

**A new algorithm for solving
multivariate polynomial problems by
means of interpolation**

Raf Vandebril

Marc Van Barel

Olivier Ruatta

Bernard Mourrain

Report TW 343, 30/07/2002



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A new algorithm for solving multivariate polynomial problems by means of interpolation

Raf Vandebril

Marc Van Barel

Olivier Ruatta

Bernard Mourrain

Report TW 343, 30/07/2002

Department of Computer Science, K.U.Leuven

Abstract

In this paper we present a new kind of algorithm, for finding a solution $(g_0(\mathbf{x}), \mathbf{g}_1(\mathbf{x}), \dots, \mathbf{g}_n(\mathbf{x}))$ of the system:

$$g_0(\mathbf{x})\mathbf{p}_0(\mathbf{x}) + \mathbf{g}_1(\mathbf{x})\mathbf{p}_1(\mathbf{x}) + \dots + \mathbf{g}_n(\mathbf{x})\mathbf{p}_n(\mathbf{x}) = \mathbf{v}(\mathbf{x}),$$

where $v(\mathbf{x})$, $\mathbf{g}_i(\mathbf{x})$ and $p_i(\mathbf{x})$ are multivariate polynomials of a certain degree, in the variables $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$. The algorithm is based on a multivariate interpolation approach, which is a straightforward extension of the univariate algorithm of Van Barel and Bultheel. In their approach interpolation points are added one after each other, taking into account the degree structure of the solution. In this paper, exactly the same is done, for solving multivariate interpolation problems. Every step a new interpolation point is introduced, so that the intermediate result satisfies all the previous interpolation conditions and also the new added one. After having added enough interpolation points we will have the solution. Another difference between the univariate and the multivariate approach, is the number of variables and their combinations. In the multivariate case we can have a lot of monomials of the same global degree but having different degrees in each variable, and these degrees play an important role in for example algebraic geometry.

We mentioned that we search for a solution of the system (not a unique solution), but in fact we get even more, we get a set of polynomial vectors which generate the module of all the solutions.

An implementation of the algorithm is made in Maple and we tested it for some different multivariate examples, varying the number of unknowns and their degree structure.

Keywords : multivariate interpolation, multivariate polynomials, updating algorithm, module of solution vectors, pivoting

AMS(MOS) Classification : Primary : 65D05, Secondary : 41A05.

A new algorithm for solving multivariate polynomial problems by means of interpolation *

Raf Vandebril [†],
Marc Van Barel [‡],
Olivier Ruatta [§],
Bernard Mourrain [¶]

1-11-2001

Abstract

In this paper we present a new kind of algorithm, for finding a solution $(g_0(\mathbf{x}), g_1(\mathbf{x}), \dots, g_n(\mathbf{x}))$ of the system:

$$g_0(\mathbf{x})p_0(\mathbf{x}) + g_1(\mathbf{x})p_1(\mathbf{x}) + \dots + g_n(\mathbf{x})p_n(\mathbf{x}) = v(\mathbf{x}),$$

where $v(\mathbf{x}), g_i(\mathbf{x})$ and $p_i(\mathbf{x})$ are multivariate polynomials of a certain degree, in the variables $\mathbf{x} = (x_1, x_2, \dots, x_n)$. The algorithm is based on a multivariate

*This research has been established due to the project Tournesol, by the embassy of France in Brussels. It is the result of a cooperation between the MaSe team of M. Van Barel and the Galaad group of B. Mourrain. This research was partially supported by the K.U.Leuven (Bijzonder Onderzoeksfonds), project “SLAP: Structured Linear Algebra Package,” grant #OT/00/16 and partially by the Fund for Scientific Research–Flanders (FWO–V), project “SMA: Structured Matrices and their Applications” grant #G.0078.01. The work of the authors is also supported by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister’s Office for Science, Technology and Culture. The scientific responsibility rests with the authors.

[†]E-Mail: raf.vandebril@cs.kuleuven.ac.be

[‡]E-Mail: marc.vanbarel@cs.kuleuven.ac.be

[§]E-Mail: olivier.ruatta@sophia.inria.fr

[¶]E-Mail: bernard.mourrain@sophia.inria.fr

interpolation approach, which is a straightforward extension of the univariate algorithm of Van Barel and Bultheel [15]. In their approach interpolation points are added one after each other, taking into account the degree structure of the solution. In this paper, exactly the same is done, for solving multivariate interpolation problems. Every step a new interpolation point is introduced, so that the intermediate result satisfies all the previous interpolation conditions and also the new added one. After having added enough interpolation points we will have the solution. Another difference between the univariate and the multivariate approach, is the number of variables and their combinations. In the multivariate case we can have a lot of monomials of the same global degree but having different degrees in each variable, and these degrees play an important role in for example algebraic geometry.

We mentioned that we search for a solution of the system (not a unique solution), but in fact we get even more, we get a set of polynomial vectors which generate the module of all the solutions.

An implementation of the algorithm is made in Maple and we tested it for some different multivariate examples, varying the number of unknowns and their degree structure.

Keywords: multivariate interpolation, multivariate polynomials, updating algorithm, module of solution vectors, pivoting

1 Introduction

The multivariate interpolation problem, is a very interesting, but not so simple problem. Not so simple, because a lot of very useful properties from the univariate case cannot be used anymore, and also the growing number of variables makes the problem much more complicated than the univariate one. Already a lot of research has been done on this topic.

We will give here a very brief overview of the methods known for multivariate interpolation. An historical overview of the multivariate interpolation problem can be found in a paper by Sauer and Gasca [4].

Carl de Boor wrote some papers concerning multivariate interpolation problems, varying from an error analysis [1], to concrete techniques for doing interpolation [2]. The latter paper presents a map, mapping a number of interpolation points to the multivariate polynomial interpolating these points.

Another important researcher in this field is T. Sauer, who wrote several papers concerning multivariate interpolation. He did not only write theoretical results, but

he also gave some attention to the computational aspects of this problem [11]. Just like de Boor, he searched for a polynomial satisfying a number of fixed interpolation conditions, leading to the extension of Lagrange interpolation [13]. Another more difficult topic he dealt with was the minimal degree interpolation [12], and, as mentioned before, he published a paper presenting a wide range overview of multivariate interpolation problems, techniques and possible solutions [4].

But the algorithm we present here is significantly different. We use a kind of updating algorithm based on the article of Van Barel and Bultheel [15], where all the interpolation points are added one after the other. Simply spoken we construct a polynomial matrix in which all the columns satisfy the interpolation conditions. This matrix will therefore be updated everytime we add one interpolation condition, in order to satisfy this new condition. But as mentioned before, the problem is more complicated than the univariate case, here the solution matrix will grow everytime we add new interpolation points. And the solution is not uniquely represented anymore, but we get a module of polynomial vectors generating all the possible solutions.

Just like in the univariate case, our algorithm could be a very useful tool, also for other topics, for example in the univariate case we have some papers by Bultheel, Van Barel and Kravanja: superfast Toeplitz solvers [17], fast Hankel and Loewner matrix solvers [5, 18], connections with rational interpolation techniques [16]. All these papers are just a few cases where the univariate interpolation problem is a very useful and helpful tool. It has to be mentioned that the problem we try to solve here, does not stand on its own. We were inspired to do a thing like this by looking at a specific topic of algebraic geometry, in which we search for the common roots of a set of polynomials in multiple variables. A very good introduction to this topic is the paper of Mourrain and Pan [7], in which there is a full and detailed explanation of the problem. A certain matrix called the resultant matrix arises (look at the papers of Sturmfels [14] and Emiris and Mourrain [3]). In one of the methods to compute common roots, a submatrix of this resultant matrix has to be inverted, and that is in fact the problem we solve here. Mourrain wrote in cooperation with Pan and Bondyfalat several papers about this topic [9, 10, 6, 8]. We have to mention that we did not take into account all the specific properties of the solution as is done in the algebraic geometry, but this does not mean that it is useless for solving this problem. Little modifications, (which we will mention during the paper), are needed to make it a useful and powerful tool in algebraic calculations.

2 The problem

Before formulating the problem, we will first explain some notations used throughout the paper. The polynomials we consider are multivariate polynomials. When we denote $p(\mathbf{x})$ this means that $p(\mathbf{x})$ represents a polynomial in the variables $\mathbf{x} = (x_1, x_2, \dots, x_n)$. Just like in the univariate case these polynomials have a basis, and that is where our first difference with algebraic geometry appears, we will consider bases, which are of full degree. For example lets take the polynomial $p(\mathbf{x}) = x_1x_2 + 2x_1 + 3x_2 + 7$. In algebraic geometry one will use a basis $\{1, x_1, x_2, x_1x_2\}$ whereas we will deal with a basis which is complete for the degree of the polynomial, so our basis looks like $\{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\}$. This implies that the degree of a multivariate polynomial is the maximum degree appearing in the basis expansion of this polynomial. The vectorspace generated by a multivariate basis will be denoted by $\langle \mathbf{x}^E \rangle$, this is an abbreviation for the set of all linear combinations of the monomials $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$, where $\alpha \in \mathbf{E} \subset \mathbf{N}^n$.

The multivariate interpolation problem we will solve is the following: consider the map:

$$\begin{aligned} \mathcal{S} : \langle \mathbf{x}^{E_0} \rangle \times \dots \times \langle \mathbf{x}^{E_n} \rangle &\rightarrow \langle \mathbf{x}^F \rangle \\ (q_0(\mathbf{x}), \dots, q_n(\mathbf{x})) &\mapsto \sum_{i=0}^n p_i(\mathbf{x})q_i(\mathbf{x}), \end{aligned}$$

where the $p_0(\mathbf{x}), \dots, p_n(\mathbf{x})$ are fixed polynomials of a certain degree. We search for the polynomials $q_0(\mathbf{x}), \dots, q_n(\mathbf{x})$ as a solution of the following system

$$\sum_{i=0}^n p_i(\mathbf{x})q_i(\mathbf{x}) = v(\mathbf{x}),$$

with $v(\mathbf{x})$ a fixed polynomial. The $\langle \mathbf{x}^{E_j} \rangle, \forall j \in \{1, \dots, n\}$ are the vector spaces spanned by the monomials \mathbf{x}^{E_j} .

Let the polynomials $p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x}), v(\mathbf{x})$ be of degrees d_0, d_1, \dots, d_n, d respectively. We search for a solution $(g_0(\mathbf{x}), g_1(\mathbf{x}), \dots, g_n(\mathbf{x}))$ that satisfies the following degree structure:

$$\deg(g_i(\mathbf{x})) \leq d - d_i \quad \forall i \in \{0, \dots, n\}. \quad (1)$$

The first step in solving the problem consists of transforming the system with an arbitrary righthandside to a homogeneous problem. (Again, as throughout the whole paper, the bold symbols denote vectors, vectors of polynomials, as well as

vectors of variables.) The problem can be written in a compact form, using matrix notations: $\mathbf{P}(\mathbf{x}) \mathbf{G}^T(\mathbf{x}) = v(\mathbf{x})$, where $\mathbf{P}(\mathbf{x})$ denotes the vector of polynomials in \mathbf{x} ,

$$\mathbf{P}(\mathbf{x}) = [p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x})],$$

in a similar way the vector polynomial $\mathbf{G}(\mathbf{x})$ is defined as

$$\mathbf{G}(\mathbf{x}) = [g_0(\mathbf{x}), g_1(\mathbf{x}), \dots, g_n(\mathbf{x})].$$

To make the system homogeneous we rewrite it as follows:

$$\mathbf{P}(\mathbf{x}) \mathbf{G}^T(\mathbf{x}) - v(\mathbf{x}) = 0.$$

When we denote the concatenations with tildes $\tilde{\mathbf{P}}(\mathbf{x}) = [\mathbf{P}(\mathbf{x}), -v(\mathbf{x})]$ and $\tilde{\mathbf{G}}(\mathbf{x}) = [\mathbf{G}(\mathbf{x}), 1]$ we obtain the following homogeneous system:

$$\tilde{\mathbf{P}}(\mathbf{x}) \tilde{\mathbf{G}}^T(\mathbf{x}) = 0.$$

To ease the notation we will omit the tildes and denote the new system as

$$\mathbf{P}(\mathbf{x}) \mathbf{G}^T(\mathbf{x}) = 0, \tag{2}$$

where here $\mathbf{P}(\mathbf{x})$ stands for the vector $[p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x}), -v(\mathbf{x})]$, and $\mathbf{G}(\mathbf{x})$ equals $[g_0(\mathbf{x}), g_1(\mathbf{x}), \dots, g_n(\mathbf{x}), 1]$.

3 Solving the problem using interpolation

The following two sections may perhaps be a little hard to understand at first look, therefore it is not so bad to simply read these two paragraphs and after sections 5 and 6 glance back at these sections.

In this section we will describe how to solve the system mentioned in the previous paragraph, using multivariate interpolation techniques. We describe here all the basic steps required for adding interpolation points, and we will also draw attention to some specific topics. In a lot of places the resemblance with the univariate algorithm of Van Barel and Bultheel [15], will be rather great, however we will not hesitate to explain these things again, to get a complete and thorough understanding of the problem and the solution method. When the approach here differs from the univariate one, also some explanation will be given why these changes are needed. Again like we did before, we mention the places where the

algebraic people have to make adaptations to make the algorithm suitable for their problem.

We use the same notation as in the previous section, so the solution we search for is the vector $\mathbf{G}(\mathbf{x}) = [g_0(\mathbf{x}), g_1(\mathbf{x}), \dots, g_n(\mathbf{x}), 1]$, which has degree structure $[d - d_0, d - d_1, \dots, d - d_n, 0]$. Using the degree structure we can calculate the number of interpolation points we need. We denote the interpolation points with

$$\Omega = (\omega_{i,j})_{i,j}, \quad (3)$$

where Ω is a matrix which consists of the coordinates of the interpolation points. The number of interpolation points corresponds to the number of rows of Ω , and the number of variables n corresponds to the number of columns. The necessary number of interpolation points equals the sum of the numbers of basis vectors coming from all the different vector spaces. When we denote with $|E_i|$ the number of monomials which span $\langle \mathbf{x}^{E_i} \rangle$ then the number of interpolation points equals $|E| = \sum_{i=1}^n |E_i|$. Take for example $\langle \mathbf{x}^{E_i} \rangle$ of degree k , then the number of monomials in the basis of this vector space equals $C_k^{n+k} = (n+k)!/(n!k!)$, where n denotes the number of variables. As already mentioned before we add all the interpolation points one after another. So at a certain step in the algorithm, we interpolate in point i , which means in the point $(x_1, x_2, \dots, x_n) = (\omega_{i,1}, \dots, \omega_{i,n})$ which we will briefly denote as $\mathbf{x} = \omega_i$. What has to be mentioned, is that in the algorithm, a lot more interpolation points are generated. Between all these interpolation points, we search for that point which gives us the most stable operation. This does not mean that we interpolate in all these points, no we just take the most convenient ones. The choice of interpolation points is also a very interesting topic: as can be seen in the numerical results (section 8), the different choices of points also give different accuracies.

During the algorithm, we update in each step a matrix of polynomials called $\mathbf{B}(\mathbf{x})$. (The algorithm will be described in the following section). In the beginning of step k , just $k - 1$ interpolation conditions will be satisfied, where at the end of this step, $\mathbf{B}(\mathbf{x})$ satisfies:

$$\mathbf{P}(\omega_j)\mathbf{B}(\omega_j) = 0 \quad \forall j \in \{1, \dots, k\}.$$

We will now briefly describe one interpolation step of the algorithm. It consists of a matrix multiplication of the matrix $\mathbf{B}(\mathbf{x})$ with the matrix $\mathbf{T}(\mathbf{x})$, from the next definition.

Definition 1 When we denote the interpolation points with Ω , then we define the matrix $\mathbf{T}_{p,k,l}(\mathbf{x})$ of dimension $l \times (l + (n - 1))$ (with l as the row dimension, p as the pivot, k as the interpolation point) as follows : $\mathbf{T}_{p,k,l}(\mathbf{x}) =$

$$\begin{pmatrix} 1 & 0 & \dots & & \dots & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & & & & & & \\ 0 & & 0 & 1 & & & & & \\ \frac{r_1}{r_p} & \frac{r_2}{r_p} & \dots & \frac{r_{p-1}}{r_p} & (x_1 - \omega_{k,1}) & \frac{r_{p+1}}{r_p} & \dots & \frac{r_{l-1}}{r_p} & \frac{r_l}{r_p} & (x_2 - \omega_{k,2}) & \dots & (x_n - \omega_{k,n}) \\ 0 & & & & & 1 & & & & & & \\ \vdots & & & & & & \ddots & & \vdots & & & \\ & & & & & & & 1 & 0 & \vdots & & \vdots \\ 0 & \dots & & & \dots & 0 & 1 & 0 & 0 & \dots & \dots & 0 \end{pmatrix}$$

where $[r_1, r_2, \dots, r_p, \dots, r_l]$, is a vector we get as input (assuming $r_p \neq 0$), and the factor $(x_1 - \omega_{k,1})$ stands on the intersection of row p with column p .

In the univariate case we use a matrix resembling a lot this one. Here the first difference between the univariate and the multivariate case arises, where in the univariate case the number of variables is one and therefore the matrix \mathbf{T} is square, here the matrix is rectangular. It can easily be seen from the structure of \mathbf{T} that it is a concatenation from a square $l \times l$ matrix with a rectangular $l \times (n - 1)$ matrix, the square part is exactly the same as in the univariate case, whereas here we have to add the $n - 1$ extra columns corresponding to the other variables x_2, x_3, \dots, x_n . The value p in $\mathbf{T}_{p,k,l}(\mathbf{x})$, is called the pivot, and will play a very important role in the algorithm. Because the algorithm is a little vague at this point we will first take a look at the algorithm. The next section deals with the correctness of the approach used, whereas the last section is concerned with stability problems.

4 The algorithm

This section briefly explains the algorithm to solve the multivariate interpolation problem. The correctness of the algorithm, is deduced in the next section, where all the steps used here will be validated with proofs. Some of the variables used here will not be so clear at this point, they will be explained in the next section.

Before we start with the description of the algorithm, you may want to perform some tests or take a look at the implementation in Maple. You can download the algorithm at <http://www.cs.kuleuven.ac.be/~marc/software>.

The algorithm as mentioned before is an updating algorithm, in every step we will add a new interpolation point. Our temporary solution will satisfy, all the previous interpolation conditions, and also the new interpolation condition we just added. In the algorithm some kind of pivoting is incorporated to improve the numerical stability. The algorithm also has to take into account the degree of the polynomials constructed, in order to obtain a solution with a satisfying degree.

For the description of the algorithm we use the same notation as used in the previous sections. We give here a high level language description, which resembles a mixture of maple and matlab code. In the first algorithmic description we give an updating step as mentioned before, in the second part we describe the main algorithm and also how to incorporate the degree structures for the solution. Remark however that a lot of easy procedures for example the searching of the pivot, are not explained in detail. They are easy to program and incorporating them here would only make the algorithm more unreadable.

Definition 2 *A brief explanation of all the variables used during the algorithm.*

- **P** corresponds to the vector $[p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x}), -v(\mathbf{x})]$.
- **B** is the matrix that will be updated every time in the procedure **IteratInterpol**, the nonhomogeneous solution will be found in column $n + 2$ of this matrix. (this is because of our choice of s in defining the s -degree.)
- *numip* equals the number of interpolation points, this number is larger then the necessary number of points.
- Ω the $\text{numip} \times n$ matrix containing the interpolation points (cfr. (3)).
- **sdeg** is a vector in which every component corresponds to the s -degree of the corresponding column of $\mathbf{B}(\mathbf{x})$.
- **deg** is the vector keeping the maximum permitted degree of the solution (cfr. (1)).
- *ipoint* is the point we want to interpolate in.
- *pivot* and p denote the pivot.
- $\mathbf{T}_{\text{pivot}, \text{ipoint}, l}$ look at definition 1.
- **resB** will denote the matrix of residuals.

algorithm 1 (Some basic procedures explained) *The searchpivot procedure, the Iteratinterpol procedure and the adaptsdegree procedure will be described with an explanation of their input and output values.*

The searchpivot procedure

procedure [pivot,row]:=searchpivot(sdeg,resB)

This procedure, searches within the columns with a minimal s-degree, for the largest component, in resB. It gives as output, the column in which this largest element is found: this is the pivot, and the row in which it was found: this is the interpolationpoint.

The procedure adaptsdegree

procedure [sdeg]:=adaptsdeg(sdeg,pivot)

This procedure recalculates sdeg of the matrix B(x). This can be done very easily when one knows the pivot of the iterationstep just performed:

sdeg:=[sdeg[1],sdeg[2], ..., sdeg[p-1], sdeg[p]+1,

sdeg[p+1], ...,sdeg[k_m],sdeg[p]+1, ..., sdeg[p]+1].

This vector will get n - 1 new columns, corresponding to the new added columns in B(x).

The procedure for one iterationstep

procedure [B,resB]:=IteratInterpol(P,B,resB,pivot,Ω,ippoint) *Here one step of iteration will be performed, to update the matrix B. The result will be stored in the matrix B. The matrix resB will also be adapted in this procedure.*

1. $l := \text{ColumnDimension}(\mathbf{B});$
2. $[r_1, r_2, \dots, r_l] := \text{resB}[ippoint, 1 : l];$
3. *construct* $\mathbf{T}_{pivot,ippoint,l}$ *with* $[r_1, r_2, \dots, r_l];$
4. $\mathbf{B} := \mathbf{B} * \mathbf{T}_{pivot,ippoint,l};$
5. *update the matrix resB;*
6. *RETURN(B,resB);*

algorithm 2 (Solving Multivariate Interpolation Problems) *The variables used in this algorithm are defined in definition 2.*

- **Initialisation** *Set all the initial values.*

1. Calculate **deg**, **sdeg**, *numip*;
2. $numip = numip + extra$ Extra corresponds to the extra number of interpolation points
3. $\mathbf{B} := identity(n + 1)$;
4. initialize Ω as a $numip \times n$ matrix.
5. initialize **resB**;

- **Main Algorithm**

```

while sdeg  $\neq [d + 1, \dots, d + 1]$  do
    [pivot, ippoint] := searchpivot(sdeg, resB);
    if resB[ippoint, pivot] is too small then
        sdeg[pivot] =  $d + 1$ ;
        incorporate column in solution module;
    else
        [B, resB] := IteratInterpol(P, B, resB, pivot,  $\Omega$ , ippoint);
        sdeg := adaptsdeg(sdeg, pivot);
    endif;
    if column(resB)[ $n + 2$ ] is too small
        stop whileloop, cause nonhomogeneous solution is found
    endif;
endwhile;

search all the columns of satisfying degree

this is the solution module

```

5 Proof of the correctness of the algorithm.

This section, proves the correctness of the algorithm, mentioned above. Also some further explanation of the variables used above, will be given.

Remark that when starting the algorithm. Before introducing the first interpolation point, the matrix $\mathbf{B}(\mathbf{x})$, has to be the identity matrix of dimension $n + 2$.

The next theorem states, that all the solutions of the multivariate interpolation problem at a certain step k , can be written as a polynomial linear combination of the columns of the matrix $\mathbf{B}(\mathbf{x})$ at that step k .

Theorem 2.1 Denote with $\mathbf{P}(\mathbf{x})$ the following vector of multivariate polynomials:

$$\mathbf{P}(\mathbf{x}) = [p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x})]$$

where \mathbf{x} stands for: $\mathbf{x} = (x_1, x_2, \dots, x_n)$.

Again we denote the interpolation points with the matrix $\Omega = (\omega_{i,j})$, where $\omega_{i,j}$ means, component j , of interpolation point i .

A polynomial vector $\mathbf{G}(\mathbf{x})$ satisfies the interpolation condition in interpolation point k if the matrix vector produkt $\mathbf{P}(\omega_k)\mathbf{G}^T(\omega_k)$ equals zero.

Denote with $\mathbf{B}^{(m)}(\mathbf{x})$ the matrix $\mathbf{B}(\mathbf{x})$ at step m of the algorithm.

Under the assumptions that m interpolation conditions are satisfied, all the vector solutions, satisfying these m interpolation conditions, can be written as a linear polynomial combination of the columns of the matrix $\mathbf{B}^{(m)}(\mathbf{x})$.

Proof

- The result will be proven by induction.
- The theorem is true for zero interpolation conditions.
- Assume that the first m interpolation conditions are satisfied, then it remains to prove that, after adding one interpolation point, all the vector polynomials satisfying these $m + 1$ interpolation conditions, can be written as a linear polynomial combination of the columns of the matrix $\mathbf{B}^{(m+1)}(\mathbf{x})$.

First of all, suppose, by induction that

$$\mathbf{B}^{(m)}(\mathbf{x}) = [b_1^{(m)} \ b_2^{(m)} \ \dots \ b_{k_m}^{(m)}],$$

(where k_m , denotes the column dimension of the matrix $\mathbf{B}^{(m)}(\mathbf{x})$) where the solution module \mathbf{S}_m is generated by the columns of $\mathbf{B}^{(m)}$.

To add a new interpolation point we first have to evaluate the product

$$\mathbf{P}(\mathbf{x})\mathbf{B}^{(m)}(\mathbf{x})$$

in the new interpolation point lets say ω_{m+1} . This gives us the residual vector:

$$\mathbf{P}(\omega_{m+1})\mathbf{B}^{(m)}(\omega_{m+1}) = \mathbf{R}_m = [r_1^{(m+1)} r_2^{(m+1)} \dots r_{k_m}^{(m+1)}],$$

We have to distinguish between two cases now. In the first case, if all the residuals are zero, this means that the $(m+1)$ th interpolation condition is dependent on the previous one, which implies that the solution spaces are the same : $\mathbf{S}_m = \mathbf{S}_{m+1}$. In this case the theorem is proven.

In the second case there exists a $p \in \{1, \dots, k_m\}$ such that $r_p^{(m+1)} \neq 0$, this already implies that $\mathbf{S}_{m+1} \subset \mathbf{S}_m$. To update the matrix $\mathbf{B}^{(m)}(\mathbf{x})$ we construct the matrix $\mathbf{T}_{p,m+1,k_m}(\mathbf{x})$. And we get: $\mathbf{B}^{(m+1)}(\mathbf{x}) = \mathbf{B}^{(m)}(\mathbf{x})\mathbf{T}_{p,m+1,k_m}(\mathbf{x})$. It is clear that all the columns of $\mathbf{B}^{(m+1)}(\mathbf{x})$ satisfy the interpolation conditions for $l = 1, \dots, m+1$.

It remains to be proven that every solution $\mathbf{U}(\mathbf{x}) \in \mathbf{S}_{m+1}$ can be written as a linear combination of the columns of $\mathbf{B}^{(m+1)}$. Because we have the inclusion $\mathbf{S}_{m+1} \subset \mathbf{S}_m$, we only have to look for all the polynomial vectors $\mathbf{U}(\mathbf{x}) \in \mathbf{S}_m$ such that the $(m+1)$ th interpolation condition is satisfied. We know that $\mathbf{U}(\mathbf{x})$ can be written as $\mathbf{U}(\mathbf{x}) = \mathbf{B}^{(m)}(\mathbf{x})\mathbf{G}^T(\mathbf{x})$, so in fact we search for all polynomial vectors $\mathbf{G}(\mathbf{x})$ such that:

$$\mathbf{P}(\omega_{m+1})\mathbf{B}^{(m)}(\omega_{m+1})\mathbf{G}^T(\omega_{m+1}) = 0$$

i.e.

$$\mathbf{R}_m \mathbf{G}^T(\omega_{m+1}) = 0 \tag{4}$$

It remains to be shown that all the polynomials $\mathbf{G}(\mathbf{x})$ satisfying (4), can be presented as:

$$\mathbf{T}_{p,m+1,k_m}(\mathbf{x})\mathbf{Q}^T(\mathbf{x}),$$

because this means that the polynomial $\mathbf{G}(\mathbf{x})$, can be written as a linear combination of the columns of $\mathbf{B}^{(m+1)}(\mathbf{x})$.

Take a polynomial vector $\mathbf{G}(\mathbf{x})$ satisfying $\mathbf{R}_m \mathbf{G}^T(\omega_{m+1}) = 0$ which in fact is

$$r_1^{(m+1)}\mathbf{G}_1(\omega_{m+1}) + \dots + r_p^{(m+1)}\mathbf{G}_p(\omega_{m+1}) + \dots + r_{k_m}^{(m+1)}\mathbf{G}_{k_m}(\omega_{m+1}) = 0,$$

which means that:

$$\mathbf{G}_p(\omega_{m+1}) = - \left(\frac{r_1^{(m+1)} \mathbf{G}_1(\omega_{m+1}) + \dots + r_{p-1}^{(m+1)} \mathbf{G}_{p-1}(\omega_{m+1})}{r_p^{(m+1)}} + \frac{r_{p+1}^{(m+1)} \mathbf{G}_{p+1}(\omega_{m+1}) + \dots + r_{k_m}^{(m+1)} \mathbf{G}_{k_m}(\omega_{m+1})}{r_p^{(m+1)}} \right) \quad (5)$$

We have in fact a lot of freedom for constructing such a polynomial vector $\mathbf{G}(\mathbf{x})$, all the polynomials $\mathbf{G}_1(\mathbf{x}), \dots, \mathbf{G}_{p-1}(\mathbf{x}), \mathbf{G}_{p+1}(\mathbf{x}), \dots, \mathbf{G}_{k_m}(\mathbf{x})$ can be chosen freely where the only restriction is the polynomial vector $\mathbf{G}_p(\mathbf{x})$ satisfying the equation here above.

The only thing left to prove is the existence of a polynomial $\mathbf{Q}(\mathbf{x})$ satisfying the equation:

$$\mathbf{T}_{p,m+1,k_m}(\mathbf{x}) \mathbf{Q}^T(\mathbf{x}) = \mathbf{G}(\mathbf{x}).$$

The following equalities follow directly from the definition of $\mathbf{T}_{p,m+1,k_m}$:

$$\mathbf{Q}_i(\mathbf{x}) = \mathbf{G}_i(\mathbf{x}) \quad \forall i \in \{1, \dots, p-1, p+1, \dots, k_m\}.$$

Writing down equation p gives us:

$$\begin{aligned} \mathbf{G}_p(\mathbf{x}) &= - \left[\frac{r_1^{(m+1)}}{r_p^{(m+1)}} \mathbf{G}_1(\mathbf{x}) + \dots + \frac{r_{p-1}^{(m+1)}}{r_p^{(m+1)}} \mathbf{G}_{p-1}(\mathbf{x}) \right. \\ &\quad \left. + \frac{r_{p+1}^{(m+1)}}{r_p^{(m+1)}} \mathbf{G}_{p+1}(\mathbf{x}) + \dots + \frac{r_{k_m}^{(m+1)}}{r_p^{(m+1)}} \mathbf{G}_{k_m}(\mathbf{x}) \right] \\ &\quad + (x_1 - \omega_{m+1,1}) \mathbf{Q}_p(\mathbf{x}) + (x_2 - \omega_{m+1,2}) \mathbf{Q}_{k_m+1}(\mathbf{x}) \\ &\quad + \dots + (x_n - \omega_{m+1,n}) \mathbf{Q}_{k_m+n-1}(\mathbf{x}). \end{aligned} \quad (6)$$

When we expand the polynomials $\mathbf{G}_i(\mathbf{x})$ in the following form:

$$\mathbf{G}_i(\mathbf{x}) = \mathbf{G}_{i,0} + (x_1 - \omega_{m+1,1}) \mathbf{G}_{i,1}(\mathbf{x}) + (x_2 - \omega_{m+1,2}) \mathbf{G}_{i,2}(\mathbf{x}) + \dots + (x_n - \omega_{m+1,n}) \mathbf{G}_{i,n}(\mathbf{x})$$

and substitute this together with (5) in (6), we get (after simplification) the following equalities

$$\mathbf{Q}_p(\mathbf{x}) = \frac{1}{r_p^{(m+1)}} \left(\sum_{k=1}^{k_m} r_k^{(m)} \mathbf{G}_{k,1}(\mathbf{x}) \right)$$

and for $\mathbf{Q}_j(\mathbf{x}) \forall j \in \{1, \dots, n-1\}$:

$$\mathbf{Q}_{k_m+j}(\mathbf{x}) = \frac{1}{r_p^{(m+1)}} \left(\sum_{k=1}^{k_m} r_k^{(m+1)} \mathbf{G}_{k,j+1}(\mathbf{x}) \right)$$

This proves the theorem. □

6 Taking into account the degree structure of the solution

The previous theorem did not say anything about the degree structure of the solution. This however is of course very important. For a good concept of degree in the multivariate case we have to give some definitions. The first concept which will need some explanation is the \mathbf{s} -degree.

Definition 3 Take $\mathbf{P}(\mathbf{x}) = [p_0(\mathbf{x}), p_1(\mathbf{x}), \dots, p_n(\mathbf{x})]$ as a vector of multivariate polynomials, and take $\mathbf{s} = [s_0, s_1, \dots, s_n]$, to be a vector of integers. Then the \mathbf{s} -degree of the vector $\mathbf{P}(\mathbf{x})$ is defined as:

$$\mathbf{s}\text{-degree}(\mathbf{P}(\mathbf{x})) = \max_{0 \leq i \leq n} \{ \deg(p_i(\mathbf{x})) - s_i \}$$

This \mathbf{s} -degree is in fact a tool for giving different importance to the degree of the different components of the vector. This will be very important to find a solution of the system satisfying our degree structure.

In our specific case we will define \mathbf{s} as follows:

$$\mathbf{s} = [-\deg(p_0(\mathbf{x})), -\deg(p_1(\mathbf{x})), \dots, -\deg(p_n(\mathbf{x})), -\deg(v(\mathbf{x})) - 1],$$

this corresponds to:

$$\mathbf{s} = [-d_0, -d_1, \dots, -d_n, -d - 1]$$

We already know how to add one interpolation point, but the choice of the pivot is in fact one of the most important things to obtain a satisfying degree structure. In every step of the algorithm, this means, adding a new interpolation condition, we have to take for pivot, the column with the lowest \mathbf{s} -degree. Suppose

there are two columns with the same, s -degree, then we take one of the two, which will give the most stable operation.

The next theorem proves that using this kind of updating, the solution we will find will be of the correct degree.

Theorem 3.1 *Regarding the interpolation problem $\sum_{i=0}^n g_i(\mathbf{x})p_i(\mathbf{x}) - v(\mathbf{x}) = 0$, where the $g_i(\mathbf{x})$ are the unknowns, and the $p_i(\mathbf{x}), v(\mathbf{x})$ are of degree respectively d_i, d . The matrix $\mathbf{B}(\mathbf{x})$ is the matrix satisfying the first m interpolation conditions, constructed in such a way that always the column with the smallest s -degree was taken as a pivot.*

Then, every solution $\mathbf{G}(\mathbf{x})$, satisfying the first m interpolation conditions with

$$s\text{-degree}(\mathbf{G}(\mathbf{x})) \leq \delta$$

can be written as

$$\mathbf{G}(\mathbf{x}) = \sum_{i=0} \mathbf{b}_i(\mathbf{x})a_i(\mathbf{x}),$$

where $\mathbf{b}_i(\mathbf{x})$ denotes the i th column of $\mathbf{B}(\mathbf{x})$ and where $a_i(\mathbf{x})$ is a polynomial of degree $\deg(a_i(\mathbf{x})) \leq \delta - \delta_i$, where $\delta_i = s\text{-degree}(\mathbf{b}_i(\mathbf{x}))$.

Proof

- Again we will proof this property by induction.
- The first step where the matrix $\mathbf{B}^{(0)}(\mathbf{x})$ equals the identity matrix is trivial.
- We assume that the property is true for the matrix $\mathbf{B}^{(m)}(\mathbf{x})$. This means, that for a solution vector $\mathbf{G}^{(m)}(\mathbf{x})$ satisfying the m first interpolation conditions, with

$$s\text{-degree}(\mathbf{G}^{(m)}(\mathbf{x})) \leq \delta$$

we have that

$$\mathbf{G}^{(m)}(\mathbf{x}) = \sum_{j=0}^{k_m} \mathbf{b}_j^{(m)}(\mathbf{x})a_j^{(m)}(\mathbf{x})$$

where

$$\begin{aligned} \deg(a_j^{(m)}(\mathbf{x})) &\leq \delta - \delta_j^{(m)} \\ s\text{-degree}(\mathbf{b}_j^{(m)}(\mathbf{x})) &\leq \delta_j^{(m)}. \end{aligned}$$

It remains for us to prove that the same statement is true after adding one more interpolation point. Assume that we add one more interpolation point ω_{m+1} . This means that $\mathbf{B}^{(m+1)}(\mathbf{x})$ equals the matrix $\mathbf{B}^{(m)}(\mathbf{x}) \mathbf{T}_{p,m+1,k_m}(\mathbf{x})$. Take now a vector $\mathbf{G}^{(m+1)}(\mathbf{x})$ satisfying the $m+1$ first interpolation conditions. Using the previous theorem we know that there exists a polynomial vector $\mathbf{A}^{(m+1)}(\mathbf{x})$ such that:

$$\begin{aligned} \mathbf{G}^{(m+1)} &= \mathbf{B}^{(m+1)}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x}) \\ &= \mathbf{B}^{(m)}(\mathbf{x}) \mathbf{T}_{p,m+1,k_m}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x}) \\ &= \sum_{j=1}^{k_m} \mathbf{b}_j^{(m)} (\mathbf{T}_{p,m+1,k_m}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x}))_j \end{aligned}$$

We can now use the induction hypothesis, which states:

$$\deg(\mathbf{T}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x})) \leq \delta - \delta_j^{(m)} \quad (7)$$

with

$$\delta_j^{(m)} = \text{s-degree}(\mathbf{b}_j^{(m)})$$

Now we have to make a distinction between the columns $p, k_m + 1, \dots, k_m + n$, and the others.

Part 1 Suppose $j \neq p, k_m + 1, \dots, k_m + n$, This means according to the definition of the matrix $\mathbf{T}(\mathbf{x})$, that $(\mathbf{T}(\mathbf{x}) \mathbf{A}^{(m+1)})_j = a_j^{(m+1)}$, thus

$$\deg(a_j^{(m+1)}) \leq \delta - \delta_j^{(m+1)}.$$

Because our matrix $\mathbf{B}^{(m+1)}(\mathbf{x})$ is constructed in such a way that the column taken as a pivot has the smallest s-degree and a residual different from zero, we get :

$$\text{s-degree}(\mathbf{b}_p^{(m)}) \leq \text{s-degree}(\mathbf{b}_j^{(m)}) \quad \forall j,$$

when the residual of $b_j^{(m)}$ is different from zero. This means that (for α as a certain factor appearing on row p and column j in T)

$$\begin{aligned} \delta_j^{(m+1)} = \text{s-degree}(\mathbf{b}_j^{(m+1)}(\mathbf{x})) &= \text{s-degree}(\mathbf{b}_j^{(m)} + \alpha \mathbf{b}_p^{(m+1)}) \\ &\leq \text{s-degree}(\mathbf{b}_j^{(m)}(\mathbf{x})) = \delta_j^{(m)}. \end{aligned} \quad (8)$$

When the residual of $b_j^{(m)}$ equals zero, α also equals zero, and therefore $\delta_j^{(m)}$ is exactly the same as $\delta_j^{(m+1)}$.

This result together with (7) gives us the wanted result for $j \neq p, k_m + 1, \dots, k_m + n$.

Part 2 Here we deal with $j = p, k_m, \dots, k_m + n$. Suppose $j = p$, all the other cases are dealt with in an analogue way. We know that

$$\deg \left(\left(\mathbf{T}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x}) \right)_p \right) \leq \delta - \delta_p^{(m)} \quad (9)$$

where

$$\delta_p^{(m)} = \mathbf{s}\text{-degree} \left(\mathbf{b}_p^{(m)} \right).$$

We can rewrite equation (9) now as:

$$\begin{aligned} \deg \left(\left(\mathbf{T}(\mathbf{x}) \mathbf{A}^{(m+1)}(\mathbf{x}) \right)_p \right) &\leq \delta - \delta_p^{(m)} \\ \deg \left((x_1 - \omega_{k,1}) a_p^{(m+1)} \right) &\leq \delta - \delta_p^{(m)} \\ 1 + \deg \left(a_p^{(m+1)} \right) &\leq \delta - \delta_p^{(m)} \\ \deg \left(a_p^{(m+1)} \right) &\leq \delta - \left(\delta_p^{(m)} + 1 \right). \end{aligned}$$

Using this last equation together with the next one proves the theorem:

$$\begin{aligned} \delta_p^{(m+1)} = \mathbf{s}\text{-degree} \left(\mathbf{b}_p^{(m+1)} \right) &= \mathbf{s}\text{-degree} \left((x_1 - \omega_{k,1}) \mathbf{b}_p^{(m)} \right) \\ &= \delta_p^{(m)} + 1 \end{aligned}$$

□

7 Modifications to the algorithm, stability issues.

First of all we mention how pivoting can be incorporated in the algorithm to gain accuracy. Pivoting will be used when multiple columns can be used as a pivot, this occurs for instance during the procedure when several columns have the same smallest \mathbf{s} -degree. Pivoting is incorporated to perform an iteration step using the column which will give us the most stable operation.

Lets say that we are at a certain point in the procedure with the corresponding s-degree of the different columns like

$$[\delta_0, \delta_1, \dots, \delta_{k_m}].$$

We now only perform iterations on the columns corresponding to the smallest s-degree. Lets denote these minimum values with $\delta_{t_1}, \delta_{t_2}, \dots, \delta_{t_\alpha}$, where all the $t_i \in \{0, \dots, k_m\}$. These values correspond to the columns $t_1 + 1, t_2 + 1, \dots, t_\alpha + 1$. From these columns we have to choose our pivot, to do this we first evaluate the residuals of all the points which can still be used as a pivot, for one point this residual looks like:

$$\mathbf{P}(\Omega_k)\mathbf{B}(\Omega_k) = [r_1, r_2, \dots, r_l]$$

After this evaluation in all these points we get a matrix called *resB*, of dimension $numip \times k_m$. The rows correspond to the different interpolation points, where there are maximum, *numip* interpolationpoints. And the columns correspond to the different residuals connected to the corresponding column of $\mathbf{B}(\mathbf{x})$. We consider only the columns: $[t_1 + 1, t_2 + 1, \dots, t_m + 1]$, the pivot is one of these columns corresponding to the column in **resB**, in which the largest element in absolute value is found. The row in which this element is found corresponds to the interpolation point we are about to use. Note that this searching of the pivot does not increase the computational cost, but it does increase the stability of the algorithm. The choice of interpolation points is also very important for the stability, as can be seen in the numerical examples (section 8), where different choices on the same example are tested, and compared.

In the next part we deal with modifications for the algorithm when the basis structures $\langle \mathbf{x}^{E_j} \rangle$ are not complete, the algorithm can still be applied but it becomes more difficult to choose a pivot. These modifications can be applied for example in the theory of algebraic varieties for finding zeros of systems of equations. It has to be mentioned that the modifications here are not incorporated in the final algorithm, eventhough the algorithm is written in such a way that it is easy to adapt it to this special case.

The first thing that changes is the degree testing of a column, it will become more difficult. Whereas in the easy case it was only necessary to know the overall degree of the vectors involved we now have to deal with very specific degree structure. We shall explain this a bit further with a bivariate example. The main problem that arises now is that when constructing the s-degree before starting with a new sequence of operations, all these columns have to be tested on the specific degree structure before they can be used as a pivot, this means that our selection of the columns will become more strict and also more intensive.

Example 1 We consider a bivariate case and compare the two different approaches. With our approach we search for a solution in the following basis : $\mathbf{x}^{E_0} = \{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\}$, $\mathbf{x}^{E_1} = \{1, x_1, x_2\}$ and $\mathbf{x}^{E_2} = \{1, x_1, x_2\}$ which are all complete. In the algebraic approach they may look for a solution in the following basis: $\mathbf{x}^{E_0} = \{1, x_1, x_2, x_1x_2\}$, $\mathbf{x}^{E_1} = \{1, x_1, x_2\}$ and $\mathbf{x}^{E_2} = \{1, x_1, x_2\}$ where the first basis is not of complete degree anymore. Now the difference becomes clear when we take a column for example which looks like: $[1 + x_1 + x_2^2, x_1 + 3x_2 + 0.1, x_1, 2]^T$ evnthough this column has degree structure $[2, 1, 1, 0]$ it can not be used as a pivot in the second case, wheras the first case only looks at the overall degrees, which are admitted and we can use it as a pivot.

When using the algebraic approach this will of course affect the complexity of the algorithm, on one hand it will decrease the number of interpolation points before the solution is found, but on the other hand a lot of time will be spend on testing polynomial vectors for their degree structure, and all this information also has to be kept in memory.

A last thing where we have to pay attention to, is the size of the residual of the pivot. When the residual of the pivot is too small (for example about the order $10^{(-14)}$, when using IEEE-double precision), it is better not to perform this step. Because doing so will probably introduce large rounding errors, because a very small residual means that the impact of this column on the solution is neglectible. So it is better not to iterate with this column as a pivot. In fact this column will be a vector incorporated in the solution module. This corresponds with $\alpha = 0$ in equation (8).

8 Numerical examples

The implementation we made is available in maple and can be found and downloaded at <http://www.cs.kuleuven.ac.be/~marc/software>. The algorithm, uses pivoting to increase the numerical stability.

We performed two different types of tests. In the first we generated several examples in two variables, varying the degree.

Figure 1 gives an idea of the error produced by the algorithm, the error is measured as follows. We generated a fixed number of points r on the unit circle of two variables, which we denote as $z_i = [x_i, y_i]$ (such that $x_i^2 + y_i^2 = 1$). We denote

the computed solution as $\mathbf{G}(\mathbf{x})$. We measure the error by computing the residual:

$$res = \max_{1 \leq i \leq r} \frac{|\sum_{j=0}^4 p_j(x_i, y_i) g_j(x_i, y_i) - v(x_i, y_i)|}{\max_{0 \leq j \leq 4} \{|p_j(x_i, y_i) g_j(x_i, y_i)|, |v(x_i, y_i)|\}},$$

which is measured for three kinds of interpolation points: random points, an equal spaced grid, and a chebyshev grid always in $[-1, 1]$.

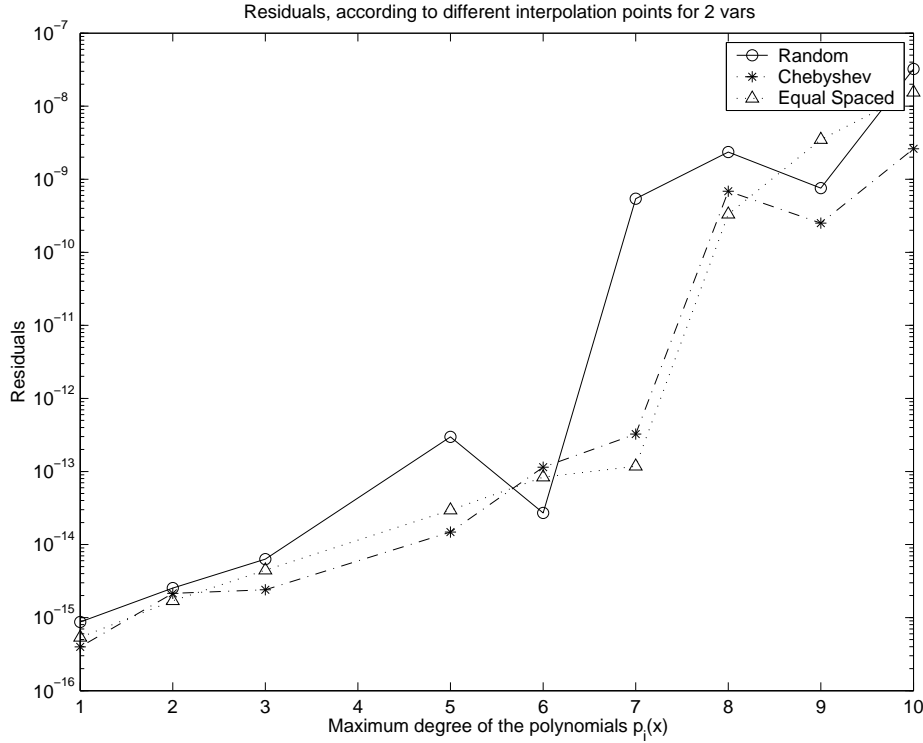


Figure 1: Relative residual norms for the algorithm, in case of two variables.

Figure 2 gives the number of iteration steps, versus the degree of the input vector $\mathbf{P}(\mathbf{x})$.

Figure 3 shows that the algorithm can be adapted reducing the maximum number of columns to compute a solution. The maximum number of columns is calculated in the following way:

$$maxnumcol = numvars + 2 + numip * (numvars - 1).$$

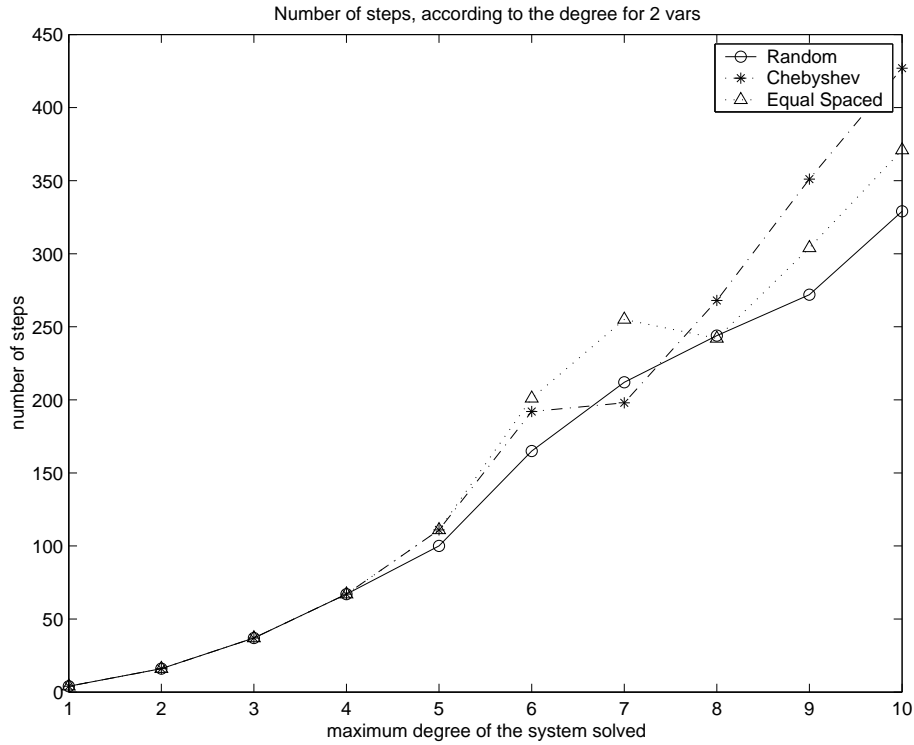


Figure 2: Number of steps versus the degree of the polynomial \mathbf{P} .

But in the algorithm, we can incorporate some cutting criterias. When a column in $\mathbf{B}(\mathbf{x})$ is too small, which means that the coefficients are very small, we can remove this column, because its residual will always be zero from now on, and this column will not contribute anymore in finding the solution. The second way to reduce the number of columns of $\mathbf{B}(\mathbf{x})$, is to pick out the solution columns, because we are searching for a module of solutions we immediately take out every column already satisfying our multivariate interpolation conditions. Because we maintain a matrix with residuals it is not so hard to see when a column is a solution of the homogeneous problem, so we can filter out these columns. This is what is presented in the next figure, the maximum number of columns, versus number of columns used in the algorithm.

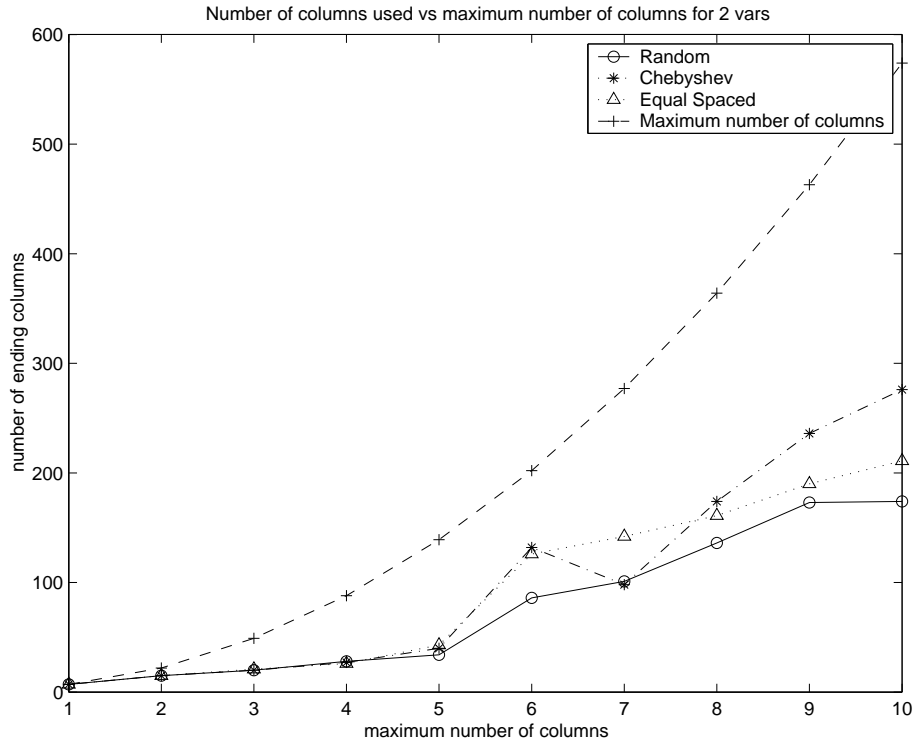


Figure 3: Number of columns, versus the maximum number of columns.

References

- [1] C. de Boor. Polynomial interpolation in several variables. *Studies in Computer Science* (J. R. Rice and R. A. DeMillo, eds.), Plenum Press, New York, 1994.
- [2] C. De Boor and A. Ron. Computational aspects of multivariate polynomial interpolation in several variables. *Math. Comp.*, 58:705–727, 1992.
- [3] Ioannis Z. Emiris and B. Mourrain. Matrices in elimination theory. *Journal of Symbolic Computation*, 28(1-2):3–43, 1999.
- [4] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Adv. Comput. Math.*, 12(4):377–410, 2000.

- [5] P. Kravanja and M. Van Barel. A fast Hankel solver based on an inversion formula for Loewner matrices. *Linear Algebra and Its Applications*, 282(1–3):275–295, September 1998.
- [6] B. Mourrain and Victor Y. Pan. Multidimensional structured matrices and polynomial systems. *Calcolo*, (Special Issue, Workshop on Toeplitz Matrices: Structure, Algorithms and Applications), 33:389–401, 1997.
- [7] B. Mourrain and Victor Y. Pan. Multivariate polynomials, duality and structured matrices. Technical Report 3513, INRIA, 1998.
- [8] Victor Y. Pan and B. Mourrain. Solving special polynomial systems by using structured matrices and algebraic residues. In F. Cucker and M. Shub, editors, *Proc. of the workshop on Foundations of Computational Mathematics (Rio de Janeiro)*, pages 287–304. Springer, 1997.
- [9] Victor Y. Pan and B. Mourrain. Asymptotic acceleration of solving multivariate polynomial systems of equations. In *ACM Symposium on Theory of Computing*, pages 488–496, 1998.
- [10] Victor Y. Pan, B. Mourrain, and D. Bondyfalat. Controlled iterative methods for solving polynomial systems. In *Proc. Annual ACM-SIGSAM Intern. Symp. on Symb. and Algebr. Comp. (ISSAC'98)*, pages 252–259. ACM Press, New York, 1998.
- [11] T. Sauer. Computational aspects of multivariate polynomial interpolation. *Advances in Comp. Math.*, 3:219–238, 1995.
- [12] T. Sauer. Polynomial interpolation of minimal degree. *Numerische Mathematik*, 78(1):59–85, 1997.
- [13] T. Sauer and Yuan Xu. On multivariate lagrange interpolation. *Mathematics of Computation*, 64(211):1147–1170, 1995.
- [14] B. Sturmfels. American mathematical society short course series: Introduction to resultants. *Applications of Computational Algebraic Geometry*, 1997.
- [15] M. Van Barel and A. Bultheel. A new approach to the rational interpolation problem: the vector case. *Journal of Computational and Applied Mathematics*, 33(3):331–346, 1990.

- [16] M. Van Barel and A. Bultheel. A new formal approach to the rational interpolation problem. *Numerische Mathematik*, 62:87–122, 1992.
- [17] M. Van Barel, G. Heinig, and P. Kravanja. A stabilized superfast solver for nonsymmetric toeplitz systems. *SIAM Journal on Matrix Analysis and its Applications*, 23(2):494–510, 2001.
- [18] M. Van Barel and P. Kravanja. A stabilized superfast solver for indefinite Hankel systems. *Linear Algebra and Its Applications*, 284(1–3):335–355, 1998.