2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Structured Sum of Squares for Networked Systems Analysis

Edward J. Hancock and Antonis Papachristodoulou

Abstract— **In this paper we introduce a structured Sum of Squares technique that enables Sum of Squares programming to be applied to networked systems analysis. By taking the structure of the network into account, we limit the size and number of decision variables in the LMI representation of the Sum of Squares, which improves the scalability of the technique for networked systems beyond taking advantage of symmetry and sparsity. We apply the technique to test non-negativity of fourth order structured polynomials in many variables and show that for these problems the technique has improved scalability over existing Sum of Squares techniques.**

I. Introduction

Dynamical networks are prevalent in many areas of science and technology; a number of techniques have been proposed to analyse these large scale systems, see, for example, [24], [13] and the references therein. However, to translate theoretical concepts on dynamical networks into useful tools, new computational techniques need to be developed. In this paper we introduce a structured Sum of Squares technique in order to apply Sum of Squares programming to networks, which allows analysis of problems which would otherwise be computationally infeasible. We apply the technique to testing non-negativity of fourth order polynomials in several variables but with a structure inherited by the networked system, in order to show the practical implementation of the technique. We show that the structured Sum of Squares scales significantly better than existing Sum of Squares techniques.

Sum of Squares (SOS) programming is a useful technique for showing the non-negativity of polynomials [18], [19]. See [17] for a tutorial and [4] for a recent review. It has extensive applications in systems theory for systems with polynomial or rational vector fields, such as finding Lyapunov functions for nonlinear systems [18], [15]. Sum of Squares techniques can also be extended to any form of smooth vector field using Generalised Sector concepts [5], polynomial approximations [3] or by recasting the problem into polynomial form [16]. The way these techniques work is by translating the polynomial representation into a quadratic form; the positivity test can then be formulated as a Linear Matrix Inequality (LMI). This LMI can be tested using efficient Semi Definite Programming (SDP) techniques such as interior point methods [18].

Although SOS programming is very useful for analysing polynomial systems, current Sum of Squares techniques do not scale well for large-scale systems. The factor limiting the scaling of the algorithm to larger systems is the size of the resulting SDP, which is a combination of the size of the respective LMI and the number of decision variables. If the polynomial is 'sparse', i.e. many of the monomials have zero coefficients, then the size of the matrix and number of decision variables can be reduced significantly [9], [23]. Structured Sum of Squares relaxations have been developed specifically for sensor networks using the sparsity of the LMI [14]. There are also a number of techniques to reduce the size of the LMI built into the preprocessing of current SOS computational tools [12].

In this paper we develop structured Sum of Squares techniques for networks by using the structure of the network to limit both the size of the LMI and number of decision variables in the resulting LMI representation. This is important as the flop cost of interior point methods is approximately linear with respect to the size of the LMI while polynomial with respect to the number of decision variables [1]. The technique improves the scalability of the Sum of Squares for networked systems without necessarily assuming sparsity or symmetry.

In the examples section we demonstrate how the technique can be applied to the problem of testing non-negativity of fourth order, structured polynomials in several variables and find that scalability depends upon both the number of nodes and the number of links in the network. For globally coupled networks we apply the technique for up to 50 variables. For 1-D lattices we apply the technique for up to 400 variables. In comparison, existing Sum of Squares techniques are found to be impractical for networks with 15-20 variables.

This paper is organised as follows. In Section II we give background and motivation. In Section III we develop the structured Sum of Squares technique. In Section IV we apply the developed technique to two examples.

II. Background and Motivation

A. Background: Sum of Squares

Sum of Squares is a useful technique for showing whether polynomial functions are non-negative. We often want to know whether a function $p(x)$ is non-negative, i.e., $p(x) \geq 0$ but this problem is NP hard even when $p(x)$ is a polynomial of degree greater than or equal to 4 [18]. However, if we show that a function is a Sum of Squares then we know that it is non-negative. A polynomial $p(x) \in \mathbb{R}[x]$ is a sum of squares if

$$p(x) = \sum_{i=1}^{m} h_i(x)^2 \tag{1}$$

This work was supported by the Clarendon Fund at Oxford University and the Overseas Research Student Award Scheme (UK Government).

E. J. Hancock and A. Papachristodoulou are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK. Email: {edward.hancock,antonis}@eng.ox.ac.uk

for some other polynomials $h_i(x)$, which implies that $p(x) \geq 0$. Thus if a polynomial is a Sum of Squares then it is non-negative.

If $p(x) \in \mathbb{R}[x]$ is a polynomial of degree $2d$, where $x \in \mathbb{R}^n$, then

$$p(x) = Z(x)^T Q(\lambda) Z(x) \tag{2}$$

where $Z(x)$ is composed of monomials of the vector x up to degree d and $Q \in \mathbb{R}^{m \times m}$ is affinely parameterised by decision variables $\lambda \in \mathbb{R}^r$. The decision variables are due to the non-uniqueness of Q in the representation of $p(x)$ in the quadratic form (2), as the elements of $Z(x)$ are not necessarily independent of each other. If a value of λ can be found such that $Q \succeq 0$ then $p(x)$ is a Sum of Squares – this can be seen by taking a Cholesky decomposition of $Q = LL^T$ such that

$$p(x) = Z(x)^T L L^T Z(x) = \sum_{i=1}^{m} h_i(x)^2 \tag{3}$$

where $h(x) = L^T Z(x)$.

In (2), Q is affinely parameterised by the decision variable λ and since we require $Q \succeq 0$, the whole problem can be viewed as an LMI. Thus Semi-Definite Programming can be used to find whether there exists a value of the decision variable such that $Q \succeq 0$. This can be implemented using toolboxes such as SOSTOOLS [20].

Sum of Squares only show that a polynonial is non-negative and so if we wish to determine whether a polynomial is positive definite, as in the case for finding Lyapunov functions, then we have to modify the technique. For example, to show that $V(x) \in \mathbb{R}[x]$ is positive definite on D it suffices to show that $V(0) = 0$ and

$$V(x) - \phi(x) \text{ is a SOS} \tag{4}$$

where $\phi(x)$ is positive definite on D.

If we wish to test $p(x)$ on a restricted domain then we can use Positivestellensatz [18], which can be viewed as a generalisation of the S-procedure, to show that a function is non-negative on a region, rather than globally. We can show that $p(x) \geq 0$ when x satisfies $\gamma(x) \leq 0$ if we can find sum of squares $\beta(x)$ as a multiplier such that

$$p(x) + \beta(x)\gamma(x) \text{ is a SOS} \tag{5}$$

See [18] for a more general version of Postivestellensatz for SOS.

Sum of Squares programming is an efficient method for small problems. However, the technique does not scale well to large problems. This is the motivation for the technique of Structured Sum of Squares that we develop in this paper.

B. Motivation: Lyapunov function candidates with a Network Structure

In this paper we aim to develop computational techniques to analyse networked systems with naturally highly structured Lyapunov function candidates, such as 'Damped Hamiltonian' or Gradient systems. Many types of networks fit into this class of system, such as power systems [2], synchronization models [6], [7], population dynamics [21] and neural networks [8], [10]. The advantage of systems of this form is that there exists a natural Lyapunov function candidate which is often highly structured and whose derivative with respect to time is typically globally non-positive. Thus if we can show that the Gradient or Hamiltonian function is positive definite then we can show that the system is stable.

An example of this case is a network of Duffing oscillators [8]. This can be written with 3rd order polynomial coupling as

$$\begin{aligned} \dot{z}_i &= -\gamma_i z_i - a_i(y_i - y_i^3) + \sum_{j=1}^{N} b_{ij}(y_j - y_i)^3 \\ \dot{y}_i &= z_i \end{aligned} \tag{6}$$

where $i = 1, \ldots, N$, $b_{ij} = b_{ji}$ and $b_{ii} = 0$. If we let

$$\begin{aligned} V = \sum_{i=1}^{N} \left[\frac{1}{2} z_i^2 + a_i \left(\frac{1}{2} y_i^2 - \frac{1}{4} y_i^4 \right) \right] \\ + \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} b_{ik} \frac{1}{4} (y_k - y_i)^4 \end{aligned} \tag{7}$$

it can be shown (See Appendix) that

$$\dot{V} = -\sum_{i=1}^{N} \gamma_i z_i^2 \leq 0 \tag{8}$$

We can see by observation that $-\dot{V}$ is a Sum of Squares and so we only need to show that V is positive definite for a domain containing the origin to show that the origin is stable [8]. It can be seen that (7) is highly structured and it is with this problem as motivation that we develop Structured Sum of Squares techniques.

III. Structured Sum of Squares

In this section we impose a structure on Sum of Squares in order for the technique to scale better for polynomials with a network structure. Firstly, we remove all monomials and decision variables associated with any missing link in the network and secondly, we assume that the coupling terms are of a certain form which allows us to choose decision variables based on network weightings.

A. Structured SOS

We use the general definition of structured Sum of Squares to mean that we will use the structure of the polynomial of interest to limit either the size of the matrix Q in (2) and/or the number of decision variables in the traditional Sum of Squares programming technique.

Suppose we wish to study whether the polynomial $p = p(x_1, \ldots, x_N) \in \mathbb{R}$ is Sum of Squares, where $x_i \in \mathbb{R}^n$ for $i = 1, \ldots, N$. Now

$$\begin{aligned} p(x_1, \ldots, x_N) = \sum_{i=1}^{N} p_i(x_i) + \sum_{i=1}^{N} \sum_{k=1}^{N} p_{ik}(x_i, x_k) \\ + \sum_{i=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} p_{ikl}(x_i, x_k, x_l) + \ldots \end{aligned} \tag{9}$$

For a networked system we often have functions of the structured form

$$p(x_1,\dots,x_N)=\sum_{i=1}^{N}p_i(x_i)+\sum_{i=1}^{N}\sum_{k=1}^{N}b_{ik}p_{ik}(x_i,x_k) \quad (10)$$

where $b_{ii}=0$, b_{ik} is non-zero if the subsystems with states x_i and x_k communicate on a network and zero otherwise.

For simplicity, we assume symmetry of the network (i.e. $b_{ik}=b_{ki}$) and we restrict analysis to systems with $p_i(0)=0$ and $p_{ik}(0,0)=0$ for all $i,k=1,\dots,N$. These assumptions are not required for structured Sum of Squares but allow it to be more easily derived and presented.

B. Structured SOS with parametrised coupling matrices

We look at the case where that the coupling term is of the same form and so analyse polynomials of the form

$$p(x_1,\dots,x_N)=\sum_{i=1}^{N}p_i(x_i)+\sum_{i=1}^{N}\sum_{k=1}^{N}b_{ik}p_c(x_i,x_k) \quad (11)$$

We can see that the structure of the Hamiltonian function (7) is of the form (11).

We first define

$$p_i(x_i)=Z_i(x_i)^TP_iZ_i(x_i) \quad (12)$$

for $i=1,\dots,N$ where $Z_i(x_i)\in\mathbb{R}^m$ is a vector of monomials and the symmetric matrix $P_i=P_i(\lambda_i)\in\mathbb{R}^{m\times m}$ is affinely parameterised by decision variables λ_i as in typical Gram matrix representations. Furthermore, we write

$$p_c(x_i,x_k)=Z_c(x_i,x_k)^TQ_cZ_c(x_i,x_k) \quad (13)$$

for $i,k=1,\dots,N$, where $Z_c(x_i,x_k)$ is a vector of monomials in x_i and x_k and $Q_c=Q_c(\lambda_c)$ is affinely parameterised by decision variables λ_c as in typical Gram matrix representations. Importantly, note that Q_c and λ_c are independent of i and k.

Now we split the above representation further such that

$$Q_c=\begin{bmatrix}Q_c^{11}&Q_c^{12}&Q_c^{13}\\Q_c^{21}&Q_c^{22}&Q_c^{23}\\Q_c^{31}&Q_c^{32}&Q_c^{33}\end{bmatrix},Z_c(x_i,x_k)=\begin{bmatrix}Z_i\\Z_k\\Z_{ik}\end{bmatrix} \quad (14)$$

where $Q_c^{12}=(Q_c^{21})^T$, $Q_c^{13}=(Q_c^{31})^T$, $Q_c^{23}=(Q_c^{32})^T$, $Z_i\in\mathbb{R}^m$ contains all of the monomials which are only a function of x_i and $Z_{ik}\in\mathbb{R}^{m_c}$ contains monomials which are a combination of both x_i and x_k. For a standard representation Z_{ik} and Z_{ki} have the same monomial elements but different ordering.

Using (12) and (14) for the parameterisation of decision variables rather than the standard form (2), we rewrite (11) in the quadratic form

$$p(x_1,\dots,x_N)=Z(x_1,\dots,x_N)^TGZ(x_1,\dots,x_N) \quad (15)$$

We let $d_i=\sum_k b_{ik}$ be the weighted degree of node i, and let L be the number of links in the network, where link l represents the edge between $\{i,k\}$ for $i<k$ as the network is symmetric with no self-loops (i.e. an undirected graph).

We let $c_{i,l}^U=b_{ik}$ if link l represents the link between $\{i,k\}$, where $i<k$, and $c_{i,l}^U=0$ otherwise. We let $c_{i,l}^L=b_{ik}$ if link l represents the link between $\{k,i\}$, where $k<i$, and $c_{i,l}^L=0$ otherwise. We let $C^L=[c_{i,l}^L]$ and $C^U=[c_{i,l}^U]$. We let $\hat{b}_{c,l}=b_{ik}$ if link l represents the link between $\{i,k\}$, with $i<k$. $B=[b_{ik}]$ is the weighted adjacency matrix and $C=C^U+C^L$ is the weighted incidence matrix.

Theorem 1: The polynomial (11) is a Sum of Squares if there exist parameters $\lambda_1,\dots,\lambda_N$ and λ_c and matrices $\bar{Q}_c^{13},\bar{Q}_c^{23},\bar{Q}_c^{33}$ such that

$$G=\begin{bmatrix}G_A&G_B\\G_C&G_D\end{bmatrix}\succeq 0 \quad (16)$$

where, using (12), (13) and (14)

$$\begin{aligned}G_A=&\text{diag}(P_1,\dots,P_N)+\text{diag}(d_1,\dots,d_N)\otimes(Q_c^{11}+Q_c^{22})\\&+B\otimes(Q_c^{12}+(Q_c^{12})^T)\\G_B=G_C^T=&C^U\otimes(Q_c^{13}+\bar{Q}_c^{23})+C^L\otimes(\bar{Q}_c^{13}+Q_c^{23})\\G_D=&\text{diag}(\hat{b}_{c,1},\dots,\hat{b}_{c,L})\otimes(Q_c^{33}+\bar{Q}_c^{33})\\&\bar{Q}_c^{13}Z_{ik}=Q_c^{13}Z_{ki}\\&\bar{Q}_c^{23}Z_{ik}=Q_c^{23}Z_{ki}\\&Z_{ki}^TQ_c^{33}Z_{ki}=Z_{ik}^T\bar{Q}_c^{33}Z_{ik}\end{aligned}$$

Moreover, if $Q_c^{13}=\bar{Q}_c^{13}$, $Q_c^{23}=\bar{Q}_c^{23}$, $Q_c^{33}=\bar{Q}_c^{33}$ then

$$G_B=G_C^T=C\otimes(Q_c^{13}+Q_c^{23})$$

$$G_D=\text{diag}(\hat{b}_{c,1},\dots,\hat{b}_{c,L})\otimes 2Q_c^{33}.$$

Proof: Combining (11)–(14) we obtain

$$\begin{aligned}&p(x_1,\dots,x_N)\\&=\sum_{i=1}^{N}Z_i^TP_iZ_i+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_i^TQ_c^{11}Z_i+b_{ik}Z_kQ_c^{22}Z_k)\\&+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_i^TQ_c^{12}Z_k+b_{ik}Z_k^TQ_c^{21}Z_i)\\&+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_i^TQ_c^{13}Z_{ik}+b_{ik}Z_k^TQ_c^{23}Z_{ik})\\&+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_{ik}^TQ_c^{31}Z_i+b_{ik}Z_{ik}^TQ_c^{32}Z_k+b_{ik}Z_{ik}^TQ_c^{33}Z_{ik})\end{aligned} \quad (17)$$

By switching subscripts and using $b_{ik}=b_{ki}$ then

$$\begin{aligned}p=&\sum_{i=1}^{N}\sum_{k=1}^{N}Z_i^TG_A^{ik}Z_k+\sum_{i=1}^{N}\sum_{k=1}^{N}b_{ik}Z_{ik}^TQ_c^{33}Z_{ik}\\&+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_i^TQ_c^{13}Z_{ik}+b_{ik}Z_k^TQ_c^{23}Z_{ik})\\&+\sum_{i=1}^{N}\sum_{k=1}^{N}(b_{ik}Z_{ik}^TQ_c^{31}Z_i+b_{ik}Z_{ik}^TQ_c^{32}Z_k)\end{aligned} \quad (18)$$

where

$$G_A^{ik}=\delta_{ik}P_i+\delta_{ik}d_i(Q_c^{11}+Q_c^{22})+b_{ik}(Q_c^{12}+(Q_c^{12})^T) \quad (19)$$

and δ_{ik} is the Kronecker delta. We next remove any monomials which are repeated in both Z_{ik} and Z_{ki}. Now

$$\begin{aligned}
&\sum_{i=1}^{N}\sum_{k=1}^{N} b_{ik} Z_i^T Q_c^{13} Z_{ik} \\
&= \sum_{i=1}^{N}\sum_{k>i}^{N} b_{ik} Z_i^T Q_c^{13} Z_{ik} + \sum_{k=1}^{N}\sum_{i>k}^{N} b_{ik} Z_i^T Q_c^{13} Z_{ik} \\
&= \sum_{i=1}^{N}\sum_{k>i}^{N} (b_{ik} Z_i^T Q_c^{13} Z_{ik} + b_{ik} Z_k^T Q_c^{13} Z_{ki}) \\
&= \sum_{i=1}^{N}\sum_{k>i}^{N} (b_{ik} Z_i^T Q_c^{13} Z_{ik} + b_{ik} Z_k^T \bar{Q}_c^{13} Z_{ik})
\end{aligned} \tag{20}$$

Thus by changing summations and switching subscripts we obtain

$$\begin{aligned}
p = &\sum_{i=1}^{N}\sum_{k=1}^{N} Z_i^T G_A^{ik} Z_k + \sum_{i=1}^{N}\sum_{k>i}^{N} b_{ik} Z_{ik}^T [Q_c^{33} + \bar{Q}_c^{33}] Z_{ik} \\
&+ \sum_{i=1}^{N}\sum_{k>i}^{N} (b_{ik} Z_i^T (Q_c^{13} + \bar{Q}_c^{23}) Z_{ik} + b_{ik} Z_k^T (\bar{Q}_c^{13} + Q_c^{23}) Z_{ik}) \\
&+ \sum_{i=1}^{N}\sum_{k>i}^{N} (b_{ik} Z_{ik}^T (Q_c^{31} + \bar{Q}_c^{32}) Z_i + b_{ik} Z_{ik}^T (\bar{Q}_c^{31} + Q_c^{32}) Z_k)
\end{aligned} \tag{21}$$

We next remove all Z_{ik} for which $b_{ik} = 0$. If we let $Z_{c,l}$ represent Z_{ik}, where l represents the link between $\{i, k\}$, with $i < k$, then

$$\begin{aligned}
p = &\sum_{i=1}^{N}\sum_{k=1}^{N} Z_i^T G_A^{ik} Z_k + \sum_{l=1}^{L} \hat{b}_{c,l} Z_{c,l}^T (Q_c^{33} + \bar{Q}_c^{33}) Z_{c,l} \\
&+ \sum_{i=1}^{N}\sum_{l=1}^{L} Z_i^T (c_{i,l}^U (Q_c^{13} + \bar{Q}_c^{23}) + c_{i,l}^L (\bar{Q}_c^{13} + Q_c^{23})) Z_{c,l} \\
&+ \sum_{i=1}^{N}\sum_{l=1}^{L} Z_{c,l}^T (c_{l,i}^U (Q_c^{31} + \bar{Q}_c^{32}) + c_{l,i}^L (\bar{Q}_c^{31} + Q_c^{32})) Z_i
\end{aligned} \tag{22}$$

where $(C^U)^T = [c_{l,i}^U]$ and $(C^L)^T = [c_{l,i}^L]$.

Thus p is a Sum of Squares if there exist decision variables such that $G \succeq 0$ where

$$\begin{aligned}
p(x_1, \ldots, x_N) &= Z(x_1, \ldots, x_N)^T G Z(x_1, \ldots, x_N) \\
Z^T &= (Z_1^T, \ldots, Z_N^T, Z_{c,1}^T, \ldots, Z_{c,L}^T)
\end{aligned} \tag{23}$$

■

With the representation we have just presented, without any preprocessing we have

$$\text{Size of LMI } = N * \text{Dim}(Z_i) + L * \text{Dim}(Z_{ik}) \tag{24}$$

$$\text{Number of Decision Variables } = N * \text{Dim}(\lambda_i) + \text{Dim}(\lambda_c) \tag{25}$$

We can see that the size of the LMI increases and the number of decision variables remains constant as the number of links in the network increases. In contrast, if we analyse (15) without using its structure to reduce the size of the LMI with

$$Z^T = (Z_1^T, \ldots, Z_N^T, Z_{1,2}^T, Z_{1,3}^T, \ldots, Z_{N-2,N}^T, Z_{N-1,N}^T) \tag{26}$$

then before preprocessing we have

$$\text{Size of LMI } = N*\text{Dim}(Z_i) + \frac{1}{2}N(N-1)*\text{Dim}(Z_{ik}) \tag{27}$$

$$\text{Number of Dec. Var. } = N*\text{Dim}(\lambda_i) + \frac{1}{2}N(N-1)*\text{Dim}(\lambda_c) \tag{28}$$

Thus we have significantly reduced the number of decision variables and the size of the LMI by introducing a structure to the Sum of Squares problem. As mentioned in the introduction, this is important as the flop cost of interior point methods is approximately linear with respect to the size of the LMI and polynomial with respect to the number of decision variables [1]. This allows the SOS algorithm to better scale for polynomials with a network structure.

IV. Examples

In the following section we give details of structured Sum of Squares computations which were carried out to analyse 4th order polynomials with a network structure.

We analysed globally coupled and 1-D lattice networks; these are the two extremes of the number of links in a connected network for a constant number of nodes. The technique was tested using symmetric networks with node and link weightings which were not identical. The CPU time t_{cpu} and Wall Clock time t_{clock} for the computations were recorded and, as the network weightings were random, the times were averaged over a number of runs. The SDP solution was carried out using SeDuMi 1.3 [22] on MATLAB (R2011a) through a YALMIP interface [11] on a 2.66GHz Intel Core i7-M620 CPU (dual core).

To analyse a 4th order polynomial where $x_i \in \mathbb{R}$ for $i = 1, \ldots, N$, the standard vectors of monomials are

$$Z_i(x_i) = (x_i, x_i^2), Z_{ik} = (x_i x_k) \tag{29}$$

From these monomial vectors Q_c is parameterised using the dependencies between monomials of $(x_i^2)(x_k^2) = (x_i x_k)^2$, $(x_i^2)(x_k) = (x_i)(x_i x_k)$ and $(x_k^2)(x_i) = (x_k)(x_i x_k)$. Also, $Z_{ik} = Z_{ki}$ and so $Q_{13} = \bar{Q}_{13}$, $Q_{23} = \bar{Q}_{23}$ and $Q_{33} = \bar{Q}_{33}$.

Example 2: We first tested the technique on the example

$$p = \sum_{i=1}^{N} a_i (x_i^2 + x_i^4) - \sum_{i=1}^{N}\sum_{k=1}^{N} b_{ik} x_i^2 x_k^2 \tag{30}$$

where $x_i \in \mathbb{R}$ for all i. Now

$$P_i = a_i \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{31}$$

and

$$Q_c = \begin{bmatrix} 0 & 0 & 0 & \lambda_{c2} & -\lambda_{c3} \\ 0 & 0 & \lambda_{c3} & -\lambda_{c1} & 0 \\ 0 & \lambda_{c3} & 0 & 0 & -\lambda_{c2} \\ \lambda_{c2} & -\lambda_{c1} & 0 & 0 & 0 \\ -\lambda_{c3} & 0 & -\lambda_{c2} & 0 & -1 + 2\lambda_{c1} \end{bmatrix}$$

where $\lambda_c = (\lambda_{c1}, \lambda_{c2}, \lambda_{c3})$ are decision variables in the SDP. Thus

$$Q_c^{11} = Q_c^{22} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, Q_c^{12} = \begin{bmatrix} 0 & \lambda_{c2} \\ \lambda_{c3} & -\lambda_{c1} \end{bmatrix} \quad (32)$$

$$Q_c^{13} = \begin{bmatrix} -\lambda_{c3} \\ 0 \end{bmatrix}, Q_c^{23} = \begin{bmatrix} -\lambda_{c2} \\ 0 \end{bmatrix}, Q_c^{33} = [-1 + 2\lambda_{c1}] \quad (33)$$

For this system, $\text{Dim}(Z_i) = 2$ and $\text{Dim}(Z_{ik}) = 1$ and so before preprocessing

$$\text{Size of LMI } = 2N + L \quad (34)$$

$$\text{Number of Decision Variables } = 3 \quad (35)$$

We tested the technique on a globally coupled network using randomly chosen weighting of $1 \leq a_i \leq 2$ and $\frac{0.5}{N} \leq b_{ij} \leq \frac{1.5}{N}$. We found that for a globally coupled network we could determine SOS for $N = 50$.

Computational time for globally coupled network

N (nodes)	10	20	30	40	50
L (links)	45	190	435	780	1225
t_{cpu} (s)	0.3	2.5	19.8	129.3	320.6
t_{clock} (s)	0.2	1.4	10.7	73.5	211.2

For the globally coupled network the limit of scaling was from the effect of L on the size of the LMI.

We next verified SOS for a 1-D lattice with a randomly chosen weighting of $1.5 \leq a_i \leq 2.5$ and $0.5 \leq b_{i,i+1} \leq 1$. For a 1-D lattice network we could determine SOS for $N = 400$ due to the lower number of links.

Computational time for 1D lattice network

N (nodes)	50	100	200	300	400
L (links)	49	99	199	299	399
t_{cpu} (s)	1.0	5.7	43.8	130.8	308.4
t_{clock} (s)	0.6	3.0	23.8	76.9	196.2

For both of the above problems existing Sum of Squares toolboxes were able to verify SOS for $N = 15$, but were computationally impractical for $N = 20$. SOSTOOLS [20] scaled in a similar manner for both network types.

Computational time for 1-D lattice with SOSTOOLS

N (nodes)	11	13	15	17
L (links)	10	12	14	16
t_{cpu} (s)	5.2	21.8	80.0	532.5

For this example Structured Sum of Squares did not appear to be conservative e.g for the 1-D lattice with $N = 50$, the technique showed SOS for $a_i = 2$, $b_{i,i+1} = 1$ but not for $a_i = 1.99$, $b_{i,i+1} = 1$, which is expected by noting that $x_i^4 + x_{i+1}^4 - 2x_i^2 x_{i+1}^2 = (x_i^2 - x_{i+1}^2)^2$.

Example 3: We also analysed the domains on which Hamiltonian functions for a Duffing oscillator network were positive definite using structured SOS, in order to show stability of system (6). Recall from (7) that

$$V = \sum_{i=1}^{N} (\frac{1}{2} z_i^2 + a_i (\frac{1}{2} y_i^2 - \frac{1}{4} y_i^4)) + \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} b_{ik} \frac{1}{4} (y_k - y_i)^4 \quad (36)$$

This can be reduced to testing

$$V_1 = \sum_{i=1}^{N} a_i (\frac{1}{2} y_i^2 - \frac{1}{4} y_i^4) + \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} b_{ik} \frac{1}{4} (y_k - y_i)^4 \quad (37)$$

This is locally positive definite about $y_i = 0$ but we require techniques to find whether it is positive definite for a larger domain. This an important starting point for analysing non-linear behaviour such as finding the region of attraction estimate if the equilibrium point is asymptotically stable.

To find the positive definite domain, we tested whether

$$p(y_1, \ldots, y_N) = V_1 - \sum_{i=1}^{N} \phi_i(y_i) \text{ is SOS} \quad (38)$$

where

$$\phi_i(y_i) = \lambda_i y_i^2 (g - y_i^2) \quad (39)$$

and $\lambda_i > 0$, which, when true, implies that V_1 is positive definite for $y_i^2 < g$.

In this case

$$p_i = a_i (\frac{1}{2} y_i^2 - \frac{1}{4} y_i^4) - \lambda_i y_i^2 (g - y_i^2) \quad (40)$$

$$p_c = \frac{1}{2} (0.25 y_k^4 - y_k^3 y_i + 1.5 y_i^2 y_k^2 - y_k y_i^3 + 0.25 y_i^4) \quad (41)$$

$$P_i = a_i \begin{bmatrix} 0.5 & 0 \\ 0 & -0.25 \end{bmatrix} - \lambda_i \begin{bmatrix} g & 0 \\ 0 & -1 \end{bmatrix}$$

where $\lambda_i > 0$ are parameters which are not due to dependencies in the monomial vector and which are used as decision variables in the SDP.

$$Q_c = \frac{1}{2} \begin{bmatrix} 0 & 0 & 0 & \lambda_{c2} & -\lambda_{c3} \\ 0 & 0.25 & \lambda_{c3} & -\lambda_{c1} & -0.5 \\ 0 & \lambda_{c3} & 0 & 0 & -\lambda_{c2} \\ \lambda_{c2} & -\lambda_{c1} & 0 & 0.25 & -0.5 \\ -\lambda_{c3} & -0.5 & -\lambda_{c2} & -0.5 & 1.5 + 2\lambda_{c1} \end{bmatrix}$$

where $\lambda_c = (\lambda_{c1}, \lambda_{c2}, \lambda_{c3})$ are decision variables in the SDP. Thus

$$Q_c^{11} = Q_c^{22} = \frac{1}{2} \begin{bmatrix} 0 & 0 \\ 0 & 0.25 \end{bmatrix} \quad (42)$$

$$Q_c^{12} = \frac{1}{2} \begin{bmatrix} 0 & \lambda_{c2} \\ \lambda_{c3} & -\lambda_{c1} \end{bmatrix}, Q_c^{13} = \frac{1}{2} \begin{bmatrix} -\lambda_{c3} \\ -0.5 \end{bmatrix} \quad (43)$$

$$Q_c^{23} = \frac{1}{2} \begin{bmatrix} -\lambda_{c2} \\ -0.5 \end{bmatrix}, Q_c^{33} = \frac{1}{2} [1.5 + 2\lambda_{c1}] \quad (44)$$

For this system, $\text{Dim}(Z_i) = 2$ and $\text{Dim}(Z_{ik}) = 1$ and so before preprocessing

$$\text{Size of LMI } = 2N + L \quad (45)$$

$$\text{Number of Decision Variables } = N + 3 \quad (46)$$

where the N comes from the positive definite function parameters.

For the globally coupled network we set $g = 1.8$ and we randomly set $\frac{0.5}{N} \leq b_{ij} \leq \frac{1.5}{N}$ and $0.5 \leq a_i \leq 1.5$.

Computational time for globally coupled network

N (nodes)	10	20	30	40	50
L (links)	45	190	435	780	1225
t_{cpu} (s)	0.3	3.2	19.7	103.4	348.1
t_{clock} (s)	0.2	1.7	10.7	59.9	229.7

Once again, the limit on scaling for global coupling was the size of the LMI due to L.

For the 1-D lattice network we set $g = 1.8$ and we randomly set $0.5 \leq b_{ij} \leq 1$ and $0.5 \leq a_i \leq 1.5$.

Computational time for 1D lattice network

N (nodes)	50	100	200	300
L (links)	49	99	199	299
t_{cpu} (s)	1.2	5.6	41.3	137.6
t_{clock} (s)	0.7	3.0	22.9	91.9

For the same problem existing Sum of Squares toolboxes were able to verify SOS for $N = 15$, with $N = 20$ computationally impractical.

It would be expected that V_1 should be positive definite for at least $g = 2$, whereas structured SOS could only verify SOS for $g < 2$ and so gave a conservative result. But this occurred for $N = 2$, where Structured SOS is equivalent to standard SOS programming, and so the conservatism was unlikely to be due to the structured SOS technique. It may be possible to overcome this using multipliers, using a transformation of variables, or using some other form of preprocessing.

V. Conclusion

In this paper we have introduced the concept of structured Sum of Squares in order to allow Sum of Squares programming to scale for networked systems. We have also applied this technique to analyse two examples of large polynomial functions with a network structure. The technique was shown to be applicable for large problems where typical Sum of Squares programming techniques do not scale well.

One future direction is to extend the above techniques to find structured Lyapunov functions for networked systems, rather than testing known Lyapunov function candidates, as well as to determine regions of attraction for large scale systems. Another future direction of research is to apply the technique to more complicated examples and a variety of network types.

VI. Appendix A

$$\begin{aligned}\dot{V} &= \sum_{i=1}^{N}(z_i(-\gamma_i z_i - a_i(y_i - y_i^3)) + a_i(y_i - y_i^3)z_i) \\ &+ \sum_{i=1}^{N} z_i \sum_{j=1}^{N} b_{ij}(y_j - y_i)^3 - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} 2b_{ik}(y_k - y_i)^3 z_i \\ &= -\sum_{i=1}^{N} \gamma_i z_i^2 \leq 0\end{aligned} \tag{47}$$

using $b_{ij} = b_{ji}$.

References

[1] P. Apkarian and H. D. Tuan. Parameterized LMIs in control theory. *SIAM Journal on Control and Optimization*, 38:1241–1264, 2000.

[2] A. R. Bergen and D. J. Hill. A structure preserving model for power system analysis. *IEEE Transactions on Power Apparatus and Systems*, 100(1):25–35, 1981.

[3] G. Chesi. Domain of attraction: Estimates for non-polynomial systems via LMIs. *Proc. 16th IFAC World Congress on Automatic Control*, 2005.

[4] G. Chesi. LMI techniques for optimization over polynomials in control: a survey. *IEEE Transactions in Automatic Control*, 55(11):2500–2510, 2010.

[5] E. J. Hancock and A. Papachristodoulou. Generalised absolute stability and sum of squares. *Proceedings of the American Control Conference*, 2011.

[6] J. L. Hemmen and W. F. Wreszinski. Lyapunov functions for the Kuramoto model of nonlinearly coupled oscillators. *Journal of Statistical Physics*, 72(1/2):145–166, 1993.

[7] A. Jadbabaie, N. Motee, and M. Barahona. On the stability of the Kuramoto model of coupled nonlinear oscillators. *Proceedings of the 2004 American Control Conference*, 2004.

[8] H. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[9] M. Kojima, S. Kim, and H. Waki. Sparsity in sum of squares polynomials. *Mathematical Programming*, 103(1):45–62, 2005.

[10] E. Kosmatopoulos and M. Christodoulou. Structural properties of gradient recurrent high-order neural networks. *IEEE Transactions on Circuits and Systems- II*, 42(9), 1995.

[11] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

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

[13] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[14] J. Nie. Sum of squares method for sensor network localisation. *Computational Optimization and Applications*, 43(2):151–179, 2009.

[15] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using sum of squares decomposition. *Proceedings of the IEEE CDC*, 2002.

[16] A. Papachristodoulou and S. Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. *Positive Polynomials in Control*, 312:23–43, 2005.

[17] A. Papachristodoulou and S. Prajna. A tutorial on sum of squares techniques for systems analysis. *Proceedings of the American Control Conference*, 2005.

[18] P. A. Parrilo. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. *PhD Thesis*, 2000.

[19] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96:293–320, 2003.

[20] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares program solver. *Proceedings of the IEEE CDC*, 2002.

[21] Y. Pykh. Lyapunov functions for Lotka-Volterra systems: An overview and problems. *Proceedings of 5th IFAC Symposium "Nonlinear Control Systems"*, 2001.

[22] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, pages 625–653, 1999.

[23] H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimisation problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[24] X. F. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6 – 20, 2003.