

# Polynomial Optimization via Recursions and Parallel Computing

A.W.J. Bierfert

June 15, 2007

## Abstract

This paper presents a generalized technique for polynomial optimization based on an  $nD$  systems approach by Bleylevens, Peeters, and Hanzon. Aspects of the technique are researched by examination of an introductory example and later extended to a continuative example. A distributed computing framework is introduced which allows research of various different aspects of the examples.

## 1 Introduction

Even with today's advancements in computing power and parallel computing the problem of finding the global minimum of a real-valued multivariate polynomial is a hard problem. However it is a problem that has a wide range of applications not only in systems and control theory but also in other scientific fields like biology, chemistry, economics, and statistics to name a few examples. The approach taken by Bleylevens, Peeters, and Hanzon [2] is taken as a basis and one goal of this paper is to generalize it. Bleylevens, Peeters, and Hanzon use an  $nD$  systems approach to find the global minimum for a special class of dominated real valued polynomials. In this special class investigation of common properties of the class allows to gain information about the global minimum of the underlying real valued polynomial. However looking at polynomials in this special class is not optimal. This paper will investigate into a generalized approach of the  $nD$  method that is not limited by dominated real valued polynomials. The motivation is to avoid the sometimes costly computation of the Gröbner basis form and not restrict the polynomials to be part of a special class.

Another goal of this paper is to develop a parallelized framework for the computations which will be needed during the research to gain insight into larger polynomial optimization problems which would not be possible with monolithic computation applications as even small problems can result in time consuming calculations.

## 2 Algebraic Background

This section will give a brief overview of the algebraic background of system theory of  $nD$  systems for polynomial optimization [2] to understand the approach taken and the generalizations that will be investigated.

### 2.1 A $nD$ systems approach to polynomial optimization

Suppose  $p(x_1, \dots, x_n) \in \mathbb{R}[x_1, \dots, x_n]$  is a real polynomial. If we are interested in computing the *infimum* over  $\mathbb{R}^n$  it is known from [2] that if we consider the *Minkowski-norm dominated polynomial*

$$p_\lambda(x_1, \dots, x_n) := \lambda(x_1^{2d} + \dots + x_n^{2d}) + p(x_1, \dots, x_n) \quad (1)$$

with  $\lambda \in \mathbb{R}^+$  and  $2d > \deg(p(x_1, \dots, x_n))$ ,  $d \in \mathbb{N}$  the information about the minimum of  $p(x_1, \dots, x_n)$  can then be found by investigation of

$$\lim_{\lambda \rightarrow 0} p_\lambda. \quad (2)$$

The resulting system of first order conditions with  $n$  polynomial equations and  $n$  variables of the form

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

with  $m = 2d - 1$  and  $f^{(i)} = -\frac{1}{2d\lambda} \frac{\partial}{\partial x_i} p(x_1, \dots, x_n) \in \mathbb{R}[x_1, \dots, x_n]$  with the total degree less than  $m$ , is in Gröbner Basis form (see [3] and [1]).

The quotient space  $\mathbb{R}[x_1, \dots, x_n]/I$  of the associated ideal  $I = \langle d^{(i)} | i \in \{1, \dots, n\} \rangle$  generated by the polynomials is a finite dimensional vector space  $\dim(N) := m^n$ . The finite dimensionality allows for the application of the matrix method of Stetter and Möller [6] to compute the solutions of the system of equations  $d^{(i)}(x_1, \dots, x_n) = 0$ ,  $i \in \{1, \dots, n\}$ . This method reformulates the problem into an eigenvalue problem. The  $nD$  systems approach presented in [2] then allows to compute the eigenvalues via recursions without using the matrix.

To achieve polynomial optimization it is feasible to look at systems of equations as they are closely related to multivariate polynomials. Suppose  $f(x)$  is a multivariate function that needs to be optimized. This can be

done by solving  $\nabla f(x) = 0$ . It is also possible given a system of equations  $g(x) = 0$  to let  $g(x) = e$  and optimize  $e^T e = 0$ .

Having a given system of polynomials in Gröbner basis form gives a strong position on the solvability of the system. In general it is not always the case that systems are in Gröbner Basis form or that it is feasible to bring them into Gröbner Basis form.

### 3 Example: Daubechies 2 Wavelets

The system of equations that will be used as a first example is the Daubechies 2 wavelet (*db2*). It is a wavelet from the Daubechies orthogonal wavelet family [4]. The properties of the system of *db2* constraint equations are well known and researched. It is not in Gröbner Basis form and does not belong to the class  $p_\lambda$  of dominated polynomials. This makes the *db2* equations a feasible example.

#### 3.1 Theory

Daubechies wavelets are a subset of the orthogonal wavelet family [7]. Orthogonal wavelet theory imposes certain constraints on the structure of the equations to gain orthogonality. The constraints are described with the dilation equation and the wavelet equation:

1. dilation equation

$$\phi(t) = \phi(2t) + \phi(2t - 1) = \sum_{k \in \mathbb{Z}} c_k \sqrt{2} \phi(2t - k) \quad (4)$$

2. wavelet equation

$$\psi(t) = \phi(2t) - \phi(2t - 1) = \sum_{k \in \mathbb{Z}} d_k \sqrt{2} \phi(2t - k) \quad (5)$$

The first constraint on the wavelet is called *normalization* and is imposed by

$$\int \phi(t)^2 dt = 1 \Leftrightarrow \sum_{k \in \mathbb{Z}} c_k^2 = 1 \quad (6)$$

The same holds for  $d_k$  with  $\sum_{k \in \mathbb{Z}} d_k^2 = 1$ . The second condition is called *double-shift orthogonality*

$$\sum_{k \in \mathbb{Z}} c_k c_{k+2l} = 0, l \neq 0 \quad (7)$$

For the wavelet function we get

$$\sum_{k \in \mathbb{Z}} d_k d_{k+2l} = 0, l \neq 0 \quad (8)$$

From  $V_0 \perp W_0$  we get that  $\sum_{k \in \mathbb{Z}} c_k d_{k+2l} = 0, l \neq 0$ , from normalization  $\sum_{k \in \mathbb{Z}} c_k^2 = 1$ , and from double-shift

orthogonality  $\sum_{k \in \mathbb{Z}} c_k + c_{k+2l} = 0, l \neq 0$ . So  $d_k$  can be chosen via the *alternating flip construction* as

$$d_k = (-1)^k c_{n-k} \quad (9)$$

This leads to the *wavelet condition*

$$\int \psi(t) dt = 0 \Leftrightarrow \sum_{k \in \mathbb{Z}} (-1)^k c_k = 0 \quad (10)$$

The last condition is the so called *scaling function constraint*

$$\int \phi(t) dt = 1 \Leftrightarrow \sum_{k \in \mathbb{Z}} c_k = \sqrt{2} \quad (11)$$

If a wavelet satisfies these conditions it is said to be orthogonal.

For the Daubechies wavelets we will refer to two filters:

- *low-pass*

$$C(z) := \sum_{k \in \mathbb{Z}} c_k z^{-k} \quad (12)$$

- *high-pass*

$$D(z) := \sum_{k \in \mathbb{Z}} d_k z^{-k} \quad (13)$$

The characteristic of the Daubechies wavelet family is a maximum amount of vanishing moments. This can be satisfied by the constraint that  $C(z)$  has as many zeros at  $z^{-1} = -1$  as possible or

$$\int x^k \psi(x) dx = 0, k = 0, 1, \dots \quad (14)$$

#### *db2* wavelets

For the *db2* wavelet we get the following filter functions:

- *low-pass*

$$C(z) := \sum_{k=0}^3 c_k z^{-k} \Leftrightarrow c_0 + c_1 z^{-1} + c_2 z^{-2} + c_3 z^{-3} \quad (15)$$

- *high-pass*

$$D(z) := \sum_{k=0}^3 d_k z^{-k} \Leftrightarrow d_0 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3} \quad (16)$$

With the low-pass filter the constraints from orthogonal wavelet theory can be formulated as

- *normalization*

$$\sum_{k=0}^3 c_k^2 = 1 \Leftrightarrow c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1 \quad (17)$$

- double-shift orthogonality

$$\sum_{k=0}^3 c_k c_{k+2l} = 0, l \neq 0 \Leftrightarrow c_0 c_2 + c_1 c_3 = 0 \quad (18)$$

- wavelet condition

$$\sum_{k=0}^3 (-1)^k c_k = 0 \Leftrightarrow c_0 - c_1 + c_2 - c_3 = 0 \quad (19)$$

- scaling function constraint

$$\sum_{k=0}^3 c_k = \sqrt{2} \Leftrightarrow c_0 + c_1 + c_2 + c_3 = \sqrt{2} \quad (20)$$

One of these conditions can be left out as it is redundant. If they are satisfied we have an orthogonal wavelet. For the wavelet to become a *db2* wavelet one more condition is needed:

$$\int x^1 \psi(x) dx = 0 \Rightarrow c_1 - 2c_2 + 3c_3 = 0 \quad (21)$$

We will leave out the normalization condition here but any one of the four wavelet conditions could be left out because of redundancy though leaving out the normalization condition as opposed to the scaling function condition can give less solutions because of the quadratic nature of the normalization condition. The resulting system of equations  $\Sigma$  can be written as:

$$\Sigma := \begin{cases} c_0 c_2 + c_1 c_3 = 0 \\ c_0 - c_1 + c_2 - c_3 = 0 \\ c_0 + c_1 + c_2 + c_3 - \sqrt{2} = 0 \\ c_1 - 2c_2 + 3c_3 = 0 \end{cases} \quad (22)$$

Solving  $\Sigma$  results in the following solutions:

variable	solution 1	solution 2
$c_0$	$\frac{1}{8}(\sqrt{2} - \sqrt{6})$	$\frac{1}{8}(\sqrt{2} + \sqrt{6})$
$c_1$	$\frac{1}{8}(3\sqrt{2} - \sqrt{6})$	$\frac{1}{8}(3\sqrt{2} + \sqrt{6})$
$c_2$	$\frac{1}{8}(\sqrt{2} + \sqrt{6})$	$\frac{1}{8}(\sqrt{2} - \sqrt{6})$
$c_3$	$\frac{1}{8}(3\sqrt{2} + \sqrt{6})$	$\frac{1}{8}(3\sqrt{2} - \sqrt{6})$

The system  $\Sigma$  is not in Gröbner Basis form but can be brought into Gröbner Basis form. One possible Gröbner Basis is:

$$\Sigma_{gb} := \begin{cases} 1 + 4\sqrt{2}c_3 - 16c_3^2 = 0 \\ \sqrt{2} - 2c_1 - 2c_3 = 0 \\ -\sqrt{2} + 4c_2 - 4c_3 = 0 \\ \sqrt{2} + 4c_0 - 4c_3 = 0 \end{cases}$$

For this Gröbner Basis also holds that  $\Sigma_{gb}$  has the same solutions as  $\Sigma$  or that  $\Sigma_{gb}(c_0, c_1, c_2, c_3) = \underline{0}$  for

the solutions of  $\Sigma$ .  $\Sigma$  will be used to generate the autonomous multidimensional system (*nD* system) instead of the Gröbner basis  $\Sigma_{gb}$ .

Let  $\sigma_0^{\alpha_0} \sigma_1^{\alpha_1} \dots \sigma_n^{\alpha_n}$  be a shift operator associated with a monomial of the form  $c_0^{\alpha_0} c_1^{\alpha_1} \dots c_n^{\alpha_n}$ . Given a multidimensional time series  $y_{t_0, t_1, \dots, t_n}$  application of the shift operator is defined as:

$$\sigma_0^{\alpha_0} \sigma_1^{\alpha_1} \dots \sigma_n^{\alpha_n} : y_{t_0, t_1, \dots, t_n} \mapsto y_{t_0 + \alpha_0, t_1 + \alpha_1, \dots, t_n + \alpha_n} \quad (23)$$

Let  $q(c_0, c_1, \dots, c_n)$  be any polynomial. The shift operator  $q$  can be associated with a homogeneous multidimensional difference equation as follows

$$q(\sigma_1, \sigma_2, \dots, \sigma_n) y_{t_0, t_1, \dots, t_n} = 0 \quad (24)$$

If this approach is applied to  $\Sigma$  the following system of four dimensional difference equations (recursions) is acquired:

$$\Sigma_r := \begin{cases} y_{t_0+1, t_1, t_2+1, t_3} + y_{t_0, t_1+1, t_2, t_3+1} = 0 \\ y_{t_0+1, t_1, t_2, t_3} - y_{t_0, t_1+1, t_2, t_3} + y_{t_0, t_1, t_2+1, t_3} - y_{t_0, t_1, t_2, t_3+1} = 0 \\ y_{t_0+1, t_1, t_2, t_3} + y_{t_0, t_1+1, t_2, t_3} + y_{t_0, t_1, t_2+1, t_3} + y_{t_0, t_1, t_2, t_3+1} - \sqrt{2} y_{t_0, t_1, t_2, t_3} = 0 \\ y_{t_0, t_1+1, t_2, t_3} - 2y_{t_0, t_1, t_2+1, t_3} + 3y_{t_0, t_1, t_2, t_3+1} = 0 \end{cases} \quad (25)$$

### 3.2 Experiments and Results

The first question that needs to be answered in this context is if it is possible to calculate all points of a given simplex of a total degree if not all points of the simplex are present. Especially interesting is the question if a set with a minimal amount of points exists. The structure of the Gröbner basis of the *db2* equations  $\Sigma_{gb}$  suggests that an initial state or possibly multiple initial states  $w_{0,0,0,0}$  exist with  $|w_{0,0,0,0}| = 1 \cdot 1 \cdot 1 \cdot 2 = 2$  which have this property. This is because of the three linear equations from  $\Sigma_{gb}$  which e.g. can be solved for  $c_3$  and the quadratic one which can then be used to determine a solution for the whole system. Initial states  $w$  with these properties will be referred to as *starting sets*  $\mathbb{S}$ .

A systematic method exists to find the starting sets for a given simplex with total degree  $d$  and initial state  $M$  with  $|M| = m$ . The first approach tries to expand on the  $M$  starting points with the given recursions and tries to fill the whole hyper plane. This however is not very efficient as every time a new point is found all possible combinations for the known set need to be tested together with all recursions which grows exponentially and is not feasible for large  $d$  and  $m$ .

A second approach is to take all  $|P| = n$  points from the given simplex with degree  $d$  except the points that are in  $M$  and look through the points in  $P$  to see if they can be calculated with the points in  $M$ . If a point can be calculated it is taken from  $P$  and added to  $M$ . If the set  $M$  is really a starting set then each run one point is removed from  $P$  and added to  $M$ . In the *db2* case this results in a worst case complexity of  $n * 3$  in the first round as three recursions relate points in the same simplex. This leads to the following proposition:

**Proposition 1** *The worst case complexity of the starting set algorithm for the db2 recursions for a simplex with total degree  $n$  is:*

$$O\left(3 \sum_{i=1}^n i\right) = O\left(3 * \frac{n(n+1)}{2}\right) \in O(n^2)$$

Suppose  $\mathbb{L}_6$  is the set of integer solutions to the equation  $t_0 + t_1 + t_2 + t_3 = 6$  with  $|\mathbb{L}_6| = 80$ . Let  $w$  be an initial state which contains  $y_{t_0, t_1, t_2, t_3}$  with  $(t_0, t_1, t_2, t_3) \in \mathbb{L}_6$ . Which are the initial states for this simplex?

Calculations show that three unique initial states of size two with total degree six exist:

$$\begin{aligned} \mathbb{S}_1 &:= \{y_{0,0,5,1}, y_{0,0,6,0}\} \\ \mathbb{S}_2 &:= \{y_{0,0,5,1}, y_{0,1,5,0}\} \\ \mathbb{S}_3 &:= \{y_{0,0,6,0}, y_{0,1,5,0}\} \end{aligned} \tag{26}$$

These starting sets are the minimal starting sets for the *db2* recursions for total degree six. If recursions one, two, and four from  $\Sigma_r$  are applied the whole simplex with total degree six can be filled. A three dimensional projection w.r.t.  $t_1, t_2, t_3$  of starting set  $\mathbb{S}_1$  is shown in Figure 1.

A projection of the filled simplex of total degree six can be found in Figure 2.

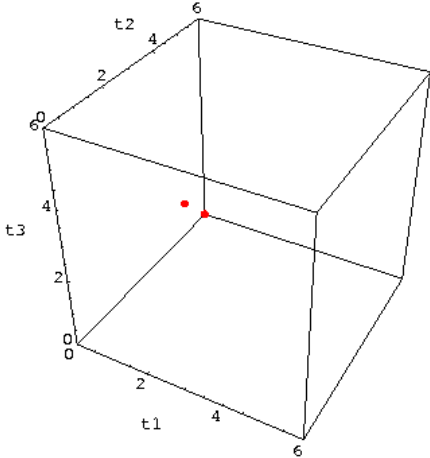


Figure 1: Starting set  $\mathbb{S}_1$

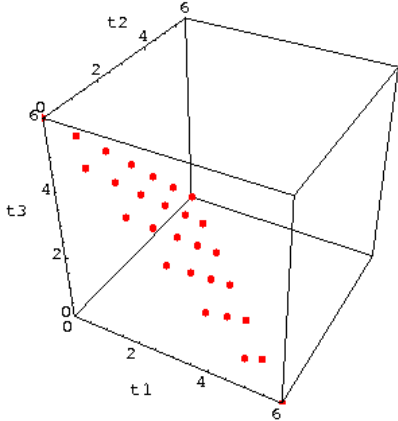


Figure 2: Filled simplex of total degree six after downward expansion with  $\mathbb{S}_1$  and recursions one, two, and four from  $\Sigma_r$ .

It is clear now that the whole simplex can be computed. The question is if the simplex with total degree five can be computed by using the values from simplex six and recursion three from  $\Sigma_r$ .

Calculations show that all values with the property  $t_0 + t_1 + t_2 + t_3 = 5$ ,  $|\mathbb{L}_5| = 56$  can be computed by just utilizing the remaining recursion. This means that with a starting set of size two not only the simplex with corresponding total degree can be computed but also the simplex below it (see Figure 3).

Further investigation showed that this is possible for all

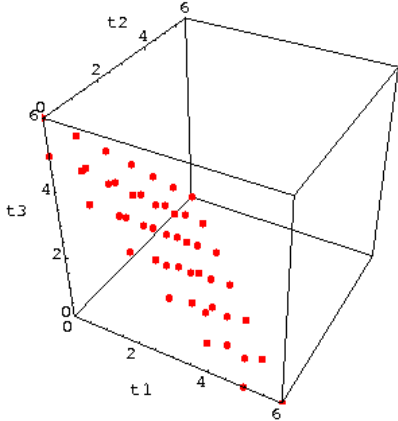


Figure 3: Hyperplane of total degree five and six computed by expansion of  $\mathbb{S}_1$  with the *db2* recursions.

starting sets  $\mathbb{S}_i, i \in \{1, 2, 3\}$ . This method can of course be applied to the expanded values from the simplex with total degree five and used to compute the simplex with total degree four. Expansion till the origin can be seen in Figure 4. This is summarized in the following propo-

sition.

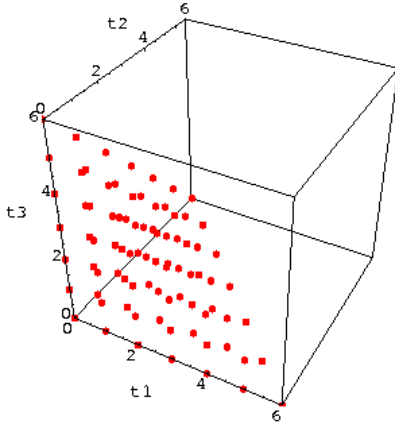


Figure 4: Expansion of  $\mathbb{S}_1$  with the *db2* recursions downwards to the origin.

**Proposition 2** *A starting set of general structure  $\mathbb{S}_i$ ,  $i \in \{1, 2, 3\}$  allows for the calculation of all values in the corresponding simplex and all simplexes with lower degree including the origin.*

From these results some more questions arise: Is there an efficient way to reach a certain simplex or certain points or the origin? This question is rather important. The above experiments show that in general it is possible to reach the origin with a given starting set of a given total degree. For this a breadth-first approach was taken. For real world calculations this is rather impractical as the number of values  $v$  in a simplex increases exponentially according to the formula

$$v := \frac{(n + d - 1)!}{(d - 1)!(n)!}$$

where  $n$  is the total degree of the simplex and  $d$  is the degree of the polynomial (4 for the *db2* case). This means that for the *db2* recursions there are  $\binom{v}{2}$  possible starting sets for a given simplex [5]. Better would be a depth-first approach to calculate values in a lower plane and allow for more selective calculations. The problem at hand is a shortest path problem of the same nature as discussed in [2].

An investigation of the *db2* recursions shows another property which solves this problem: Suppose  $\mathbb{S}_1$  is the given starting set for total degree six then  $\{y_{0,0,4,1}, y_{0,0,5,0}\}$  is a starting set for total degree five. In general this holds for all *db2* starting sets of size 2 with  $n \in \mathbb{N}$ :

$$\begin{aligned} & \{y_{0,0,n-1,1}, y_{0,0,n,0}\} \\ & \{y_{0,0,n-1,1}, y_{0,1,n-1,0}\} \\ & \{y_{0,0,n,0}, y_{0,1,n-1,0}\} \end{aligned} \quad (27)$$

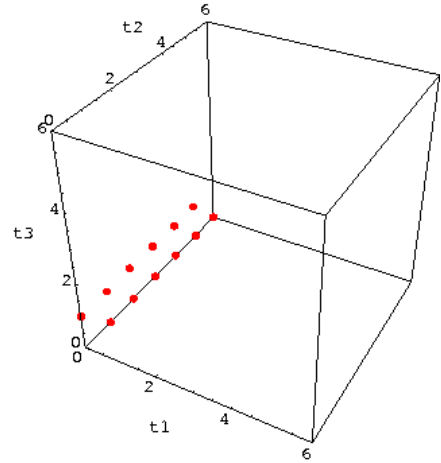


Figure 5: Three dimensional projection of the starting sets with characteristic  $\{y_{0,0,n-1,1}, y_{0,0,n,0}\}$  for  $n \in \{1, \dots, 6\}$ .

The next question is how this property can be used to derive a depth-first search. Consider the starting set  $\mathbb{S}_1$ . Investigation shows that a shortest path with five nodes exists between the starting set and the starting set in the next lower plane. This is an unique optimal path as all values that are computed are needed and need to be computed. This means that all other possible paths to arrive at the starting set in the lower plane are longer as they also need to compute these values in addition to more ones. The shortest path is documented below and how it can be obtained by applying the *db2* recursions (recursion numbers are according to  $\Sigma_r$ ). A graphical representation can be seen in Figures 6 and 7.

$$\begin{aligned} & \{y_{0,0,5,1}, y_{0,0,6,0}\} \xrightarrow{4} y_{0,1,5,0} \xrightarrow{3} y_{1,0,5,0} \xrightarrow{1} y_{0,1,4,1} \\ & \xrightarrow{4} y_{0,0,4,2} \xrightarrow{2} y_{1,0,4,1} \xrightarrow{3,3} \{y_{0,0,4,1}, y_{0,0,5,0}\} \end{aligned}$$

A shortest path with five intermediate nodes also exists for starting set  $\mathbb{S}_3$ . The above is summarized in the following propositions:

**Proposition 3** *All starting sets that follow the general structure of  $\mathbb{S}_1$  and  $\mathbb{S}_3$  have a shortest path with five intermediate nodes.*

**Proposition 4** *All starting sets that follow the general structure of  $\mathbb{S}_2$  have a shortest path with seven intermediate nodes.*

This means that this depth-first algorithm derived from the starting sets unique shortest path properties has a linear complexity: Five intermediate points plus two points of the lower starting set to get to the next lower starting set means a complexity of  $O(7)$ . If started at a

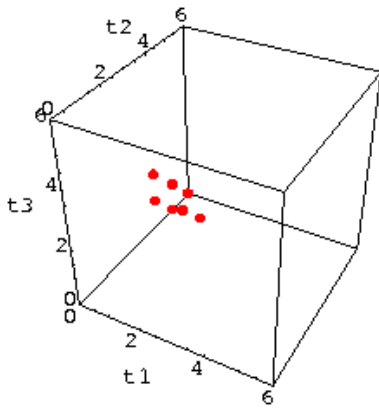


Figure 6: Unique shortest path for starting set  $S_1$  from simplex with total degree 6 to the starting set  $\{y_{0,0,4,1}, y_{0,0,5,0}\}$  of the simplex of total degree 5.

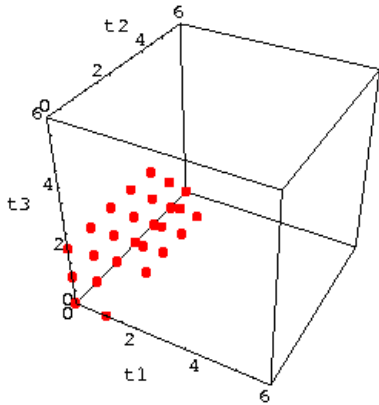


Figure 7: Unique shortest path for starting set  $S_1$  from simplex with total degree 6 to the origin.

simplex of total degree  $n$  to compute the values of the origin a complexity of  $O((5 + 2) * n) \in O(n)$  emerges.

**Proposition 5** *The depth-first algorithm for the db2 starting sets  $S_i, i \in \{1, \dots, 3\}$  has a complexity of  $O(n)$ .*

The next question that needs to be answered is if this set of recursions is consistent meaning that all possible paths to reach one point give the same value. Experiments with randomized path calculations have been conducted to see if all values are the same. The experiments suggest that this set of recursions is consistent. This is expressed by the following conjecture:

**Conjecture 1** *The db2 recursions are consistent and converge towards solutions of the system  $\Sigma$ .*

This means that the recursions derived from the db2 equations in fact work like the inverse power method [8]. The power method is an eigenvalue solver that uses iteration to find the largest dominant eigenvalue and its

corresponding eigenvector. Suppose  $b_0$  is a random vector or the approximation of the dominant eigenvector, then the power method will generate an eigenvalue  $\lambda$  and an eigenvector  $v$  such that  $Av = \lambda v$  by iteration of

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$$

As no matrix decomposition is needed for this method it is useful for large matrix problems. Derived from the power method is the inverse power method which instead of approximating the largest eigenvalue approximates the smallest eigenvalue by calculating the largest reciprocal eigenvalue  $\lambda^{-1}$ .

Given the starting set  $S_1$  with associated values (1,1) calculations to the origin show the values for the simplex with total degree one and the origin in Table 1.

point	value
$y_{1,0,0,0}$	$\frac{5120\sqrt{2}}{9}$
$y_{0,1,0,0}$	$\frac{26624\sqrt{2}}{27}$
$y_{0,0,1,0}$	$\frac{7168\sqrt{2}}{27}$
$y_{0,0,0,1}$	$-\frac{4096\sqrt{2}}{27}$
$y_{0,0,0,0}$	$\frac{45056}{27}$

Table 1: Calculations for starting set  $S_1$  with values (1, 1) from the simplex with total degree six to the origin.

To normalize the eigenvector everything is divided by the value of  $y_{0,0,0,0}$  and approximate solutions for  $c_0, c_1, c_2, c_3$  from  $\Sigma$  can be obtained. This means that this set of recursions converges towards the solution of the db2 equations. Convergence can be increased by starting calculations on a simplex farther away from the origin. Table 2 shows a number of calculations and their accuracy.

One property of the power method is that it converges for arbitrary starting values. This set of recursions also converges for arbitrary starting values though starting values which approximate the solutions will converge faster.

During the experiments another property of the recursions was discovered. Suppose  $S_1$  with values (1, 1) is given. If the value for  $y_{2,0,0,0}$  is computed the following relation holds:  $y_{2,0,0,0}$  can be considered as being  $c_0^2$  and  $y_{1,0,0,0}$  being  $c_0$  then dividing  $y_{2,0,0,0}$  by  $y_{1,0,0,0}$  should also give a value for  $c_0$ . In this example this yields a result of 0.48318963381080747501 which is indeed an approximate value of  $c_0$ . This relation also holds for  $y_{0,2,0,0}$  and  $y_{0,1,0,0}$  and so on. Further investigation shows that

$n$	variable	value	absolute error
exact	$c_0$	$\frac{1}{8}(\sqrt{2} + \sqrt{6})$	
	$c_1$	$\frac{1}{8}(3\sqrt{2} + \sqrt{6})$	
	$c_2$	$\frac{1}{8}(\sqrt{2} - \sqrt{6})$	
	$c_3$	$\frac{1}{8}(3\sqrt{2} - \sqrt{6})$	
6	$c_0$	$\frac{15}{22\sqrt{2}}$	0.0008446532468
	$c_1$	$\frac{13}{11\sqrt{2}}$	0.0008446532424
	$c_2$	$\frac{7}{22\sqrt{2}}$	0.0008446532136
	$c_3$	$-\frac{\sqrt{2}}{11}$	0.0008446532757
20	$c_0$	$\frac{413403}{605264\sqrt{2}}$	$2.228767074 * 10^{-12}$
	$c_1$	$\frac{1240209}{1048348\sqrt{2}}$	$2.228767074 * 10^{-12}$
	$c_2$	$\frac{89043}{280904\sqrt{2}}$	$3.104272076 * 10^{-11}$
	$c_3$	$-\frac{29681}{162180\sqrt{2}}$	$3.104272076 * 10^{-11}$

Table 2: Convergence table for the  $db2$  recursions.  $n$  is the total degree of the simplex.

these results are less accurate than calculations downwards to the origin. This relationship is also present for all simplexes such that e.g.  $y_{n,0,0,0}$  divided by  $y_{n-1,0,0,0}$  gives an approximate value for  $c_0$  and so on.

The last question that is remaining is why this system converges against one solution of the  $db2$  equations. As known from [2] with the Gröbner Basis it can be selected to move into one of the four directions to select the dominant eigenvalue of the current direction. In the case of the recursions from  $\Sigma_r$  the selection is done by the starting sets  $\mathbb{S}_i, i \in \{1, 2, 3\}$ . The calculations downwards to the origin move along the  $c_2$  axis. This way the solution is an eigenvector to the dominant eigenvalue of  $c_2$ .

## 4 Example: Daubechies 3 Wavelets

After obtaining the results from the previous chapter about the Daubechies 2 wavelets, the question on how these results apply to the Daubechies 3 ( $db3$ ) wavelet equations was imminent.

### 4.1 Theory

The system  $\Gamma$  of  $db3$  equations can be derived w.r.t. the introduced wavelet theory as:

$$\Gamma := \begin{cases} c_0c_2 + c_1c_3 + c_2c_4 + c_3c_5 = 0 \\ c_0c_4 + c_1c_5 = 0 \\ c_0 - c_1 + c_2 - c_3 + c_4 - c_5 = 0 \\ c_0 + c_1 + c_2 + c_3 + c_4 + c_5 - \sqrt{2} = 0 \\ c_1 - 2c_2 + 3c_3 - 4c_4 + 5c_5 = 0 \\ c_2 - 3c_3 + 6c_4 - 10c_5 = 0 \end{cases} \quad (28)$$

The normalization condition has been left out because of redundancy. If  $\Gamma$  is solved four solutions exist. The complex solutions can be discarded as only real polynomials are discussed. The exact solutions are:

$$\begin{aligned} c_0 &= \frac{1}{32}(\sqrt{2} + 2\sqrt{5} + \sqrt{10 + 4\sqrt{10}}) \\ c_1 &= \frac{1}{32}(5\sqrt{2} + 2\sqrt{5} + 3\sqrt{10 + 4\sqrt{10}}) \\ c_2 &= \frac{1}{16}(5\sqrt{2} - 2\sqrt{5} + \sqrt{10 + 4\sqrt{10}}) \\ c_3 &= \frac{1}{16}(5\sqrt{2} - 2\sqrt{5} - \sqrt{10 + 4\sqrt{10}}) \\ c_4 &= \frac{1}{32}(5\sqrt{2} + 2\sqrt{5} - 3\sqrt{10 + 4\sqrt{10}}) \\ c_5 &= \frac{1}{32}(\sqrt{2} + 2\sqrt{5} - \sqrt{10 + 4\sqrt{10}}) \end{aligned}$$

or

$$\begin{aligned} c_0 &= \frac{1}{32}(\sqrt{2} + 2\sqrt{5} - \sqrt{10 + 4\sqrt{10}}) \\ c_1 &= \frac{1}{32}(5\sqrt{2} + 2\sqrt{5} - 3\sqrt{10 + 4\sqrt{10}}) \\ c_2 &= \frac{1}{16}(5\sqrt{2} - 2\sqrt{5} - \sqrt{10 + 4\sqrt{10}}) \\ c_3 &= \frac{1}{16}(5\sqrt{2} - 2\sqrt{5} + \sqrt{10 + 4\sqrt{10}}) \\ c_4 &= \frac{1}{32}(5\sqrt{2} + 2\sqrt{5} + 3\sqrt{10 + 4\sqrt{10}}) \\ c_5 &= \frac{1}{32}(\sqrt{2} + 2\sqrt{5} + \sqrt{10 + 4\sqrt{10}}) \end{aligned}$$

The system  $\Gamma$  is not in Gröbner Basis form but a possible Gröbner Basis  $\Gamma_{gb}$  exists which has the same solutions as  $\Gamma$ :

$$\Gamma_{gb} := \begin{cases} 9 - 96\sqrt{2}c_5 - 3072c_5^2 - 8192\sqrt{2}c_5^3 + 65536c_5^4 = 0 \\ 3c_4 + 6c_5 + 32\sqrt{2}c_5^2 - 256c_5^3 = 0 \\ -3\sqrt{2} + 24c_3 + 192c_5 + 512\sqrt{2}c_5^2 - 4096c_5^3 = 0 \\ 9\sqrt{2} - 24c_1 + 168c_5 + 512\sqrt{2}c_5^2 - 4096c_5^3 = 0 \\ -3\sqrt{2} + 8c_2 + 16c_5 = 0 \\ 3\sqrt{2} - 24c_0 + 96c_5 + 256\sqrt{2}c_5^2 - 2048c_5^3 = 0 \end{cases}$$

If the shift operator from the previous section is applied to  $\Gamma$  a set of six dimensional difference equations (recur-

sions) is acquired:

$$\Gamma_r := \left\{ \begin{array}{l} y_{t_0+1,t_1,t_2+1,t_3,t_4,t_5} + y_{t_0,t_1+1,t_2,t_3+1,t_4,t_5} + \\ y_{t_0,t_1,t_2+1,t_3,t_4+1,t_5} + y_{t_0,t_1,t_2,t_3+1,t_4,t_5+1} = 0 \\ \\ y_{t_0+1,t_1,t_2,t_3,t_4+1,t_5} + y_{t_0,t_1+1,t_2,t_3,t_4,t_5+1} = 0 \\ \\ y_{t_0+1,t_1,t_2,t_3,t_4,t_5} - y_{t_0,t_1+1,t_2,t_3,t_4,t_5} + \\ y_{t_0,t_1,t_2+1,t_3,t_4,t_5} - y_{t_0,t_1,t_2,t_3+1,t_4,t_5} + \\ y_{t_0,t_1,t_2,t_3,t_4+1,t_5} - y_{t_0,t_1,t_2,t_3,t_4,t_5+1} = 0 \\ \\ y_{t_0+1,t_1,t_1,t_2,t_3,t_4,t_5} + y_{t_0,t_1+1,t_2,t_3,t_4,t_5} + \\ y_{t_0,t_1,t_2+1,t_3,t_4,t_5} + y_{t_0,t_1,t_2,t_3+1,t_4,t_5} + \\ y_{t_0,t_1,t_2,t_3,t_4+1,t_5} + y_{t_0,t_1,t_2,t_3,t_4,t_5+1} - \\ \sqrt{2}y_{t_0,t_1,t_2,t_3,t_4,t_5} = 0 \\ \\ y_{t_0,t_1+1,t_2,t_3,t_4,t_5} - 2y_{t_0,t_1,t_2+1,t_3,t_4,t_5} \\ + 3y_{t_0,t_1,t_2,t_3+1,t_4,t_5} - 4y_{t_0,t_1,t_2,t_3,t_4+1,t_5} \\ + 5y_{t_0,t_1,t_2,t_3,t_4,t_5+1} = 0 \\ \\ y_{t_0,t_1,t_2+1,t_3,t_4,t_5} - 3y_{t_0,t_1,t_2,t_3+1,t_4,t_5} \\ + 6y_{t_0,t_1,t_2,t_3,t_4+1,t_5} - 10y_{t_0,t_1,t_2,t_3,t_4,t_5+1} = 0 \end{array} \right. \quad (29)$$

## 4.2 Experiments and Results

As with the *db2* equations the first research was conducted to find possible minimal starting sets. The Gröbner Basis equations from  $\Gamma_{gb}$  suggest a minimal initial state  $w_{0,0,0,0,0,0}$  with size  $|w_{0,0,0,0,0,0}| = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 2 \cdot 2 = 4$ . The *db2* starting set algorithm was extended to find possible starting sets for the *db3* recursions by adjusting for calculations with the *db3* recursions. This results in the following worst case complexity:

**Proposition 6** *The worst case complexity of the starting set algorithm for the *db3* recursions for a simplex with total degree  $n$  is:*

$$O\left(5 \sum_{i=1}^n i\right) = O\left(5 * \frac{n(n+1)}{2}\right) \in O(n^2)$$

The calculations show that for the *db3* recursions the minimal initial state set size varies between simplexes with a different total degree. For the simplex with total degree one four different starting sets of size three exist:

$$\begin{aligned} \mathbb{T}_1 &:= \{y_{0,0,0,0,0,1}, y_{0,0,0,0,1,0}, y_{0,0,1,0,0,0}\} \\ \mathbb{T}_2 &:= \{y_{0,0,0,0,0,1}, y_{0,0,0,0,1,0}, y_{0,0,0,1,0,0}\} \\ \mathbb{T}_3 &:= \{y_{0,0,0,0,0,1}, y_{0,0,0,1,0,0}, y_{0,0,1,0,0,0}\} \\ \mathbb{T}_4 &:= \{y_{0,0,0,0,1,0}, y_{0,0,0,1,0,0}, y_{0,0,1,0,0,0}\} \end{aligned} \quad (30)$$

For total degree two the smallest starting set size is five. Further experiments show that the minimal starting set size increases by two for each simplex till at least total degree five. This leads to the following proposition:

**Proposition 7** *For each simplex with total degree  $n$  a starting set of size  $2n + 1$  exists which in itself is a minimal set.*

As discussed in the *db2* section a breadth-first approach to go from one simplex to the next is not feasible for real world calculations as well as looking for starting sets with a high total degree. To check convergence a relation between starting sets on simplexes with different total degree is needed. Experiments show that such relations known from the *db2* starting sets also exists for the *db3* recursions.

Figure 8 shows such an expansion from the starting set

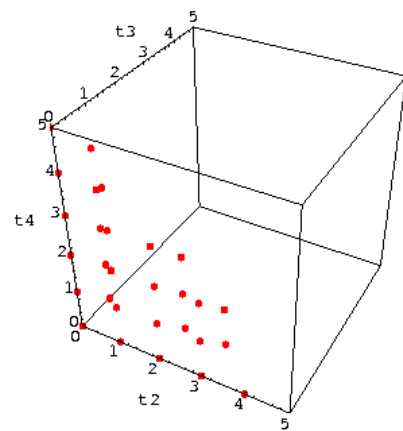


Figure 8: Projection of a possible starting set expansion from starting set  $\mathbb{T}_4$  w.r.t.  $t_i, i \in \{2, 3, 4\}$  from total degree zero to five.

$\mathbb{T}_4$  into the direction of  $c_4$ . What is remarkable in the *db3* case is that it is possible to expand into different directions.

**Proposition 8** *It is possible to select a variable  $c_i, i \in \{2, \dots, 5\}$  from a given *db3* starting set to generate starting sets of simplexes with higher or lower total degree w.r.t. to the chosen direction.*

In most cases various forms of expansions for one direction are possible making it inefficient to specify a generic shortest path problem like it is possible in the *db2* case. Table 3 shows expansions from  $\mathbb{T}_1$  in all possible directions.

This discovery is important when looking at consistency and convergence of the *db3* recursions. Calculations with different total degrees and different starting set expansions and into different directions show inconsistent results. This proves the following proposition:

**Proposition 9** *The *db3* recursions are inconsistent and do not converge towards solutions of the system  $\Gamma$ .*

axis	starting sets of degree $n$ derived from $\mathbb{T}_1$
$c_2$	$\{y_{0,0,1,0,0,1}, y_{0,0,1,0,1,0}, y_{0,0,2,0,0,0}, y_{0,0,0,0,2,0}, y_{0,0,0,1,1,0}\}$
$c_3$	$\{y_{0,0,0,1,0,1}, y_{0,0,0,1,1,0}, y_{0,0,1,1,0,0}, y_{0,0,0,0,2,0}, y_{0,0,0,0,1,1}\}$
$c_4$	$\{y_{0,0,0,0,1,1}, y_{0,0,0,0,2,0}, y_{0,0,1,0,1,0}, y_{0,0,1,1,0,0}, y_{0,0,2,0,0,0}\}$
$c_5$	$\{y_{0,0,0,0,0,2}, y_{0,0,0,0,1,1}, y_{0,0,1,0,0,1}, y_{0,0,0,2,0,0}, y_{0,0,1,1,0,0}\}$

Table 3: Possible starting sets of total degree two for directions  $c_i, i \in \{2, \dots, 5\}$  derived from starting set  $\mathbb{T}_1$ .

## 5 Parallel and Distributed Computing

One goal of this project was to develop a parallel computing framework which allows for the huge computations that were needed to be computed in reasonable time. An example for a time consuming problem is the search for valid starting sets. While the algorithm for one starting set belongs to the complexity class  $O(n)$  the number of sets increases exponentially. Suppose again that a simplex of total degree  $n$  has  $v := \frac{(n+d-1)!}{(d-1)!(n)!}$  values. To find all starting sets of total degree 6 for the  $db2$  equations a total of  $\binom{8^4}{2} = 3486$  sets need to be checked. This task can be computed by a modern computer in a couple of seconds. However for the  $db3$  equations this situation changes rapidly. For simplexes with a rather small degree huge amounts of sets need to be checked. The number of sets for a simplex with a total degree four and a set size four in the  $db3$  case amounts to  $\binom{126}{4} = 10009125$  possible starting sets. On a moderate personal computer with a test rate of six sets per second this amounts to more than 16 days of raw computing time. The nature of this problem however does not give any room for big optimizations. Two concepts are presented in the next sections which can improve the overall performance.

### 5.1 Threading

One possibility to improve performance is to utilize threading. Threading is a technique which allows processes to split themselves into two or more simultaneously running tasks which can access the same resources. On single CPU machines threading does in most cases only improve the performance if a lot of I/O operations are involved like e.g. for compiling source code. The trend with modern CPUs however is to offer multiple CPU cores in one casing. Computers used in research usually also utilize multiple CPU cores which makes threading a cheap way of parallelizing computations. Most programming languages and operation systems provide easy

support for threading.

For the starting set problem this can improve computation time because the calculations for each possible starting set are independent from each other. Thus in theory the calculation time  $t$  could be shortened by the number of CPUs as  $\frac{t}{\#CPUs}$ .

### 5.2 Distribution

Distributed computing is another approach to achieve performance gains for complex calculations. The idea is that tasks are sent over a network connection to client nodes which compute the task and return the results to a central server which collects and evaluates the results. This solution has some benefits over threading. Especially in the past computers with multiple CPU Cores were quite expensive whereas sharing tasks over a network became cheap. The increase of processing power on normal home computer systems also leads to more and more deployments of distributed computing. On the downside speed improvements to the starting set problem are hard to measure as the single tasks are easy to compute and thus a fast and solid network connection between the nodes and the server is required to achieve a good performance gain.

### 5.3 Distribution Framework

Threading as well as distributed computing does provide unique possibilities to increase performance of complex calculations. As part of this thesis a distribution framework which combines both methods has been developed and successfully tested on some of the calculations. The main server application is used to keep track of the task that is being calculated. It displays various statistics about the active task, the connected clients, and about progression of the clients and the task. It also allows for easy submission of new tasks by viewing all possible options.

The client interface is completely console based. The idea is that the control application distributes the tasks to the idle clients and the client computes the subtasks and reports back once all subtasks have been computed. The client hereby uses threading as appropriate for the amount of CPUs to compute multiple tasks at a time. A scheduler algorithm in the distribution server distributes tasks such that 'fast' clients get more work while 'slower' clients get less subtasks to calculate. This way an optimal performance can be reached. Depending on the computational complexity of the subtasks some adjustments and tune-ups need to be made to achieve optimal performance as cheap calculations would need a lot of network traffic while sending multiple subtasks to prevent this could result in unwanted behavior for computational complex subtasks.

This solution however proved to be valuable as the com-

puting resources especially on a single computer scale were limited.

## 6 Conclusions

The research to gain insight into a new approach for polynomial optimization has produced some interesting results. Especially the research on the  $db2$  equations has shown some significant results.

The structure discovered in the  $db2$  starting sets which can be combined with a shortest path depth first search with a computational complexity of  $O(n)$  allows for arbitrary precise computations of a solution to the system of  $db2$  equations in a feasible timeframe.

Applying these results to the  $db3$  equations has also proven to be a valuable step. One example is the general structure of the starting sets which partly can be explained with the Daubechies wavelet condition of vanishing moments. The starting sets of the  $db2$  recursions have the  $t_0$  coordinate always set to zero, the  $db3$  recursions always have the first two coordinates  $t_0$  and  $t_1$  set to zero. This is just one aspect where  $db2$  and  $db3$  have some similar properties. The minor adjustments to the research tools used also shows that analyzing the  $db2$  example made exploring the  $db3$  example possible and easier because both systems of equations share a lot of common properties.

The  $db3$  recursions are not consistent and thus it is not possible to apply the shortest-path depth-first to get convergence towards solutions of the  $db3$  equations.

A lot of research could still be done on both sets of equations. The  $db3$  example is much more complex but should have some structure that can be used to achieve consistency and convergence as can be seen from  $\Gamma_{gb}$ . For this problem different approaches and methods will be needed to exploit this structure without going back to the dominated class of polynomials and the Gröbner Basis form. Another task will be to find all possible solutions of the  $db2$  equations as this should be possible as seen from the structure in the  $\Sigma_{gb}$  equations.

These results however give some valuable insight and pointers into a generalized approach of optimizing polynomial problems with recursions without using special classes of polynomials and without the need of a Gröbner Basis calculation.

The parallel computing framework developed as one goal of the project proved to be essential for exploring properties of the  $db3$  wavelets. A good performance could be achieved by sending 500 sets to each client at a time to prevent continuous network usage and avoid long delays by sending too many sets. The  $db3$  example mentioned in the previous section with 10009125 possible starting sets could be calculated by using seven dual core computers in about 2 days. Some interesting aspects could

be seen during the calculations.

The framework was written in Java to be independent from architectures and operating systems. Windows as well as Linux computers were used during the research. Performance statistics show that the Java Runtime Environment (JRE) for Windows performed better than the JRE for Linux. The same trend could be seen between 32-bit versions of the JRE which performed better than the 64-bit versions.

Possible future enhancements could be to utilize native machine code for further speed gain as well as to change the distribution architecture to allow for generalized objects making it a multi-purpose parallelized distributed computing application.

## 7 Acknowledgments

Great appreciation and thanks to my supervisors Ralf Peeters and Ivo Bleylevens for presenting the opportunity for this interesting topic, thanks for their help, insight, and support in researching on this thesis, and thanks for a memorable experience.

## References

- [1] Ajwa, Iyad A., Liu, Zhuojun, and Wang, Paul S. (1995). Groebner Bases Algorithm. Technical report, ICM Technical Reports Series.
- [2] Bleylevens, Ivo, Peeters, Ralf, and Hanzon, Bernard (2007). Efficiency improvement in an  $nD$  systems approach to polynomial optimization. *Journal of Symbolic Computation* 42, pp. 30–53.
- [3] Buchberger, Bruno (1986). In *Recent Trends in Multidimensional System Theory*, Chapter A Criterion for Detecting Unnecessary Reductions in Polynomial Ideal Theory, pp. 184–232. D. Reidel Publishing Company.
- [4] Daubechies, Ingrid (2004). *Ten Lectures on Wavelets*. SIAM, 8th edition.
- [5] Graham, Ronald L., Knuth, Donald E., and Patashnik, Oren (2002). *Concrete Mathematics – A Foundation for Computer Science*. Addison–Wesley, 2nd edition.
- [6] Möller, H. Michael and Stetter, Hans J. (1995). *Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems*, Vol. 70 of *Numerische Mathematik*, pp. 311–329.
- [7] Strang, Gilbert and Nguyen, Truong (1996). *Wavelets and Filter Banks*. Wellesley-Cambridge Press.
- [8] Strang, Gilbert (2005). *Linear Algebra and its applications*. Brooks/Cole, 4th edition.