Input-Output Systems Analysis Using Sum of Squares Programming: A Decomposition Approach

James Anderson and Antonis Papachristodoulou

Abstract— **A method for estimating the $\mathcal{L}_2$ gain of a nonlinear system using sum of squares (SOS) programming and dynamical system decomposition is presented. Typically SOS approaches to systems analysis are only computationally tractable for systems of a modest state dimension comprising of low order vector fields. We present a dynamical system decomposition approach that extends the class of systems that the $\mathcal{L}_2$ gain can be estimated for using SOS methods to include those of large state dimension with high vector field degree.**

I. INTRODUCTION

In this paper we describe methods based on dynamical system decomposition that allow for the scalable analysis of nonlinear polynomial systems. The decomposition approach is used in conjunction with the sum of squares (SOS) decomposition and dissipation inequalities to obtain bounds on the $\mathcal{L}_2$-gain of a class of systems that due to the dimension of their state vector cannot be analyzed directly using traditional SOS programming algorithms.

For nonlinear systems obtaining an upper bound on the $\mathcal{L}_2$-gain is not a trivial task. Traditional approaches rely on the solution of the Hamilton-Jacobi equation [1], [2] which can, if asymptotic stability of the system under study is assumed, be reduced to satisfying a dissipation inequality [3]. However, both approaches rely on the construction of a Storage function $S(x)$ that is required to meet certain positivity conditions: $S(x)$ must be positive definite and its time derivative along system trajectories must be bounded by a quadratic supply rate function. In reality, constructing Storage functions is a difficult task and in general requires a lot of intuition about a specific system. The task is further complicated by the realization that certifying the required positivity conditions, even if they are polynomial is an $\mathcal{NP}$-hard problem when the degree of the polynomial under consideration is greater than four [4].

The sum of squares decomposition provides a relaxation to polynomial non-negativity and under certain additional constraints can test positivity of a polynomial [5]. What makes this approach particularly attractive is that SOS decompositions can be computed by solving a Semidefinite Programme (SDP) [6] which is a convex optimization problem and as such has a worst case polynomial-time complexity [7].

Recent work that has applied SOS programming techniques to input-output stability analysis can be found in [8], [9]. These approaches are however limited to systems with fewer than five or six states depending on the degree of the system vector field. For systems with a larger state dimension or high order vector fields the underlying SDPs that require solving become too large for current solvers to handle. Algebraic techniques for reducing the complexity of the SOS decomposition based on symmetry and the Newton polytope method are described in [10]. However, even when these methods are used, analyzing dynamical systems with seven or more states is challenging.

In [11] a technique based on decomposing an autonomous dynamical system into a set of interacting subsystems was proposed and a system containing 16 states was successfully analyzed using the Lyapunov framework. The decomposition algorithm casts the dynamical system as a weighted graph. Vertices represent states and the edge weights represent the energy flow between states when the system is released from an initial condition that maximizes the energy flow on the edges of the network (in the sense of some norm). The idea is to then partition the graph into subgraphs such that states that interact strongly are kept in the same partition and the energy that flows between partitions is minimized. The dynamical system is then decomposed into a set of subsystems connected in feedback corresponding to how the graph was partitioned. Thereafter stability certificates for each of the subsystems produced from the decomposition using SOS programming can be integrated to form a composite Lyapunov function [12] to certify stability of the original system. This method was shown to be more computationally efficient than a direct approach which many times is not possible for large-scale systems.

In this paper we show how the decomposition approach can be applied to input-output analysis of nonlinear systems. The paper is organized as follows: In the next section an overview of traditional methods for calculating $\mathcal{L}_2$-gain of a nonlinear system is given and some basic ideas from sum of squares programming and algebraic graph theory are presented. In Section III the decomposition algorithm is described and a method for estimating $\mathcal{L}_2$ gain based on system decomposition is presented. Examples are given in Section IV before the paper is concluded in Section V.

II. BACKGROUND

In this paper we are concerned with nonlinear dynamical systems of the form

$$\begin{aligned} \dot{x} &= f(x) + G(x)u, \quad x(0) = x_0 \\ y &= h(x), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the input and $y \in \mathbb{R}^p$ is the output. We assume $f(x)$ is locally Lipschitz

J. Anderson and A. Papachristodoulou are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, U.K. OX1 3PJ {james.anderson,antonis}@eng.ox.ac.uk

ISBN 978-963-311-370-7

and $G(x)$ and $h(x)$ are continuous over $\mathbb{R}^n$. G is an $n \times m$ matrix and $h : \mathbb{R}^n \to \mathbb{R}^p$, both of which vanish at the origin. Throughout this work we restrict our attention to polynomial functions.

For the space of piecewise continuous, square-integrable functions we have the standard $\mathcal{L}_2[0, \infty)$ norm defined by

$$\|u\|_{\mathcal{L}_2} = \sqrt{\int_0^\infty u^T(t)u(t)dt} < \infty. \tag{2}$$

Note that the standard vector norm $\sqrt{z^T z}$ is denoted by $\|z\|$.

Frequently in control system design and system analysis one views a system as an input-output mapping from an external disturbance signal u to an output signal y. In this setting it is desirable to determine the magnitude of the amplification of the disturbance from input to output. A subsequent performance objective may then be to minimize this amplification. Traditionally this is carried out by comparing some norm of the input and output signals. Typically the $\mathcal{L}_2$ norm as defined above is used and the gain from input to output is known as the system $\mathcal{L}_2$-gain defined by

$$\sup_{\|u\|_{\mathcal{L}_2} \neq 0} \frac{\|y\|_{\mathcal{L}_2}}{\|u\|_{\mathcal{L}_2}} \tag{3}$$

with $x(0) = 0$, i.e. the maximum ratio of output to input energy. A system is said to be finite-gain $\mathcal{L}_2$ stable if (3) is finite for all $u \in \mathcal{L}_2$.

To achieve design objectives that incorporate such criteria a method for calculating the $\mathcal{L}_2$ gain (or to provide an upper bound of it) is required. The motivation of this paper is to provide a scalable SOS programming solution to this problem for systems with a high state dimension. In the next section traditional methods for calculating system gains are described.

A. $\mathcal{L}_2$ Gain Estimation

For the case of stable linear time-invariant (LTI) systems of the form

$$\dot{x} = Ax + Bu \tag{4a}$$
$$y = Cx + Du \tag{4b}$$

with a frequency domain representation $G(s) = C(sI - A)^{-1}B + D$, the $\mathcal{L}_2$ gain of the system is given by $\sup_{\omega \in \mathbb{R}} \|G(j\omega)\|_2$. This is also referred to as the $\mathcal{H}_\infty$ norm of $G(j\omega)$. In this regard LTI systems are special case of systems whose $\mathcal{L}_2$ norm can be measured exactly. In general it is only possible to find an upper bound for the $\mathcal{L}_2$-gain of a nonlinear system; we now outline two traditional methods for computing such an upper bound:

Theorem 1: [2] Consider the nonlinear time-invariant system given by (1). Let γ be a scalar such that $\gamma > 0$ and suppose there exists a positive semidefinite, continuously differentiable function $S(x)$ that satisfies the Hamilton-Jacobi Inequality:

$$\frac{\partial S}{\partial x} f(x) + \frac{1}{2\gamma^2} \frac{\partial S}{\partial x} G(x) G^T(x) \left(\frac{\partial S}{\partial x} \right)^T + \frac{1}{2} h^T(x) h(x) \leq 0 \tag{5}$$

for all $x \in \mathbb{R}^n$. For each x_0, system (1) is finite-gain $\mathcal{L}_2$ stable and its $\mathcal{L}_2$-gain is less than or equal to γ.

Proof: The proof can be found in [1], [2] and [12]. ■

As can be seen from Theorem 1, even determining an upper bound on the $\mathcal{L}_2$ gain can be challenging for several reasons; i) computing $S(x)$ and ensuring it is psd is not trivial for high order systems, and ii) the value of γ is linked to the choice of $S(x)$ thus obtaining tight bounds is difficult.

Finding solutions to the Hamilton-Jacobi inequality directly can be both analytically and computationally challenging, particularly when the system has a high state dimension. However, in [2] Theorem 1 is shown to be equivalent to the the following theorem based on dissipation inequalities.

Theorem 2: System (1) has $\mathcal{L}_2$-gain less than or equal to γ if there exists a continuously differentiable function $S(x)$ that satisfies:

$$S(0) = 0, \qquad S(x) > 0$$
$$\frac{\partial S}{\partial x}(f(x) + G(x)u) \leq \gamma^2 u^T u - h(x)^T h(x) \quad \forall x, u$$

Proof: A derivation and an introduction to dissipation inequalities are given in [3]. ■

In the next section an algorithmic method for constructing polynomial Storage functions used in Theorem 2 is described for systems that evolve according to polynomial vector fields. The remainder of the paper then focusses on extending this method to high dimensional systems by employing a dynamical system decomposition approach.

B. Sum of Squares Programming

For the restricted case of polynomial systems, energy based stability proofs such as Lyapunov and Storage functions that rely on dissipation-like inequalities reduce to polynomial non-negativity tests. Such tests are known to $\mathcal{NP}$-hard when the degree of the polynomial is greater than 4. In this work the sum of squares decomposition provides a tractable, algorithmic method for constructing Storage functions.

A multivariate polynomial $p(x) \in \mathbb{R}[x_1, \ldots, x_n]$, (where $\mathbb{R}[x]$ is the ring of polynomials with real coefficients) is a sum of squares if it can be written as $p(x) = \sum p_i^2(x)$ for some polynomials $p_i(x)$. It is clear from this definition that a SOS polynomial is therefore non-negative. The test to determine if $p(x)$ admits an SOS decomposition is based on an equivalent formulation: A polynomial $p(x)$ of degree $2d$ is SOS if and only if there exists a vector of monomials $Z(x)$ up to degree d and a matrix $Q \succeq 0$ such that

$$p(x) = Z(x)^T Q Z(x). \tag{6}$$

The monomials in Z will not be algebraically independent, therefore the elements in Q will be affinely related to each other. Finding $Q \succeq 0$ that satisfies the above relationship is a Linear Matrix Inequality (LMI) feasibility problem and is solved using semidefinite programming. The task of ensuring that a polynomial with unknown coefficients admits an SOS decomposition is also a semidefinite programme, for this reason it is possible to construct Lyapunov and Storage functions by solving an appropriate optimization problem.

The freely available MATLAB toolbox SOSTOOLS [13] can be used in conjunction with the SDP solver SeDuMi [14] to formulate and solve all SOS programmes in this work.

Before proceeding, the region of interest of the state space must be formally described as system (1) is likely to have multiple equilibria thus the following SOS programmes need to hold only locally. It is assumed without loss of generality that we are interested in the equilibrium point at the origin and we want to ensure that the Storage function satisfies the constraints described within a domain $\mathcal{D} = \left\{x \in \mathbb{R}^n \mid \|x\|^2 < \epsilon\right\}$. Writing this as a polynomial inequality we have

$$r(x) \triangleq x^T x - \epsilon \leq 0, \quad \epsilon > 0. \tag{7}$$

The following sum of squares programme illustrates how Theorem 2 can be implemented using the ideas described above to calculate an upper bound on the $\mathcal{L}_2$-gain.

Program 1: Obtaining an upper bound on the $\mathcal{L}_2$-gain of a system of the form (1):

Find a polynomial $S(x)$ *with* $S(0) = 0$
and a positive definite polynomial function $\varphi_1(x), \varphi_2(x)$
and SOS polynomials $p_1(x), p_2(x)$

Solve:

$$\begin{aligned} \text{min.} \quad & \gamma \\ \text{s.t.} \quad & S(x) + r(x)p_1(x) - \varphi_1(x) \text{ is SOS} \\ & -\frac{\partial S}{\partial x}(f(x) + G(x)u) - h(x)^T h(x) \\ & +\gamma u^T u + r(x)p_2(x) \text{ is SOS} \end{aligned}$$

Then the $\mathcal{L}_2$*-gain is less than* $\sqrt{\gamma}$.

For more details on dynamical systems analysis using SOS techniques see [5].

As outlined earlier, the size of the underlying SDP that requires solving grows rapidly as the dimension of the system increases. In the next section we introduce some basic concepts from algebraic graph theory that are required in the sequel. These ideas are used to decompose a dynamical system into smaller interconnected subsystems that are of a suitable size for SOS analysis.

C. Algebraic Graph Theory

In the forthcoming sections we will describe a method for representing a dynamical system of the form $\dot{x} = f(x)$ as a weighted graph denoted $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The graph consists of a set of N vertices $\mathcal{V}$ and M edges $\mathcal{E}$. Individual vertices are denoted $v_i \in \mathcal{V}$ for $i = 1, \ldots, N$ and edges $e = (v_i, v_j) \in \mathcal{E}$. The presence of edge $(v_i, v_j) \in \mathcal{E}$ indicates that there is a directed link from v_i to v_j. In this work we consider directed weighted graphs in which case $(v_i, v_j) \in \mathcal{E}$ does not imply $(v_j, v_i) \in \mathcal{E}$. The *adjacency matrix* of a graph $A(\mathcal{G})$ is an $N \times N$ matrix such that $A_{ij} > 0$ if $(v_i, v_j) \in \mathcal{E}$ else $A_{ij} = 0$. The *Laplacian* matrix of $\mathcal{G}$ defined by

$$L(\mathcal{G}) = \text{diag}(A(\mathcal{G})\mathbf{1}) - A(\mathcal{G}), \tag{8}$$

where $\mathbf{1} = [1, \ldots, 1]^T$, with appropriate dimension. An alternative formulation of the Laplacian matrix for *undirected* graphs that will be used later is given by

$$L(\mathcal{G}) = C(\mathcal{G})C(\mathcal{G})^T \tag{9}$$

where $C(\mathcal{G}) \in \mathbb{R}^{N \times M}$ is the unweighted graph incidence matrix where $C_{ij} = 1$ if edge j enters v_i, -1 if edge j leaves v_i and 0 otherwise. The Laplacian matrix has several properties that make it useful for graph partitioning. In particular it is singular as each row sum is equal to zero. The number of zero eigenvalues of $L(\mathcal{G})$ is equal to the number of connected components in the graph. Associated to the zero eigenvalue is the eigenvector $\mathbf{1}$ of dimension $N \times 1$. We now turn our attention to the graph partitioning problem, for a comprehensive treatment of algebraic graph theory see [15].

The k-way graph partitioning problem seeks to place each vertex in $\mathcal{G}$ into one of k sets in such a manner that the sum of the weights of the edges connecting vertices in different sets is minimized; it is usually desirable to ensure that the number of vertices in each set is equal. In this work we focus on the case where $k = 2$, this is known as graph bisection. Multiple partitions are found by recursively applying the bisection procedure. Formally the problem is to construct k subgraphs $\mathcal{G}_k(\mathcal{V}_k, \mathcal{E}_k)$ such that $\mathcal{E}_k = \{(v_i, v_j) \in \mathcal{E} | v_i, v_j \in \mathcal{V}_k\}$, $\bigcup_{i=1}^k \mathcal{V}_i = \mathcal{V}$ and $\bigcap_{i=1}^k \mathcal{V}_i = \emptyset$.

The k-way partitioning problem is an $\mathcal{NP}$-hard problem, however there are many algorithms that attempt to solve the problem at least sub-optimally. See [16], [17] and the references therein which describe many approaches to the problem. The approach taken here is to use the spectral properties of $\mathcal{G}$ [18], [19] to partition the graph. The formulation of the partitioning problem presented in the next section is for undirected graphs, it is therefore necessary for us to symmetrize our graphs before applying the following algorithms. A brief description of the spectral partitioning algorithm is now given.

1) Spectral Partitioning: Formulating the graph bisection problem as an optimization problem is done as follows: Given a symmetric weighted graph $\mathcal{G}$, assign integer values $z_i = \pm 1$ for $i = 1, \ldots, N$ that define which partition each vertex in $\mathcal{G}$ belongs to, and minimize

$$J(z) = \frac{1}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}(z_i - z_j)^2. \tag{10}$$

where A is the graph adjacency matrix. Clearly we wish to avoid the trivial case where all z_i's have the same sign. Optimization (10) can be reformulated using $L(\mathcal{G})$:

$$\begin{aligned} \text{minimize} \quad & J(z) = \frac{1}{2} z^T L(\mathcal{G}) z \\ \text{subject to} \quad & z_i^2 = 1, \\ & z^T \mathbf{1} \in [-N + 1, N - 1] \end{aligned} \tag{11}$$

The spectral partitioning algorithm approximates (11) by dropping the two constraints. The vector z is aligned with the eigenvector corresponding to the second smallest eigenvalue of $L(\mathcal{G})$[1] denoted by $\hat{\lambda}$. Vertex v_i is than assigned to partition

[1]It is assumed that all graphs are connected.

z_i according to the rule $z_i = \text{sign}(\hat{\lambda}_i)$ for $i = 1, \ldots, N$. When it is desirable to have an equal number of vertices in each partition $\hat{\lambda}_i$ should be sorted into ascending order and the first $N/2$ vertices corresponding to the sorted eigenvector should be assigned to one partition and the remaining vertices to the other.

III. Dynamical System Decomposition

Recent work in [11] described a decomposition algorithm for autonomous nonlinear dynamical systems that was used to construct composite Lyapunov functions. This *energy based decomposition* approach will be described in the sequel and then applied to non-autonomous systems for input-output analysis.

Consider the system

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \tag{12}$$

with the state vector $x \in \mathbb{R}^n$. For large n analyzing the stability properties of (12) using SOS methods is intractable. Thus it is desirable to decompose the system into M interacting subsystems of smaller state dimension. Stability certificates for each of the subsystems are then searched for and then integrated to provide a composite stability certificate for (12). For the case of $M = 2$ the decomposition is as follows (for the sake of clarity we drop the dependence on time from our notation):

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) + g_1(x_1, u_1), \quad y_1 = x_1, && \text{(13a)} \\ \dot{x}_2 &= f_2(x_2) + g_1(x_2, u_2), \quad y_2 = x_2, && \text{(13b)} \\ & u_1 = y_2, u_2 = y_1, && \text{(13c)} \end{aligned}$$

where $x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}$ and $n_1 + n_2 = n$. The functions g_1, g_2 represent the subsystem interactions and $g_1(x_1, 0) = g_2(x_2, 0) = 0$. The objective of the decomposition is to minimize the effect g_i has on the dynamics of $\dot{x}_i = f(x_i)$. When this perturbation is small it is possible to construct Lyapunov functions $V_i(x_i)$ for each $\dot{x}_i = f_i(x_i)$ and then certify stability of (12) through the composite Lyapunov function $V_c = \sum \alpha_i V_i(x_i)$ with $\alpha_i > 0, \forall i$. The details of how to do this are given in [11]. In Section IV an example with three partitions is presented.

A. Energy Based Decomposition

The proposed decomposition algorithm turns (12) into (13) in such a manner that facilitates the construction of a composite Lyapunov function by minimizing the effect the subsystem interactions have on the drift dynamics. This is achieved by describing (12) as a graph where each state is represented by a vertex with edge (v_i, v_j) if $\dot{x}_i$ is a function of x_j. The weights in the adjacency matrix are assigned based on the flow of energy between interacting states. The notion of energy is defined in Step 4 of the algorithm below. The spectral partitioning algorithm then cuts the graph in a manner that minimizes the energy flow between vertices. The algorithm is outlined below:

1) Construct an *unweighted* graph $\mathcal{G}$ of (12) as described above. Create the graph incidence matrix $C(\mathcal{G})$.
2) Let $F \triangleq \left.\frac{\partial f}{\partial x}\right|_{x=x^*}$ and consider the linear dynamical system $\dot{x} = Fx, y = C^T x$, i.e. the output of the system is seen on the edges of the graph.
3) Calculate the initial condition $\hat{x}$ that will produce the largest energy at the output (maximum $\mathcal{L}_2$ gain). This is done by solving the Lyapunov equation
$$F^T P + PF = -CC^T \tag{14}$$
for $P \succ 0$, $\hat{x}$ is then the unit vector aligned with the eigenvector corresponding to the largest eigenvalue of P.
4) The weighted energy matrix $W(\mathcal{G})$ is then constructed according to
$$W_{ij} \triangleq \|y\|^2_{\mathcal{L}_2} = \hat{x}^T \Psi_{ij} \hat{x} \tag{15}$$
where $\Psi_{ij} \succ 0$ solves the Lyapunov equation $F^T \Psi_{ij} + \Psi_{ij} F = -\delta_i \delta_j^T$ where δ_i denotes the i^{th} column of C.
5) Form the Laplacian: $L(\mathcal{G}) = \text{diag}(W(\mathcal{G})\mathbf{1}) - W(\mathcal{G})$.
6) Symmetrize the Laplacian by setting $\widehat{L}(\mathcal{G}) = \frac{1}{2}(L(\mathcal{G}) + L(\mathcal{G})^T)$ and apply the spectral partitioning algorithm on $\widehat{L}(\mathcal{G})$.

The nonlinear system is then decomposed according to the partition of the vector z. Each subsystem can then be analyzed individually, integrating the results at the end.

B. Non-autonomous System Decomposition

In this section we apply the decomposition approach to nonlinear systems with the aim of estimating their $\mathcal{L}_2$-gain. The systems of interest may have nonlinear dynamics, as well as nonlinear input and output maps. The decomposition algorithm will break system (1) into M subsystems, for $M = 2$ the interacting subsystems are given by

$$\begin{aligned} \dot{x}_1 &= f_{11}(x_1) + f_{12}(x_1, z_2) + g_{11}(x_1)u + g_{12}(x_1, z_2)u \\ \dot{x}_2 &= f_{22}(x_2) + f_{21}(x_2, z_1) + g_{22}(x_2)u + g_{21}(x_1, z_2)u \\ z_1 &= x_1 \\ z_2 &= x_2 \\ y &= h_1(x_1) + h_2(x_2) + h_{12}(x_1, x_2). \end{aligned} \tag{16}$$

Here u is the external input, y is the external output, $h_{12}(x_1, x_2) = h(x) - h_1(x_1) - h_2(x_2)$ and z_i, z_j provide the internal system coupling. This is shown as a block diagram in Figure 1.

Of course, trying to analyze (16) is still difficult computationally as there is a lot of subsystem interaction. Instead of working with (16) directly we assume that the decomposition has minimized the effect of f_{12}, f_{21}, g_{12} and g_{21} on the dynamics thus we can ignore them (temporarily) and focus on the two systems which are uncoupled (at least in state):

$$\begin{aligned} \dot{x}_1 &= f_{11}(x_1) + g_{11}(x_1)u, \quad y_1 = h_1(x_1), && \text{(17a)} \\ \dot{x}_2 &= f_{22}(x_2) + g_{22}(x_2)u, \quad y_2 = h_2(x_2), && \text{(17b)} \end{aligned}$$

these system representations are of the same form as (1) which we can analyze using SOS Program 1 when the state dimension is sufficiently small. The main goal however is to obtain a bound on the $\mathcal{L}_2$-gain of (1) with a larger state

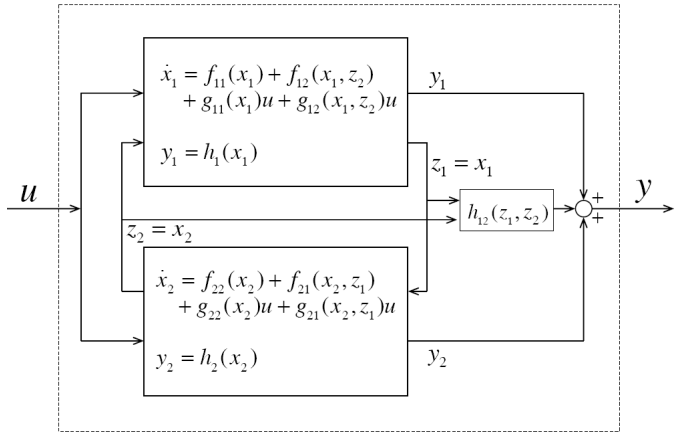

Fig. 1. Decomposition of a non-autonomous dynamical system given by (16).

dimension through the lower dimensional subsystems (17) obtained by applying the energy based decomposition. This will be the focus of the next section.

C. $\mathcal{L}_2$ Gain Estimation via System Decomposition

The proceeding algorithm can be used to obtain an upper bound of the $\mathcal{L}_2$ gain of system (1):

1) Apply the decomposition algorithm from Section III-A to the autonomous part ($\dot{x} = f(x)$) of (1).
2) Use the resulting state partition to construct system (16).
3) Use SOS programme 1 to construct Storage functions S_i and γ_i for $i = 1, \dots M$ corresponding to each subsystem in (17).
4) Solve the following SOS program:

Program 2: Find $\alpha_i > 0$ for $i = 1, \dots M$, an SOS polynomial $p(x)$ and define $r(x)$ as in (7)
$S_c := \sum_{i=1}^{M} \alpha_i S_i$

$$\begin{aligned} \text{min.} \quad & \zeta \\ \text{s.t.} \quad & \zeta > 0 \\ & -\frac{\partial S_c}{\partial x}(f(x) + G(x)u) - h(x)^T h(x) + \zeta u^T u \\ & + r(x)p(x) \quad \text{is SOS} \end{aligned} \tag{18}$$

5) $\sqrt{\zeta}$ is an upper bound on the $\mathcal{L}_2$ gain of (1).

Remark 1: For systems described by high order vector fields or with large state dimension SOS program 1 will be computationally intractable. In comparison, the integration stage of the decomposition approach (step 4) is essentially a search for scalar multipliers that verify a polynomial is SOS, as the Storage functions have already been constructed in the previous step.

IV. Example

Two examples are presented in this section that use the decomposition algorithm to obtain $\mathcal{L}_2$-gain bounds. The first example provides a comparison between the bound obtained using a direct method and the bound achieved using the decomposition algorithm. The second example uses the decomposition approach to obtain a bound on a system that is too large for a direct method to be applied.

A. Comparison System

In the first example we consider a fourth order system with a quartic vector field:

$$\begin{aligned} \dot{x}_1 &= -3x_1 - x_1x_2^2 + x_1x_4 + 2x_1x_4u - x_1x_2u \\ \dot{x}_2 &= -2x_1^2 + 0.5x_1 - x_1^2x_2^2 - 3x_2 - 4x_1x_3u \\ \dot{x}_3 &= -x_3^4 - x_3^2x_4^2 + 1.5x_4 - x_3 - x_4u \\ \dot{x}_4 &= -x_4^2 - 4x_3 - 2x_1 + x_3x_4u \\ y_1 &= x_1 + x_2 \\ y_2 &= x_4 \end{aligned} \tag{19}$$

This system is sufficiently small to be analyzed directly allowing us to make a direct comparison between gain estimates and the size of the optimization problems. Sum of squares programming was first used to construct a quadratic Lyapunov function that certifies local asymptotic stability of (19) with $u = 0$. The SOS decomposition of the Lyapunov $V(x) = Z(x)^T Q Z(x)$ is $Z(x) = [x_4, x_3, x_2, x_1]^T$,

$$Q = \begin{bmatrix} 0.0199 & -0.0059 & 0.0008 & -0.0030 \\ -0.0059 & 0.0384 & -0.0006 & 0.0059 \\ 0.0008 & -0.0006 & 0.0065 & 0.0003 \\ -0.0030 & 0.0059 & 0.0003 & 0.0053 \end{bmatrix}.$$

Implementing SOS Program 1 we obtain an upper bound on the $\mathcal{L}_2$ gain of (19) of 0.0907 (to 4dp). Applying the decomposition algorithm to (19) with $u = 0$, the partitions $\mathcal{X}_1 = [x_1, x_2]$, $\mathcal{X}_2 = [x_3, x_4]$ are obtained resulting in the following decomposition:

$$\begin{aligned} f_{11} &= \begin{pmatrix} -3x_1 - x_1x_2^2 \\ -2x_1^2 + 0.5x_1 - x_1^2x_2^2 - 3x_2 \end{pmatrix} \\ g_{11} &= \begin{pmatrix} x_1 - x_2x_1 \\ 0 \end{pmatrix} \quad g_{12} = \begin{pmatrix} -x_4 \\ x_3x_4 \end{pmatrix} \\ f_{22} &= \begin{pmatrix} -x_3^4 - x_4^2x_3^2 + 1.5x_4 - x_3 \\ -x_4^2 - 4x_3 \end{pmatrix} \\ g_{21} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad g_{22} = \begin{pmatrix} -x_4 \\ x_3x_4 \end{pmatrix} \\ h_1 &= x_1 + x_2 \quad h_2 = x_4 \quad h_{12} = 0. \end{aligned}$$

Storage functions $S_1(\mathcal{X}_1)$ and $S_2(\mathcal{X}_2)$ are then constructed for the decoupled systems (17) where

$$\begin{aligned} S_1(\mathcal{X}_1) &= 1.427x_2^2 + 0.018x_1x_2 + 0.503x_1^2 \\ S_2(\mathcal{X}_2) &= 1.168x_4^2 - 1.037x_3x_4 + 3.113x_3^2. \end{aligned}$$

The multipliers $\alpha_1 = 2.1505$ and $\alpha_2 = 1$ are found as described by the methods in Section III-C that produce composite Storage function $S_c = \alpha_1 S_1 + \alpha_2 S_2$ for system (1) with an upper $\mathcal{L}_2$-gain bound of 0.1711 (to 4dp). This bound can be tightened at the expense of searching for higher order Storage functions, keeping S_1 and using a quartic Storage function S_2 a bound of 0.162 is found.

$S(x)$ construction	Size of LMI	CPU time
Full system	2796×457	1.94s
Subsystem for $\mathcal{X}_1$	241×66	0.19s
Subsystem for $\mathcal{X}_2$	241×66	0.19s
Composite verification	1423×244	0.89s

TABLE I

COMPUTATIONAL COST OF CONSTRUCTING STORAGE FUNCTIONS AND ESTIMATING $\mathcal{L}_2$ FOR THE SUBSYSTEMS, THE COMPOSITE SYSTEM AND THE FULL SYSTEM.

In Table 1 the computational cost of the direct approach is compared with the decomposition approach. It can be seen that the decomposition approach takes a total of 1.27s to compute the subsystem Storage functions and compute the composite Storage function, in comparison a direct method takes 1.94s and requires the solution of a more complex SDP. Note that the second column in Table 1 displays *LMI dimension* $\times$ *no. constraints*. In the next example a bound will be obtained using the decomposition approach on a system that is too large for direct methods to be applied.

B. Lotka-Volterra Dynamical System

The second example illustrates the practical application of the decomposition based approach. We consider a modified version of the Lotka-Volterra example system given in [11]. In its standard form there are known scalable methods for constructing stability certificates based on diagonal Lyapunov functions. However these methods are only valid when the system is dissipative, in the example we consider, this assumption is not met.

The modified system which now includes inputs and outputs is given below

$$\begin{aligned} \dot{x}_i &= x_i \left(b_i - x_i - \sum_{j=1}^{n} A_{ij} x_j \right) + \sum_{k=1}^{m} G_{ik}(x) u_k \\ y &= h(x) \end{aligned}$$

where $u \in \mathbb{R}^m$ is the input vector and $G \in \mathbb{R}^{n \times m}$ describes the input-state interactions. It is assumed that the input signals all belong to $\mathcal{L}_2$. The system has 16 states, 2 inputs and 2 outputs (1 linear, 1 nonlinear). The decomposition algorithm breaks the system into 3 subsystems and successfully finds 3 Storage functions for the subsystems. A composite Storage function is then found and a bound on the $\mathcal{L}_2$ gain is obtained.

While the details of the dynamics have not been presented in order to save space, the relevance of this result is that a direct approach cannot be used to assess the gain of a system of this size. In this instance the solver could not even parse the problem without running out of memory. In comparison the decomposition approach required the solution of two LMIs of dimension 7100×7100 subject to 1030 constraints and one of 15606×15606 with 2009 constraints for the individual Storage functions. Each of these was solved in under $15s$ on a 3.33GHz processor with 2.5Gb of memory. To verify that the composite Storage function is a Storage function for the original system an 23975×23975 LMI with 5135 constraints was solved in 15mins.

V. CONCLUSION

A method for obtaining bounds on the $\mathcal{L}_2$ gain of nonlinear systems via dynamical system decomposition has been presented. The decomposition approach is particularly suitable when combined with SOS programming as Storage functions can be algorithmically constructed. Furthermore, system decomposition actually extends the application of SOS programming to systems analysis to include systems with a higher state dimension than a direct SOS approach can handle.

It was observed that the upper bounds achieved using this approach may be conservative in cases where a direct approach is possible. However, for practical purposes conservativeness is preferable to intractability which is many times the case for large systems. Future work will seek to reduce conservativeness in the obtainable bounds. It would also be interesting to extend the decomposition approach to deal with uncertain systems.

REFERENCES

[1] A. J. van der Schaft, "On a state space approach to nonlinear H_∞ control," *Systems & Control Letters*, vol. 16, no. 1, pp. 1 – 8, 1991.

[2] ——, "L_2-gain analysis of nonlinear systems and nonlinear state-feedback H_∞ control," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 770 –784, 1992.

[3] C. Ebenbauer, T. Raff, and F. Allgöwer, "Dissipation inequalities in systems theory: An introduction and recent results," in *Proceedings of the International Congress on Industrial and Applied Mathematics 2007*, vol. Invited Lectures of the International Congress on Industrial and Applied Mathematics 2007. European Mathematical Society Publishing House, 2009, pp. 23–42.

[4] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, no. 2, pp. 11–129, 1987.

[5] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Proceedings IEEE Conference on Decision and Control*, 2002.

[6] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, Caltech, Pasadena, CA, 2000.

[7] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[8] A. Papachristodoulou and C. Papageorgiou, "Robust stability and performance analysis of a longitudinal aircraft model using sum of squares techniques," in *Proceedings of the Mediterranean Conference on Control and Automation*, 2005.

[9] W. Tan, A. Packard, and T. Wheeler, "Local gain analysis of nonlinear systems," in *Proceedings of the American Control Conference*, 2006.

[10] P. Parrilo, "Exploiting algebraic structure in sum of squares programming," 2005, in *Positive Polynomials in Control*, Springer-Verlag.

[11] J. Anderson and A. Papachristodoulou, "A network decomposition approach for efficient sum-of-squares programming based analysis," in *To appear: Proceedings of the 2010 American Contol Conference*, 2010.

[12] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, Inc., 2001.

[13] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "SOSTOOLS – Sum of Squares Optimization Toolbox, User's Guide," 2002, available at `http://www.eng.ox.ac.uk/control/sostools/`.

[14] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999.

[15] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag, Inc., 2000.

[16] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[17] S. E. Schaeffer, "Graph clustering," *Computer Science Review I*, pp. 27–64, 2007.
[18] M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematical Journal*, vol. 25, pp. 619–633, 1975.
[19] F. R. K. Chung, *Spectral Graph Theory*, ser. Regional conference series in mathematics. American Mathematical Society, 1997, no. 92.