Available online at www.sciencedirect.com
ScienceDirect
Journal of Symbolic Computation 42 (2007) 30–53
Journal of Symbolic Computation
www.elsevier.com/locate/jsc

Efficiency improvement in an nD systems approach to polynomial optimization

Ivo Bleylevens[a,*], Ralf Peeters[a], Bernard Hanzon[b]

[a] *Department of Mathematics, Universiteit Maastricht, Maastricht, The Netherlands*
[b] *School of Mathematical Sciences, University College Cork, Cork, Ireland*

Received 6 November 2005; accepted 12 March 2006
Available online 15 September 2006

Abstract

The problem of finding the global minimum of a so-called Minkowski-norm dominated polynomial can be approached by the matrix method of Stetter and Möller, which reformulates it as a large eigenvalue problem. A drawback of this approach is that the matrix involved is usually very large. However, all that is needed for modern iterative eigenproblem solvers is a routine which computes the action of the matrix on a given vector. This paper focuses on improving the efficiency of computing the action of the matrix on a vector. To avoid building the large matrix one can associate the system of first-order conditions with an nD system of difference equations. One way to compute the action of the matrix efficiently is by setting up a corresponding shortest path problem and solving it. It turns out that for large n the shortest path problem has a high computational complexity, and therefore some heuristic procedures are developed for arriving cheaply at suboptimal paths with acceptable performance.

Keywords: Global polynomial optimization; Gröbner basis; Stetter–Möller matrix method; nD systems; Large eigenvalue problem

1. Introduction

Finding the global minimum of a real-valued multivariate polynomial is a problem which has several useful applications in systems and control theory as well as in many other quantitative

* Corresponding address: Department of Mathematics, Universiteit Maastricht, P.O. Box 616, 6200 Maastricht, The Netherlands. Tel.: +31 43 388 3497; fax: +31 43 388 4910.
E-mail addresses: i.bleylevens@math.unimaas.nl (I. Bleylevens), ralf.peeters@math.unimaas.nl (R. Peeters), b.hanzon@ucc.ie (B. Hanzon).

0747-7171/$ - see front matter
doi:10.1016/j.jsc.2006.03.008

sciences including statistics, mathematical finance, economics, systems biology etc. Non-convexity and the possible existence 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 $p_\lambda(x_1, \ldots, x_n) = q(x_1, \ldots, x_n) + \lambda(x_1^{2d} + \cdots + x_n^{2d})$, 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 Hanzon and Jibetean (2003) and Jibetean (2003). Extensions and further applications are discussed in Section 9.

If a polynomial has a global minimum then it can be found by solving the system of first-order conditions and computing the critical values. For a dominated polynomial p_λ 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 can be applied; see Möller and Stetter (1995). This leads to a set of commuting $N \times N$ matrices $A_{x_1}, \ldots, A_{x_n}$ whose eigenvalues, corresponding to common eigenvectors, 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. This is discussed in Section 2.

A drawback of this approach is that $N = (2d-1)^n$ is usually very large. However, all that is needed for modern iterative eigenproblem solvers (e.g. based on Jacobi–Davidson (Sleijpen and van der Vorst, 1996; Fokkema et al., 1998) or Arnoldi methods (Lehoucq et al., 1998)) is a routine which computes the action of the matrix at hand on a given vector v. These solvers also allow one to focus on certain subsets of eigenvalues. The huge number of required iterations is the main reason why the action of a matrix A_r has to be computed efficiently. This paper focuses on this aspect of the optimization technique.

To avoid building the large matrix 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 multidimensional time series $y_{t_1,t_2,\ldots,t_n}$. This set-up is presented in Section 3. Then calculation of the action of A_r^{T} on a given vector v requires solving for $y_{t_1,t_2,\ldots,t_n}$ using the difference equations. (Note that A_r and A_r^{T} have the same eigenvalues.) The vector v corresponds to an initial state of the associated nD system. See Attasi (1976) and Fornasini et al. (1993) for similar ideas in the 2D case.

One way to compute efficiently the action of A_r^{T} on v is by first setting up a corresponding shortest path problem and applying an algorithm, like Dijkstra's algorithm (Dijkstra, 1959) or Floyd's algorithm (Floyd, 1962), to solve it. This is the topic of Section 6. 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 action of A_r^{T} based on a shortest path solution can be shown to depend only linearly on the total degree of the polynomial r. Interestingly, suboptimal paths can easily be designed which also achieve a numerical complexity which depends linearly on the total degree of r. 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. For 3D systems the situation is more complicated but a number of partial results are available and presented in this paper.

In Section 8 the approach of this paper is demonstrated by means of a worked example and compared to other approaches available in the literature: SOSTOOLS, GloptiPoly and SYNAPS. It will turn out that at this stage of the development our method is outperformed by these

techniques in computation time but our approach still has attractive features, as will be discussed in Section 9. For further background of the constructive algebra and systems theory aspects of this approach one can refer to (Hanzon and Hazewinkel, 2006).

2. Algebraic background

Let $q(x_1, \ldots, x_n) \in \mathbb{R}[x_1, \ldots, x_n]$ be a real polynomial. We are interested in computing its *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*

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

Note that the nomenclature for this family derives from the property that the value of p_λ is dominated by the term $\lambda(x_1^{2d} + \cdots + x_n^{2d})$ when the Minkowski $2d$-norm $\|(x_1, \ldots, x_n)\|_{2d} = (x_1^{2d} + x_2^{2d} + \cdots + x_n^{2d})^{1/(2d)}$ becomes large. Consequently, the polynomial p_λ has a global minimum over $\mathbb{R}^n$ for each $\lambda \in \mathbb{R}^+$. In fact information about the infimum (and its 'location') of the polynomial $q(x_1, \ldots, x_n)$ can be obtained by studying what happens to the global minima and the corresponding minimizing points, of $p_\lambda(x_1, \ldots, x_n)$ for $\lambda \downarrow 0$; see Hanzon and Jibetean (2003) and Jibetean (2003). The global minimizers of $p_\lambda(x_1, \ldots, x_n)$ are of course among the stationary points of this polynomial, which are the 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

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

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

It will be convenient to write $d^{(i)}(x_1, \ldots, x_n)$ in the form

$$d^{(i)}(x_1, \ldots, x_n) = x_i^m - f^{(i)}(x_1, \ldots, x_n), \quad (i = 1, \ldots, n), \tag{3}$$

with $m = 2d - 1$ and $f^{(i)} = -\frac{1}{2d\lambda}\frac{\partial}{\partial x_i} q(x_1, \ldots, x_n) \in \mathbb{R}[x_1, \ldots, x_n]$ of total degree strictly less than m. Because of this structure, (i) the set of polynomials $\{d^{(i)} \mid i = 1, \ldots, n\}$ is in Gröbner basis form with respect to any total degree monomial ordering and (ii) the associated variety V, the solution set to the system of equations (2), has dimension zero and the number of solutions in $\mathbb{C}^n$ is finite. (For further details see Cox et al. (1998) or Proposition 3.1 and Theorem 2.1 in Hanzon and Jibetean (2003) and the references given there.) The associated ideal $I = \langle d^{(i)} \mid i = 1, \ldots, n \rangle$ 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

$$B = \{x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \mid \alpha_1, \alpha_2, \ldots, \alpha_n \in \{0, 1, \ldots, m-1\}\}. \tag{4}$$

For definiteness we will choose the total degree reversed lexicographical monomial ordering throughout this paper, unless stated otherwise. Note that any other total degree monomial ordering could be chosen instead.

Finite dimensionality of the quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$ means that the matrix method of Stetter and Möller can be applied to compute all the (complex) solutions to the system of equations $d^{(i)}(x_1, \ldots, x_n) = 0$, $(i = 1, \ldots, n)$, by recasting it into the form of a large eigenvalue problem. (For further details see e.g. Hanzon et al. (1998).)

The n-tuple of commuting matrices $(A_{x_1}, A_{x_2}, \ldots, A_{x_n})$ yields a *matrix solution* of the system of polynomial equations (2). Any common eigenvector of these matrices $A_{x_1}, A_{x_2}, \ldots, A_{x_n}$ leads to a scalar solution, constituted by the corresponding n-tuple of eigenvalues. All scalar solutions can be obtained in this way. 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 $d^{(i)}(x_1, \ldots, x_n) = 0$ $(i = 1, \ldots, n)$. 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.

3. An *n*D systems approach

To compute the eigenvalues of A_r, one may as well consider its transpose A_r^{T}. In this section we pursue a state-space approach with respect to the computation of the action of A_r^{T} on a vector v. To this end we set up an autonomous multidimensional system, also called an nD system, associated with the set of polynomials $d^{(i)}$ $(i = 1, \ldots, n)$. With any monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ we associate an nD shift operator $\sigma_1^{\alpha_1} \sigma_2^{\alpha_2} \cdots \sigma_n^{\alpha_n}$ which acts on any multidimensional time series $y_{t_1,t_2,\ldots,t_n}$ according to the rule

$$\sigma_1^{\alpha_1} \sigma_2^{\alpha_2} \cdots \sigma_n^{\alpha_n} : y_{t_1,t_2,\ldots,t_n} \mapsto y_{t_1+\alpha_1,t_2+\alpha_2,\ldots,t_n+\alpha_n}. \tag{5}$$

Imposing the usual linearity properties, this allows one to associate a homogeneous multidimensional difference equation with an arbitrary polynomial $r(x_1, x_2, \ldots, x_n)$ as follows:

$$r(\sigma_1, \sigma_2, \ldots, \sigma_n) y_{t_1,t_2,\ldots,t_n} = 0. \tag{6}$$

Applying this set-up to the system of polynomial equations (2) a system of n linear homogeneous multidimensional difference equations is obtained, which can be written in the form

$$y_{t_1,\ldots,t_{i-1},t_i+m,t_{i+1},\ldots,t_n} = f^{(i)}(\sigma_1, \ldots, \sigma_n) y_{t_1,t_2,\ldots,t_n}, \quad (i = 1, \ldots, n). \tag{7}$$

This expresses the fact that the value of $y_{\bar{t}_1,\bar{t}_2,\ldots,\bar{t}_n}$ at any multidimensional 'time instant' $\bar{t} = (\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_n)$, such that $\max\{\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_n\}$ is greater than or equal to m, can be obtained from the set of values of $y_{t_1,t_2,\ldots,t_n}$ for which the multidimensional time instants have a total time $|t| := t_1 + t_2 + \cdots + t_n$ strictly less than the total time $|\bar{t}| = \bar{t}_1 + \bar{t}_2 + \cdots + \bar{t}_n$. As a consequence, any multidimensional time series $y_{t_1,t_2,\ldots,t_n}$ satisfying this system of recursions is uniquely determined by the finite set of (total degree reversed lexicographically ordered) values:

$$w_{0,0,\ldots,0} := \{y_{t_1,t_2,\ldots,t_n} \mid t_1, t_2, \ldots, t_n \in \{0, 1, \ldots, m-1\}\}. \tag{8}$$

Conversely, each choice for $w_{0,0,\ldots,0}$ yields a corresponding solution for $y_{t_1,t_2,\ldots,t_n}$. In state-space terms, the set of values $w_{0,0,\ldots,0}$ acts as an *initial state* for the autonomous homogeneous system of multidimensional difference equations (7). This point of view can be formalized by introducing the *state vector* $w_{t_1,t_2,\ldots,t_n}$ at the multidimensional time instant $(t_1, t_2, \ldots, t_n)$ as the set of values

$$w_{t_1,t_2,\ldots,t_n} := \{y_{t_1+s_1,t_2+s_2,\ldots,t_n+s_n} \mid s_1, s_2, \ldots, s_n \in \{0, 1, \ldots, m-1\}\}. \tag{9}$$

According to this definition, two state vectors $w_{t_1,t_2,\ldots,t_n}$ and $w_{t_1+\alpha_1,t_2+\alpha_2,\ldots,t_n+\alpha_n}$, with $\alpha_i \geq 0$ for all $i = 1, \ldots, n$, are related by

$$w_{t_1+\alpha_1,t_2+\alpha_2,\ldots,t_n+\alpha_n} = \sigma_1^{\alpha_1}\sigma_2^{\alpha_2}\cdots\sigma_n^{\alpha_n} w_{t_1,t_2,\ldots,t_n}, \tag{10}$$

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 the basis B for the quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$ and the associated monomial ordering, it holds that

$$w_{t_1,\ldots,t_{i-1},t_i+1,t_{i+1},\ldots,t_n} = \sigma_i w_{t_1,t_2,\ldots,t_n} = A_{x_i}^{\mathrm{T}} w_{t_1,t_2,\ldots,t_n}, \tag{11}$$

where the matrix A_{x_i} again denotes the matrix associated with multiplication by the polynomial x_i within the quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$. As a consequence, the general solution to the autonomous multidimensional system with initial state $w_{0,0,\ldots,0}$ is given by

$$w_{t_1,t_2,\ldots,t_n} = (A_{x_1}^{\mathrm{T}})^{t_1}(A_{x_2}^{\mathrm{T}})^{t_2}\cdots(A_{x_n}^{\mathrm{T}})^{t_n} w_{0,0,\ldots,0}. \tag{12}$$

More generally, for an arbitrary polynomial $r(x_1, \ldots, x_n)$ it holds that

$$r(\sigma_1, \ldots, \sigma_n) w_{t_1,\ldots,t_n} = A_{r(x_1,\ldots,x_n)}^{\mathrm{T}} w_{t_1,\ldots,t_n}. \tag{13}$$

Note that $y_{t_1,t_2,\ldots,t_n}$ can be read off from this as being an element of $w_{t_1,t_2,\ldots,t_n}$.

Example 1. Let the dominated polynomial to be minimized be chosen as: $p_\lambda(x_1, x_2) = \lambda(x_1^4 + x_2^4) + x_1^3 + 2x_1^2 + 3x_1x_2$ for which $n = 2$, $d = 2$, $\lambda = 1$ and $q(x_1, x_2) = x_1^3 + 2x_1^2 + 3x_1x_2$. The first-order conditions are given by

$$\begin{cases} d^{(1)}(x_1, x_2) = x_1^3 + \frac{3}{4}x_1^2 + x_1 + \frac{3}{4}x_2 = 0 \\ d^{(2)}(x_1, x_2) = x_2^3 + \frac{3}{4}x_1 = 0 \end{cases}$$

of which the real solutions constitute the stationary points of $p_1(x_1, x_2)$. The ideal I is generated by $d^{(1)}(x_1, x_2)$ and $d^{(2)}(x_1, x_2)$. The quotient space $\mathbb{R}[x_1, \ldots, x_n]/I$ is of dimension $(2d-1)^n = 9$. A basis is given by the set of monomials $\{1, x_1, x_1^2, x_2, x_1x_2, x_1^2x_2, x_2^2, x_1x_2^2, x_1^2x_2^2\}$. In terms of this basis (which corresponds to a convenient permutation of the total degree reversed lexicographic monomial ordering) the matrices A_{x_1} and A_{x_2} are easily computed as

$$A_{x_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{9}{16} \\ 0 & 1 & -\frac{3}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{3}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{3}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\frac{3}{4} \end{pmatrix},$$

$$A_{x_2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{3}{4} & 0 & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{3}{4} & \frac{9}{16} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{9}{16} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Of course, for such small sizes computations are easy and it is not difficult to verify that A_{x_1} and A_{x_2} commute. Both matrices have distinct eigenvalues, which can be paired according to the existence of common eigenspaces, yielding nine solutions in $\mathbb{C}^2$. To focus attention on the computation of the actions of the transposed matrices $A_{x_1}^{\mathrm{T}}$ and $A_{x_2}^{\mathrm{T}}$ directly from $d^{(1)}(x_1, x_2)$ and $d^{(2)}(x_1, x_2)$, without computing these matrices explicitly, the following associated system of two-dimensional difference equations is considered:

$$\begin{cases} y_{t_1+3,t_2} = -\frac{3}{4} y_{t_1+2,t_2} - y_{t_1+1,t_2} - \frac{3}{4} y_{t_1,t_2+1} \\ y_{t_1,t_2+3} = -\frac{3}{4} y_{t_1+1,t_2}. \end{cases}$$

An initial state for this autonomous two-dimensional system is constituted by $w_{0,0} = (y_{0,0}, y_{1,0}, y_{2,0}, y_{0,1}, y_{1,1}, y_{2,1}, y_{0,2}, y_{1,2}, y_{2,2})^{\mathrm{T}}$ which corresponds to a 3×3 square array of values at integer time instants (t_1, t_2) in the two-dimensional time plane. Then the action of the matrix $A_{x_1}^{\mathrm{T}}$ on $w_{0,0}$ yields the vector $A_{x_1}^{\mathrm{T}} w_{0,0} = w_{1,0} = (y_{1,0}, y_{2,0}, y_{3,0}, y_{1,1}, y_{2,1}, y_{3,1}, y_{1,2}, y_{2,2}, y_{3,2})^{\mathrm{T}}$, which also corresponds to a 3×3 square array of integer time instants (t_1, t_2) in the two-dimensional time plane. Compared to the location of the initial state $w_{0,0}$, this array is shifted by one unit along the t_1-axis. Similarly, the action of the matrix $A_{x_2}^{\mathrm{T}}$ on $w_{0,0}$ yields the vector $A_{x_2}^{\mathrm{T}} w_{0,0} = w_{0,1} = (y_{0,1}, y_{1,1}, y_{2,1}, y_{0,2}, y_{1,2}, y_{2,2}, y_{0,3}, y_{1,3}, y_{2,3})^{\mathrm{T}}$. The action of the matrix $p_1(A_{x_1}^{\mathrm{T}}, A_{x_2}^{\mathrm{T}})$ on $w_{0,0}$ can be obtained by a linear combination of the individual actions of the matrices $(A_{x_1}^{\mathrm{T}})^4$, $(A_{x_2}^{\mathrm{T}})^4$, $(A_{x_1}^{\mathrm{T}})^3$, $(A_{x_1}^{\mathrm{T}})^2$ and $A_{x_1}^{\mathrm{T}} A_{x_2}^{\mathrm{T}}$.

A straightforward deployment of the Stetter–Möller matrix method for computing the global minimum and an associated global minimizer (over $\mathbb{R}^n$) for the real dominated polynomial p_λ now proceeds as follows. First a suitable choice for the polynomial r is made and the corresponding matrix A_r^{T} is constructed. Then its eigenvalues and corresponding eigenvectors are computed. If the eigenvalues all have geometric multiplicity one, then each eigenvector can be applied to the matrices $A_{x_i}^{\mathrm{T}}$ to obtain their eigenvalues ξ_i, $i = 1, \ldots, n$. The point $(\xi_1, \ldots, \xi_n)$ then yields a complex solution of the system of equations (2) and all solutions can be obtained in this way. By choosing r appropriately one can ascertain that all the eigenvalues of A_r^{T} have geometric multiplicity one. This will in fact hold for a 'generic' choice of r. Having found all the solutions in $\mathbb{C}^n$, we just restrict to the real solutions. These real solutions are then plugged into the criterion $p_\lambda(x_1, \ldots, x_n)$. The smallest value obtained in this way yields the global minimum and the corresponding minimizer(s) can be read off.

However, a serious bottleneck in this approach from a computational point of view is constituted by solving the eigenproblem of the matrix A_r^{T}. As a matter of fact, the $N \times N$ matrix A_r^{T} quickly grows large, since $N = m^n$. On the other hand, A_r^{T} is related to the nD system (7)

and in addition the matrix A_r^{T} may be highly sparse and structured. This holds in particular for the choices $r(x_1, \ldots, x_n) = x_i$.

Note that also the eigenvectors are structured: if v is an eigenvector of $A_{x_i}^{\mathrm{T}}$ with a corresponding eigenvalue ξ_i, then $A_{x_i}^{\mathrm{T}} v = \xi_i v$. 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: $w_{0,\ldots,0,1,0,\ldots,0} = \xi_i v$, which relates to a shift in the multidimensional time space by 1 in the direction of the ith time axis only.

Remark 2. If all the eigenvalues of A_r^{T} have (algebraic) multiplicity one, then it is well known that $A_r^{\mathrm{T}} = V \Lambda L^{\mathrm{T}}$, where $L^{\mathrm{T}} V = I$ and V is a generalized Vandermonde matrix. The rows of V^{T} consist of the ordered basis B evaluated at the points $x = v(k) \in \mathbb{C}^n$, $k = 1, 2, \ldots, N$, where $v(1), v(2), \ldots, v(N)$ are the complex solutions to the system of equations (2). Furthermore $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix with $\lambda_k = r(v(k))$, $k = 1, 2, \ldots, N$. The matrix L has the property that its kth column consists of the coefficients l_{jk} of the Lagrange interpolation polynomials $l_k(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{N} l_{j,k} b(j)$, where $b(j)$ denotes the jth basis element in B, with the basic interpolation property that $l_k(v(k)) = 1$ and $l_k(v(j)) = 0$ for all $j \neq k$. In linear algebra terms this says that $L^{\mathrm{T}} = V^{-1}$. The fact that the eigenvectors of A_r^{T} are columns of the generalized Vandermonde matrix V was noted by Stetter, which is the reason that eigenvectors of this form are sometimes called *Stetter vectors*. The structure of these vectors can be used to read off the point $v(k)$ directly without having to apply the eigenvectors to A_{x_i}, $i = 1, 2, \ldots, n$. This is because the monomials x_i, $i = 1, 2, \ldots, n$, all are in the basis B, and hence their values appear in the Stetter vector.

4. Employing iterative solution methods for large eigenvalue problems

Modern methods for the solution of large eigenvalue problems are the iterative methods of Arnoldi or Jacobi–Davidson. Such methods have the attractive feature that they do not operate on the matrix A_r^{T} directly. Instead they iteratively perform the action of the linear operator at hand, for which it suffices to implement a computer routine that computes this action for any given vector. The nD system approach supports this and it offers a framework for computing the action of A_r^{T} on a vector v, by initializing the initial state as $w_{0,\ldots,0} := v$ and using the n recursions (7) in combination with the relationship (13). Such an approach 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 $r_{\alpha_1,\ldots,\alpha_n} w_{\alpha_1,\ldots,\alpha_n}$. This makes clear that for *any* choice of polynomial r the vector $r(\sigma_1, \ldots, \sigma_n) w_{0,\ldots,0}$ can be constructed from the *same* multidimensional time series $y_{t_1,\ldots,t_n}$ which is completely determined by (and computable from) the n difference equations (7) and the initial state $w_{0,\ldots,0}$.

There are two possible ways to retrieve the minimal value and the global minimizer(s) of the polynomial $p_\lambda(x_1, \ldots, x_n)$.

(i) If attention is focused on the computation of all the *stationary points* of the criterion p_λ first, then the actions of the matrices $A_{x_i}^{\mathrm{T}}$, for all $i = 1, \ldots, n$, play a central role since their eigenvalues constitute the coordinates of these stationary points. Given an initial state $w_{0,\ldots,0}$, which is composed of m^n values of the time series $y_{t_1,\ldots,t_n}$ (those for which $t_j < m$ for all j), each of these actions requires the computation of only m^{n-1} additional values of the time series $y_{t_1,\ldots,t_n}$ (namely those for which $t_i = m$ and $t_j < m$ if $j \neq i$). However, most of these additional values cannot be obtained directly (by application of a single recursion from the set (7)) from

the initial state $w_{0,\ldots,0}$. They require the use of more than one recursion from the set (7) and they involve the computation of values of $y_{t_1,\ldots,t_n}$ at certain other multidimensional time instants as well.

(ii) If attention is focused on the computation of the *critical values* of the criterion p_λ first, then the polynomial r may be chosen as p_λ which requires the computation of values of $y_{t_1,\ldots,t_n}$ at multidimensional time instants which are somewhat further away from the origin $(0,\ldots,0)$ than in the previous approach. For the computation of the action of the matrix $A_{p_\lambda}^{\mathrm{T}}$ the set of multidimensional time instants $(t_1,\ldots,t_n)$ at which the value of $y_{t_1,\ldots,t_n}$ needs to be determined is clearly larger than for each of the matrices $A_{x_i}^{\mathrm{T}}$. But for $A_{p_\lambda}^{\mathrm{T}}$ only the smallest real eigenvalue is required that corresponds to a real stationary point, whereas the eigenvalues of the matrices $A_{x_i}^{\mathrm{T}}$ correspond to the ith coordinate of the stationary points. Arnoldi and Jacobi–Davidson methods allow the user to 'zoom in' on a few eigenvalues, whereas in the case of $A_{x_i}^{\mathrm{T}}$ all the real eigenvalues need to be computed, for each $i = 1, 2, \ldots, n$, and all resulting real critical points have to be substituted into the criterion function to find the global optimum.

In this paper, approach (ii) of computing only the smallest real eigenvalue of the matrix $A_{p_\lambda}^{\mathrm{T}}$ will be investigated.

The following research questions are addressed in the remainder of this paper: (i) For a given multidimensional time instant $(t_1,\ldots,t_n)$, what is the most efficient way 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 (7)? And as a closely related question, what is the most efficient way to compute the whole state vector $w_{t_1,\ldots,t_n}$? (ii) Can we design a suboptimal heuristic procedure that computes the state vector $w_{t_1,\ldots,t_n}$ at acceptable computational costs? (iii) Is the polynomial optimization method using the nD system feasible and what is its performance compared to those of other methods?

5. A linear complexity result for computing $y_{t_1,\ldots,t_n}$

The first research question raised above is especially important when the efficiency of just one iteration of the iterative eigenvalue solver is under consideration. In this single iteration the state vector $w_{t_1,\ldots,t_n}$ has to be computed in an efficient way. This is in contrast to the case where we study the efficiency of solving the eigenvalue problem as a whole.

The following result addresses the computational complexity that can be achieved by an optimal algorithm to compute $y_{t_1,\ldots,t_n}$ from $w_{0,\ldots,0}$. For each multidimensional time instant $t = (t_1,\ldots,t_n)$ let the 'total time' be denoted by $|t| := t_1 + \cdots + t_n$.

Theorem 3. *Consider a set of n multidimensional recursions of the form* (7) *and 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 only the recursions* (7)*, has a computational complexity which increases at least linearly with the total time* $|t|$.

Proof. Each recursion from the set (7) 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 (7). 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 $w_{t_1,\ldots,t_n} = (A_{x_1}^{\mathrm{T}})^{t_1}(A_{x_2}^{\mathrm{T}})^{t_2}\cdots(A_{x_n}^{\mathrm{T}})^{t_n} w_{0,0,\ldots,0}$, it can be computed by the joint action of $t_1 + \cdots + t_n = |t|$ matrices of the form $A_{x_i}^{\mathrm{T}}$. It is not difficult to compute a fixed uniform upper bound on the computational complexity involved in the action of each of the matrices $A_{x_i}^{\mathrm{T}}$, 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 $w_{t_1,\ldots,t_n}$.

6. Formulation as a shortest path problem

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 (7) can be cast into the form of a *shortest path problem* (SPP). As it turns out, this SPP will quickly become huge and difficult to solve. However, it is possible to set up a *relaxation of the shortest path problem* (RSPP) which is considerably smaller and easier to solve. In the 2D case (with full recursions and uniform costs) the RSPP can be solved analytically and its solution happens to solve the SPP too. A central role in this approach is played by the notion of stable patterns, which are shifted along the 2D grid. The same approach leads to partial results in the 3D case (and in higher dimensions). It also underlies the design of the *heuristic methods* discussed in Section 7.

In general, a standard formulation of a shortest path problem requires the specification of a weighted directed graph $G = (V, E, W, v_I, v_T)$, consisting of a set V of nodes, a set $E \subseteq V \times V$ of edges, a weight function $W : E \to \mathbb{R}$, a set of initial nodes $v_I \in V$ and a set of terminal nodes $v_T \in V$. To compute a shortest path from v_I to v_T with smallest total weight, one may apply any classical algorithm (e.g., those of Dijkstra or Floyd). The set V should correspond to the various 'states' in which the computational procedure can be. It is natural to relate a node $v \in V$ in some way to a set of multidimensional time instants $(t_1, \ldots, t_n)$ for which the value of $y_{t_1,\ldots,t_n}$ is either already available or still requires computation. The edges E represent 'state transitions' and they are naturally associated with the recursions in the set (7). The weight function W is used to reflect the computational costs (e.g., the number of flops) associated with these recursions.

In setting up a shortest path problem formulation, one may run into the problem that the number of elements in V becomes infinite, since for $n \geq 2$ it may already happen that one can apply an infinite sequence of recursions without ever arriving at the specified multidimensional time instant $(t_1, \ldots, t_n)$. To avoid this, one may work backwards from $(t_1, \ldots, t_n)$, by figuring out sets of time instants with smaller total time which may assist in the computation of $y_{t_1,\ldots,t_n}$. Another feature of the problem is that many computations can be carried out in parallel since their exact order does not matter, so that many alternatives exist with equal performance. Already for small values of n, m and $|t|$ this means that the graph G can become very large. Two helpful observations for constructing a useful shortest path formulation are: (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. Therefore, a node $v \in V$ can naturally be associated with a *set* of time instants all having the same total time, rather than with individual time instants. This is formalized in the following definition.

Definition 4. 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 (i.e., the set of all its subsets). Let $V_\star$ be the set of time instants corresponding to $w_{0,\ldots,0}$ (i.e., for which $\max\{t_1, \ldots, t_n\} < m$).

Given a specified time instant $t = (t_1, \ldots, t_n)$, define V as the cartesian product $V_1 \times V_2 \times \cdots \times V_{|t|}$. Define the set of initial nodes as $v_I = \{(\phi, \ldots, \phi)\}$ (where ϕ denotes the empty set) and the set of terminal nodes v_T to consist of those tuples $v = (v1, \ldots, v|t|)$ for which $v_{|t|} = \{t\}$.

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 $v_k = v_{k+1} = \cdots = v_{|t|} = \phi$ and the set $\tilde{v}_k$ consists entirely of time instants $(\bar{t}_1, \ldots, \bar{t}_n)$ at which $y_{\bar{t}_1,\ldots,\bar{t}_n}$ can be computed from the values at the time instants contained in the union of sets $V_\star \cup v_1 \cup v_2 \cup \cdots \cup v_{k-1}$ through the application of a single recursion from the set (7).

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 computations for a time instant in the set $\tilde{v}_k$, which constitutes the difference between v and $\tilde{v}$ through the application of a recursion from the set (7), require a certain number of flops as determined by the number of terms involved in that recursion. The computational costs of a transition from v to $\tilde{v}$ are defined as the sum of all the (minimal) costs for the elements of that set $\tilde{v}_k$.

The weighted directed graph G is defined as the tuple $G = (V, E, W, v_I, v_T)$. The shortest path problem (SPP) associated with G models the optimal computation of $y_{t_1,\ldots,t_n}$ from $w_{0,\ldots,0}$ using the recursions (7).

Note that the graph G for the SPP has a tree structure rather than a network structure, which makes it possible to apply branch and bound techniques for tree searching. However, because $V = V_1 \times V_2 \times \cdots \times V_{|t|}$, the size of the SPP quickly becomes very large.

An interesting *relaxation of the shortest path problem* (RSPP) is obtained when condition (ii) in the definition of E is replaced by the less restrictive condition that the set $\tilde{v}_k$ consists entirely of time instants $(t_1, \ldots, t_n)$ at which $y_{t_1,\ldots,t_n}$ can be computed from the values at the time instants contained in the union of sets $V_\star \cup T_1 \cup \cdots \cup T_{k-2} \cup v_{k-1}$ through the application of a single recursion from the set (7). In this case, each node $v = (v_1, \ldots, v_{|t|}) \in V$ can be restricted to the set $v_k \in V_k$ for which k is as large as possible with v_k non-empty. Then V can be redefined as $V = V_1 \cup \cdots \cup V_{|t|-1} \cup V_{|t|}$, the set of initial nodes can be replaced by a single initial node $v_I = \phi$ and the set of terminal nodes by a single terminal node $v_T = \{t\}$.

The number of nodes in V is considerably smaller for the RSPP than for the SPP, and it becomes much easier to compute a solution. The optimal value of the RSPP provides a lower bound for the optimal value of the SPP. In the case where the recursions in (7) are not sparse, this lower bound is likely to be close to the optimal value of the SPP.

For the RSPP, each node in V is a collection of time instants sharing the same total time, say k. Such a node is a subset of T_k, and the sets T_k correspond to parallel hyperplanes in $\mathbb{N}_0^n$. Therefore, it is natural to compare the geometric patterns exhibited by nodes, by means of *translations* in $\mathbb{N}_0^n$. Such a translation τ_s involving a translation vector $s = (s_1, \ldots, s_n) \in \mathbb{Z}^n$ acts on a node N by acting element-wise on each of the time instants $(t_1, \ldots, t_n)$ contained in N according to the rule $(t_1, \ldots, t_n) \mapsto (t_1 + s_1, \ldots, t_n + s_n)$. To describe the structure of an optimal solution to the RSPP, the concept of a *stable pattern* is useful.

Definition 5. In the graph $G = (V, E, W, v_I, v_T)$ corresponding to the RSPP, a node $N_b \in V$ is said to exhibit a *stable pattern* if there exists a translation τ_s with the property that $N_a = \tau_s(N_b)$ is a node in V for which (N_a, N_b) is an edge in E.

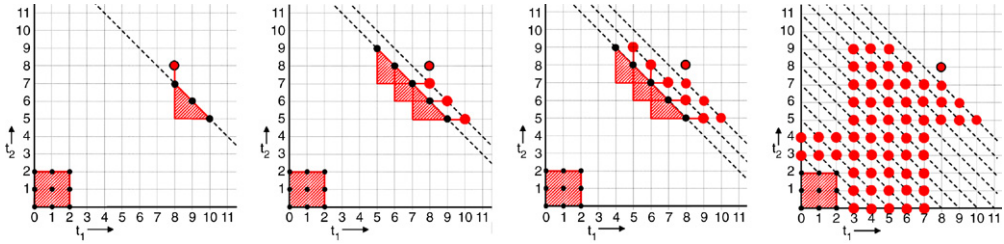

Fig. 1. Constructing and translating a minimal stable pattern for $n = 2$, $m = 3$.

Clearly, translation vectors s associated with stable patterns have the property that $|s| = -1$. Note that a stable pattern may be associated with several different translation vectors. A situation of special interest occurs for translation vectors s of the form $s = (0, \ldots, 0, -1, 0, \ldots, 0)$ as they correspond to (backward) shifts along one of the n different time axes.

When a node $N \in V_k$ exhibits a stable pattern for some associated translation τ_s, then repeated application of τ_s produces a sequence of nodes in V with decreasing total times $k, k-1, k-2, \ldots$, until the boundary of the non-negative orthant $\mathbb{N}_0^n$ is reached. Now the idea is to construct an optimal path which solves the RSPP in a backwards fashion as a composition of three parts: (i) a part which connects the terminal node $v_T = \{t\}$ to a node exhibiting a stable pattern, (ii) a part in which the stable pattern is repeatedly translated using an associated translation τ_s with $|s| = -1$, (iii) a part which connects it to the initial node $v_I = \phi$.

6.1. The 2D case

In the 2D case it is possible to solve the RSPP analytically, for the situation of 'full recursions' (i.e., the polynomials $f^{(i)}(x_1, x_2)$, $i = 1, 2$, involve *all* the possible terms of total degree $\leq m-1$ with non-zero coefficients) when 'uniform costs' are applied (i.e., the costs associated with the application of a recursion to compute a value y_{t_1,t_2} are always the same, for each recursion).

Of course, if a stable pattern is to assist in the construction of a shortest path, the costs associated with its translation need to be as small as possible. For the case of full recursions and uniform costs, this implies that the size of a stable pattern needs to be as small as possible (i.e., the number of time instants involved). A stable pattern is called a *minimal stable pattern* if it does not contain a strict subset which is also a stable pattern.

In the 2D case, the nodes are subsets of the diagonals for which $t_1 + t_2$ is constant. As an example, for $m = 3$ a minimal stable pattern consists of a subset of five consecutive points on a diagonal. This is depicted in Fig. 1, together with translations on several subdiagonals. For an arbitrary value of m we have the following result.

Proposition 6. *In the 2D case with full recursions and uniform costs, a minimal stable pattern consists of* $2m - 1$ *consecutive points on a diagonal of constant total time. This minimal stable pattern can be associated both with a shift in the direction of the* t_1*-axis and with a shift in the direction of the* t_2*-axis.*

Proof. To reach a point on a diagonal T_k (with total time k) requires one of the two full recursions. Each such full recursion involves m consecutive points on the first subdiagonal T_{k-1}. Therefore, any stable pattern involves at least a subset of m consecutive points. To compute such a subset, each of its m points also requires one of the two full recursions. Consider two consecutive points of this subset and suppose they are computed by employing two different

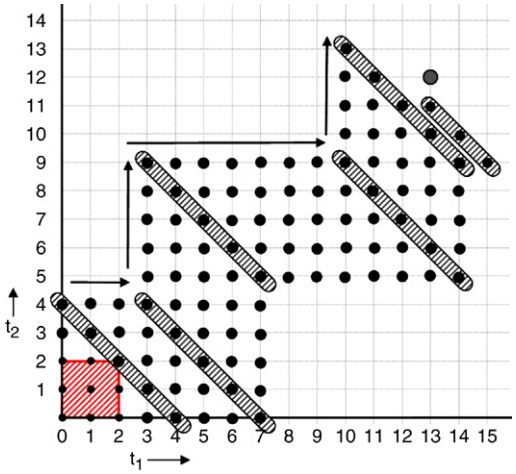

Fig. 2. A shortest path for computing $y_{13,12}$.

recursions, then this requires either $2m-1$ consecutive points on the next subdiagonal T_{k-2}, or it requires $2m$ points (consisting of two disconnected groups of m points) on T_{k-2}, which is less efficient. When all m points involve the same recursion this also requires $2m-1$ consecutive points on t_{k-2}. Hence any stable pattern involves at least a subset of $2m-1$ consecutive points.

Now, a subset of $2m-1$ consecutive points constitutes a stable pattern. To see this, one may require the $m-1$ points with largest t_1-values to be computed with the recursion involving $f^{(1)}$, and the $m-1$ points with largest t_2-values with the recursion involving $f^{(2)}$. The point in the middle (the mth point of the stable pattern) may be computed either with the recursion involving $f^{(1)}$ or with the recursion involving $f^{(2)}$. In either case, this involves only a subset of $2m-1$ consecutive points on the next subdiagonal. Hence we are dealing with a minimal stable pattern.

If the middle point of the stable pattern is computed with the recursion involving $f^{(1)}$, the translation involved is $s=(-1,0)$, which constitutes a (backward) shift in the t_1-direction. When the recursion involving $f^{(2)}$ is employed, the translation involves $s=(0,-1)$ which constitutes a (backward) shift in the t_2-direction. Hence the minimal stable pattern can be associated with shifts in both directions of the time axes. □

The proof above makes clear that the minimal stable pattern of $2m-1$ consecutive points can be used to construct the second part of an optimal solution path for the RSPP in this 2D case, using a sequence of translations involving the vectors $(-1,0)$ and $(0,-1)$. It also is indicated how the first part of such a solution can be constructed which connects the terminal node to such a stable pattern. For the third part, note that the diagonal with total time $2m-2$ consists of precisely $2m-1$ consecutive points in the non-negative orthant. Its middle point, with coordinates $(m-1, m-1)$, corresponds to the right upper corner of the set of time instants associated with $w_{0,0}$ and is therefore contained in $V_\star$. To compute y_{t_1,t_2} at each of the other $2m-2$ points requires the recursions $f^{(1)}$ and $f^{(2)}$ in an obvious way: for each points only one of the two recursions is feasible depending on which time coordinate is $\geq m$. It should be clear that all points with total time $\leq 2m-2$ not in $V_\star$ require computation. This is summarized in the following result, which is illustrated in Fig. 2.

Proposition 7. *In the 2D case with full recursions and uniform costs, a solution to the RSPP has the following structure.* (i) *A part starting at* $v_I=\phi$ *followed by the nodes* $v_m=T_m$, $v_{m+1}=$

$T_{m+1}, \ldots, v_{2m-2} = T_{2m-2}$. (ii) *A part involving the nodes* $v_{2m-1}, v_{2m}, \ldots, v_{|t|-2}$ *with the property that each of the nodes* v_k *in this part involves a minimal stable pattern of* $2m-1$ *consecutive points on the diagonal* T_k *and that each two consecutive nodes are related by a translation* τ_s *with* $s = (-1, 0)$ *or* $s = (0, -1)$. (iii) *A part involving the nodes* $v_{|t|-1}$ *and* $v_T = \{t\}$. *The node* $v_{|t|-1}$ *involves m consecutive points on the diagonal* $T_{|t|-1}$ *such that there is an edge from* $v_{|t|-2}$ *to* $v_{|t|-1}$ *and an edge from* $v_{|t|-1}$ *to* v_T.

Note that the above discussion is valid only when the terminal node $v_T = \{t\}$ satisfies $|t| \geq 2m$. For a terminal node with total time $\leq 2m-1$, no stable pattern occurs in the optimal path.

Interestingly, any solution to the RSPP in the 2D case with full recursions and uniform costs is also a solution to the SPP itself. To see this, note that the RSPP has been obtained by considering only the time instants (t_1, t_2) with $t_1 + t_2 = k - 1$ and disregarding those with $t_1 + t_2 < k - 1$ when the computation of a value y_{t_1,t_2} at a time instant with $t_1 + t_2 = k$ is investigated. However, the relevant time instants with $t_1 + t_2 < k - 1$ are all in the triangular area of points that are both below and to the left of the $2m-1$ points in the stable pattern on the diagonal with total time $k-1$. Hence, when the stable pattern is shifted along the time axes, this area is eventually entirely covered. See again Figs. 1 and 2.

Corollary 8. *In the 2D case with full recursions and uniform costs, a solution to the RSPP is also a solution to the SPP.*

6.2. The 3D case

In the 3D case the situation is more complicated, also when restricting to the case of full recursions and uniform costs. Depending on the value of m, there may be several minimal stable patterns of different geometrical shape. Also, not every such stable pattern is associated with all of the three shifts along each of the time axes. On the other hand, it is possible to generalize the constructions in the 2D case in a straightforward way to construct stable patterns which do have a symmetric geometrical shape and which are associated with all of the three shifts, although they are not minimal. Here we present partial results for the cases $m = 2$ and $m = 3$. These results also provide a rationale for the heuristic computational procedures developed and discussed in Section 7.

In the 3D case with $m = 2$, with full recursions and uniform costs, the initial state $w_{0,0,0}$ corresponds to values of y_{t_1,t_2,t_3} at the time instants in $V_\star$, for which $t_1, t_2, t_3 \in \{0, 1\}$. Starting from the initial state, the three full recursions allow for the computation of y_{t_1,t_2,t_3} at the time instants $(2, 0, 0)$, $(0, 2, 0)$ and $(0, 0, 2)$, respectively, for which values are required at the time instants $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ and $(0, 0, 0)$. This is depicted in Fig. 3(a). In Fig. 3(b) a stable pattern is shown which allows for shifts in the directions of all the three time axes. This is a special instance of the following more general result for the nD case.

Proposition 9. *In the nD case a stable pattern is exhibited by translations of the set of time instants* $N = \{(t_1, \ldots, t_n) \mid t_1 + \cdots + t_n = n(m-1); t_1 \geq 0, \ldots, t_n \geq 0\}$. *This stable pattern allows for shifts along each of the n time axes.*

Proof. From the values of $y_{t_1,\ldots,t_n}$ at the time instants contained in $V_\star$, all its values in the non-negative orthant can be computed. This includes all the values at time instants for which the total time is $\leq n(m-1)$. Note that $(m-1, \ldots, m-1)$ is the time instant with the

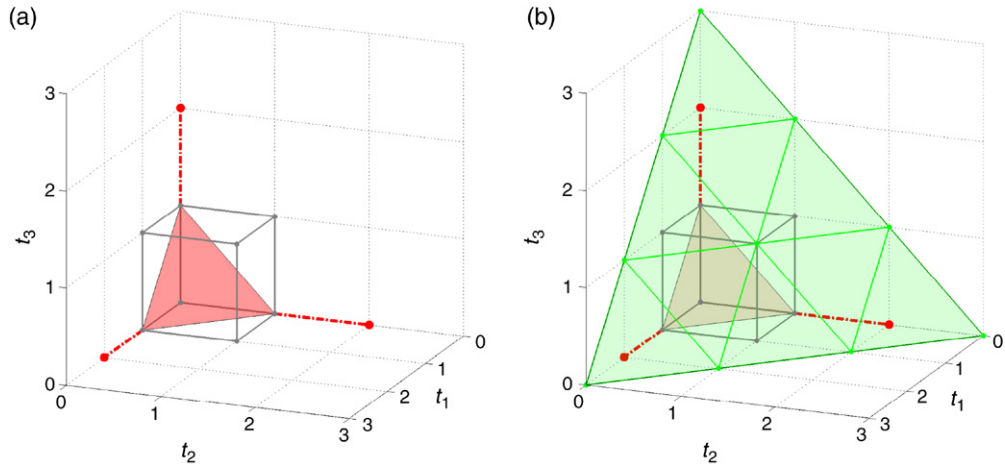

Fig. 3. (a) The initial state $w_{0,0,0}$ and the full recursions for computing $y_{2,0,0}$, $y_{0,2,0}$, $y_{0,0,2}$. (b) A stable pattern (green) for shifts in all three directions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

largest total time contained in $V_\star$. Now, the set N describes all the time instants in the non-negative orthant with total time equal to $n(m-1)$. Together with all the time instants with a smaller total time, they allow for the (immediate) computation of $y_{t_1,\ldots,t_n}$ at all the time instants with total time $n(m-1)+1$. This includes all the translations of N with translation vectors $s=(1,0,\ldots,0)$, $s=(0,1,\ldots,0)$, $\ldots$, $s=(0,\ldots,0,1)$. This procedure can then be repeated, yielding points with total time $n(m-1)+2$, etc., showing the pattern of points in N to yield a stable pattern. □

However, for $n>2$ the stable pattern N in the above proposition is not *minimal*. To investigate minimality of stable patterns it is helpful to study the relationship between a point of total time k and nearby points of total time $k-1$. (Alternatively, one may construct minimal stable patterns by deleting points from a stable pattern of the form N until stability is lost.)

In Fig. 4(a) it is shown for the case $n=3$ and $m=2$ how the points with total time 4 (blue layer) are connected along the directions of the time axes (red lines) to nearby points of total time 3 (green layer). In Fig. 4(b) the points in these two consecutive layers are arranged in convenient triangulated patterns and shown from a different viewpoint.

In Fig. 5 it is visualized how the three different full recursions associated with $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ may assist in the computation of a point with total time 4, and also which points of total time 3 (red triangle) are required to achieve this.

For $n=3$ and $m=2$, the stable pattern N in the previous proposition has ten points. When two of its corner points are deleted, a *minimal* stable pattern of eight points remains, which however allows only for a shift in just *one* of the directions of the time axes. But when only one corner point is deleted, a stable pattern results which allows for shifts in *all three* directions of the time axes.

As a consequence, a solution to the RSPP needs to take into account the coordinate values of the time instant $t=(t_1,t_2,t_3)$ associated with the terminal node v_T. First, a stable pattern containing nine points can be used for shifting along the two axes corresponding to the two smallest values of t_1, t_2 and t_3. Then a cheaper minimal stable pattern containing eight points can be used for shifting along the remaining third axis (this should be a subpattern of the previous

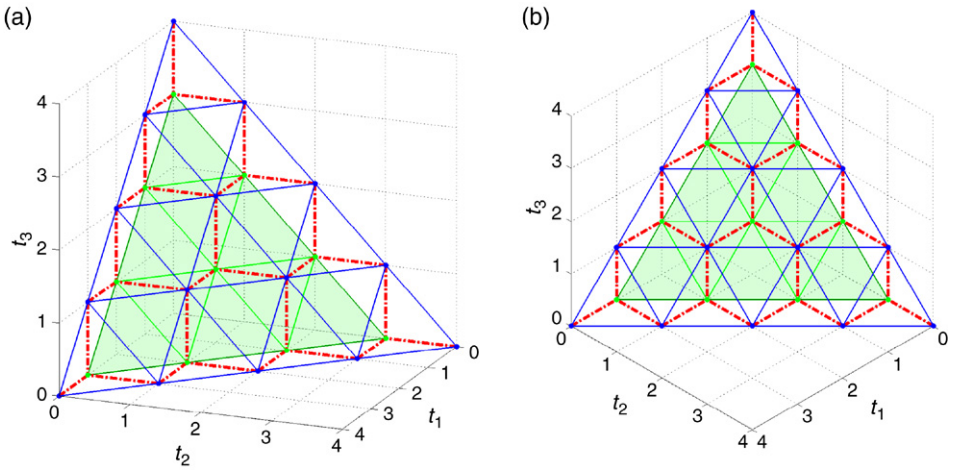

Fig. 4. (a) Connections of the points with total time 4 to nearby points of total time 3. (b) A different viewpoint. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

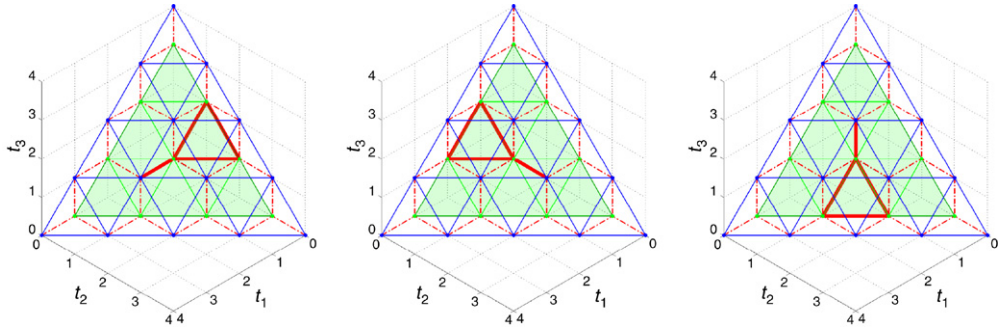

Fig. 5. Computation of points with total time 4 using the three different recursions (associated with $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$) involving points with total time 3. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

pattern of nine points). Note that such a solution also constitutes a solution to the SPP itself, because these stable patterns have convex configurations.

For the case $n = 3$ and $m = 3$, the non-minimal stable pattern N is depicted in Fig. 6(a). A smaller stable pattern which allow for shifts in just one of the directions of the time axes is depicted in Fig. 6(b) (green layer). When three points in one of the corners of N are deleted, a stable pattern remains which still allows for shifts in all three directions of the time axes. This demonstrates the complexity of the shortest path problem and its relaxation.

7. Some heuristic procedures for the shortest path problem

From the discussion in the previous section it is clear that the size and the complexity of the SPP increases rapidly with n. However, optimal solution of the SPP is not a goal by itself; it serves to facilitate the efficient computation of state vectors $w_{t_1,\ldots,t_n}$. Note that in our application we need to solve the SPP only once, and that this solution can then be used in every iteration of

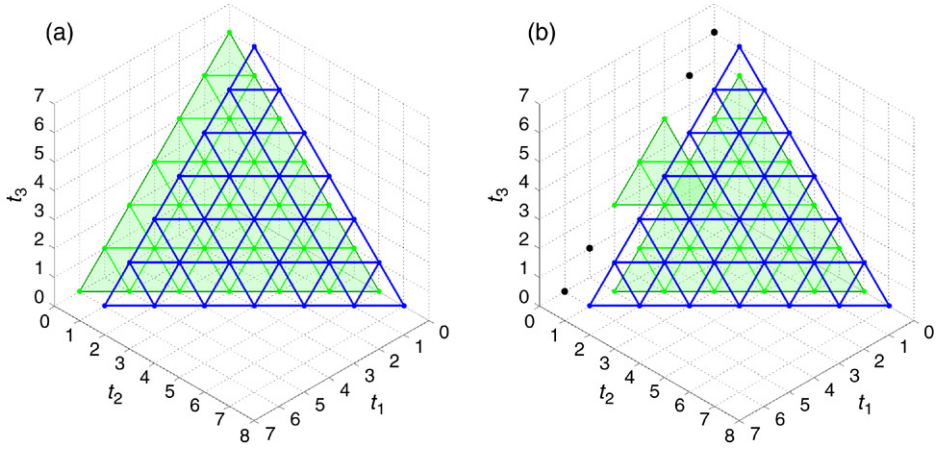

Fig. 6. (a) A non-minimal stable pattern for $n = 3$ and $m = 3$ allowing for shifts in three time directions. (b) A non-minimal stable pattern allowing for a shift in just one time direction. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the iterative eigenvalue solvers. For the overall efficiency of the approach the number of iterations determines whether it is worthwhile to set up and solve the SPP, or if it is advisable to avoid this and to employ cheaper but suboptimal heuristic methods instead.

Using the insights from the previous section, five heuristic methods were developed to calculate the value $y_{t_1,\ldots,t_n}$ that appears in a state vector $w_{t_1,t_2,\ldots,t_n}$. The performance of these methods was compared for a number of situations involving full recursions and uniform costs.

The first four heuristic methods under investigation are the following.

(i) The *linear method* starts by computing the values of $y_{t_1,\ldots,t_n}$ at all the points on and below the hyperplane which corresponds to the total time $n(m-1)$, thereby just covering the area (hypercube) of initial values. Then this stable pattern of values N (see Proposition 9) is shifted step by step, with each step involving a single shift in one of the axis directions, until the requested location is reached.

(ii) The *diagonal method* proceeds by first computing $y_{t_1,\ldots,t_n}$ for all the time instants with constant total time $|t| = 1$, then all those with total time $|t| = 2$ and so on, until the requested location is reached.

(iii) The *equalizing method* proceeds by computing a value $y_{t_1,\ldots,t_n}$ by employing that recursion of the system (7) which reduces the *largest* possible coordinate of the time instant $y_{t_1,\ldots,t_n}$. (It uses a lexicographic ordering in the case there is more than one largest coordinate.) The path between the initial state $w_{0,\ldots,0}$ and the requested location is determined in a backwards fashion, i.e., starting from the requested location.

(iv) The *axis method* uses the same methodology as the equalizing method with the only difference that it applies that recursion of the system (7) which reduces the *smallest* possible coordinate of the time instant $y_{t_1,\ldots,t_n}$.

For $n = 2$ and $n = 3$ the paths to a few arbitrarily chosen points have been computed using the four described methods. The results of some of these points are collected in Tables 1 and 2. For each method the total number of all stored points required for computing the requested point (i.e., the size of the path), the total number of flops and the actual running time (in ms obtained on a PC platform with an Intel Pentium PIV 2.8 GHz processor and 512 MB internal memory) are given for each requested point.

Table 1
Two-dimensional case (2×2 initial state $w_{0,0}$)

Coordinates of requested point	Method	Linear	Diagonal	Equalizing	Axis
(0, 10)	Stored points	36	91	46	34
	Flops	100	265	167	118
	Cpu time (ms)	7	6	2	2
(0, 100)	Stored points	306	5 356	1921	304
	Flops	910	16 060	7697	1 468
	Cpu time (ms)	16	40	55	20
(10, 10)	Stored points	66	276	64	143
	Flops	209	820	239	598
	Cpu time (ms)	30	20	0	20
(100, 100)	Stored points	606	20 706	604	5 948
	Flops	2009	62 110	2399	27 758
	Cpu time (ms)	20	120	50	220

Table 2
Three-dimensional case ($2 \times 2 \times 2$ initial state $w_{0,0,0}$)

Coordinates of requested point	Method	Linear	Diagonal	Equalizing	Axis
(0, 0, 10)	Stored points	120	560	159	88
	Flops	344	2 216	832	399
	Cpu time (ms)	12	18	4	1
(0, 0, 100)	Stored points	1 020	192 920	29 808	808
	Flops	3 044	771 656	214 696	4 449
	Cpu time (ms)	32	1 017	1 875	33
(10, 10, 10)	Stored points	320	7140	248	1 868
	Flops	1 040	28 536	1 379	10 565
	Cpu time (ms)	20	40	0	20
(100, 100, 100)	Stored points	3 020	4 728 720	2 408	281 948
	Flops	10 040	18 914 856	14 204	1 892 839
	Cpu time (ms)	50	265 008	150	781 646

From this experiment we may conclude that the linear method does indeed exhibit a linear complexity with respect to $|t|$ (see Tables 1 and 2 for some results). But in higher dimensions (e.g., $n > 10$) the linear method may become inefficient because the simplex entirely covering the hypercube of initial values becomes very large. Although it constitutes a stable pattern associated with shifts in all n time directions, it is non-minimal and especially for points near the initial hypercube more efficient paths can be constructed. The equalizing method performs best for points having (almost) equal coordinates (for example the point (10, 10)). For $n = 2$ this method generates stable patterns which constitute an optimal solution to the shortest path problem for y_{t_1,t_2} with $t_1 = t_2$. For other points it is less efficient. The axis method performs well for points near the coordinate axes (for example the point (0, 10)). For $n = 2$ this method generates an optimal solution to the shortest path problem for y_{t_1,t_2} with $t_1 = 0$ or $t_2 = 0$. The diagonal method does not exhibit a linear numerical complexity with respect to $|t|$ and is highly inefficient.

To further support and visualize these statements, some simulation experiments have been performed with $n = 2$ and $m = 3$, where the requested state vectors are $w_{0,500}$, $w_{250,250}$ and $w_{125,375}$.

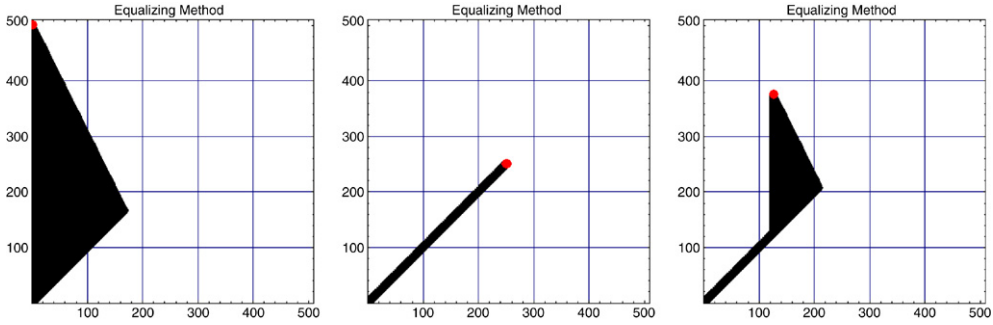

Fig. 7. Points required by the equalizing method for computing state vectors at time instants (0, 500), (250, 250) and (125, 375) with a 3×3 initial state.

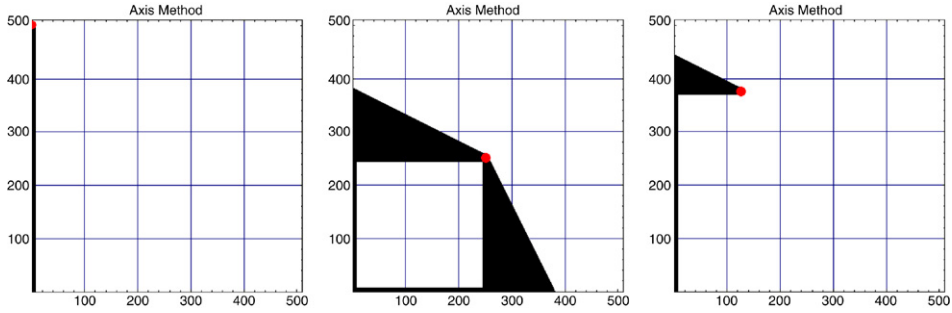

Fig. 8. Points required by the axis method for computing state vectors at time instants (0, 500), (250, 250) and (125, 375) with a 3×3 initial state.

For the equalizing method, the points which are required for computing the state vectors $w_{0,500}$, $w_{250,250}$ and $w_{125,375}$, respectively, are displayed in three separate plots in Fig. 7. Likewise, the above claims for the axis method are visualized in Fig. 8.

The axis and equalizing methods are only efficient in some specific situations as shown in the Figs. 7 and 8. Performance worsens if one wants to compute a whole set of state vectors involving several different time instants at once, such as is required, for example, for the operator A_r^{T} when r involves several terms. In Fig. 9 the points are displayed which are computed by the linear, equalizing and axis methods to facilitate the computation of five requested state vectors at the time instants (0, 500), (125, 375), (250, 250), (375, 125) and (500, 0) simultaneously. The linear method clearly is the most efficient. The numbers of points evaluated by these three methods are 7460, 113 988 and 54 597, respectively.

Because the diagonal, equalizing and axis methods are not efficient in every situation and because the linear method is only efficient for small values of n, but becomes less efficient when the dimension of the problem increases, a fifth method was implemented. It is a combination of the equalizing and axis method and it applies that recurrence relation of the system (7) which requires a minimal number of new points to be calculated. It also proceeds in a backwards fashion but when constructing a path it takes the points into account that have already been included in the path. This method becomes more efficient than the linear method for larger values of n. Fig. 10 shows a plot of the evaluated points needed for the computation of the state vectors at the time instants (0, 500), (125, 375), (250, 250), (375, 125) and (500, 0) using this fifth method.

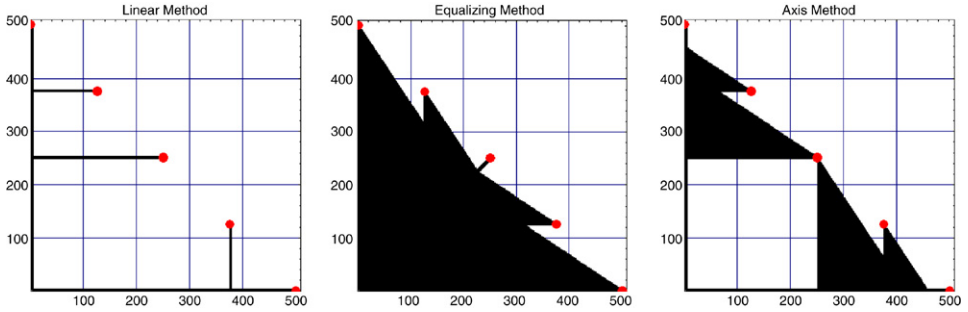

Fig. 9. Points required by the linear, equalizing and the axis methods for computing state vectors at time instants (0, 500), (125, 375), (250, 250), (375, 125) and (500, 0) with a 3×3 initial state.

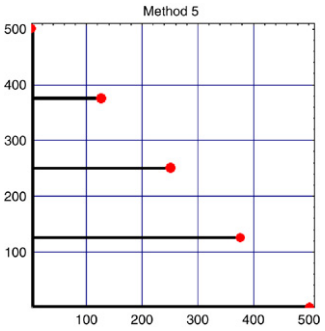

Fig. 10. Points required by the fifth method with a 3×3 initial state.

Obviously, more sophisticated methods can also be designed which take the global structure of the requested points into account.

8. Comparison of computational methods for a worked example

In this section we discuss the third research question stated in the end of Section 4: the feasibility of the method discussed in this paper is studied and, to get insight, its performance is compared to the performance of other methods.

The global minimum and its location is computed for a (Minkowski-norm) dominated polynomial p_λ with $n = 4$, $d = 4$, $\lambda = 1$ and $q(x_1, x_2, x_3, x_4) = x_1x_2x_3^2x_4^2 + 3x_1x_2 + x_2x_3 + x_3x_4^2 + 2x_3x_4 + x_4 + 8$:

$$\begin{aligned} p_1(x_1, x_2, x_3, x_4) \; &= (x_1^8 + x_2^8 + x_3^8 + x_4^8) \\ &\quad + x_1x_2x_3^2x_4^2 + 3x_1x_2 + x_2x_3 + x_3x_4^2 + 2x_3x_4 + x_4 + 8. \end{aligned} \tag{14}$$

The first-order conditions of problem (14) are given by

$$\begin{cases} d^{(1)}(x_1, x_2, x_3, x_4) = x_1^7 + \frac{1}{8}x_2x_3^2x_4^2 + \frac{3}{8}x_2 = 0 \\ d^{(2)}(x_1, x_2, x_3, x_4) = x_2^7 + \frac{1}{8}x_1x_3^2x_4^2 + \frac{3}{8}x_1 + \frac{1}{8}x_3 = 0 \\ d^{(3)}(x_1, x_2, x_3, x_4) = x_3^7 + \frac{1}{4}x_1x_2x_3x_4^2 + \frac{1}{8}x_4^2 + \frac{1}{8}x_2 + \frac{1}{4}x_4 = 0 \\ d^{(4)}(x_1, x_2, x_3, x_4) = x_4^7 + \frac{1}{4}x_1x_2x_3^2x_4 + \frac{1}{4}x_3x_4 + \frac{1}{4}x_3 + \frac{1}{8} = 0. \end{cases} \tag{15}$$

Table 3
Minimal real eigenvalue of the explicit matrix A_{p_1}

Application	Method	Eigenvalue	Time (s)
Mathematica	Solution of system (15)	4.09516474435915	744
Mathematica	Construction of A_{p_1} and computation of all eigenvalues	4.09516474435915	465 + 315
Matlab	Construction of A_{p_1} and computation of all eigenvalues	4.09516474435924	465 + 176

Before we discuss the computation of the global minimum of this polynomial using an nD system in combination with an iterative eigenvalue solver, the global minimum and its location are computed by using some more 'conventional' methods: (i) the system of first-order partial derivatives which make up the first-order conditions is solved; (ii) the matrix A_{p_1} is constructed explicitly and the global minimum is computed as the smallest real eigenvalue of all the eigenvalues of this matrix. For these purposes standard routines from the software packages Mathematica and Matlab have been employed. For all experiments throughout this section, use was made of Matlab 7.0.1 and Mathematica 5.2, running on an Intel Pentium PIV 2.8 GHz platform with 512 MB of internal memory.

The real solutions of system (15) constitute the stationary points of $p_1(x_1, x_2, x_3, x_4)$. This system is solved using Mathematica with the *NSolve* routine. By substituting each real solution into polynomial (14), we end up with 11 critical values: 8.003511027703947, 7.329726207721671, 6.742495497462370, 6.723499319606622, 6.045721670341264, 5.8666 17734968560, 5.731485856745822, 5.624408534031756, 5.491307287483400, 4.4825275670 57376, 4.095164744359150. The corresponding stationary points can be classified as a global minimizer, four local non-global minimizers, and six saddle points. The smallest of these critical values yields the global minimum.

In an attempt to obtain similar information in an independent fashion, the matrix A_{p_1} has been constructed explicitly using exact computation. The involved quotient space $\mathbb{R}[x_1, x_2, x_3, x_4]/I$ is of dimension $(2d-1)^n = 2401$. Therefore the matrix A_{p_1} has 2401 distinct eigenvalues. This matrix is highly sparse: it contains only 43 178 non-zero elements. See Fig. 11 for a representation of the sparsity structure of the matrix A_{p_1}.

Building the matrix A_{p_1} took 465 s. All of its eigenvalues are computed numerically in two ways: using the eigenvalue solver *Eigenvalues* in Mathematica and the eigenvalue solver *Eig* in Matlab. In Table 3 the results of these computations are collected.

The outcomes of these computations agree up to a limited number of decimal digits. Therefore, to gain insight in the accuracy of the various methods, all the locations of the global minimum found with the above described 'conventional' methods have been used as the starting point for a local search method. The following coordinates for the global minimizer have been obtained using a thirty-digit working precision: $x_1 = +0.876539213106233894587289929758$, $x_2 = -0.903966282304642050057296045914$, $x_3 = +0.862027936174326572650513966373$, $x_4 = -0.835187476756286528192781820247$. The corresponding criterion value of this point is computed as 4.095164744359157279 770316678156. These values have been used as the true minimizer and global minimum of the polynomial $p_1(x_1, x_2, x_3, x_4)$ for the purpose of accuracy analysis of the numerical outcomes of the various computational approaches.

Following the approach of this paper, we then proceeded to determine the global minimum of the polynomial (14) using the nD system implementation of the linear operator A_{p_1} to compute

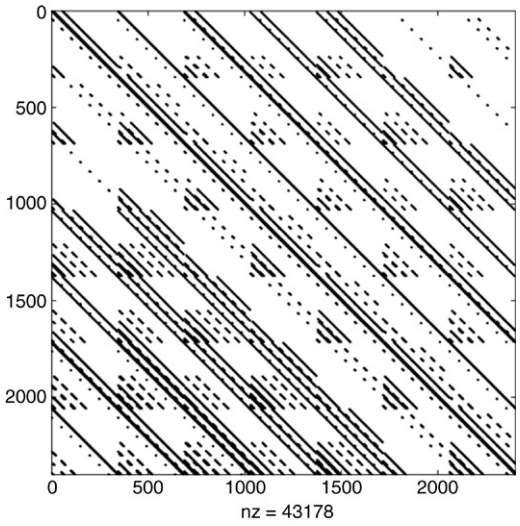

Fig. 11. Sparsity structure of the matrix A_{p_1}.

Table 4
Minimal eigenvalue of the operator A_{p_1} (1 eigenvalue calculated)

Method	Eigenvalue	Accuracy	# Operator actions	Time (s)
Eigs	4.09516474405595	3.03×10^{-10}	3040	173
Jdqr	4.09516474449668	1.38×10^{-10}	312	21
Jdqz	4.09516474427060	8.86×10^{-11}	302	21

only its smallest real eigenvalue with an iterative eigenvalue solver. The coordinates of the global minimum are computed from the corresponding eigenvector, employing the Stetter vector structure. For this purpose the iterative eigenvalue solvers *Jdqr*, *Jdqz* and *Eigs* have been used. Jdqr is a normal and Jdqz is a generalized iterative eigenvalue solver. Both methods employ Jacobi–Davidson methods coded in Matlab. The method Eigs is an iterative standard Matlab eigenproblem solver which uses (restarted) Arnoldi methods through ARPACK. In Table 4 the results of these computations are displayed. The columns denote the methods used, the minimal eigenvalue computed, the difference between this eigenvalue and the 'true' eigenvalue computed above, the number of required operator actions, and the running time. For these computations the default parameters of the various methods are used.

The global minimum computed by the Jdqz method produces the critical value that is the closest to the critical value computed by the local search method. For this setting, the nD systems approach uses the fewest operator actions. The coordinates of the stationary point corresponding to this global minimum are computed as: $x_1 = +0.876539213107485$, $x_2 = -0.903966282291641$, $x_3 = +0.862027936168838$, $x_4 = -0.835187476763094$.

For the problem of finding the global minimum of a polynomial there are several other specialized software packages available, which employ different approaches. To put the performance of the nD systems approach of this paper into perspective, we will briefly discuss the outcomes of the computation of the global minimum of polynomial (14) by the software packages SOSTOOLS, GloptiPoly and SYNAPS.

Table 5
Results of SOSTOOLS, GloptiPoly and SYNAPS

Method	x_1	x_2	x_3	x_4	Global minimum	Accuracy	Time (s)
SOSTOOLS	–	–	–	–	4.09516477401837	2.97×10^{-8}	10
GloptiPoly	0.876535	−0.903963	0.862021	−0.835180	4.09516476247764	1.81×10^{-8}	11
SYNAPS	0.876536	−0.903965	0.862026	−0.835184	4.09516474461324	2.50×10^{-10}	2

SOSTOOLS is a Matlab toolbox for formulating and solving sum of squares (SOS) problems (see Prajna et al. (2004) and Parrilo (2003)). It uses the Matlab solver SeDuMi (see Sturm (1999)) to solve the involved semi-definite programs (SDP). To compute the global minimum, SOSTOOLS searches for the largest possible γ for which $p_1(x_1, x_2, x_3, x_4) - \gamma$ is still a sum of squares. This γ may be the global minimum $p_\star$ we are looking for, depending on whether the polynomial $p_1(x_1, x_2, x_3, x_4) - p_\star$ can be written as a sum of squares of polynomials (see Parrilo and Sturmfels (2003)).

GloptiPoly (see Henrion and Lasserre (2003)) solves a multivariable polynomial optimization problem by building and solving convex linear matrix inequality (LMI) relaxations of the problem using SeDuMi. It produces a series of lower bounds which converge to the global optimum we are looking for. The theory of moments and positive polynomials is used in the implementation of this software (see Lasserre (2001, 2002)).

SYNAPS (Reis et al., 2002) is a C++ environment for symbolic and numeric computations. It provides a routine for searching for the real solutions of a polynomial system of equations like system (15) within a given domain. All the solutions of this system can be substituted into the polynomial (14) to find the global minimum.

The results of the computations using the above mentioned software packages (using the default parameters) are collected in Table 5. Note that SOSTOOLS does not return the coordinates of the global minimum.

When comparing the results in Tables 4 and 5, we see that the methods SOSTOOLS, GloptiPoly and SYNAPS are faster than the methods using our nD system approach. Moreover, the method Eigs performs poorly: it uses very many operator actions and needs a lot of running time. But where the methods in Table 5 appear to be faster, they are not as accurate as the iterative eigenvalue solvers in Table 4. This may be due to the default tolerance settings in these software packages (a trade-off is involved between computation time and accuracy). Although the methods in Table 5 up to now outperform the approach described in this paper in running time, the nD system approach leaves us the possibility to increase its performance and accuracy by making use of the internal structure and the sparsity of the problem and parallelizing the computations involved. This is in order to make it possible to tackle larger problems in the future.

9. Conclusions and discussion

The approach used in this paper to compute the global minimum of a dominated polynomial can be extended in several ways. One immediate extension involves the dominating term $\lambda(x_1^{2d} + \cdots + x_n^{2d})$ with $\lambda > 0$, which may obviously be replaced by a dominating term of the form $\lambda_1 x_1^{2d} + \cdots + \lambda_n x_n^{2d}$ with $\lambda_i > 0$ for all $i \in \{1, 2, \ldots, n\}$. Depending on the monomials which are allowed to feature in the polynomial $q(x_1, \ldots, x_n)$ and depending on the chosen monomial ordering, one may also extend the approach by using a dominating term of the form

$\lambda_1 x_1^{2d_1} + \cdots + \lambda_n x_n^{2d_n}$ with $\lambda_i > 0$ and possibly different (but well-chosen) positive integers d_i. Also, one may consider generalizations which involve weighted total degree monomial orderings.

As we have argued in Section 2, the range of applicability of the presented method extends beyond that of dominated polynomials. In Hanzon and Jibetean (2003) it is shown how the infimum of an *arbitrary* multivariate real polynomial $q(x_1, \ldots, x_n)$ can be found as the limit for $\lambda \downarrow 0$ of the global minima of the dominated polynomials $q(x_1, \ldots, x_n) + \lambda \|(x_1, \ldots, x_n)\|_{2d}^{2d}$. There it is also shown that if $\lambda > 0$ decreases below a certain threshold value, no more bifurcations with respect to the set of stationary points will take place. Since the dominating term $\lambda \|(x_1, \ldots, x_n)\|_{2d}^{2d}$ can be regarded as a *penalty term* which has been added to the polynomial function $q(x_1, \ldots, x_n)$, it also allows one to compute bounds on the achievable global minimum of $q(x_1, \ldots, x_n)$ from the outcomes for values of $\lambda > 0$, especially once the location of a global minimizer is known to be confined to some compact subset of $\mathbb{R}^n$. However, if $\lambda > 0$ is taken too small, numerical problems are likely to arise.

A second possible approach for dealing with an arbitrary real polynomial $q(x_1, \ldots, x_n)$ is applicable if the first-order conditions generate an ideal with zero-dimensional variety and if it is known that the polynomial has a global minimum. In that case the first-order conditions can be brought into Gröbner basis form with respect to a total degree ordering (using the Buchberger algorithm for example). Then the Stetter–Möller matrix approach is applicable to find the critical points and critical values of the polynomial and algorithms like the one developed in the present paper could be constructed. How these approaches would work out precisely is left for future research.

The method of the present paper can also be employed to find the global minimum of a real multivariate *rational* function in an iterative way by solving a sequence of associated global minimization problems involving real polynomials; see Jibetean (2003). In this case, there are good opportunities for speeding up the iteration process by combining this algebraic approach with conventional numerical local search methods, which allow one to compute upper bounds on the global minimum. The general set-up of an iteration step of such combined methods consists of the computation of a candidate value for the global minimum (e.g. by local search) and a test of global minimality of this candidate value (by the algebraic construction of an eigenvalue problem as in the approach of the present paper). If the global minimality test fails, the method will yield a new starting point for a local search method which is guaranteed to lead to an improved candidate value for the global minimum.

With respect to efficiency of the approach discussed in this paper, several issues deserve further investigation. First, the known structure of the eigenvectors of the matrix A_r^{T} in the case of one-dimensional eigenspaces has not yet been employed in the Arnoldi and Jacobi–Davidson iterative methods, either to speed up convergence or to improve accuracy. This issue is currently under investigation. Second, there are several parts in the approach which allow for parallel computation. This holds true for the iterative eigenvalue solvers themselves, but it also applies to the computation of the action of A_r on a given vector v. For instance, when $r(x_1, \ldots, x_n)$ involves several terms, then each term is associated with a particular set of points in the n-dimensional time space, and the associated computations can be done in parallel. Also, the computation of the values of $y_{t_1,\ldots,t_n}$ at time instants of the same total time can all be done in parallel. The concept of a stable pattern can be used to organize such parallel computations efficiently. Third, it is still unclear for which polynomials $r(x_1, \ldots, x_n)$ it is more efficient to explicitly construct and use the matrix A_r^{T} and when it is more efficient to compute the related action using the nD system approach. Furthermore it is also unclear whether it is more efficient to determine first all the stationary points using a cheap matrix $A_{x_i}^{\mathrm{T}}$ and then to select the global

minimum, or to determine first only the optimal critical value using the more expensive matrix A_p^{T} and then to determine the corresponding global minimizer.

Finally, the approach may also be useful in situations where the (dominated) polynomials involve one or more symbolic parameters. The transformation into a large eigenvalue problem then leads to a parameterized family of operators A_r^{T}, of which the smallest real eigenvalues and their corresponding eigenvectors can be studied. Variation in the value of the smallest eigenvalue can then be considered to some extent separately from variation in the corresponding eigenvector, which determines the location of the minimizer.

Acknowledgements

The authors would like to thank the anonymous referees for helpful suggestions for improving the paper.

References

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

Cox, D., Little, J., O'Shea, D., 1998. Using Algebraic Geometry. Springer-Verlag.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

Floyd, R.W., 1962. Algorithm 97: Shortest path. Communications of the ACM 5 (6), 345.

Fokkema, D., Sleijpen, G., van der Vorst, H., 1998. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. SIAM Journal on Scientific Computing 20 (1), 94–125.

Fornasini, E., Rocha, P., Zampieri, S., 1993. State space realization of 2-D finite-dimensional behaviours. SIAM Journal on Control and Optimization 31 (6), 1502–1517.

Hanzon, B., Hazewinkel, M. (Eds.), 2006. Constructive Algebra and Systems Theory. Edita KNAW, Amsterdam.

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

Hanzon, B., Maciejowski, J., Chou, C., 1998. Model reduction in H2 using matrix solutions of polynomial equations. Technical Report CUED/F–INFENG/TR.314. Engineering Department, Cambridge University, Cambridge, UK.

Henrion, D., Lasserre, J.-B., 2003. GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi. ACM Transactions on Mathematical Software (TOMS) 29 (2), 165–194.

Jibetean, D., 2003. Algebraic optimization with applications to system theory. Ph.D. Thesis. Vrije Universiteit, Amsterdam.

Lasserre, J.B., 2001. Global optimization with polynomials and the problem of moments. SIAM Journal on Optimization 11 (3), 796–817.

Lasserre, J.B., 2002. An explicit equivalent positive semidefinite program for nonlinear 0–1 programs. SIAM Journal on Optimization 12 (3), 756–769.

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

Möller, H., Stetter, H., 1995. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. Numerische Mathematik 70, 311–329.

Parrilo, P.A., 2003. Semidefinite programming relaxations for semialgebraic problems. Mathematical Programming 96 (2), 293–320.

Parrilo, P.A., Sturmfels, B., 2003. Minimizing polynomial functions. Algorithmic and quantitative real algebraic geometry. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 60, 83–100.

Prajna, S., Papachristodoulou, A., Seiler, P., Parrilo, P.A., 2004. SOSTOOLS: Sum of squares optimization toolbox for MATLAB. http://www.mit.edu/~parrilo/sostools.

Reis, G., Mourrain, B., Rouillier, R., Trébuchet, P., 2002. An environment for symbolic and numeric computation. In: Proceedings of the International Conference on Mathematical Software. World Scientific, pp. 239–249. http://www-sop.inria.fr/galaad/software/synaps.

Sleijpen, G., van der Vorst, H., 1996. A Jacobi–Davidson iteration method for linear eigenvalue problems. SIAM Journal on Matrix Analysis and Applications 17 (2), 401–425.

Sturm, J., 1999. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software 11, 625–653. http://sedumi.mcmaster.ca.