



**HAL**  
open science

# Local solution clustering for a triangular system of polynomials

Rémi Imbach

► **To cite this version:**

| Rémi Imbach. Local solution clustering for a triangular system of polynomials. 2018. hal-01825708v1

**HAL Id: hal-01825708**

**<https://hal.science/hal-01825708v1>**

Preprint submitted on 28 Jun 2018 (v1), last revised 9 Dec 2019 (v3)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Local solution clustering for a triangular system of polynomials

Rémi Imbach \*  
Courant Institute of Mathematical Sciences  
New York University, USA  
Email: remi.imbach@nyu.edu

June 26, 2018

## Abstract

This report is about finding clusters of complex solutions of triangular systems of polynomial equations. We introduce the local solution clustering problem for a system of polynomial equations, that is grouping all its complex solutions lying in an initial complex domain in clusters smaller than a given real number  $\epsilon > 0$ , and counting the sum of multiplicities of the solutions in each clusters. For triangular systems, we propose a criterion based on the Pellet theorem to count the sum of the multiplicities of the solutions in a cluster. We also propose an algorithm for solving the local solution clustering problem for triangular systems, based on a recent near-optimal algorithm for clustering the complex roots of univariate polynomials. Our algorithm is numeric and certified. We implemented it and compared it with two homotopy solvers for randomly generated triangular systems. Our solver always give correct answers, is often faster than the homotopy solver that gives often correct answers, and sometimes faster than the one that gives sometimes correct results.

## 1 Introduction

This report considers the long-standing problem of finding the complex solutions of a system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  of  $d$  polynomial equations in  $d$  unknowns, where  $\mathbf{z} = (z_1, \dots, z_d)$ ,  $\mathbf{f} : \mathbb{C}^d \rightarrow \mathbb{C}^d$  has components  $f_1, \dots, f_d$  and  $f_i \in \mathbb{C}[z_1, \dots, z_d]$  for  $1 \leq i \leq d$ .

We will say that  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  is *triangular* if  $f_1, \dots, f_d$  satisfy  $f_i \in \mathbb{C}[z_1, \dots, z_i]$  for  $1 \leq i \leq d$ . We are interested here in finding clusters of solutions of triangular systems and counting the sums of multiplicities of solutions in cluster. Solving triangular systems may be seen as a fundamental task in polynomial equations solving, since algebraic approaches (Gröbner basis, CAD, resultants, ...) generally reduce the original system to a triangular system.

Isolating the complex solutions of a polynomial system in an initial domain can be set as follows:

**Local solution isolation problem:**

**Given:** a polynomial map  $\mathbf{f} : \mathbb{C}^d \rightarrow \mathbb{C}^d$ , a box  $\mathbf{B} \subset \mathbb{C}^d$ ,  $\epsilon > 0$

**Output:** a set  $\{\Delta^1, \dots, \Delta^l\}$  of pairwise disjoint polydiscs of radius  $\leq \epsilon$  where:

- each solution of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  in  $\mathbf{B}$  is in a unique  $\Delta^j$ , and
- each  $\Delta^j$  contains a solution of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ .

---

\*Rémi's work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 676541. It has also been supported by NSF Grants # CCF-1563942 and # CCF-1564132.

A *polydisc*  $\Delta$  is a vector  $(\Delta_1, \dots, \Delta_d)$  of complex discs. The *center* of  $\Delta$  is the vector of the centers of its components and the *radius*  $r(\Delta)$  of  $\Delta$  is the max of the radii of its components. If  $\delta$  is any positive real number, we note  $\delta\Delta$  the polydisc  $(\delta\Delta_1, \dots, \delta\Delta_d)$  that has the same center than  $\Delta$  and radius  $\delta r(\Delta)$ . A *box*  $\mathbf{B} \in \mathbb{C}^d$  is a vector  $(B_1, \dots, B_d)$  of complex boxes. The *center* of  $\mathbf{B}$  is the vector of the centers of its components. A box  $\mathbf{B} \in \mathbb{C}^d$  is a *square complex box* if for all  $1 \leq i \leq d$ , the real and the imaginary parts of  $B_i$  have the same widths. The *width* of a *square complex box* is the max of the widths of its components; we note it  $w(\mathbf{B})$ . If  $\delta$  is any positive real number, we note  $\delta\mathbf{B}$  the square complex box  $(\delta B_1, \dots, \delta B_d)$  that has the same center than  $\mathbf{B}$  and width  $\delta w(\mathbf{B})$ .

We introduce three notions to define the local solution clustering problem. Let  $\mathbf{a} \in \mathbb{C}^d$  be a solution of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ . The *multiplicity* of  $\mathbf{a}$  in  $\mathbf{f}$ , also called the *intersection multiplicity* of  $\mathbf{a}$  in  $\mathbf{f}$  is classically defined by localization of rings as in [ZFX11][Def. 1, p. 61]. An equivalent definition uses dual spaces (see [DZ05][Def. 1, p. 117]). We will note it  $m(\mathbf{a}, \mathbf{f})$ .

For any set  $S \subseteq \mathbb{C}^d$ , we note  $Z(S, \mathbf{f})$  the set of solutions of  $\mathbf{f}$  in  $S$ , and  $\#(S, \mathbf{f})$  the sum of multiplicities of solutions of  $\mathbf{f}$  in  $S$ .

**Local solution clustering problem:**

**Given:** a polynomial map  $\mathbf{f} : \mathbb{C}^d \rightarrow \mathbb{C}^d$ , a square complex box  $\mathbf{B} \subset \mathbb{C}^d$ ,  $\epsilon > 0$

**Output:** a set of pairs  $\{(\Delta^1, m^1), \dots, (\Delta^l, m^l)\}$  where:

- the  $\Delta^j$ 's are pairwise disjoint polydiscs of radius  $\leq \epsilon$ ,
- $m^j = \#(\Delta^j, \mathbf{f}) = \#(3\Delta^j, \mathbf{f})$  for all  $1 \leq j \leq l$ , and
- $Z(\mathbf{B}, \mathbf{f}) \subseteq \bigcup_{j=1}^l Z(\Delta^j, \mathbf{f})$ .

**Our contributions** In this report, we propose an algorithm for solving the local solution clustering problem for triangular systems.

To this end, we propose a formula to count the sum of multiplicities in a cluster  $Z(\Delta, \mathbf{f})$  under some conditions on  $Z(\Delta, \mathbf{f})$ . Our formula is derived from a result of [ZFX11] that links the intersection multiplicity of a solution of a triangular systems to multiplicities in fibers.

We introduce the *towers of algebraic clusters* that encode clusters of solutions of triangular systems in stacks (or towers) of clusters of roots of univariate polynomials and show that the so-called  $T_*$ -test introduced in [BSSY17] and based on the Pellet theorem can be used to count the sum of multiplicities of the solutions in a cluster encoded by a tower of algebraic clusters.

Our algorithm to solve the local solution clustering problem for triangular systems is based on a recent clustering algorithm for univariate polynomial (see [BSS+16]). We prove its correctness and its termination.

We implemented and experimented our algorithm; we show with two benchmarks that it compares advantageously with two homotopy solvers: HOM4PS-2.0 that is fast but not robust and Bertini that is more robust but slower.

**Structure of the report** Sec. 2 is about counting the sum of multiplicities of the solution in a cluster. In Sec. 3 we describe our algorithm for clustering solutions of a triangular system, and prove its correctness. Experimental results are proposed in 4.

The rest of the introduction is dedicated to a brief description of related works and to introducing definitions and notations used throughout this report.

## 1.1 Related works

We focus our discussion on polynomial systems, triangular or not, having of  $d$  equations involving  $d$  unknowns and having a zero-dimensional set of solutions. One can divide approaches for solving such polynomials in two categories: the numeric ones and the algebraic ones; the latter category proceeds by making the system to solve triangular, then solving numerically the triangular system.

The numeric solvers using homotopy ([W<sup>+</sup>05]) have demonstrated their superiority in that they can handle systems in higher dimension and with higher degrees than the other approaches. They however rely on a numerical step, the homotopy path tracking, that is not certified in efficient implementations<sup>1</sup>; the results of such implementations come with no guarantee of correctness. There is a recent literature on using certified path tracking methods in homotopy solving and providing a complexity analysis ([BL12, Lai17]).

Concerning algebraic solvers, there are plenty of papers in the literature describing how to transform (non-triangular) system of polynomials into triangular polynomial systems, (called characteristic sets, triangular sets of regular chains [ALM99]) via Gröbner basis computation, Rational Univariate Representation or resultant theory ([ALM99, Rou99]).

Only a few are interested in solving triangular polynomial systems. For systems having only regular solutions, this task reduces to iteratively isolate roots of univariate polynomials in fibers. It however requires to be able to isolate roots in fibers with high precision. In most cases, only the real solutions are sought, and the Descartes' rule of signs combined with subdivision provides an efficient way to do so (see [KRS16] for a recent implementation). Some works address the case where a triangular system has non-regular solutions, *i.e.* solutions with multiplicities greater than 1. In this context, a numerical root isolator may not terminate; an idea exposed in [CGY09] is to use, jointly with a numerical root isolator based on the rule of signs, evaluation bounds for a system to decide that a solution has multiplicity greater than one; remark that this approach does not allow to compute the multiplicity of a solution. The authors of [ZFX11] propose a formula to compute the multiplicity of a solution of a triangular system: the latter multiplicity is the product of the multiplicities of the components of a solution in the fibers. Then, by using square free factorization of univariate polynomials specialized in fibers, they describe an algorithm to retrieve the real solutions of a triangular system with their multiplicities.

Here we “soften” the problem of isolating the solutions of a triangular system of polynomial equations while counting their multiplicities by translating it in the local solution clustering problem. We also search clusters of complex solutions. For this, we leverage of a recent algorithm for solving the *local root clustering problem*, (*i.e.* the univariate case of the local solution clustering problem) proposed in [BSS<sup>+</sup>16]. It is based on subdivision of the initial complex square box, and uses the Pellet theorem combined with Graeffe iterations to count the sum of multiplicities of the roots in a cluster. It does not require the knowledge of the exact coefficients of the polynomial, but instead uses a black-box that, for a given precision  $L \in \mathbb{N}$ ,  $L > 1$ , returns  $L$ -bit approximations of its coefficients. Hence, the polynomial the roots are sought can have any numbers as coefficients, in particular algebraic numbers, that come with such a black-box.

## 1.2 Definitions and notations

Let  $\mathbf{a} = (a_1, \dots, a_d)$  be a point of  $\mathbb{C}^d$ ,  $\Delta$  be the polydisc  $(\Delta_1, \dots, \Delta_d)$ ,  $\mathbf{B}$  be the complex box  $(B_1, \dots, B_d)$ ,  $\mathbf{f} : \mathbb{C}^d \rightarrow \mathbb{C}^d$  be the polynomial map which components are  $(f_1, \dots, f_d)$ , and  $\mathbf{z}$  be

---

<sup>1</sup> we considered here HOM4PS-2.0 and Bertini

$(z_1, \dots, z_d)$ . For any  $1 \leq i \leq d$ , we note:

- $\mathbf{a}^{[i]}$  the point  $(a_1, \dots, a_i) \in \mathbb{C}^i$ ,
- $\Delta^{[i]}$  the polydisc  $(\Delta_1, \dots, \Delta_i)$ ,
- $\mathbf{B}^{[i]}$  the complex box  $(B_1, \dots, B_i)$ ,
- $\mathbf{f}^{[i]} : \mathbb{C}^i \rightarrow \mathbb{C}^i$  the polynomial map which components are  $(f_1, \dots, f_i)$ ,
- $\mathbf{z}^{[i]}$  the vector of  $i$  unknowns  $(z_1, \dots, z_i)$ ,
- $m(\mathbf{a}^{[i]}, \mathbf{f}^{[i]})$  the multiplicity of  $\mathbf{a}^{[i]}$  in  $\mathbf{f}^{[i]}$ .

If  $1 < i \leq d$ , and  $\mathbf{b} = (b_1, \dots, b_{i-1})$  is any point in  $\mathbb{C}^{i-1}$ , we note  $(f_i)_{\mathbf{b}}$  the univariate polynomial in  $\mathbb{C}[z_i]$  obtained by specializing  $f_i$  in  $\mathbf{b}$ . If  $\mathbf{a}$  is a solution of  $\mathbf{f}$ , we call multiplicity of  $a_i$  in  $f_i$  in the fiber  $\mathbf{a}^{[i-1]}$  the multiplicity of the root  $a_i$  of the polynomial  $(f_i)_{\mathbf{a}^{[i-1]}}$ . We note it  $m(a_i, (f_i)_{\mathbf{a}^{[i-1]}})$ .

If  $B \subset \mathbb{C}$  is a square complex box with center  $c$  and width  $w$ , we note  $\Delta(B)$  the disc with center  $c$  and radius  $\frac{3}{4}w$ :  $\Delta(B)$  contains  $B$ . If  $\mathbf{B} = (B_1, \dots, B_d) \subset \mathbb{C}^d$  is a square complex box, We note  $\Delta(\mathbf{B})$  the polydisc  $(\Delta(B_1), \dots, \Delta(B_d))$ :  $\Delta(\mathbf{B})$  contains  $\mathbf{B}$ .

A polydisk  $\Delta$  is called an *isolator* if  $\#(\Delta, \mathbf{f}) = \#(3\Delta, \mathbf{f})$ . Any non-empty set of the form  $Z(\Delta, \mathbf{f})$  is called a *cluster* of solutions of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ , and it is a *natural* cluster if  $\Delta$  is an isolator.

We call  $L$ -bit approximation of  $a \in \mathbb{C}$  a dyadic complex number  $\tilde{a}$  that coincides with  $a$  to  $L$  bits after the binary point, that is  $|a - \tilde{a}| \leq 2^{-L}$ . We call  $L$ -bit approximation of  $\mathbf{a} = (a_1, \dots, a_i) \in \mathbb{C}^i$  a vector  $\tilde{\mathbf{a}} \in \mathbb{C}^i$  such that  $\tilde{a}_j$  is an  $L$ -bit approximation of  $a_j$  for  $1 \leq j \leq i$ . If  $g$  is a univariate polynomial, we call  $L$ -bit approximation of  $g$  a univariate polynomial  $\tilde{g}$  which coefficients are  $L$ -bit approximations of the coefficients of  $g$ .

## 2 Sum of multiplicities in clusters of solutions

[ZFX11] links the multiplicity of the solutions of a triangular system to multiplicities in fibers. In subsec. 2.2, we elaborate on this result and define some conditions under which it can be applied. We introduce in subsec. 2.3 the notions of algebraic clusters and towers of algebraic clusters that are a special instance of clusters of solutions of triangular systems verifying the above conditions.

The  $T_*$ -test defined in [BSSY17] is based on the Pellet theorem and Graeffe iterations, and allows to compute the sum of multiplicities of the roots of a univariate polynomial in a disc. In subsec. 2.4 we show how this test to compute towers of algebraic clusters by computing sums of multiplicities of roots in a disc in a fiber of multivariate polynomials.

We first introduce in subsec. 2.1 two thought examples to illustrate our discussion.

### 2.1 Two examples

Let  $\delta$  be a strictly positive integer. We define the triangular system  $\mathbf{g}(\mathbf{z}) = 0$  where  $\mathbf{g} = (g_1, g_2)$  as follows:

$$\begin{cases} (z_1 - 2^{-\delta})(z_1 + 2^{-\delta}) = 0 \\ (z_2 - 2^{2\delta} z_1^2) z_2 = 0 \end{cases} \quad (1)$$

$\mathbf{g}(\mathbf{z}) = 0$  has 4 solutions that are real:  $\mathbf{a}^1 = (2^{-\delta}, 0)$ ,  $\mathbf{a}^2 = (2^{-\delta}, 1)$ ,  $\mathbf{a}^3 = (-2^{-\delta}, 1)$  and  $\mathbf{a}^4 = (-2^{-\delta}, 0)$ .

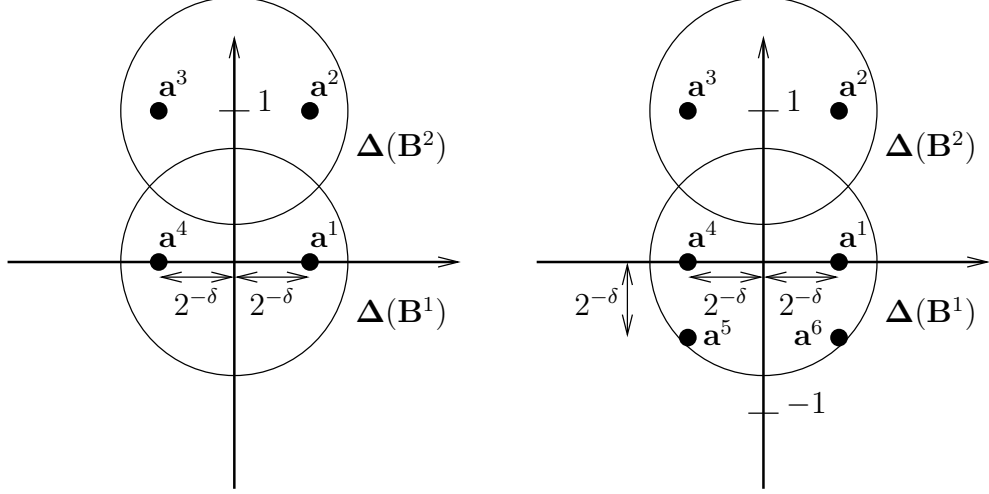


Figure 1: Left: the solutions of  $\mathbf{g}(\mathbf{z}) = 0$  defined in Eq. 1, with  $\delta = 1$ . Right: the solutions of  $\mathbf{h}(\mathbf{z}) = 0$  defined in Eq. 2, with  $\delta = 1$ .  $\mathbf{B}^1$  (resp.  $\mathbf{B}^2$ ) is the square complex box of  $\mathbb{C}^2$  with center  $(0,0)$  (resp.  $(0,1)$ ) and width  $2 * 2^{-\delta}$ .

We define the triangular system  $\mathbf{h}(\mathbf{z}) = 0$  where  $\mathbf{h} = (h_1, h_2)$  as follows:

$$\begin{cases} (z_1 - 2^{-\delta})^2(z_1 + 2^{-\delta}) = 0 \\ (z_2 - 2^\delta z_1^2)^2(z_2 - 1)z_2 = 0 \end{cases} \quad (2)$$

$\mathbf{h}(\mathbf{z}) = 0$  has 5 solutions that are real:  $\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3, \mathbf{a}^4, \mathbf{a}^5 = (-2^{-\delta}, -2^{-\delta})$  and  $\mathbf{a}^6 = (2^{-\delta}, -2^{-\delta})$ . For  $1 \leq i \leq 6$ , we will note  $\mathbf{a}^i = (a_1^i, a_2^i)$ . The solutions of both  $\mathbf{g}(\mathbf{z}) = 0$  and  $\mathbf{h}(\mathbf{z}) = 0$  are depicted in Fig. 1.

## 2.2 Sum of multiplicities in a cluster

Let us recall the following result that allows to count the multiplicity of a solution of a triangular system when knowing multiplicities in fibers.

**Theorem 1** ([ZFX11]). *The multiplicity of a solution  $\mathbf{a} = (a_1, \dots, a_d)$  of  $\mathbf{f}(\mathbf{z}) = 0$  where  $\mathbf{f} = (f_1, \dots, f_d)$  is  $\prod_{i=1}^d m_i$ , where  $m_1$  is the multiplicity of the root  $a_1$  in  $f_1$  and for  $2 \leq i \leq d$ ,  $m_i$  is the multiplicity of the root  $a_i$  in  $(f_i)_{\mathbf{a}^{[i-1]}}$ .*

We apply this result to compute the multiplicities of solutions of the systems  $\mathbf{g}(\mathbf{z}) = 0$  and  $\mathbf{h}(\mathbf{z}) = 0$  respectively defined in Eq. 1 and Eq. 2.  $\mathbf{a}^1$  has multiplicity 1 in  $\mathbf{g}$ :  $m(\mathbf{a}^1, \mathbf{g}) = m(a_1^1, g_1) * m(a_2^1, (g_2)_{a_1^1}) = 1 * 1$ .  $\mathbf{a}^1$  has multiplicity 2 in  $\mathbf{h}$ :  $m(\mathbf{a}^1, \mathbf{h}) = m(a_1^1, h_1) * m(a_2^1, (h_2)_{a_1^1}) = 2 * 1$ . With the same computations, we have:  $\mathbf{a}^2$  has multiplicity 1 in  $\mathbf{g}$  and 2 in  $\mathbf{h}$ ,  $\mathbf{a}^3$  and  $\mathbf{a}^4$  have multiplicity 1 in  $\mathbf{g}$  and in  $\mathbf{h}$ ,  $\mathbf{a}^5$  has multiplicity 2 in  $\mathbf{h}$  and  $\mathbf{a}^6$  has multiplicity 4 in  $\mathbf{h}$ .

We derive the following formula from Thm. 1:

**Corollary 2** (of Thm. 1). *With the notations of Thm. 1, one has:  $\forall 1 < i \leq d, m(\mathbf{a}^{[i]}, \mathbf{f}^{[i]}) = m(a_i, (f_i)_{\mathbf{a}^{[i-1]}}) \times m(\mathbf{a}^{[i-1]}, \mathbf{f}^{[i-1]})$*

We generalize thm. 1 to compute the sum of multiplicities of the solutions in a cluster.

**Theorem 3.** Let  $Z(\Delta, \mathbf{f})$  be a cluster of solutions with  $\Delta = (\Delta_1, \dots, \Delta_d)$  and  $\mathbf{f} = (f_1, \dots, f_d)$ . Let the integers  $m_1, \dots, m_d$  be such that:

- (i)  $m_1 = \#(\Delta_1, f_1)$  ( $m_1$  is the sum of multiplicities of roots of  $f_1$  in  $\Delta_1$ ), and
- (ii)  $\forall \mathbf{a} \in Z(\Delta, \mathbf{f}), \forall 1 < i \leq d, m_i = \#(\Delta_i, (f_i)_{\mathbf{a}^{[i-1]}})$  ( $m_i$  is the sum of multiplicities of the roots of  $(f_i)_{\mathbf{a}^{[i-1]}}$  for any solution  $\mathbf{a}$  in the cluster).

Then  $\#(\Delta, \mathbf{f}) = m_1 \times m_2 \times \dots \times m_d$ .

Before proving this result, we illustrate it with the systems  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$  and  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$  defined in Eq. 1 and Eq. 2. Let  $\mathbf{B}^1 = (B_1^1, B_2^1)$  be the square complex box centered in  $(0, 0)$  having width  $2 * 2^{-\delta}$  and  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ ,  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$  be defined as in Eq. 1 and Eq. 2. We apply Thm. 3 to compute multiplicities in  $\Delta(\mathbf{B}^1)$ :  $\#(\Delta(\mathbf{B}^1), \mathbf{g}) = 2$ . Indeed,  $Z(\Delta(\mathbf{B}^1), \mathbf{g}) = \{\mathbf{a}^1, \mathbf{a}^4\}$ . If  $m_1 = 2$  and  $m_2 = 1$ , one has:  $m_1 = \#(\Delta(B_1^1), g_1)$  and the condition (i) of Thm. 3 is satisfied. Then  $m_2 = \#(\Delta(B_2^1), (g_2)_{\mathbf{a}_1^1}) = \#(\Delta(B_2^1), (g_2)_{\mathbf{a}_1^4})$  and the condition (ii) of Thm. 3 is satisfied.

One also has  $\#(\Delta(\mathbf{B}^1), \mathbf{h}) = 9$ ; let  $m_1 = 3$  and  $m_2 = 3$ . First,  $Z(\Delta(\mathbf{B}^1), \mathbf{h}) = \{\mathbf{a}^1, \mathbf{a}^4, \mathbf{a}^5, \mathbf{a}^6\}$ , and recall that  $a_1^1 = a_1^6 = 2^{-\delta}$  and  $a_1^4 = a_1^5 = -2^{-\delta}$ . One has  $m_1 = \#(\Delta(B_1^1), h_1)$  and the condition (i) of Thm. 3 is satisfied. Then  $m_2 = \#(\Delta(B_2^1), (h_2)_{\mathbf{a}_1^1}) = \#(\Delta(B_2^1), (h_2)_{\mathbf{a}_1^4})$ , and the condition (ii) of Thm. 3 is satisfied.

Let  $\mathbf{B}^2 = (B_1^2, B_2^2)$  be the square complex box centered in  $(0, 1)$  having width  $2 * 2^{-\delta}$ . With the same reasoning,  $\#(\Delta(\mathbf{B}^2), \mathbf{g}) = 2 * 1$ , and  $\#(\Delta(\mathbf{B}^2), \mathbf{h}) = 3 * 1$ . The real parts of  $\Delta(\mathbf{B}^1)$  and  $\Delta(\mathbf{B}^2)$  are depicted in Fig. 1.

**Proof of Thm 3:** Let  $1 < i \leq d$ . We note  $\Pi_{i-1} : \mathbb{C}^i \rightarrow \mathbb{C}^{i-1}$  the projection of a point on its  $i - 1$  first components, and  $\Pi^i : \mathbb{C}^i \rightarrow \mathbb{C}$  the projection of a point on its last component. We extend these maps to subsets of  $\mathbb{C}^i$ : if  $S \subseteq \mathbb{C}^i$ ,  $\Pi_{i-1}(S)$  and  $\Pi^i(S)$  are the sets which elements are the projections of elements of  $S$ .

Suppose that  $\Pi_{i-1}(Z(\Delta^{[i]}, \mathbf{f}^{[i]}))$  has  $l$  distinct points  $\mathbf{a}^1, \dots, \mathbf{a}^l$  with respective multiplicities  $m^1, \dots, m^l$  in  $\mathbf{f}^{[i-1]}$  and let  $m = \sum_{j=1}^l m^j$ .  $Z(\Delta^{[i]}, \mathbf{f}^{[i]})$  can be partitioned in  $l$  sets  $S^1, \dots, S^l$  such that a solution  $\mathbf{a} \in Z(\Delta^{[i]}, \mathbf{f}^{[i]})$  is in  $S^j$  if and only if  $\Pi_{i-1}(\mathbf{a}) = \mathbf{a}^j$ .

Remark that  $Z(\Pi^i(S^j), (f_i)_{\mathbf{a}^j}) = Z(\Delta_i, (f_i)_{\mathbf{a}^j})$ ; then  $\#(\Pi^i(S^j), (f_i)_{\mathbf{a}^j}) = m_i$  from hypothesis (ii) of Thm 3. We can now apply corollary 2 and obtain  $\#(S^j, (f_i)_{\mathbf{a}^j}) = \#(\Delta_i, (f_i)_{\mathbf{a}^j}) \times m(\mathbf{a}^j, \mathbf{f}^{[i]})$ . We now sum the  $\#(S^j, (f_i)_{\mathbf{a}^j})$  over  $j$  and obtain:  $\#(\Delta^{[i]}, \mathbf{f} \Delta^{[i]}) = m_i \times m$ .

We use this argument in an inductive scheme to obtain the formula of Thm. 3. □

### 2.3 Towers of algebraic clusters

We introduce here the notions of algebraic clusters and towers of algebraic clusters. Towers of algebraic clusters are special instances of clusters satisfying conditions of Thm. 3 that can be computed with the  $T_*$ -test of [BSSY17].

**Definition 4** (Algebraic clusters). We call algebraic cluster a triplet  $(B, m, g)$  where  $B \subset \mathbb{C}$  is a square complex box,  $m \geq 1$  is an integer and  $g$  is a univariate polynomial, such that  $\#(B, g) = \#(\Delta(B), g) = m$  and  $Z(\Delta(B), g)$  is a natural cluster.

Consider  $g_1$  and  $h_1$  respectively defined in Eq. 1 and in Eq. 2. Let  $B_1^1$  be the square complex box centered in 0 with width  $2 * 2^{-\delta}$ .  $(B_1^1, 2, g_1)$  and  $(B_1^1, 3, h_1)$  are both algebraic clusters.

We introduce now Towers of algebraic clusters representing clusters of solutions of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  as stacks of clusters of solutions.

**Definition 5** (Towers of Algebraic Clusters, TAC). *We call  $d$ -dimensional tower of algebraic clusters, or  $d$ -TAC, a triple  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  where  $\mathbf{B} = (B_1, \dots, B_d)$  and the  $B_i$ 's are square complex boxes,  $\mathbf{m} = (m_1, \dots, m_d)$  and the  $m_i$ 's are integers greater than 1, and  $\mathbf{f} = (f_1, \dots, f_m)$  such that:*

- (1)  $(B_1, m_1, f_1)$  is an algebraic cluster,
- (2)  $\forall 2 \leq i \leq d, (B_i, m_i, (f_i)_{\mathbf{b}^{[i-1]}})$  is an algebraic cluster, where  $\mathbf{b}$  is any point in  $\Delta(\mathbf{B})$ .

If the triple  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is a  $d$ -TAC, then for any  $1 \leq i \leq d$ ,  $(\mathbf{B}^{[i]}, \mathbf{m}^{[i]}, \mathbf{f}^{[i]})$  is an  $i$ -TAC.

Let  $\mathbf{B}^1$  and  $\mathbf{B}^2$  be defined as in subsec. 2.2, and  $\mathbf{g}, \mathbf{h}$  be defined as in Eq. 1 and Eq. 2. There exist no 2-TAC for  $\mathbf{g}$  having  $\mathbf{B}^1$  or  $\mathbf{B}^2$  as box. To see this, remark that  $-2^{-\delta}$ , 0 and  $2^{-\delta}$  are three points of  $\mathbf{B}_1^1$ , and consider the three polynomials  $(g_2)_{-2^{-\delta}}$ ,  $(g_2)_0$  and  $(g_2)_{2^{-\delta}}$ .  $(g_2)_{-2^{-\delta}}$  and  $(g_2)_{2^{-\delta}}$  have each 1 root of multiplicity 1 in  $\mathbf{B}_1^1$  while  $(g_2)_0 = z_2^2$  has 1 root of multiplicity 2 in  $\mathbf{B}_1^1$ , hence it is not possible to satisfy condition (2) of def. 5. In the case of  $\mathbf{B}^2$ ,  $(g_2)_{-2^{-\delta}}$  and  $(g_2)_{2^{-\delta}}$  have both 1 root of multiplicity 1 in  $\mathbf{B}_2^2$  while  $(g_2)_0$  has no root in  $\mathbf{B}_2^2$ .

In contrast, if  $\delta \geq 3$ ,  $(\mathbf{B}^1, (3, 3), \mathbf{h})$  and  $(\mathbf{B}^2, (3, 1), \mathbf{h})$  are 2-TACs. As it has been remarked above,  $(B_1^1, 3, h_1)$  is an algebraic cluster and since  $B_1^2 = B_1^1$  so is  $(B_1^2, 3, h_1)$ . Consider now the polynomial  $h_2(z_1, z_2) = (z_2 - z_1)^2(z_2 - 1)z_2$ . If  $z_2 \in \delta(B_2^1)$  then  $z_2 < \frac{3}{16} < 1$  and for any  $z_1 \in B_1^1$ ,  $h_2$  has 3 roots counted with multiplicity in  $\delta(B_2^1)$  and in  $3\delta(B_2^1)$ . Hence for any  $b \in B_1^1$ ,  $(B_2^1, 3, (h_2)_b)$  is an algebraic cluster, and  $(\mathbf{B}^1, (3, 3), \mathbf{h})$  is a 2-TAC. It is easy to apply the same argument to show that  $(\mathbf{B}^2, (3, 1), \mathbf{h})$  is a 2-TAC.

**Theorem 6.** *Let  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  be a  $d$ -TAC where  $\mathbf{m} = (m_1, \dots, m_d)$ . Then  $Z(\Delta(\mathbf{B}), \mathbf{f})$  is a natural cluster of solutions of  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ , and  $\#(\mathbf{B}, \mathbf{f}) = \#(\Delta(\mathbf{B}), \mathbf{f}) = \prod_{i=1}^d m_i$ .*

**Proof of Thm 6:** We give the proof for  $d = 2$ . It is straightforward to generalize it. Let  $((B_1, B_2), (m_1, m_2), (f_1, f_2))$  be a 2-TAC. Then  $(B_1, m_1, f_1)$  is an algebraic cluster:  $\#(B_1, f_1) = \#(\Delta(B_1), f_1) = \#(3\Delta(B_1), f_1) = m_1$ . Let  $Z(B_1, f_1) = \{a^1, \dots, a^l\}$ .

$(B_2, m_2, (f_2)_a)$  is an algebraic cluster for any  $a \in \Delta(B_1)$ , and in particular for  $a^1, \dots, a^l$ . Hence for any  $1 \leq i \leq l$ ,  $(B_2, m_2, (f_2)_{a^i})$  is an algebraic cluster:  $\#(B_2, (f_2)_{a^i}) = \#(\Delta(B_2), (f_2)_{a^i}) = \#(3\Delta(B_2), (f_2)_{a^i}) = m_2$ . Let  $Z(B_2, (f_2)_{a^i}) = \{a^{i,1}, \dots, a^{i,l_i}\}$ .

It is a consequence of Thm. 3 that  $\#(\Delta((B_1, B_2)), (f_1, f_2)) = \prod_{i=1}^2 m_i = m_1 * m_2$ . Then  $\forall 1 \leq i \leq l, \forall 1 \leq j \leq l_i, (a^i, a^{i,j})$  is a solution of  $(f_1, f_2)$  and is in  $Z((B_1, B_2), (f_1, f_2))$  but also in  $Z(\Delta(B_1, B_2), (f_1, f_2))$  and in  $Z(3\Delta(B_1, B_2), (f_1, f_2))$ . Finally,  $Z(3\Delta(B_1, B_2), (f_1, f_2))$  do not contain other solution than the  $(a^i, a^{i,j})$ 's because  $Z(B_1, f_1)$  and  $Z(B_2, (f_2)_{a^i})$  are natural clusters for any  $i = 1, \dots, l$ .  $\square$

## 2.4 Computing towers of algebraic clusters with the $T_*$ -test.

We show here how to use the  $T_*$ -test defined in [BSSY17] to check that a triplet  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is a tower of algebraic clusters. To check the condition (2) of Def. 5, for a given  $1 < i \leq d$ , one has to count the sum of multiplicities of roots of all the univariate polynomials  $(f_i)_{\mathbf{b}}$  that are the specializations of the multivariate polynomial  $f_i$  for any  $\mathbf{b}$  in the polydisk  $\Delta(\mathbf{B}^{[i-1]})$ .

The  $T_*$ -test counts the sum of multiplicities of the roots of a univariate polynomial  $g$  in a complex disc  $\Delta$ .  $g$  is given to the  $T_*$ -test as a black-box that, for a given precision  $L \in \mathbb{N}$ ,  $L > 1$ , returns



---

**Algorithm 1** *getAppTAC*( $L, \mathbf{B}$ )

---

**Input:** An integer  $L > 1$ , a square complex box  $\mathbf{B} \subset \mathbb{C}^i$ .

**Output:** A pair  $(\mathbf{c}, success)$  where  $\mathbf{c}$  is an  $L$ -bit approximation of the center of  $\mathbf{B}$  and  $success \in \{0, 1\}$ . If  $success = 1$ ,  $\mathbf{c}$  is an  $L$ -bit approximation of any  $\mathbf{b}$  in  $\Delta(\mathbf{B})$ .

- 1:  $(\mathbf{c}, success) \leftarrow$  ( an  $L$ -bit approximation of the center of  $\mathbf{B}$ , 0)
  - 2: **if**  $r(\Delta(\mathbf{B})) \leq 2^{-L}$  **then**
  - 3:      $success \leftarrow 1$
  - 4: **return**  $(\mathbf{c}, success)$
- 

---

**Algorithm 2** *getAppFib*( $L, \mathbf{B}, g$ )

---

**Input:** An integer  $L > 1$ , a square complex box  $\mathbf{B} \subset \mathbb{C}^{i-1}$ , a polynomial  $g \in \mathbb{C}[z_1, \dots, z_i]$ .

**Output:** A pair  $(\tilde{g}, success)$  where  $\tilde{g}$  is an  $L$ -bit approximation of  $(g)_{\mathbf{c}}$  with  $\mathbf{c}$  the center of  $\mathbf{B}$ , and  $success \in \{0, 1\}$ . If  $success = 1$ ,  $\tilde{g}$  is an  $L$ -bit approximation of  $(g)_{\mathbf{b}}$  for any  $\mathbf{b}$  in  $\Delta(\mathbf{B})$ .

- 1:  $L' \leftarrow L$
  - 2: **repeat**
  - 3:      $(\mathbf{c}, success) \leftarrow$  *getAppTAC*( $L', \mathbf{B}$ )
  - 4:      $\tilde{g} \leftarrow$   $L'$ -bit approximation of  $g$
  - 5:      $\tilde{g} \leftarrow$  substitute  $\mathbf{z}^{[i-1]}$  with  $\mathbf{c}$  in  $\tilde{g}$
  - 6:      $L' \leftarrow 2 * L'$
  - 7: **until**  $\tilde{g}$  is a  $L$ -bit approximation or  $success = 0$
  - 8: **return**  $(\tilde{g}, success)$
- 

an  $L$ -bit approximation of  $g$ . Hence, the coefficients of  $g$  can be any number, particularly algebraic numbers, that come with such a black-box.

We provide in Algo. 1 the black-box *getAppTAC*( $L, \mathbf{B}$ ) computing for a given precision  $L$  and a square complex box  $\mathbf{B}$  an  $L$ -bit approximation of the center of  $\mathbf{B}$ . If the radius of  $\Delta(\mathbf{B})$  is less than  $2^{-L}$ , then an  $L$ -bit approximation of the center of  $\mathbf{B}$  is an  $L$ -bit approximation of any  $\mathbf{b} \in \Delta(\mathbf{B})$ . *getAppTAC*( $L, \mathbf{B}$ ) fails when  $r(\Delta(\mathbf{B})) > 2^{-L}$ ; if it succeeds it returns an  $L$ -bit approximation of any  $\mathbf{b} \in \Delta(\mathbf{B})$ .

The procedure *getAppFib*( $L, \mathbf{B}, g$ ) defined in Algo. 2 computes, for a precision  $L$ , a square complex box  $\mathbf{B} \subset \mathbb{C}^{i-1}$  and  $g \in \mathbb{C}[\mathbf{z}^{[i]}]$ , an  $L$ -bit approximation  $\tilde{g}$  of  $g$  specialized in the center of  $\mathbf{B}$ . For this, it computes an  $L'$ -bit approximation, with  $L' \geq L$ , of the center of  $\mathbf{B}$  with *getAppTAC*( $L', \mathbf{B}$ ), and fails if the latter procedure fails. As a consequence, if *getAppFib*( $L, \mathbf{B}, g$ ) succeeds, its output is an  $L$ -bit approximation of  $(g)_{\mathbf{b}}$  for any  $\mathbf{b} \in \Delta(\mathbf{B})$ .

We now consider a version of the  $T_*$ -test noted  $T_*(\Delta, getApp)$  modified as follows. The black-box *getApp* given in input can fail. If the latter situation happens,  $T_*(\Delta, getApp)$  returns  $-1$ . Otherwise,  $T_*(\Delta, getApp)$  returns  $-1$  or a positive number as defined in [BSSY17]. If it returns a positive integer  $m$ , then  $\Delta$  contains  $m$  roots counted with multiplicities (of the polynomial approximated by the black-box *getApp*).

The following assertion holds:

**Lemma 7.** *Let  $\mathbf{B} \subset \mathbb{C}^{i-1}$  be a square complex box,  $\Delta \subset \mathbb{C}$  be a disk and  $g$  a polynomial in  $\mathbb{C}[\mathbf{z}^{[i]}]$ . If  $T_*(\Delta, getAppFib(\cdot, \mathbf{B}, g)) = m \geq 0$ , then  $\#(\Delta, (g)_{\mathbf{b}}) = m$  for any  $\mathbf{b} \in \Delta(\mathbf{B})$ .*

Let *getAppPol*( $L, g$ ) be a black box returning an  $L$ -bit approximation of  $g \in \mathbb{C}[z_1]$  (it never

fails). The following proposition is a consequence of Lem. 7 and states how to check that the triple  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is a TAC.

**Proposition 8.** *Let  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  be a triple where  $\mathbf{B} = (B_1, \dots, B_d)$ ,  $\mathbf{m} = (m_1, \dots, m_d)$  and  $\mathbf{f} = (f_1, \dots, f_f)$  satisfy:*

- (i)  $T_*(\Delta(B_1), \text{getAppPol}(\cdot, f_1)) = T_*(3\Delta(B_1), \text{getAppPol}(\cdot, f_1)) = m_1$ ,
- (ii)  $\forall 2 \leq i \leq d, T_*(\Delta(B_i), \text{getAppFib}(\cdot, \mathbf{B}^{[i-1]}, f_i)) = T_*(3\Delta(B_i), \text{getAppFib}(\cdot, \mathbf{B}^{[i-1]}, f_i)) = m_i$ .

Then  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is a  $d$ -TAC.

### 3 Clustering the solutions of a triangular system

We build on the clustering algorithm of [BSS<sup>+</sup>16] a recursive algorithm that solves the local solution clustering problem for triangular systems. We suppose the existence of the procedure *ccluster* specified in Lem. 10 and using the  $T_*$ -test, that implements the clustering algorithm of [BSS<sup>+</sup>16].

In Algo. 3 we propose a recursive procedure *clusterTriSys*( $\mathbf{f}, \mathbf{B}, \epsilon$ ) solving the local solution clustering problem for  $\mathbf{f}, \mathbf{B}, \epsilon$  where  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  is a triangular systems, as asserted in Prop. 9 below.

**Proposition 9.** *Let  $d > 0$  and  $1 \leq i \leq d$  be two integers,  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  be a triangular system where  $\mathbf{f} = (f_1, \dots, f_d)$ ,  $\mathbf{B} = (B_1, \dots, B_d) \subset \mathbb{C}^d$  be a square complex box and  $\epsilon \in \mathbb{R}$  s.t.  $0 < \epsilon < \frac{1}{2}$ .*

*The procedure *clusterTriSys*( $\mathbf{f}^{[i]}, \mathbf{B}^{[i]}, \epsilon$ ) defined in Algo. 3 terminates and returns a list  $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}^{[i]}) \mid 1 \leq j \leq l\}$  of  $i$ -TACs. Letting  $\mathbf{m}^j = (m_1^j, \dots, m_i^j)$  for  $1 \leq j \leq l$ , the set  $\{Z(\Delta(\mathbf{B}^j), \mathbf{f}^{[i]}) \mid 1 \leq j \leq l\}$  satisfies:*

- (a)  $\Delta(\mathbf{B}^j)$  are pairwise disjoint polydiscs of radius  $\leq \epsilon$ ,
- (b)  $\Pi_{k=1}^i m_k^j = \#(\Delta(\mathbf{B}^j), \mathbf{f}^{[i]}) = \#(3\Delta(\mathbf{B}^j), \mathbf{f}^{[i]})$  for all  $j$ ,
- (c)  $Z(\mathbf{B}^{[i]}, \mathbf{f}^{[i]}) \subseteq \bigcup_{1 \leq j \leq l} Z(\Delta(\mathbf{B}^j), \mathbf{f}^{[i]}) \subseteq Z(2\mathbf{B}^{[i]}, \mathbf{f}^{[i]})$ .

If we prove that Prop. 9 holds for  $i = d$ , then Algo. 3 computes a solution of the local solution clustering problem.

We prove this by induction in Subsec. 3.4. Before describing our main algorithm, we specify the procedure *ccluster*:

**Lemma 10** (see [BSS<sup>+</sup>16]). *Let  $g$  be a univariate polynomial and *getApp*( $L, g$ ) be a black-box computing an  $L$ -bit approximation of  $g$  for any  $L > 1$ ,  $B \subset \mathbb{C}$  be a square complex box and  $\epsilon \in \mathbb{R}$  such that  $0 < \epsilon < 1$ . *ccluster*(*getApp*( $\cdot, g$ ),  $B, \epsilon$ ) terminates and computes a set  $\{(B^j, m^j) \mid 1 \leq j \leq l\}$  satisfying:*

1. the  $\Delta(B^j)$ 's are pairwise disjoint discs with of radius less than  $\epsilon$ ,
2.  $m^j = \#(\Delta(B^j), g) = \#(3\Delta(B^j), g)$ , and
3.  $Z(B, g) \subseteq \bigcup_j Z(B^j, g) \subseteq Z(2B, g)$ .

The procedure  $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$  defined in Algo. 3 computes  $d$ -TACs for the system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ . It first computes  $d - 1$ -TACs for the system  $\mathbf{f}^{[d-1]}(\mathbf{z}^{[d-1]}) = \mathbf{0}$  with the recursive call in step 3. This step provides a list  $\mathcal{S} = \{(\mathbf{B}_*^j, m_*^j, \mathbf{f}^{[d-1]}) | 1 \leq j \leq l'\}$  of  $d - 1$ -TACs. Then for each  $d - 1$ -TAC  $(\mathbf{B}_*^j, m_*^j, \mathbf{f}^{[d-1]})$  of  $\mathcal{S}$ , the clusters of  $(f)_d$  specialized in any point of  $\mathbf{B}_*^j$  are computed in step 5. While performing this step,  $(\mathbf{B}_*^j, m_*^j, \mathbf{f}^{[d-1]})$  is possibly split into several TACs. This is performed by Algo. 5 and Algo. 6 described in subsec. 3.2 and subsec. 3.3.

The terminal case of Algo. 3 is the case where  $d = 1$ , *i.e.*  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  is a system of one polynomial equation. This case is discussed in subsec. 3.1.

---

**Algorithm 3**  $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$

---

**Input:** A triangular system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  where  $\mathbf{f} = (f_1, \dots, f_d)$ , a square complex box  $\mathbf{B} = (B_1, \dots, B_d) \subset \mathbb{C}^d$ , a real number  $\epsilon \in \mathbb{R}$  s.t.  $0 < \epsilon < \frac{1}{2}$ .

**Output:** A list  $\{(\mathbf{B}^j, m^j, \mathbf{f}) | 1 \leq j \leq l\}$  of  $d$ -TACs .

- 1:  $\mathcal{R} \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$
  - 2: **if**  $d > 1$  **then**
  - 3:      $\mathcal{S} \leftarrow clusterTriSys(\mathbf{f}^{[d-1]}, \mathbf{B}^{[d-1]}, \epsilon)$  *//recursive call*
  - 4:     **for**  $(\mathbf{B}^{cur}, m^{cur}, \mathbf{f}^{[d-1]}) \in \mathcal{S}$  **do**
  - 5:          $\mathcal{R}_{temp} \leftarrow clusterPolInFiber(f_d, (\mathbf{B}^{cur}, m^{cur}, \mathbf{f}^{[d-1]}), B_d, \epsilon)$  *//see Algo. 5*
  - 6:          $append(\mathcal{R}, \mathcal{R}_{temp})$
  - 7: **else** *//d = 1*
  - 8:      $\mathcal{R} \leftarrow clusterPol(f_1, B_1, \epsilon)$  *//the terminal case: see Algo. 4*
  - 9: **return**  $\mathcal{R}$
- 

### 3.1 The terminal case

The terminal case, *i.e.* finding clusters of roots of  $f_1 \in \mathbb{C}[z_1]$  of size less than  $\epsilon$  in the square complex box  $B \subset \mathbb{C}$  is addressed by calling  $ccluster(getAppPol(\cdot, f_1), B, \epsilon)$

The procedure  $clusterPol(g, B, \epsilon)$  defined in Algo. 4 calls  $ccluster$  and for each pair  $(B^j, m^j)$  in the output of  $ccluster$ , forms the triplet  $(\mathbf{B}^j, m^j, (g))$  as in step 4. of Algo. 4.

---

**Algorithm 4**  $clusterPol(g, B, \epsilon)$

---

**Input:** A polynomial  $g \in \mathbb{C}[x_1]$ , an initial box  $B$ , a real number  $\epsilon \in \mathbb{R}$  s.t.  $0 < \epsilon < \frac{1}{2}$ .

**Output:** A list  $\{(\mathbf{B}^j, m^j, \mathbf{f}) | 1 \leq j \leq l\}$ , with  $\mathbf{f} = (g)$ , of 1-TACs.

- 1:  $\mathcal{R} \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$
  - 2:  $\mathcal{S} \leftarrow ccluster(getAppPol(\cdot, g), B, \epsilon)$
  - 3: **for**  $(B^j, m^j) \in \mathcal{S}$  **do**
  - 4:      $\mathcal{R} = \mathcal{R} \cup \{((B^j), (m^j), (g))\}$
  - 5: **return**  $\mathcal{R}$
- 

### 3.2 The non-terminal case

We consider now the non-terminal case, that is, be given an  $i - 1$ -TAC  $(\mathbf{B}, m, \mathbf{f}^{[i-1]})$  where  $1 < i \leq d$ , compute the clusters of solutions of  $f_i$  specialized in any point of  $\mathbf{B}$ .

This case is addressed in the procedure *clusterPolInFiber* defined in Algo. 5 by calling the clustering procedure *ccluster*. For this, we define the black-box *getAppFib2* (see Algo. 6 in subsec. 3.3). This black-box computes an  $L$ -bit approximation of  $f_i$  specialized in any point of  $\Delta(\mathbf{B})$ . It never fails in computing such an approximation: if the radius of  $\Delta(\mathbf{B})$  is too big to do so, the  $i-1$ -TAC  $(\mathbf{B}, m, \mathbf{f}^{[i-1]})$  is shrunk and possibly split in several  $i-1$ -TACs that are appended to a working list  $\mathcal{L}$ .

*clusterPolInFiber* pops the working list  $\mathcal{L}$  initialized to  $\{(\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i-1]})\}$  and processes the  $i-1$ -TACs until  $\mathcal{L}$  is empty.

---

**Algorithm 5** *clusterPolInFiber*( $g, (\mathbf{B}, \mathbf{m}, \mathbf{f}), B, \epsilon$ )

---

**Input:** A polynomial  $g \in \mathbb{C}[\mathbf{z}^{[i]}]$ , an  $i-1$ -TAC  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ , a square complex box  $B \subset \mathbb{C}$ , a real number  $\epsilon \in \mathbb{R}$  s.t.  $0 < \epsilon < \frac{1}{2}$ .

**Output:** A list  $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}') \mid 1 \leq j \leq l\}$ , with  $\mathbf{f}' = (\mathbf{f}, g)$ , of  $i$ -TACs.

```

1:  $\mathcal{L} \leftarrow \{(\mathbf{B}, \mathbf{m}, \mathbf{f})\}$ 
2:  $\mathcal{R} \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$ 
3: while  $\mathcal{L} \neq \emptyset$  do
4:    $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f}) \leftarrow pop(\mathcal{L})$ 
5:    $getApp \leftarrow getAppFib2(., g, (\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f}), \mathcal{L})$  //see Algo. 6
6:    $\mathcal{S} \leftarrow ccluster(getApp, B, \epsilon)$  //  $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$  and  $\mathcal{L}$  are possibly modified
7:   for  $(B^j, m^j) \in \mathcal{S}$  do
8:      $\mathcal{R} = \mathcal{R} \cup \{((\mathbf{B}^{cur}, B^j), (\mathbf{m}^{cur}, m^j), (\mathbf{f}, g))\}$ 
9: return  $\mathcal{R}$ 

```

---

### 3.3 The black box

Algo. 6 defines the black-box *getAppFib2*( $L, g, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{L}$ ) that computes an  $L$ -bit approximation  $\tilde{g}$  of  $(g)_{\mathbf{b}}$  for any point of  $\Delta(\mathbf{B})$ . It uses the procedure *getAppFib*( $L, \mathbf{B}, g$ ) defined in Algo. 2 but never fails: when the latter call fails, the TAC  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is shrunk until *getAppFib*( $L, \mathbf{B}, g$ ) succeeds.

If  $i-1 = 1$ , the  $i-1$ -TAC  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is shrunk by calling *clusterPol*( $f_1, \mathbf{B}^{[1]}, 2^{-L'}$ ), that returns a list  $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}) \mid 1 \leq j \leq l\}$  of  $i-1$ -TACs such that the radius of  $\Delta(\mathbf{B}^j)$  is smaller than  $2^{-L'}$ .

If  $i-1 > 1$ , the  $i-1$ -TAC  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  is shrunk by calling *clusterPolInFiber* as in step 6 that returns a list  $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}) \mid 1 \leq j \leq l\}$  of  $i-1$ -TACs where the radius of  $\Delta(\mathbf{B}^j)$  is smaller than  $2^{-L'}$ .

Shrinking a TAC can obviously split it into several TACs. If so, one of the obtained TAC replaces  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  and the other ones are stored in the list  $\mathcal{L}$ .

### 3.4 Correctness and termination

We use an induction scheme to prove Prop. 9. Let  $n$  be an integer such that  $1 \leq n \leq d$ . We note  $P[9](n)$  the property: Prop. 9 is true for  $i = n$ .

When called with inputs  $\mathbf{f}^{[1]} = f_1, \mathbf{B}^{[1]} = (B)$  where  $B \subset \mathbb{C}$  and  $\epsilon$ , *clusterTriSys*( $\mathbf{f}^{[1]}, \mathbf{B}^{[1]}, \epsilon$ ) calls *clusterPol*( $f_1, B, \epsilon$ ) and returns its output. To prove  $P[9](1)$ , we introduce the following corollary that is a direct consequence of Lem. 10.

---

**Algorithm 6** *getAppFib2*( $L, g, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{L}$ )

---

**Input:** A precision  $L \in \mathbb{N}$  s.t.  $L > 1$ ,  $g \in \mathbb{C}[\mathbf{z}^{[i]}]$ , an  $i - 1$ -TAC  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  where  $\mathbf{f} = (f_1, \dots, f_{i-1})$  and  $\mathbf{B} = (B_1, \dots, B_{i-1})$ , a list  $\mathcal{L}$  of  $i - 1$ -TACs.

**Output:** Modifies in-place  $(\mathbf{B}, \mathbf{m}, \mathbf{f})$  in  $(\mathbf{B}', \mathbf{m}', \mathbf{f})$  and  $\mathcal{L}$  in  $\mathcal{L}'$  such that  $\mathcal{L} \subseteq \mathcal{L}'$  and if  $\mathbf{a} \in Z(\mathbf{B}, \mathbf{f})$ , then  $\mathbf{a}$  is either in  $Z(\mathbf{B}', \mathbf{f})$  or in a TAC in  $\mathcal{L}'$ .

Returns an  $L$ -bit approximation  $\tilde{g}$  of  $(g)_{\mathbf{b}}$  for any  $\mathbf{b} \in \Delta(\mathbf{B}')$ .

```

1:  $L' \leftarrow L, \mathcal{S} \leftarrow \emptyset$ 
2:  $(\tilde{g}, \text{success}) \leftarrow \text{getAppFib}(L, \mathbf{B}, g)$  //see Algo. 2
3: while  $\text{success} = 0$  do
4:    $L' \leftarrow 2L'$ 
5:   if  $i - 1 > 1$  then
6:      $\mathcal{S} \leftarrow \text{clusterPolInFiber}(f_{i-1}, (\mathbf{B}^{[i-2]}, \mathbf{m}^{[i-2]}, \mathbf{f}^{[i-2]}), B_{i-1}, 2^{-L'})$  //see Algo. 5
7:   else
8:      $\mathcal{S} \leftarrow \text{clusterPol}(f_1, \mathbf{B}^{[1]}, 2^{-L'})$ 
9:    $(\mathbf{B}, \mathbf{m}, \mathbf{f}) \leftarrow \text{pop}(\mathcal{S})$ 
10:   $\text{append}(\mathcal{L}, \mathcal{S})$  and  $\mathcal{S} \leftarrow \emptyset$ 
11:   $(\tilde{g}, \text{success}) \leftarrow \text{getAppFib}(L, \mathbf{B}, g)$  //see Algo. 2
12: return  $\tilde{g}$ 

```

---

**Corollary 11** (of Lem. 10). *For any univariate polynomial  $f_1 \in \mathbb{C}[z_1]$ , for any square complex box  $B \subset \mathbb{C}$  and any real number  $\epsilon$  s.t.  $0 < \epsilon < \frac{1}{2}$ ,  $\text{clusterPol}(f_1, B, \epsilon)$  terminates and returns a list  $\{(\mathbf{B}^j, \mathbf{m}^j, (f_1)) \mid 1 \leq j \leq l\}$  of 1-TACs, where  $\mathbf{m}^j = (m_1^j)$  for  $1 \leq j \leq l$ , satisfying:*

- (1.a)  $\Delta(\mathbf{B}^j)$  are pairwise disjoint polydiscs of radius  $\leq \epsilon$ ,
- (1.b)  $\prod_{i=1}^1 m_i^j = \#(\Delta(\mathbf{B}^j), (f_1)) = \#(3\Delta(\mathbf{B}^j), (f_1))$  for all  $j$ , and
- (1.c)  $Z(B, (f_1)) \subseteq \bigcup_{1 \leq i \leq l} Z(\Delta(\mathbf{B}^j), (f_1)) \subseteq Z(2B, (f_1))$ .

Let now  $n \geq 1$  and suppose that P[9]( $n$ ) holds. We show that P[9]( $n + 1$ ) holds. When  $d \geq n + 1 > 1$ ,  $\text{clusterTriSys}(\mathbf{f}^{[n+1]}, \mathbf{B}^{[n+1]}, \epsilon)$  calls  $\text{clusterTriSys}(\mathbf{f}^{[n]}, \mathbf{B}^{[n]}, \epsilon)$  that terminates and returns a list  $\mathcal{S}$  of  $n$ -TACs satisfying (a), (b) and (c) of Prop. 9 for  $i = n$ . Then it calls  $\text{clusterPolInFiber}(f_{n+1}, (\mathbf{B}^{\text{cur}}, \mathbf{m}^{\text{cur}}, \mathbf{f}^{[n]}), B_{n+1}, \epsilon)$  for each  $n$ -TAC  $(\mathbf{B}^{\text{cur}}, \mathbf{m}^{\text{cur}}, \mathbf{f}^{[n]})$  in  $\mathcal{S}$ , and appends the output in a list  $\mathcal{R}$ . We introduce the following proposition, of which we postpone the proof:

**Proposition 12.** *Let  $1 \leq i < d$ ,  $f_{i+1} \in \mathbb{C}[\mathbf{z}^{[i+1]}]$ ,  $(\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i]})$  be an  $i$ -TAC,  $B \subset \mathbb{C}$  be a square complex box and  $\epsilon \in \mathbb{R}$  s.t.  $0 < \epsilon < \frac{1}{2}$ .*

*$\text{clusterPolInFiber}(f_{i+1}, (\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i]}), B, \epsilon)$  terminates and returns a list  $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}^{[i+1]}) \mid 1 \leq j \leq l\}$  of  $i + 1$ -TACs, where  $\mathbf{m}^j = (m_1^j, \dots, m_{i+1}^j)$  for  $1 \leq j \leq l$ , satisfying:*

- (2.a)  $\Delta(\mathbf{B}^j)$  are pairwise disjoint polydiscs of radius  $\leq \epsilon$ ,
- (2.b)  $\prod_{k=1}^{i+1} m_k^j = \#(\Delta(\mathbf{B}^j), \mathbf{f}^{[i+1]}) = \#(3\Delta(\mathbf{B}^j), \mathbf{f}^{[i+1]})$  for all  $j$ , and
- (2.c)  $Z((\mathbf{B}, B), \mathbf{f}^{[i+1]}) \subseteq \bigcup_{1 \leq i \leq l} Z(\Delta(\mathbf{B}^j), \mathbf{f}^{[i+1]}) \subseteq Z(2(\mathbf{B}, B), \mathbf{f}^{[i+1]})$ .

From Prop. 12 with  $i = n$ ,  $clusterTriSys(\mathbf{f}^{[n+1]}, \mathbf{B}^{[n+1]}, \epsilon)$  terminates and returns a list  $\mathcal{R} = \{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}^{[n+1]}) | 1 \leq j \leq l\}$ . From Prop. 12 with  $i = n$ , any element of this list is an  $n + 1$ -TAC.

The  $\Delta(\mathbf{B}^j)$ 's have radius less than  $\epsilon$  and are pairwise disjoint because for  $k \neq l$ , if  $\Delta(\mathbf{B}^j)^{[n]} \neq \Delta(\mathbf{B}^k)^{[n]}$  then  $\Delta(\mathbf{B}^j)^{[n]}$  and  $\Delta(\mathbf{B}^k)^{[n]}$  are pairwise disjoint (property (a) of Prop. 9 for  $i = n$ ). Otherwise,  $(\Delta(\mathbf{B}^j)^{[n]} = \Delta(\mathbf{B}^k)^{[n]})$ , the  $\Delta(\mathbf{B}^j)^{[n]}$  and  $\Delta(\mathbf{B}^k)^{[n]}$  (property (2.a) of Prop. 12 for  $i = n$ ). Hence property (a) of Prop. 9 for  $i = n + 1$  holds.

Property (b) of Prop. 9 for  $i = n + 1$  is a direct consequence of property (2.b) of Prop. 12 for  $i = n$ .

To prove the first inclusion of property (c) of Prop. 9 for  $i = n + 1$ , we consider a point  $\mathbf{a} = (a_1, \dots, a_{n+1})$  of  $Z((\mathbf{B}^{[n]}, B_{n+1}), \mathbf{f}^{[n+1]})$ . From property (c) of Prop. 9 for  $i = n$ , it exists an  $n$ -TAC  $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f}^{[n]})$  returned by  $clusterTriSys(\mathbf{f}^{[n]}, \mathbf{B}^{[n]}, \epsilon)$  such that  $a^{[n]} \in \Delta(\mathbf{B}^{cur})$ . Since  $a_{n+1} \in B_{n+1}$  and from property (2.c) of Prop. 12 for  $i = n$ ,  $\mathbf{a}$  is in an  $n + 1$ -TAC returned by  $clusterPolInFiber(f_{n+1}, (\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f}^{[n]}), B_{n+1}, \epsilon)$ , hence it exists a  $j$  such that  $\mathbf{a} \in \Delta(\mathbf{B}^j)$ . To prove the second inclusion of property (c) of Prop. 9 for  $i = n + 1$ , we consider a  $j$  and a point  $\mathbf{a} = (a_1, \dots, a_{n+1})$  of  $Z((\mathbf{B}^j, B_{n+1}), \mathbf{f}^{[n+1]})$ . It exists an  $n$ -TAC  $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f}^{[n]})$  returned by  $clusterTriSys(\mathbf{f}^{[n]}, \mathbf{B}^{[n]}, \epsilon)$  such that  $a^{[n]} \in \Delta(\mathbf{B}^{cur})$  and from property (c) of Prop. 9 for  $i = n$ ,  $a^{[n]} \in Z(2\mathbf{B}^{[n]}, \mathbf{f}^{[n]})$ , and from property (2.c) of Prop. 12 for  $i = n$ ,  $a_{n+1}$  is in  $2B_{n+1}$ .

Then the  $n + 1$ -TACs returned by  $clusterTriSys(\mathbf{f}^{[n+1]}, \mathbf{B}^{[n+1]}, \epsilon)$  satisfy properties (a), (b) and (c) of Prop. 9 for  $i = n + 1$  and we have  $P[9](n) \Rightarrow P[9](n + 1)$ , what concludes the proof of Prop. 9.  $\square$

**Proof of Prop. 12:** We also provide an inductive proof for Prop. 12. Let  $n$  be an integer such that  $1 \leq n < d$ . We note  $P[12](n)$  the property: Prop. 12 is true for  $i = n$ .

Since the procedure  $clusterPolInFiber$  calls indirectly the procedure  $getAppFib2$ , we introduce the following proposition:

**Proposition 13.** *Let  $1 \leq i < d$ , let  $(\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i]})$  be an  $i$ -TAC and  $f_{i+1} \in \mathbb{C}[\mathbf{z}^{[i+1]}]$ . Then  $getAppFib2(L, f_{i+1}, (\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i]}), \mathcal{L})$  terminates. Let  $\tilde{f}_{i+1}$  be the output of this call and let  $(\mathbf{B}', \mathbf{m}', \mathbf{f}^{[i]})$  and  $\mathcal{L}'$  be the modified instances of  $(\mathbf{B}, \mathbf{m}, \mathbf{f}^{[i]})$  and  $\mathcal{L}$ .*

*Then  $\tilde{f}_{i+1}$  is an  $L$ -bit approximation of  $(f_{i+1})_{\mathbf{b}}$  for any  $\mathbf{b} \in \Delta(\mathbf{B}')$ . Furthermore,  $(\mathcal{L}' \setminus \mathcal{L}) \cup \{(\mathbf{B}', \mathbf{m}', \mathbf{f}^{[i]})\} = \{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}^{[i]}) | 1 \leq j \leq l\}$  is a set of  $i$ -TACs, where  $\mathbf{m}^j = (m_1^j, \dots, m_i^j)$  for  $1 \leq j \leq l$ , satisfying:*

- (3.a)  $\Delta(\mathbf{B}^j)$  are pairwise disjoint polydiscs of radius  $\leq \epsilon$ ,
- (3.b)  $\prod_{k=1}^i m_k^j = \#(\Delta(\mathbf{B}^j), \mathbf{f}^{[i]}) = \#(3\Delta(\mathbf{B}^j), \mathbf{f}^{[i]})$  for all  $j$ , and
- (3.c)  $Z(\mathbf{B}, \mathbf{f}^{[i]}) = \bigcup_{1 \leq i \leq l} Z(\Delta(\mathbf{B}^j), \mathbf{f}^{[i]})$ .

We note  $P[13](n)$  the property: Prop. 13 is true for  $i = n$ .

We first prove  $P[13](1)$ . First,  $getAppFib(L, \mathbf{B}, f_1)$  given in Algo. 2 terminates for any  $L$  and  $\mathbf{B}$  and if  $\mathbf{B}$  is sufficiently small, it returns a pair  $(1, \tilde{f}_1)$  where  $\tilde{f}_1$  is an  $L$ -bit approximation of  $(f_1)_{\mathbf{b}}$  for any  $\mathbf{b} \in \Delta(\mathbf{B})$ . In  $getAppFib2(L, f_2, (\mathbf{B}, \mathbf{m}, \mathbf{f}^{[1]}), \mathcal{L})$ ,  $\mathbf{B}$  is shrink with  $clusterPol(f_1, \mathbf{B}, 2^{-L'})$  with increasing  $L'$  until it is sufficiently small so that  $getAppFib(L, \mathbf{B}, f_1)$  succeeds. Since  $clusterPol$  terminates (Corollary. 11),  $getAppFib2(L, f_2, (\mathbf{B}, \mathbf{m}, \mathbf{f}^{[1]}), \mathcal{L})$  terminates, and  $\tilde{f}_2$  is  $L$ -bit approximation of  $(f_2)_{\mathbf{b}}$  for any  $\mathbf{b} \in \Delta(\mathbf{B}')$ .

The call  $clusterPol(f_1, \mathbf{B}, 2^{-L'})$  returns a list of 1-TAC verifying conditions (1.a), (1.b) and (1.c) of Corollary. 11. It is clear that they satisfy conditions (3.a), (3.b) of Prop. 13. The equality

in condition (3.c) holds because  $(\mathbf{B}, \mathbf{m}, \mathbf{f}^{[1]})$  is a TAC, and as a consequence  $Z(\mathbf{B}, \mathbf{f}^{[1]})$  is a natural cluster, then  $Z(\mathbf{B}, \mathbf{f}^{[1]}) = Z(2\mathbf{B}, \mathbf{f}^{[1]})$ .

We now prove P[12](1). Each  $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$  in the working list  $\mathcal{L}$  managed by  $clusterPolInFiber(f_2, (\mathbf{B}, \mathbf{m}, \mathbf{f}^{[1]}), B, \epsilon)$  is an 1-TAC and after the call to  $ccluster$  that terminates from Lem. 10, any solution that was in  $Z(\mathbf{B}^{cur}, f_1)$  before the call to  $ccluster$  is either in  $Z(\mathbf{B}^{cur}, f_1)$  or in an  $i$ -TAC in  $\mathcal{L}$ . Then  $ccluster$  finds all the clusters of roots of  $f_2$  in  $B$  in the fiber over  $\mathbf{B}^{cur}$ , and properties (2.a), (2.b) and (2.c) of Prop. 12 holds.

Let now  $n \geq 1$  and suppose that P[12]( $n$ ) holds. Then P[12]( $n + 1$ ) holds. One can easily transform the proof of P[13](1) into the proof of P[13]( $n + 1$ ), and the proof of P[12](1) into the proof of P[12]( $n + 1$ ). It concludes the proof of Prop. 12.  $\square$

## 4 Implementation and benchmarks

In this section we describe briefly a prototype implementation of our local solution clustering algorithm for triangular systems. Then we present comparative results with two homotopy solvers.

### 4.1 A prototype implementation

We described recently our implementation of the root clustering algorithm of [BSS<sup>+</sup>16]. The C implementation<sup>2</sup> comes with an interface for Julia<sup>3</sup> called `Ccluster.jl`<sup>4</sup>. The package provides the function `ccluster` that implements the eponymous procedure described in Subsec. 3.1. This function also accepts in input a black-box delivering  $L$ -bit approximation of the coefficients of a polynomial.

We used `Ccluster.jl` as the corner stone of our Julia implementation of the local solution clustering algorithm for triangular systems described in Sec. 3. This implementation provides solver called `tcluster` that takes as input a triangular system  $\mathbf{f}(\mathbf{z}) = 0$ , a square complex box  $\mathbf{B}$  and a size  $\epsilon$ .

We used `tcluster` to cluster the solutions of  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ , where  $\mathbf{g}$  is defined as in Eq. 1. For any  $\delta$  and any input size  $\epsilon$ , we obtain 4 clusters containing respectively the solutions  $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$  of  $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ , and the sum of multiplicities in each cluster is 1.

Let now  $\delta = 100$ , and consider the system  $\mathbf{h}(\mathbf{z}) = \mathbf{0}$ , where  $\mathbf{h}$  is defined as in Eq. 2. When clustering its solutions with `tcluster` and  $\epsilon = 2^{-10}$ , we obtain 2 clusters: one contains  $\mathbf{a}_1, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6$  and has a sum of multiplicities 9, the other ones contains  $\mathbf{a}_2, \mathbf{a}_3$ , and has a sum of multiplicities 3. When calling `tcluster` with  $\epsilon = 2^{-50}$ , we obtain 4 clusters: one contains  $\mathbf{a}_1, \mathbf{a}_6$  and the sum of multiplicities is 6, one contains  $\mathbf{a}_4, \mathbf{a}_5$  and the sum of multiplicities is 3, and the other ones contain respectively  $\mathbf{a}_2$  and  $\mathbf{a}_3$  with multiplicities 2 and 1. When calling `tcluster` with  $\epsilon = 2^{-100}$ , the 6 solutions are each in a cluster with the appropriated multiplicity.

### 4.2 Benchmarks

Here we present benchmarks of `tcluster` for randomly generated triangular systems of polynomial equations with or without multiple solutions. We compare our solver with two homotopy solvers.

<sup>2</sup> <https://github.com/rimbach/Ccluster>

<sup>3</sup> <https://julialang.org/>

<sup>4</sup> <https://github.com/rimbach/Ccluster.jl>



**The homotopy solvers** The homotopy-continuation approach to solve a system of  $d$  polynomial equations in  $d$  unknowns, called *target* system, proceeds in two steps. First, an upper bound  $D$  on the number of solutions (counted with multiplicities) of the target system is computed and an *initial* system having  $D$  complex solutions, which solutions are known, is generated. Then, an homotopy is defined that links by one-dimensional smooth manifolds, called homotopy paths, each solution of the initial system to a solution of the target system. Each complex solution of the target system is the end-point of an homotopy path. However, when the homotopy paths are tracked with numerical non-certified method, as it is the case for efficient existing implementations, some solutions can be missing in particular when the path tracker jumps from a path to another. Note also that when a solution of the target system has a multiplicity  $m > 1$ ,  $m$  homotopy paths have this solution as an endpoint. Deciding that to paths have the same end-point in a numerical framework requires some dedicated techniques.

HOM4PS-2.0 <sup>5</sup> and Bertini <sup>6</sup> ([BHSW]) are two homotopy solvers. HOM4PS-2.0 is known for being very fast but not robust at all (*i.e.* it does not find all the solutions). Bertini is in general slower than HOM4PS-2.0 but much more robust; it implements a path-tracker using Adaptive Multi-Precision (AMP) that is an option for Bertini . Below we will use this option for Bertini and call it Bertini AMP . Bertini AMP uses a test based on  $\alpha$ -theory to decide that two paths have the same end-point.

**The systems** We use here the approach proposed in [CGY09] to generate triangular systems of polynomial equations without multiple solutions or with multiple solutions. The *type* of a triangular system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  where  $\mathbf{f} = (f_1, \dots, f_d)$  is a list  $(n_1, \dots, n_d)$  where  $n_i = \deg_{z_i}(f_i)$ . Notice that the number of solutions counted with multiplicities of a system of *generic dense* polynomials of type  $(n_1, \dots, n_d)$  is  $d_1 \times \dots \times d_n$ .

A random dense polynomial  $f_i$  in  $\mathbb{C}[\mathbf{z}^{[i]}]$  of degree  $n_i$  in  $z_i$  with integer coefficients in  $[-2^{-9}, 2^9]$  is recursively generated as follows: if  $i = 1$ ,  $f_1$  is a dense polynomial in  $\mathbb{C}[z_1]$  of degree  $n_1$  which coefficients are random integers in the range  $[-2^{-9}, 2^9]$ . If  $i > 1$ ,  $f_i = \sum_{j=0}^{n_i} g_j z_i^j$  where  $g_j \in \mathbb{C}[\mathbf{z}^{[i-1]}]$  is a random dense polynomial of degree  $n_i - j$  in  $z_{i-1}$  with integer coefficients in  $[-2^{-9}, 2^9]$ . A random dense triangular system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  of type  $(n_1, \dots, n_d)$  is generated by generating successively random dense polynomials  $f_i$  of degrees  $n_i$  in  $z_i$  with integer coefficients in  $[-2^{-9}, 2^9]$ . We will assume that such systems have only solutions of multiplicity one.

A triangular system  $\mathbf{f}(\mathbf{z}) = \mathbf{0}$  of type  $(n_1, \dots, n_d)$  with multiple solutions is generated as follows (see [CGY09]):  $f_1$  is a random dense polynomial in  $\mathbb{C}[z_1]$  of degree  $n_1$  with integer coefficients in  $[-2^{-9}, 2^9]$ . For  $i = 2, \dots, n$ ,  $f_i = a_i^2(b_i z_i + c_i)^{\lfloor \frac{n_1+1}{2} \rfloor - \lfloor \frac{a_i}{2} \rfloor}$  where  $a_i$  is a random dense polynomial in  $\mathbf{z}^{[i]}$  with degree  $\lfloor \frac{n_i}{2} \rfloor$  in  $z_1$  with integer coefficients in  $[-2^{-9}, 2^9]$ , and  $b_i, c_i$  are random dense polynomials in  $\mathbf{z}^{[i-1]}$  of degrees  $n_i$  in  $z_{i-1}$  with integer coefficients in  $[-2^{-9}, 2^9]$ .

**The benchmarks** We propose two benchmarks: one concerning solving triangular systems with different types without multiple solutions (see Table. 1) and one concerning triangular systems with multiple solutions (see Table. 2).

In both cases, we generated 5 systems of each type (see columns #sys in Tables) and solved systems with the two homotopy solvers (columns HOM4PS-2.0 and Bertini AMP ) and two versions

<sup>5</sup> [http://www.math.nsysu.edu.tw/~leetsung/works/HOM4PS\\_soft.htm](http://www.math.nsysu.edu.tw/~leetsung/works/HOM4PS_soft.htm)

<sup>6</sup> <https://bertini.nd.edu/>



type	#sys	tcluster local		tcluster global		HOM4PS-2.0		Bertini AMP	
		(#Sols:#Clus)	$\tau_\ell$ (s)	(#Sols:#Clus)	$\tau_g$ (s)	#Sols	$\tau_h$ (s)	#Sols	$\tau_b$ (s)
(3,3)	5	(2.00,2.00)	0.01	(9,9)	0.02	9	0.00	9	0.0
(6,6)	5	(11.4,11.4)	0.02	(36,36)	0.08	36	0.00	36	0.06
(9,9)	5	(31.2,31.2)	0.08	(81,81)	0.28	81	0.00	81	0.37
(3,3,3)	5	(4.00,4.00)	0.02	(27,27)	0.10	27	0.00	27	0.02
(6,6,6)	5	(34.2,34.2)	0.11	(216,216)	0.68	216	0.06	216	1.17
(9,9,9)	5	(149,149)	0.55	(729,729)	2.40	713	0.47	729	29.3
(3,3,3,3)	5	(3.00,3.00)	0.02	(81,81)	0.31	81	0.01	81	0.16
(6,6,6,6)	5	(63.4,63.4)	0.38	(1296,1296)	4.92	1274	1.37	1296	24.2
(9,9,9,9)	5	(559,559)	4.08	(6561,6561)	33.2	6036	111	6560	1605
(3,3,3,3,3)	5	(5.00,5.00)	0.08	(243,243)	1.47	243	0.11	243	0.64
(6,6,6,6,6)	5	(155,155)	2.42	(7776,7776)	49.6	7730	28.6	7776	318
(9,9,9,9,9)	5	(1739,1739)	36.5	(59049,59049)	523	-	-	?	>3600

Table 1: Solving triangular systems of randomly generated polynomials with `tcluster`, `HOM4PS-2.0` and `Bertini AMP`. For `tcluster` local (resp. global) we used  $\epsilon = 53$  and the initial domain is  $([-1, 1] + \sqrt{-1}[-1, 1])^d$  (resp.  $([-5, 5] \times 10^5 + \sqrt{-1}[-5, 5] \times 10^5)^d$ ).

of `tcluster`: “`tcluster` local” searches the clusters in the initial domain  $([-1, 1] + \sqrt{-1}[-1, 1])^d$  (where  $d$  is the dimension of the system) and “`tcluster` global” searches the clusters in the initial domain  $([-5, 5] \times 10^5 + \sqrt{-1}[-5, 5] \times 10^5)^d$ .

The columns `#Sols` give the average number of solutions found by each solver for each type, and  $\tau_\ell$ ,  $\tau_g$ ,  $\tau_h$  and  $\tau_b$  the average sequential times in seconds required by each solver, on a Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz machine with linux. For `tcluster` local and `tcluster` global, the columns `#Clus` give the average number of clusters found.

For the system we tested, `tcluster` global found all the solutions: it can be verified by comparing for each type  $(n_1, \dots, n_d)$  `#Sols` to  $\prod_{i=1}^d n_i$ .

**Table 1** reports benchmarks for triangular systems whose equations are random dense polynomials with integers coefficients within  $[-2^{-9}, 2^9]$ . For type (9,9,9,9,9), `Bertini AMP` has been stop after 1 hour. For type (9,9,9,9,9) `HOM4PS-2.0` terminates with a segmentation fault.

Homotopy solvers should find all the solutions of a system. `Bertini AMP` failed in this task for one system of type (9,9,9,9) but acknowledged that two paths have crossed. `HOM4PS-2.0` returns incorrect results but do not warn about this. In contrast, `tcluster` global always finds the appropriated number of solutions.

Regardless of the correction of the results, `tcluster` global is faster than `HOM4PS-2.0` for systems of types (9,9,9,9), and faster than `Bertini AMP` for systems which components have a degree greater than 3. The local version of `tcluster` is in general faster than the other solvers.

**Table 2** reports benchmarks for triangular systems having multiple solutions. For systems of each type, `tcluster` finds clusters containing one solution and for each cluster, the sum of multiplicities of the solutions in the cluster is the multiplicity of the unique solution it contains. For each system, the number of clusters found by `tcluster` global is then the number of distinct complex solutions the system has. `HOM4PS-2.0` fails in finding all the solutions. `Bertini AMP` also computes the multiplicity of solutions, and its output is correct except for one system of type (6,6) where it fails to decide equality of solutions. For type (9,9,9), `Bertini AMP` has been stop after 1 hour.

type	#sys	tcluster local		tcluster global		HOM4PS-2.0		Bertini AMP	
		(#Sols:#Clus)	$\tau_\ell$ (s)	(#Sols:#Clus)	$\tau_g$ (s)	#Sols	$\tau_h$ (s)	#Sols	$\tau_b$ (s)
(3,3)	5	(3.60,2.40)	0.03	(9,6)	0.02	8.60	0.00	6	1.59
(6,6)	5	(10.8,5.40)	0.02	(36,18)	0.09	36	0.00	21.8	3.63
(9,9)	5	(23.8,13.6)	0.08	(81,45)	0.29	67.4	0.06	45	218
(3,3,3)	5	(7.00,3.60)	0.04	(27,12)	0.07	25	0.01	12	25.1
(6,6,6)	5	(35.2,8.80)	0.11	(216,54)	0.49	210	0.16	54	47.9
(9,9,9)	5	(113,37.6)	0.56	(729,225)	2.46	357	18.9	?	>3600

Table 2: Solving triangular systems of randomly generated polynomials with multiple roots with `tcluster`, `HOM4PS-2.0` and `Bertini AMP`. For `tcluster` local (resp. global) we used  $\epsilon = 53$  and the initial domain is  $([-1, 1] + \sqrt{-1}[-1, 1])^d$  (resp.  $([-5, 5] \times 10^5 + \sqrt{-1}[-5, 5] \times 10^5)^d$ ).

`tcluster` global is always faster than `Bertini AMP`, and is faster than `HOM4PS-2.0` for systems of type (9, 9, 9).

## References

- [ALM99] Philippe Aubry, Daniel Lazard, and Marc Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1):105 – 124, 1999.
- [BHSW] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. Available at [bertini.nd.edu](http://bertini.nd.edu) with permanent doi: [dx.doi.org/10.7274/R0H41PB5](https://doi.org/10.7274/R0H41PB5).
- [BL12] Carlos Beltrn and Anton Leykin. Certified numerical homotopy tracking. *Experimental Mathematics*, 21(1):69–83, 2012.
- [BSS<sup>+</sup>16] Ruben Becker, Michael Sagraloff, Vikram Sharma, Juan Xu, and Chee Yap. Complexity analysis of root clustering for a complex polynomial. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '16*, pages 71–78, New York, NY, USA, 2016. ACM.
- [BSSY17] Ruben Becker, Michael Sagraloff, Vikram Sharma, and Chee Yap. A near-optimal subdivision algorithm for complex root isolation based on the pellet test and newton iteration. *Journal of Symbolic Computation*, 2017.
- [CGY09] Jin-San Cheng, Xiao-Shan Gao, and Chee-Keng Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768–785, 2009.
- [DZ05] Barry H Dayton and Zhonggang Zeng. Computing the multiplicity structure in solving polynomial systems. In *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 116–123. ACM, 2005.
- [KRS16] Alexander Kobel, Fabrice Rouillier, and Michael Sagraloff. Computing real roots of real polynomials ... and now for real! In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '16*, pages 303–310, New York, NY, USA, 2016. ACM.

- [Lai17] Pierre Lairez. A deterministic algorithm to compute approximate roots of polynomial systems in polynomial average time. *Foundations of Computational Mathematics*, 17(5):1265–1292, Oct 2017.
- [Rou99] Fabrice Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.
- [W<sup>+</sup>05] Ii Charles W Wampler et al. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.
- [ZFX11] Zhihai Zhang, Tian Fang, and Bican Xia. Real solution isolation with multiplicity of zero-dimensional triangular systems. *Science China Information Sciences*, 54(1):60–69, 2011.