Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

An nD-systems approach to global polynomial optimization with an application to H_2 model order reduction

Ivo Bleylevens, Ralf Peeters and Bernard Hanzon

***Abstract*— The problem of finding the global minimum of a multivariate polynomial can be approached by the matrix method of Stetter-Möller, which reformulates it as a large eigenvalue problem. The linear operators involved in this approach are studied using the theory of nD-systems. This supports the efficient application of iterative methods for solving eigenvalue problems such as Arnoldi methods and Jacobi-Davidson methods. This approach is demonstrated by an example which addresses optimal H_2-model reduction of a linear dynamical model of order 10 to order 9.**

***Index Terms*— global polynomial optimization, Stetter-Möller matrix method, linear operator, nD-system, large eigenvalue problem, H_2 model reduction.**

I. INTRODUCTION

Finding the global minimum of a real-valued multivariate polynomial is a problem which has several useful applications in systems and control theory. Non-convexity and the aspect of local optima, make this into a hard problem. In this paper we present a technique which uses nD-systems for finding the *global* minimum of a special class of dominated polynomials. These are polynomials of the form $p_\lambda(x_1,\ldots,x_n) = q(x_1,\ldots,x_n)+\lambda(x_1^{2d}+\ldots+x_n^{2d})$, where $q(x_1,\ldots,x_n)$ is a real polynomial of total degree less than $2d$ and where λ is a positive real number. This class is of interest because information on the global minimum of q can be obtained from p_λ by letting λ tend to zero, see [6], [8]. The method can be extended to study rational functions too, see [8].

The global minimum of a polynomial can be found by solving the system of first order conditions and computing the values at these stationary points. For a dominated polynomial p_λ (with $\lambda > 0$ fixed) this leads to a system of polynomial equations in Gröbner basis form with respect to any total degree monomial ordering. Such a system has a finite number of solutions, so that the Stetter-Möller matrix method [10] can be applied. This leads to a set of commuting matrices $(A_{x_1},\ldots,A_{x_n})$ whose eigenvalues, corresponding to a common eigenvector, yield the stationary points of p_λ. Each matrix A_{x_i} represents the linear operator of multiplication by x_i in the quotient space $\mathbb{R}[x_1,\ldots,x_n]/I$, where I is the ideal generated by the first order derivatives of p_λ. For any given polynomial $r(x_1,\ldots,x_n)$ the eigenvalues of the matrix $A_r = r(A_{x_1},\ldots,A_{x_n})$ give the values of r at the stationary points of p_λ.

I. Bleylevens and R. Peeters are with the Mathematics Department, Universiteit Maastricht, PO Box 616, 6200 MD Maastricht, The Netherlands {i.bleylevens,ralf.peeters}@math.unimaas.nl

B. Hanzon is with the School of Mathematics, University College Cork, Western Road, Cork, Ireland b.hanzon@ucc.ie

A drawback of this approach is that the size $N = (2d-1)^n$ of A_r quickly grows large. However, all that is needed for modern iterative eigenproblem solvers (e.g. based on Jacobi-Davidson or Arnoldi methods [11], [4]) is a routine which computes the action of A_r on a given vector v. The huge number of required iterations is the main reason why this action has to be computed efficiently. This paper focuses on this aspect of the optimization technique.

To avoid building A_r one can associate the system of first order derivatives of p_λ with an nD-system of difference equations, by interpreting the variables in the polynomial equations as shift operators $\sigma_1,\ldots,\sigma_n$ working on a multi-dimensional time series $y(t_1,\ldots,t_n)$. In this way, calculation of the left action of A_r on a given vector v requires solving for $y(t_1,\ldots,t_n)$ using the difference equations. The vector v corresponds to an initial state of the associated nD-system. See [1], [5] for similar ideas in the 2D case.

One way to compute efficiently the left action of A_r on v is by first setting up a corresponding shortest path problem and to apply an algorithm like Dijkstra's or Floyd's algorithm, to solve it. A drawback is that the computation of an optimal shortest path along these lines can be quite expensive. On the other hand, the numerical complexity of the computation of the left action of A_r based on a shortest path solution can be shown to depend only linearly on the total degree of the polynomial r. Interestingly, a suboptimal path which also achieves a numerical complexity which depends linearly on the total degree of r is easily designed. In the case of 2D-systems when there is no additional structure in the first order derivatives of p_λ, the shortest path problem can be solved analytically.

As an application of these techniques, the topic of H_2-model reduction from order n to $n-1$ is addressed. A worked example which involves the reduction of a system of order 10 to a system of order 9 is given in the last section.

II. ALGEBRAIC BACKGROUND

Let $q(x_1,\ldots,x_n) \in \mathbb{R}[x_1,\ldots,x_n]$ be a real polynomial of which we are interested in computing the *global infimum* over $\mathbb{R}^n$. Let d be a positive integer such that $2d$ (strictly) exceeds the total degree of $q(x_1,\ldots,x_n)$ and consider the one-parameter family of what will be called *(Minkowski-norm) dominated polynomials*:

$$p_\lambda(x_1,\ldots,x_n) := \lambda(x_1^{2d}+\ldots+x_n^{2d})+ q(x_1,\ldots,x_n), \quad \lambda \in \mathbb{R}^+ \tag{1}$$

Note that the value of p_λ is dominated by the term $\lambda(x_1^{2d}+\ldots+x_n^{2d})$ when the Minkowski $2d$-norm $\|(x_1,\ldots,x_n)\|_{2d}$

0-7803-9568-9/05/$20.00 ©2005 IEEE

becomes large. Consequently, since its total degree $2d$ is constructed to be even, the polynomial p_λ has a global minimum over $\mathbb{R}^n$ for each $\lambda \in \mathbb{R}^+$. Information about the global infimum of the polynomial $q(x_1, \ldots, x_n)$ can be obtained by studying what happens to the global minima of $p_\lambda(x_1, \ldots, x_n)$ for $\lambda \downarrow 0$, see [8]. The global minimizers of $p_\lambda(x_1, \ldots, x_n)$ are real solutions to the corresponding system of first order conditions. This leads to a system of n polynomial equations in n variables of the form

$$d^{(i)}(x_1, \ldots, x_n) = 0, \qquad (i = 1, \ldots, n), \tag{2}$$

where

$$d^{(i)}(x_1, \ldots, x_n) = x_i^{2d-1} + \frac{1}{2d\lambda}\frac{\partial}{\partial x_i} q(x_1, \ldots, x_n). \tag{3}$$

It will occasionally be convenient to write $d^{(i)}(x_1, \ldots, x_n)$ in the form $d^{(i)}(x_1, \ldots, x_n) = x_i^m - f^{(i)}(x_1, \ldots, x_n)$ with $m = 2d-1$ and $f^{(i)} \in \mathbb{R}[x_1, \ldots, x_n]$ of total degree strictly less than m. Because of this structure, the set of polynomials $\{d^{(i)} \,|\, i = 1, \ldots, n\}$ is in Gröbner basis form with respect to any total degree monomial ordering. The associated variety V, the solution set to the system of equations (2), therefore has dimension zero and the number of solutions in $\mathbb{C}^n$ is finite. The associated ideal $I = \langle d^{(i)} \,|\, i = 1, \ldots, n\rangle$ generated by these polynomials, yields a quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$ which is a finite dimensional vector space of dimension $N := m^n$. A monomial basis for this quotient space is given by the set

$$B = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \,|\, \alpha_1, \ldots, \alpha_n \in \{0, 1, \ldots, m-1\}\}. \tag{4}$$

Finite dimensionality of the quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$ makes that the matrix method of Stetter-Möller can be applied to compute the solutions to the system of equations (2), by recasting it into the form of a large eigenvalue problem.

A crucial observation in this approach is that polynomial multiplication within $\mathbb{R}[x_1, \ldots, x_n]/I$ is a *linear* operation. Given any basis for $\mathbb{R}[x_1, \ldots, x_n]/I$, for instance the basis B introduced above, it therefore is possible to compute the matrix A_r associated with the linear operation of multiplication by a polynomial $r(x_1, \ldots, x_n)$ within $\mathbb{R}[x_1, \ldots, x_n]/I$. It then holds that the eigenvalues of this matrix A_r are equal to the values of r at all the (complex) solutions of the system of equations (2). Moreover, for any two polynomials $r(x_1, \ldots, x_n)$ and $s(x_1, \ldots, x_n)$ the corresponding matrices A_r and A_s commute.

III. AN nD-SYSTEMS APPROACH

In this paper we pursue a state-space approach with respect to the computation of the action of the linear operation of multiplication by a polynomial r within $\mathbb{R}[x_1, \ldots, x_n]/I$, i.e., the action of the matrix A_r. More precisely, we will be concerned with the *left action* of A_r rather than with its right action. To this end we associate an autonomous multidimensional system (an nD-system) with the set of polynomials $d^{(i)}$. With each monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ we associate an nD-shift operator $\sigma_1^{\alpha_1} \cdots \sigma_n^{\alpha_n}$ which acts on a multidimensional time series $y_{t_1,\ldots,t_n}$ according to the rule

$$\sigma_1^{\alpha_1} \cdots \sigma_n^{\alpha_n} : y_{t_1,\ldots,t_n} \mapsto y_{t_1+\alpha_1,\ldots,t_n+\alpha_n}. \tag{5}$$

Imposing the usual linearity properties, this allows one to associate a polynomial $r(x_1, \ldots, x_n)$ with the homogeneous multidimensional difference equation $r(\sigma_1, \ldots, \sigma_n) y_{t_1,\ldots,t_n} = 0$. In this way, the system of polynomial equations (2) yields a system of n homogeneous multidimensional difference equations of the form:

$$y_{t_1,\ldots,t_{i-1},t_i+m,t_{i+1},\ldots,t_n} = f^{(i)}(\sigma_1, \ldots, \sigma_n) y_{t_1,\ldots,t_n} \tag{6}$$

for $i = 1, \ldots, n$. This expresses the fact that the value of $y_{t_1,\ldots,t_n}$ at any multidimensional 'time instant' $t = (t_1, \ldots, t_n)$ such that $\max\{t_1, \ldots, t_n\} \geq m$, can be obtained from the set of values for which the multidimensional time instants have a total time (strictly) less than the total time $|t| := t_1 + \ldots + t_n$. As a consequence, any multidimensional time series $y_{t_1,\ldots,t_n}$ satisfying this system of recursions is uniquely determined by the (ordered) set of values

$$w_{0,\ldots,0} := \{y_{t_1,\ldots,t_n} \,|\, t_1, \ldots, t_n \in \{0, 1, \ldots, m-1\}\}. \tag{7}$$

Conversely, each choice for $w_{0,\ldots,0}$ yields a corresponding solution for $y_{t_1,\ldots,t_n}$. In state-space terms, the (ordered) set of values $w_{0,\ldots,0}$ acts as an *initial state* for the autonomous homogeneous system of multidimensional difference equations (6). This point of view can be formalized by introducing the *state vector* $w_{t_1,\ldots,t_n}$ at the multidimensional time instant $(t_1, \ldots, t_n)$ as the (ordered) set of values

$$w_{t_1,..,t_n} := \{y_{t_1+s_1,..,t_n+s_n} | s_1, .., s_n \in \{0, .., m-1\}\} \tag{8}$$

According to this definition, two state vectors $w_{t_1,\ldots,t_n}$ and $w_{t_1+\alpha_1,\ldots,t_n+\alpha_n}$, with $\alpha_i \geq 0$ $(i = 1, \ldots, n)$, are related by

$$w_{t_1+\alpha_1,\ldots,t_n+\alpha_n} = \sigma_1^{\alpha_1} \cdots \sigma_n^{\alpha_n} w_{t_1,\ldots,t_n}, \tag{9}$$

where the nD-shift operates on such state vectors in an element-wise fashion. Since this operator is linear, the latter relation can also be cast in the usual matrix-vector form. This requires a choice of basis. If this choice is made to correspond to (4) it holds that

$$w_{t_1,\ldots,t_{i-1},t_i+1,t_{i+1},\ldots,t_n} = \sigma_i w_{t_1,\ldots,t_n} = A_{x_i}^T w_{t_1,..,t_n}, \tag{10}$$

where the matrix A_{x_i} again denotes the matrix previously associated with multiplication by the polynomial x_i. Note that there is a transpose involved in this relationship. As a consequence $w_{t_1+\alpha_1,\ldots,t_n+\alpha_n} = (A_{x_1}^T)^{\alpha_1} \cdots (A_{x_n}^T)^{\alpha_n} w_{t_1,\ldots,t_n}$, which shows that the general solution to the autonomous multidimensional system with initial state $w_{0,\ldots,0}$ is given by

$$w_{t_1,\ldots,t_n} = (A_{x_1}^T)^{t_1} \cdots (A_{x_n}^T)^{t_n} w_{0,\ldots,0}. \tag{11}$$

For an arbitrary polynomial $r(x_1, \ldots, x_n)$ it holds that

$$r(\sigma_1, \ldots, \sigma_n) w_{t_1,\ldots,t_n} = A_{r(x_1,\ldots,x_n)}^T w_{t_1,\ldots,t_n}. \tag{12}$$

Our interest is in computing the eigenvalues of the matrix $A_{r(x_1,\ldots,x_n)}$, which are the same as those of its transpose

$A^T_{r(x_1,\ldots,x_n)}$ and which may conveniently be studied from the perspective of the autonomous nD-system introduced above. Note that if v is a left eigenvector of A_{x_i} (hence a right eigenvector of $A^T_{x_i}$) with a corresponding eigenvalue μ_i, then it holds that $A^T_{x_i} v = \mu_i v$. In terms of the nD-system this implies that the choice $w_{0,\ldots,0} := v$ for the initial state produces a scaled version for the state: $w_{0,\ldots,0,1,0,\ldots,0} = \mu_i v$, which relates to a shift in the multidimensional time space by 1 in the direction of the i-th time axis only. However, the vectors $w_{0,\ldots,0}$ and $w_{0,\ldots,0,1,0,\ldots,0}$ have $m^n - m^{(n-1)}$ elements in common (in shifted positions), showing that the left eigenvectors of A_{x_i} exhibit a special structure and will be called *Stetter vectors*. Because all matrices A_r commute, the eigenspaces of *all* these matrices can be spanned by the *same* set of Stetter vectors. The entries of these Stetter vectors are products of powers of the eigenvalues μ_i, $(i = 1, \ldots, n)$ of the shift operators σ_i, $(i = 1, \ldots, n)$.

At a solution $(\xi_1, \ldots, \xi_n)$ to the system of equations (2) it holds that there exists a common Stetter vector v such that each of the coordinates ξ_i is an eigenvalue of the matrix A_{x_i} for which v is a left eigenvector. Then for each arbitrary polynomial r the value $r(\xi_1, \ldots, \xi_n)$ is an eigenvalue of the matrix A_r for which v is also a left eigenvector. Conversely, for each eigenvalue μ_r of A_r there exists a solution $(\xi_1, \ldots, \xi_n)$ to the system (2) such that $\mu_r = r(\xi_1, \ldots, \xi_n)$. Then there also exists a corresponding Stetter vector v which is a left eigenvector of A_r for the eigenvalue μ_r as well as a left eigenvector of each of the matrices A_{x_i} for the eigenvalue ξ_i, $(i = 1, \ldots, n)$. As a consequence, if μ_r is an eigenvalue of A_r with multiplicity one, then the associated solution $(\xi_1, \ldots, \xi_n)$ to the system (2) for which $\mu_r = r(\xi_1, \ldots, \xi_n)$ can be retrieved from a corresponding left eigenvector of A_r.

A straightforward deployment of the Stetter-Möller matrix method for computing the global minimum for p_λ now proceeds as follows. First a suitable choice for the polynomial r is made and the matrix A_r is constructed. Then its eigenvalues and eigenvectors are computed, from which the corresponding solutions to the system (2) are determined. The real solutions are plugged into the criterion $p_\lambda(x_1, \ldots, x_n)$. The smallest value yields the global minimum and any corresponding minimizer can be read off.

However, a serious bottleneck in this approach from a computational point of view is constituted by the eigenvalue/vector calculations that have to be performed for the matrix A_r. As a matter of fact, the $N \times N$ matrix A_r quickly grows large, since $N = m^n$. As we have argued above, the (left) eigenvectors of A_r can always be chosen to be structured vectors. Therefore we are dealing with a large *structured* eigenvalue problem.

IV. ITERATIVE SOLUTION METHODS FOR LARGE EIGENVALUE PROBLEMS

State-of-the-art methods for the solution of large eigenvalue problems are the iterative methods of Arnoldi ([9]) or Jacobi-Davidson ([4], [11]). Such methods have the attractive feature that they do not operate on the matrix A_r directly. Instead they iteratively perform the action of the linear operator at hand, for which it suffices to implement a computer routine that is able to compute this action for any given vector v. The nD-system approach offers a framework to compute this action by initializing the initial state as $w_{0,\ldots,0} := v$ and using the n recursions (6) in combination with the relationship (12) to obtain the vector $r(\sigma_1, \ldots, \sigma_n) w_{0,\ldots,0} = A^T_r w_{0,\ldots,0}$, which entirely avoids an explicit construction of the matrix A_r. Note that $r(\sigma_1, \ldots, \sigma_n) w_{0,\ldots,0}$ consists of a linear combination of state vectors $w_{t_1,\ldots,t_n}$; each monomial term $r_{\alpha_1,\ldots,\alpha_n} x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ that occurs in $r(x_1, \ldots, x_n)$ corresponds to a weighted state vector $r_{\alpha_1,\ldots,\alpha_n} w_{\alpha_1,\ldots,\alpha_n}$.

If attention is focused on the computation of all the *stationary points* of p_λ, then the actions of the matrices $A^T_{x_i}$, for all $i = 1, \ldots, n$, play a central role since their eigenvalues constitute the coordinates of these stationary points.

If instead attention is focused on the computation of the *values* of p_λ at the stationary points, then on the one hand the computation of the action of $A^T_{p_\lambda}$ becomes more expensive. But on the other hand, for $A^T_{p_\lambda}$ only the smallest real eigenvalue is required that corresponds to a real stationary point, an option supported by Arnoldi or Jacobi-Davidson methods; to avoid the computation of all the eigenvalues may lead to huge savings.

V. EFFICIENT COMPUTATION OF STATE VECTOR

In this section we address an efficient way for computing $y_{t_1,\ldots,t_n}$ for a given multidimensional time instant $(t_1, \ldots, t_n)$. The computational complexity that can be achieved by an optimal algorithm to compute the value of $y_{t_1,\ldots,t_n}$ from a given initial state $w_{0,\ldots,0}$ using the n difference equations (6) is addressed by the following result. For each multidimensional time instant $t = (t_1, \ldots, t_n)$ recall that the 'total time' is denoted by $|t| := t_1 + \ldots + t_n$. Then it is not difficult to show that an optimal algorithm has a computational complexity that increases linearly with the total time $|t|$.

Theorem 5.1: Consider a set of n multidimensional recursions of the form (6) and a let an initial state $w_{0,\ldots,0}$ be given. Then every algorithm that computes the value of $y_{t_1,\ldots,t_n}$, using the recursions (6), has a computational complexity which increases at least linearly with the total time $|t|$.

Proof. Each recursion from the set (6) allows one to compute the value of $y_{t_1,\ldots,t_n}$ from a set of values for which the total times are all within the range $|t|-m, |t|-m+1, \ldots, |t|-1$. The largest total time among the entries of the initial state $w_{0,\ldots,0}$ corresponds to $y_{m-1,\ldots,m-1}$ and is equal to $n(m-1)$. Therefore, to express $y_{t_1,\ldots,t_n}$ in terms of the quantities contained in the initial state requires at least $\lceil (|t| - n(m-1))/m \rceil$ applications of a recursion from the set (6). Hence, the computational complexity of any algorithm along such lines increases at least linearly with $|t|$. □

On the other hand, it is not difficult to design an algorithm which achieves a computational complexity that is indeed linear in $|t|$. This may proceed as follows. Since $y_{t_1,\ldots,t_n}$ is contained in $w_{t_1,\ldots,t_n} = (A^T_{x_1})^{t_1} \cdots (A^T_{x_n})^{t_n} w_{0,\ldots,0}$, it can be computed by the joint action of $t_1 + \ldots + t_n = |t|$

matrices of the form $A_{x_i}^T$. It is not difficult to compute a fixed uniform upper bound on the computational complexity involved in the action of each of the matrices A_{x_i}, because only the time instants that have a total time which does not exceed $n(m-1)$ can assist in this computation and their number is finite. In view of the previous theorem this shows that an optimal algorithm for the computation of $y_{t_1,\ldots,t_n}$ has a computational complexity that increases linearly with the total time $|t|$. Clearly, similar arguments and results also hold for the computation of a state vector $w_{t_1,\ldots,t_n}$. The problem of finding an *optimal* algorithm for the computation of $y_{t_1,\ldots,t_n}$ from $w_{0,\ldots,0}$ using the recursions (6), can be cast into the form of a *shortest path problem*. In general, a standard formulation of a shortest path problem requires the specification of a directed graph $G = (V, E, W, v_I, v_T)$, consisting of a set V of vertices, a set $E \subseteq V \times V$ of edges, a weight function $W : E \to \mathbb{R}$, and an indication of an initial vertex $v_I \in V$ and a terminal vertex $v_T \in V$. To compute a shortest path from v_I to v_T, which is a path connecting v_I and v_T entirely consisting of edges in E achieving a smallest total weight, one may apply any standard algorithm (e.g. Dijkstra's or Floyd's algorithm). In this formulation, the set V should correspond to the various 'states' in which the computational procedure can be and it is natural to relate a state $v \in V$ in some way to a set of multidimensional time instants $(t_1, \ldots, t_n)$ for which the value of $y_{t_1,\ldots,t_n}$ either is already available or, depending on the set-up, still requires computation. The edges E relate to 'state transitions'. Therefore they are naturally associated with the recursions in the set (6). The weight function W specifies the computational 'costs', (e.g., the number of flops) associated with these recursions. To avoid applying an infinite sequence of recursions without ever arriving at the specified multidimensional time instant $(t_1, \ldots, t_n)$, one may start from the time instant $(t_1, \ldots, t_n)$ and work backwards, by figuring out sets of time instants which may assist in the computation of $y_{t_1,\ldots,t_n}$. Note that the total time $|t|$ then provides an upper bound on the total time of all such time instants, which makes V into a finite set. Another feature of the problem is that for many subsets of computations the exact order in which they are carried out does not matter, so that a lot of permutations of actions achieve equivalent performance. Already for small values of n, m and $|t|$ this makes that the graph G can become very large. One helpful observation in constructing a useful shortest path formulation has already been mentioned above: the total time $|t|$ provides a strict upper bound on the set of time instants which may assist in the computation of $y_{t_1,\ldots,t_n}$ when a recursion from the set (6) is applied. Therefore: (i) any sequence of time instants which facilitates the computation of $y_{t_1,\ldots,t_n}$ from $w_{0,\ldots,0}$ can always be reorganized such that the total time increases monotonically; (ii) the computation of values at time instants having the same total time can be carried out in any arbitrary order. This makes it natural to relate the vertices $v \in V$ to *sets* of time instants having the same total time, rather than to individual time instants. This is formalized by the following definitions.

Definition 5.2: For $k = 1, 2, \ldots$, let T_k be the set of all multidimensional time instants $t = (t_1, \ldots, t_n) \in \mathbb{N}_0^n$ for which $|t| = k$ and $\max\{t_1, \ldots, t_n\} \geq m$. Let V_k be the power set of T_k. Let $V_\star$ be the set of time instants corresponding to the initial state $w_{0,\ldots,0}$ (i.e., for which $\max\{t_1, \ldots, t_n\} < m$).

Given a specified multidimensional time instant $t = (t_1, \ldots, t_n)$, define V as the set of all tuples $v = (v_1, \ldots, v_{|t|}) \in V_1 \times \cdots \times V_{|t|}$. Define the initial state as $v_I = (\phi, \ldots, \phi)$ (where ϕ denotes the empty set) and define the set of terminal states v_T to consist of those tuples for which $v_{|t|}$ consists of the time instant t only.

Define E as the set of all the ordered pairs $(v, \tilde{v}) \in V \times V$ such that: (i) $\tilde{v}_k = v_k$ for precisely $|t| - 1$ values of k from the set $\{1, \ldots, |t|\}$; (ii) for the unique value of k such that $\tilde{v}_k \neq v_k$ it holds that $v_k = v_{k+1} = \ldots = v_{|t|} = \phi$ and the set $\tilde{v}_k$ consists entirely of time instants at which each value of $y_{t_1,\ldots,t_n}$ can be computed from the values at the time instants contained in the union of sets $V_\star \cup v_1 \cup \ldots \cup v_{k-1}$ through the application of a single recursion from the set (6).

Define $W : E \to \mathbb{R}$ to reflect the computational costs involved in the transitions from v to $\tilde{v}$ contained in the set E. The computation of an element from $\tilde{v}_k$ through the application of a recursion from the set (6) requires a certain number of flops which is determined by the number of terms involved in that recursion. If the element from $\tilde{v}_k$ can be computed in several ways, the minimal number of flops involved should be counted. The computational costs of a transition from v to $\tilde{v}$ are defined as the sum of all the (minimal) costs to compute the elements of the set $\tilde{v}_k$.

Define the weighted directed graph G as the tuple $G = (V, E, W, v_I, v_T)$. This specifies an associated shortest path problem which models the optimal computation of $y_{t_1,\ldots,t_n}$ from $w_{0,\ldots,0}$ using the recursions (6).

Note that the graph G has a tree structure rather than a network structure, which makes it possible to apply branch and bound techniques for tree searching. Note also that the set of terminal states v_T is easily replaced by a single terminal state, by connecting the states in v_T to a joint end node $v_\dagger$ with a zero associated cost. An interesting *relaxation* of this shortest path formulation is obtained when the condition (ii) in the definition of the set of edges E is replaced by the condition that the set $\tilde{v}_k$ consists entirely of time instants at which each value of $y_{t_1,\ldots,t_n}$ can be computed from the values at the time instants contained in the union of sets $V_\star \cup T_1 \cup \ldots \cup T_{k-2} \cup v_{k-1}$ through the application of a single recursion from the set (6). In this case, each state $v \in V$ may be restricted to the element v_k for which k is as large as possible with v_k non-empty (and the definition of the edges E should be adapted accordingly). This further reduces the size of the shortest path problem. The value of the shortest path thus obtained provides a lower bound for the value of a true optimal path, but in case the recursions in (6) are not sparse, this lower bound is likely to be close to the optimal value.

For the two-dimensional case in the situation where the recursions in (6) describe 'full patterns' (i.e., all the terms

possibly participating in the recursions are present), it is possible to solve the relaxation of the shortest path problem analytically. This solution also applies to the shortest path problem itself and it allows for the design of an optimal algorithm for the computation of y_{t_1,t_2} from $w_{0,0}$ and the two available recursions. When displayed on a 2D grid, the computation of a point located at (t_1, t_2) requires the values of m consecutive points on the diagonal just below the diagonal for (t_1, t_2). To compute these m points, $2m-1$ consecutive points are required on the diagonal one further below. However, to compute these $2m-1$ points, again only $2m-1$ points on the next diagonal below are required and this pattern continues.

Because the shortest path problem in the setting described above has a high computational complexity, some heuristic methods to compute $y_{t_1,\ldots,t_n}$ were developed and tested: (i) The *linear method* first calculates all the values $y_{t_1,\ldots,t_n}$ with $|t| \leq n(m-1)$, which precisely covers the hypercube of initial values $w_{0,\ldots,0}$. Then the pattern of values with $|t| = n(m-1)$ is shifted step by step, with each step involving a single shift in one of the axis directions, until the requested location is reached. Note that the shifted patterns of values can always be computed from the values already available. (ii) The *diagonal method* proceeds by computing the values $y_{t_1,\ldots,t_n}$ for all the time instances for which $|t|$ is constant, increasing $|t|$ one by one, until the requested location is reached. (iii) The *equalizing method* computes a requested point by applying the recursion which reduces the largest coordinate of the time instant. (iv) The *axis method* computes a requested point by applying the recursion which reduces the smallest possible coordinate of the time instant.

In higher dimensions (e.g., $n > 10$) the linear method encounters serious problems because the simplex entirely covering the initial hypercube becomes very large. The 'equalizing method' performs best for points having (almost) equal coordinates. The 'axis method' performs better for points near the coordinate axes.The 'diagonal method' never exhibits a linear numerical complexity with respect to $|t|$ and is therefore very inefficient.

To support some of these statements, simulation experiments have been performed with $n = 2$ and $m = 3$, where the state vectors w_{t_1,t_2} for the time instances $t_{0,500}$, $t_{125,375}$, $t_{250,250}$, $t_{375,125}$ and $t_{500,0}$ are to be computed. In Figure 1 the points are displayed which are computed to facilitate the computation of the desired state vectors from the 3×3 initial state $w_{0,0}$. In the upper left corner the results for the linear method are plotted. This way of computing the time instances is the most efficient. In the upper right plot the points calculated by the inefficient diagonal method are shown. The lower left plot displays the points required by the equalizing method and the lower right plot concerns the axis method. The number of points evaluated by these four methods are 7460, 127756, 113988 and 54597, respectively. Because the linear method is efficient for small values of n but becomes inefficient when the dimension of the problem increases, a fifth method has been implemented with almost the same performance as the linear method. This method

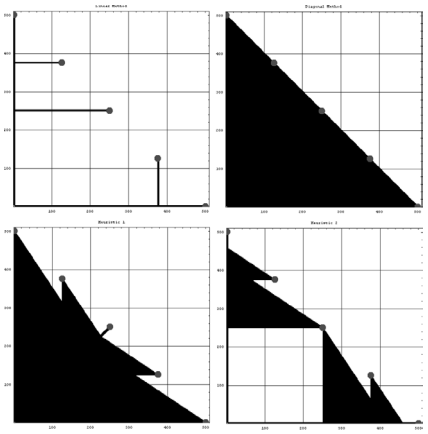

Fig. 1. Points calculated by the various methods

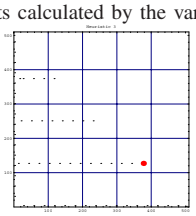

Fig. 2. Points calculated by the fifth method

proceeds by keeping track of values already required for computation and applying at each point a recurrence relation that needs a minimal number of new values to be calculated (see figure 2). This method turns out to be more efficient than the linear method for larger values of n.

VI. H_2 MODEL REDUCTION FROM ORDER n TO ORDER $n-1$

A. Theory

Suppose we are dealing with a (causal) stable linear time-invariant SISO system in continuous time, with transfer function $H(s)$. If the system involves a finite dimensional state space, then the dimension n of the state space is called the system order and $H(s)$ is a proper rational function of McMillan degree n. In [7] a method is described for computing a globally optimal H_2-approximation $G(s)$ of order $n-1$ for such a given function $H(s)$. Only the case where $H(s)$ has n distinct poles is examined. Note that the H_2-norm provides a system theoretically meaningful way to measure the distance between $H(s)$ and $G(s)$. In [7], this problem is transformed into the problem of finding the solution set of a system of n quadratic polynomial equations in n variables $x_1, \ldots, x_n$:

$$\begin{aligned} x_1^2 &= m_{1,1}x_1 + m_{1,2}x_2 + \ldots + m_{1,n}x_n \\ &\vdots \\ x_n^2 &= m_{n,1}x_1 + m_{n,2}x_2 + \ldots + m_{n,n}x_n \end{aligned}$$

This system is in Gröbner basis form; the coefficients $m_{i,j}$ are complex. Therefore the system has a finite set of (complex) solutions. For each solution a corresponding approximation $G(s)$ can be computed. But $G(s)$ is only a feasible solution if it is real and stable. Several local minima will exist and there will always be a global minimum.

As it happens, a complex homogeneous polynomial p of degree 3 can be computed that coincides with the H_2-model reduction criterion at the stationary points $(x_1, \ldots, x_n)$: $p(x_1, \ldots, x_n) = p_1 x_1^3 + \ldots + p_n x_n^3$. Using the system of quadratic equations as the set of polynomials $d^{(i)}$ in (2) and the polynomial p of order 3 as the function r in (12), the theory of the previous sections for computing the global optimum of a polynomial can be applied. Note that in [7] the matrices A_{x_i} were constructed explicitly; the techniques of the present paper make it possible to avoid this. This has the benefit that the matrices A_{x_i} which are of size $2^n \times 2^n$ do not have to be stored into memory, which may lead to significant savings on required memory and CPU time. In [7] the highest possible order to perform model reduction was 9, for larger values severe memory problems occurred. Here we demonstrate the potential of the present paper by an example which involves the reduction of a system of order 10 to a system of order 9.

B. An example

The stable system of order 10 considered in this example has the following transfer function:

$$(8.00769 \cdot 10^{-6} + 1.71974s + 260.671s^2 + 1254.63s^3 + 899.401s^4 + 1246.85s^5 + 181.506s^6 + 276.659s^7 - 1.22269s^8 + 15.77s^9)/ (1.60154 \cdot 10^{-7} + 0.0541886s + 17.9691s^2 + 147.766s^3 + 138.541s^4 + 252.726s^5 + 99.6262s^6 + 71.4313s^7 + 19.6362s^8 + 5.15548s^9 + s^{10})$$

To compute the reduced order model, a system of 10 quadratic equations needs to be solved, yielding a finite number of complex solutions $(x_1, \ldots, x_{10})$. The associated complex homogeneous polynomial of degree 3 which coincides with the H_2-criterion function at these solutions is computed as: $3.83344 \cdot 10^{-20} x_1^3 + (1.36467 \cdot 10^{-14} - 1.00376 \cdot 10^{-15} i) x_2^3 + (1.36467 \cdot 10^{-14} + 1.00376 \cdot 10^{-15} i) x_3^3 + (3.58175 \cdot 10^{-7} + 2.48189 \cdot 10^{-7} i) x_4^3 + (3.58175 \cdot 10^{-7} - 2.48189 \cdot 10^{-7} i) x_5^3 + 0.415865 x_6^3 - (9.03372 \cdot 10^{-10} - 1.13998 \cdot 10^{-9} i) x_7^3 - (9.03372 \cdot 10^{-10} + 1.13998 \cdot 10^{-9} i) x_8^3 + 1.93482 \cdot 10^7 x_9^3 + 1.97345 \cdot 10^{13} x_{10}^3$.

All the calculations were done on a Dual Xeon Pentium IV 3.2 GHz with 4096 MB internal memory. To be able to compare the results of the approach of the present paper with the methodology described in [7], we first constructed the 1024×1024 matrix A_p explicitly with *Mathematica* 5.0.1. Then the built-in function `Eigensystem` was used to calculate the eigensystem of A_p. Together this required 30332 seconds and 350 MB of memory. The matrix A_p has 45 real eigenvalues, with the smallest eigenvalue computed as $4.895931822960051 \cdot 10^{-4}$. This global minimum has as coordinates $x_1 = 78.6020 - 4.25234 \cdot 10^{-7} i$, $x_2 = -0.180875 - 0.343180i$, $x_3 = -0.180875 + 0.343180i$, $x_4 = -0.00155184 - 0.00377204i$, $x_5 = -0.00155184 + 0.00377204i$, $x_6 = 0.000289407$, $x_7 = -0.00193041 + 0.00371561i$, $x_8 = -0.00193041 - 0.00371561i$, $x_9 = 5.99264 \cdot 10^{-6}$, $x_{10} = 2.91654 \cdot 10^{-6}$. The corresponding optimal stable approximation $G(s)$ of order 9 is given by:

$$(-1.14790 + 261.273s + 1254.42s^2 + 899.312s^3 + 1246.83s^4 + 181.477s^5 + 276.658s^6 - 1.22433s^7 + 15.7700s^8)/ (0.0538714 + 17.9585s + 147.756s^2 + 138.523s^3 + 252.714s^4 + 99.6209s^5 + 71.4287s^6 + 19.6358s^7 + 5.15532s^8 + s^9)$$

To compute only the smallest non-zero real eigenvalue/eigenvector, we used iterative sparse eigenvalue solvers which allow for the computation of the eigenvalues/eigenvectors of the matrices without specifying them explicitly. For this purpose the methods `Jdqr`, `Jdqz` and `Eigs` can be used. `Eigs` is a standard Matlab routine which uses the (restarted) Arnoldi-methods through ARPACK ([9]). `Jdqr` (eigensolver) and `Jdqz` (generalized eigensolver) use Jacobi-Davidson methods ([4], [11]).

Because the condition numbers of the operators involved are very large, a balancing technique was used (see [3]) which reduces the norm of the matrix by using methods of Perron-Frobenius and the direct iterative method.

The smallest real non-zero eigenvalue was computed using the software `Jdqz`. For the operator associated with A_p the zero solution was discarded which decreases the dimension of the operator by one to 1023. The computation by `Jdqz` took 23800 seconds and used 150 MB of memory. The smallest real eigenvalue computed in this way is $4.895931819886820 \cdot 10^{-4}$. The corresponding solution is computed from the Stetter vector as: $x_1 = 78.6020 - 4.25235 \cdot 10^{-7} i$, $x_2 = -0.180875 - 0.343180i$, $x_3 = -0.180875 + 0.343180i$, $x_4 = -0.00155199 - 0.00377204i$, $x_5 = -0.00155184 + 0.00377205i$, $x_6 = 0.000289407 - 5.73819 \cdot 10^{-11} i$, $x_7 = -0.00193041 + 0.00371561i$, $x_8 = -0.00193042 - 0.00371561i$, $x_9 = 5.99264 \cdot 10^{-6} + 3.38181 \cdot 10^{-14} i$, $x_{10} = 2.91654 \cdot 10^{-6} - 1.574765 \cdot 10^{-14} i$. The stable approximation $G(s)$ of order 9 computed from this solution happens to agree entirely with the approximation previously given above, showing the potential of this approach.

REFERENCES

[1] S. Attasi, Modelling and recursive estimation for double indexed sequences, in: R.K. Mehra and D.G. Lainiotis (eds.), *System Identification: Advances and Case Studies*, pp. 289–348, Academic Press, New York, 1976.

[2] I. Bleylevens, B. Hanzon and R. Peeters, A multidimensional systems approach to polynomial optimization, *Proceedings of the MTNS 2004*, July 5-9, 2004, Leuven, Belgium, 2004.

[3] T.-Y. Chen and J. Demmel, Balancing sparse matrices for computing eigenvalues, *Linear Algebra and Its Applications*, **309**, pp. 261–287, 2000.

[4] D.R. Fokkema, G.L.G. Sleijpen and H.A. van der Vorst, Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils, *SIAM J. Sc. Comput.*, **20**, (1), pp. 94–125, 1998.

[5] E. Fornasini, P. Rocha and S. Zampieri, State space realization of 2-D finite-dimensional behaviours, *SIAM J. Contr. Opt.*, **31**, (6), pp. 1502–1517, 1993.

[6] B. Hanzon and D. Jibetean, Global minimization of a multivariate polynomial using matrix methods, *Journal of Global Optimization*, **27**, 1–23, 2003.

[7] B. Hanzon, J.M. Maciejowski, C.T. Chou, Model reduction in H2 using matrix solutions of polynomial equations, Techn. Rep. CUED/F-INFENG/TR.314, Engin. Dept., Cambridge, UK, 1998.

[8] D. Jibetean, Algebraic Optimization with Applications to System Theory, Ph.D. Thesis, Vrije Universiteit Amsterdam, 2003.

[9] R.B. Lehoucq, D.C. Sorensen and C. Yang, ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, *SIAM Publications*, Philadelphia, 1998.

[10] H.M. Möller and H.J. Stetter, Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems, *Numerische Mathematik*, **70**, pp. 311-329, 1995.

[11] G.L.G. Sleijpen and H.A. van der Vorst, A Jacobi-Davidson iteration method for linear eigenvalue problems, *SIAM J. Matrix Anal. Appl.*, **17**, pp. 401–425, 1996.