Decentralized Computation for Robust Stability of Large-scale Systems with Parameters on the Hypercube

Reza Kamyar and Matthew M. Peet

***Abstract*— In this paper, we propose a parallel algorithm to solve the problem of robust stability of systems with large state-space and with large number of uncertain parameters. The dependence of the system on the parameters is polynomial and the parameters are assumed to lie in a hypercube. Although the parameters are assumed to be static, the method can also be applied to systems with time-varying parameters. The algorithm relies on a variant of Polya's theorem which is applicable to polynomials with variables inside a multi-simplex. The algorithm is divided into formulation and solution subroutines. In the formulation phase, we construct a large-scale semidefinite programming problem with structured elements. In the solution phase we use a structured primal-dual approach to solve the structured semidefinite programming problem. In both subroutines, computation, memory and communication is efficiently distributed over hundreds and potentially thousands of processors. Numerical tests confirm the accuracy and scalability of the proposed algorithm.**

I. INTRODUCTION

Models often contain uncertainty. Uncertainties come from many sources including linearization, model reduction or unknown operating conditions. In this paper, we address the robust stability problem of large-scale systems with polynomial dependence on several unknown parameters which lie on the hypercube. The problem of robust analysis and control of uncertain systems has thoroughly studied in references such as [1], [2], [3] in both the frequency domain [4] and the time domain [5], [6].

In this paper, we consider parameter-dependent systems of the form $\dot{x}(t) = A(\alpha)x(t) + B(\alpha)u(t)$. Neglecting the input, $A(\alpha)$ is stable for all $\alpha \in \Delta$ if and only if there exists a function $P(\alpha)$ such the Lyapunov inequality

$$A(\alpha)^T P(\alpha) + P(\alpha)A(\alpha) < 0$$

for all $\alpha \in \Delta$. Several recent results have considered ways of using Linear Matrix Inequalities (LMIs) to construct functions P which satisfy this Lyapunov inequality [7], [8], [9], [10]. For example, it was shown in [11] that if A is polynomial, then we can assume that P is also polynomial. Unfortunately, however, this only helps us slightly, as the problem of verifying that a polynomial is positive is known to be NP-hard [12].

To find polynomial functions P which satisfy the Lyapunov inequality, researchers [13] have recently turned to Sum-of-Squares methods [14], [15] and Positivstellensatz results [16] to construct increasingly accurate and increasingly complex LMI-based tests for stability. Unfortunately, due to the inherent intractability of the problem of polynomial optimization, Sum-of-Squares (SOS) based algorithms typically run out of memory for even relatively small-sized problems. This fact makes it difficult to solve SOS-based algorithms on current desktop machines.

In this paper, we consider the problem of finding a polynomial solution to the Lyapunov inequality using high-performance computing resources including cluster computers and supercomputers. Similar to desktop computers, supercomputers have relatively slow per-core clock speed. Unlike desktop computers, however, supercomputers have hundreds of thousands of processors. This means that if all processors can be used simultaneously, the speedup from desktop to supercomputer can be large, resulting in an improved ability to solve problems with high dimension or several uncertain variables. Unfortunately, in order to simultaneously utilize hundreds of thousands of processors, our algorithms must be very carefully structured.

Unfortunately, although a great deal of effort has gone into creating parallel SDP solvers [17], optimization of positive matrices is known to be an inherently sequential problem [18], which means that on a massively parallel machine, certain centralized computations will increasingly dominate the run-time, resulting in logarithmic speedup curves in a phenomenon known as Amdahl's law [19].

In this paper, we avoid Amdahl's law by creating algorithms with decentralized structure. Unfortunately, although SOS is a very powerful tool for optimization of polynomials, the LMI conditions associated with this methodology do not have a readily exploitable structure. For this reason, we explore an alternative approach to optimization of polynomials based on Polya's theorem [20], [21].

The use of Polya's theorem to create an SDP-based test for robust stability is not new, and has been explored in e.g. [22], [23] for both the simplex and the hypercube. In previous work it has been shown that the computational complexity of the robust stability test based on Polya's algorithm is roughly equivalent to that using the SOS framework.

In this paper, our approach is to show how to use parallel computing to set up and solve polynomial optimization problems based on Polya's lemma. This work is an extension of the work in [24], [25], wherein we considered the problem of robust stability on a simplex: $\{x : \sum x_i = 1, x_i \geq 0\}$. In this previous work, we were able to achieve scalability on hundreds of processors and solve robust analysis problems

Reza Kamyar is a Ph.D student with the Cybernetic Systems and Control Lab (CSCL), Department of Mechanical, Material and Aerospace Engineering, Illinois Institute of Technology, Chicago, IL, 60616 USA, rkamyar@hawk.iit.edu

Matthew M. Peet is an assistant professor with the department of Mechanical, Material and Aerospace Engineering, Illinois Institute of Technology, Chicago, IL, 60616 USA, mpeet@iit.edu

in systems with 100 states. Unfortunately, the simplex is a rather restrictive form of uncertainty set - it does not allow for parameters which take values on the interval or on a polytope. Additionally, we hope to eventually extend our algorithms to the problem of nonlinear stability, which required that the polynomial be defined over a set which include the origin - which the simplex does not.

Our approach is to use a recent extension of Polya's algorithm to the multi-simplex [23] to create a parallel algorithm for optimization of polynomials with variables defined on the multi-simplex. When we say variables lie on the multi-simplex, we mean that different subsets of the variables lie on different simplexes. We show how parallel machines can be used with this new version of Polya's algorithm to define a block-structure LMI problem of a form which is similar to the case of a single simplex - at the cost of an increased number of variables. We then apply the structured primal-dual approach of [25] to solve this structured LMI.

The result is an algorithm with almost no centralized computation, memory or communication - creating near-ideal speed-up. Specifically, we show that the communication operations per processor is proportional to $\frac{1}{N_c}$, where N_c is the number of processors used by the algorithm. This implies that by increasing the number of processors, we actually decrease the communication overhead per processor and improve the speed-up. Naturally, there exists an upper-bound for the number of processors which can be used by the algorithm, beyond which, no speed-up is gained. This upper-bound is proportional to the number of uncertain parameters in the system and for practical problems will be far larger than the number of available processers.

This paper is arranged as follows. In Section II, we describe the notation. In Section III we show how to represent the given uncertain system matrix in the form of a class of homogeneous polynomials with uncertain parameters inside the unit multi-simplex - the form required for input to the algorithm. In Section IV, we explain Polya's algorithm, the stability conditions and the SDP problem associated with this algorithm. The parallel setup algorithm and its complexity analysis are addressed in Section V. Finally, the speed-up and accuracy of the algorithm are demonstrated by two examples in Section VI.

II. Notation

We denote a monomial by $\alpha^\gamma = \prod_{i=1}^{l} \alpha_i^{\gamma_i}$, where $\alpha \in \mathbb{R}^l$ is the vector of variables and $\gamma \in \mathbb{N}^l$ is the vector of exponents. For a given number of variables, l, define the set $W_d := \left\{\gamma \in \mathbb{N}^l : \sum_{i=1}^{l} \gamma_i = d\right\}$ which represents the set of monomials of degree d in l variables. We represent the homogeneous matrix-valued polynomial $P(\alpha)$ of degree d_p as

$$P(\alpha) = \sum_{h \in W_{d_p}} P_h \alpha^h,$$

where $P_h \in \mathbb{R}^{n\times n}$ are the coefficients of the monomials and W_{d_p} is the set of the exponents of the monomials in $P(\alpha)$. Consider the case where $\alpha = (\alpha_1, \cdots, \alpha_N)$ with $\alpha_i \in \mathbb{R}^{l_i}$, and $h = (h_1, \cdots, h_N)$, where $h_i \in W_{d_{p_i}}$. Then we represent a class of homogeneous polynomials of degree $d_p = \sum_{i=1}^{N} d_{p_i}$ as

$$P(\alpha) = \sum_{h_1 \in W_{d_{p_1}}} \cdots \sum_{h_N \in W_{d_{p_N}}} P_{\{h_1,\cdots,h_N\}} \alpha_1^{h_1} \cdots \alpha_N^{h_N}.$$

We define $D_p = (d_{p_1}, \cdots, d_{p_N})$ as the vector of the degrees of the variables $\alpha_1, \cdots, \alpha_N$. We often use the lexicographical ordering of monomials, wherein $\alpha^{h_1} \cdots \alpha^{h_N}$ precedes $\alpha^{h'_1} \cdots \alpha^{h'_N}$, if the left-most non-zero entry of $h - h' = (h_1 - h'_1, \cdots, h_N - h'_N)$ is positive. We denote the l-dimensional unit simplex by Δ_l as

$$\Delta_l := \left\{ \alpha \in \mathbb{R}^l, \sum_{i=1}^{l} \alpha_i = 1, \alpha_i \geqslant 0 \right\}.$$

The unit multi-simplex $\tilde{\Delta}_{\{l_1,\cdots,l_N\}}$ is the Cartesian product of N unit simplexes; i.e., $\tilde{\Delta}_{\{l_1,\cdots,l_N\}} := \Delta_{l_1} \times \cdots \times \Delta_{l_N}$. The subspace of symmetric matrices and cone of positive definite symmetric matrices in $\mathbb{R}^{n\times n}$ are denoted by $\mathbb{S}_n$ and $\mathbb{S}_n^+$, respectively. $\text{diag}(X_1, \cdots, X_m)$ denotes a block-diagonal matrix in $\mathbb{R}^{mn\times mn}$ whose diagonal blocks are $X_1, \cdots, X_m \in \mathbb{R}^{n\times n}$. I_n and 0_n are the identity and zero matrices of dimension n. We will use a standard basis for $\mathbb{S}_n$, which is defined as

$$[E_k]_{ij} = \begin{cases} 1 & i = j = k \\ 0 & \text{otherwise} \end{cases}, \quad \text{for } k \leq n$$

and $[E_k]_{ij} = [A_k]_{ij} + [A_k]_{ij}^T$, for $k > n$, where

$$[A_k]_{ij} = \begin{cases} 1 & i = j-1 = k-n \\ 0 & \text{otherwise.} \end{cases}$$

The canonical basis for $\mathbb{R}^n$ is denoted by e_i for $i = 1, \cdots, n$, where

$$e_i = [0 \ldots 0 \underbrace{1}_{i^{th}} 0 \ldots 0].$$

$\mathbf{1}_n \in \mathbb{N}^n$ is the vector with all elements equal to 1.

III. Preliminaries

In this paper, we address the stability of the uncertain linear systems of the form

$$\dot{x}(t) = A(\alpha) x(t), \tag{1}$$

where $A(\alpha) \in \mathbb{R}^{n\times n}$ is a homogeneous matrix-valued polynomial of degree d_a and can be represented as

$$A(\alpha) = \sum_{h_1 \in W_{d_{a_1}}} \cdots \sum_{h_N \in W_{d_{a_N}}} A_{\{h_1,\cdots,h_N\}} \alpha_1^{h_1} \cdots \alpha_N^{h_N}, \tag{2}$$

with variables in the multi-simplex $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$.

A. Representing $A(\alpha)$ in homogeneous form with $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$

In this section, we show how to transform non-homogeneous polynomials with parameters on the hypercube into homogeneous polynomials on the multi-simplex, as introduced in II.

Case 1: *Non-homogeneous $A(\alpha)$, $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$*
Suppose the polynomial $A(\alpha)$ with N_a monomials and

$\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$ is not homogeneous. Define $D_{a,k} = (d_{a_{1,k}},\cdots,d_{a_{N,k}}) \in \mathbb{N}^N$ for $k = 1,\cdots,N_a$, where $d_{a_{i,k}}$ is the total degree of the variables inside Δ_{l_i}, in the k^{th} monomial of $A(\alpha)$ according to lexicographical ordering. To evaluate the stability of the system $\dot{x}(t) = A(\alpha)x(t)$ for all $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$, we evaluate the stability of $\dot{x}(t) = B(\alpha)x(t)$, where $B(\alpha) = A(\alpha)$ for all $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$ and where $B(\alpha)$ is obtained as follows.

1) Initialize $B = A$.
2) For $k = 1,\cdots,N_a$, multiply the k^{th} monomial of $B(\alpha)$, according to lexicographical ordering, by $\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i,j}^{(d_{a_i}-d_{a_{i,k}})}\right)$, where

$$d_{a_i} = \max_{k=1,\cdots,N_a} d_{a_{i,k}} \quad \text{for} \quad i = 1,\cdots,N. \tag{3}$$

As an example, consider the non-homogeneous polynomial

$$A(\alpha) = A_1(\alpha_{1,1} + \alpha_{12})\alpha_{2,1} + A_2\alpha_{1,2}^2 + A_3\alpha_{2,2},$$

where $N = 2, N_a = 4, (\alpha_{1,1},\alpha_{1,2}),(\alpha_{2,1},\alpha_{2,2}) \in \Delta_2$ and $D_{a,1} = (1,1), D_{a,2} = (1,1)$, $D_{a,3} = (2,0), D_{a,4} = (0,1)$. The homogeneous form of $A(\alpha)$ is

$$\begin{aligned}
A(\alpha) &= A_1(\alpha_{1,1} + \alpha_{12})^2\alpha_{2,1} + A_2\alpha_{1,2}^2(\alpha_{2,1} + \alpha_{2,2})\\
&\quad + A_3(\alpha_{1,1} + \alpha_{1,2})^2\alpha_{2,2}\\
&= A_{\{(2,0),(1,0)\}}\alpha_{1,1}^2\alpha_{2,1} + A_{\{(2,0),(0,1)\}}\alpha_{1,1}^2\alpha_{2,2}\\
&\quad + A_{\{(1,1),(1,0)\}}\alpha_{1,1}\alpha_{1,2}\alpha_{2,1} + A_{\{(1,1),(0,1)\}}\alpha_{1,1}\alpha_{1,2}\alpha_{2,2}\\
&\quad + A_{\{(0,2),(1,0)\}}\alpha_{1,2}^2\alpha_{2,1} + A_{\{(0,2),(0,1)\}}\alpha_{1,2}^2\alpha_{2,2},
\end{aligned}$$

where

$$A_{\{(2,0),(1,0)\}} = A_1, \quad A_{\{(2,0),(0,1)\}} = A_3, \quad A_{\{(1,1),(1,0)\}} = 2A_1$$

$$A_{\{(1,1),(0,1)\}} = 2A_3, \quad A_{\{(0,2),(1,0)\}} = A_1 + A_2,$$

$$A_{\{(0,2),(0,1)\}} = A_2 + A_3.$$

Case 2: *$A(\alpha)$ with α inside the hypercube Φ_N*
Define hypercube $\Phi_N \subset \mathbb{R}^N$ as follows.

$$\Phi_N := \{\alpha \in \mathbb{R}^N : |\alpha_i| \le r_i, \text{ for } i = 1,\cdots,N\}.$$

Suppose $\alpha = (\alpha_{1,1},\cdots,\alpha_{N,1}) \in \Phi_N$. To evaluate the stability of the system $\dot{x}(t) = A(\alpha)x(t)$ for all $\alpha \in \Phi_N$, we evaluate the stability of $\dot{x}(t) = B(\alpha)x(t)$ for all $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$, where $\{A(\alpha) : \alpha \in \Phi_N\} = \{B(\alpha) : \alpha \in \Delta_{\{l_1,\cdots,l_N\}}\}$ and where $B(\alpha)$ is obtained as follows.

1) Initialize $B = A$.
2) Define new variables $\alpha'_{i,1} = \frac{\alpha_i + r_i}{2r_i} \in [0,1]$, for $i = 1,\cdots,N$ and substitute α' for α in $B(\alpha)$.
3) Define a new set of variables $\beta_1,\cdots,\beta_N \in [0,1]$ such that $(\alpha'_{i,1},\beta_i) \in \Delta_2$ for $i = 1,\cdots,N$.
4) Recalling that $D_{a,k} = (d_{a_{1,k}},\cdots,d_{a_{N,k}})$ is the vector of degrees of k^{th} monomial in $A(\alpha')$, for $k = 1,\cdots,N_a$, multiply the k^{th} monomial in $B(\alpha')$, according to lexicographical ordering, by $\prod_{i=1}^N (\alpha'_{i,1} + \beta_i)^{d_{a_i} - d_{a_{i,k}}}$, where d_{a_i} is the same as in (3).

IV. Problem setup

In this section, we derive a set of stability conditions for systems with uncertainties on the multi-simplex. We apply Polya's theorem [20] to a Lyapunov inequality to construct a set of stability conditions that are independent of the uncertainty parameters. The following is a Lyapunov stability condition [22].

Theorem 1: The linear system (1) is stable if and only if there exists a polynomial matrix $P(\alpha)$ such that $P(\alpha) > 0$ and

$$\mathrm{A}^T(\alpha)P(\alpha) + P(\alpha)\mathrm{A}(\alpha) < 0 \tag{4}$$

for all $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$.

The stability condition in Theorem 1 is intractable [12]. To construct a sequence of tractable conditions we use a variant of Polya's theorem [20] for polynomials with variables inside the unit multi-simplex. This result can be implied from the work in [23].

Theorem 2: (Polya's Theorem) The homogeneous polynomial $F(\alpha) > 0$ for all $\alpha \in \tilde{\Delta}_{\{l_1,\cdots,l_N\}}$ if and only if for all sufficiently large d,

$$\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i,j}\right)^d F(\alpha) \tag{5}$$

has all positive definite coefficients.
In the following subsection we derive the parameter-independent stability conditions associated with Polya's theorem.

A. Polya's algorithm and its associated stability conditions

In order to verify the robust stability of system (1) we look for a $P(\alpha) \in \mathbb{S}_n^+$ that satisfies (4). According to [26], we can assume that P is homogeneous. If we let P be homogeneous and of degree $d_p = \sum_{i=1}^N d_{p_i}$ then we can represent P as

$$P(\alpha) = \sum_{h_N \in W_{d_{p_N}}} \cdots \sum_{h_1 \in W_{d_{p_1}}} P_{\{h_1,\cdots,h_N\}} \alpha_1^{h_1} \cdots \alpha_N^{h_N} \tag{6}$$

where we are trying to find the matrices $P_{\{h_1,\cdots,h_N\}} \in \mathbb{R}^{n\times n}$. Using Polya's theorem, the constraints of Theorem 1 can be written as follows:

$$\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i,j}\right)^d P(\alpha) \quad \text{and}$$

$$-\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i,j}\right)^d (\mathrm{A}^T(\alpha)P(\alpha) + P(\alpha)\mathrm{A}(\alpha))$$

have all positive definite coefficients.
By substituting for $A(\alpha)$ and $P(\alpha)$ from (2) and (6) in the above conditions and calculating the coefficients of monomials, there exist $\beta_{h,\gamma}$ and $H_{h,\gamma}$ such that the conditions can be expressed as

$$\sum_{h_1 \in W_{d_1}} \cdots \sum_{h_N \in W_{d_N}} \beta_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} P_{\{h_1,\cdots,h_N\}} > 0 \tag{7}$$

for $\gamma_1 \in W_{d_{p_1}+d}, \cdots, \gamma_N \in W_{d_{p_N}+d}$, and

$$\sum_{h_1 \in W_{d_1}} \cdots \sum_{h_N \in W_{d_N}} \Big(H^T_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} P_{\{h_1,\cdots,h_N\}} + P_{\{h_1,\cdots,h_N\}} H_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} \Big) < 0 \quad (8)$$

for $\gamma_1 \in W_{d_{p_1}+d_{a_1}+d}, \cdots, \gamma_N \in W_{d_{p_N}+d_{a_N}+d}$.

Calculating β:
To calculate the $\{\beta_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}}\}$ coefficients and $\{H_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}}\}$ we provide the following recursive formulas. These formulas are the generalized form of the recursive formulas which were provided in [27] for the case of a single simplex. First for $\gamma_1 \in W_{d_{p_1}}, \cdots, \gamma_N \in W_{d_{p_N}}$, and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ set

$$\beta^{(0)}_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = \begin{cases} 1 & h_1 = \gamma_1, \cdots, h_N = \gamma_N \\ 0 & \text{otherwise.} \end{cases}$$

Then for $i = 1, \cdots, d$, $\gamma_1 \in W_{d_{p_1}} + i, \cdots, \gamma_N \in W_{d_{p_N}} + i$ and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ we have

$$\beta^{(i)}_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = \sum_{\substack{\lambda_N \in W_{d_{p_N}+i-1} \\ \lambda_N = \gamma_N - e_j \\ j=1\ldots l_N}} \cdots \sum_{\substack{\lambda_1 \in W_{d_{p_1}+i-1} \\ \lambda_1 = \gamma_1 - e_j \\ j=1\ldots l_1}} \beta^{(i-1)}_{\{h_1,\cdots,h_N\},\{\lambda_1,\cdots,\lambda_N\}}.$$

Finally, set $\beta_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = \beta^{(d)}_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}}$, where $\gamma \in W_{d_p+d}$.

Calculating H:
To calculate $\{H_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}}\}$ coefficients, first for $\gamma_1 \in W_{d_{p_1}+d_{a_1}}, \cdots, \gamma_N \in W_{d_{p_N}+d_{a_N}}$ and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ let

$$H^{(0)}_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = \sum_{\substack{\delta_N \in W_{d_{a_N}} \\ \delta_N + h_N = \gamma_N}} \cdots \sum_{\substack{\delta_1 \in W_{d_{a_1}} \\ \delta_1 + h_1 = \gamma_1}} \mathrm{A}_{\{\delta_1,\cdots,\delta_N\}}.$$

Then, for $i = 1, \ldots, d$, $\gamma_1 \in W_{d_{p_1}+d_{a_1}+i}, \cdots, \gamma_1 \in W_{d_{p_N}+d_{a_N}+i}$ and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ we have

$$H^{(i)}_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = \sum_{\substack{\lambda_N \in W_{d_{p_N}+d_{a_N}+i-1} \\ \lambda_N = \gamma_N - e_j \\ j=1\ldots l_N}} \cdots \sum_{\substack{\lambda_1 \in W_{d_{p_1}+d_{a_1}+i-1} \\ \lambda_1 = \gamma_1 - e_j \\ j=1\ldots l_1}} H^{i-1}_{\{h_1,\cdots,h_N\},\{\lambda_1,\cdots,\lambda_N\}}.$$

Finally, set $H_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}} = H^d_{\{h_1,\cdots,h_N\},\{\gamma_1,\cdots,\gamma_N\}}$, where $\gamma_1 \in W_{d_{p_1}+d_{a_1}+d}, \cdots, \gamma_N \in W_{d_{p_N}+d_{a_N}+d}$.

To solve the LMI conditions in (7) and (8), we express them in the form of a dual Semi-Definite Programming (SDP) problem with a block-diagonal structure that is suitable for decentralized computation. The following section defines the dual SDP problem.

B. The SDP problem elements

Semidefinite programming is the optimization of a linear objective function over the cone of positive definite matrices and subject to linear matrix equality and/or inequality constraints. Given the SDP elements $C \in \mathbb{S}_m$, $a \in \mathbb{R}^k$ and $B_i \in \mathbb{S}_m$, we define the dual formulation of SDP as

$$\begin{aligned} &\min_{y,Z} \quad a^T y \\ &\text{subject to} \quad \sum_{i=1}^{K} y_i B_i - C = Z \\ &\qquad\qquad Z \succeq 0 \,,\; y \in \mathbb{R}^K \end{aligned}$$

where $Z \in \mathbb{S}_m$ and $y \in \mathbb{R}^K$ are the dual variables. We define the SDP elements associated with the conditions (7) and (8) as follows. The element C is

$$C = \mathrm{diag}(C_1, \cdots C_L, C_{L+1}, \cdots C_{L+M}), \quad (9)$$

where for a given d and $N-$dimensional multi-simplex,

$$L = \prod_{i=1}^{N} \frac{(d_{p_i} + d + l_i - 1)!}{(d_{p_i} + d)!(l_i - 1)!} \quad (10)$$

is the number of monomials in $\prod_{i=1}^{N} \left(\sum_{j=1}^{l_i} \alpha_{i,j} \right)^d P(\alpha)$ and

$$M = \prod_{i=1}^{N} \frac{(d_{p_i} + d_{a_i} + d + l_i - 1)!}{(d_{p_i} + d_{a_i} + d)!(l_i - 1)!} \quad (11)$$

is the number of monomials in

$$\prod_{i=1}^{N} \left(\sum_{j=1}^{l_i} \alpha_{i,j} \right)^d (A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha)),$$

and for $j = 1, \cdots, L+M$,

$$C_j = \begin{cases} \varepsilon I_n \zeta_j, & 1 \le j \le L \\ 0_n, & L+1 \le j \le L+M, \end{cases} \quad (12)$$

where $\zeta \in \mathbb{N}^L$ is calculated as follows. First let

$$\zeta^{(0)} = \left[\frac{(d_{p_N} + d)!}{\prod_{i=1}^{l_N} h_{N_{(i,1)}}!}, \quad \cdots, \quad \frac{(d_{p_N} + d)!}{\prod_{i=1}^{l_N} h_{N_{(i,f(l_N, d_{p_N}+d))}}!} \right],$$

where $h_N \in \Delta_{l_N}$, and the scalar $h_{N(i,j)}$ denotes the exponent of the i^{th} variable in the j^{th} element of $W_{d_{P_N}}$ according to lexicographical ordering and where

$$f(l,g) = \frac{(l+g-1)!}{g!(l-1)!}$$

is the number of monomials in a polynomial of degree g with l variables. Then for $k = 1, \cdots, N$, set

$$\zeta^{(k)} = \zeta^{(k-1)} \otimes \left[\frac{(d_{p_{r(k)}} + d)!}{\prod_{i=1}^{l_{r(k)}} h_{r(k)_{(i,1)}}!}, \quad \cdots, \quad \frac{(d_{p_{r(k)}} + d)!}{\prod_{i=1}^{l_{r(k)}} h_{r(k)_{(i,f(l_{r(k)}, d_{p_{r(k)}}+d))}}!} \right],$$

where $r(k) = N - k + 1$ and $h_{N-k+1} \in \Delta_{N-k+1}$. Finally, set $\zeta = \zeta^{(N)}$.
For $i = 1, \cdots, K$, the elements B_i are defined as

$$B_i = \mathrm{diag}(B_{i,1}, \cdots, B_{i,L}, B_{i,L+1}, \cdots, B_{i,L+M}), \quad (13)$$

where
$$K = \frac{n(n+1)}{2} \prod_{i=1}^{N} \frac{(d_{p_i}+l_i-1)!}{d_{p_i}!(l_i-1)!}, \tag{14}$$
is the number of dual variables of SDP problem that is equal to the total number of upper-triangular elements in the coefficients of $P(\alpha)$ and where

$$B_{i,j} = \begin{cases} \sum_{h_N \in W_{d_{p_N}}} \cdots \sum_{h_1 \in W_{d_{p_1}}} \beta_{\{h_1,\cdots,h_N\},\{\lambda_{1_j},\cdots,\lambda_{N_j}\}} P_{\{h_1,\cdots,h_N\}}(e_i), \\ \qquad\qquad \text{for} \quad 1 \le j \le L \\ -\sum_{h_N \in W_{d_{p_N}}} \cdots \sum_{h_1 \in W_{d_{p_1}}} H^T_{\{h_1,\cdots,h_N\},\{\gamma_{1_{j-L}},\cdots,\gamma_{N_{j-L}}\}} P_{\{h_1,\cdots,h_N\}}(e_i) \\ \qquad +P_{\{h_1,\cdots,h_N\}}(e_i) H_{\{h_1,\cdots,h_N\},\{\gamma_{1_{j-L}},\cdots,\gamma_{N_{j-L}}\}}, \\ \qquad\qquad \text{for} \quad L+1 \le j \le L+M, \end{cases} \tag{15}$$

where λ_{i_j} is the j^{th} element of $W_{d_{p_i}+d_{a_i}}$ using lexicographical ordering and $\gamma_{i_{j-L}}$ is the $(j-L)^{th}$ element of $W_{d_{p_i}+d_{a_i}+d}$ using lexicographical ordering and
$$P_{\{h_1,\cdots,h_N\}}(x) = \sum_{k=1}^{\tilde{N}} E_k \, x_{k+N(I_{\{h_1,\cdots,h_N\}}-1)},$$
where E_k is the basis of $\mathbb{S}_n$ from Section II, I_h is the lexicographical index of h, $\tilde{N} = \frac{n(n+1)}{2}$ and n is the dimension of system (1). Finally, by setting
$$a = \vec{1} \in \mathbb{R}^K,$$
the SDP problem associated with Polya's algorithm is defined. In the following section, we propose a parallel setup algorithm to calculate the SDP elements defined in this section.

V. Parallel setup algorithm

In this section we propose a parallel setup algorithm that represents the stability conditions in (7) and (8) in the form of an SDP problem with elements in (9) and (13). A brief description of the algorithm is presented below, wherein we suppose the algorithm is run on N_c processors.

A. Algorithm description

Inputs:

The dimension of the hypercube (or multi-simplex) N, the dimensions of the simplexes $l_1,\cdots,l_N$, the vector of degrees of variables in $A(\alpha)$ and $P(\alpha)$ that are $D_a = (d_{a_1},\cdots,d_{a_N})$ and $D_p = (d_{p_1},\cdots,d_{p_N})$, The dimension of the state-space n, the coefficients of $A(\alpha)$ and the Polya's exponent d_{max}.

Step 1: *Initialization*

For $i = 1,\cdots,N_c$, processor i

1) Receives the input variables from the user and sets $d = 0$.
2) Calculates the number of monomials in $P(\alpha)$ and $A(\alpha)$ denoted by N_{p_0} and N_a using (10).
3) Sets $b = \sum_{k=1}^{N} l_k$.
4) Creates the set of exponents of the monomials in $P(\alpha)$ denoted by $Z^i_{p_0} = \{p^i_{0_1},\cdots,p^i_{0_{N'_p}}\}$, where $p^i_{0_j} \in \mathbb{N}^b$.

Step2: *Constructing the sets of exponents for $A(\alpha)$ and $P(\alpha)$*

For $i = 1,\cdots,N_c$, processor i

1) Sets $N_p = N_{p0}$.
2) Creates the set of exponents of the monomials in $P(\alpha)$ with lexicographical indices $(i-1)N'_p$ to iN'_p denoted by $Z^i_p = \{p^i_1,\cdots,p^i_{N'_p}\}$, where $p^i_j \in \mathbb{N}^b$ and
$$N'_p = \mathtt{floor}\left(\frac{N_p}{N_c}\right). \tag{16}$$
3) Receives the coefficients of the monomials in $A(\alpha)$ with lexicographical indices $(i-1)N'_a$ to iN'_a, from the user, where
$$N'_a = \mathtt{floor}\left(\frac{N_a}{N_c}\right).$$
4) Creates the set of exponents of the monomials in $A(\alpha)$ with lexicographical indices $(i-1)N'_a$ to iN'_a denoted by $Z^i_a = \{a^i_1,\cdots,a^i_{N'_a}\}$, where $a^i_j \in \mathbb{N}^b$.

Step 3: *Constructing the set of exponents for $P(\alpha)A(\alpha)$*

For $i = 1,\cdots,N_c$, processor i

1) Calculates the number of monomials in $P(\alpha)A(\alpha)$ denoted by N_{pa} and using (11).
2) Creates the set of exponents of the monomials in $P(\alpha)A(\alpha)$ with lexicographical indices $(i-1)N'_{pa}$ to iN'_{pa} denoted by $Z^i_{pa} = \{pa^i_1,\cdots,pa^i_{N'_{pa}}\}$, where $pa^i_j \in \mathbb{N}^{\sum_{k=1}^{N}(l_k)}$, where
$$N'_{pa} = \mathtt{floor}\left(\frac{N_{pa}}{N_c}\right). \tag{17}$$
3) For $j = 1,\cdots,N_{p_0}$ and $k = 1,\cdots,N'_a$, if $a^i_k + p^i_{0_j} \notin Z^i_{pa}$, then sends the coefficient of the monomial in $A(\alpha)$ with the exponent a^i_k to the processors with index r, such that $a^i_k + p^i_{0_j} \in Z^r_{pa}$.
4) For $j = 1,\cdots,N'_{pa}$ and $k = 1,\cdots,N'_p$, if $pa^i_j - p^i_k \notin Z^i_a$, then receives the coefficient of the monomial in $A(\alpha)$ with the exponent $pa^i_j - p^i_k$, from the processors with index r, such that $pa^i_j - p^i_k \in Z^r_a$.

Step 4: *Initializing β and H coefficients*

For $i = 1,\cdots,N_c$, processor i

1) For $j = 1,\cdots,N_{p_0}$ and $k = 1,\cdots,N'_{p_0}$ sets
$$\beta^i_{j,k} = \begin{cases} 1 & j = (i-1)N'_{p_0} + k \\ 0 & \text{otherwise.} \end{cases}$$
2) For $j = 1,\cdots,N_{p_0}$ and $k = 1,\cdots,N'_{pa}$, sets $H^i_{j,k} = \mathbf{0}_n$.
3) For $j = 1,\cdots,N_{p_0}$ and $k = 1,\cdots,N'_a$, sets $H^i_{j,I} = H^i_{j,I} + A_{(i-1)N'_a+k}$, where I is the lexicographical index of the monomial with exponent $a^i_k + p^i_{0_j}$ in Z^i_{pa} and where $A_{(i-1)N'_a+k}$ is the coefficient of

the monomial in $A(\alpha)$ with lexicographical index $(i-1)N'_a+k$.

Step 5: *Updating the set of exponents of $P(\alpha)$ and $P(\alpha)A(\alpha)$*

For $i=1,\cdots,N_c$, processor i

1) Sets $d=d+1$.
2) For $j=1,\cdots,N$, sets $d_{p_j}=d_{p_j}+1$ and $d_{a_j}=d_{a_j}+1$.
3) Calculates N_{p_2} using (10) and N_{pa_2} using (11), N'_{p_2} using (16) and N'_{pa_2} using (17).
4) Calculates the set of exponents of $\prod_{i=1}^N(\sum_{j=1}^{l_i}\alpha_{i,j})^d P(\alpha)$ denoted by $Z^i_{p_2}$ and the set of exponents of $\prod_{i=1}^N(\sum_{j=1}^{l_i}\alpha_{i,j})^d P(\alpha)A(\alpha)$ denoted by $Z^i_{pa_2}$.

Step 6: *Updating β and H coefficients*

1) For $i=1,\cdots,N_c$ and for $j=1,\cdots,N'_p$, if the monomial with exponent $p^i_j+\mathbf{1}_b\in Z^i_{p_2}$, then processor i sets $\beta^i_{k,j}=\beta^i_{k,j}+\beta^i_{k,I}$ for $k=1,\cdots,N'_{p0}$, where I is the lexicographical index of $p^i_j+\mathbf{1}_b$. If $p^i_j+\mathbf{1}_b\in Z^r_{p_2}$ where $r\neq i$, then processor r sends $\beta^i_{k,I}$ to processor i, and processor i sets $\beta^i_{k,j}=\beta^i_{k,j}+\beta^i_{k,I}$ for $k=1,\cdots,N'_{p0}$.
2) For $i=1,\cdots,N_c$ and for $j=1,\cdots,N'_{pa}$, if the monomial with exponent $pa^i_j+\mathbf{1}_b\in Z^i_{pa_2}$, then processor i sets $H^i_{k,j}=H^i_{k,j}+H^i_{k,I}$ for $k=1,\cdots,N'_{p0}$, where I is the lexicographical index of $pa^i_j+\mathbf{1}_b$. If $pa^i_j+\mathbf{1}_b\in Z^r_{pa_2}$ where $r\neq i$, then processor r sends $H^r_{k,I}$ to processor i, and processor i sets $H^i_{k,j}=H^i_{k,j}+H^r_{k,I}$ for $k=1,\cdots,N'_{p0}$.

Step 7: *Updating parameters*

For $i=1,\cdots,N_c$, processor i

1) Sets $N_p=N_{p_2}$, $N_{pa}=N_{pa_2}$, $N'_p=N'_{p_2}$ and $N'_{pa}=N'_{pa_2}$.
2) Sets $Z^i_p=Z^i_{p_2}$ and $Z^i_{pa}=Z^i_{pa_2}$.
3) If $d<d_{max}$, go to Step 5.

Step 8: *Calculating the SDP elements*

For $i=1,\cdots,N_c$, processor i

1) Calculates C_j for $j=(i-1)N'_{pa}+1,\cdots,iN'_{pa}$ using (12).
2) Calculates $B_{j,k}$ for $j=1,\cdots,K$ and $k=(i-1)N'_{pa}+1,\cdots,iN'_{pa}$ using (15), where K is defined in (14).

Outputs:

The SDP elements C and B_i for $i=1,\cdots,K$.

B. Complexity analysis

In this section, the algorithm performance in terms of speed-up, computation cost, communication cost and memory requirement is discussed.

Computation cost:

The most computationally expensive part of the algorithm is the calculation of $B_{i,j}$ elements for $i=1,\cdots,K$ and $j=1,\cdots,L+M$. It can be shown that, if the number of processors is equal to L, then the number of operations per processor at each iteration is

$$\sim K\cdot L_0\cdot\texttt{floor}(\frac{L+M}{N_c})\cdot n^5\sim n^5\prod_{i=1}^N l_i^{2d_{p_i}+d_{a_i}+d},$$

where $L_0=\prod_{i=1}^N\dfrac{(d_{p_i}+l_i-1)!}{(d_{p_i})!(l_i-1)!}$. Recall that N_c is the number of processors, d_{p_i} is the degree of the variable $\alpha_i\in\Delta_{l_i}$ in the polynomial $P(\alpha)$, d_{a_i} is the degree of the variable $\alpha_i\in\Delta_{l_i}$ in the polynomial $A(\alpha)$ and d is the Polya's exponent. In case of systems with uncertain parameters inside a single-simplex, this confirms that the number of operations scales as $l^{2d_p+d_a+d}n^5$ as reported in [25]. The number of operations versus the dimension of hypercube N is plotted in Fig. 1 for different Polya's exponents d. The figure shows that for the case of hypercube, the number of operations grows exponentially with the dimension of the hypercube; whereas for the case of simplex the number of operations grows polynomially. This means that the algorithms developed in this paper can handle fewer uncertain variables than was possible for a single simplex [27].

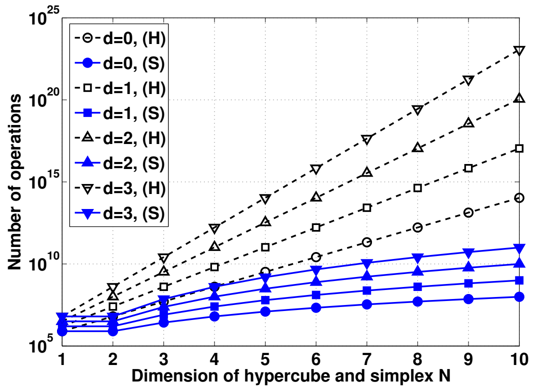

Fig. 1. The number of operations versus the dimension of the hypercube N, for different Polya's exponents d. (H) stands for hypercube and (S) stands for simplex.

Communication cost:

In the worst scenario, where every processor sends all its $\{H\}$ coefficients to all the other processors, it can be shown that the communication cost per processor at each iteration is

$$\sim\left(\texttt{floor}(\frac{L}{N_c})+\texttt{floor}(\frac{M}{N_c})\right)n^2\cdot\prod_{i=1}^N l_i\sim\frac{n^2}{N_c}\prod_{i=1}^N l_i^{d_{p_i}+d_{a_i}+d+1}$$

where $N_c\leq L$. Therefore, by increasing the number of processors, the communication cost per processor decreases and the scalability of the algorithm is improved. In case of the systems with uncertain parameters inside a single-simplex, the number of communication operations

$$\sim\frac{n^2}{N_c}l^{d_p+d_a+d+1}$$

which is the same result as reported in [25]. Again, it can be shown that the communication cost increases exponentially with the dimension of the hypercube; whereas in the case of simplex the communication cost increases polynomially.

Speed-up:
In the proposed setup algorithm, the calculation of $\{\beta\}$ and $\{H\}$ coefficients is distributed among the processors. The calculation is performed with no centralized (sequential) operation. As a result the algorithm can achieve near-ideal speed-up, i.e.

$$SP_N = \frac{N}{D+NS} = \frac{N}{1+0} = N,$$

Where $D=1$ is the ratio of the operations performed by all processors simultaneously to total operations performed simultaneously and sequentially. $S=0$ is the ratio of the operations performed sequentially to total operations performed simultaneously and sequentially.

Memory requirement:
The amount of memory for storing the SDP elements versus the number of uncertain parameters in the unit hypercube and unit simplex, for different dimensions of the state-space n and Polya's exponents d is shown in Fig. 2. In all cases of the hypercube we use $d_{p_i} = d_{a_i} = 1$ for $i = 1, \cdots, N$. The figure shows that for the case of hypercubes, the required memory increases exponentially with the number of uncertain parameters, whereas for the case of the simplex the required memory increases polynomially.

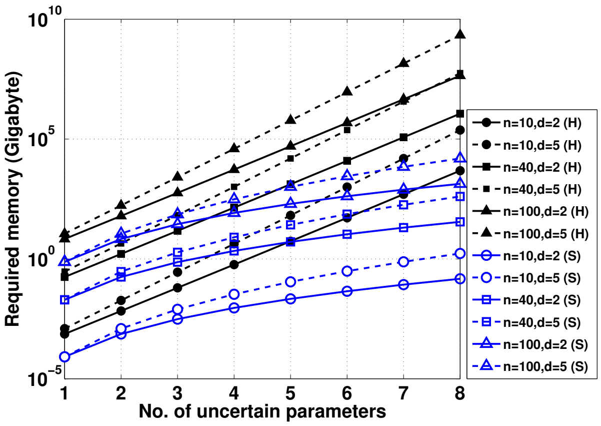

Fig. 2. Required memory for the calculation of SDP elements versus the number of uncertain parameters in the unit hypercube and unit simplex, for different state-space dimensions n and Polya's exponents d. (H) stands for hypercube and (S) stands for simplex.

VI. Examples

In this section, we evaluate the accuracy and the scalability of the proposed algorithm in the following examples.

Example 1: *Accuracy*
To illustrate accuracy, we first consider a simple system where the system matrix is a linear function of the uncertain parameters [13].

$$\dot{x}(t) = \left(A_0 + \sum_{i=1}^{4} \alpha_i A_i \right) x(t),$$

where $\alpha_i \in [-b, b]$ for $i = 1, \cdots, 4$ and

$$A_0 = \begin{bmatrix} -3 & 0 & -1.7 & 3 \\ -0.2 & -2.9 & -1.7 & -2.6 \\ 0.6 & 2.6 & -5.8 & -2.6 \\ -0.6 & 2.9 & -3.3 & -2.4 \end{bmatrix} \quad A_1 = \begin{bmatrix} 1.1 & -2.7 & -0.4 & -1.1 \\ 2.2 & 0.7 & -1.5 & 0.4 \\ -1.2 & -1.1 & 0.7 & 3 \\ -1.2 & -2.2 & -3.2 & -1.4 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1.6 & 2.7 & 0.1 & 0.6 \\ -3.6 & 0.4 & -0.1 & 2.3 \\ -1.1 & 2 & -0.7 & -1.8 \\ -2.6 & -1.5 & -1 & 0.8 \end{bmatrix} \quad A_3 = \begin{bmatrix} -0.6 & 1.5 & 0.5 & -1.6 \\ 0.2 & -0.1 & 0.2 & 0.3 \\ -0.1 & -0.2 & -0.2 & 1.2 \\ -0.5 & -1.2 & 1.7 & -0.1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -0.4 & -0.1 & -0.3 & 0.1 \\ 0.1 & 0.3 & 0.2 & 0 \\ 0 & 0.2 & -0.3 & 0.1 \\ 0.1 & -0.2 & -0.2 & 0 \end{bmatrix}$$

The goal is to find the maximum b such that the system is stable for all $\alpha_i \in [-b, b]$. Using the procedure in section III-A, we first represent the system in the homogeneous form with uncertain parameters inside the unit hypercube; i.e,

$$\dot{x}(t) = \left(A'_{0_b} + \sum_{i=1}^{4} \alpha'_i A'_{i_b} \right) x(t), \quad \alpha' \in \overline{\Phi}_4, \tag{18}$$

where $\overline{\Phi}$ is the unit hypercube. Then we solve the problem

$$\begin{aligned} \max \quad & b \\ \text{s.t.} \quad & \text{system (18) is stable for all } \alpha \in \overline{\Phi}_4 \end{aligned}$$

using a bisection search, where at each iteration of the bisection search we use the proposed setup algorithm to construct the SDP problem and use the parallel SDP solver in [27] to solve the SDP. The resulting optimal values for b are shown in Fig. 3 for different degrees of $P(\alpha)$ and Polya's exponents d. The maximum value for b is found to be 0.8739 which matches the value obtained in [13].

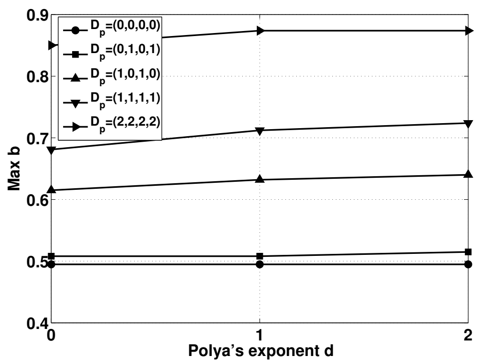

Fig. 3. Maximum bound on uncertain parameters for stability versus Polya's exponent for different degrees D_p of $P(\alpha)$

Example 2: *Scalability*
A parallel algorithm is scalable, if by using N_c processors it can solve a problem N_c times faster than solving the same problem using one processor. Thus, the speed-up of the ideal scalable algorithm is linear. To test the scalability of our algorithm, we run the algorithm using two random uncertain systems with state-spaces of dimension $n=5$ and

10. The tests are implemented on a linux-based Karlin cluster computer at Illinois Institute of Technology. In all the runs, $D_p = (2,2,2,2), D_a = (1,1,1,1)$ and $\alpha \in \Phi_4$. Fig. 4 shows the computation time of the algorithm versus the number of processors, for two different state-space sizes and two different number of iterations (Polya's exponents d). We claim that the linearity of the curves in all cases implies near-perfect scalability of the algorithm.

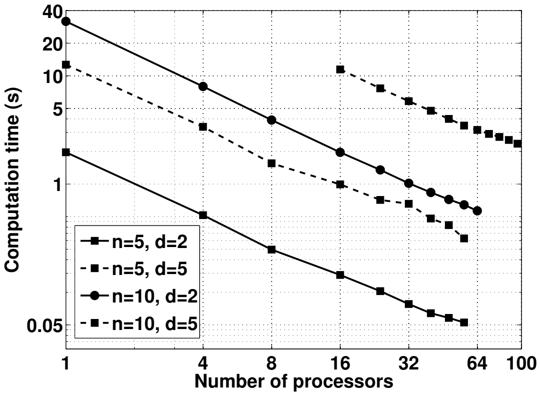

Fig. 4. Computation time of the algorithm versus the number of processors, for different state-space dimensions n and Polya's exponents d

VII. Conclusion and Future Works

In this paper, we have designed a scalable parallel algorithm to construct a set of structured LMI conditions for solving optimization problems with constraints in the form of polynomial positivity over the hypercube. This algorithm can be used to perform robust analysis and control of systems with uncertainties which lie within the hypercube. This approach can be readily extended to positivity on polytopes through vector scaling and the addition of redundant constraints. The algorithm was tested on large cluster computers and the complexity compared to the case of stability on a single simplex [25]. Near-perfect scalability of algorithm was demonstrated for large numbers of processors. Ongoing work involves extending our algorithm to the problem of nonlinear stability and implementation of our algorithm on GPU-based supercomputers.

Acknowledgments

The authors would like to gratefully acknowledge funding from NSF Grant # CMMI-1100376.

References

[1] K. Zhou and J. Doyle, *Essentials of Robust Control*. Prentice Hall, 1998.

[2] R. A. Freeman and P. V. Kokotovic, *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*. Modern Birkhauser Classics, 2008.

[3] G. E. Dullerud and F. Paganini, *A course in Robust Control Theory, A Convex Approach*. Springer, 2005.

[4] V. L. Kharitonov, "Asymptotic stability of an equilibrium position of a family of systems of linear differential equations," *ifferentsialnye Uravneniya*, vol. 14, no. 2, pp. 2086–2088, 1978.

[5] J. Shamma, "Robust stability with time-varying structured uncertainty," *Automatic Control, IEEE Transactions on*, vol. 39, no. 4, pp. 714–724, 1994.

[6] A. Packard and J. Doyle, "Quadratic stability with real and complex perturbations," *Automatic Control, IEEE Transactions on*, vol. 35, no. 2, pp. 198–201, 1990.

[7] P. Gahinet, P. Apkarian, and M. Chilali, "Affine parameter-dependent lyapunov functions and real parametric uncertainty," *IEEE Transactions on Automatic Control*, vol. 41, pp. 436–442, Mar 1996.

[8] R. C. L. F. Oliveira and P. L. D. Peres, "Stability of polytopes of matrices via affine parameter-dependent Lyapunov functions: Asymptotically exact LMI conditions," *Linear Algebra Appl.*, vol. 405, pp. 209–228, Aug 2005.

[9] P. A. Bliman, "A convex approach to robust stability for linear systems with uncertain scalar parameters," *SIAM J. Control Optim*, vol. 42, no. 3-4, pp. 2016–2042, 2004.

[10] G. Chesi, A. Garulli, A. Tesi, and A. Vicino, "Polynomially parameter-dependent lyapunov functions for robust stability of polytopic systems: an LMI approach," *IEEE Transactions on Automatic Control*, vol. 50, pp. 365–370, Mar 2005.

[11] P. A. Bliman, "An existence result for polynomial solutions of parameter-dependent LMIs," *Syst. Control Lett.*, vol. 51, pp. 165–169, Mar 2004.

[12] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," *Math. Operat. Res.*, vol. 23, no. 4, pp. 769–805, 1998.

[13] G. Chesi, "Establishing stability and instability of matrix hypercubes," *Systems & control letters*, vol. 54, no. 4, pp. 381–388, 2005.

[14] P. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2009.

[15] P. Rajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: a general purpose sum of squares programming solver," in *Proceedings of IEEE Conference on Decision and Control*, 2002.

[16] G. Stengle, "A nullstellensatz and a positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, no. 2, pp. 87–97, 1973.

[17] M. Yamashita, K. Fujisawa, and M. Kojima, "SDPARA: Semidefinite programming algorithm parallel version," *Parallel Computing*, vol. 29, pp. 1053–1067, 2003.

[18] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

[19] G. M. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," No. 30, pp. 483–485, AFIPS Conference Proceedings, 1967.

[20] G. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*. Cambridge University Press, 1934.

[21] C. W. Scherer, "Relaxations for robust linear matrix inequality problems with verifications for exactness," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 365–395, 2005.

[22] R. C. L. F. Oliveira and P. L. D. Peres, "Parameter-dependent LMIs in robust analysis: Characterization of homogeneous polynomially parameter-dependent solutions via LMI relaxations," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1334–1340, Jul 2007.

[23] R. C. L. F. Oliveira, P.-A. Bliman, and P. L. D. Peres, "Robust LMIs with parameters in multi-simplex: Existence of solutions and applications," pp. 2226–2231, Proceedings of IEEE Conference on Decision and Control, 2008.

[24] M. M. Peet and Y. V. Peet, "A parallel-computing solution for optimization of polynomials," Proceedings of the American Control Conference, Jun-Jul 2010.

[25] R. Kamyar, M. M. Peet, and Y. Peet, "Solving large-scale robust stability problems by exploiting the parallel structure of polyas theorem," *Submmited to IEEE Transactions on Automatic Control*, 2012.

[26] P. Bliman, R. C. L. F. Oliveira, V. F. Montagner, and P. L. D. Peres, "Existence of homogeneous polynomial solutions for parameter-dependent linear matrix inequalities with parameters in the simplex," in *Proceedings of the 45nd IEEE Conference on Decision and Control*, Dec 2006.

[27] R. Kamyar and M. M. Peet, "Decentralized computation for robust stability analysis of large state-space systems using polyas theorem," in *Proceedings of the American Control Conference*, Jun-Jul 2012.