SOStab: a Matlab Toolbox for Transient Stability Analysis

Stéphane Drobot*, Matteo Tacchi†, Carmen Cardozo* and Colin Jones‡
* R&D division, Réseau de Transport d'Électricité, La Défense, France
† Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France
‡ Automatic Control Laboratory, EPFL, Lausanne, Switzerland

***Abstract*—This paper presents a new Matlab toolbox, aimed at facilitating the use of polynomial optimization for stability analysis of nonlinear systems. In the past decade several decisive contributions made it possible to recast this type of problems as convex optimization ones that are tractable in modest dimensions. However, available software requires their user to be fluent in Sum-of-Squares programming, preventing them from being more widely explored by practitioners. To address this issue, SOStab entirely automates the writing and solving of optimization problems, and directly outputs relevant data for the user, while requiring minimal input. In particular, no specific knowledge of optimization is needed for implementation. The toolbox allows a user to obtain outer and inner approximates of the Region of Attraction (RoA) of the operating point of different grid connected devices such as synchronous machines and power converters.**

***Index Terms*—AC power systems; Software; Sum-of-squares programming; Transient stability analysis.**

I. Introduction

A *RoA* characterizes a set of initial conditions within the state space of a dynamical system for which trajectories demonstrate stability properties, such as hitting a target (finite time horizon RoA [1], [2]) or converging towards a secure zone of the state space (infinite time horizon RoA [3]–[5]). Practical computation of approximations for both types of RoA is achievable by solving a SoS Programming (SoSP) problem within the framework known as Lasserre's Moment-Sums-of-Squares (SoS) hierarchy, which involves resolving Semidefinite Programming (SDP) problems. While accommodating nonlinear behaviors, this approach comes with convergence guarantees [6]–[8], providing an infinite set of *certified* stable operating conditions through a singular computation.

Consequently, RoA has captured the interest of the power systems and control community. Notably, the potential application of SoSP to the transient stability problem have been explored over the past decade [9]–[13]. Typically performed through contingency analyses using time-domain simulations, transient stability assessment is becoming increasingly challenging as the system operates near its limits in an environment of growing variability and uncertainty.

This work was supported by the Swiss National Science Foundation under the NCCR Automation project, grant agreement 51NF40 180545, and the French company RTE, under the RTE-EPFL partnership n°2022-0225.

In this scenario, having a means to rapidly eliminate situations with guaranteed stability proves invaluable. To serve this purpose, direct methods relying on simplified models, including those based on energy functions or on the Extended Equal Area Criterion (EEAC) [14], are currently being revisited. In this context, RoA-based approaches emerge as a natural alternative, with scalability issues being diligently addressed by the applied mathematics community (see [15] for details and [16]–[20] for existing solutions).

In this work, we address a set of more practical barriers that currently limit further development in this direction, such as:

- The lack of a user-friendly interface for a practical implementation and application of the Moment-SoS hierarchy.
- The conditioning of the underlying Linear Matrix Inequality (LMI), which depends on the specific formulation of the problem.

More precisely, the existing frameworks [21]–[23] require users to manually write and solve SoSP problems to obtain the RoA approximation. In contrast, the proposed SOStab Matlab toolbox fully automates the SoSP aspect, eliminating the need for users to possess knowledge of the Moment-SoS hierarchy. The toolbox operates with minimal input requirements, namely dynamics, state constraints, equilibrium point, time horizon, target set, and a complexity parameter d. It outputs the stability certificate describing the RoA approximation and provides graphical representation in selected state coordinates. SOStab has been developed and made publicly available which allows users to compute RoA of non-linear dynamical systems.

To the best of the authors' knowledge, the only existing toolbox for RoA approximation in a finite time horizon setting is SparseDynamicSystem [18], coded in the Julia language. A notable distinction lies in the fact that the Julia toolbox exclusively supports polynomial dynamics, whereas SOStab is built on the Matlab codes supporting [12], specifically tailored for AC power systems that include phase variables and trigonometric dynamics.

The article is organized as follows: first Section II provides theoretical background on the definition of RoA and the calculation of inner and outer approximations, introducing the methods behind the tool and their relevance to the power system transient stability problem. Basics about the SoSP framework are included in Appendix A. Then, Section III presents the models of the two case studies considered in

this work for illustrative purposes. Section IV includes an overview of the SOStab toolbox and offers guidance on its usage, while simulation results are discussed in Section V. Finally, Section VI summarizes the main contributions of this work and elucidates on potential improvements.

II. Methodological Framework

Consider a generic differential system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{1a}$$

with polynomial dynamics $\boldsymbol{f} \in \mathbb{R}[\boldsymbol{x}]$, equilibrium $\boldsymbol{x}^\star \in \mathbb{R}^n$. Define a threshold $\Delta\boldsymbol{x} \in (0,\infty)^n$ and state constraint

$$\boldsymbol{x}(t) \in \mathbb{X} := [\boldsymbol{x}^\star \pm \Delta\boldsymbol{x}], \tag{1b}$$

$$[\boldsymbol{x}^\star \pm \Delta\boldsymbol{x}] := [x_1^\star - \Delta x_1, x_1^\star + \Delta x_1] \times \ldots \times [x_n^\star - \Delta x_n, x_n^\star + \Delta x_n].$$

Definition 1 (RoA [1], [2])**.** Given a time horizon $T > 0$ and a closed target set $\mathbb{M} \subset \mathbb{X}$, the Region of Attraction $\mathcal{R}_T^{\mathbb{M}}$ of $\mathbb{M}$ in time T is defined as the set of all initial conditions of trajectories that hit the target $\mathbb{M}$ at time T, i.e.:

$$\mathcal{R}_T^{\mathbb{M}} := \left\{ \boldsymbol{x}(0) \in \mathbb{R}^n \ : \ \begin{array}{c} \forall t \in [0,T], \ (1) \text{ holds} \\ \boldsymbol{x}(T) \in \mathbb{M} \end{array} \right\}. \tag{2}$$

The transient stability problem can be reformulated as the search for a RoA in the state space of the post-fault system. Determining the duration a fault can last before a synchronous generator loses stability is equivalent to identifying how *far* the trajectories can diverge before they can no longer return to an acceptable operating point. Here, the vector field $\boldsymbol{f}$ models the nonlinear system dynamics, and the target $\mathbb{M}$ represents a positively invariant vicinity of a stable equilibrium point $\boldsymbol{x}^\star$, both in the post-disturbance state. Consequently, the term *initial condition* refers to the system state at the moment of fault clearance, and T characterizes, in some manner, the recovery time. Finally, state constraints in Eq. (1b) set bounds on state variables accounting for system limits.

Notice that here the dynamcis $\boldsymbol{f}$ are assumed to be polynomial: for non-polynomial dynamics (e.g. phase-related sines and cosines), a preprocessing (Taylor expansion or variables change) is required to enforce polynomial structure.

SOStab builds on [1], [2] and is specifically designed to approximate such an $\mathcal{R}_T^{\mathbb{M}}$ for ellipsoids $\mathbb{M}$ described by

$$\mathbb{M} := \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\mathbf{A}(\boldsymbol{x} - \boldsymbol{x}^\star)\| \leq \varepsilon\} \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a reshaping matrix such that $\det(\mathbf{A}) = 1$, and $\varepsilon > 0$ is an error tolerance.

From a computational viewpoint, considering finite time horizon RoA improves the properties of the resulting SoS programs: they are convex, while their infinite time horizon counterparts are bilinear (and thus nonconvex), which has important consequences on the convergence properties of the corresponding algorithms (see e.g. [11]). In practice, this is a reasonable assumption as operational points of physical system, such as power systems, are constantly moving, making the results of the analysis relevant for a limited time window. In exchange for the finite time horizon, it is necessary to consider target sets that are not reduced to a point, defined by parameters $\mathbf{A}$ and ε; in SOStab, the default value for $\mathbf{A}$ is the identity matrix, but sometimes (e.g. when variables evolve on different time scales), it is useful to make a different choice. For instance, in a two-dimensional singular perturbation framework $|\dot{x}_1| \sim a^2|\dot{x}_2|$, one would set

$$\mathbf{A} = \begin{pmatrix} 1/a & 0 \\ 0 & a \end{pmatrix}$$

i.e. take an ellipse of equation $(x_1 - x_1^\star)^2/a^2 + a^2(x_2 - x_2^\star)^2 \leq \varepsilon^2$ for $\mathbb{M}$, which is justified by the physical reasoning that, for a trajectory starting at (x_1, x_2) and converging to $(x_1^\star, x_2^\star)$ in infinite time, it holds

$$x_1^\star - x_1 = \int_0^\infty \dot{x}_1(t)\, dt \sim a^2 \int_0^\infty \dot{x}_2(t)\, dt = a^2(x_2^\star - x_2),$$

and those two terms should have equal contribution in the description of the target $\mathbb{M}$, otherwise the resulting RoA will be skewed. Also, from Eq. (3) it holds that $\mathrm{vol}(\mathbb{M}) \propto \varepsilon^n$, where vol denotes the n-dimensional volume of a set, so that the choice of the tolerance parameter ε is directly related to the desired size of the target $\mathbb{M}$.

SOStab takes as input the problem data $(\boldsymbol{f}, \Delta\boldsymbol{x}, T, \mathbf{A}, \varepsilon)$, as well as a parameter $d \in 2\mathbb{N}$ which sets the accuracy and complexity of the approximation, and outputs the following objects for outer and inner RoA estimation:

$$\boldsymbol{y}_d^{out} = \begin{pmatrix} \lambda_d^{out} & v_d^{out} & w_d^{out} \end{pmatrix} \tag{4a}$$

$$\boldsymbol{y}_d^{in} = \begin{pmatrix} \lambda_d^{in} & v_d^{in} & w_d^{in} \end{pmatrix}, \tag{4b}$$

with $\lambda_d^{in/out} \geq 0$, $v_d^{in/out} \in \mathbb{R}_d[t, \boldsymbol{x}]$ degree d polynomials in $(t, \boldsymbol{x})$ and $w_d^{in/out} \in \mathbb{R}_d[\boldsymbol{x}]$ degree d polynomials in $\boldsymbol{x}$, such that the following inclusion guarantees hold:

$$\mathcal{R}_T^{\mathbb{M}} \subset \{\boldsymbol{x} \in \mathbb{R}^n \ : \ v_d^{out}(0, \boldsymbol{x}) \geq 0\} =: \mathcal{R}_d^{out} \tag{5a}$$

$$\mathcal{R}_T^{\mathbb{M}} \supset \{\boldsymbol{x} \in \mathbb{R}^n \ : \ v_d^{in}(0, \boldsymbol{x}) < 0\} =: \mathcal{R}_d^{in}. \tag{5b}$$

In words, $v_d^{out}(0, \boldsymbol{x}) \geq 0$ (resp. $v_d^{in}(0, \boldsymbol{x}) < 0$) means that $\boldsymbol{x}$ belongs to an outer (resp. inner) approximation $\mathcal{R}_d^{out}$ (resp. $\mathcal{R}_d^{in}$) of $\mathcal{R}_T^{\mathbb{M}}$. Moreover, λ_d is an upper bound of the approximation error, and w_d is often used to check numerical results: it should separate $\mathbb{X}$ into a zone where its values are close to 0 and one where they are above 1; one of those two zones missing means that the solver failed to detect the RoA. Eventually, the framework comes with precision guarantees:

$$\mathrm{vol}\left(\mathcal{R}_d^{in/out}\right) \underset{d\to\infty}{\longrightarrow} \mathrm{vol}\left(\mathcal{R}_T^{\mathbb{M}}\right). \tag{6}$$

In words, when the modelling parameter d goes to infinity, the volume of the error made by the approximation schemes vanishes (see Appendix A or [1], [2] for further details and Appendix B for an elementary example).

In the transient stability problem, inner approximations are more relevant as they provide stability certificates for all states within, even if it means excluding some post-fault conditions leading to stable trajectories. Outer approximations ensure the inclusion of all stable states but lack conservatism guarantees. The gap between the two may inform about the approximation accuracy with respect to the exact RoA.

III. Models for case study

In this section we present the model of two different test cases, proposed in the literature, to illustrate the tool capabilities, flexibility and performance. The first one is the well known synchronisation loop of classic grid-following converters: the phase locked loop (PLL [24]–[26]). The second consists of a synchronous machine described by a third order model, to which a governor and automatic voltage regulator (AVR) representations are added. The grid side is represented by an infinite bus [10], [13].

A. Phase Locked Loop 2^{nd} order model

Fig. 1 shows a generic PLL block diagram [24]. Here an angular state variable θ is required to match a reference θ_{ref}; to that end, the system computes the sine of the phase difference ϕ, multiplies it by some gain K and takes it through a low-pass filter with transfer function $F(s) = \frac{1+\tau_2 s}{\tau_1 s}$, resulting in the following differential system:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} \omega \\ -K\frac{\tau_2}{\tau_1}\cos(\phi)\omega - \frac{K}{\tau_1}\sin(\phi) \end{pmatrix}. \tag{7}$$

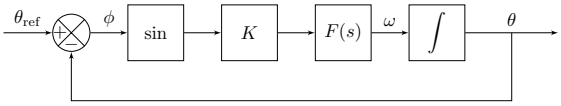

Fig. 1. Block scheme of a PLL system.

Following [24], in the PLL setting the time constants τ_1 and τ_2 are functions of the gain K, a natural frequency ω_n and a damping ratio ζ:

$$\tau_1 = \frac{K}{\omega_n^2} \qquad \tau_2 = \frac{2\zeta}{\omega_n} \tag{8}$$

In our case study, we will use SOStab to compute the RoA of the PLL system described by Eq. (7) with parameter values as described in Tab. I, and where sine and cosine have been replaced by their degree 10 Taylor expansions (denoted `si` and `co` respectively), so that the dynamics are polynomial.

Parameter	K	ω_n	ζ
Value	1 s^{-1}	10.813 s^{-1}	1.3303

TABLE I
Parameters for phase locked loop.

B. Single machine - infinite bus with regulations

Following [10], Fig. 2 represents a synchronous machine connected to an infinite bus with voltage $\boldsymbol{v}_s$ through a power transmission line with impedance $Z_t = R_t + jX_t$. The physics of the synchronous machine are modelled by the swing equation as well as the electromotive force dynamics:

$$\dot{\theta} = \omega - \omega_s \tag{9a}$$
$$2H\dot{\omega} = P_m - (v_d i_d + v_q i_q + r i_d^2 + r i_q^2) \tag{9b}$$
$$T'_{d_0}\dot{e}'_q = -e'_q - (x_d - x'_d)i_d + E_{fd} \tag{9c}$$

where θ is the machine angle in a rotating frame synchronized with the grid (hence the grid frequency ω_s in (9a)) and the current $\boldsymbol{i}$ and voltage $\boldsymbol{v}$ are described in the rotating frame by the Kirchhoff laws:

$$i_q = \frac{(X + x'_d)\|\boldsymbol{v}_s\|\sin\theta - (R + r)(\|\boldsymbol{v}_s\|\cos\theta - e'_q)}{(R + r)^2 + (X + x'_d)(X + x_q)}$$
$$i_d = \frac{X + x_q}{R + r} i_q - \frac{\|\boldsymbol{v}_s\|\sin\theta}{R + r}$$
$$v_d = x_q i_q - r i_d$$
$$v_q = R i_q + X i_d + \|\boldsymbol{v}_s\|\cos\theta$$

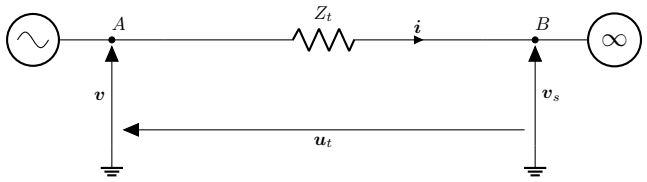

Fig. 2. A synchronous machine connected to an infinite bus.

Moreover, the model includes quadratic AVR dynamics

$$T_a\dot{E}_{fd} = -E_{fd} + K_a(V^2 - \|\boldsymbol{v}\|^2) \tag{9d}$$

as well as turbine governor equation

$$T_g\dot{P}_m = -P_m + P + K_g(\omega_s - \omega) \tag{9e}$$

where V and P are given set points for the voltage magnitude $\|\boldsymbol{v}\|$ and mechanical power P_m. In our case study, we will use SOStab to compute the RoA of the 5D SMIB model given by Eq. (9) with parameter values as described in Tab. II, along with the variable change

$$\boldsymbol{x} = (\sin\theta, \cos\theta, \omega, e'_q, E_{fd}, P_m) \tag{10}$$

so that the dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ are polynomial.

Parameter	T'_d	x_d	x'_d	x_q	r
Value	9.67 s	2.38 pu	0.336 pu	1.21 pu	0.002 pu
Parameter	H	ω_s	R	X	$V = \|\boldsymbol{v}_s\|$
Value	3 s	1 pu	0.01 pu	1.185 pu	1 pu
Parameter	T_a	K_a	T_g	K_g	P
Value	1 s	70 pu	0.4 pu	0.5 pu	0.7 pu

TABLE II
Parameters for single machine - infinite bus.

IV. Computing RoA with SOStab

In this section we showcase a step by step procedure to assess the stability of our test cases with SOStab.

A. Installation

SOStab is a freeware subject to the General Public Licence (GPL) policy available for Matlab. It can be downloaded at:

`https://github.com/droste89/SOStab`

SOStab requires YALMIP [21], as well as a semidefinite solver. Mosek [27] is used by default, but it can be replaced by any other solver, provided they are installed and interfaced through YALMIP.

B. Overview

SOStab is designed to compute a classifier, denoted as $v(0,\cdot)$, which can be evaluated at any state $\boldsymbol{x}$ of the considered system. The sign of $v(0,\boldsymbol{x})$ indicates whether the state belongs or not to the selected RoA approximation, thereby characterizing the stability of the trajectory starting at $\boldsymbol{x}$. The toolbox requires the following minimal input:

$$\begin{aligned}&\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{f} \in \mathbb{R}[\mathbf{x}]^n, \quad T > 0\\ &\boldsymbol{x}^\star \in \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}\}\\ &\Delta\boldsymbol{x} \in (0,\infty)^n, \quad d \in 2\mathbb{N}, \quad \varepsilon > 0,\end{aligned}$$

where $\boldsymbol{f} \in \mathbb{R}[\mathbf{x}]$ means that the system dynamics should have a polynomial form (up to Taylor expansion or variable change). Then, it identifies the problem to be solved as: "compute approximations $\mathcal{R}_d$ of $\mathcal{R}_T^{\mathbb{M}}$ with dynamics $\boldsymbol{f}$, admissible set $[\boldsymbol{x}^\star \pm \Delta\boldsymbol{x}]$ and target set $\mathbb{M} = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\boldsymbol{x} - \boldsymbol{x}^\star\| \leq \epsilon\}$ ".

SOStab also admits two optional input arguments:

- a shape matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ such that $\det(\mathbf{A}) = 1$ reshapes the target set as $\mathbb{M} = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ \|\mathbf{A}(\boldsymbol{x} - \boldsymbol{x}^\star)\| \leq \epsilon\}$;
- in case the system at hand involves trigonometric functions of phase variables $\theta_1, \ldots, \theta_N$, it is also possible to specify a phase index matrix $\boldsymbol{\Theta} \in \mathbb{R}^{N\times 2}$ whose first (resp. second) column consists of the indices of the sines (resp. cosines) of the θ_i in the recasted variable $\boldsymbol{x}$.

The use of the toolbox consists of four steps (see Fig. 3):

1) The initialization is performed from input $(\boldsymbol{x}^\star, \Delta\boldsymbol{x})$ (and optional input $\boldsymbol{\Theta}$); it defines the admissible set $[\boldsymbol{x}^\star \pm \Delta\boldsymbol{x}]$ and identifies the dimension and variables of the system.
2) The user then inputs the dynamics $\boldsymbol{f}$ of the system and adjusts optional settings of the toolbox.
3) The RoA approximation itself is performed from input (d, T, ε) (with optional input $\mathbf{A}$); it outputs the SDP solutions $\boldsymbol{y}_d = (\lambda_d, v_d, w_d)$.
4) Graphic representations of the solutions can eventually be plotted; the choice of the plots abscissa and ordinate and plotting options are up to the user.

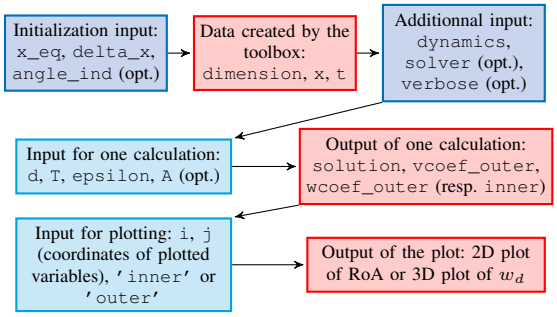

Fig. 3. Flowchart of the toolbox workings

We are now going to demonstrate the implementation of those four steps on the PLL and SMIB test cases.

Remark 1. In the current version of the toolbox, when using an optional argument, one should also specify all optional arguments that appear before in the method call.

C. Initialization and dynamics specification

The first initialization step consists in choosing the threshold $\Delta\boldsymbol{x}$ defining the admissible set $\mathbb{X}$; For the PLL, we follow the reference [24] and ask $\Delta\phi = \pi$ rad and $\Delta\omega = 20\pi$ rad/s (the equilibrium is trivially $\boldsymbol{x}^\star = \mathbf{0}$). Similarly, we identify an equilibrium point $\boldsymbol{x}^\star$ and set a threshold $\Delta\boldsymbol{x}$ for the SMIB model (with variable $\boldsymbol{x}$ described as in Eq. (10)).

```
PLLeq = [0;0];   DeltaPLL = [pi; 20*pi];
SMIBeq = [sin(1.539); cos(1.539); ...
          1; 1.070; 2.459; 0.7];
DeltaSMIB = [1; 1; 1; 1; 20; 4];
```

Remember that the PLL dynamics are approximated by their Taylor expansion, while we study the exact SMIB model through a variable change. Hence, for the SMIB model, we need to specify which coordinates of $\boldsymbol{x}$ are actually trigonometric functions of the original variable, by adding the input:

```
angle_ind = [1,2];
```

Here `angle_ind` $= \boldsymbol{\Theta}$ is the phase index matrix with only one line as the SMIB model features only one phase angle θ; the first (resp. second) column gives the position of $\sin\theta$ (resp. $\cos\theta$) in the recasted variable

$$\boldsymbol{x} = (\sin\theta, \cos\theta, \omega, e'_q, E_{fd}, P_m).$$

Next, for each of our two test cases we create an instance of SOStab with the following commands:

```
PLL = SOStab(PLLeq, DeltaPLL);
SMIB = SOStab(SMIBeq, DeltaSMIB, Z);
```

The initial call creates an instance of the class, and defines a number of internal properties, among which one can find the following useful ones:

- internal copies of the inputs `XXXeq` and `DeltaXXX` (and optionally `angle_ind`, empty by default)
- `dimension`: problem dimension (number of variables)
- `x`: a YALMIP sdpvar polynomial object, of the dimension of the problem. It represents the variable $\boldsymbol{x}$ and is called by the user to define the dynamics of the system
- `t`: sdpvar polynomial of size 1, representing the time variable t, which can be needed to define the dynamics of the system (if non-autonomous)
- `solver`, the choice of the solver used in the optimization, defined as Mosek by default, it can also be SeDuMi
- `verbose`, the value of the verbose parameters of the YALMIP optimization calls, defined at 2 by default (all numerical SDP solver info displayed), it can also be 1 (selection of info displayed) or 0 (no info about the numerical resolution of the underlying SDP)
- `dynamics`, a YALMIP polynomial defining the polynomial dynamics $\boldsymbol{f}$ of the system.

Eventually, after declaring all required parameters (see Tab. I and II), the system dynamics are introduced:

```
PLL.dynamics = [PLL.x(2); ...
-K/tau1*(si(PLL.x(1)) + tau2*co(PLL.x(2)))];
```

where `PLL.x` denotes the variable $\boldsymbol{x}$ of the PLL system (automatically defined by `SOStab` when it is first called). For the sake of brevity the code for SMIB dynamics is reported in Appendix C. A crucial note is that it features the *recasted* dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, meaning that $\dot{\theta} = \omega - \omega_s$ is replaced with the dynamics of its recasted images $s = \sin\theta$, $c = \cos\theta$:

$$\dot{s} = (\omega - \omega_s)c \tag{11a}$$
$$\dot{c} = (\omega_s - \omega)s \tag{11b}$$

D. RoA approximation

The inner and outer RoA approximation of the system defined by the call of `SOStab` are then computed by the methods `SoS_in` and `SoS_out`, respectively. For the PLL system, we first compute a degree 16 outer approximation of the time $T = 1$ RoA of $\mathbb{M} = \left\{(\phi, \omega) \in \mathbb{R}^2 \ : \ 20\phi^2 + 0.05\omega^2 \leq 1.7^2\right\}$ using the following commands:

```
T=1;          epsilon = 1.7;          d=16;
A = [[20^(1/2) 0] ; [0 20^(-1/2)]];
[vol, vc, wc] = PLL.SoS_out(d,T,epsilon,A);
```

In the SMIB case, we seek a degree 6 inner approximation of the time $T = 10$ RoA of $\mathbb{M} = \{\boldsymbol{x} \in \mathbb{R}^6 \ : \ \|\boldsymbol{x} - \boldsymbol{x}^\star\| \leq 0.2\}$, hence we enter the commands:

```
T=10;          epsilon = 0.2;          d=6;
[vol, vc, wc] = SMIB.SoS_in(d,T,epsilon);
```

Additionnal properties are related to a specific solution of the optimization problem. They are calculated at each call of the optimization and stored until the next call, *i.e.* each of them correponds to the previous optimization call:

- internal copies of the inputs `d`, `T`, `epsilon` (and optional `A`, set to identity by default); recall that `d` is the degree of the polynomials v_d and w_d
- `vcoef_outer`, the coefficients of the solution v_d^{out} for the last calculated outer approximation of the ROA
- `wcoef_outer`, the coefficients of the solution w_d^{out} for the last calculated outer approximation of the ROA
- `vcoef_inner`, the coefficients of the solution v_d^{in} for the last calculated inner approximation of the ROA
- `wcoef_inner`, the coefficients of the solution w_d^{in} for the last calculated inner approximation of the ROA
- `solution`, a volume approximation of the last calculated ROA, *ie* the solution λ_d of the optimization problem

The output of SOStab can be used as follows: as introduced in Section II, λ_d and w_d are used to assess the numerical solver performance; the key output is then the polynomial classifier v_d, which is used as follows: in *any* situation where the `dynamics` are valid for the considered system (be it initialization or post-fault transients) and its state is represented by $\boldsymbol{x}(t)$, $v_d^{in}(0, \boldsymbol{x}(t)) < 0$ means that the system will remain in the secure zone $\mathbb{X}$ and hit its target: $\boldsymbol{x}(t+T) \in \mathbb{M}$; conversely, if $v_d^{out}(0, \boldsymbol{x}(t)) < 0$, then either the state will leave $\mathbb{X}$ before time $t+T$ or it will miss its target: $\boldsymbol{x}(t+T) \notin \mathbb{M}$; hence, the only remaining uncertainty will be when both $v_d^{in/out}(0, \boldsymbol{x}(t)) \geq 0$. Hence, assessing the stability of a configuration $\boldsymbol{x}(t)$ boils down to evaluating the polynomial $v_d(0, \boldsymbol{x}(t))$, which can be done instantly.

V. Case study

In this section, we display and comment the results obtained by running the codes presented in IV.

A. RoA of the Phase Locked Loop system

When called according to the instruction of Section IV to assess the stability of the PLL model, SOStab returns the approximate surface `vol`$= \lambda_d^{out}$ of the computed RoA estimate in the phase space, as well as two vectors `vc` and `wc` consisting of the coefficients of polynomials v_d^{out} and w_d^{out} respectively. We run SOStab for the PLL system for various values of d and compile the results in Tab. III. As λ_d^{out} is a proxy for the size of the outer RoA estimate, the smaller it is, the more accurate the approximation. Here one can observe that as expected the accuracy increases with d at the price of higher computational time. The CPU times were obtained on a Macbook laptop with an Apple M2 chip and 16 GB of RAM.

d	λ_d^{out}	CPU time (s)
4	4.0000	2.8169
8	3.5892	4.6729
12	3.1284	14.5575
16	2.9346	45.2185

TABLE III
OUTPUTS OF SOSTAB DEPENDING ON PRECISION PARAMETER d

Once the optimization problem is solved, the following command plots two-dimensional slices of the boundary

$$\partial\mathcal{R}_d^{out} = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ v(0, \boldsymbol{x}) = 0\}$$

```
PLL.plot_roa(1,2,'outer');
```

where the first two arguments indicate the indices of the represented variables (respectively in abscissa and ordinate), in case there are more than two. The string `'outer'` indicates that the toolbox plots an outer estimate of the RoA.

For inner RoA approximation [2], the commands are

```
[vol, vc, wc] = PLL.SoS_in(d, T, epsilon);
PLL.plot_roa(1, 2, 'inner',1);
```

Here the last argument with value `1` is an optional argument that asks SOStab to also represent the target set $\mathbb{M}$ in the figure. The admissible set $\mathbb{X}$ is the whole plotting window, so that it is always visualized. This yields the plot represented in Fig. 4, which can be compared to [24, Fig. 10, Right].

It can also be interesting to display 3D-plots of polynomials v_d^{out} and w_d^{out}, which can be performed by the commands (see Fig. 5: one retrieves the shape of the inner RoA estimate):

```
PLL.plot_v(1, 2, 'outer');
PLL.plot_w(1, 2, 'outer');
```

Of course, one can also represent the certificates v_d^{in} and w_d^{in} obtained in inner approximation, simply by setting the last argument at `'inner'`.

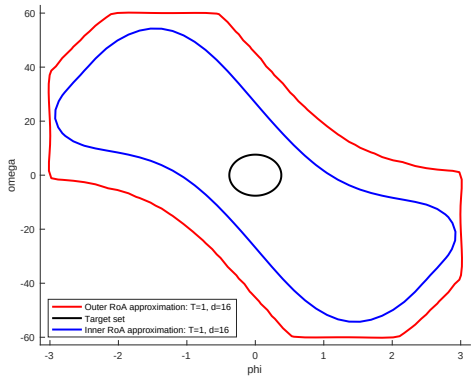

Fig. 4. Inner and outer RoA approximations for the PLL system.

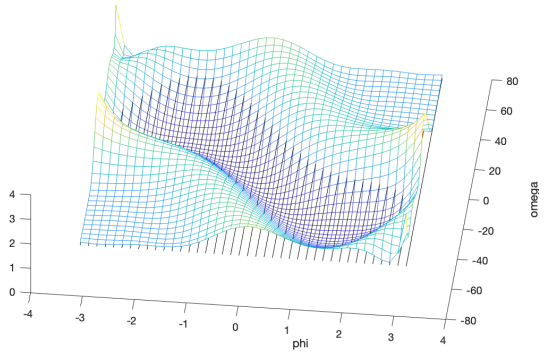

Fig. 5. Plot of w_d^{in} for the PLL system

Those 3D plots can be useful when the validity of the RoA estimate is unclear. Indeed, although those estimate are theoretically certified, SOStab can fail to compute an RoA approximation, due to bad conditioning and numerical solver inaccuracy; to detect such behaviour, one can plot the graph of w_d: if it is almost flat with values $w_d(\boldsymbol{x}) \simeq 1$ everywhere, then the RoA estimation failed.

B. RoA of the single machine - infinite bus system

Instead of performing Taylor expansions as in the previous section, it is also possible to directly tackle trigonometric functions, through the algebraic change of variables in Eq. (10) [12].

After the commands displayed in Section IV, a user has access to the description of the inner RoA estimate through the polynomial $v_d^{in}(0, \mathbf{x})$. It is also possible to compute an inner estimate and plot both on a 2D graph in the (θ, ω) coordinates (see Fig. 6), with the following commands:

```
SMIB.plot_roa([1,2],3,'inner',1);
SMIB.SoS_out(d, T, epsilon);
SMIB.plot_roa([1,2],3,'outer',1);
```

Here the argument `[1,2]` means that the abscissa of the plot should be the phase variable θ, which SOStab knows only from its images $x_1 = \sin\theta$ and $x_2 = \cos\theta$. More generally, setting `[i,j]` in the coordinate arguments of SOStab results in the abscissa of the plot being the phase variable θ such that $x_i = \sin\theta$ and $x_j = \cos\theta$.

The second argument `3` sets $x_3 = \omega$ as the ordinate of the plot. Another choice such as `6,3` (without brackets) would result in plot coordinates $(x_6, x_3) = (P_m, \omega)$ as in Fig. 7. Those figures are similar to the existing results [10, Fig. 5.b-c]. For the degree 6 estimates of the SMIB model RoA, the outer approximation took 49.085 seconds, while the inner approximation (much more difficult in practice due to conditioning technicalities) took approximately 2 hours.

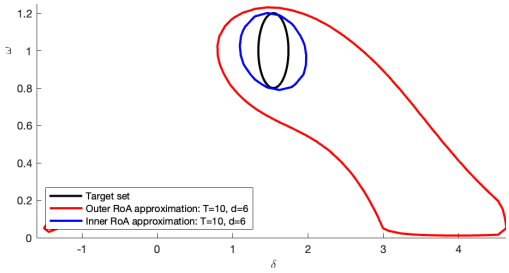

Fig. 6. RoA approximation of the SMIB in the (δ, ω) coordinates.

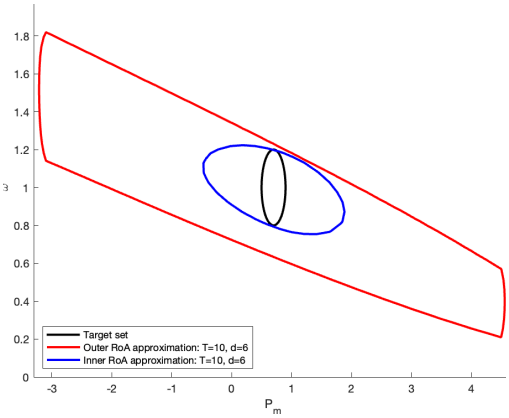

Fig. 7. RoA approximation of the SMIB in the (P_m, ω) coordinates.

C. A curse of dimensionality

The above examples all include very simplified models of power system devices, with a 2D 2OM PLL and a 5D SMIB model. This is due to the fact that, for computational times and numerical conditioning to be reasonable, the toolbox currently works only with low-dimensional systems. Indeed, SOStab is internally asked to solve LMIs of combinatorial size $\binom{n+d+1}{d}$ where n is the number of variables modelling the system and d is the degree of the certificates v_d and w_d. The current computational limits of a standard use of SoSP in stability analysis were reached in [10], [13] and the 5D SMIB model. A more promising approach for scaling the method consists in splitting the considered problems into subproblems of more modest size, based on the structure of the problem. For instance, [18] exploits the algebraic notions of term sparsity and chordal sparsity to handle up to 16 dynamical variables while keeping convergence guarantees. More physics-oriented are the methods in [16] (resp. [17]), where causality (resp. time scale separation) is exploited to decompose the full

problem into more tractable subproblems. Such ideas have been successfully implemented in the simpler (because static) AC-OPF problem setting, for which *continental-scale* systems have been handled by SoSP methods [19], [20]. While very promising, these methodologies pose significant challenges when it comes to their automation into a Matlab toolbox, and hence were not implemented in the current version of SOStab. This, although out of the scope of the present article, will be deeply investigated in future works.

VI. Discussions and conclusions

This work presented a new Matlab Toolbox called SOStab, which aims at helping a non-expert in polynomial optimization to use the frameworks developed in [1], [2], [12] through a plug-and-play interface. Such interface would greatly facilitate the setup of new experiments to leverage the conditioning issues; further, it could be included in future schemes aimed at scaling the method to high dimensions.

A. Contribution highlights

In practice, [1], [2], [12] only provide a *methodology* for RoA approximation. As detailed in those references as well as Appendix A, the Moment-SoS hierarchy consists in solving SoSP problems of increasing size, which requires to follow a number of steps, related to real algebraic geometry and optimization:

1) Defining the geometric characteristics of the problem (polynomial dynamics $\boldsymbol{f}$, time horizon T, admissible and target sets $\mathbb{X}$ and $\mathbb{M}$)
2) Defining an algebraic description of these geometric characteristics: polynomials $\boldsymbol{g} \in \mathbb{R}[\mathrm{x}]^m$ s.t. $\mathbb{X} = \{\boldsymbol{x} : \boldsymbol{g}(\boldsymbol{x}) \geq 0\}$ and $\boldsymbol{h} \in \mathbb{R}[\mathrm{x}]^\ell$ s.t. $\mathbb{M} = \{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) \geq 0\}$
3) Coding a method to integrate polynomials over $\mathbb{X}$ (i.e. computing the moments of the Lebesgue measure on $\mathbb{X}$)
4) Writing the SoSP problems with explicit positivity constraints; this requires introducing internal SoS certificates as decision variables
5) Recasting SoS constraints as LMIs, solving the resulting SDP problem and converting the solution back to the polynomial framework
6) Extracting the corresponding certificates v_d, w_d and using them to characterize and plot relevant representations of the computed RoA approximation.

Existing frameworks [21]–[23] have been designed to automate step 5 which appears in all instances of SoSP. As a result, they are very flexible in their use, but they also require their user to perform steps 1–4 and 6 by hand, which involves solid knowledge in SoSP and may be time consuming and prone to human errors (especially in step 4) ; hence their use is usually not smooth, even for experts.

In contrast, with SOStab, only step 1 is left to the user, and all the other operations are automatically performed. More precisely, intrinsic properties of the dynamical system are defined as presented in Section IV-C, and then settings for finite horizon RoA are the input of methods `SoS_in` and `SoS_out`; with this, steps 2–5 are performed through a call to YALMIP, for inner and outer RoA approximation respectively, and output an optimal value `solution` and the coefficients of the optimal polynomials v_d and w_d.

The benefits of this contribution are the following:

(a) Knowledge on SoS programming becomes optional to use the Lasserre hierarchy for RoA approximation.
(b) The user input is significantly reduced, limiting implementation efforts.
(c) The toolbox comes with a plug-and-play design that allows one to repeat multiple experiments, reproduce existing results from the literature and solve new problems.

B. Limitations and future works

However, some limitations remain to be leveraged. For instance, while convex, SDP problems can be ill-conditioned, which sometimes results in poor numerical behavior with $w_d^\star \simeq 1$ and meaningless plots. It is possible to rescale SoS constraints to mitigate that phenomenon, although finding the appropriate rescalings is non-trivial.

Moreover, inner RoA approximation requires an algebraic representation of the boundary $\partial\mathbb{X}$ of the admissible set $\mathbb{X}$ [2], and while choosing a box is the most physically relevant (and the easiest to integrate polynomials on), it induces some numerical difficulties that would not arise if $\mathbb{X}$ were described by a single polynomial. This can be solved either by changing the description of $\partial\mathbb{X}$, or by changing the admissible set $\mathbb{X}$.

Last but not least, at a given number n of variables (such that $\boldsymbol{x} \in \mathbb{R}^n$ and given precision degree d, SOStab (and the Lasserre hierarchy in general) requires to solve size $\binom{n+d+1}{d}$ SDP problems, which quickly becomes intractable on any computer. To tackle this issue, structure exploiting methods have been developed in [15]–[20], which consist in splitting the underlying SDP problems into problems of smaller size, while keeping as much computing precision as possible; these techniques bear the potential for scaling the hierarchy up to more realistic power systems such as fully modelled power converters or low order models of distributions network (which exhibit a radial structure one can exploit in computations).

Future works on the SOStab class will include:

(a) Running the toolbox on more sophisticated case studies such as power converters
(b) Improving the inner approximation scheme to increase the accuracy of each relaxation
(c) Supporting richer admissible $\mathbb{X}$ and target $\mathbb{M}$ sets, such as ellipsoids, ℓ^p-balls, annuli...
(e) Supporting bases of polynomials other than monomials (Chebyshev, Legendre, trigonometric polynomials)
(f) Exporting the toolbox to other softwares compatible with existing SDP solvers, such as Julia or Python
(g) Adding structure exploiting methods to scale the method to higher dimensional dynamical systems

References

[1] D. Henrion and M. Korda, "Convex computation of the region of attraction of polynomial control systems", *IEEE Transactions on Automatic Control* 59(2):297–312, 2013.

[2] M. Korda, D. Henrion and C.N. Jones, "Inner approximations of the region of attraction for polynomial dynamical systems", *IEEE Transactions on Automatic Control* 59(2):297–312, 2013.

[3] Z.W. Jarvis-Wloszek, *Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-square optimization.* PhD thesis, University of California, 2003.

[4] A. Oustry, M. Tacchi and D. Henrion, "Inner approximations of the maximal positively invarian set for polynomial dynamical systems", *IEEE Control Systems Letters* 3(3):733–738, 2019.

[5] M. Korda, D. Henrion and C.N. Jones, "Convex computation of the maximal positively invariant set for polynomial control systems", *SIAM Journal on Control and Optimization* 52(5):2944–2969, 2014.

[6] J.B. Lasserre, *Moments, Positive Polynomials and Their Applications.* Imperial College Press, 2010.

[7] D. Henrion, M. Korda and J.B. Lasserre, *The Moment-SOS Hierarchy.* World Scientific, 2021.

[8] M. Tacchi, "Convergence of Lasserre's hierarchy: the general case", *Optimization Letters* 16(3):1015–1033, 2022.

[9] M. Anghel, F. Milano and A. Papachristodoulou, "Algorithmic construction of Lyapunov functions for power system stability analysis", *IEEE Transactions on Circuits and Systems I* 60(9):2533-2546, 2013.

[10] M. Tacchi, B. Marinescu, M. Anghel, S. Kundu, S. Benahmed and C. Cardozo, "Power system transient stability analysis using sum of squares programming", in *Power Systems Computation Conference*, 2018.

[11] S. Izumi, H. Somekawa, X. Xin and T. Yamasaki, "Estimating regions of attraction of power systems by using sum of squares programming", *Electrical Engineering* 100:2205–2216, 2018.

[12] C. Josz et al."Transient stability analysis of power systems via occupation measures", in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2019.

[13] A. Oustry et al. "Maximal positively invariant set determination for transient stability assessment in power systems", *IEEE 58th Conference on Decision and Control (CDC)*, Nice (France), pp. 6572–6577, 2019.

[14] A. Bahmanyar et al., "Extended equal area criterion revisited: a direct method for fast transient stability analysis," Energies 14:1–48, 2021.

[15] M. Tacchi, *Moment-SOS hierarchy for large scale set approximation. Application to power systems transient stability analysis.* PhD thesis, INSA Toulouse, 2021.

[16] M. Tacchi, C. Cardozo, D. Henrion, J.B. Lasserre, "Approximating regions of attraction of a sparse polynomial differential system", *IFAC-PapersOnLine*, 53(2):3266–3271, 2020.

[17] I. Subotić et al. "A Lyapunov framework for nested dynamical systems on multiple time scales with application to converter-based power systems", *IEEE Transactions on Automatic Control*, 66(12):5909–5924, 2021.

[18] J. Wang et al. Exploiting term sparsity in moment-SoS hierarchy for dynamical systems. *IEEE Transactions on Automatic Control* 68(12):8232–8237, 2023.

[19] C. Josz and D.K. Molzahn, "Lasserre hierarchy for large scale polynomial optimization in real and complex variables", SIAM Journal on Optimization 28(2):1017–1048, 2018.

[20] A. Le Franc, V. Magron, J.B. Lasserre, M. Ruiz and P. Panciatici, "Minimal sparsity for second-order moment-SOS relaxations of the AC-OPF problem". IEEE Transactions on Power Systems (early access):1–10, 2023.

[21] J. Löfberg, "Pre- and post-processing sum-of-squares programs in practice", *IEEE Transactions on Automatic Control* 54(5):1007–1011, 2009.

[22] D. Henrion, J.B. Lasserre and J. Löfberg, "GloptiPoly 3: moments, optimization and semidefinit programming", *Optimization Methods and Software* 24(4-5):761–779, 2009.

[23] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P.A. Parrilo, M.M. Peet and D. Jagt, *SOSTOOLS – Sum of Squares Optimization Toolbox for MATLAB*, published online in 2002.

[24] T.-C. Wang, S. Lall and T.-Y. Chiou, "Polynomial method for PLL controller optimization", *Sensors* 11:6575–6592, 2011.

[25] C. Zhang, M. Molinas, J. Lyu, H. Zong and X. Cai, "Understanding the nonlinear behaviour and synchronizing stability of a grid-tied VSC under grid voltage sags", *IEEE 8th Renewable Power Generation Conference (RPG)*, Shanghai (China), 2019.

[26] C. Zhang, M. Molinas, Z. Li and X. Cai, "Synchronizing stability analysis and region of attraction estimation of greed-feeding VSCs using Sum-of-Squares programming", *Frontiers in Energy Research* 8, article 56, 2020.

[27] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.0*, 2022.

APPENDIX

A. Lasserre hierarchy for Region of attraction

In this section, the generic problem of computing the finite time RoA of a given target set is presented, along with the SoS framework to address it. Consider the system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{12a}$$

with vector field $\boldsymbol{f} \in C^\infty(\mathbb{R}^n)^n$ and state constraint

$$\boldsymbol{x}(t) \in \mathbb{X} \tag{12b}$$

for some subset $\mathbb{X} \subset \mathbb{R}^n$ representing security constraints.

Definition 2. Given a time horizon $T \in (0, \infty]$ and a closed target set $\mathbb{M} \subset \mathbb{X}$, the Region of Attraction (RoA) of $\mathbb{M}$ in time T is defined as

$$\mathcal{R}_T^{\mathbb{M}} := \left\{ \boldsymbol{x}(0) \in \mathbb{R}^n \ : \ \begin{array}{c} \forall t \in [0,T), \quad \boldsymbol{x}(t) \in \mathbb{X} \\ \operatorname{dist}(\boldsymbol{x}(t), \mathbb{M}) \underset{t \to T}{\longrightarrow} 0 \end{array} \right\} \tag{13}$$

Remark 2. Definition 1 covers many frameworks, such as:

- Infinite time RoA ($T = \infty$, $\mathbb{X} = \mathbb{R}^n$, often $\mathbb{M} = \{0\}$)
- Maximal positively invariant set ($T = \infty$, $\mathbb{M} = \mathbb{X} \subsetneq \mathbb{R}^n$)
- Constrained finite time RoA ($T < \infty$, $\mathbb{X}$ compact)

We now introduce an infinite dimensional Linear Programming (LP) problem that is related to the constrained finite horizon RoA (see [1] for details):

$$\begin{aligned} W^\star := \inf \ & \int_{\mathbb{X}} w(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ \text{s.t. } & v \in C^\infty(\mathbb{R}^{n+1}), w \in C^\infty(\mathbb{R}^n) \\ & w \geq 0 && \text{on } \mathbb{X} && \text{(14a)} \\ & w \geq v(0,\cdot) + 1 && \text{on } \mathbb{X} && \text{(14b)} \\ & \mathcal{L}_{\boldsymbol{f}} v := \partial_t v + \boldsymbol{f}^\top \boldsymbol{\partial}_{\boldsymbol{x}} v \leq 0 && \text{on } \mathbb{\Gamma} && \text{(14c)} \\ & v(T,\cdot) \geq 0 && \text{on } \mathbb{M} && \text{(14d)} \end{aligned}$$

where $\mathbb{\Gamma} := [0,T] \times \mathbb{X} \Subset \mathbb{R}^{n+1}$ denotes a time-state cylinder.

Proposition 1 ([1]). *Let* (v, w) *be feasible for* (14)*. Then,*

$$\begin{aligned} \mathcal{R}_T^{\mathbb{M}} &\subset \{\boldsymbol{x} \in \mathbb{R}^n \ : \ v(0,\boldsymbol{x}) \geq 0\} && \text{(15a)} \\ &\subset \{\boldsymbol{x} \in \mathbb{R}^n \ : \ w(\boldsymbol{x}) \geq 1\} =: \mathbb{L}(w \geq 1) && \text{(15b)} \end{aligned}$$

With Proposition 1, constraint (14a) enforces $w \geq \mathbb{1}_{\mathcal{R}_T^{\mathbb{M}}}$, where $\mathbb{1}_{\mathbb{A}}$ denotes the boolean indicator function of $\mathbb{A} \subset \mathbb{R}^n$ (with value 1 in $\mathbb{A}$ and 0 elsewhere). Moreover, it is proven in [1] that for any minimizing sequence $(v_d, w_d)_{d \in \mathbb{N}}$ for (14), one has $w_d \underset{d \to \infty}{\longrightarrow} \mathbb{1}_{\mathcal{R}_T^{\mathbb{M}}}$ in the sense of $L^1(\mathbb{X})$, so that the volume of the approximation error $\mathbb{L}(w_d \geq 1) \setminus \mathcal{R}_T^{\mathbb{M}}$ converges to 0.

The Moment-SoS hierarchy allows its user to compute such a minimizing sequence, under the following assumptions.

Assumption 1. All considered inputs are polynomial:

1.1. $\boldsymbol{f} \in \mathbb{R}[\mathbf{x}]^n$, so that $v \in \mathbb{R}[\mathrm{t}, \mathbf{x}] \Longrightarrow \mathcal{L}_{\boldsymbol{f}} v \in \mathbb{R}[\mathrm{t}, \mathbf{x}]$

1.2. $\mathbb{X} = \cap_{i=1}^m \mathbb{L}(g_i \geq 0) =: \mathbb{L}(\boldsymbol{g} \in \mathbb{R}_+^m)$ with $\boldsymbol{g} \in \mathbb{R}[\mathbf{x}]^m$

1.3. $\mathbb{M} = \cap_{j=1}^{\ell} \mathbb{L}(h_j \geq 0) = \mathbb{L}(\boldsymbol{h} \in \mathbb{R}_+^{\ell})$ with $\boldsymbol{h} \in \mathbb{R}[\mathbf{x}]^{\ell}$ where $\mathbf{x}$ (resp. t) denotes the dimension n (resp. 1) indeterminate (i.e. identity function, which can be evaluated in any state $\boldsymbol{x} \in \mathbb{R}^n$, resp. time $t \in \mathbb{R}$).

Then, it is possible to work with polynomial Sums-of-Squares (SoS), with the following definitions.

Definition 3. Let $p \in \mathbb{R}[\mathbf{x}]$. Then, considering $g_0 := 1$,

- p is SoS *iff* $p = q_1^2 + \ldots + q_N^2$, $q_1, \ldots, q_N \in \mathbb{R}[\mathbf{x}]$
- $p \in \mathcal{Q}(\boldsymbol{g})$ *iff* $p = s_0\, g_0 + \ldots + s_m\, g_m$, $s_0, \ldots, s_m$ SoS
- $p \in \mathcal{Q}_d(\boldsymbol{g})$ *iff* $p \in \mathcal{Q}(\boldsymbol{g})$ with $\max(\deg s_i\, g_i) \leq d$

Since SoS polynomials are nonnegative by design, it is clear from Definition 3 that any $p \in \mathcal{Q}_d(\boldsymbol{g})$ (resp. $\mathcal{Q}_d(\boldsymbol{h})$) is nonnegative on $\mathbb{X}$ (resp. $\mathbb{M}$), which gives access to a strenghtening of problem (14):

$$
\begin{aligned}
W_d^\star := \inf \quad & \int_{\mathbb{X}} w(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \\
\text{s.t. } & v \in \mathbb{R}[\mathrm{t}, \mathbf{x}], w \in \mathbb{R}[\mathbf{x}] \\
& w && \in \mathcal{Q}_d(\boldsymbol{g}) && \text{(16a)} \\
& w - v(0, \mathbf{x}) - 1 && \in \mathcal{Q}_d(\boldsymbol{g}) && \text{(16b)} \\
& -\mathcal{L}_{\boldsymbol{f}} v && \in \mathcal{Q}_d(\boldsymbol{g}, (T - \mathrm{t})\, \mathrm{t}) && \text{(16c)} \\
& v(T, \mathbf{x}) && \in \mathcal{Q}_d(\boldsymbol{h}) && \text{(16d)}
\end{aligned}
$$

where $\Gamma = \mathbb{L}((T - \mathrm{t})\, \mathrm{t} \geq 0) \times \mathbb{X} = \mathbb{L}((\boldsymbol{g}, (T - \mathrm{t})\, \mathrm{t}) \in \mathbb{R}_+^{m+1})$. Problem (16) consists in looking for feasible (v, w) for (14) under the form of polynomials, restricting inequality constraint (14x) into SoS constraint (16x), x=a–d. The advantage of this new problem is that the decision variables are now finite dimensional vectors of coefficients, and the SoS constraints can be recast as LMIs [6]. Thus, assuming knowledge of the moments of the Lebesgue measure on $\mathbb{X}$ (i.e. being able to integrate polynomials on $\mathbb{X}$, e.g. if $\mathbb{X}$ is a ball or a box), one is able to solve (16) on a standard computer, provided that n and d are small enough to make the LMIs tractable.

As the new feasible space is strictly included in the former, there is a relaxation gap: $\forall d \in 2\mathbb{N},\ W^\star < W_d^\star$. However, it is proved in [1] that, if $\mathbb{X}$ and $\mathbb{M}$ are bounded, $W_d^\star \underset{d \to \infty}{\longrightarrow} W^\star$. Thus, solving instances of (16) gives access to converging outer approximations of the RoA.

B. A standard polynomial system

To further illustrate how SOStab is able to reproduce existing results in SoSP for stability analysis, one can consider a reversed-time Van der Pol oscillator, as in [2]:

$$
\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} -2x_2 \\ 0.8x_1 + 10(x_1^2 - 0.21)x_2 \end{pmatrix} \tag{17}
$$

The dynamics are polynomial and the stable equilibrium $\boldsymbol{x}^\star$ of the system is at the origin, so that it takes very few lines of code to get interesting results:

```
VdP = SOStab([0;0], [1.1;1.1]);
VdP.dynamics= [-2*VdP.x(2); 0.8*VdP.x(1)
+ 10*(VdP.x(1)^2 - 0.21)*VdP.x(2)];
d = 12;          T = 1;          epsilon = 0.5;
```

```
[vol, vc, wc] = VdP.SoS_out(d, T, epsilon);
VdP.plot_roa(1, 2, 'outer');
[vol, vc, wc] = VdP.SoS_in(d, T, epsilon);
VdP.plot_roa(1, 2, 'inner',1);
```

This gives the plot represented in Fig. 8, which reproduces results presented in [1, Figure 2] and [2, Figure 3].

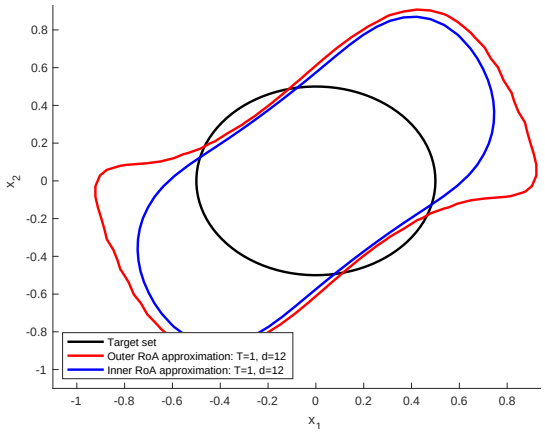

Fig. 8. Inner and outer RoA approximations for the Vanderpol system.

C. Code for the SMIB RoA dynamics

```
Td = 9.67;  xd = 2.38;  xpd = 0.336;
xq = 1.21;  H = 3;  r = 0.002;  w = 1;
R = 0.01;  X = 1.185;  V = 1;  Ta = 1;
Ka = 70;  Tg = 0.4;  Kg = 0.5;  P = 0.7;

iq = ((X+xpd)*V*SMIB.x(1) - (R+r)*...
     (V*SMIB.x(2)-SMIB.x(4)))/...
     ((R+r)^2 + (X+xpd)*(X+xq));
id = (X+xq)/(R+r)*iq - 1/(R+r)*V*SMIB.x(1);
vd = xq*iq - r*id;
vq = R*iq + X*id + V*SMIB.x(2);
Vt = vd^2+vq^2;

SMIB.dynamics = [(SMIB.x(3)-weq)*...
   SMIB.x(2); (weq-SMIB.x(3))*SMIB.x(1);...
   (SMIB.x(6) - vd*id - vq*iq - r*id^2...
     - r*iq^2)/(2*H);...
(SMIB.x(5) - SMIB.x(4) + (xpd-xd)*id)/Td;...
(Ka*(V-Vt) - SMIB.x(5))/Ta ;...
(P - SMIB.x(6) + Kg*(w-SMIB.x(3)))];
```