Efficiency improvement in an $n$D-systems approach to polynomial optimization

Ivo Bleylevens$^a$, Ralf Peeters$^a$, Bernard Hanzon$^b$

$^a$Dept. Mathematics, Universiteit Maastricht, Maastricht, The Netherlands.
$^b$School of Mathematical Sciences, University College Cork, Cork, Ireland.

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-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 (such as methods based on Arnoldi and Jacobi-Davidson techniques) is a routine which computes the action of the matrix on a given vector. The huge number of required iterations is the main reason why the action of the matrix on a vector has to be computed efficiently. This paper focuses on this aspect of the optimization technique. To avoid building the large matrix one can associate the system of first order conditions with an $n$D-system of difference equations, by interpreting the variables in the polynomial equations as shift operators working on a multidimensional time series. Then one way to compute the action of the matrix on a vector efficiently, is by setting up a corresponding shortest path problem and to apply an algorithm like Dijkstra’s or Floyd’s algorithm, to solve it. In the case of 2D-systems the shortest path problem can be solved analytically. For $n$D-systems the situation is more complicated but a number of partial results are available. For large $n$ the shortest path problem has a high computational complexity, and therefore some heuristic procedures are developed to arrive cheaply at suboptimal paths with acceptable performance. The approach and techniques presented in this paper are demonstrated by means of a worked example involving a dominated polynomial of 4 variables and total degree 8.

Key words: global polynomial optimization, Gröbner basis, Stetter-Möller matrix method, $n$D-systems, large eigenvalue problems

* An earlier version of this paper was presented at MEGA2005.

Email addresses: i.bleylevens@math.unimaas.nl (Ivo Bleylevens), ralf.peeters@math.unimaas.nl (Ralf Peeters), b.hanzon@ucc.ie (Bernard Hanzon).

Preprint submitted to Elsevier Science 6 November 2005
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 sciences including statistics, mathematical finance, economics, systems biology etc. Non-convexity and the possible existence of local optima (often very many), 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 $\lambda$ is a positive real number. This class is of interest because information on the global minimum of $q$ can be obtained from $p_\lambda$ by letting $\lambda$ tend to zero, see Hanzon and Jibetean (2003); Jibetean (2003). Extensions and further applications are discussed in Section 10.

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 $p_\lambda$ 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 $p_\lambda$. 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 $p_\lambda$. 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 $p_\lambda$. 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 or Arnoldi methods, see Sleijpen and van der Vorst (1996); Fokkema et al. (1998); 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 $p_\lambda$ 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 $n$D-system. See Attasi (1976); 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 to apply an algorithm, like Dijkstra’s algorithm or Floyd’s algorithm (see Dijkstra (1959), Floyd (1962)), to solve it. This is the topic of Section 7. 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 $p_{\lambda}$, 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 9 the approach of this paper is demonstrated by means of a worked example and compared to other approaches and techniques available in the literature.

2 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 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

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

Note that the nomenclature for this family derives from the property that the value of $p_{\lambda}$ is dominated by the term $\lambda(x_1^{2d} + \ldots + x_n^{2d})$ when the Minkowski $2d$-norm $\|(x_1,\ldots,x_n)\|_{2d} = (x_1^{2d} + x_2^{2d} + \ldots + x_n^{2d})^{1/2d}$ becomes large. Consequently, the polynomial $p_{\lambda}$ 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), 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 condi-
tions. This leads to a system of $n$ polynomial equations in $n$ variables of the form

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

where

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

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

$$d^{(i)}(x_1, \ldots, x_n) = x_i^m - f^{(i)}(x_1, \ldots, x_n), \quad (i = 1, \ldots, n), \quad (4)$$

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

$$B = \{ x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \mid \alpha_1, \alpha_2, \ldots, \alpha_n \in \{0, 1, \ldots, m - 1\} \}. \quad (5)$$

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 $\mathbb{R}[x_1, \ldots, x_n]/I$ makes that the matrix method of Stetter-Möller can be applied to compute all the (complex) solutions to the system of equations $d^{(i)}(x_1, \ldots, x_n) = 0, \quad (i = 1, \ldots, n)$, 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 $d^{(i)}(x_1, \ldots, x_n) = 0, \quad (i = 1, \ldots, n)$. 

4
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

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 action of \( A_r^T \) rather than with the action of \( A_r \). To this end we set up an autonomous multidimensional system, also called an \( n \)D-system, associated with the set of polynomials \( d^{(i)}, (i = 1, \ldots, n) \). With any monomial \( x_1^{a_1}x_2^{a_2} \cdots x_n^{a_n} \) we associate an \( n \)D-shift operator \( \sigma_1^{a_1}\sigma_2^{a_2} \cdots \sigma_n^{a_n} \) which acts on any multidimensional time series \( y_{t_1, t_2, \ldots, t_n} \) according to the rule

\[
\sigma_1^{a_1}\sigma_2^{a_2} \cdots \sigma_n^{a_n} : y_{t_1, t_2, \ldots, t_n} \mapsto y_{t_1+a_1, t_2+a_2, \ldots, t_n+a_n}.
\]  

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:

\[
r(\sigma_1, \sigma_2, \ldots, \sigma_n)y_{t_1, t_2, \ldots, t_n} = 0.
\]  

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:

\[
y_{t_1, t_2, \ldots, t_n} = f^{(i)}(\sigma_1, \ldots, \sigma_n)y_{t_1, t_2, \ldots, t_n}, \quad (i = 1, \ldots, n).
\]  

This expresses the fact that the value of \( y_{t_1, t_2, \ldots, t_n} \) at any multidimensional ‘time instant’ \( \tilde{t} = (\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_n) \), such that \( \max\{\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_n\} \) 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 \(|\tilde{t}| := t_1 + t_2 + \ldots + t_n\) strictly less than the total time \(|\tilde{t}| = \tilde{t}_1 + \tilde{t}_2 + \ldots + \tilde{t}_n\). 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 values (total degree reversed lexicographically ordered; cf. (5)):

\[
w_{0,0,\ldots,0} := \{ y_{t_1, t_2, \ldots, t_n} \mid t_1, t_2, \ldots, t_n \in \{0, 1, \ldots, m-1\} \}.
\]

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 (8). 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
\[
w_{t_1, t_2, \ldots, t_n} := \{y_{t_1+s_1, t_2+s_2, \ldots, t_n+s_n} \mid s_1, s_2, \ldots, s_n \in \{0, 1, \ldots, m-1\}\}.
\] (10)

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
\[
w_{t_1+\alpha_1 t_2+\alpha_2, \ldots, t_n+\alpha_n} = \sigma_1^{\alpha_1} \sigma_2^{\alpha_2} \cdots \sigma_n^{\alpha_n} w_{t_1, t_2, \ldots, t_n},
\] (11)
where the \(n\)D-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 previous choice of basis for the quotient space \(\mathbb{R}[x_1, \ldots, x_n]/I\) and the choice of the ordering of the monomials, it holds that
\[
w_{t_1, t_2, \ldots, t_n} = \sigma_i w_{t_1, t_2, \ldots, t_n} = A_{x_i}^T w_{t_1, t_2, \ldots, t_n},
\] (12)
where the matrix \(A_{x_i}\) again denotes the matrix associated with multiplication by the polynomial \(x_i\) within the quotient space \(\mathbb{R}[x_1, \ldots, x_n]/I\). Note that its transpose \(A_{x_i}^T\) is involved in this relationship. As a consequence,
\[
w_{t_1+\alpha_1 t_2+\alpha_2, \ldots, t_n+\alpha_n} = (A_{x_1}^T)^{\alpha_1} (A_{x_2}^T)^{\alpha_2} \cdots (A_{x_n}^T)^{\alpha_n} w_{t_1, t_2, \ldots, t_n},
\] (13)
which shows that the general solution to the autonomous multidimensional system with initial state \(w_{0,0,\ldots,0}\) is given by
\[
w_{t_1, t_2, \ldots, t_n} = (A_{x_1}^T)^{t_1} (A_{x_2}^T)^{t_2} \cdots (A_{x_n}^T)^{t_n} w_{0,0,\ldots,0}.
\] (14)
and \(y_{t_1, t_2, \ldots, t_n}\) can be read off from this as it is an element of \(w_{t_1, t_2, \ldots, t_n}\). More generally, for an arbitrary polynomial \(r(x_1, \ldots, x_n)\) it holds that
\[
r(\sigma_1, \ldots, \sigma_n) w_{t_1, \ldots, t_n} = A_{r(x_1, \ldots, x_n)}^T w_{t_1, \ldots, t_n}.
\] (15)

**Example 1** To demonstrate the above described techniques a small worked example is given. The dominated polynomial to be minimized is chosen as:
\[
p_1(x_1, x_2) = \lambda(x_1^4 + x_2^4) + x_1^3 + 2x_1^2 + 3x_1x_2
\]
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
\[
4x_1^3 + 3x_1^2 + 4x_1 + 3x_2 = 0
\]
\[
4x_2^3 + 3x_1 = 0
\]
of which the real solutions constitute the stationary points of \(p_1(x_1, x_2)\). The ideal \(I\) is generated by the first order partial derivatives of this polynomial. The quotient space \(\mathbb{R}[x_1, \ldots, x_n]/I\) is of dimension \((2d - 1)^n = 9\). A basis
for this space is given by (the equivalence classes of) the set of monomials
\{1, x_1, 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 corre-
sponds to a convenient permutation of the total degree reversed lexicographic
monomial ordering) the matrices \(A_{x_1}\) and \(A_{x_2}\) which represent the linear op-
erations of multiplication by \(x_1\) and \(x_2\), respectively, are easily computed as

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

and

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

Of course, for such small sizes computations are easy and it is not difficult
to verify that the matrices \(A_{x_1}\) and \(A_{x_2}\) commute. Both matrices have 9 dis-
tinct eigenvalues, which can be paired according to the existence of common
eigenspaces, yielding 9 solutions in \(\mathbb{C}^2\). Restricting to real solutions, 3 sta-
tionary points are obtained: \((-0.631899, 0.779656), (0.346826, -0.638348)\) and
\((0, 0)\), which can be classified, respectively as a global minimizer, a local mini-
mizer and a saddle point. The corresponding critical values of \(p_1(x_1, x_2)\) are,
respectively, \(-0.402778, -0.201376\) and 0.

To focus attention on the computation of the actions of the matrices \(A_{x_1}^T\) and
\(A_{x_2}^T\) directly from the underlying first order partial derivatives of \(p_1(x_1, x_2)\)
which generate the ideal \( I \), without computing these matrices explicitly, the following associated system of 2-dimensional difference equations is considered:

\[
4y_{t+3,t_2} + 3y_{t+2,t_2} + 4y_{t+1,t_2} + 3y_{t,t_2+1} = 0 \\
4y_{t,t_2+3} + 3y_{t+1,t_2} = 0
\]

which is rewritten in the following equivalent form:

\[
y_{t+3,t_2} = -\frac{3}{4}y_{t+2,t_2} - y_{t+1,t_2} - \frac{3}{4}y_{t,t_2+1} \\
y_{t,t_2+3} = -\frac{3}{4}y_{t+1,t_2}
\]

An initial state for this autonomous 2-dimensional system is constituted by \( w_{0,0} = (y_{0,0}, y_{1,0}, y_{2,0}, y_{0,1}, y_{1,1}, y_{2,1}, y_{0,2}, y_{1,2}, y_{2,2})^T \) which corresponds to a \( 3 \times 3 \) square array of values at integer time instants \((t_1, t_2)\) in the 2-dimensional time plane. Then the action of the matrix \( A_{x_2}^T \) on \( w_{0,0} \) yields the vector \( A_{x_2}^T w_{0,0} = (y_{0,0}, y_{0,1}, y_{0,2}, y_{1,0}, y_{1,1}, y_{1,2}, y_{2,0}, y_{2,1}, y_{2,2})^T \), which also corresponds to a \( 3 \times 3 \) square array of integer time instants \((t_1, t_2)\) in the 2-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.

The action of the matrix \( A_{x_2}^T \) on \( w_{0,0} \) yields the vector \( A_{x_2}^T w_{0,0} = (y_{0,1}, y_{1,1}, y_{2,1}, y_{0,2}, y_{1,2}, y_{2,2}, y_{0,3}, y_{1,3}, y_{2,3})^T \), which can be computed in a similar fashion. More generally, the action of any matrix \( r(A_{x_1}^T, A_{x_2}^T) \) on \( w_{0,0} \), where \( r(x_1, x_2) \) is some arbitrary polynomial, can be computed as a linear combination of the values of the 2D time series \( y_{t_1,t_2} \) at various \( 3 \times 3 \) square arrays of 2D time instants. For instance, the action of the matrix \( p_1(A_{x_1}^T, A_{x_2}^T) \) on \( v \) can be obtained by a linear combination of the individual actions of the matrices \( (A_{x_1}^T)^4, (A_{x_2}^T)^4, (A_{x_1}^T)^3, (A_{x_2}^T)^3 \) and \( A_{x_1}^T, A_{x_2}^T \) on \( w_{0,0} \). Here it is noted that the action of any matrix of the form \( (A_{x_1}^T)^{\alpha_1} (A_{x_2}^T)^{\alpha_2} \) on \( w_{0,0} \) yields the vector \( (A_{x_1}^T)^{\alpha_1} (A_{x_2}^T)^{\alpha_2} w_{0,0} = (y_{01,0}, y_{01,1}, y_{01,2}, y_{11,0}, y_{11,1}, y_{11,2}, y_{21,0}, y_{21,1}, y_{21,2})^T \).

Recall that our interest is in computing the eigenvalues of the matrix 
\( A_r(x_1,...,x_n) \), which coincide with the eigenvalues of its transpose \( A_r^T(x_1,...,x_n) \) and which may conveniently be studied from the perspective of the autonomous \( n \)D-system introduced above. Note that if \( v \) is an eigenvector of \( A_{x_i}^T \) with a corresponding eigenvalue \( \xi_i \), then it holds that

\[
A_{x_i}^T v = \xi_i v. \tag{16}
\]

In terms of the \( n \)D-system this implies that the choice \( w_{0,...,0} := v \) for the initial state produces a scaled version for the state: \( w_{0,...,1,0,...,0} = \xi_i v \), 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 \( w_{0,0,...,0} \) and \( w_{0,0,...,1,0,...,0} \) have \( m^n - m^{n-1} \)
elements in common (in shifted positions), showing that the eigenvectors of
\( A_{x_i}^T \) exhibit a special structure.

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 \( p_\lambda \) now proceeds as follows. First a suitable
choice for the polynomial \( r \) is made and the corresponding matrix \( A_{r}^T \) is con-
structed. 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 \( A_{x_i} \) to obtain the values \( \xi_i, i = 1, \ldots , n \). By
choosing \( r \) appropriately one can ascertain that all the eigenvalues of \( A_r \) have
geometric multiplicity one. This will in fact hold for a ‘generic’ choice of \( r \).

Remark: In fact, in case all the eigenvalues of \( A_r \) are distinct, i.e. all have
(algebraic) multiplicity one, then it is well-known that \( A_r^T = V \Lambda L^T \), where
\( L^T V = I \) 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 \( \Lambda = \text{diag}(\lambda_1, \ldots , \lambda_N) \) is a diagonal ma-
trix with \( \lambda_k = r(v(k)), k = 1, 2, \ldots , N \). The matrix \( L \) has the property that
its \( k \)-th 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 \( j \)-th ba-
sis 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 \( L^T = V^{-1} \).
The fact that the eigenvectors of \( A_r^T \) are columns of the generalized Vander-
monde matrix \( V \) was noted by Stetter, which is the reason that eigenvectors
of this form are sometimes called Stetter vectors. The structure of these vec-
tors 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 \( B \), hence their values appears in the Stetter
vector!

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 corre-
sponding minimizer(s) can be read off.

However, a serious bottleneck in this approach from a computational point of
view is constituted by the eigenvalue and eigenvector calculations that have
to be performed for 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 \( n \)D-system (8) and in addition the matrix \( A_r^T \) may be highly sparse and
structured for particular choices of the polynomial \( r \) (this typically occurs if
iterative methods for large eigenvalue problems

4 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 (see, e.g., Lehoucq et al. (1998)) or Jacobi-Davidson (see, e.g., Fokkema et al. (1998); Sleijpen and van der Vorst (1996)). Such methods have the attractive feature that they do not operate on the matrix $A^T R$ 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 vector it is supplied with. The nD-system approach supports this and it offers a framework to compute the action of $A^T R$ on a given vector $v$, by initializing the initial state as $w_0 := v$ and using the recursions (8) in combination with the relationship (15) to obtain the vector $r(\sigma_1, \ldots, \sigma_n) w_0 = A^T R w_0$. Such an approach entirely avoids an explicit construction of the matrix $A^T R$. Note that $r(\sigma_1, \ldots, \sigma_n) w_0$ 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 $r_{\alpha_1, \ldots, \alpha_n} w_{t_1, \ldots, t_n}$. This makes clear that for any choice of polynomial $r$ the vector $r(\sigma_1, \ldots, \sigma_n) w_0$ 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 (8) and the initial state $w_0$. 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 $p_\lambda$ first, 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. Given an initial state $w_0$, 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 (8)) from the initial state $w_0$. They require the use of more than one recursion from the set (8) and they involve the computation of the value 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 $p_\lambda$ first, rather than on the stationary points, then the polynomial $r$ may be chosen as $p_\lambda$ 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 $A^T_{p\lambda}$, 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 $A^T_{x_i}$. But for $A^T_{p\lambda}$, only the smallest real eigenvalue is required that corresponds to a real stationary point, whereas the eigenvalues of the matrices $A^T_{x_i}$ correspond to the $i$-th coordinate of the stationary points (for which it is usually far less clear in advance in which range the value at the global optimum will be located). Therefore, when dealing with $A^T_{p\lambda}$, one may focus on the calculation of only a few eigenvalues, as one is interested in the smallest real eigenvalue that corresponds to a real solution point (Arnoldi and Jacobi-Davidson methods allow the user to ‘zoom in’ on a few eigenvalues), whereas in case of $A^T_{x_i}$ 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 $A^T_{p\lambda}$ will be used.

5 Problem statement

This raises the following three research questions studied in 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 (8)? And as a closely related question, what is the most efficient way to compute the state vector $w_{t_1, \ldots, t_n}$? (ii) For which polynomials $r(x_1, \ldots, x_n)$ is it more efficient to compute the related action using the $n$D-system approach and when is it more efficient to explicitly construct and use the matrix $A_r$? (iii) For the problem of finding a global minimizer of the polynomial criterion $p\lambda$, which of the following approaches is more efficient: first to determine all the stationary points using a cheap matrix $A^T_{x_i}$ and then to select the global minimum, or first to determine only the optimal critical value using the more expensive matrix $A^T_{p\lambda}$ and then to determine the corresponding global minimizer?

6 Linear complexity for computing a value of the multidimensional time series

With respect to the first research question raised above, the following results address 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 (8). For each multidimensional time instant $t = (t_1, \ldots, t_n)$, let the ‘total time’ be 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 6.1** Consider a set of \(n\) multidimensional recursions of the form (8) 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 the recursions (8) has a computational complexity which increases at least linearly with the total time \(|t|\).

**Proof.** Each recursion from the set (8) 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 \(\lceil \frac{(|t| - n(m - 1))/m}{|t|} \rceil\) applications of a recursion from the set (8). Hence, the computational complexity of any algorithm along such lines increases at least linearly with \(|t|\). \(\square\)

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

### 7 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 (8) is in general not easy, but can be cast into the form of a shortest path problem. There are various ways do this. 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 (or nodes), 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 classical algorithm (e.g., the algorithm of Dijkstra (see Dijkstra (1959)) or the algorithm of Floyd (see Floyd (1962))). 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 (8). The weight function \( W \) specifies the computational ‘costs’, (e.g., the number of flops) associated with these recursions.

In setting up a shortest path 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 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 (8) 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 7.1** 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 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 \( v = (v_1, \ldots, v_{|t|}) \in V_1 \times V_2 \times \cdots \times V_{|t|} \). Define the initial state as \( v_I = (\phi, \ldots, \phi) \) (where \( \phi \) 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 \((t_1, \ldots, t_n)\) at which \(y_{t_1, \ldots, t_n}\) can be computed from the values at the time instants contained in the union of sets \( V_\ast \cup v_1 \cup v_2 \cup \ldots \cup v_{k-1} \) through the application of a single recursion from the set \((8)\).

Define \( W : E \rightarrow \mathbb{R} \) 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 \((8)\) 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 \( G = (V, E, W, v_\ast, v_T) \). 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 \((8)\).

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 \( \hat{v} \) 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 \((t_1, \ldots, t_n)\) at which \(y_{t_1, \ldots, t_n}\) can be computed from the values at the time instants contained in the union of sets \( V_\ast \cup T_1 \cup \ldots \cup T_{k-2} \cup v_{k-1} \) through the application of a single recursion from the set \((8)\). 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 and makes it easier to compute a solution. The graph then no longer has a tree structure. 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 \((8)\) are not sparse, this lower bound is likely to be close to the optimal value.

### 7.1 The 2D-case

In the 2D-case (i.e., \( n = 2 \)) it is possible to solve the relaxation of the shortest path problem analytically, for the situation where the recursions in \((8)\) describe ‘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) and ‘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 and for each
time instant \((t_1, t_2)\). In fact, in the 2D-case the solution for the relaxation turns out to be feasible for the shortest path problem itself too. For \(n > 2\) the situation becomes more complicated, but the machinery introduced in this section allows one to derive partial results. For the 3D-case, such results are given in the next subsection.

As described above, the directed graph \(G = (V, E, W, v_I, v_T)\) for the relaxation of the shortest path problem involves a set \(V\) of ‘vertices’ or ‘nodes’, which are connected by (directed) edges in \(E\). With each value \(k = 1, 2, \ldots\) a finite set of nodes can be associated. These are constituted by all the subsets of time instants with total time \(k\) (thereby excluding the time instants in the set \(V^*\)). An edge occurs in \(E\) from a node \(N_a\) associated with a total time \(k\) to a node \(N_b\) associated with a total time \(k + 1\) if and only if \(y_{t_1, \ldots, t_n}\) can be computed at all the time instants \((t_1, \ldots, t_n)\) in \(N_b\) from the values of \(y_{t_1, \ldots, t_n}\) at the time instants in \(N_a\), at time instants with total time less than \(k\), and at the time instants in \(V^*\). Since a node consists of time instants all sharing the same total time, the ‘internal structure’ of nodes can be compared by translation. To describe the structure of an optimal solution to the relaxation of the shortest path problem, the concept of a ‘stable pattern’ exhibited by a node is useful.

**Definition 7.2** In the graph \(G = (V, E, W, v_I, v_T)\) corresponding to the relaxation of the shortest path problem, a node \(N_b \in V\) is said to exhibit a stable pattern if there exists a node \(N_a \in V\) which is a translation of \(N_b\) and for which there is an edge from \(N_a\) to \(N_b\) (i.e., \((N_a, N_b) \in E\)).

A translation \(T_s\) involving a translation vector \(s = (s_1, \ldots, s_n)\) acts on a node \(N_b\) by acting element-wise on each of the time instants \((t_1, \ldots, t_n)\) contained in \(N_b\), according to the rule \((t_1, \ldots, t_n) \mapsto (t_1 + s_1, \ldots, t_n + s_n)\). Clearly, translation vectors \(s\) associated with stable patterns have the property that \(s_1 + s_2 + \ldots + s_n = -1\). 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 shifts along the (negative) directions of the time axes.

When a node \(N_b \in V\) exhibits a stable pattern with an associated translation \(T_s\), it holds that \(N_a = T_s(N_b) = N_b + s\) is also a node in \(V\) with a total time decreased by 1. Repeated application of the translation \(T_s\) produces a sequence of nodes in \(V\) with decreasing total times \(k, k - 1, k - 2, \ldots\), until translation takes a node outside the non-negative orthant. When several translations are associated with the same stable pattern, more complicated sequences of translations can be considered. Now the idea is to construct an optimal path which solves the relaxation of the shortest path problem as a composition of three parts: (i) an initial part which connects the initial node \(v_I\) to a node exhibiting a stable pattern, (ii) a middle part which consists entirely of translations of the stable pattern, (iii) a terminal part, which connects a node exhibiting a
stable pattern to the terminal node \( v_T \).

Of course, if a stable pattern is to assist in the construction of a shortest path, the associated costs need to be as small as possible. Since we are considering the case of full recursions and uniform costs, this means that the size of a stable pattern needs to be as small as possible. The size of a stable pattern associated with a certain node, say \( N_b \), is defined as its cardinality \( |N_b| \) (i.e., the number of time instants in \( N_b \)). A stable pattern is called a minimal stable pattern if it does not contain a strict subset which is also a stable pattern.

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

In the 2D-case, the nodes are the subsets of the diagonals for which \( t_1 + t_2 \) is constant. For the situation where \( m = 3 \), a minimal stable pattern consists of a subset of 5 consecutive points on a diagonal. This is depicted in Figure 1, together with translations on several subdiagonals. For an arbitrary value of \( m \) we have the following result.

**Proposition 7.3** 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 compute a point on a diagonal with total time \( k \) one of the two full recursions is required. Such a full recursion involves \( m \) consecutive points on
the first subdiagonal (with total time $k - 1$). 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 recursions, then this requires either $2m - 1$ consecutive points on the next subdiagonal (with total time $k - 2$) or it requires $2m$ points (consisting of two groups of $m$ points). When there are no two consecutive points employing two different recursions, then all $m$ points involve the same recursion and this also requires $2m - 1$ consecutive points on the next subdiagonal (with total time $k - 2$). 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 $m$-th 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 shift in the (negative) $t_1$-direction. When the recursion involving $f^{(2)}$ is employed, the translation involves $s = (0, -1)$, which constitutes a shift in the (negative) $t_2$-direction. Hence the minimal stable pattern can be associated with shifts in both directions of the time axes.

From the proof above, it is clear that the minimal stable pattern of $2m - 1$ consecutive points can be used to construct the middle part of a solution to the relaxation of the shortest path problem in the 2D-case. It also is indicated how the terminal part of such a solution can be constructed. To compute a point on a diagonal with total time $k$, this first requires the computation of $m$ consecutive points at the subdiagonal with total time $k - 1$, and then a stable pattern of $2m - 1$ points on the next subdiagonals (with total time $k - 2, k - 3, k - 4, \ldots$). It remains to investigate the initial part of a solution. To do this, 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 contained in $V_*$. 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 $V_*$ require computation. This is summarized in the following result, which is illustrated in Figure 2.
Proposition 7.4 In the 2D-case with full recursions and uniform costs, a solution to the relaxation of the shortest path problem has the following structure.

(i) An initial part starting at \( v_I \) and involving the nodes \( v_m, v_{m+1}, \ldots, v_{2m-2} \), defined as \( v_k = \{ (t_1, t_2) | t_1 + t_2 = k, t_1 \geq 0, t_2 \geq 0, \max t_1, t_2 \geq m \} \) for \( k = m, m+1, \ldots, 2m-2 \).

(ii) A middle part involving the nodes \( v_{2m-1}, v_{2m}, \ldots, v_{\ell-2} \), with \( \ell \geq 2m \) denoting the total time of the terminal node \( v_T \). Each of the nodes \( v_k \) in this part involves a minimal stable pattern of \( 2m-1 \) consecutive points on the diagonal with total time \( k \). Each two consecutive nodes are related by a translation \( T_s \) with \( s = (-1,0) \) or \( s = (0,-1) \).

(iii) A terminal part involving the nodes \( v_{\ell-1} \) and \( v_T \). The node \( v_{\ell-1} \) involves \( m \) consecutive points on the diagonal with total time \( \ell - 1 \) such that there is an edge from \( v_{\ell-2} \) to \( v_{\ell-1} \) and an edge from \( v_{\ell-1} \) to \( v_T \).

Note that the above discussion is valid only when the terminal node \( v_T \) involves a point having a total time \( \geq 2m \). For a terminal node with total time \( \leq 2m-1 \), no stable pattern occurs in the optimal path because there is no room for it in the non-negative orthant below the diagonal with total time \( 2m - 2 \). Such a situation can easily be investigated in an analogous way though, and it is not difficult to compute a shortest path either. The details of this are left to the reader.

Interestingly, any solution to the relaxation of the shortest path problem in the 2D-case with full recursions and uniform costs is also a solution to the shortest path problem itself. To see this, note that the relaxation 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 (negative) time axes, this area is eventually entirely covered. See again Figures 1-2.

Corollary 7.5 In the 2D-case with full recursions and uniform costs, a solution to the relaxation of the shortest path problem is also a solution to the
shortest path problem itself.

7.2 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, i.e., 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 of minimal size. Here we present partial results for the cases $m = 2$ and $m = 3$. These results also provide a basis for the heuristic computational procedures developed and discussed in Section 8.

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 $\mathbb{V}$, 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 Figures 3-4. In Figure 4 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.

![Fig. 3. (a) The initial state $w_{0,0,0}$. (b) Full recursions to compute $y_{2,0,0}$, $y_{0,2,0}$, $y_{0,0,2}$.](image-url)
Proposition 7.6 In the nD-case a stable pattern is exhibited by translations of the set of time instants \( N = \{(t_1, \ldots, t_n) \mid t_1 + \ldots + t_n = n(m-1); t_1 \geq 0, \ldots, t_n \geq 0\} \). Such a stable pattern allows for translations in each of the directions of the time axes.

Proof. From the values of \( y_{t_1, \ldots, t_n} \) at the time instants contained in \( V_\ast \), 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 largest total time contained in \( V_\ast \). 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 \( s = (1,0,\ldots,0), (0,1,\ldots,0), \ldots, s = (0,\ldots,0,1) \). 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. \( \Box \)

However, for \( n > 2 \) the stable pattern \( N \) in the above proposition is not a minimal stable pattern. 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 \( k - 1 \). One may also construct minimal stable patterns by deleting points from a stable pattern of the form \( N \) until ‘stability’ of the pattern is lost.

In Figure 5 it is shown for the case \( n = 3 \) and \( m = 2 \) how the points with total time \(|t| = 4\) are connected along the directions of the time axes to nearby

Fig. 4. Stable pattern (green) for shifts in all three directions.
points of total time $|t| = 3$. In Figure 6 the points in these two consecutive layers of constant total times 3 and 4 are arranged in convenient triangulated patterns and shown from two different viewpoints.

Fig. 5. Connections of the points with total time $|t| = 4$ to nearby points of total time $|t| = 3$.

Fig. 6. Two different viewpoints of the connections between layers of consecutive total time.
In Figure 7 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 are required to achieve this.

![Figure 7](image)

Fig. 7. Computation of points with total time $|t| = 4$ using three different recursions involving points with $|t| = 3$.

In the case $n = 3$ and $m = 2$, the stable pattern $N$ in the above proposition consists of 10 points. When two of its corner points are deleted, a minimal stable pattern of 8 points remains, which however allows only for a shift in just one of the directions of the time axes. When one corner point is deleted, a stable pattern of 9 points remains. This pattern is the smallest stable pattern which allows for a shift in each of the three directions of the time axes.

As a consequence, a solution to the relaxation of the shortest path problem needs to take into account the value of the time instant $(t_1, t_2, t_3)$ associated with the terminal node $v_T$. An ‘expensive’ stable pattern containing 9 points can be used for shifting along the two axes corresponding to the two smallest values of $t_1$, $t_2$ and $t_3$. Next, a ‘cheap’ stable pattern containing 8 points, which should be a subpattern of the previous stable pattern of 9 points, can be used for shifting along the third axis, corresponding to the largest value of $t_1$, $t_2$ and $t_3$. Note that such a solution constitutes also a solution to the shortest path problem itself, because the stable patterns have a convex configuration.

In the case $n = 3$ and $m = 3$, the non-minimal stable pattern $N$ is depicted in Figure 8. Smaller stable patterns which allow for shifts in just one of the directions of the time axes are depicted in Figure 9. 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.
8 Some heuristic procedures for the shortest path problem

From the discussion in the previous section it is clear that the complexity and the size of the shortest path problem increases rapidly with $n$. However, optimal solution of the shortest path problem is not a goal by itself; it serves to facilitate the efficient computation of state vectors $w_{t_1,t_2,\ldots,t_n}$. To this end, five heuristic methods were developed to calculate the value $y_{t_1,\ldots,t_n}$ that appears in a state vector $w_{t_1,t_2,\ldots,t_n}$ with a suboptimal but acceptable computational effort, thereby building on the knowledge about the optimal solutions in 2D-case and 3D-case obtained in the previous section. The performance of these methods is compared for the situation where we have 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 diagonal (simplex) which corresponds to the total time \( n(m - 1) \), thereby just covering the area (hypercube) of initial values. Then this stable pattern of values (see Proposition 7.6) is shifted step by step, with each step involving a single shift in one of the axis directions, until the requested location is reached. The sequence of shifts may depend on the multidimensional time instant at the requested location, but this does not affect the total amount of points encountered in this procedure and it is unimportant for the purpose of this section.

(ii) The diagonal method proceeds by first computing \( y_{t_1,...,t_n} \) for all the time instants with constant total time \( |t| = 1 \), then all those with total time \( |t| = 2 \), then with total time \( |t| = 3 \), and so on, increasing the value of \( |t| \) one by one until the requested location is reached.

(iii) The equalizing method proceeds by computing a value \( y_{t_1,...,t_n} \) by employing that recursion of the system (8) which reduces the largest possible coordinate of the time instant \( y_{t_1,...,t_n} \). (It uses a lexicographic ordering in case there is more than one largest coordinate.) The path between the initial state \( w_{0,...,0} \) and the requested location is determined in a backwards fashion, i.e., starting from the requested location.

(iv) The axis method proceeds by computing a value \( y_{t_1,...,t_n} \) by employing that recursion of the system (8) which reduces the smallest possible coordinate of the time instant \( y_{t_1,...,t_n} \). (It uses a lexicographic ordering in case there is more than one smallest possible coordinate.) The path between the initial state \( w_{0,...,0} \) and the requested location is determined in a backwards fashion.

A slightly more sophisticated fifth method has been designed on the basis of insights gained from the test experiments with the four methods introduced above. It is discussed later in this section.

For \( n = 2 \) and \( n = 3 \) the paths to some arbitrarily chosen points have been computed using the four described methods. The results are collected in Tables 1-2 below. For each method the 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 (floating point operations) and the actual running time (in ms obtained on a PC platform with an Intel Pentium PIV 2.8GHz processor and 512MB internal memory) are given for each requested point.

From Tables 1-2 we may conclude that the linear method indeed exhibits a linear complexity with respect to the total time \( |t| \). 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 points \((10, 10), (50, 50)\) or \((100, 100)\). For \( n = 2 \) this method generates stable patterns which constitute
Table 1
2-Dimensional case (2 × 2 initial state)

<table>
<thead>
<tr>
<th>Coordinates requested point</th>
<th>Method</th>
<th>Linear</th>
<th>Diagonal</th>
<th>Equalizing</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stored points</td>
<td>36</td>
<td>91</td>
<td>46</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>100</td>
<td>265</td>
<td>167</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(0,10)</td>
<td>Stored points</td>
<td>156</td>
<td>1431</td>
<td>546</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>460</td>
<td>4285</td>
<td>2181</td>
<td>718</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>32</td>
<td>23</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>(0,50)</td>
<td>Stored points</td>
<td>306</td>
<td>5356</td>
<td>1921</td>
<td>304</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>910</td>
<td>16060</td>
<td>7697</td>
<td>1468</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>16</td>
<td>40</td>
<td>55</td>
<td>20</td>
</tr>
<tr>
<td>(0,100)</td>
<td>Stored points</td>
<td>66</td>
<td>276</td>
<td>64</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>209</td>
<td>820</td>
<td>239</td>
<td>598</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>30</td>
<td>20</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>(50,50)</td>
<td>Stored points</td>
<td>306</td>
<td>5356</td>
<td>304</td>
<td>1723</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>1009</td>
<td>16060</td>
<td>1199</td>
<td>7918</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>40</td>
<td>40</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>(100,100)</td>
<td>Stored points</td>
<td>606</td>
<td>20706</td>
<td>604</td>
<td>5948</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>2009</td>
<td>62110</td>
<td>2399</td>
<td>27758</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>20</td>
<td>120</td>
<td>50</td>
<td>220</td>
</tr>
</tbody>
</table>

Table 2
3-Dimensional case (2 × 2 × 2 initial state)

<table>
<thead>
<tr>
<th>Coordinates requested point</th>
<th>Method</th>
<th>Linear</th>
<th>Diagonal</th>
<th>Equalizing</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Stored points</td>
<td>120</td>
<td>560</td>
<td>159</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>344</td>
<td>2216</td>
<td>832</td>
<td>399</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>12</td>
<td>18</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>(0,0,10)</td>
<td>Stored points</td>
<td>520</td>
<td>27720</td>
<td>4825</td>
<td>408</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>1544</td>
<td>110856</td>
<td>32328</td>
<td>2199</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>23</td>
<td>193</td>
<td>89</td>
<td>18</td>
</tr>
<tr>
<td>(0,0,50)</td>
<td>Stored points</td>
<td>1020</td>
<td>192920</td>
<td>29808</td>
<td>808</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>3044</td>
<td>771656</td>
<td>214696</td>
<td>4440</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>32</td>
<td>1017</td>
<td>1875</td>
<td>33</td>
</tr>
<tr>
<td>(0,0,100)</td>
<td>Stored points</td>
<td>606</td>
<td>20706</td>
<td>604</td>
<td>5948</td>
</tr>
<tr>
<td></td>
<td>Flops</td>
<td>2009</td>
<td>62110</td>
<td>2399</td>
<td>27758</td>
</tr>
<tr>
<td></td>
<td>Cpu time(ms)</td>
<td>20</td>
<td>120</td>
<td>50</td>
<td>220</td>
</tr>
</tbody>
</table>

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 points (0,10), (0,50) or (0,100). 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 \(t_1 = 0\) 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 \( w_{0,500}, w_{125,375}, w_{250,250}, w_{375,125} \) and \( w_{500,0} \).

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 Figure 10. Likewise, the above claims for the axis method are visualized in Figure 10.

The axis and equalizing methods are only efficient in some specific situations as shown in the Figures 10 and 11. Things get worse if one wants to compute a whole set of state vectors involving several different time instants at once, such as required for the operator \( A_t^r \) when \( r \) involves several terms. In Figure 12 the points are displayed which are computed by the four 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.

In the upper left corner the points computed by the linear method are plotted. This way of computing simultaneous paths to the requested time instants clearly is the most efficient. In the upper right plot the points calculated by
the inefficient diagonal method are shown. Using this method, obviously too
many points are computed. 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.

Fig. 12. Points required by the four methods for computing the points (0, 500),
(125, 375), (250, 250), (375, 125) and (500, 0) with a 3 × 3 initial state.

Because the linear method is efficient for small values of \( n \) but becomes less
efficient when the dimension of the problem increases, a fifth method has been
implemented with almost the same performance as the linear method in this
small experiment. This fifth method applies that recurrence relation of the sys-
tem 8 that requires a minimal number of new points to be calculated. It also
proceeds in a backwards fashion (starting from a requested point and moving
towards the initial state) but when constructing a path it takes the points
into account that have already been included in the path. It turns out that
this method becomes more efficient than the linear method for larger values
of \( n \). Figure 13 shows a plot of the evaluated points needed for the computa-
tion of the points (0, 500), (125, 375), (250, 250), (375, 125) and (500, 0). This
fifth method only takes advantage of available points that are close to the
point under consideration. Obviously, more sophisticated methods can also be
designed which take the global structure of the requested points into account.

To get an idea of the complexity of the first four heuristic methods another
experiment was carried out in the 2D-case and in the 3D-case. The relative growth in the number of stored points needed for the computation of a requested point was investigated for two situations: when moving along the diagonal \( t_1 = t_2 \) and when moving along a time axis. The increase in the number of stored points is displayed in Tables 3-4. An increase by a factor 2 in two dimensions indicates, for example, the increase of stored points when comparing the points \((0, 50)\) and \((0, 100)\) or the points \((50, 50)\) and \((100, 100)\). An increase factor of 5 in three dimensions concerns the increase of stored points when moving from \((10, 10, 10)\) to \((50, 50, 50)\), or from \((0, 0, 10)\) to \((0, 0, 50)\).

**Table 3**
Increase in the number of stored points along the diagonal.

<table>
<thead>
<tr>
<th>Increase factor</th>
<th>Diagonal method: ( n = 2 / n = 3 )</th>
<th>Linear/Equalizing method: ( n = 2 ) and ( n = 3 )</th>
<th>Axis method: ( n = 2 / n = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.9 / 7.6</td>
<td>2.0</td>
<td>3.5 / 5.1</td>
</tr>
<tr>
<td>5</td>
<td>19.4 / 86.9</td>
<td>4.8</td>
<td>12.0 / 29.6</td>
</tr>
<tr>
<td>10</td>
<td>75.0 / 662.3</td>
<td>9.4</td>
<td>41.6 / 150.9</td>
</tr>
</tbody>
</table>

**Table 4**
Increase in the number of stored points along the axis.

<table>
<thead>
<tr>
<th>Increase factor</th>
<th>Diagonal method: ( n = 2 / n = 3 )</th>
<th>Linear/Axis method: ( n = 2 ) and ( n = 3 )</th>
<th>Equalizing method: ( n = 2 / n = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.7 / 7.0</td>
<td>2.0</td>
<td>3.5 / 6.2</td>
</tr>
<tr>
<td>5</td>
<td>15.7 / 49.5</td>
<td>4.5</td>
<td>11.9 / 30.3</td>
</tr>
<tr>
<td>10</td>
<td>58.9 / 344.5</td>
<td>8.8</td>
<td>41.8 / 187.5</td>
</tr>
</tbody>
</table>

From the Tables 3 and 4 it is concluded that for the equalizing method the number of points required for the computation of points with almost equal coordinates indeed grows linearly with the increase factor. For the axis method a similar statement holds for points along the time axes.

### 9 Comparison of computational methods for a worked example

In this section we demonstrate the computational approach described in the previous sections by an example. This concerns the computation of the minimum value and the coordinates of the global minimizer of a (Minkowski-norm)
dominated polynomial in 4 variables and of total degree 8. This polynomial is
chosen as:

\[
p_1(x_1, x_2, x_3, x_4) = (x_1^8 + x_2^8 + x_3^8 + x_4^8) + x_1x_2x_3x_4^2 + 3x_1x_2 + x_2x_3 + 3x_4^2 + 2x_3x_4 + x_4 + 8.
\]  

(17)

In other words, \( n = 4, d = 4, \lambda = 1 \) and \( q(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4^2 + 3x_1x_2 + x_2x_3 + 3x_4^2 + 2x_3x_4 + x_4 + 8. \)

Before we discuss the computation of the global minimum of this polynomial
using the \( n \)D-system approach and the iterative eigenvalue solvers, 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 all the eigenvalues are computed using standard eigenproblem solvers.

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, Service Pack 3, and Mathematica 5.2, running
on an Intel Pentium PIV 2.8GHz platform with 512 MB of internal memory.

The first order partial derivatives of the polynomial (17), which make up the
first order conditions, are given by:

\[
\begin{align*}
8x_1^7 + x_2^2x_3^2 + 3x_2, \\
8x_2^7 + x_2^2x_3^2 + 3x_1 + x_3, \\
8x_3^7 + 2x_4^2x_1x_2x_3 + x_2^2 + x_2 + 2x_4, \\
8x_4^7 + 2x_4x_1x_2x_3^2 + 2x_4x_3^2 + 2x_3 + 1.
\end{align*}
\]  

(18)

The real solutions of the corresponding system of first order conditions are the
stationary points of the polynomial \( p_1(x_1, x_2, x_3, x_4) \). The quotient space
is of dimension \((2d - 1)^n = 2401\). The matrices \( A_{x_1}, A_{x_2}, A_{x_3}, \) and \( A_{x_4} \), which
represent multiplication by the variables \( x_1, x_2, x_3, \) and \( x_4 \), respectively, are
of dimensions \( 2401 \times 2401 \). All these matrices have 2401 distinct eigenvalues.

The matrices \( A_{x_1} \) and the matrix \( A_{p_1} \) are highly sparse; they contain \( 2401^2 = 5,764,801 \) elements of which 4619 are nonzero in the matrix \( A_{x_1} \) and 43178 are
nonzero in the matrix \( A_{p_1} \). See Figure 14 for a representation of the sparsity
structure of the matrix \( A_{x_1} \) (on the left hand side) and of the matrix \( A_{p_1} \) (on
the right hand side).
The system of first order conditions (18) was solved using Mathematica with the built-in `NSolve` function. By substituting each solution into the polynomial $p_1(x_1, x_2, x_3, x_4)$ and selecting the real valued solutions, we end up with 11 real critical values of the polynomial $p_1(x_1, x_2, x_3, x_4)$:


The corresponding stationary points can be classified as a global minimizer, four local non-global minimizers, and six saddle points. The smallest one of these critical values yields the global minimum: $4.09516474435915$.

In an attempt to obtain similar information in an independent fashion, the matrix $A_{p_1}$ has been constructed explicitly in Mathematica using exact computation. This took 465 seconds. Then all the 2401 eigenvalues have been computed numerically in three ways: using the eigenproblem solver in Mathematica, the general eigenvalue solver `Eig` in Matlab and the iterative eigenvalue solver `Eigs` in Matlab. In Table 5 the results of these computations are collected.

Table 5

<table>
<thead>
<tr>
<th>Application</th>
<th>Method</th>
<th>Eigenvalue</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematica</td>
<td>solution of system (18)</td>
<td>$4.09516474435915$</td>
<td>$744$</td>
</tr>
<tr>
<td>Mathematica</td>
<td>construction of $A_{p_1}$ and computation of all eigenvalues</td>
<td>$4.09516474435915$</td>
<td>$465 + 315$</td>
</tr>
<tr>
<td>Matlab</td>
<td>construction of $A_{p_1}$ and computation of all eigenvalues</td>
<td>$4.09516474435924$</td>
<td>$465 + 176$</td>
</tr>
<tr>
<td>Matlab</td>
<td>construction of $A_{p_1}$ and computation of only the smallest real eigenvalue</td>
<td>$4.09516474360116$</td>
<td>$465 + 16$</td>
</tr>
</tbody>
</table>
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 four locations of the global minimum found with the above described ‘conventional’ methods, have been used as the starting point for a local search method. Using this local search method with a thirty digit working precision, the following coordinates for the global minimizer have been obtained:

\[
\begin{align*}
x_1 &= +0.876539213106233894587289929758 \\
x_2 &= -0.903966282304642050057296045914 \\
x_3 &= +0.862027936174326572650513966373 \\
x_4 &= -0.835187476756286528192781820247
\end{align*}
\] (19)

The corresponding criterion value of this point is computed as 4.095164744359157279770316678156. 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 \( p_1 \) using the nD-system implementation of the linear operator \( A_{p_1} \) to compute only its smallest real eigenvalue with an iterative eigenvalue solver (instead of working with the explicit matrix \( A_{p_1} \)). The coordinates of the global minimum were computed from the corresponding eigenvector (employing the Stetter vector structure). For this purpose the iterative eigenvalue solvers Jdqr, Jdqz (see Fokkema et al. (1998); Sleijpen and van der Vorst (1996)) and Eigs (see Lehoucq et al. (1998)) have been used. Jdqr is an iterative eigenvalue solver for Matlab and Jdqz is a generalized iterative eigenvalue solver for Matlab. Both methods employ Jacobi-Davidson methods. The method Eigs is a standard built-in Matlab routine which uses (restarted) Arnoldi-methods through ARPACK.

For a fast convergence of iterative eigenvalue methods, like Jdqr, Jdqz and Eigs, an appropriate preconditioner is needed. The choice was made for a simple preconditioner: the Jacobi-preconditioner (or diagonal preconditioner) was used, which builds the inverse of the diagonal of the matrix (or the linear operator) involved. This preconditioner is also used in a matrix-free fashion. Furthermore some balancing technique was used to balance the linear operator, see Chen and Demmel (2000). Balancing a matrix (or a linear operator) \( A \) means finding a similarity transform \( D^{-1}AD \), with \( D \) a diagonal matrix, such that, for each \( i \), row \( i \) and column \( i \) have (approximately) the same norm. The algorithms described in Chen and Demmel (2000) help to reduce the norm of the matrix by using methods of Perron-Frobenius and the direct iterative method.
In Table 6 the results of these computations are displayed. This concerns the method used, the minimal eigenvalue computed, the difference between this eigenvalue and the ‘true’ eigenvalue computed above with the local search method, the number of required operator actions during the computation, and the running time.

Table 6

<table>
<thead>
<tr>
<th>Method</th>
<th>Eigenvalue</th>
<th>Accuracy</th>
<th># Operator actions</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigs, no preconditioning</td>
<td>4.095164744055955</td>
<td>3.03 × 10⁻¹⁰</td>
<td>3040</td>
<td>173</td>
</tr>
<tr>
<td>Eigs, preconditioning</td>
<td>4.09516477853951</td>
<td>3.42 × 10⁻⁸</td>
<td>2472</td>
<td>142</td>
</tr>
<tr>
<td>Jdq, no preconditioning</td>
<td>4.0951647449668</td>
<td>1.38 × 10⁻¹⁰</td>
<td>312</td>
<td>21</td>
</tr>
<tr>
<td>Jdq, preconditioning</td>
<td>4.09516474564636</td>
<td>1.29 × 10⁻¹⁰</td>
<td>311</td>
<td>23</td>
</tr>
<tr>
<td>Jdqz, no preconditioning</td>
<td>4.09516474427060</td>
<td>8.86 × 10⁻¹¹</td>
<td>302</td>
<td>21</td>
</tr>
<tr>
<td>Jdqz, preconditioning</td>
<td>4.09516474483449</td>
<td>4.75 × 10⁻¹⁰</td>
<td>399</td>
<td>28</td>
</tr>
<tr>
<td>Eigs, no preconditioning, balanced</td>
<td>4.09516469968153</td>
<td>5.47 × 10⁻⁸</td>
<td>2752</td>
<td>162</td>
</tr>
<tr>
<td>Eigs, preconditioning, balanced</td>
<td>4.09516488133683</td>
<td>1.37 × 10⁻⁷</td>
<td>2872</td>
<td>175</td>
</tr>
<tr>
<td>Jdq, no preconditioning, balanced</td>
<td>4.09516473924322</td>
<td>5.12 × 10⁻⁹</td>
<td>421</td>
<td>28</td>
</tr>
<tr>
<td>Jdq, preconditioning, balanced</td>
<td>4.09516474814855</td>
<td>3.79 × 10⁻⁹</td>
<td>416</td>
<td>28</td>
</tr>
<tr>
<td>Jdqz, no preconditioning, balanced</td>
<td>4.09516473593991</td>
<td>8.42 × 10⁻⁹</td>
<td>389</td>
<td>28</td>
</tr>
<tr>
<td>Jdqz, preconditioning, balanced</td>
<td>4.09516473519265</td>
<td>9.17 × 10⁻⁹</td>
<td>489</td>
<td>35</td>
</tr>
</tbody>
</table>

The global minimum computed by the Jdqz method without preconditioning and without balancing produces the critical value that is the closest to the critical value computed by the local search method with high working precision. For these settings, the nD-systems approach uses less operator actions and is the fastest of all the methods in Table 6. Using the corresponding eigenvector, the coordinates of the stationary point corresponding to the minimal critical value are computed as:

\[
\begin{align*}
    x_1 &= +0.876539213107485 \\
    x_2 &= -0.90396628291641 \\
    x_3 &= +0.862027936168838 \\
    x_4 &= -0.835187476763094
\end{align*}
\]

Nowadays, for the problem of finding the global minimum of a polynomial there are several specialized software packages available, which employ different approaches. To put the performance of the nD-systems approach of this paper into perspective, we now briefly discuss the outcomes of the computation of the global minimum of polynomial \( p_1 \) given in (17) by the software packages SOSTOOLS, GloptiPoly and SYNAPS.

SOSTOOLS is a free Matlab toolbox for formulating and solving sum of squares (SOS) problems (see Prajna et al. (2004), Parrilo (2003)). This toolbox uses the Matlab solver SeDuMi (see Sturm (1999)) to solve the semi-definite programs (SDP) that are involved in this approach. For computing the global minimum, SOSTOOLS searches for the largest possible \( \gamma \) for which \( p_1(x_1, x_2, x_3, x_4) - \gamma \) is still a sum of squares. This \( \gamma \) may be the global minimum \( p_\star \) we are looking for, see (Parrilo and Sturmfels (2003)), depending
on whether the polynomial $p_1(x_1, x_2, x_3, x_4) - p_*$ can be written as a sum of squares of polynomials. Note that the nD-systems approach of the present paper does not suffer from such a limitation.

GloptiPoly (see Henrion and Lasserre (2003)) is a Matlab package that also uses the SeDuMi solver. GloptiPoly solves a multivariable polynomial optimization problem by building and solving convex linear matrix inequality (LMI) relaxations of the problem. The function produces a series of lower bounds which converge to the global optimum one is looking for. The theory of moments and positive polynomials is used in the implementation of this software (see Lasserre (2001) and Lasserre (2002) for further details).

SYNAPS (Reis et al. (2002)) is an environment for symbolic and numeric computations, mainly developed in C++. SYNAPS provides data structures for manipulating objects like matrices and polynomials. It also provides a routine to search for the real solutions of a polynomial system of equations within a given domain. Therefore it is possible to apply this function to our system of equations (18). All the solutions of this system, can be substituted into the polynomial (17) to find the global minimum.

The results of solving the global optimization problem for the polynomial $p_1$ using SOSTOOLS, GloptiPoly and SYNAPS are collected in Table 7.

<table>
<thead>
<tr>
<th>Method</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>Global Minimum</th>
<th>Accuracy</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOSTOOLS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.0951477401387</td>
<td>$2.97 \times 10^{-8}$</td>
<td>10</td>
</tr>
<tr>
<td>GloptiPoly</td>
<td>0.876535</td>
<td>-0.903963</td>
<td>0.862021</td>
<td>-0.835180</td>
<td>4.0951476247764</td>
<td>$1.81 \times 10^{-8}$</td>
<td>11</td>
</tr>
<tr>
<td>SYNAPS</td>
<td>0.876536</td>
<td>-0.903965</td>
<td>0.862026</td>
<td>-0.835184</td>
<td>4.0951474461324</td>
<td>$2.50 \times 10^{-10}$</td>
<td>2</td>
</tr>
</tbody>
</table>

When comparing the results of Table 6 to the results in Table 7, we see that the methods SOSTOOLS, GloptiPoly and SYNAPS are faster than the methods mentioned in table 6. Moreover, the built-in Matlab method Eigs performs bad: it uses very many operator actions and needs a lot of running time compared to these specialized software packages. But where the methods in table 7 appear to be faster, they are not as accurate as the iterative eigenvalue solvers which use the nD-systems approach described in this paper. Up till now, the methods Jdq$\alpha$ and Jd$qz$ perform the best for this optimization problem when accuracy is concerned. Furthermore these methods give us the possibility to tackle very large eigenvalue problems and to parallelize the iterative process of computing the eigenvalues of a linear operator in the future.

10 Conclusions and Discussion

The approach taken 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 $\lambda(x_1^{2d} + \ldots + x_n^{2d})$ with $\lambda > 0$, which may obvi-
ously be replaced by a dominating term of the form $\lambda_1 x_1^{2d} + \ldots + \lambda_n x_n^{2d}$ 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 $\lambda_1 x_1^{2d_1} + \ldots + \lambda_n x_n^{2d_n}$ 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 $q(x_1, \ldots, x_n) + \lambda \| (x_1, \ldots, x_n) \|^2_{2d}$. 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 $\lambda \| (x_1, \ldots, x_n) \|^2_{2d}$ 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 to deal 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 in Gröbner basis form with respect to a total degree ordering, by applying a suitable computer algebra algorithm such as the Buchberger algorithm. 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. They would however differ in the details. 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 computing and 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). In case 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 matrices $A_r$ and $A_r^T$ in case of one-dimensional eigenspaces has not yet been employed in the iterative methods of Arnoldi and Jacobi-Davidson, neither to speed up convergence nor 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.

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.

References


