dc} - i_L \tag{3.10}$$

$$\frac{di_{dc}}{dt'} = \frac{i^*_{dc} - i_{dc}}{\tau'} \tag{3.11}$$

A feedforward/feedback control scheme is used to regulate the output voltage. Specifically, the commanded source current is a combination of the load current and the voltage error.

$$i_{dc}^* = i_L + G(v_{dc}^* - v_{dc}) \tag{3.12}$$

where v_{dc}^* is the commanded output voltage and G is the gain. The system comprised by (3.10) and (3.11) and its feedforward/feedback structure is illustrated as a normalized model in Figure 3.2.

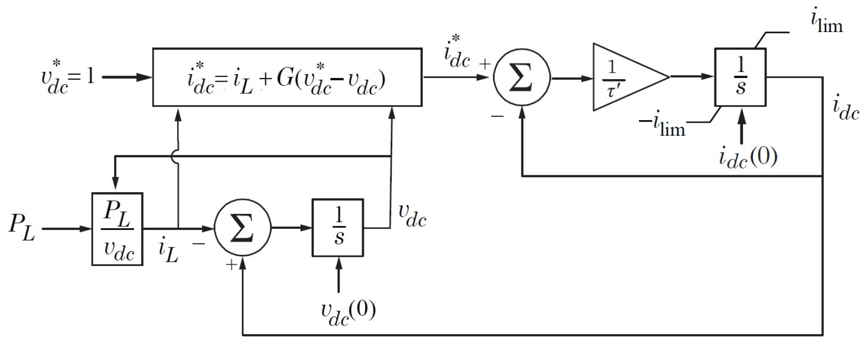

Fig. 3.2. Normalized model of the power-electronic system in Figure 3.1. Blocks with gain $\frac{1}{s}$ represent integration with respect to time.

Substituting (3.2) into (3.10) and (3.12) into (3.11) yields the following system of equations.

$$\begin{aligned} \frac{dv_{dc}}{dt'} &= i_{dc} - \frac{P_L}{v_{dc}} \\ \frac{di_{dc}}{dt'} &= \frac{1}{\tau'}\left(\frac{P_L}{v_{dc}} + G(v_{dc}^* - v_{dc}) - i_{dc}\right) \end{aligned} \tag{3.13}$$

The parameters associated with (3.13) are G, τ', and $i_{\lim}$, while the inputs are v_{dc}^* and P_L. In the following studies, it is shown how the choice of the parameters affects the stability of trajectories governed by (3.13).

3.2 Effects of the Source Time Constant and Transient Overload Capacity on Stability

It is desirable to see how each parameter of (3.13) determines the stability of its trajectories on an individual basis. To do this, numerical simulations were conducted in [2] with (3.13) operating at its rated voltage ($v_{dc}^{*} = 1$) and between ± 1 of its rated load power ($-1 \leq P_L \leq 1$). Starting with an investigation of τ' and i_{lim}, the gain was chosen arbitrarily and P_L was stepped from 0 to 1 per unit. It is worthwhile to note that a practical response time of the source, τ', is a nonzero value. This implies that the source current cannot change instantaneously, so it is expected that the source will need to supply more than its rated amperage when the system is stepped to its rated load power. Therefore, it is important to choose i_{lim} to be greater than 1 per unit to maintain stability during such transients. The difference between i_{lim} and 1 is henceforth referred to as the system's transient overload capacity.

It was concluded in [2] that a trade-off exists between the responsiveness of the source and its transient overload capacity. In Figure 3.3, the maximum allowable time constant for stability was determined for multiple values of i_{lim}.

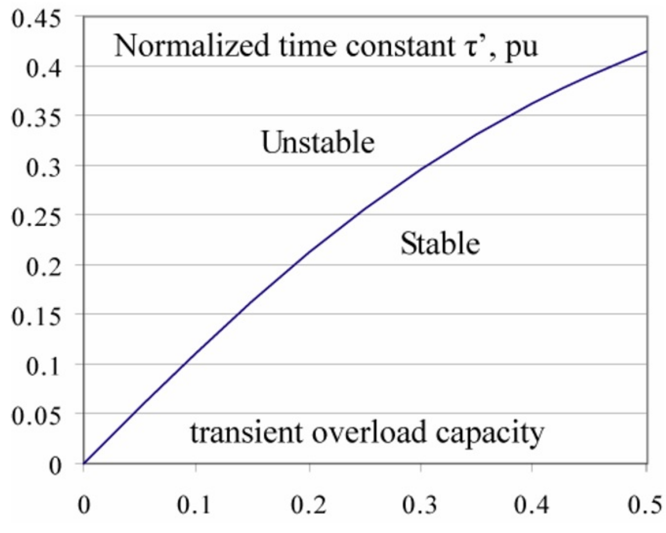

Fig. 3.3. Normalized time constant versus transient overload capacity.

The gain associated with Figure 3.3 was $G = 2$, and the initial condition was $v_{dc} = 1$, $i_{dc} = 0$. One important result of Figure 3.3 is that the time constant approaches a maximum limit, equal to 0.581. Based on this information, any transient overload capacity beyond 50 percent results in marginal improvement to the maximum allowable time constant, so i_{lim} was chosen to be 1.5 per unit. Hence, any source time constant less than or equal to 0.4 seconds in scaled time should result in a stable trajectory for a load step from 0 to 1 per unit. In any case, decreasing the time constant always increases a trajectories tendency towards stability.

As for the influence of gain on stability, increasing the gain should increase the number of stable trajectories. Intuitively, a larger response from the controller might compensate for a slower response time. Ultimately, it is anticipated that increasing the gain should increase the area of the RAS for a given equilibrium point. A similar effect is anticipated with respect to the RLDS if the upper limit of the load power is decreased. A less demanding load decreases the chance that a load step will exhaust the available energy stored in the output capacitor. To test these hypotheses, the RAS for all equilibrium points associated with (3.13) must be calculated.

3.3 Computation of the Region of Asymptotic Stability

Since the RAS is critical to the stability claims made in this thesis, having an accurate calculation of its boundary is imperative. As a solution, an algorithm was developed that yields a high-resolution boundary of a given RAS for an n-dimensional system, provided a time-domain simulation of the system exists. The objective of the RAS calculator is to locate the boundary of the set of all initial conditions that allow their resultant trajectories to converge to the specified equilibrium point. As opposed to studies performed in [7] wherein genetic algorithms were used to calculate the RAS, the algorithm proposed in this thesis searches for points that satisfy the definition of the RAS through repeated trial simulations. Methods of this nature are not typically preferable, but are nonetheless a practical choice for low-dimensional systems.

For an n-dimensional system, the RAS is located by fixing $n-1$ states in a predefined search space and performing a binary search on the n'th state. Once the *minimum* value for convergence is found, the process repeats, fixing $n-1$ new points. After iterating through the entire search space, the result is a fine boundary that separates the set of initial conditions that converge from all other initial conditions. Generally speaking, finding the minimum state for convergence is not sufficient to articulate the RAS boundary. In the most general case, a minimum *and* maximum bound on the n'th state will need to be located. For example, if the system has 3 dimensions, then the RAS will be some volume in $\Re^3$. Fixing 2 dimensions will result in a binary search along a straight line in $\Re$. Depending on the convexity of the RAS, there are 1 or more intervals in the search space where the initial conditions will converge. Only by locating every interval can the RAS boundary be fully resolved.

Properties of (3.13) have been exploited to expedite the binary search process. In particular, no upper bound on the voltage state exists within the search space, so only the minimum voltage for convergence needs to be located. This is because there is no maximum energy state of the bus capacitor such that the power demand cannot be met. In other words, there exists only a minimum voltage level where a trajectory is guaranteed to be guided to equilibrium. In the general case, an upper bound and lower bound need to be located, so several binary searches would need to be conducted. Another property of the system that can be exploited is that the voltage state converges to 1 per unit no matter the choice of input, u. This implies that the test for convergence can be greatly simplified. Simply check if the voltage state is near 1 per unit instead of checking for convergence to every unique state in the RSSE. The non-existence of an upper bound on the RAS with respect to voltage and the knowledge that voltage always converges to 1 per unit both increase the speed of the RAS calculator.

The procedure of the RAS calculator can be summarized by the following psuedo-code:

```
CalculateRAS
    $x_1 \Leftarrow linspace(x_{1_{\text{lower}}}, x_{1_{\text{upper}}}, N)$
    $x_2 \Leftarrow linspace(x_{2_{\text{lower}}}, x_{2_{\text{upper}}}, N)$
                ..........
    $x_{n-1} \Leftarrow linspace(x_{n-1_{\text{lower}}}, x_{n-1_{\text{upper}}}, N)$
    for $i = 1 : N$
        $x_1 \Leftarrow x(i)$
        for $j = 1 : N$
            $x_2 \Leftarrow x(j)$
                ..........
            $x_{n_{\text{lower}}} \Leftarrow A$
            $x_{n_{\text{upper}}} \Leftarrow B$
            for $k = 1 : M$
                $x_n \Leftarrow (x_{n_{\text{lower}}} + x_{n_{\text{upper}}})/2$
                simulate $'model.slx'$
                if $\text{Xn. data(end)} > 0.99$
                    $x_{n_{\text{upper}}} \Leftarrow x_n$
                    else
                    $x_{n_{\text{lower}}} \Leftarrow x_n$
                end if
            end for
            $x_{n_{\text{minimum}}} \Leftarrow x_n$
        end for
    end for
```

As seen in the preceding psuedo-code, there are $n-1$ nested for-loops for an n-dimensional system. The outermost loop fixes state $x_1(t_0)$, while the second outermost loop fixes $x_2(t_0)$, and so on until the innermost loop performs the binary search. To locate the minimum $x_n(t_0)$ for convergence, the binary search uses a predetermined range $[\underline{x_n}(t_0), \overline{x_n}(t_0)]$ and takes the initial guess to be the average of the upper and lower bounds of this range. The binary search simulates the ensuing trajectory using a Simulink$^{\text{TM}}$ model. Inputs to the model are the power demand and the initial condition, and the outputs are the system states. If the trajectory does not converge, the lower bound of the search range is updated to be the initial guess. If the trajectory converges, the upper bound of the search range is updated to be the initial guess. This procedure is illustrated in Figure 3.4.

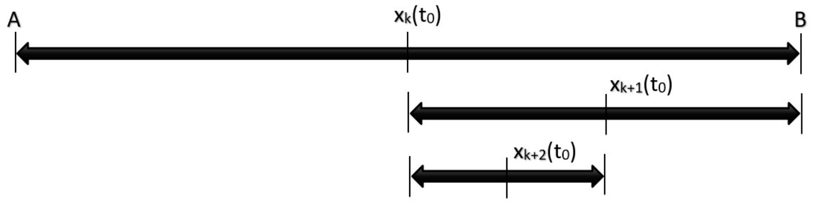

Fig. 3.4. Graphical representation of the binary search algorithm. The black arrows represent the range of values being searched, while ‘A’ and ‘B’ denote the minimum and maximum bounds on the search interval. In this example, initial guess x_k was found not to converge, so all states lower than x_k are excluded in the next search iteration, $k+1$.

The binary search mechanics are a reflection of the following assumptions: If an $x_n(t_0)$ causes the initial condition to converge, all $x_n(t_0)$ greater will also converge. If an $x_n(t_0)$ causes the initial condition to diverge, all $x_n(t_0)$ lower will also diverge. Since only the minimum voltage state is a requirement to articulate the RAS boundary, a single binary search is sufficient.

The resolution of the boundary is dependent on the choice of the search space. For example, if $x_{1,2,\ldots,n-1}(t_0)$ are defined as N equally spaced points within the search space, increasing N will greatly increase the resolution of the solution, albeit at great expense to the computational time. The most significant thing that can be done to increase the speed of the RAS algorithm is to condense the search space. This can be done by decreasing the resolution of the calculated boundary, or by reducing the number of fixed states through reduced-order modeling techniques. It can be noted that for n states, there are equally as many for-loops. Reducing the order of the system reduces the number of nested for-loops, thus increasing algorithm speed.

3.4 Locating the RSSE and RLDS for (3.13)

The RAS for $x_e = (1, 1)$ was obtained using the algorithm described in the previous section with parameters $G = 2$, $\tau' = 0.2$, $i_{\text{lim}} = 1.5$, $v_{dc}^* = 1$, and $P_L = 1$. This RAS is illustrated as the shaded region in Figure 3.5. The trajectories shown in Figure 3.5 were obtained by simulating the system in (3.13) with Simulink$^{\text{TM}}$ R2018b.

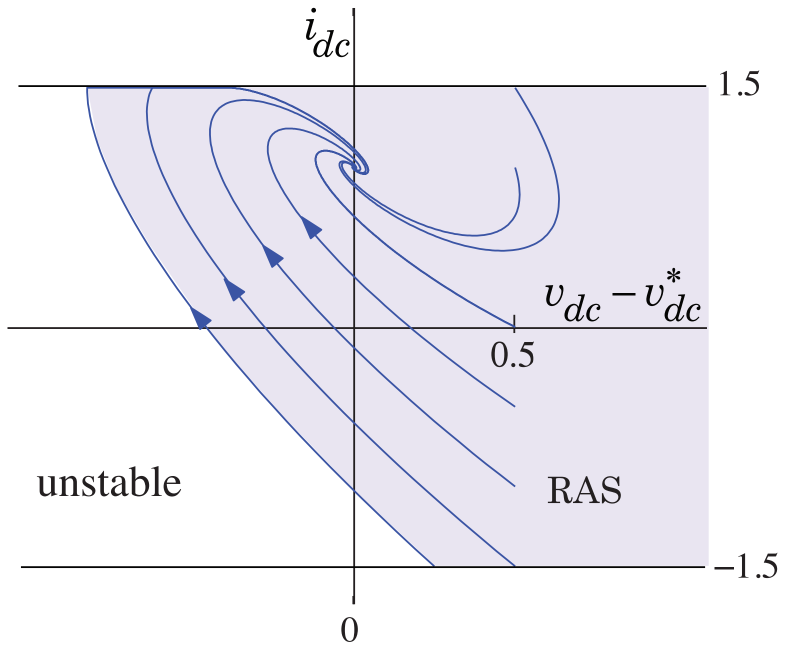

Fig. 3.5. Sample trajectories and the RAS (shaded region) for $x_e = (1, 1)$

The initial conditions for the trajectories were chosen to be $v_{dc} = 1.5$, while i_{dc} varies between its upper and lower limits. In this figure only, the voltage error, $v_{dc} - v^*_{dc}$, was plotted on the horizontal axis. From the figure, it is evident that the steady-state voltage error for the proposed control strategy is zero. It may be recalled that only a single boundary of the RAS is articulated by the algorithm. As mentioned in Section 3.3, the hard limits on current force the upper and lower boundaries, and all states to the right of the calculated boundary are within the RAS.

The speculations in Section 3.2 regarding the influence of gain and load power on stability can be resolved by using the RAS calculator. It was determined that increasing the gain does in fact increase the area of the RAS for $x_e = (1, 1)$. To illustrate this point, the RAS in Figure 3.5 was computed again with $G = 5$ and a third time with $G = 8$. The resultant spaces are overlayed in in Figure 3.6.

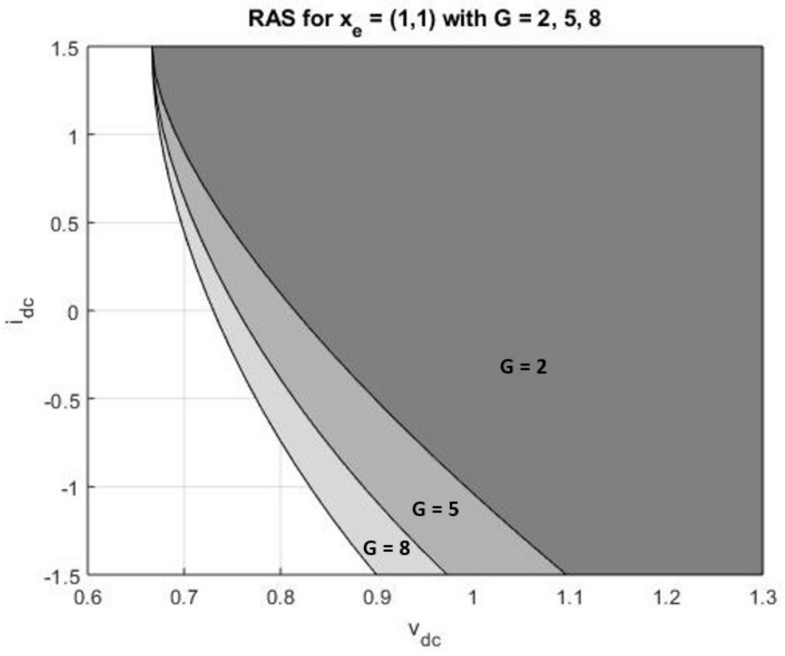

Fig. 3.6. The RAS for $x_e = (1, 1)$ is shown for $G = 2, 5,$ and 8 from right to left, respectively.

By repeating these calculations for multiple equilibrium points, it was verified that increasing the gain results in a larger RAS area for any arbitrary equilibrium point of (3.13). While each increase in gain resulted in a marginal increase in the total RAS area, it nonetheless suggests that increasing the system gain increases the number of possible stable trajectories. Before any conclusions can be made regarding the influence of the maximum load power on stability, the RSSE and RLDS for (3.13) must first be calculated.

Calculating the RSSE is a simply a matter of expressing x_e as a function of the inputs, P_L and v_{dc}^*. Since the intention is to operate the system at its rated voltage, v_{dc}^* is fixed at 1 pu. P_L is taken to be any value within $[-1, 1]$ pu, accounting for rated conditions and regenerative operation. It is easily shown that for all equilibrium states of (3.13), the steady-state voltage error is zero and the source current takes whatever value that P_L is chosen to be. Therefore, the RSSE for $-1 \leq P_L \leq 1$ is

$$\text{RSSE} = \{(v_{dc}, i_{dc}) : v_{dc} = 1 \text{ and } -1 \leq i_{dc} \leq 1\} \tag{3.14}$$

The RSSE of (3.13) is illustrated in Figure 3.7.

Locating the boundary of the RLDS is a matter of calculating the RAS_i for all $x_{e,i} \in$ RSSE and overlapping them on a single plot. Their intersection, then, can be obtained by inspection, thus yielding the RLDS. Note that since the RSSE is a continuous space, this would imply that an infinite number of RAS_i need to be calculated. This is not necessary however, as a pattern becomes evident when several RAS_i are plotted. As P_L varies from $+1$ to -1, the RAS boundaries shift in the direction of negative v_{dc}. Again, since all points to the right of the calculated boundaries are within each RAS, the intersection of all RAS_i is equal to $\text{RAS}_{P_L=1}$. Thus, the RLDS for (3.13) is equal to its RAS for $x_e = (1, 1)$. As a way of illustrating this procedure, the RAS_i for several values of P_L are shown in Figure 3.7. It can be seen that the RLDS is the most heavily shaded region in the figure.

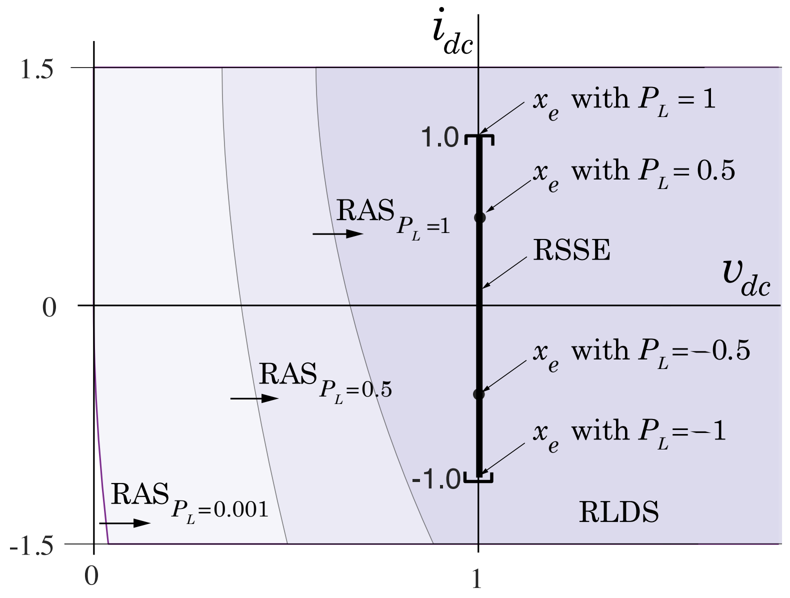

Fig. 3.7. The RAS for $P_L = 0.001, 0.5,$ and 1 are shown. The RLDS is equal to the RAS associated with $P_L = 1$. In this figure, G is assumed to be 5.

From Figure 3.7, it is evident that the upper bound on P_L determines the boundary of the RLDS, and therefore plays a major role in the stability of system trajectories. Decreasing the upper limit on P_L increases the area of the RLDS significantly, thus giving way to notions such as single-step stability and large-displacement stability. For (3.13) to be single-step stable, the RSSE must be a subset of the RLDS. From Figure 3.7, single-step stability can be confirmed by inspection. Therefore, (3.13) can safely handle any single load step within ± 1 pu.

Single-step stability, however, does not ensure stability for large input changes that occur rapidly in succession. In other words, large-displacement stability is not yet guaranteed. To ensure that stability is maintained under the description of $u(t)$ given in Section 2.3, the RSS for (3.13) must be a subset of the RLDS. By calculating the RSS and overlaying the RLDS, large-displacement stability can be confirmed by inspection. Much like the RAS calculation, special attention was given to producing an accurate, high resolution RSS boundary. In the following chapter, several methods for computing the RSS are set forth, compared, and analyzed. Once the RSS boundary is located, the design paradigm for large-displacement stability can be validated.

4. COMPUTATION OF THE REACHABLE STATE-SPACE

Three methods to calculate the RSS of a nonlinear system are described in this chapter. To preface the methods of computation, the first section of this chapter is dedicated to describing an example RSS computation using techniques that other authors have established for similar problems. The second, third, and fourth sections in this chapter are dedicated to describing the reachability analysis techniques used by CORA for MATLAB, the discrete-space search algorithm, and the advancing fronts algorithm, respectively. The performance of the latter two algorithms is analyzed and compared in the fifth section. Lastly, the calculated RSS is used to choose parameters for a dc power system in accordance with the design paradigm for large-displacement stability.

4.1 Reachable Sets and the RSS

One possible way to calculate the RSS for (3.13) is to use reachable sets. This is a natural first inclination because other authors have explored the computation of reachable sets in great detail in [3] and [4]. As an example of how reachable sets could be used to compute the RSS, it is useful to consider the following system, consisting of two coupled oscillators taken from [4]. The state dynamics of the system are governed by

$$\frac{d}{dt}x(t) = \begin{pmatrix} \sigma & -\omega \\ \omega & \sigma \end{pmatrix} x(t) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(t) \tag{4.1}$$

where the initial condition set it $x(0) = \alpha[\frac{1}{10}, 0]^T$ for $\alpha \in [-1, 1] \subset \Re$ and the input set is $u \in [-1, 1] \subset \Re$. The uncertainties in the initial conditions are reflected by the choice to represent them as the line segment from $(-\frac{1}{10},0)$ to $(\frac{1}{10},0)$. Similarly, the uncertainty in the input $u(t)$ is modeled as a strictly arbitrary value within [-1,1]. The solution to the coupled oscillators in (4.1) is readily obtained.

$$x(t) = \alpha \begin{pmatrix} e^{\alpha t} cos(\omega t) \\ e^{\alpha t} sin(\omega t) \end{pmatrix} x(0) + \int_0^t \begin{pmatrix} e^{\alpha(t-\tau)} cos(\omega(t-\tau)) \\ e^{\alpha(t-\tau)} sin(\omega(t-\tau)) \end{pmatrix} u(\tau) d\tau \qquad (4.2)$$

Reachability analysis techniques presented in [3] and [4] allow for the exact calculation of the reachable set at an arbitrary time $t = T$ for the linear system in (4.1). In Figure 4.1, the reachable set at time $t = \frac{\pi}{2}$ was computed using CORA, a software compatible with MATLAB that uses the techniques in [3].

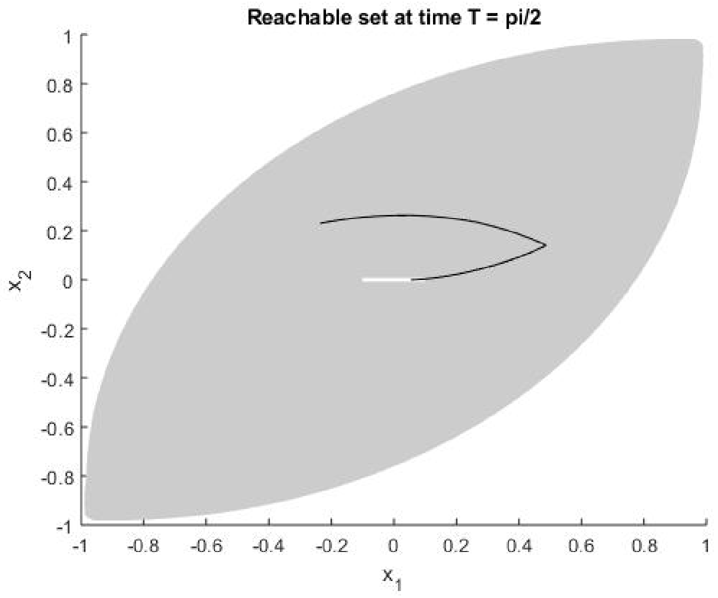

Fig. 4.1. The reachable set at time $\frac{\pi}{2}$ seconds plotted using CORA. The grey region represents the set that contains all possible trajectories originating from the white rectangle, which is the initial condition set. The black line is a sample trajectory following the solution (4.2) with arbitrary inputs. System parameters are $\sigma = -0.1$ and $\omega = 1$.

Although the RSS is uniquely defined in the context of the RSSE of (3.13), it can be assumed that the equivalent RSS for the system in (4.1) is the reachable set at $t = T$

when T is sufficiently large. Using CORA, the RSS for (4.1) was calculated and is shown in Figure 4.2. Therein, 10 seconds is being used as a practical realization of the "infinite" time horizon associated with the RSS. In the case of the dynamics in (4.2), 10 seconds is sufficient time for these trajectories to traverse every reachable state, so the reachable set at time $t = 10$ is equal to the system's RSS. It is interesting to note that a sufficiently large T implies that the system has reached steady state, but the boundary of the RSS typically contains states that occur in the transient response.

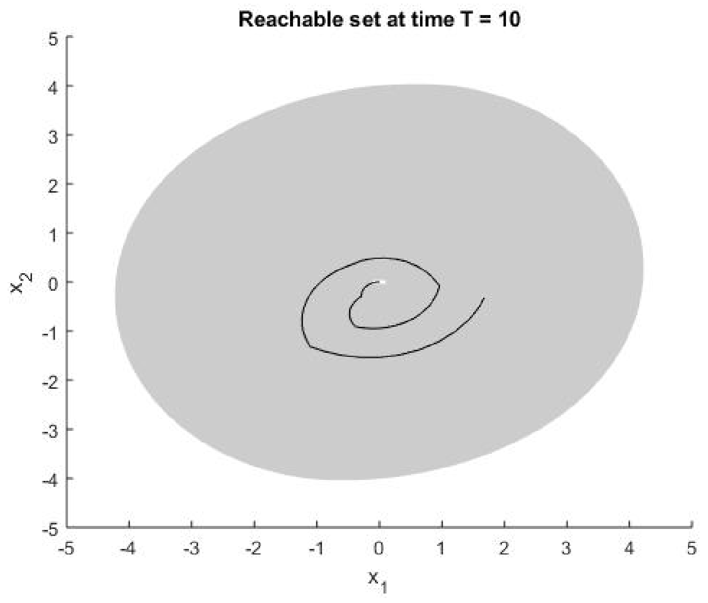

Fig. 4.2. The reachable set at time 10 seconds plotted using CORA. System parameters are $\sigma = -0.1$ and $\omega = 1$

.

The use of reachable sets appears promising, but other methods of calculating the RSS will also be explored. An important distinction between the reachable set at time $t = T$ and the RSS is the boundary of each set. At the boundary of the RSS, all state derivatives will point inside the space because, by definition, no point on the edge can be driven to a state that is outside the RSS. At time $t = T$, however, there may be reachable states not yet traversed, and therefore the state derivatives at the boundary of the reachable set at time $t = T$ may point inside or outside the set. Only when all derivatives along the boundary point inside the space has the RSS been resolved. Theoretically, an algorithm could be designed that locates boundaries where all derivatives of the system in (3.13) point inside the space enclosed by the boundary.

4.2 Use of CORA for MATLAB

The Continuous Reachability Analyzer (CORA) for MATLAB was employed in finding the RSS for (3.13). The toolbox, which was developed by Mathias Althoff and Niklas Kochdumper, has the capability to quickly produce reachable sets of many different systems with linear, nonlinear, and hybrid dynamics. CORA's algorithm for computing reachable sets of nonlinear systems is summarized in this section.

CORA defines the reachable set at time $t = \tau$ to be the set that contains all trajectories that originate from $x_0 \in X_0$ and are driven by uncertain inputs $u(t) \in U(t)$ [6].

$$\mathcal{R}(\tau) = \{X(\tau; x_0, u(.)) | x_0 \in X_0, \forall t : u(t) \in U(t)\} \tag{4.3}$$

The objective of CORA's algorithm is to obtain the reachable set at time $t = k\tau$ for every time step k so that the reachable set from time $t = t_0$ to $t = T$ can be expressed, which is defined as the union of all reachable sets within the time horizon.

$$\mathcal{R}([t_0, T]) = \bigcup_{[t_0, T]} \mathcal{R}(t) \tag{4.4}$$

Althoff and many other contributors to Reachability Analysis have elected to use zonotopes to represent sets. This is in part because zonotopes are closed under linear transformations, which means reachable sets of linear systems can be calculated exactly. Additionally, zonotopes afford computational efficiency compared to polytopes, ellipses, or other set representations [3]. A zonotope $Z \subset \Re^{n_z}$ is defined as the finite Minkowski sum of line segements [5].

$$Z = z^0 + \sum_{i=1}^{h} \alpha_i z^i : \alpha_i \in [-1, 1], \forall i \tag{4.5}$$

where $z^0 \subset \Re^{n_z}$ is its center, $z^i \subset \Re^{n_z}$ are its generators, and h is the number of generators. For example, if the initial condition set were a square centered on $[1, 2]^T$ with side length equal to 2, the corresponding zonotope would be

$$X(t_0) = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \right\} \tag{4.6}$$

The input, if modeled as the line-segment between [-1,1], would be expressed as

$$U = \{0, (\ 1)\} \tag{4.7}$$

A graphical description of a zonotope is displayed in Figure 4.3.

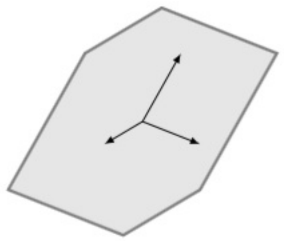

Fig. 4.3. A zonotope with h = 3 generating vectors. The Minkowski sum is essentially all possible linear combinations of the generators, thus spanning the shaded region.

CORA's algorithm for nonlinear reachable sets takes an abstraction of the system's dynamics at every timestep. Since this abstraction is based on conservative linearization, zonotopes are used in nonlinear systems as well [3]. The flow of the nonlinear algorithm is described in Figure 4.4. Once the system is linearized, the reachable set at time $t = k\tau$ is calculated without the set of error due to linearization considered. The reachable set due to linearization errors is managed separately in order to ensure that the reachable set at $t = (k+1)\tau$ is an over-approximation.

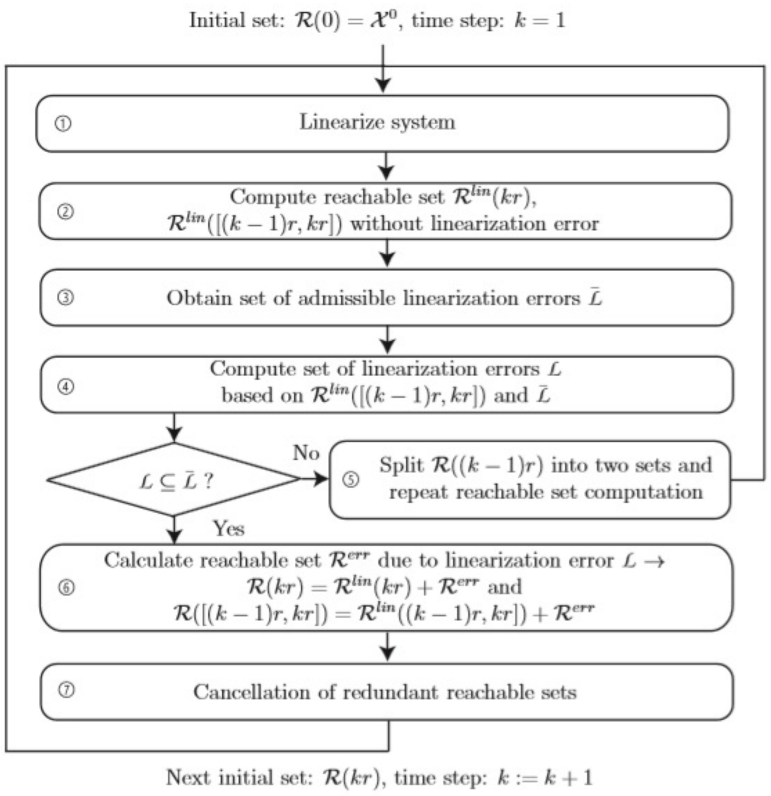
Initial set: $\mathcal{R}(0) = \mathcal{X}^0$, time step: $k = 1$
① Linearize system
② Compute reachable set $\mathcal{R}^{lin}(kr)$, $\mathcal{R}^{lin}([(k-1)r, kr])$ without linearization error
③ Obtain set of admissible linearization errors $\bar{L}$
④ Compute set of linearization errors L based on $\mathcal{R}^{lin}([(k-1)r, kr])$ and $\bar{L}$
$L \subseteq \bar{L}$?
No
⑤ Split $\mathcal{R}((k-1)r)$ into two sets and repeat reachable set computation
Yes
⑥ Calculate reachable set $\mathcal{R}^{err}$ due to linearization error $L \rightarrow$ $\mathcal{R}(kr) = \mathcal{R}^{lin}(kr) + \mathcal{R}^{err}$ and $\mathcal{R}([(k-1)r, kr]) = \mathcal{R}^{lin}((k-1)r, kr]) + \mathcal{R}^{err}$
⑦ Cancellation of redundant reachable sets
Next initial set: $\mathcal{R}(kr)$, time step: $k := k+1$

Fig. 4.4. Flow chart depicting CORA's Nonlinear Reachable Set calculator.

One important feature of the nonlinear reachable set algorithm is if the *allowable* set due to errors does not enclose the *calculated* set due to errors, then the allowable set due to errors must be enlarged to preserve an over-approximation. If the enlargement is unsuccessful, the previous reachable set must be split into partial reachable sets before proceeding. The act of splitting the previous reachable set is done in effort to reduce the size of the calculated set due to linearization errors. Finally, the reachable set at time $t = k\tau$ is taken to be

$$\mathcal{R}(k\tau) = \mathcal{R}^{lin}(k\tau) + \mathcal{R}^{err}(k\tau) \tag{4.8}$$

As an artifact of splitting sets into partial sets, the computational burden is compounded, and the algorithm may never converge to a solution. This shortcoming of CORA becomes increasingly evident when the toolbox is applied to (3.13). Evidently, the algorithm is unable to converge to a solution when the input to (3.13) is uncertain and within $[-1, 1]$. There might be several subtleties in the system that prevent convergence. For example, instabilities in the numerical solution to (3.13) cause the wrapping effect, which is documented in [4] and investigated extensively in [3]. The wrapping effect is the name given to uncontrolled set-expansion due to the propogation of recurring over-approximations. In the context of the algorithm described in the previous section, reachable sets split indefinitely because the set of linearization errors becomes increasingly large as the algorithm proceeds through many timesteps. This expansion effect can be observed in Figure 4.5. A solution that has converged properly can be seen in Figure 4.6.

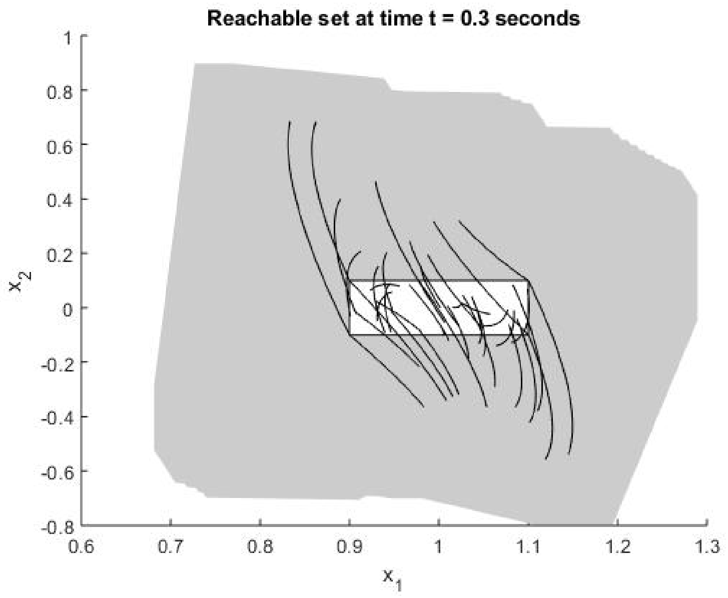

Fig. 4.5. Reachable set at time t = 0.3 seconds for the system in (3.13). For times $t > 0.3$, convergence was not achieved. The accuracy of the solution is obtained by inspection. Since the grey sets do not tightly enclose the simulated trajectories, the accuracy of this solution was determined to be less than satisfactory.

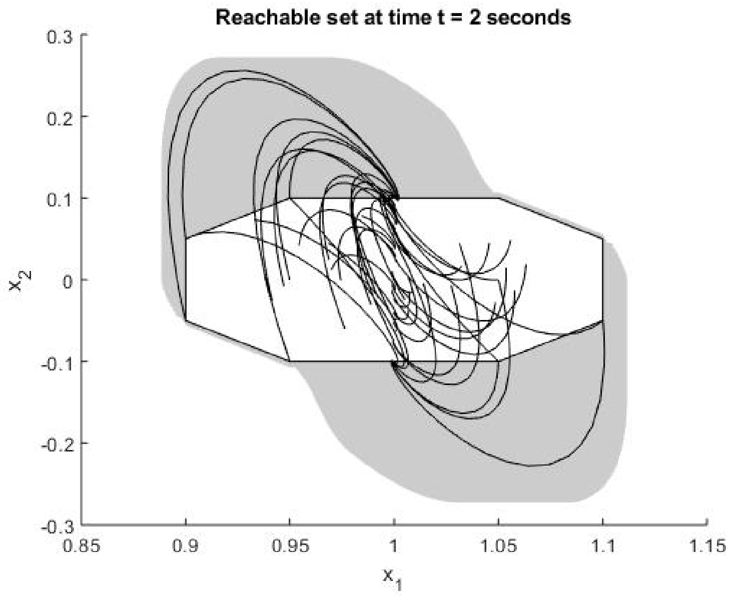

Fig. 4.6. Reachable set at time $t = 2$ seconds for the system in (3.13). Convergence was achieved when the input was allowed to vary within $[-0.1, 0.1]$ as opposed to $[-1, 1]$. In contrast with Figure 4.5, the grey sets tightly enclose the simulated trajectories, indicating that this reachable set is accurate.

If the system in (3.13) is being described very rigorously, it is in fact a hybrid system as opposed to nonlinear. This is due to the presence of hard limits on x_2, which causes the dynamics to change when $x_2 > 1.5$ or $x_2 < -1.5$. Essentially, the nonlinear reachable set approach will always be invalid because it could never accommodate the hard limits. Furthermore, this classification greatly increases the modeling difficulty in CORA because the hybrid systems toolbox requires a certain degree of expertise and experience to use correctly. The high entry-level curve is not exclusive to the hybrid systems toolbox. Using CORA in any capacity generally requires a profound understanding of reachability analysis, set theory, and numerical analysis. Since so much of the results obtained through CORA are dependent on setting up the algorithm parameters correctly, it is unlikely that a novice user will be able to easily obtain good results.

4.3 Discrete Space Search

The objective of the discrete space search algorithm is to locate the boundary of the RSS by using trial simulations to traverse the entirety of the space. If every reachable state can be located with trajectories, then the boundary is readily constructed by inspection. The difficulty lies in freely modulating the input $u(t)$ such that all states within the RSS are visited. Although there always exists an arbitrary function $u(t)$ that drives a trajectory to the boundary of the RSS [4], these exact input functions are not known a priori. However, by discretizing a search space and simulating trajectories recursively, it is possible to extract the RSS boundary without knowledge of the exact inputs. It is shown in Figure 4.7 how the boundary is deduced from the sample of points in the RSS.

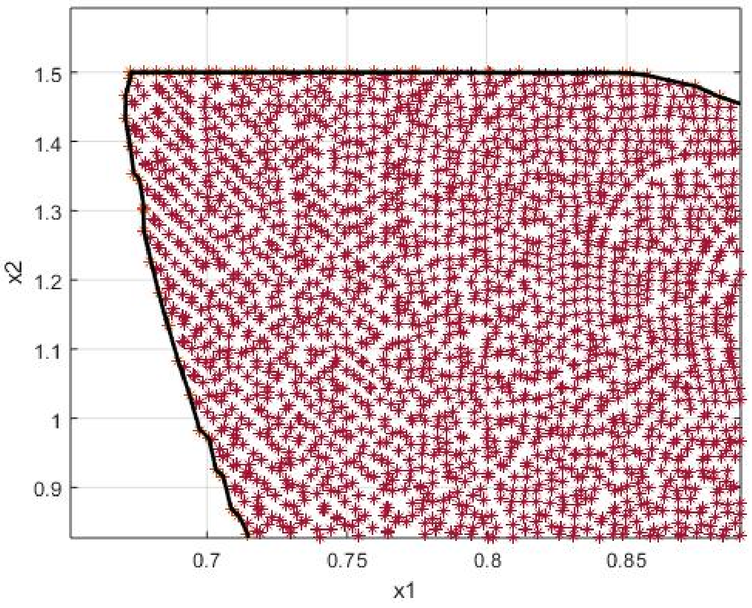

Fig. 4.7. A sampling of the RSS for the system in (3.13). Stars indicate points that have been reached by trajectories while the boundary is the black line connecting the outermost points. The boundary was constructed using MATLAB's *boundary* function

.

The typical simulation strategy might consist of repeated trials, each with a different input sequence. The discrete space search algorithm is different in that it is recursive. It simulates a single trajectory that starts within the RSSE with $u = 1$. While the trajectory is updated with the forward Euler method, the algorithm checks if each updated state has been visited by a previously simulated trajectory. If it has not, the trajectory is simulated recursively, this time with both $u = 1$ and $u = -1$. The act of calling the simulation recursively splits the trajectory into two paths, one corresponding to each input u. When these new trajectories reach untraversed states, they will split off into four more paths, and so on until no new paths can be found. This procedure allows $u(t)$ to be modulated in such a way that the simulated trajectory will always be provided the necessary input to travel along a path that has not been traversed. Eventually, no simulated trajectory will find a new state and the algorithm is henceforth allowed to terminate.

One subtlety in this input modulation strategy is that $u(t)$ is not continuous on $[-1, 1]$, but is switched between the two extremities. This raises the concern that there are trajectory paths corresponding to $u \in (-1, 1)$ not explored by the algorithm. As a counterargument, it is useful to recall that any derivative function $f(x, u)$ with $u \in (-1, 1)$ can be represented as a linear combination of $f(x, 1)$ and $f(x, -1)$. In other words, there always exists a and b such that

$$af(x, 1) + bf(x, -1) = f(x, u), \qquad u \in (-1, 1) \tag{4.9}$$

Graphically, (4.9) implies that any derivative function corresponding to $u \in (-1, 1)$ will "point" between the derivatives corresponding to $u = 1$ and $u = -1$. This, in turn, implies that any trajectory with $u \in (-1, 1)$ will be enclosed by trajectories with $u = 1$ and $u = -1$. Thus, all allowable inputs are accounted for in this modulation scheme.

Since it is unlikely that a single point of interest will be ever be traversed more than once, the algorithm lumps nearby points into a rectangular domain and checks if the domain has been traversed. In other words, when the simulation is updated, the new point exists in a predetermined domain. If any point in this domain has been previously traversed, the trajectory is allowed to proceed, and the function is *not* called recursively. In practice, each domain has a corresponding binary variable. If the domain has not yet been traversed, its value is '0'. Otherwise, the value of the binary variable is '1'. It is illustrated in Figure 4.8 how the logical assignment of the domains maps to a sample point of the RSS.

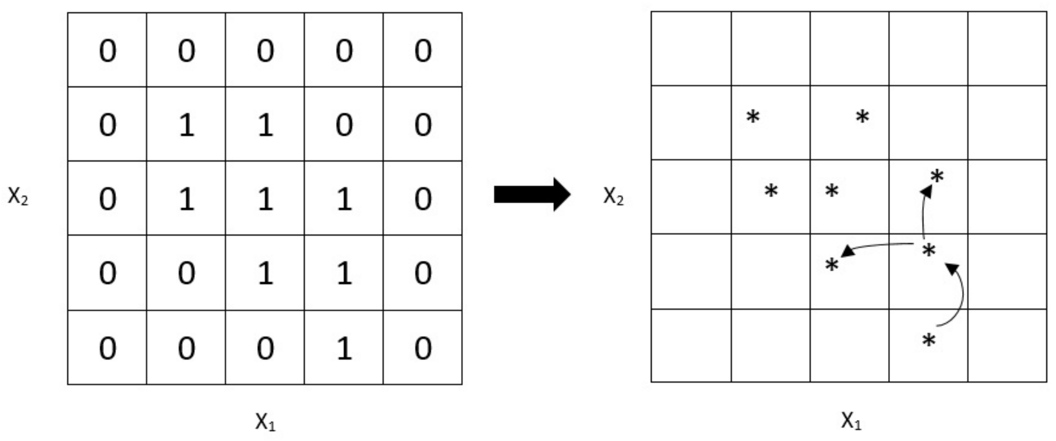

Fig. 4.8. State-space domains with their assigned logicals (left). When a simulated trajectory enters a domain with logical '0', its logical is updated to '1'. The trajectory point is then added to the sample of the RSS in its corresponding domain (right). The arrows represent trajectories taken to reach each domain.

This method of checking requires that the search space be discretized, hence the name "discrete space search". Properties of the system of interest may be exploited to choose the search space. For example, the hard limits on x_2 in (3.13) made it a natural choice to define the search space with bounds $[-1.5, 1.5]$ with respect to x_2. Furthermore, it is often desirable that the discrete domains fit evenly into the space. In this application, the search space was chosen to be rectangular with rectangular domains for simplicity. It would possible, for example, to define an arbitrarily shaped search space where the domains are the Delaunay Triangulation of the space, but such spaces are not considered in this application. The choice of search space is generally trivial, but it must completely enclose the RSS.

The entire procedure of the discrete space search algorithm can be summarized by the following pseudo-code:

```
Main
    $rss \Leftarrow \text{zeros}(N)$
    $x \Leftarrow x_0$ where $0 = f(x_0, 0)$
    call $Recursive(x, 1)$
    call $Recursive(x, -1)$


Recursive$(x, u)$
    calculate $x_e$ where $0 = f(x_e, u)$
    $x_n \Leftarrow x$
    while $\|x_n - x_e\| < \varepsilon$
        $x_n \Leftarrow x_n + hf(x_n, u)$
        $i, j \Leftarrow quantize(x_n)$
        if $rss(i, j) = 0$
            $rss(i, j) \Leftarrow 1$
            $v_{dc}(count) \Leftarrow x_n(1)$
            $i_{dc}(count) \Leftarrow x_n(2)$
            $count \Leftarrow count + 1$
            call $Recursive(x_n, 1)$
            call $Recursive(x_n, -1)$
        end if
    end while
```

As seen in *Main*, the algorithm is initialized with an arbitrary equilibrium state and an NxN array of logical zeros. *Recursive* is immediately called twice with one input extreme per call. Within *Recursive*, the new equilibrium point corresponding to the chosen input is calculated, and the trajectory is simulated from x until convergence with x_e is reached. While the trajectory is simulated, *quantize* locates the discrete domain associated with x_n. If the logical associated with domain *i,j* is 0, the domain has not been visited. When this is the case, the logical is updated to 1, and each component of the state is stored. Finally, and the function is called recursively, splitting into two trajectories. When the algorithm terminates, the stored states are used to construct the RSS boundary.

There are several limitations that are inherent to the architecture of the discrete space search algorithm. As a consequence of using discrete domains, it is extremely difficult to produce a smooth, high resolution boundary. It is useful to consider the example in Figure 4.9. If the true boundary of the RSS is considered to be the black line, the true boundary cannot be accurately represented by one state per domain. Were the domain area made very small, the resolution of the boundary would increase but at great expense to the speed of the algorithm and memory of the computer. At best, the discrete space search will always yield a very crude approximation of the RSS boundary.

Another consequence of using discrete domains is that the algorithm will always yield an under-approximation of the true boundary. This is because the algorithm does not search for states within the furthest reachable domain. It is useful to consider a trajectory that has reached the furthest reachable domain. While it may be true that no reachable domains exist beyond its current domain, reachable states could exist within its current domain beyond its current state. Since these reachable states are never searched for, the discrete space search will always result in a slight under-approximation of the RSS boundary.

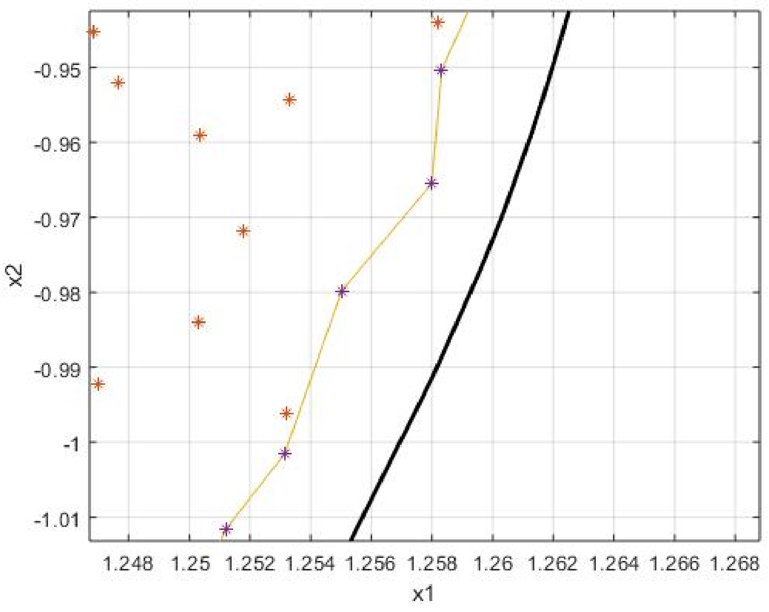

Fig. 4.9. The boundary of the RSS as computed by the discrete space search algorithm (jagged, light line) and the true boundary of the RSS (smooth, dark line).

.

Since the algorithm is both recursive and simulation based, more limitations and concerns arise. All of the numerical stability and global error considerations associated with the forward Euler method also apply to the algorithm. Due to the algorithms recursive nature, memory consumption is also of great concern. This is especially true in the case of higher-dimensional systems. If rectangular domains are used, the number of domains grows according to N^d, where N is the number of domains per dimension, and d is the dimension of the system. Increasing the number of domains drastically increases the number of recursive function calls, which has the potential to exhaust memory resources. Furthermore, in tools such as MATLAB, the number of recursions that is allowed to take place is limited, thus limiting either the size or resolution of the search space.

4.4 Advancing Fronts: An Algorithm for Approximating the Reachable State-Space of a Nonlinear System

The objective of the advancing front algorithm is to locate the set of points which, when treated as the boundary of a closed region in state-space, cannot be driven outside of the enclosed region for all allowable inputs to the system. By definition, this set of points constitutes the boundary of the system's RSS. The numerically calculated RSS then, will consist of its boundary and all points enclosed by its boundary.

As a visualization, it is useful to consider a small arc length on the edge of an arbitrary and convex region such as that of Figure 4.10. At each point along the arc there is an associated vector of state derivatives determined by the system dynamics.

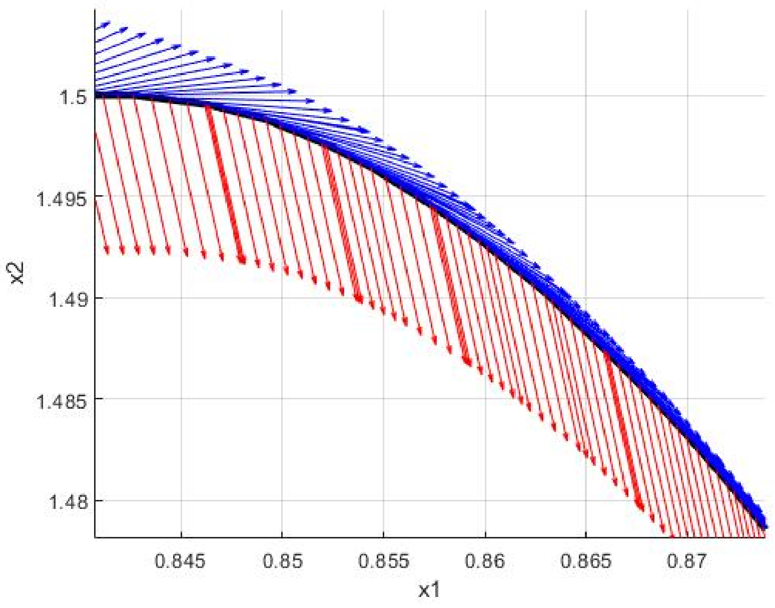

Fig. 4.10. State derivatives for $u = 1$ (blue) and $u = -1$ (red) are evaluated at points along the black border. Points near the bottom right are approximately at the edge of the RSS, while points near the top left indicate that there are reachable states above the enclosed region

If this example region is an arbitrary set of reachable states, then points outside the region are not reachable states if and only if all derivatives along the arc point inside, or are tangent to, the region. Hence, the algorithm is satisfied if and only if, at every point along the boundary, there is no allowable input u who's corresponding derivative points outside the region. If a state derivative vector points outside the region, this would imply that the region needs to be *expanded* to include more reachable states. In practice, the set of interest is not arbitrary, but is the set of initial conditions. Indeed, the intention is to expand the set of initial conditions into the RSS.

When there is evidence that there are reachable states outside of the arc length, or 'front', of interest, an expansion should be made to include the remaining reachable states. The front, therefore, is not fixed but rather elastic in nature. Each point along the front is allowed to move outward from the enclosed region subject to the underlying vector field of state derivatives. The procedure of properly expanding these fronts into the true RSS boundary is as follows:

- Between every node, determine if an expansion is possible.
- If an expansion is possible, perform an expansion
- Wrap the new set of nodes in a boundary
- Increase the resolution of the boundary

In order to properly articulate the inner-workings of each step in the advancing fronts algorithm, it must first be defined what it means for the state derivative vectors to point *inside* or *outside* of the space. Additionally, it is important to define what is considered to be the numerically calculated boundary, or *edge*, of a two-dimensional region. In this context, the edge of a space is considered to be a sampling of significant points called nodes and all points along the straight-line segments that connect the nodes. Together, the nodes and segments form a closed path that binds a space. A graphical explanation of approximating the edge of a space as nodes and segments is given in Figure 4.11. The surface normal then, is easily defined as a vector that is perpendicular to the surface at any point along one of its straight-line segments, not at the nodes. Based on this description, all points along a node-connecting segment have the same normal vector. It is difficult to define the normal at the nodes, but the algorithm can succeed without this definition.

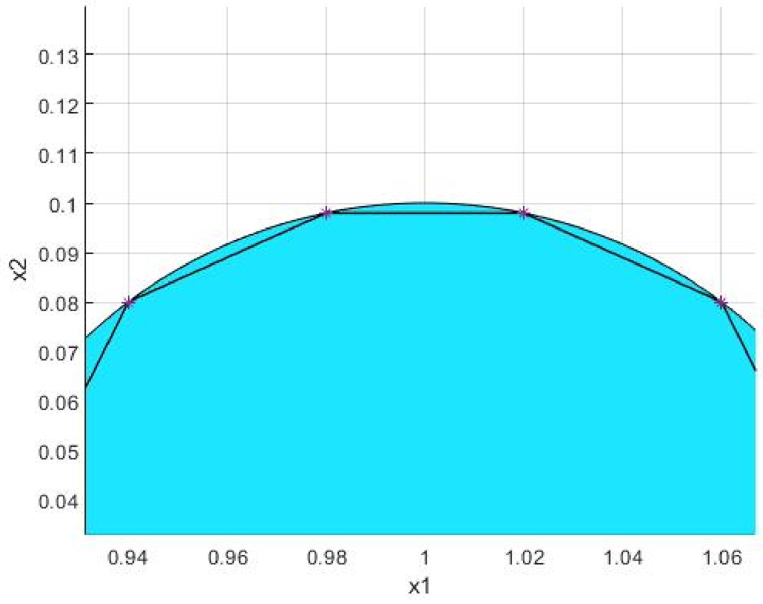

Fig. 4.11. A perfect circle (shaded) is crudely approximated by straight line-segments and nodes. The *edge* of a two-dimensional surface will be defined as a sampling of points (nodes) and straight line segments between them. In practice, the resolution of the approximation will be much higher.

Based on these definitions of edge and surface normal, *inside* and *outside* can be defined. Let the angle formed between a state derivative vector and the surface normal be called θ. If θ is less than 90 degrees, the state derivative points outside the space. If θ is greater than 90 degrees, the state derivative points inside the space. The concepts of the inside and outside regions of the boundary are illustrated in detail in Figure 4.12. Challenges with these definitions occur when the space is non-convex. For example, a state derivative might satisfy this definition of *outside* at an arc length

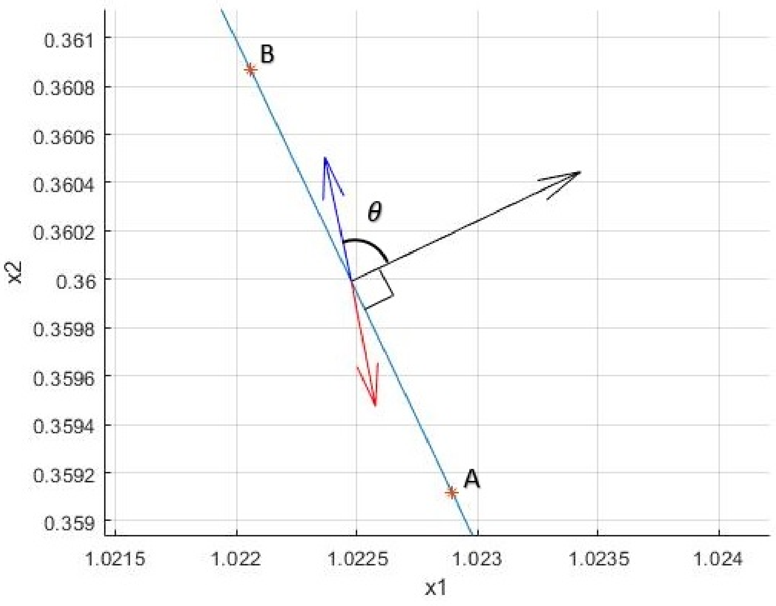

Fig. 4.12. The surface normal (black), and state derivatives for 2 different allowable inputs are evaluated at the midpoint of segment $\overrightarrow{AB}$. θ is the angle formed between a state derivative and the normal. The derivative corresponding to the blue arrow points outside the region ($\theta < 90$), while the derivative corresponding to the red arrow points inside the space ($\theta \geq 90$).

of interest, but if a ray were to extended pointing in the same direction, the ray might intersect the space again at some point, thus suggesting that the derivative actually points *inside*. The same situation occurs if there are small divots in the front. The solution to this challenge is implemented in the "wrapping" step of the algorithm. Therein, accommodations are made to preserve the definitions of *inside* and *outside*.

The first step for advancing fronts is to determine the expandability of the edge between any two nodes. As previously discussed, reachable states exist outside the space if there are state derivatives at the edge that point outside the space. Thus, the state derivatives and normals are evaluated at the midpoints between nodes. The expansion cannot take place at the nodes because the normal need be defined, and the derivatives are roughly equal across all points along each segment, so the midpoints were a suitable location choice to evaluate these quantities. Obtaining the normal between nodes ‘A’ and ‘B’ is as simple as taking $B - A$ and rotating that vector by 90 degrees. Note that this definition implies that the length of the normal is dependent on the length of $\overrightarrow{AB}$. If θ corresponding to any allowable input is less than 90 degrees, this implies an expansion can be made, so the node is marked as advanceable. Otherwise, the node remains fixed at its position. Referring back to Figure 4.12, this particular front can be advanced from the midpoint between nodes ‘A’ and ‘B’ since the derivative for $u = 1$ points outside the space.

Each midpoint deemed advanceable by the algorithm is assigned a proposed expansion which is independent from all other proposed expansions. The nature of a proposed expansion can be described in terms of magnitude and direction. The direction is indicated by the normal vector, as this is the vector that points outward from the enclosed region at every point. Its magnitude is determined by the projection of the state derivative onto the surface normal, $\mathbf{n}$.

$$\operatorname{proj}_{\mathbf{n}} \mathbf{f}(\mathbf{x}, \mathbf{u}) = \frac{\mathbf{n}^T \mathbf{f(x,u)}}{|\mathbf{n}|^2} \mathbf{n} \tag{4.10}$$

Which is equivalent to

$$\operatorname{proj}_{\mathbf{n}} \mathbf{f}(\mathbf{x}, \mathbf{u}) = \frac{|\mathbf{n}||\mathbf{f(x,u)}| \cos\theta}{|\mathbf{n}|^2} \mathbf{n} \tag{4.11}$$

Applying the definition of the unit normal to (4.11), it can be shown that there is no dependence on the length of the normal.

$$\operatorname{proj}_{\hat{\mathbf{n}}} \mathbf{f}(\mathbf{x}, \mathbf{u}) = |\mathbf{f(x,u)}| \cos\theta \hat{n} \tag{4.12}$$

This is desirable because the length of each normal is dependent on the length of each segment. Since each segment will have a different length, each expansion would be dependent on each segment length. Thus, (4.12) will be used to define the projection.

Since there are multiple allowable inputs, there are multiple state derivatives that can be chosen to project. The one selected for projection is the derivative who's θ is less than 90 degrees, and who's projection is the largest compared to all other inputs.

The equation that governs the expansion proposal is given in (4.13).

$$\mathrm{RSS}_{i,k+1} = \mathrm{RSS}_{i,k} + \mathrm{proj}_{\hat{\mathbf{n}}}\, \mathbf{f}(\mathbf{x}, \mathbf{u}) * \delta \tag{4.13}$$

Here, $\mathrm{RSS}_{i,k}$ is the i'th midpoint after the k'th expansion, $\hat{n}$ is the unit normal vector, $\mathbf{f(x,u)}$ is the state derivative, and δ is analogous to Δt in the context of a forward-euler iterator. The presense of δ in (4.13) allows the rate of expansion to be controlled, ultimately giving the user the freedom to increase or decrease the speed and precision of the algorithm. An example of a proposed advancement is shown in Figure 4.13.

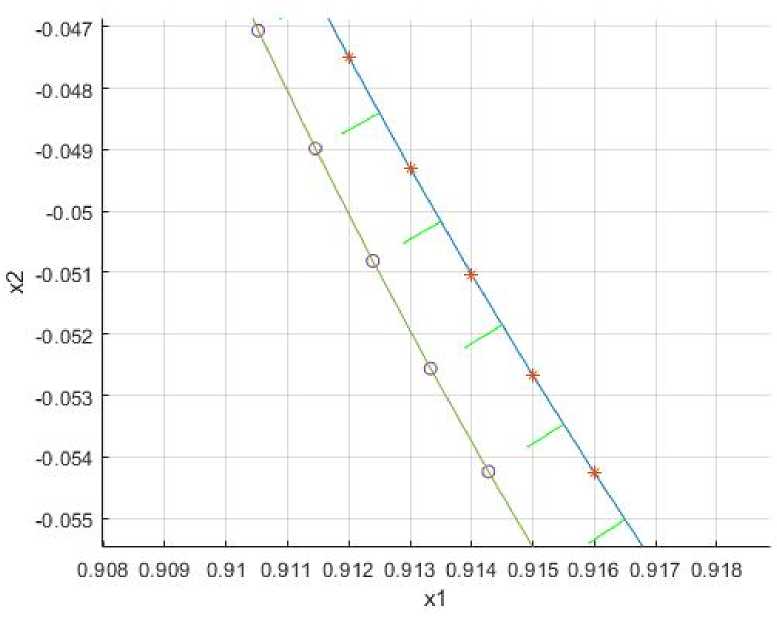

Fig. 4.13. An example of a proposed advancement. The starred nodes represent RSS_k, while the circled nodes represent RSS_{k+1}. The green arrows represent the surface normals at each midpoint, indicating the placement of RSS_{k+1} relative to RSS_k

It is important to define $\mathrm{RSS}_{i,k+1}$ as the i'th *node* after the present iteration. According to this definition, (4.13) takes the midpoints of the current iteration, advances them outward, and takes the expanded midpoints to be the nodes of the next iteration. Once an expansion proposal has been made for every node, the proposals are either approved or rejected by an expansion limiter. Once the k'th expansion has been implemented, the new set of nodes is wrapped in a boundary.

The primary motivation for "wrapping" each new set of nodes in a boundary is to preserve the integrity of the outward expansion direction. Throughout the advancement procedure, neighboring nodes will not expand at the same rate, giving rise to imperfections and divots along the edge of the space. These cause performance issues with the algorithm because the edge will not proceed in the correct direction. One solution to this problem is to "wrap" each iteration with a boundary of predetermined tightness using the MATLAB function, *boundary*.

Boundary accepts an array of points and returns a single conforming border that is guaranteed to contain every point that in the array. This is noticeably similar to the concept of taking the convex hull of a set of points. The difference is that MATLAB's *boundary* also accepts an argument between 0 and 1 which corresponds to the desired tightness of the boundary. A very tight boundary is allowed to shrink towards the interior of the hull, whereas the convex hull is a strictly convex set. This gives rise to another motivation to wrap the nodes, which is to maintain convexity throughout the expansion process. If it is known in advance that the RSS of a system is convex, the user can always loosely wrap the nodes using *boundary*, or by using MATLAB's *convhull*. It is illustrated in Figure 4.14 how the presence of a divot along the edge leads to improper expansion and how wrapping can influence the edge formation. The advantage of processing the edge of the RSS at each iteration is that it eliminates the possibility that three neighboring nodes form a very large interior angle. Therefore, the wrapped set can be guaranteed to expand outward upon the next iteration.

As the edge of the RSS expands, the line-segments between nodes will increase in length. The result is an increasingly crude approximation of the edge of the space. This effect is further exacerbated by calling MATLAB's *boundary* function because the resultant boundary eliminates nodes, as seen in Figure 4.14. In other words, the

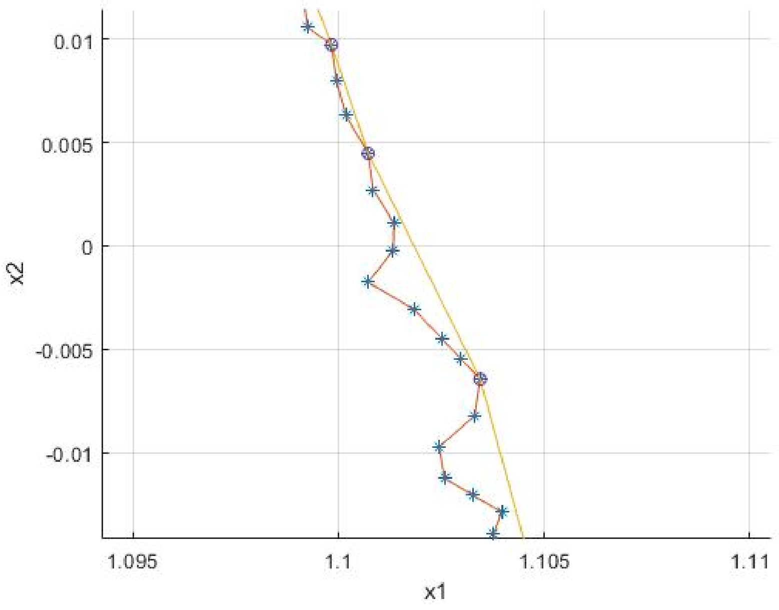

Fig. 4.14. The unwrapped front (starred nodes) expanding in the positive x1 direction. Segments on the interior of the divot are subject to expand inside the space because of the surface normal directions. Applying MATLAB's *boundary* eliminates these features, thus preserving the outward expansion direction (circled nodes). Additionally, it can be seen that applying *boundary* reduces the total number of nodes used to approximate the edge of the RSS.

resolution of the boundary decreases throughout the duration of the algorithm. The average resolution can be loosely defined as

$$resolution = \frac{l_{\text{boundary}}}{\bar{l}_{\text{segment}}} \tag{4.14}$$

where l_{boundary} is the length of the boundary and $\bar{l}_{\text{segment}}$ is the average segment length. Not only is a crude boundary undesirable, but if the resolution becomes too low, the low-resolution boundary becomes subject to a kind of "traveling" effect. When this happens, the collection of nodes migrates with respect to its center, leading to an extremely inaccurate result. This effect can be seen explicitly in Figure 4.15.

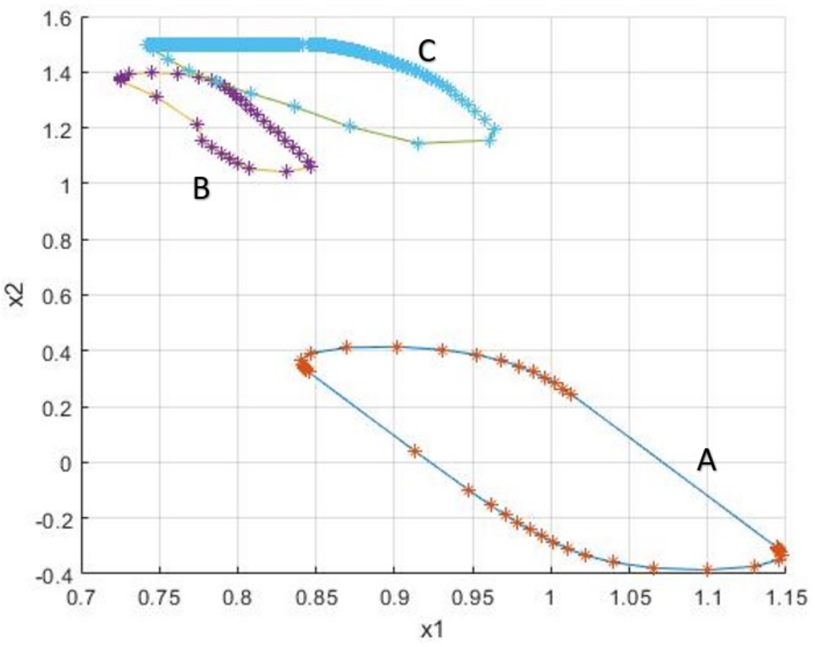

Fig. 4.15. The RSS boundary after 200 advancements (labelled 'A'), 900 global advancements ('B'), and 1500 advancements ('C'). Rather than expanding outward to articulate the boundary, the low-resolution surface changes position relative to its center.

Accordingly, the resolution of the boundary needs to be increased at every iteration to mitigate these two problems. There are many ways to increase the resolution of the boundary, but perhaps the simplest method was determined to yield satisfactory performance. The strategy is as follows: If the length between any two adjacent nodes is too large, insert a new node at the midpoint between them. What length is considered to be too large is ultimately up to the user. In this application, the threshold segment length was chosen to be much smaller than the average segment length of the initial condition set. This is because the final resolution is expected to be much larger than that of the initial condition set.

Now, it is not expected that the strategy will always preserve a desired resolution. For example, if a segment length at the end of an advancement is 5 times the threshold segment length, the strategy will result in 2 segment lengths that are each 2.5 times the threshold segment length. However, in choosing a small enough δ, the total number of iterations required for convergence increases, thus increasing the number of times that the resolution is increased. As long as δ is sufficiently small, this strategy yields a smooth, high-resolution boundary.

When the front is allowed to advance freely outward, the potential for the front to converge to an inaccurate boundary still exists. While most of the front will converge accurately, subtle numerical features in (4.13) permit the front to expand outside the true RSS boundary. If this occurs, the correct edge can never be resolved because the algorithm inhibits reverse expansion. As an example, it is useful to consider the generalized case which has been illustrated in Figure 4.16. Therein, an advancing front is incident upon the true RSS boundary.

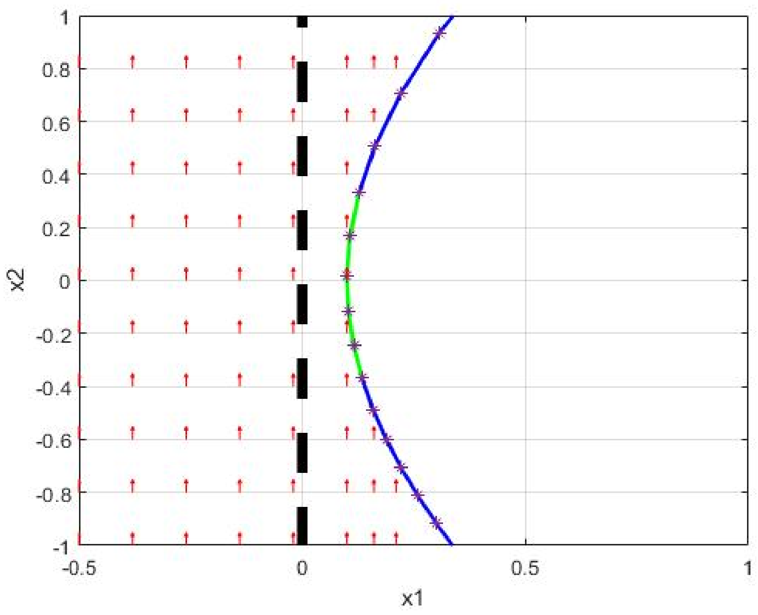

Fig. 4.16. The correct edge of the RSS (dashed black line) and an advancing front. Nodes along the green arc will experience a slowed expansion while nodes along the blue arcs will experience a more rapid expansion.

Advancing fronts searches for boundaries where the state derivatives are tangent to the boundary everywhere. In the example in Figure 4.16, all derivatives along the dashed black line point strictly in the x_2 direction, so the dashed black line is a "barrier" to a moving front. As the front converges to the true solution, the derivatives approach tangency to the front, so the projections of the derivatives onto the surface normal are decreasing with each iteration. Since the expansion is governed by the projections, every midpoint along the front will expand at a different rate because every segment lies at a different angle to the underlying derivatives. The front will experience a slowed, convergent expansion for an arc length that is nearly parallel to the barrier. In contrast, the front will experience a rapid expansion for an arc length that lies at an angle to the barrier. For nodes located at the junction of the green and blue arcs, it is possible for a single node to "jump" the barrier in one iteration of (4.13). The result is a cascading effect that allows the entire front to expand outside the true boundary. The significant disparity in expansion rates between neighboring nodes is the root of the inaccurate convergence or non-convergence problem.

As a way to address this issue, an expansion limiter is employed. The expansion limiter accepts proposals from (4.13) and checks for uneven expansion rates. That is to say, the proposed expansion distance at any node is significantly larger than that of its neighboring nodes. When an uneven expansion is detected, the node is frozen in place, along with the next 2 nodes in sequence and the previous 2 nodes in sequence. In this application, it was determined that if a proposed expansion is 100 times greater than that of a neighboring node, the expansion limiter should replace that expansion with a zero.

With the expansion limiter in place, the advancing front never passes the true edge of the RSS. Upon convergence to the correct RSS boundary, the expansion limiter will actually inhibit the final iterations required to satisfy the stop-condition. During the final iterations, the absolute expansion distance proposed for each node will be a sparse column. Many nodes have reached a satisfactory place, while few others may need to advance an extremely small distance. As an artifact of expansion limiter's functionality, the final expansions will be removed from the column, freezing the advancing front in place everywhere. To remedy this, a comparison is made from the average proposed distance to the average actual distance. Upon convergence, the average proposed distance will be significantly higher than the average actual distance because the distance limiter replaces many expansion proposals with zeros. It was determined that when the mean of the proposed distances is greater than 50 times the mean of the actual distances, the expansion limiter is shut off so that the front can proceed to its final destination.

When the underlying vector field of system dynamics is considered, it is possible to draw several boundaries that all satisfy the algorithm's stop-condition. That is, all derivative vectors along the edge of the space point inside, or tangent to, the space. This leads to some ambiguity in where the true boundary lies. It is shown in Figure 4.17 how it is possible for the algorithm to converge to two different boundaries.

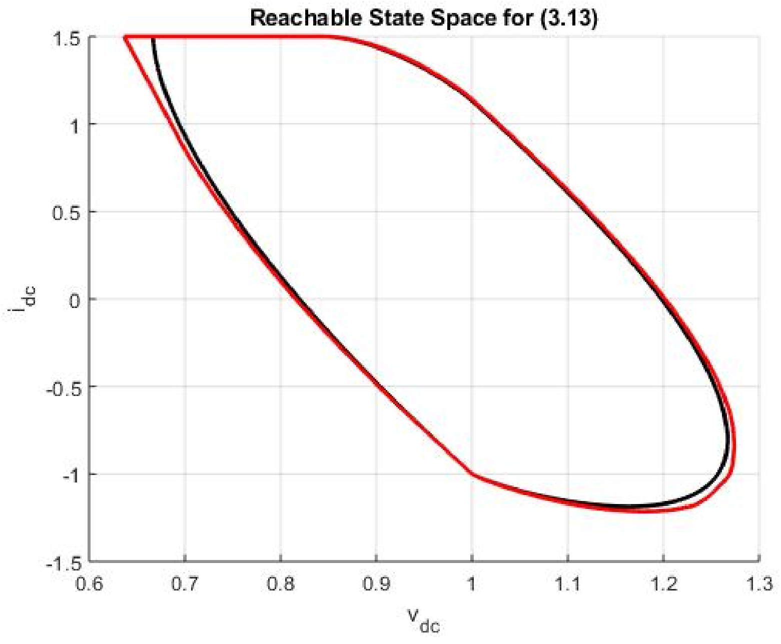

Fig. 4.17. The RSS for (3.13) as produced by Advancing Fronts with two different values for δ.

As a way of searching for the most accurate RSS, it is necessary to repeat the algorithm with different values of δ and different initial condition sets. Theoretically, all arbitrary initial condition sets that are wholly contained by the RSS should converge to the exact same RSS. Additionally, adjusting delta for many different trials will provide insight as to where there are subtle numerical instabilities in the space. As δ is made very small, the precision of the algorithm increases greatly, similar to how decreasing Δt decreases the error in the forward-euler method. In this application, the most accurate RSS boundary was taken to be the result that satisfies the stop-condition with the least total enclosed area. This was found by comparing the results with $\delta = 0.001$ and with $\delta = 0.0001$. Upon further decreasing δ, the algorithm converges to a unique solution.

4.5 Comparing the Performance of Advancing Fronts and the Discrete Space Search

Both the advancing fronts and the discrete space search yielded similar results, as seen in Figure 4.18. Parameters for the results in Figure 4.18 were $G = 2$, $\tau' = 0.2$, $i_{\text{lim}} = 1.5$, $v_{dc}^{*} = 1$, and $-1 \leq P_L \leq 1$. However subtle the differences may be, articulating them will ultimately result in improved performance, and therefore a more accurate LDSM. The boundary calculated by advancing fronts completely encloses the boundary calculated by the discrete space search. Some arc lengths are very tightly enclosed, while other lengths of the boundary differ by more than a full domain width. If the discrete space search is assumed to be working perfectly according to its design, then it will always locate the correct domains where the boundary lies. If advancing fronts is also assumed to be working perfectly, then for every domain that contains the boundary, it will locate the correct set of states within that domain. Based on these assumptions, it is anticipated that the two boundaries will differ by half a domain width on average. In theory, the largest possible separation between the two boundaries in a single rectangular domain would be equal to the length of the diagonal of the rectangle.

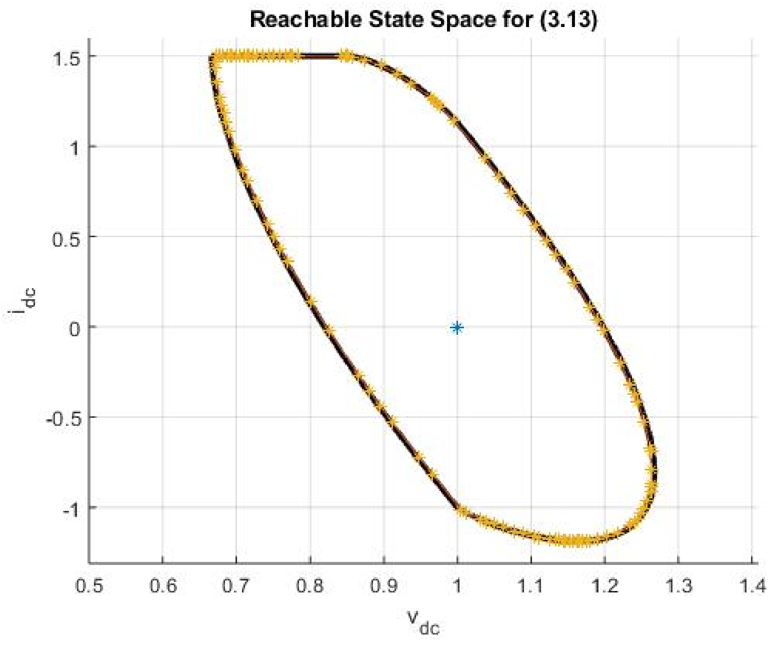

Fig. 4.18. The boundaries of the RSS of (3.13) as computed by the advancing fronts algorithm (solid line), and the discrete space search algorithm (starred line). There two methods very closely agree on the true solution.

The largest separation between the results of each algorithm is shown in Figure 4.19, which is a magnified vew of the top left corner of the boundaries shown in Figure 4.18. In the discrete space search, the dimensions of the rectangular domains are 0.0045 by 0.016 per unit, with respect to the v_{dc} and i_{dc} axes. The maximum separation was determined to be 0.007 per unit, purely in the direction of the v_{dc} axis. Since 0.007 per unit is more than the maximum theoretical separation in the v_{dc} direction, this indicates that the advancing fronts method is not working perfectly. The result from the advancing fronts method is indeed a slight over-approximation of the true RSS boundary, while the discrete space search is a slight under-approximation of the true RSS boundary.

As a way to confirm these speculations, derivative vectors can be placed on top of the boundaries given by each algorithm. As seen in Figure 4.19, states along the discrete space search boundary can be driven further outward because there are

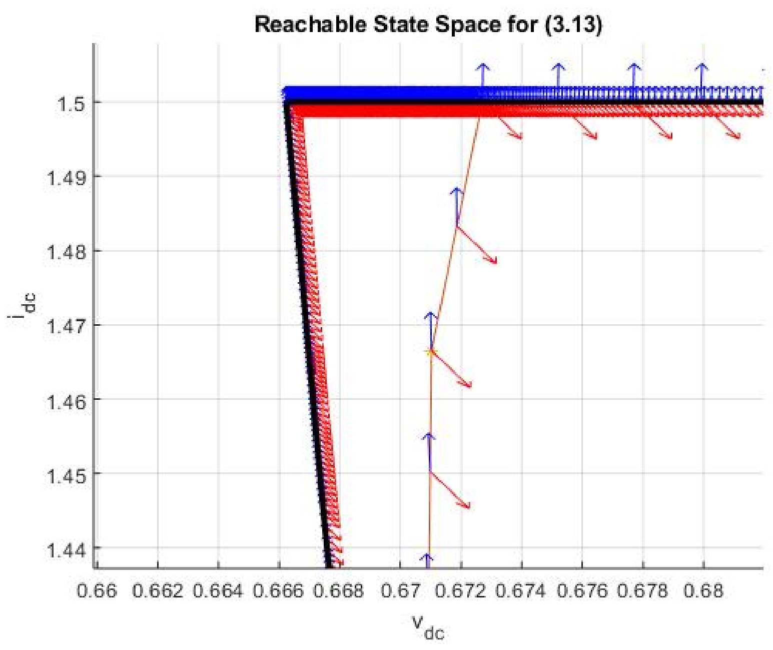

Fig. 4.19. State derivative arrows for $u = 1$ and $u = -1$ overlayed on the discrete space search boundary (right) and the advancing fronts boundary (left). Note that the arrows do not point inside the RSS where the boundary is at i_{lim}, but the limit implies that there are no reachable states above it.

derivatives that point outside the space. In contrast, when derivative vectors are placed on top of the boundary located by advancing fronts, not a single vector for either input extreme, $u = 1$ or $u = -1$, points outside the boundary. This is the single most significant piece of evidence that the solution obtained by the advancing fronts algorithm is more accurate than the discrete space search. In the corner pictured in Figure 4.19, arrows are not perfectly tangent to the boundary. The θ formed by these arrows is at most 91 degrees, which gives additional evidence that this boundary is a slight over-approximation in this local region.

There is an important subtlety in the argument for an accurate RSS boundary. As seen in Figure 4.19, only the state derivatives corresponding to the minimum and maximum inputs are evaluated for the stop-condition of the algorithm. If the derivatives corresponding to the input extremes both point inside the RSS, then no input exists within the allowable set that could drive a state outside the RSS. However, since the allowable input set is continuous between its extremes, it is necessary to prove that all inputs between $u = 1$ and $u = -1$ also point inside the RSS.

To prove this, it is essential to show that the set of derivatives at every point in state-space is a convex set. The allowable set of inputs is $U = [-1, 1] \in \Re$, which is a convex set by inspection. Since the derivatives in (3.13) vary linearly with respect to P_L, they are an affine mapping of U. An affine mapping of a convex set is itself a convex set. Thus, the set derivatives is a convex set. By definition of a convex set, elements of the set of derivatives are closed under convex combinations. Therefore, any derivative for $u \in (-1, 1)$ can be represented as the convex combination of $f(x, 1)$ and $f(x, -1)$.

$$\lambda f(x, 1) + (1 - \lambda) f(x, -1) = f(x, u), \qquad u \in (-1, 1), \quad \lambda \in [0, 1] \tag{4.15}$$

Note that the definition of a convex combination restricts values of λ to be strictly positive. In other words, only positive linear combinations of the derivatives corresponding to $u = 1$ and $u = -1$ can be taken. As a result, the derivatives for inputs $u \in (-1, 1)$ point between the derivatives for inputs $u = 1$ and $u = -1$. Figure 4.20 illustrates the derivatives evaluated at $u = 1$ and $u = -1$, as well as for several inputs between $u = 1$ and $u = -1$. The only way to form a vector from $f(x, 1)$ and $f(x, -1)$ that does not point between them would be to take a combination such that λ were negative, which is forbidden by (4.15). Thus, if the derivatives at the input extremes satisfy the stop-condition, then all derivatives in the allowable set of inputs do as well.

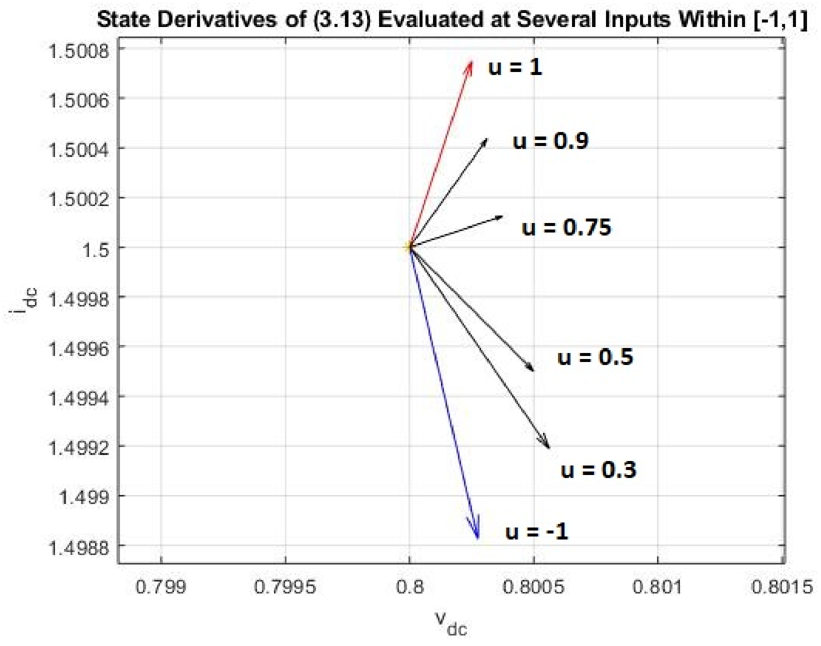

Fig. 4.20. Derivatives in (3.13) evaluated at $u = 1$ (red arrow) and $u = -1$ (blue arrow). The inputs corresponding to the black arrows are $u = 0.9$, $u = 0.75$, $u = 0.5$, and $u = 0.3$.

Numerical simulations of (3.13) provide additional evidence that the advancing fronts algorithm yields the most accurate RSS. It was determined to be possible to simulate a trajectory that traverses states outside the RSS boundary given by the discrete space search, but not outside the boundary given by advancing fronts. This was achieved by stepping the input from $P_L = 1$ to $P_L = -1$ and back to $P_L = 1$. The time intervals between steps were selected in such a way that the ensuing trajectory travels as close as possible to the boundary of the RSS. In Figure 4.21, the simulated trajectory is traced in blue and overlayed on top of Figure 4.18.

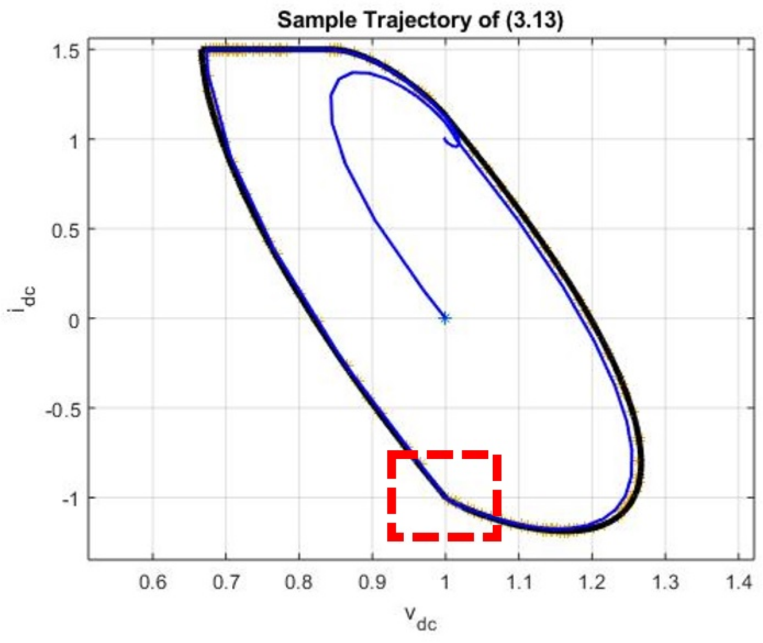

Fig. 4.21. The simulated trajectory starts from $v_{dc} = 1$, $i_{dc} = 0$. The trajectory travels between the two boundaries in the region near $v_{dc} = 1$, $i_{dc} = -1$.

Since it is not immediately evident where the trajectory travels between the two boundaries, this location has been outlinde by the dashed box. The region contained by the dashed box has been magnified in Figure 4.22. Therein, it can be seen that the simulated trajectory travels between the two boundaries upon convergence to $v_{dc} = 1$, $i_{dc} = -1$, and does not return until after P_L is stepped from -1 to $+1$ after a significant amount of time has passed.

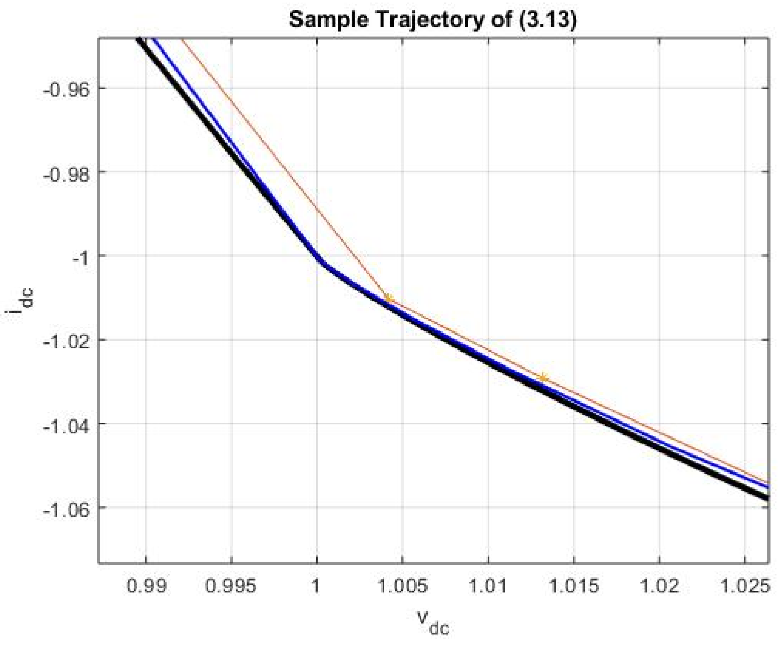

Fig. 4.22. The sample trajectory (blue line) travels just outside the RSS calculated by the discrete space search (yellow line), but always remains inside the RSS calculated by advancing fronts (black line).

4.6 Designing for Large-Displacement Stability

Using the results from the advancing front algorithm, the RSS for (3.13) was superimposed onto the RLDS, as done in Figure 4.23. It is evident that the chosen parameters have ensured large-displacement stability when the parameters for (3.13) are $-1 \leq P_L \leq 1$, $v_{dc}^{*} = 1$, $G = 2$, $i_{\text{lim}} = 1.5$, and $\tau' = 0.2$. Furthermore, the LDSM can be articulated more accurately than what was possible with the discrete-space search. With the chosen parameters, the boundary of the RSS lies extremely close to the edge of the RLDS, resulting in a very narrow LDSM of 0.0041 per unit.

An LDSM of 0.0041 means that even a 0.5 percent difference between the calculated RSS and the actual RSS is not tolerable. However, this level of error is a reasonable expectation, especially from a nonlinear model. Even though the RSS boundary is a conservative estimate, a real system with harmonics, EMI, and other

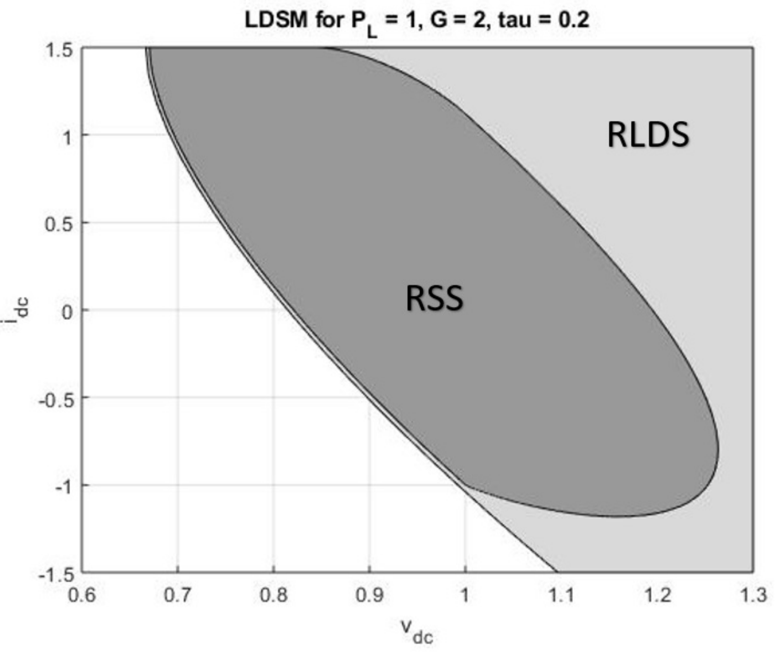

Fig. 4.23. The RSS and RLDS for (3.13). Associated parameters are $-1 \leq P_L \leq 1$, $v_{dc}^* = 1$, $G = 2$, $i_{\lim} = 1.5$, and $\tau' = 0.2$.

disturbances could easily reach a state outside the calculated RSS. Since the LDSM is small, this might also mean that it could reach states outside the RLDS. In addition, all of the design claims in this thesis thus far have been applied to an average-value model. This is convenient and desirable, so long as the average-value model accurately reflects the behavior of a more detailed model. A more detailed model, wherein each switching event is simulated, will have a higher peak amperage. Therefore, its RSS will extend further than that of the average-value model and potentially outside of the RLDS. When this is the case, even though the average-value model is large-displacement stable, a small LDSM implies that the results might not translate to the physical system.

To mitigate these concerns, design parameters must be reconsidered so that the LDSM is much wider. From investigations done in Chapter 3, it was determined that increasing the gain, decreasing the time constant, and decreasing the upper bound on

P_L contribute positively in the way of stability. Making similar adjustments to these parameters can also widen the LDSM. First, the RSS and RLDS were recalculated for a gain of 5 with all other parameters fixed. Adjusting the gain of (3.13) from 2 to 5 resulted in an LDSM of 0.0416 per unit, approximately 10 times wider than the LDSM for the parameters chosen in Figure 4.23. The RLDS and RSS for $G = 5$ are shown in Figure 4.24.

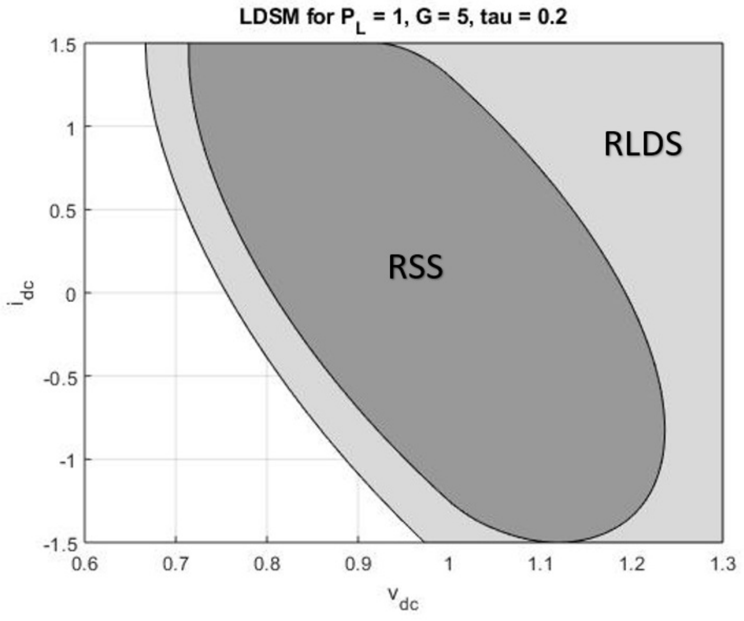

Fig. 4.24. The RSS and RLDS for (3.13). Associated parameters are $-1 \leq P_L \leq 1$, $v_{dc}^* = 1$, $G = 5$, $i_{\lim} = 1.5$, and $\tau' = 0.2$.

Another calculation was performed where the time constant of the controller was made to be 0.1, all other parameters fixed, which is realizable with faster semiconducting devices such as GaN or SiC. Adjusting the time constant resulted in an LDSM of 0.1524 per unit, approximately 47 times wider than the LDSM for the parameters chosen in Figure 4.23. The RLDS and RSS for $\tau' = 0.1$ are shown in Figure 4.25. A third way to increase the LDSM is to operate the system at less than its rated power. As shown in Section 3.4, the upper limit on P_L ultimately defines the RLDS boundary. Referring back to Figure 3.7, if the system were operating at half its rated

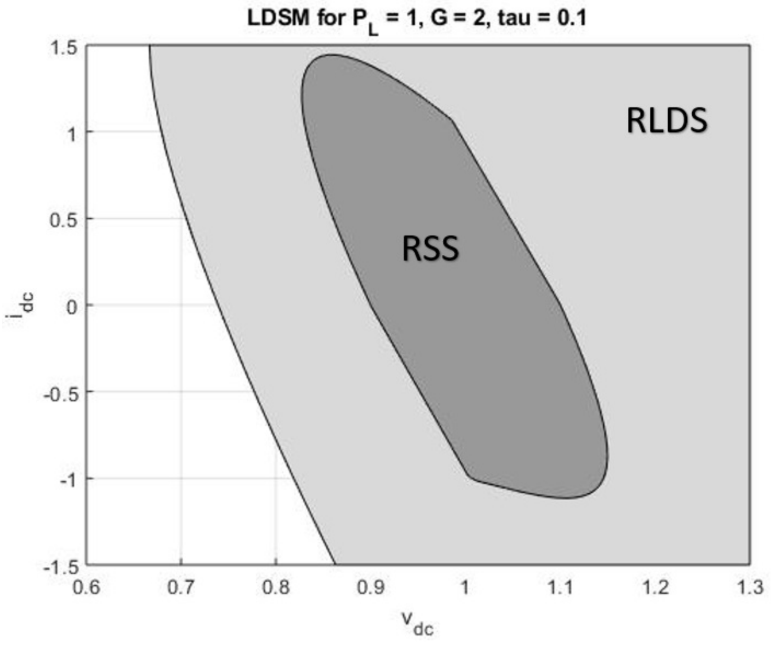

Fig. 4.25. The RSS and RLDS for (3.13). Associated parameters are $-1 \leq P_L \leq 1$, $v_{dc}^{*} = 1$, $G = 2$, $i_{\lim} = 1.5$, and $\tau' = 0.1$.

P_L, a significant amount of area is added to the RLDS. Operating at half its rated power would simultaneously condense the area of the RSS, thus drastically increasing the LDSM. In Figure 4.26, the RSS and RLDS are shown for the system operating at half its rated power, resulting in an LDSM of 0.4564, approximately 100 times wider than the LDSM for the parameters chosen in Figure 4.23. The RLDS and RSS for $0.5 \leq P_L \leq 0.5$ are shown in Figure 4.26.

In summary, the chosen parameters $-1 \leq P_L \leq 1$, $v_{dc}^{*} = 1$, $G = 2$, $i_{\lim} = 1.5$, and $\tau' = 0.2$ produced a system that was indeed large-displacement stable, but with an extremely narrow LDSM. Such a small LDSM means that these parameters are not ideal, but nevertheless provide an excellent starting place given that they are minimally viable for large-displacement stability. Increasing the gain or decreasing the time constant are both proven techniques to widen the LDSM. Increasing the gain from 2 to 5 afforded an LDSM approximately 10 times larger, while decreasing

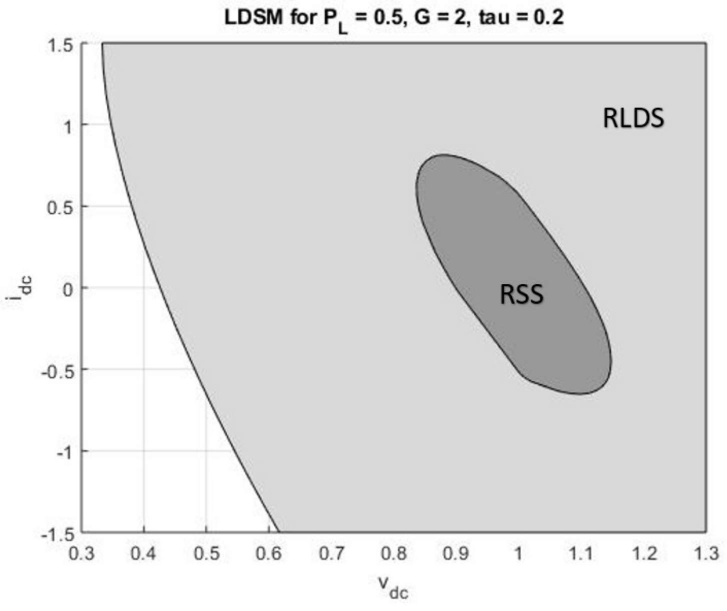

Fig. 4.26. The RSS and RLDS for (3.13). Associated parameters are $-0.5 \leq P_L \leq 0.5$, $v_{dc}^* = 1$, $G = 2$, $i_{\lim} = 1.5$, and $\tau' = 0.2$.

the time constant from 0.2 to 0.1, akin to increasing the switching frequency, afforded an LDSM approximately 40 times larger. Lastly, if the system were to operate at less than its rated power, the LDSM is sure to increase. In other words, designing the system with a higher rating than its intended operating condition is a safe choice when large-displacement stability is a concern.

These results can be immediately applied to arbitrary voltage and currents. For example, a 270-V, 50-kW source is considered. The 1.5 per unit transient overload capacity used previously implies that the source must be able to supply at most 278 A for brief moments following a load step. If the actual capacitance is 1 mF, then from (3.7) the per unit capacitance is calculated to be 1.46 ms. Using (3.9), the source time constant is calculated to be 0.292 ms. The source bandwidth is given by $\frac{1}{2\pi\tau} = 545$ Hz. All together, a dc power system with 1 mF capacitance, 50 percent transient overload capacity, and 545 Hz bandwidth should guarantee large-displacement stability so long as the load power remains within $-50\text{kW} \leq P_L \leq 50\text{kW}$. A smaller bus capacitance is possible, but the switching frequency would have to increase to achieve the desired bandwidth.

5. DETAILED SYSTEM STUDY

To verify the proposed conditions that guarantee large-displacement stability, the design paradigm set forth in Section 2.3 and the studies performed in Section 4.6 were used to establish the specifications for a detailed, practicable model of a dc source. The model extends the circuit in Figure 3.1 by realizing the current source as a 3-phase generator and active rectifier, and the controller as a space-vector modulator and several regulators. This chapter has been reserved for introducing the detailed model, running test-simulations, and validating the average-value model described in Chapter 3. Using the results of the test-simulations, it is shown that the stability of the detailed model is in agreement with that of the average-value model.

5.1 Model Description

All elements of the average-value model are preserved in the detailed model. The current source is now comprised of a permanent-magnet ac (PMAC) machine connected to a dc bus through an active six-step converter which has been shown in Figure 5.1. The PMAC dynamics are modeled using Park's equations [10].

$$v_{qs}^r = r_s i_{qs}^r + \omega_r L_d i_{ds}^r + \omega_r \lambda_m + L_q \frac{di_{qs}^r}{dt} \tag{5.1}$$

$$v_{ds}^r = r_s i_{ds}^r - \omega_r L_q i_{qs}^r + L_d \frac{di_{ds}^r}{dt} \tag{5.2}$$

$$T_e = \frac{P}{2}\left[\lambda_m i_{qs}^r + (L_d - L_q) i_{qs}^r i_{ds}^r\right] \tag{5.3}$$

In (5.1) through (5.3), subscripts q and d are used to denote q-axis and d-axis variables, while superscripts are used to denote the chosen reference frame. It is typical to choose the rotor reference frame for synchronous machines, hence the superscript r. The same r is used in ω_r to denote the rotor's angular velocity. Additionally, λ_m is the peak stator flux linkage due to the permanent magnets, and L_q and L_d are the q-axis and d-axis inductances, respectively. Lastly, the subscript e in (5.3) is used to denote that this is the torque produced by electromagnetic interactions, and P is the number of poles that the PMAC has.

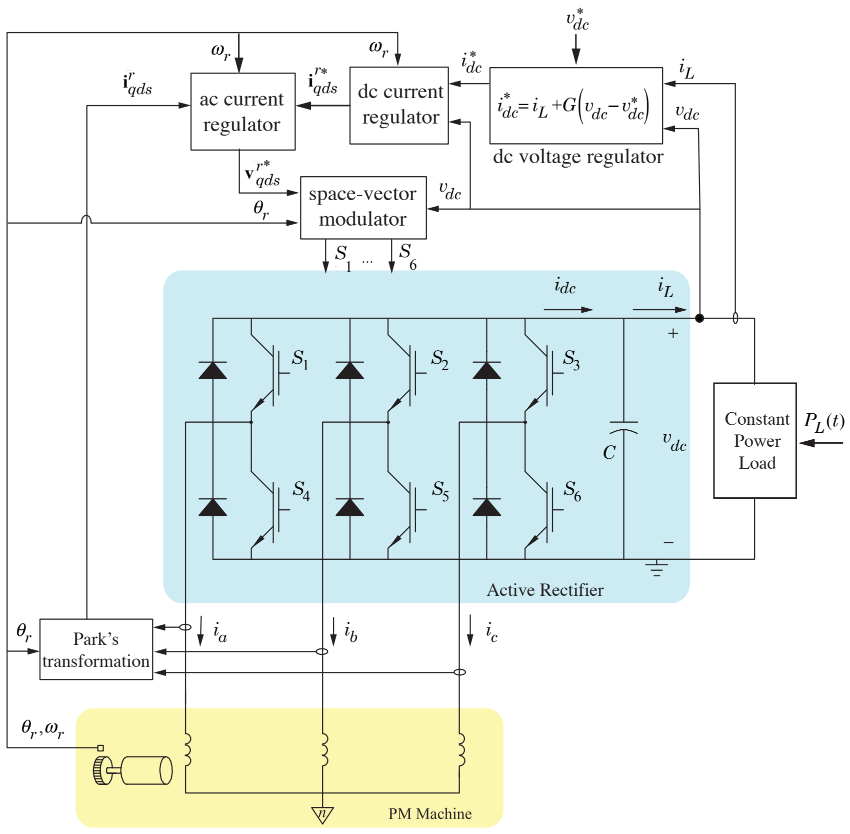

Fig. 5.1. Circuit/Block diagram of the detailed model. The converter is contained in the shaded box labelled "Active Rectifier" while the PMAC is contained in the shaded box labelled "PM Machine".

The converter is bi-directional, so it can operate as an inverter or a rectifier as needed. Each phase-leg of the converter has a high side and low side transistor accompanied by a flyback diode. The semiconducting devices in the converter are modeled as ideal switches, so no voltage drops due to forward-biasing, leakage currents, or any other physical behaviors are considered. The bus capacitor and constant-power load remain exactly as they are in Chapter 3, which models these elements as ideal. As far as the architecture of the circuit is concerned, the model in this chapter is exactly the same as the the average-value model, save for the current source.

The control system consists of a space-vector modulator, an ac current regulator, a dc current regulator, and a dc bus voltage regulator arranged in hierarchic fashion. Outputs of the space-vector modulator are the signals that are used to trigger the switches. Its inputs are the commanded q-axis and d-axis stator voltages, and the measured rotor angle, θ_r. The modulator receives the qd voltages from the ac current regulator which employs a feedforward component of each. Specifically,

$$v_{qs,\text{ff}}^{r*} = r_s i_{qs}^r + \omega_r L_d i_{ds}^r + \omega_r \lambda_m \tag{5.4}$$

$$v_{ds,\text{ff}}^{r*} = r_s i_{ds}^r - \omega_r L_q i_{qs}^r \tag{5.5}$$

where the subscript, ff, denotes feedforward. Feedback components are also needed in this control structure, so the block in Figure 5.2 was included in the ac current regulator. Its gains K_p and K_i should be set such that the transfer function relating i_{dc} to i_{dc}^* satisfies the bandwidth requirements set forth in Section 4.6. In this application, K_i was set to zero as an effort to reduce the dimension of this model.

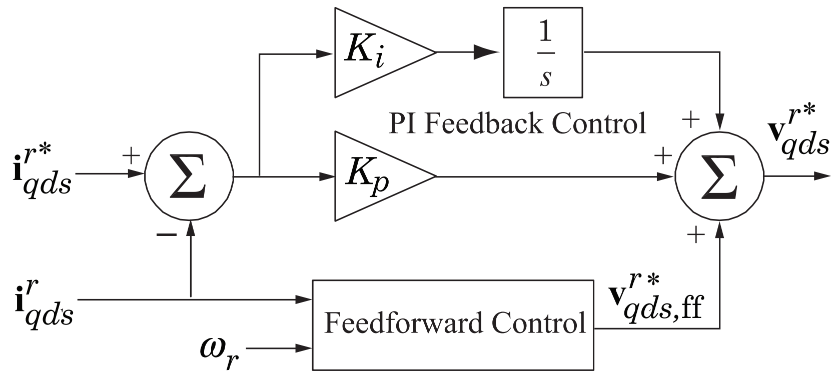

Fig. 5.2. Stator ac current regulator. Its inputs are the commanded and actual qd currents and the rotor speed. Its outputs are the commanded qd stator voltages.

The commanded stator currents that are the inputs to the ac current regulator are produced by the dc current regulator. Therein, the required electric power P_e^* is calculated and is ultimately used to demand more or less current from the PMAC. To do this, the rotor speed is divided from P_e^*, thus yielding the desired torque, T_e^*. If the PMAC is non-salient, ($L_d = L_q$), then from (5.3) it is evident that the d-axis current does not contribute to torque. Hence, i_{ds}^* is set to 0, while i_{qs}^* is set to whatever is needed to achieve T_e^*. The dc regulator block is summarized in Figure 5.3.

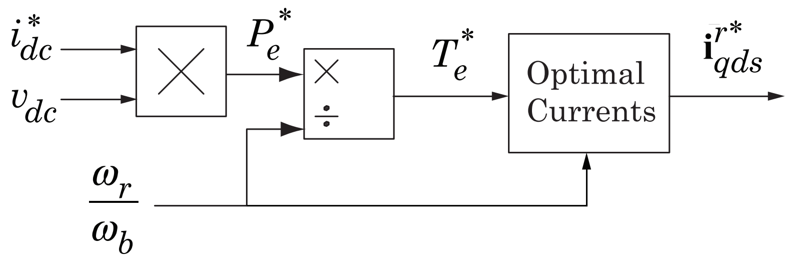

Fig. 5.3. Dc current regulator. Its inputs are the commanded dc current, the actual dc voltage, and the rotor speed. Its outputs are the commanded qd currents.

5.2 Simulation Results

Based on the established design paradigm, feasible parameters for the detailed system are displayed in Table 5.1. Before committing to these parameters for a design, it is necessary to show that the conditions for large-displacement stability is still present in the detailed model. That is, the RSS must be a subset of the RLDS with a sufficiently wide LDSM. The detailed system has 3 dimensions, so the discrete space search algorithm was modified accordingly to calculate the RSS. The RLDS was computed by the same algorithm described in Section 3.3 with no modifications needed. The results of these calculations are displayed in Figure 5.4, where the RSS is the yellow-green "wedge" shaped space above the RLDS, which is the space above the blue surface.

Parameter	Value	Description
P_b	50 kW	rated power
ω_b	$2\pi 400$ rad/s	base (rated) frequency
$V_{b,dc}$	270 V	base (rated) dc voltage
$I_{b,dc}$	185 A	base (rated) dc current
$V_{b,ac}$	108 V	rated peak ac phase-to-neutral voltage
$I_{b,ac}$	309 A	rated peak ac phase current
$\omega_b L_q$	0.1155 Ω	PMAC q-axis reactance
$\omega_b L_d$	0.1155 Ω	PMAC d-axis reactance
r_s	0.01 Ω	PMAC stator resistance
$\omega_b \lambda_m$	108 V	PMAC back emf at rated speed
C	1.5 mF	dc bus capacitance
K_p	5 pu	stator current regulator proportional gain
G	2 pu	dc voltage regulator feedback gain

Table 5.1.

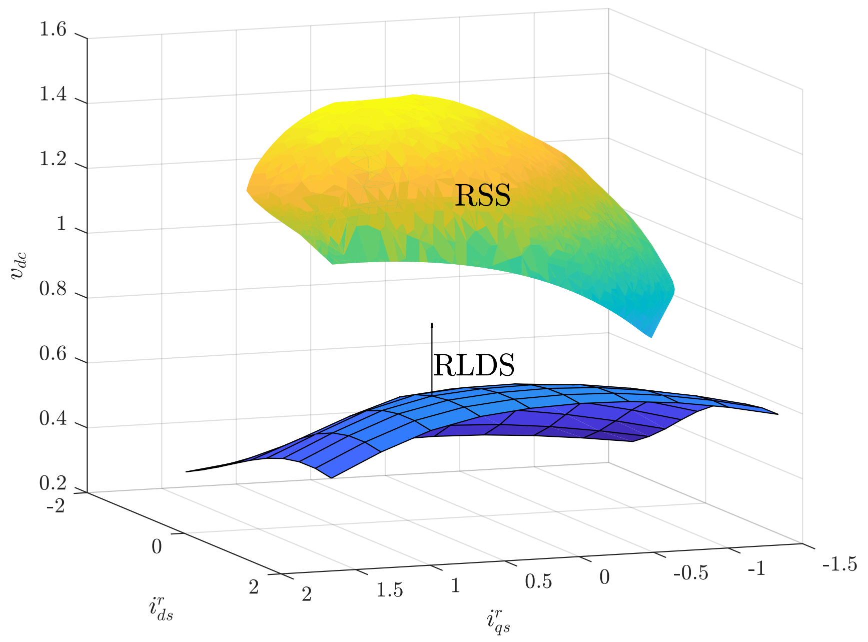

Fig. 5.4. The RLDS boundary and the RSS for the detailed model. The RLDS is the volume above the blue surface

As seen in Figure 5.4, the RLDS encloses all of the RSS, indicating that the first condition for large-displacement stability is present in the detailed system. In this example, the chosen parameters resulted in a significant separation between the boundaries of each space. For the chosen parameters in Figure 5.4, the LDSM is 0.1576 pu, which was established by taking the norm of the difference of each point on each boundary. This indicates a relatively high tolerance for modelling error, system noise, or any other unpredictable physical effects.

As additional validation for the chosen parameters, both models were simulated under a rigorous load-stepping regime in Simulink$^{\text{TM}}$ R2018b. In Figures 5.5-5.8, the system response for step changes in P_L from 0 to 1 at 10 ms, 1 to -1 at 15 ms, and from -1 to 1 at 22 ms are displayed. Results for each model have been superimposed. In each plot, the dashed line corresponds to the average-value model dynamics, while the solid line corresponds to the detailed model.

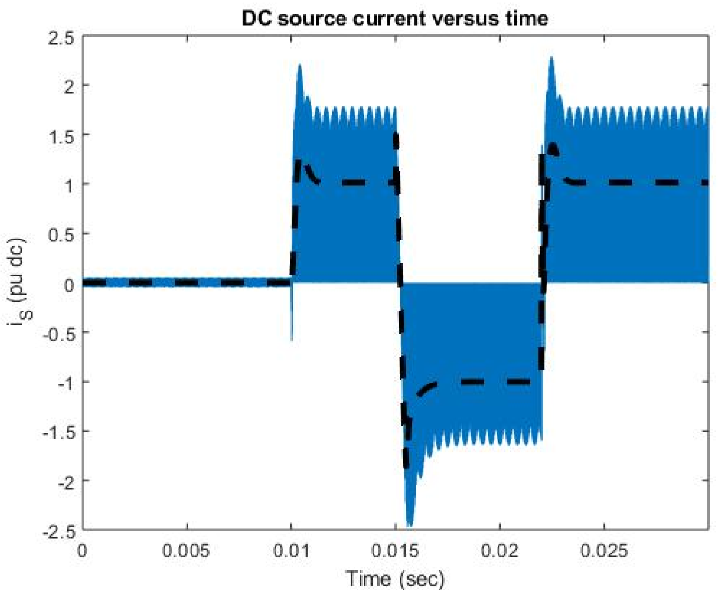

Fig. 5.5. DC source current versus time. In the results from the detailed model, i_S peaks much higher than that of the results from the average-value model, as expected.

Note how i_{dc} varies greatly between the two models. Since the detailed model simulates every switching event, the peak value of i_{dc} will always be larger than that of the average-value model. It is desirable to have a wide LDSM to accommodate for this reality. If the LDSM obtained from the average-value model is very narrow, the practical system might be able to reach states beyond the boundary of the RLDS on account of the differences between the two models with respect to the dc current. For example, if the load was stepped while i_{dc} is at its peak, there exists a small probability that this will destabilize the trajectory. Designing the system with an LDSM of sufficient length eliminates this possibility.

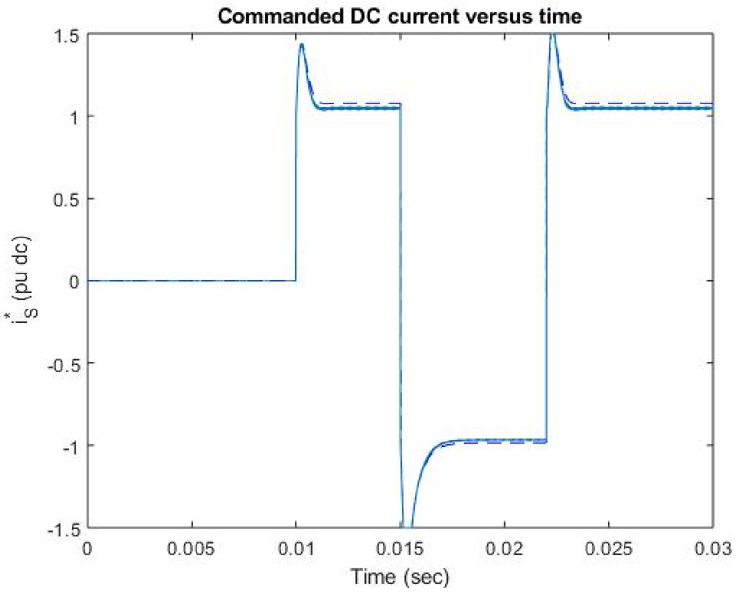

Fig. 5.6. Commanded DC source current versus time. The commanded current is very close in both models, providing evidence that results from the average-value model apply equally as well to the detailed model.

Lastly, a study was conducted on the detailed model where the input varies aggressively between its upper and lower bounds in an attempt to destabilize the model. The inputs in this simulation are square waves with a frequency of 1 kHz. Square waves could represent a pulsed load, which is known to exercise a system drastically.

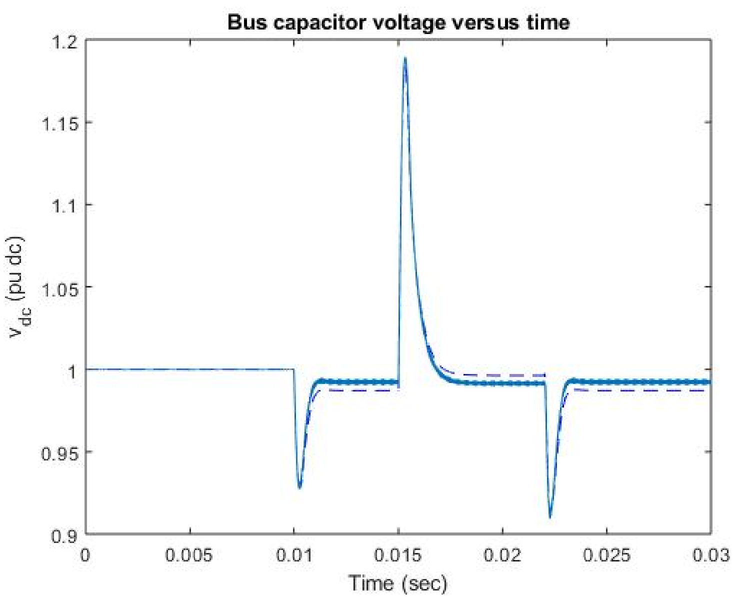

Fig. 5.7. Bus voltage versus time. As in the case of Figure 5.6, the bus voltage is very close in both models, providing evidence that results from the average-value model apply equally as well to the detailed model.

The results of the study are shown in Figure 5.8. The study contains five individual plots. The first plot consists of the dc voltage with respect to time. Even under the intense variation of load power in this experiment, the dc voltage manages to stay above 50 percent of its rated value. The second and third plots are the commanded dc current and the actual dc current, respectively. The dc current swings wildly in this experiment, reaching levels of more than 2 per unit, though large-displacement stability is always retained. The fourth plot is the current of a single phase of the the PMAC. The machine current also exceeds 2 per unit, so the system will never be able to truly handle the excitation provided in this experiment. In light of this, the system remained stable throughout the entire simulation. Despite the attempts to destabilize the model by applying the pulsed load in the fifth plot, the system remains on a stable course as seen in Figure 5.8. This provides evidence that the large-displacement stability criteria is very effective for use in system design.

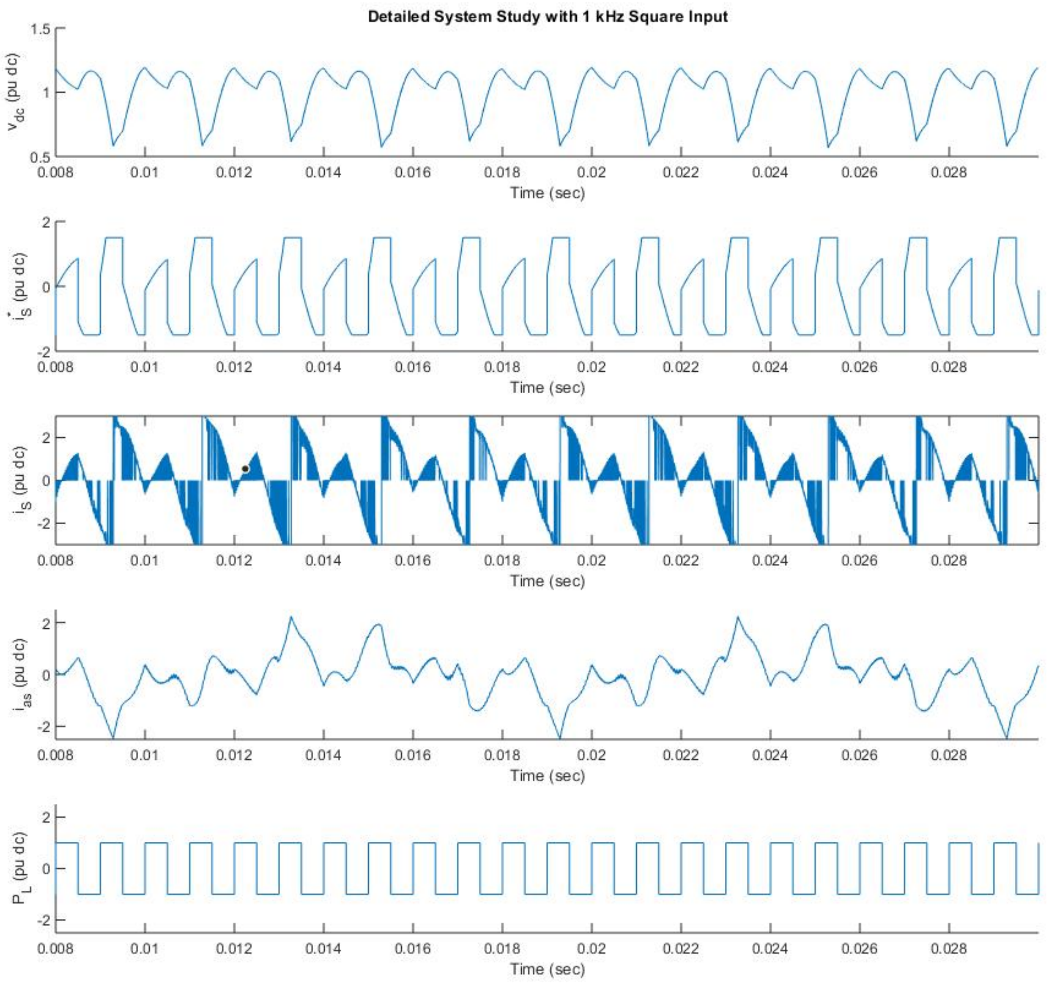

Fig. 5.8. System response for repeated step changes in P_L. The system remains stable through the largest possible displacement in load power.

6. SUMMARY, CONCLUSIONS, AND POTENTIAL FUTURE WORK

In this thesis, an algorithm for calculating the RSS of a nonlinear system was developed, effectively eliminating the need for reachability analysis. The algorithm, entitled "Advancing Fronts" yields a smooth, high resolution, conservative boundary by advancing the set of initial conditions into the RSS. Advancing Fronts was also shown to out-perform other algorithms for calculating the RSS, such as the discrete space search. The algorithm was then used to help choose parameters for an idealized dc power system such that the system is guaranteed to be large-displacement stable under bounded piecewise continuous variations in load power.

The design criteria for achieving large-displacement stability established in [1] was revisited. For a system to be large-displacement stable, its reachable state-space must be a subset of the region of large-displacement stability. As an additional precaution, the minimum difference between the boundaries of the RSS and RLDS must be sufficiently wide to account for modeling uncertainties and system noise. Designing the system is a matter of choosing parameters that ensure that these conditions are met under normal, rated conditions. In Chapter 4, the minimum viable parameters were discovered that ensure large-displacement stability, but with a narrow LDSM. Several solutions were proposed to widen the LDSM, including increasing the gain, decreasing the controller's time constant, or designing the system to be over-rated for its application in terms of load power. Such solutions saw an increase in the LDSM by a factor of 10, 40, and 100, respectively. Finally, these results were validated with a detailed model consisting of a 3-phase, permanent-magnet generator connected to an active rectifier. The detailed model remained large-displacement stable under a strenuous variation in load power, within ± 1 per unit of its power rating.

Although Advancing Fronts yielded a smooth, high-resolution boundary for the RSS, there are still many improvements that could be made in terms of capability and overall design. First, the algorithm has only been implemented for a two-dimensional system, so extending the algorithms capacity to handle a three or higher-dimensional system would be a natural next pursuit. Should the algorithm be extended to higher dimensions, it would be a practical solution for vastly more systems, including the detailed model studied in Chapter 5.

Second, the accommodations made for uneven expansion (i.e. wrapping the nodes, limiting the expansions) are less than elegant. A potential way to mitigate all of the problems associated with expansion would be to solve for the optimal expansion at every node during each iteration. Rather than the current structure, which is to propose an expansion and then review it, an optimization problem could be constructed that satisfies several constraints. These constraints could be that no expansion is too large, all expansions are relatively equal, convexity is maintained, or anything else that is deemed necessary for the optimal expansion. Instead of wrapping the nodes or taking other precautions, simply solve the optimization problem, advance the front, and repeat. Constructing the optimization might be laborious, but it has the potential to dramatically increase the speed and performance of the advancing fronts algorithm. The basic concept of the algorithm remains the same, but all of the precautionary measures would be eliminated.

The designs in Section 4.6 were shown to be large-displacement stable; however, they are not necessarily in compliance with established military standards [12]. Another area for potential future work would be to validate that these 2 designs are in compliance with MIL-STD 704. For example, MIL-STD 704 sets limits on the output voltage transients for a single disturbance. The advancing fronts algorithm could be used to show that the minimum and maximum reachable states for the output voltage during a transient response never exceeds the limits imposed in MIL-STD 704 at any time. Alternatively, CORA could potentially be used to calculate time-dependent

RSS boundaries over the time horizon specified in MIL-SD 704, thus demonstrating compliance.

REFERENCES

REFERENCES

[1] O. Wasynczuk, and T. Craddock, “Ensuring Large-Displacement Stability in Aircraft and Shipboard DC Power Systems,” presented at the Proc. of 2019 IEEE Electric Ship Technologies Symposium (ESTS), Westin Crystal City Hotel, Arlington, VA, 2019.

[2] O. Wasynczuk, M, Gries, B. Selby, P, Lamm, “Designing for Large-Displacement Stability in Aircraft Power Systems,” *SAE Transactions Journal of Aerospace*, April 2009, pp. 894-902.

[3] Althoff, M., Stursberg, and Buss. “Reachability Analysis of Nonlinear Systems with Uncertain Parameters Using Conservative Linearization.” *2008 47th IEEE Conference on Decision and Control* (2008): 4042-048. Web.

[4] Villegas Pico, and Aliprantis, “Reachability Analysis of Linear Dynamic Systems with Constant, Arbitrary, and Lipschitz Continuous Inputs.” *Automatica* 95 (2018): 293-205. Web.

[5] Yang, and Scott. “A Comparison of Zonotope Order Reduction Techniques.” *Automatica* 95 (2018): 378-384. Web.

[6] CORA 2018 Manual, Mathias Althoff and Niklas Kochdumper, Technische Universität München, 85748 Garching, BRD. 2018, pp. 54-55. Accessed on: July, 5, 2019. [Online]. Available: https://tumcps.github.io/CORA/data/Cora2018Manual.pdf

[7] B. P. Loop, S. D. Sudhoff, S. H. Zak, and E. L. Zivi, “An Optimization Approach to Estimating Stability Regions using Genetic Algorithms,” American Control Conference, June 8-10, 2005. Portland, OR.

[8] M. Vidyasagar, **Nonlinear Systems Analysis**, Englewood Cliffs, NJ: Prentice-Hall, 1992.

[9] R. D. Middlebrook, “Input Filter Considerations in Design and Application of Switching Regulators,” *IEEE Proceedings of IASAM*, 1976.

[10] P. C. Krause, O. Wasynczuk, and S. D. Pekarek, **Electromechanical Motion Devices**, John Wiley & Sons, 2012.

[11] S. H. Zak, **Systems and Control**, Oxford University Press, 2003.

[12] Military Standard, ”Aircraft Electric Power Systems”, Department of Defense - United States of America, MIL-STD-704E, May 1, 1991.