Clustering Complex Zeros of Triangular System of Polynomials

Rémi Imbach, Marc Pouget, Chee Yap

Clustering Complex Zeros of Triangular System of Polynomials*

Rémi Imbach[1], Marc Pouget[2], and Chee Yap[1]

[1] Courant Institute of Mathematical Sciences, New York University, USA
remi.imbach@nyu.edu, yap@cs.nyu.edu
[2] Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
marc.pouget@inria.fr

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 cluster. 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 and regular chains for state of the art 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 fundamental problem of finding the complex solutions of a system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ of n polynomial equations in n complex variables $\mathbf{z} = (z_1, \ldots, z_n)$. The system $\mathbf{f} = (f_1, \ldots, f_n) : \mathbb{C}^n \to \mathbb{C}^n$ is **triangular** in the sense that $f_i \in \mathbb{C}[z_1, \ldots, i]$ for $1 \leq i \leq n$. Throughout this paper, we use boldface symbols to denote vectors and tuples; for instance $\mathbf{0}$ stands for $(0, \ldots, 0)$.

We are interested in finding clusters of solutions of triangular systems and in counting the total multiplicity of solutions clusters. Solving triangular systems is a fundamental task in polynomial equations solving, since many algebraic approaches (Gröbner basis, CAD, resultants,...) generally reduce the original system to triangular systems.

* Rémi's work is supported by the European Union's Horizon 2020 research and innovation programme No. 676541, NSF Grants # CCF-1563942, # CCF-1564132 and # CCF-1708884. Chee's work is supported by NSF Grants # CCF-1423228 and # CCF-1564132.

The problem of isolating the complex solutions of a polynomial system in an initial region-of-interest (ROI) is defined as follows: let $Z(\mathbf{B}, \mathbf{f})$ denote the set of solutions of $\mathbf{f}$ in $\mathbf{B}$, regarded[3] as a multiset.

Local solution isolation problem:
Given: a polynomial map $\mathbf{f} : \mathbb{C}^n \to \mathbb{C}^n$, a polybox $\mathbf{B} \subset \mathbb{C}^n$, $\epsilon > 0$
Output: a set $\{\mathbf{\Delta}^1, \ldots, \mathbf{\Delta}^l\}$ of pairwise disjoint polydiscs of radius $\leq \epsilon$ where
- $Z(\mathbf{B}, \mathbf{f}) = \bigcup_{j=1}^{l} Z(\mathbf{\Delta}^j, \mathbf{f})$.
- each $Z(\mathbf{\Delta}^j, \mathbf{f})$ is a singleton.

There are two issues with the above formulation: deciding if $Z(\mathbf{\Delta}^j, \mathbf{f})$ is a singleton, and deciding if such a singleton lies in $\mathbf{B}$, are two "zero problems" that requires exact computation. Generally, this can only be decided if $\mathbf{f}$ is algebraic. Even in the algebraic case, this may be very expensive. In [3] these two issues are side-stepped by defining the clustering problem.

Before proceeding, we fix some general notations for this paper. A *polydisc* $\mathbf{\Delta}$ is a vector $(\Delta_1, \ldots, \Delta_n)$ of complex discs. The *center* of $\mathbf{\Delta}$ is the vector of the centers of its components and the *radius* $r(\mathbf{\Delta})$ of $\mathbf{\Delta}$ is the max of the radii of its components. If δ is any positive real number, we note $\delta\mathbf{\Delta}$ the polydisc $(\delta\Delta_1, \ldots, \delta\Delta_n)$ that has the same center than $\mathbf{\Delta}$ and radius $\delta r(\mathbf{\Delta})$. A *square complex box* B is a complex interval $[l_1, u_1]+\sqrt{-1}([l_2, u_2])$ where $u_2-l_2 = u_1-l_1$; the *width* $w(B)$ of B is u_1-l_1 and the *center* of B is $u_1+\frac{w(B)}{2}+\sqrt{-1}(u_2+\frac{w(B)}{2})$. A *polybox* $\mathbf{B} \in \mathbb{C}^n$ is a vector $(B_1, \ldots, B_n)$ of square complex boxes. The *center* of $\mathbf{B}$ is the vector of the centers of its components; the *width* $w(\mathbf{B})$ of $\mathbf{B}$ is the max of the widths of its components. If δ is any positive real number, we note $\delta\mathbf{B}$ the polybox $(\delta B_1, \ldots, \delta B_n)$ 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 $\mathbf{a} \in \mathbb{C}^n$ 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 [21, Def. 1, p. 61], we note it $\#(\mathbf{a}, \mathbf{f})$. An equivalent definition uses dual spaces, see [9, Def. 1, p. 117]. For any set $S \subseteq \mathbb{C}^n$, we note $Z(S, \mathbf{f})$ the set of distinct solutions of $\mathbf{f}$ in S, and $\#(S, \mathbf{f})$ the sum of multiplicities of solutions of $\mathbf{f}$ in S. In the context of numerical algorithm, the notion of cluster of solutions is more meaningful than that of solution with multiplicity since the perturbation of a multiple solution generates a cluster. We thus "soften" the problem of isolating the solutions of a triangular system of polynomial equations while counting their multiplicities by translating it into the local solution clustering problem defined as follows:

[3] A **multiset** S is a pair $(\underline{S}, \mu)$ where $\underline{S}$ is an ordinary set called the **underlying set** and $\mu : \underline{S} \to \mathbb{N}$ assigns a positive integer $\mu(x)$ to each $x \in \underline{S}$. Call $\mu(x)$ the **multiplicity** of x in S, and $\mu(S) := \sum_{x \in \underline{S}} \mu(x)$ the **multiplicity** of S. Also, let $|S|$ denote the cardinality of $\underline{S}$. If $|S| = 1$, then S is called a **singleton**. We can form the union of two multisets, $S \cup S'$ whose underlying set is $\underline{S} \cup \underline{S'}$ and the multiplicities add up as expected.

Clustering problem:
Given: a polynomial map $\mathbf{f} : \mathbb{C}^n \to \mathbb{C}^n$, a polybox $\mathbf{B} \subset \mathbb{C}^n$, $\epsilon > 0$
Output: a set of pairs $\{(\mathbf{\Delta}^1, m^1), \ldots, (\mathbf{\Delta}^l, m^l)\}$ where:
- the $\mathbf{\Delta}^j$s are pairwise disjoint polydiscs of radius $\leq \epsilon$,
- $m^j = \#(\mathbf{\Delta}^j, \mathbf{f}) = \#(3\mathbf{\Delta}^j, \mathbf{f})$ for all $1 \leq j \leq l$, and
- $Z(\mathbf{B}, \mathbf{f}) \subseteq \bigcup_{j=1}^{l} Z(\mathbf{\Delta}^j, \mathbf{f}) \subseteq Z(2\mathbf{B}, \mathbf{f})$.

In this reformulation of the root isolation problem, we have remove the two "zero problems" noted above: we output clusters to avoid the first problem, and we allow the output to contain zeroes outside the ROI $\mathbf{B}$ to avoid the second one. We choose $2\mathbf{B}$ for simplicity; it is easy to replace the factor of 2 by $1 + \delta$ for any desired $\delta > 0$.

Outline. In the remaining of this section we explain our contribution, summarize previous work and the local univariate solution clustering from [3]. In Sec. 2, we define the notion of *tower of algebraic clusters* together with a recursive method to compute the sum of multiplicities of the solutions it contains. Our algorithm for solving the local solution clustering problem for triangular systems is introduced in Sec. 3. The implementation and experimental results are presented in Sec. 4.

1.1 Our contributions

We propose an algorithm for solving the local solution clustering problem for triangular systems with zero-dimensional solution set. To this end, we propose a formula to count the sum of multiplicities in a cluster. Our formula is derived from a result of [21] that links the intersection multiplicity of a solution of a triangular systems to multiplicities in fibers. We define *towers of algebraic clusters* to 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 [4] 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 polynomials [3], an implementation of which is detailed in [13]. This univariate clustering algorithm is based on subdivision of the initial complex square box, and uses Pellet's test 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, instead it uses a black-box that is able to give approximations of them to any desired precision. This model is thus well adapted to solve in the fibers of a triangular system.

We implemented and experimented our algorithm; we show 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.

1.2 Related work

There is a vast literature on solving polynomial systems and we can only refer to book surveys and references therein, see for instance [10,19]. On the algebraic side, symbolic tools like Goebner basis, resultant, rational univariate parameterization, triangularization, enable to reduce the problem to the univariate case. These methods are global: they do not take advantage of solving in a predefined small domain. Being symbolic, these methods handle all input, in particular with solutions with multiplicities, and are certified but at the price of a high complexity that limits their use in practice.

On the numerical side, one can find subdivision and homotopy methods. The main advantage of subdivision methods is their locality: the practical complexity depends on the size of the solving domain and the number of solutions in this domain. Their main drawback is that they are only practical for low dimensional systems. On the other hand, homotopy methods are efficient for high dimensional systems, they are not local but solutions are computed independently from one another. Numerical methods only work for restricted classes of systems and the certification of the output remains a challenge. Multiprecision arithmetic, interval analysis, deflation and α-theory are now classical tools to address this certification issue [18,11,5,20].

In the univariate case, practical certified algorithms are now available for real and complex solving that match the best known complexity bounds together with efficient implementations [15,13].

Only a few work address the specific problem of solving triangular polynomial systems. The solving can then be performed coordinate by coordinate by specialisation and univariate solving in fibers. For real solving, see [6] for the regular case and [7,21] for systems with solutions with multiplicities.

1.3 Definitions and notation

For any n-dimensional vector $\mathbf{v} = (v_1, \ldots, v_n)$ with $n > 1$, we note $\dot{\mathbf{v}}$ its last component (*i.e.* v_n) and $\overline{\mathbf{v}}$ for the $(n-1)$-dimensional vector $(v_1, \ldots, v_{n-1})$ of its other components. By abuse of notation, we sometimes write $\mathbf{v} = (\overline{\mathbf{v}}, \dot{\mathbf{v}})$. The modulus of a complex number z is noted $|z|$ and the norm of a vector is $||v|| = \max_{i=1\ldots n}|v_i|$.

If $\mathbf{b}$ is any point in $\mathbb{C}^{n-1}$, $\mathbf{z} = (z_1, \ldots, z_n)$ and $g \in \mathbb{C}[\mathbf{z}]$, we note $(g)_{\mathbf{b}}$ the univariate polynomial $g(\mathbf{b}, \dot{\mathbf{z}})$ in $\mathbb{C}[\dot{\mathbf{z}}]$ obtained by specializing g at $\mathbf{b}$. If $\mathbf{a} \in \mathbb{C}^n$ is a solution of $\mathbf{f}$, we call multiplicity of $\dot{\mathbf{a}}$ in $\dot{\mathbf{f}}$ in the fiber $\overline{\mathbf{a}}$ the multiplicity of the root $\dot{\mathbf{a}}$ of the polynomial $(\dot{\mathbf{f}})_{\overline{\mathbf{a}}}$. We note it $\#(\dot{\mathbf{a}}, (\dot{\mathbf{f}})_{\overline{\mathbf{a}}})$.

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 $\frac{3}{4}w$, notice that $\Delta(B)$ contains B. If $\mathbf{B} = (B_1, \ldots, B_n) \subset \mathbb{C}^n$ is a polybox, we note $\Delta(\mathbf{B})$ the polydisc $(\Delta(B_1), \ldots, \Delta(B_n))$, notice that $\Delta(\mathbf{B})$ contains $\mathbf{B}$.

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

Algorithm 1 T_*^L test: $T_*^L(\Delta, L, \widetilde{f})$

Input: A complex disc Δ, an integer $L > 1$, a polynomial $\widetilde{f}$ of degree d.
Output: An integer $k \in \{-2, -1, 0, \ldots, d\}$. If $k \geq 0$ and $\widetilde{f}$ is an L-bit approximation of $f \in \mathbb{C}[z_1]$, then $\#(\Delta, f) = k$.

Algorithm 2 $clusterPol(f, B, \epsilon)$

Input: A polynomial $f \in \mathbb{C}[z_1]$, an initial box B, a real number $\epsilon \in \mathbb{R}$ s.t. $0 < \epsilon$.
Output: A list $\{(B^j, m^j) | 1 \leq j \leq l\}$ so that $\{(\Delta(B^j), m^j) | 1 \leq j \leq l\}$ is a solution of the clustering problem for $n = 1$.

We call L-bit approximation of $a \in \mathbb{C}$ a dyadic complex number $\tilde{a}$ that satisfies $|a - \tilde{a}| \leq 2^{-L}$. We call L-bit approximation of $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$ a vector $\tilde{\mathbf{a}} \in \mathbb{C}^n$ such that $\tilde{a_i}$ is an L-bit approximation of a_i for $1 \leq i \leq n$, in other words $||\tilde{\mathbf{a}} - \mathbf{a}|| \leq 2^{-L}$. 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.

1.4 Pellet's theorem based clustering for univariate polynomials

We recall in Algo 1 the specifications of the T_*^L test introduced in [4]. It combines Pellet's theorem and Graeffe iterations to determine the number of solutions counted with multiplicities of a polynomial $f \in \mathbb{C}[z]$ in a complex disc Δ, using an L-bit approximation $\widetilde{f}$ of f. It returns -2 if L is to small to decide, -1 if f has roots close to the boundary of Δ and an integer $k \geq 0$ only if f has k roots counted with multiplicities in Δ.

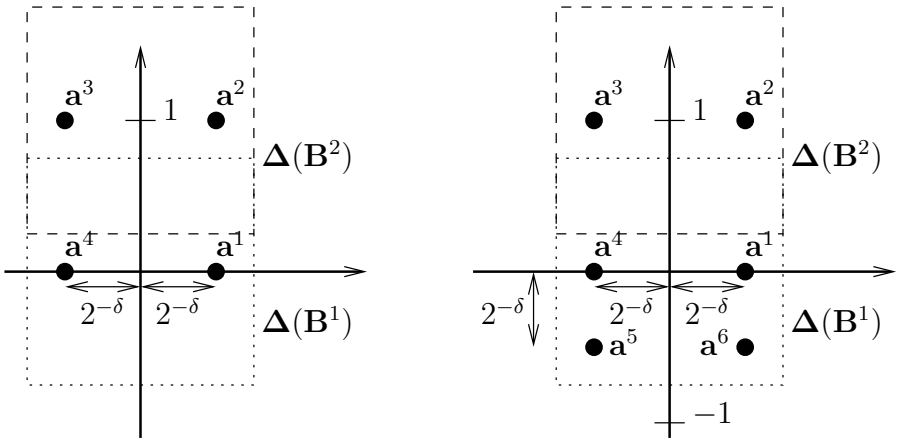
The T_*^L test can be embedded in the so-called T_* test that takes in input Δ and a black-box producing L-bit approximations of f. The T_* test applies the T_*^L test for Δ and increases L whenever the latter test returns -2.

In [3], the T_* test is combined with subdivision of the initial complex square box and Newton iterations to obtain the local clustering procedure $clusterPol$, specified in Algo. 2, that solves the clustering problem for $n = 1$.

The input polynomial f in Algo. 2 is given as an approximation procedure that for a given strictly positive integer L returns an L-bit approximation of f. An implementation of the procedure $clusterPol$ is described in [13].

2 Sum of multiplicities in clusters of solutions

We extend in Sec. 2.2 a result of [21] to an inductive formula giving the sum of multiplicities of solutions of a triangular system in a cluster. In Sec. 2.3, we introduce the Towers of Algebraic Clusters (TAC) that are special instances of clusters in which our formula can be applied. In Sec. 2.4, we show how to use the T_*^L test to prove that a cluster is a TAC. We first define in Sec. 2.1 two illustrative examples.



$\#(\mathbf{a}^3, \mathbf{g}) = 1 \times 1$	$\#(\mathbf{a}^2, \mathbf{g}) = 1 \times 1$	$\#(\mathbf{a}^3, \mathbf{h}) = 1 \times 1$	$\#(\mathbf{a}^2, \mathbf{h}) = 2 \times 1$
$\#(\mathbf{a}^4, \mathbf{g}) = 1 \times 1$	$\#(\mathbf{a}^1, \mathbf{g}) = 1 \times 1$	$\#(\mathbf{a}^4, \mathbf{h}) = 1 \times 1$	$\#(\mathbf{a}^1, \mathbf{h}) = 2 \times 1$
$\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{g}) = 2 \times 1$		$\#(\mathbf{a}^5, \mathbf{h}) = 1 \times 2$	$\#(\mathbf{a}^6, \mathbf{h}) = 2 \times 2$
$\#(\mathbf{\Delta}(\mathbf{B}^2), \mathbf{g}) = 2 \times 1$		$\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{h}) = 3 \times 3$	
		$\#(\mathbf{\Delta}(\mathbf{B}^2), \mathbf{h}) = 3 \times 1$	

Fig. 1. In the left (resp. right), the solutions of $\mathbf{g}(\mathbf{z}) = 0$ (resp. $\mathbf{h}(\mathbf{z}) = 0$) defined in Eq. 1 (resp. 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 $2 * 2^{-\delta}$. The boxes in dashed lines are the real parts of $\mathbf{\Delta}(\mathbf{B}^1)$ and $\mathbf{\Delta}(\mathbf{B}^2)$. In the frame, the multiplicities of solutions of each system computed with the formula of [21].

2.1 Two examples

Let $\delta > 0$ be an integer. We define the triangular systems $\mathbf{g}(\mathbf{z}) = (g_1(z_1), g_2(z_1, z_2) = \mathbf{0}$ and $\mathbf{h}(\mathbf{z}) = (h_1(z_1), h_2(z_1, z_2)) = \mathbf{0}$ as follows:

$$(\mathbf{g}(\mathbf{z}) = \mathbf{0}) : \begin{cases} (z_1 - 2^{-\delta})(z_1 + 2^{-\delta}) = 0 \\ (z_2 - 2^{2\delta} z_1^2) z_2 = 0 \end{cases} \tag{1}$$

$$(\mathbf{h}(\mathbf{z}) = \mathbf{0}) : \begin{cases} (z_1 - 2^{-\delta})^2 (z_1 + 2^{-\delta}) = 0 \\ (z_2 + 2^{\delta} z_1^2)^2 (z_2 - 1) z_2 = 0 \end{cases} \tag{2}$$

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

2.2 Sum of multiplicities in a cluster

We recall the result of [21] for counting multiplicities of solutions of triangular systems knowing multiplicities in fibers, and rephrase it in an inductive setting.

Theorem 1 ([21]). *Let* $\mathbf{a} \in \mathbb{C}^n$ *be a solution of the triangular system* $\mathbf{f}(\mathbf{z}) = 0$. *The multiplicity of* $\mathbf{a}$ *in* $\mathbf{f}$ *is*

$$\#(\mathbf{a}, \mathbf{f}) = \#(\dot{\mathbf{a}}, (\dot{\mathbf{f}})_{\overline{\mathbf{a}}}) \times \#(\overline{\mathbf{a}}, \overline{\mathbf{f}}).$$

We extend Thm. 1 to a formula giving the sum of multiplicities of the solutions of $\mathbf{f}(\mathbf{z}) = 0$ in a cluster $Z(\mathbf{\Delta}, \mathbf{f})$.

Theorem 2. *Let* $Z(\mathbf{\Delta}, \mathbf{f})$ *be a cluster of solutions of the triangular system* $\mathbf{f}(\mathbf{z}) = 0$. *If there is an integer* $m \geq 1$ *so that for any solution* $\mathbf{a} \in Z(\mathbf{\Delta}, \mathbf{f})$, *one has* $m = \#(\dot{\mathbf{\Delta}}, (\dot{\mathbf{f}})_{\overline{\mathbf{a}}})$, *then*

$$\#(\mathbf{\Delta}, \mathbf{f}) = m \times \#(\overline{\mathbf{\Delta}}, \overline{\mathbf{f}}).$$

We apply Thm. 1 to compute the multiplicities of solutions of $\mathbf{g}(\mathbf{z}) = 0$ and $\mathbf{h}(\mathbf{z}) = 0$ (see Eq. 1 and Eq. 2). $\mathbf{a}^1$ has multiplicity 1 in $\mathbf{g}$: $\#(\mathbf{a}^1, \mathbf{g}) = \#(a_1^1, g_1) \times \#(a_2^1, (g_2)_{a_1^1}) = 1 \times 1$. $\mathbf{a}^1$ has multiplicity 2 in $\mathbf{h}$: $\#(\mathbf{a}^1, \mathbf{h}) = \#(a_1^1, h_1) \times \#(a_2^1, (h_2)_{a_1^1}) = 2 \times 1$. The multiplicities of other solutions are given in fig. 1.

Let $\mathbf{B}^1 = (B_1^1, B_2^1)$ be the polybox centered in $(0, 0)$ having width $2 \times 2^{-\delta}$. $Z(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{g}) = \{\mathbf{a}^1, \mathbf{a}^4\}$ and $\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{g}) = 2$. Since $\#(\Delta(B_2^1), (\dot{\mathbf{g}})_{\overline{\mathbf{a}^1}}) = \#(\Delta(B_2^1), (\dot{\mathbf{g}})_{\overline{\mathbf{a}^4}}) = 1$, one can apply Thm. 2 and obtain $\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{g}) = 2 \times 1$.

$Z(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{h}) = \{\mathbf{a}^1, \mathbf{a}^4, \mathbf{a}^5, \mathbf{a}^6\}$ and $\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{h}) = 9$. Since $\#(\Delta(B_2^1), (\dot{\mathbf{g}})_{\overline{\mathbf{a}^1}}) = \#(\Delta(B_2^1), (\dot{\mathbf{g}})_{\overline{\mathbf{a}^4}}) = 3$, one can apply Thm. 2 and obtain $\#(\mathbf{\Delta}(\mathbf{B}^1), \mathbf{h}) = 3 \times 3$.

Let $\mathbf{B}^2 = (B_1^2, B_2^2)$ be polybox centered in $(0, 1)$ having width $2 \times 2^{-\delta}$. One can apply Thm. 2 and $\#(\mathbf{\Delta}(\mathbf{B}^2), \mathbf{g}) = 2 \times 1$, and $\#(\mathbf{\Delta}(\mathbf{B}^2), \mathbf{h}) = 3 \times 1$. The real parts of $\mathbf{\Delta}(B^1)$ and $\mathbf{\Delta}(B^2)$ are depicted in Fig. 1.

Proof (of Thm. 2.). Remark that $Z(\mathbf{\Delta}, \mathbf{f}) = \{\mathbf{a} \in \mathbf{\Delta} | \mathbf{f}(\mathbf{a}) = \mathbf{0}\}$ can be defined in an inductive way as $Z(\mathbf{\Delta}, \mathbf{f}) = \{(\mathbf{b}, c) \in \mathbf{\Delta} | \mathbf{b} \in Z(\overline{\mathbf{\Delta}}, \overline{\mathbf{f}})$ and $c \in Z(\dot{\mathbf{\Delta}}, (\dot{\mathbf{f}})_{\mathbf{b}})\}$. We use Thm. 1 to write $\#(\mathbf{\Delta}, \mathbf{f})$ as

$$\sum_{(\mathbf{b},c) \in Z(\mathbf{\Delta}, \mathbf{f})} \#(\mathbf{b}, \overline{\mathbf{f}}) \times \#(c, (\dot{\mathbf{f}})_{\mathbf{b}}) = \sum_{\mathbf{b} \in Z(\overline{\mathbf{\Delta}}, \overline{\mathbf{f}})} \left(\#(\mathbf{b}, \overline{\mathbf{f}}) \times \sum_{c \in Z(\dot{\mathbf{\Delta}}, (\dot{\mathbf{f}})_{\mathbf{b}})} \#(c, (\dot{\mathbf{f}})_{\mathbf{b}}) \right)$$

and by definition of m,

$$\#(\mathbf{\Delta}, \mathbf{f}) = \sum_{\mathbf{b} \in Z(\overline{\mathbf{\Delta}}, \overline{\mathbf{f}})} \#(\mathbf{b}, \overline{\mathbf{f}}) \times m = m \times \#(\overline{\mathbf{\Delta}}, \overline{\mathbf{f}})$$

□

2.3 Towers of algebraic clusters

A TAC is a special instance of a polydisc containing a cluster of solutions. A TAC satisfies the hypothesis of Thm. 2, then one can compute the sum of multiplicities of the solutions in it (see Thm. 3).

Definition 1 (Algebraic clusters). *We call algebraic cluster a triple* (B, m, g) *where* $B \subset \mathbb{C}$ *is a square complex box,* m *is an integer* ≥ 1 *and* g *is a univariate polynomial, such that* $\#(\Delta(B), g) = \#(3\Delta(B), g) = m$.

Let g_1 and h_1 be defined as in Eqs. 1 and 2, and B_1^1 be as defined in 2.2. $(B_1^1, 2, g_1)$ and $(B_1^1, 3, h_1)$ are both algebraic clusters.

Definition 2 (Towers). *We call tower a triple* $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ *where* $\mathbf{B} = (B_1, \ldots, B_n)$ *is a polybox,* $\mathbf{m} = (m_1, \ldots, m_n)$ *is a vector of integers* ≥ 1 *and* $\mathbf{f} = (f_1, \ldots, f_n)$ *is a triangular system.*

We call n *the height of* $(\mathbf{B}, \mathbf{m}, \mathbf{f})$.

Definition 3 (Towers of Algebraic Clusters, TAC). *Let* $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ *be a tower of height* n. $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ *is a tower of algebraic clusters or TAC, if either* $n = 1$ *and* (B_1, m_1, f_1) *is an algebraic cluster, or* $n > 1$ *and*

(*i*) $(\overline{\mathbf{B}}, \overline{\mathbf{m}}, \overline{\mathbf{f}})$ *is a TAC and*
(*ii*) $\forall \mathbf{b} \in \Delta(\mathbf{B}), (\dot{\mathbf{B}}, \dot{\mathbf{m}}, (\dot{\mathbf{f}})_{\overline{\mathbf{b}}})$ *is an algebraic cluster.*

Note that if (B, m, g) is an algebraic cluster, $Z(\Delta(B), g)$ is a natural cluster. One can easily see with an inductive reasonning that if $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ is a TAC, $\#(\Delta(\mathbf{B}), \mathbf{f}) = \#(3\Delta(\mathbf{B}), \mathbf{f})$ and $\Delta(\mathbf{B})$ is a natural cluster of solutions of $\mathbf{f}$.

Let $\mathbf{g}, \mathbf{h}$ be defined as in Eqs. 1 and 2 and $\mathbf{B}^1 = (B_1^1, B_2^1)$, $\mathbf{B}^2 = (B_1^2, B_2^2)$ be as defined in 2.2. There exist no TAC for $\mathbf{g}$ having $\mathbf{B}^1$ or $\mathbf{B}^2$ as box: $-2^{-\delta}$, 0 and $2^{-\delta}$ are three points of B_1^1 and B_1^2; 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 B_2^1 while $(g_2)_0 = z_2^2$ has 1 root of multiplicity 2 in B_2^1: there is no $\mathbf{m}$ that satisfy condition (*ii*) of def. 3. In the case of $\mathbf{B}^2$, $(g_2)_{-2^{-\delta}}$ and $(g_2)_{2^{-\delta}}$ have both 1 root of multiplicity 1 in B_2^2 while $(g_2)_0$ has no root in 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 $h_2(z_1, z_2) = (z_2 + 2^\delta z_1^2)^2(z_2 - 1)z_2$. 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.

One can count the sum of multiplicities of the solutions in a TAC with the following formula:

Theorem 3. *Let* $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ *be a TAC of height* > 1. *Then*

$$\#(\Delta(\mathbf{B}), \mathbf{f}) = \dot{\mathbf{m}} \times \#(\Delta(\overline{\mathbf{B}}), \overline{\mathbf{f}})$$

Proof. Let $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ be a TAC and $\mathbf{a}^1, \ldots, \mathbf{a}^l$ be the solutions in the cluster $Z(\Delta(\mathbf{B}), \mathbf{f})$. From the definition of a TAC, one has $\forall 1 \leq i \leq l$, $(\dot{\mathbf{B}}, \dot{\mathbf{m}}, (\dot{\mathbf{f}})_{\overline{\mathbf{a^i}}})$ is an algebraic cluster and $\#(\Delta(\dot{\mathbf{B}}), (\dot{\mathbf{f}})_{\overline{\mathbf{a^i}}}) = \dot{\mathbf{m}}$. One can then apply Thm. 2 to obtain the consequence of Thm. 3. □

2.4 A sufficient condition for a tower to be a TAC.

The T_*^L test described in Sec. 1.4 can be used to prove inductively that a tower is a tower of algebraic clusters, as claimed in Thm. 4. Let us first introduce the following technical lemma.

Lemma 1. *Let $n > 1$, $\mathbf{z} = (z_1, \ldots, z_n)$, $L \geq 1$ an integer, $g \in \mathbb{C}[\mathbf{z}]$ and $\mathbf{c} \in \mathbb{C}^{n-1}$. Let $\delta_n = \delta_n(L, g, \mathbf{c}) = L + (n-1)(1 + \lceil \log(\frac{(d+1)M^d}{\min(1, \frac{1}{d||g||})}) \rceil)$, where d is an upper bound on the partial degrees of g and $M = ||\mathbf{c}|| + 1$.*

If $\mathbf{b} \in \mathbb{C}^{n-1}$ is a δ_n-bit approximation of $\mathbf{c}$ and $\widetilde{g} \in \mathbb{C}[\mathbf{z}]$ is a δ_n-bit approximation of g then $(\widetilde{g})_{\mathbf{c}}$ is an L-bit approximation of $(g)_{\mathbf{b}}$.

For convenience, in the following, we denote $\delta_n(L, g, \mathbf{c})$ as $\delta(L, g, \mathbf{c})$ when the value of n is the number of variables of g. The proof of Lemma 1 is done by induction on the dimension. The main technical tool is Lemma 2 bounding the loss of precision by evaluation of a univariate polynomial.

Lemma 2. *Let g be a univariate polynomial in $\mathbb{C}[z]$ of degree d, $c \in \mathbb{C}$. Let L, M and δ_2 be defined as in Lemma 1.*

If $b \in \mathbb{C}$ is a δ_2-bit approximation of c and $\widetilde{g} \in \mathbb{C}[z]$ is a δ_2-bit approximation of g then $(\widetilde{g})_c = \widetilde{g}(c)$ is an L-bit approximation of $(g)_b = g(b)$.

Proof. Writing $g = \sum_{i=0}^d g_i z^i$ and $\widetilde{g} = \sum_{i=0}^d \widetilde{g}_i z^i$, one has $||\widetilde{g} - g|| = \max_{i=1\ldots d} |\widetilde{g}_i - g_i| \leq 2^{-\delta_2}$.

The triangular inequality yields $|\widetilde{g}(c) - g(b)| \leq |\widetilde{g}(c) - g(c)| + |g(c) - g(b)|$.

The first term is bounded as follows:

$$
\begin{aligned}
|\widetilde{g}(c) - g(c)| &\leq \sum_{i=0}^{d} |\widetilde{g}_i - g_i||c|^i \leq ||\widetilde{g} - g|| \sum_{i=0}^{d} (|c|+1)^d \leq 2^{-\delta_2}(d+1)M^d \\
&\leq \frac{2^{-L}}{2} 2^{-\lceil \log(\frac{(d+1)M^d}{\min(1, \frac{1}{d||g||})}) \rceil} (d+1)M^d \leq \frac{2^{-L}}{2} 2^{-\log(\frac{(d+1)M^d}{\min(1, \frac{1}{d||g||})})} (d+1)M^d \\
&\leq \frac{2^{-L}}{2} \frac{\min(1, \frac{1}{d||g||})}{(d+1)M^d} (d+1)M^d \leq \frac{2^{-L}}{2}
\end{aligned}
$$

For the second term, integrating g' on the segment $[b, c]$ yields that $|g(c) - g(b)| = |\int_{[b,c]} g'(t)| \leq \int_{[b,c]} |g'(t)| \leq |b - c| \max_{t \in [b,c]} |g'(t)|$. In addition, b is a δ_2-bit approximation of c so that $|b - c| \leq 2^{-\delta_2} \leq 1$ since $\delta_2 \geq 1$. One can write $t \in [b, c]$ as $t = c + \lambda(b - c)$ with $\lambda \in [0, 1]$, thus $|t| \leq |c| + |c - b| \leq |c| + 2^{-\delta_2} \leq |c| + 1 = M$. The second term is thus bounded as follows:

$$
\begin{aligned}
|g(c) - g(b)| &\leq |c - b| \max_{t \in [b,c]} |g'(t)| \leq 2^{-\delta_2} \max_{t \in [b,c]} \sum_{i=0}^{d} i|g_i||t|^{i-1} \leq 2^{-\delta_2} ||g|| \sum_{i=1}^{d} dM^{i-1} \\
&\leq 2^{-\delta_2} d^2 ||g|| M^{d-1} \leq \frac{2^{-L}}{2} \frac{\min(1, \frac{1}{d||g||})}{(d+1)M^d} d^2 ||g|| M^{d-1} \\
&\leq \frac{2^{-L}}{2} \min(1, \frac{1}{d||g||}) d||g|| \frac{dM^{d-1}}{(d+1)M^d} \leq \frac{2^{-L}}{2}
\end{aligned}
$$

One thus concludes that $|\widetilde{g}(c)-g(b)|\leq 2^{-L}$, that is $\widetilde{g}(c)$ is an L-bit approximation of $g(b)$. □

Proof (of Lemma 1). The initial case of the induction is for $n=2$, $\mathbf{z}=(z_1,z_2)$, $g\in\mathbb{C}[\mathbf{z}]$ and $c\in\mathbb{C}$. Let b be a δ_2-bit approximation of c and $\widetilde{g}\in\mathbb{C}[\mathbf{z}]$ be a δ_2-bit approximation of g. Writing $g=\sum_{i=0}^{d} g_i(z_1)z_2^i$ and $\widetilde{g}=\sum_{i=0}^{d}\widetilde{g}_i(z_1)z_2^i$, one has $||\widetilde{g}_i-g_i||\leq||\widetilde{g}-g||\leq 2^{-\delta_2}$.

By Lemma 2, for all $i\in\{0,\ldots,d\}$, $\widetilde{g}_i(c)$ is a L-bit approximation of $g_i(b)$ and thus

$$||(\widetilde{g})_c-(g)_b||=\max_{i\in\{0,\ldots,d\}}|\widetilde{g}_i(c)-g_i(b)|\leq 2^{-L}$$

So that $(\widetilde{g})_c$ is an L-bit approximation of $(g)_b$.

Assume the result holds for $n-1$. Let $g,\widetilde{g}\in\mathbb{C}[\mathbf{z}]=\mathbb{C}[z_1,\ldots,z_n]$ and $\mathbf{c},\mathbf{b}\in\mathbb{C}^{n-1}$ such that $\mathbf{b}$ is a δ_n-bit approximation of $\mathbf{c}$ and $\widetilde{g}$ is a δ_n-bit approximation of g. Writing $g=\sum_{i=0}^{i=d} g_i(z_1,\ldots,z_{n-1})z_n^i$ and $\widetilde{g}=\sum_{i=0}^{i=d}\widetilde{g}_i(z_1,\ldots,z_{n-1})z_n^i$, one has $\widetilde{g}_i$ is a δ_n-bit approximation of g_i and $(b_1,\ldots,b_{n-2})$ is δ_n-bit approximation of $(c_1,\ldots,c_{n-2})$. The induction hypothesis yields that $(\widetilde{g}_i)_{(c_1,\ldots,c_{n-2})}$ is a δ_2-bit approximation of $(g_i)_{(b_1,\ldots,b_{n-2})}$.

In addition, b_{n-1} is a δ_n-bit approximation of c_{n-1} and hence also a δ_2-bit approximation c_{n-1}. By Lemma 2, for all $i\in\{0,\ldots,d\}$, $(\widetilde{g}_i)_{(c_1,\ldots,c_{n-2})}(c_{n-1})$ is a L-bit approximation of $(g_i)_{(b_1,\ldots,b_{n-2})}(b_{n-1})$. One concludes that

$$||(\widetilde{g})_{\mathbf{c}}-(g)_{\mathbf{b}}||=\max_{i\in\{0,\ldots,d\}}|(\widetilde{g}_i)_{(c_1,\ldots,c_{n-2})}(c_{n-1})-(g_i)_{(b_1,\ldots,b_{n-2})}(b_{n-1})|\leq 2^{-L}$$

□

Based on Lemma 1, we show that for a given tower $(\mathbf{B},\mathbf{m},\mathbf{f})$ of height >1 such that $(\overline{\mathbf{B}},\overline{\mathbf{m}},\overline{\mathbf{f}})$ is a TAC, and providing that the width of $\overline{\mathbf{B}}$ is small enough, the T_*^L test can be used to certify that $(\mathbf{B},\mathbf{m},\mathbf{f})$ is a TAC.

Theorem 4. *Let $(\mathbf{B},\mathbf{m},\mathbf{f})$ be a tower of height >1 such that $(\overline{\mathbf{B}},\overline{\mathbf{m}},\overline{\mathbf{f}})$ is a TAC. Let $\mathbf{c}$ be the center of $\overline{\mathbf{B}}$, $\delta(L,\dot{\mathbf{f}},\mathbf{c})$ be defined as in Lemma 1 and $\widetilde{f}$ be a $\delta(L,\dot{\mathbf{f}},\mathbf{c})$-bit approximation of $\dot{\mathbf{f}}$.*

(i) If $r(\Delta(\overline{\mathbf{B}}))\leq 2^{-\delta(L,\dot{\mathbf{f}},\mathbf{c})}$ then $(\widetilde{f})_{\mathbf{c}}$ is an L-bit approximation of $(\dot{\mathbf{f}})_{\mathbf{b}}$ for any $\mathbf{b}\in\Delta(\overline{\mathbf{B}})$.

(ii) If $T_*^L(\Delta(\dot{\mathbf{B}}),L,(\widetilde{f})_{\mathbf{c}})=T_*^L(3\Delta(\dot{\mathbf{B}}),L,(\widetilde{f})_{\mathbf{c}})=\dot{\mathbf{m}}$, then $(\mathbf{B},\mathbf{m},\mathbf{f})$ is a TAC.

Proof. (i): if $\mathbf{c}$ is the center of $\Delta(\overline{\mathbf{B}})$ and $r(\Delta(\overline{\mathbf{B}}))\leq 2^{-\delta(L,\dot{\mathbf{f}},\mathbf{c})}$ then $\forall\mathbf{b}\in\Delta(\overline{\mathbf{B}})$, $||\mathbf{c}-\mathbf{b}||\leq 2^{-\delta(L,\dot{\mathbf{f}},\mathbf{c})}$ and from Lemma 1, $(\widetilde{f})_{\mathbf{c}}$ is an L-bit approximation of $(\dot{\mathbf{f}})_{\mathbf{b}}$. (ii): from (i), $(\widetilde{f})_{\mathbf{c}}$ is an L-bit approximation of $(\dot{\mathbf{f}})_{\mathbf{b}}$ for any $\mathbf{b}\in\Delta(\overline{\mathbf{B}})$ and, from the specification of the T_*^L test (see Algo. 1 in subsec. 1.4) one has $\#(\Delta(\dot{\mathbf{B}}),(\dot{\mathbf{f}})_{\mathbf{b}})=\#(3\Delta(\dot{\mathbf{B}}),(\dot{\mathbf{f}})_{\mathbf{b}})=\dot{\mathbf{m}}$ for any $\mathbf{b}\in\Delta(\overline{\mathbf{B}})$. As a consequence, the triple $(\dot{\mathbf{B}},\dot{\mathbf{m}},(\dot{\mathbf{f}})_{\mathbf{b}})$ is an algebraic cluster for any $\mathbf{b}\in\Delta(\overline{\mathbf{B}})$ and $(\mathbf{B},\mathbf{m},\mathbf{f})$ is a TAC. □

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

Input: A triangular system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ where $\mathbf{f} = (f_1, \ldots, f_n)$, a polybox $\mathbf{B} = (B_1, \ldots, B_n) \subset \mathbb{C}^n$, a real number $\epsilon \in \mathbb{R}$ s.t. $0 < \epsilon < \frac{1}{2}$.
Output: A list $\{(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f}) | 1 \leq j \leq l\}$ of TACs of height n satisfying (a), (b), (c), (d) of Prop. 1.

1: $\mathcal{R} \leftarrow \emptyset$
2: **if** $n > 1$ **then**
3: $\quad \mathcal{S} \leftarrow clusterTriSys(\overline{\mathbf{f}}, \overline{\mathbf{B}}, \epsilon)$ *//recursive call; $\mathcal{S}$ contains TACs of height $n-1$*
4: $\quad$ **while** $\mathcal{S}$ is not empty **do**
5: $\quad\quad (\mathbf{B}^{cur}, \mathbf{m}^{cur}, \overline{\mathbf{f}}) \leftarrow pop(\mathcal{S})$
6: $\quad\quad f \leftarrow ApproximateInFiber(., \dot{\mathbf{f}}, (\mathbf{B}^{cur}, \mathbf{m}^{cur}, \overline{\mathbf{f}}), \mathcal{S})$
7: $\quad\quad \mathcal{T} \leftarrow clusterPol(f, \dot{\mathbf{B}}, \epsilon)$ *//possibly appends TACs of height $n-1$ to $\mathcal{S}$*
8: $\quad\quad$ **for** $(B^j, m^j) \in \mathcal{T}$ **do**
9: $\quad\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{((\mathbf{B}^{cur}, B^j), (\mathbf{m}^{cur}, m^j), (\overline{\mathbf{f}}, \dot{\mathbf{f}}))\}$
10: **else** *//$n = 1$*
11: $\quad \mathcal{T} \leftarrow clusterPol(f_1, B_1, \epsilon)$ *//the terminal case*
12: $\quad$ **for** $(B^j, m^j) \in \mathcal{T}$ **do**
13: $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{(B^j, m^j, f_1)\}$
14: **return** $\mathcal{R}$

3 Clustering the solutions of a triangular system

Our main procedure $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$, defined in Algo. 3, computes natural clusters of size at most ϵ of the triangular system $\mathbf{f} = \mathbf{0}$ where $\mathbf{f} = (f_1, \ldots, f_n)$, in the polybox $\mathbf{B}$. $f_1, \ldots, f_n$ are given as procedures returning L-bits approximations of $f_1, \ldots, f_n$ for a given L. It returns the clusters in a list $\mathcal{R}$ of TACs of height n. Proposition 1 shows that it solves the local solution clustering problem.

Proposition 1. *Let $n \geq 1$, $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ be a triangular system, $\mathbf{B}$ be a polybox and ϵ be a real number, $0 < \epsilon \leq \frac{1}{2}$. Algorithm $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$ terminates and returns a list $\{(\mathbf{B}^1, \mathbf{m}^1, \mathbf{f}), \ldots, (\mathbf{B}^l, \mathbf{m}^l, \mathbf{f})\}$ with $l \geq 0$, such that:*

(a) the polydiscs $\Delta(\mathbf{B}^j)$ are pairwise disjoint with $r(\Delta(\mathbf{B}^j)) \leq \epsilon$,
(b) $\forall 1 \leq j \leq l$, let $\mathbf{m}^j = (m_1^j, \ldots, m_n^j)$; then $\Pi_{k=1}^n m_k^j = \#(\mathbf{B}^j, \mathbf{f}) = \#(\Delta(\mathbf{B}^j), \mathbf{f}) = \#(3\Delta(\mathbf{B}^j), \mathbf{f})$ (i.e. $Z(\Delta(\mathbf{B}^j), \mathbf{f})$ is a natural cluster),
(c) $\bigcup_{1 \leq j \leq l} Z(\Delta(\mathbf{B}^j), \mathbf{f}) \subseteq Z(2\mathbf{B}, \mathbf{f})$
(d) $Z(\mathbf{B}, \mathbf{f}) \subseteq \bigcup_{1 \leq j \leq l} Z(\Delta(\mathbf{B}^j), \mathbf{f})$.

3.1 The terminal case ($n = 1$)

The terminal case, *i.e.* finding clusters of roots of $f_1 \in \mathbb{C}[z_1]$ of size less that ϵ in the square complex box $B_1 \subset \mathbb{C}$ is addressed by calling $clusterPol$ (Algo. 2). For each pair (B^j, m^j) in the output of $clusterPol(f_1, B_1, \epsilon)$, the TAC of height 1, $(\mathbf{B}^j, \mathbf{m}^j, f_1)$, is constructed in Step 13 of Algo. 3.

Since the procedure $clusterPol$ solves the clustering problem for $n = 1$, Prop. 1 is true for $n = 1$.

3.2 The non-terminal case ($n > 1$)

Let us suppose that $n > 1$ and that Prop. 1 is true until rank $n-1$. The recursive call $clusterTriSys(\overline{\mathbf{f}}, \overline{\mathbf{B}}, \epsilon)$ in Step 3 of Algo. 3 terminates and returns a list $\mathcal{S}$ of TACs of height $n-1$. If $\mathcal{S}$ is empty, $Z(\overline{\mathbf{B}}, \overline{\mathbf{f}})$ is empty and $clusterTriSys(\overline{\mathbf{f}}, \overline{\mathbf{B}}, \epsilon)$ terminates and returns an empty list $\mathcal{R}$.

Suppose now that $\mathcal{S}$ is not empty, and let $\mathcal{S} = \{(\mathbf{B}^1, \mathbf{m}^1, \mathbf{f}), \ldots, (\mathbf{B}^l, \mathbf{m}^l, \mathbf{f})\}$ with $l \geq 1$. By hypothesis at rank $n-1$, one has

(a') the polydiscs $\Delta(\mathbf{B}^j)$ are pairwise disjoint with $r(\Delta(\mathbf{B}^j)) \leq \epsilon$,
(b') $\forall 1 \leq j \leq l$, let $\mathbf{m}^j = (m_1^j, \ldots, m_n^j)$; then $\Pi_{k=1}^n m_k^j = \#(\mathbf{B}^j, \overline{\mathbf{f}}) = \#(\Delta(\mathbf{B}^j), \overline{\mathbf{f}}) = \#(3\Delta(\mathbf{B}^j), \overline{\mathbf{f}})$,
(c') $\bigcup_{1 \leq j \leq l} Z(\Delta(\mathbf{B}^j), \overline{\mathbf{f}}) \subseteq Z(2\overline{\mathbf{B}}, \overline{\mathbf{f}})$
(d') $Z(\overline{\mathbf{B}}, \overline{\mathbf{f}}) \subseteq \bigcup_{1 \leq j \leq l} Z(\Delta(\mathbf{B}^j), \overline{\mathbf{f}})$.

As a consequence of (a'), (b') and (d'), any solution $\mathbf{a}$ of $\mathbf{f}$ in $\mathbf{B}$ is in exactly one $(\mathbf{B}^j, \dot{\mathbf{B}})$. The next step is then, for all $\Delta(\mathbf{B}^j)$ in $\mathcal{S}$, to isolate the clusters of solutions of the polynomial $\dot{\mathbf{f}}$ specialized in the fiber $\mathbf{c}$, where $\mathbf{c} = c(\mathbf{B}^j)$.

In Step 5 of Algo. 3, $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \overline{\mathbf{f}})$ is a TAC of height $n-1$ in $\mathcal{S}$, let $\mathbf{c}$ be the center of $\mathbf{B}^{cur}$ and $L > 1$ be an integer. According to Thm. 4, there exists $\delta > L$ so that if $r(\Delta(\mathbf{B}^{cur})) \leq 2^{-\delta}$, $\widetilde{f}$ is an δ-bit approximation of $\dot{\mathbf{f}}$ and $B \subset \dot{\mathbf{B}}$ satisfies $T_*^L(\Delta(B), L, (\widetilde{f})_{\mathbf{c}}) = T_*^L(3\Delta(B), L, (\widetilde{f})_{\mathbf{c}}) = \dot{\mathbf{m}}$, then $((\mathbf{B}^{cur}, B), (\mathbf{m}^{cur}, m), \mathbf{f})$ is a TAC of height n. In other words, when $r(\Delta(\mathbf{B}^{cur}))$ is small enough, one can find all the clusters of solutions of radius less that ϵ of $(\dot{\mathbf{f}})_{\mathbf{c}}$ in $\dot{\mathbf{B}}$ with $clusterPol$.

However $r(\Delta(\mathbf{B}^{cur}))$ may not be small enough and $\mathbf{B}^{cur}$ has to be refined by calling $clusterTriSys(\overline{\mathbf{f}}, \mathbf{B}^{cur}, \epsilon')$ with the suitable value of ϵ'; notice that this will possibly split the TAC $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$ of height $n-1$ into several TACs. In that case, $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$ is taken as one of these TACs, and the others are appended to $\mathcal{S}$. This refinement of $\mathbf{B}^{cur}$ is done by the call to $ApproximateInFiber$ that computes an L-bit approximation of $(\dot{\mathbf{f}})_{\mathbf{b}}$ for any $\mathbf{b} \in \mathbf{\Delta}(\mathbf{B}^{cur})$. The function $ApproximateInFiber$ is described in Algo. 4 and satisfies the following proposition whose proof is postponed to Sec. 3.3.

Proposition 2. *Let $L > 1$, $n \geq 1$, $f \in \mathbb{C}[z_1, \ldots, z_n]$, $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ be a TAC of height $n-1$ and $\mathcal{S}$ be a list of TACs of height $n-1$. Suppose that Prop. 1 is true until rank $n-1$, then $ApproximateInFiber(L, f, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{S})$ terminates. After its execution, $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ is modified in $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ and $\mathcal{S}$ in $\mathcal{S}'$, satisfying: for any $\mathbf{a} \in Z(\mathbf{B}, \mathbf{f})$, there is a unique TAC $(\mathbf{B}'', \mathbf{m}'', \mathbf{f})$ in $\{(\mathbf{B}', \mathbf{m}', \mathbf{f})\} \cup \mathcal{S}'$ such that $\mathbf{a} \in Z(\mathbf{B}'', \mathbf{f})$.*

The procedure returns an L-bit approximation $\tilde{f}$ of $(f)_{\mathbf{b}}$ for any $\mathbf{b} \in \mathbf{\Delta}(\mathbf{B}')$.

In Step 7 of Algo. 3, $ApproximateInFiber(L, \dot{\mathbf{f}}, (\mathbf{B}^{cur}, \mathbf{m}^{cur}, \overline{\mathbf{f}}), \mathcal{S})$ is used as the approximation function for $clusterPol$, to find the clusters of solutions of $\dot{\mathbf{f}}$ over the fiber $\mathbf{B}^{cur}$.

To finish the proof of Prop. 1, we make the following remarks. At any execution of the **while** loop in Step 4 of Algo. 3:

(i) for the TAC $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$ of height $n-1$ obtained in Step 5 of Algo. 3, the call to *clusterPol* in Step 7 of Algo. 3 terminates, returns a list $\{(B^j, m^j) | 1 \leq j \leq l\}$ and modifies $(\mathbf{B}^{cur}, \mathbf{m}^{cur}, \mathbf{f})$ in $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ satisfying:
  (a") the $\Delta((\mathbf{B}', B^j))$'s are pairwise disjoint polydiscs of radius less than ϵ,
  (b") the $((\mathbf{B}', B^j), (\mathbf{m}', m^j), \mathbf{f})$'s are TACs of height n,
  (c") $\bigcup_j Z((\mathbf{B}', B^j), \mathbf{f}) \subseteq Z((\mathbf{B}, , 2\dot{\mathbf{B}}), \mathbf{f})$,
  (d") for any $\mathbf{a} \in Z((\mathbf{B}^{cur}, \dot{\mathbf{B}}), \mathbf{f})$, $\mathbf{a}$ is either in a $(\mathbf{B}', B^j)$, or in a $(\mathbf{B}'', \dot{\mathbf{B}})$ where $(\mathbf{B}'', \mathbf{m}'', \bar{\mathbf{f}})$ is a TAC in $\mathcal{S}$,

(ii) $\mathcal{R}$ is a list of TACs of height n satisfying properties $(a), (b), (c)$ of Prop. 1

(iii) $\mathcal{S}$ is a list of TACs of height $n-1$ satisfying properties $(a'), (b'), (c')$ stated above,

(iv) for any $\mathbf{a} \in Z(\mathbf{B}, \mathbf{f})$, there is either a TAC $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ of height n in $\mathcal{R}$ so that $\mathbf{a} \in \mathbf{B}'$ or a TAC $(\mathbf{B}'', \mathbf{m}'', \bar{\mathbf{f}})$ of height $n-1$ in $\mathcal{S}$ so that $\bar{\mathbf{a}} \in \mathbf{B}''$ and $\dot{\mathbf{a}} \in \dot{\mathbf{B}}$

(i) is a consequence of Prop. 2 and Lem. **??** for the termination, (a") and (c"); (b") and (d") are consequences of Prop. 2 and Thm. 4.
(ii) is a direct consequence of (i), (iii) is a consequence of Prop. 1 at rank $n-1$ and Prop. 2, and (iv) is a consequence of (i), (ii) and (iii).

After the execution of the **while** loop in Step 4 of Algo. 3, $\mathcal{S}$ is empty and from (iv) above, $\mathcal{R}$ is a list of TACs of height n satisfying properties $(a), (b), (c), (d)$ of Prop. 1, which concludes the proof of Prop. 1.

3.3 Approximating a polynomial specialized in a fiber

The approximation function $ApproximateInFiber(L, f, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{S})$ defined in Algo. 4 computes, for a given $L > 1$, a given polynomial $f \in \mathbb{C}[\mathbf{z}]$ and a given TAC $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ of height $n-1$ an L-bit approximation of $(f)_{\mathbf{b}}$ for any $\mathbf{b} \in \Delta(\mathbf{B})$; this requires the width of $\mathbf{B}$ to be small enough. If it is not, $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ is refined, *i.e.* $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ is modified in $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ so that the width of $\mathbf{B}'$ is small enough. This refinement process may require to split $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ into several TACs $(\mathbf{B}^j, \mathbf{m}^j, \mathbf{f})$ for $1 \leq j \leq l$ so that any solution of $\mathbf{f}$ in $\mathbf{B}$ is in exactly one $\mathbf{B}^j$. In that case, $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ is chosen as $(\mathbf{B}^1, \mathbf{m}^1, \mathbf{f})$ and $ApproximateInFiber(L, f, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{S})$ will output an L-bit approximation of $(f)_{\mathbf{b}}$ for any $\mathbf{b} \in \Delta(\mathbf{B}')$ and append $(\mathbf{B}^2, \mathbf{m}^2, \mathbf{f}), \ldots, (\mathbf{B}^l, \mathbf{m}^l, \mathbf{f})$ to the list $\mathcal{S}$ of TACs of height $n-1$. A TAC $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ of height $n-1$ with $r(\Delta(\mathbf{B})) \leq \epsilon'$, is refined at size $\epsilon < \epsilon'$ with $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$.

In Algo. 4 and in the proof of Prop. 2 below we assume that we can compute $\delta(L, f, \mathbf{c})$ as defined in Lemma. 1. In practice, we do not compute it but increase the value of L' with trials and errors using ball arithmetic (see Subsec. 4.1).

Proof (of Prop. 2). From the termination and correctness of $clusterTriSys(\mathbf{f}, \mathbf{B}, \epsilon)$ at rank $n-1$ and from Lemma 1, if the condition of the **while** loop is reached, $ApproximateInFiber(L, f, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{S})$ terminates and is correct. In order to prove the termination of the **while** loop, we remark that $\delta(L, g, \mathbf{c})$ is bounded for $\mathbf{c} \in \mathbf{B}$. □

Algorithm 4 *ApproximateInFiber*$(L, f, (\mathbf{B}, \mathbf{m}, \mathbf{f}), \mathcal{S})$

Input: A precision $L \in \mathbb{N}$ s.t $L > 1$, $f \in \mathbb{C}[z_1, \ldots, z_n]$, a TAC $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ of height $n-1$ and a list $\mathcal{S}$ of TACs of height $n-1$.

Output: Modifies in-place $(\mathbf{B}, \mathbf{m}, \mathbf{f})$ in $(\mathbf{B}', \mathbf{m}', \mathbf{f})$ and $\mathcal{S}$ in $\mathcal{S}'$ satisfying $\mathcal{S} \subseteq \mathcal{S}'$ and for any $\mathbf{a} \in Z(\mathbf{B}, \mathbf{f})$, there is a unique TAC $(\mathbf{B}'', \mathbf{m}'', \mathbf{f})$ in $\{(\mathbf{B}', \mathbf{m}', \mathbf{f})\} \cup \mathcal{S}'$ such that $\mathbf{a} \in Z(\mathbf{B}'', \mathbf{f})$.
Returns an L-bit approximation $\tilde{f}$ of $(f)_{\mathbf{b}}$ for any $\mathbf{b} \in \mathbf{\Delta}(\mathbf{B}')$.

1: $(\mathbf{B}', \mathbf{m}', \mathbf{f}) \leftarrow (\mathbf{B}, \mathbf{m}, \mathbf{f})$
2: $\mathbf{c} \leftarrow$ center of $\mathbf{B}'$
3: $L' \leftarrow \delta(L, f, \mathbf{c})$ as defined in Lemma 1
4: **while** $r(\Delta(\mathbf{B}')) > 2^{-L'}$ **do**
5: $\quad \mathcal{T} \leftarrow clusterTriSys(\mathbf{f}, \mathbf{B}', 2^{-L'})$
6: $\quad (\mathbf{B}', \mathbf{m}', \mathbf{f}) \leftarrow pop(\mathcal{T})$
7: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{T}$
8: $\quad \mathbf{c} \leftarrow$ center of $\mathbf{B}'$
9: $\quad L' \leftarrow \delta(L, f, \mathbf{c})$ as defined in Lemma 1
10: $\tilde{f} \leftarrow L'$-bit approximation of f
11: $\tilde{f} \leftarrow$ substitute $(z_1, \ldots, z_{n-1})$ with $\mathbf{c}$ in $\tilde{f}$
12: $(\mathbf{B}, \mathbf{m}, \mathbf{f}) \leftarrow (\mathbf{B}', \mathbf{m}', \mathbf{f})$
13: **return** $\tilde{f}$

4 Implementation and benchmarks

We implemented in `Julia`[4] a prototype version of the procedure *clusterTriSys* given in Algo. 3, named hereafter `tcluster`. This prototype uses, as routine for clustering roots of univariate polynomials given by approximations, the function `ccluster` provided by the `Julia` package `Ccluster.jl`.[5] implementing procedure of Algo. 2.

In `tcluster`, the way a polynomial specialized in a fiber is approximated differs from the procedure depicted in Algo. 4. We describe the implemented procedure in Sec. 4.1. In Sec. 4.2 we show how `tcluster` performs on systems having clusters of solutions. In Sec. 4.3 we propose benchmarks for solving random dense triangular systems with only regular solutions, and with solutions with multiplicities; `tcluster` is compared with two homotopy solvers. In Sec. 4.4 we use `tcluster` to cluster solutions of system triangularized with regular chains.

All the timings given below have been obtained on a Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz machine with linux.

4.1 Approximating a polynomial in a fiber

The description of procedure *ApproximateInFiber* in Algo. 4 uses an upper bound of the loss of precision when evaluating an L'-bit approximation of a polynomial $f \in \mathbb{C}[\mathbf{z}]$ on $\mathbf{c} \in \mathbb{C}^{n-1}$ which is an L'-bit approximation of $\mathbf{b} \in$

[4] https://julialang.org/
[5] https://github.com/rimbach/Ccluster.jl

$\mathbb{C}^{n-1}$. Instead of using this upper bound, we rely in our implementation on ball arithmetic, a variant of interval arithmetic, to get an L-bit approximation of f specialized in any point of $\Delta(\mathbf{B})$ where $\mathbf{B} = (B_1, \ldots, B_{n-1})$ as follows.

A complex ball is a square[6] complex box. We note $\square\mathbb{C}[\mathbf{z}]$ the set of polynomials in $\mathbf{z}$ which coefficients are square complex boxes. Let $\square f \in \square\mathbb{C}[\mathbf{z}]$; we note $w(\square f)$ the width of its widest coefficient and $c(\square f)$ the polynomial in $\mathbb{C}[\mathbf{z}]$ which coefficients are the centers of the coefficients of $\square f$. Remark that if $w(\square f) \leq 2^{-L'}$, then $c(\square f)$ is an L'-bit approximation of any $f \in \square f$ where $f \in \mathbb{C}[\mathbf{z}]$ (f and $\square f$ are considered as the vectors of their coefficients in the monomial basis for $\in$). For a given polybox $\mathbf{B} = (B_1, \ldots, B_{n-1})$, one can evaluate $\square f$ on $\frac{3}{2}\mathbf{B}$ to obtain $(\square f)_{\frac{3}{2}\mathbf{B}}$ [7] satisfying $\forall \mathbf{b} \in \Delta(\mathbf{B}), (f)_{\mathbf{b}} \in (\square f)_{\frac{3}{2}\mathbf{B}}$. If $w((\square f)_{\frac{3}{2}\mathbf{B}}) \leq 2^{-L}$, then $c((\square f)_{\frac{3}{2}\mathbf{B}})$ is an L-bit approximation of $(f)_{\mathbf{b}}$, $\forall \mathbf{b} \in \Delta(\mathbf{B})$; otherwise we increase L', refine $\mathbf{B}$ with $clusterTriSys(\mathbf{f}, \mathbf{B}, 2^{-L'})$ (and possibly split the cluster), use the black-box to obtain $\square f$ with $w(\square f) \leq 2^{-L'}$ and evaluate $\square f$ on $\frac{3}{2}\mathbf{B}$ until $w((\square f)_{\frac{3}{2}\mathbf{B}}) \leq 2^{-L}$.

We used the ball arithmetic library `arb` (see [14]), interfaced in `Julia` through the package `Nemo`[8] in our implementation.

4.2 Clustering ability

We present here the clustering ability of our solver on toy examples of triangular systems though to have solutions in clusters. Consider the triangular systems $\mathbf{g} = (f, g_2) = \mathbf{0}$ and $\mathbf{h} = (f, h_2) = \mathbf{0}$ defined as:

$$(\mathbf{g} = \mathbf{0}) : \begin{cases} z_1^{d_1} - (2^\delta z_1 - 1)^c = 0 \\ z_2^{d_2} z_1^{d_2} - 1 = 0 \end{cases} \tag{3}$$

$$(\mathbf{h} = \mathbf{0}) : \begin{cases} z_1^{d_1} - (2^\delta z_1 - 1)^c = 0 \\ z_2^{d_2} - z_1^{d_2} = 0 \end{cases} \tag{4}$$

f has a cluster of c roots of multiplicity 1 in a disk centered in $2^{-\delta}$ with radius 2^{-b} where $b = (d_1\delta + \delta - 1)/c$ (see [17]); we note S_1 the set of roots in this cluster. When $d_1 > c > 1$, the roots in S_1 have modulus less than $2^{-\delta} + 2^{-b} \leq 2^{-\delta+1} = \hat{\gamma}$. We note S_2 the $d_1 - c$ roots of f that are not in S_1. The roots in S_2 have all multiplicity 1 and have modulus of the order of $\gamma = 2^a$ where $a = \frac{c\delta}{d_1 - c}$.

The d_2 roots of g_2 are on a complex circle centered in 0 with radius greater than $\hat{\gamma}^{-1}$ when $z_1 \in S_1$, and of order γ^{-1} when $z_1 \in S_2$. The d_2 roots of h_2 are on a complex circle centered in 0 with radius less than $\hat{\gamma}$ when $z_1 \in S_1$, and of order γ when $z_1 \in S_2$.

We suppose now $d_1 = 30$, $c = 10$, $\delta = 128$ and $d_2 = 10$. Therefore, one has $\gamma = 2^{64}$, $b \simeq 397$ and $\hat{\gamma} = 2^{-127}$; all the solutions of $\mathbf{g} = \mathbf{0}$ and $\mathbf{h} = \mathbf{0}$

[6] here we restrict complex balls to square boxes for the sake of simplicity. General description do not need this restriction.

[7] this is defined up to an evaluation scheme; in practice we use the multivariate extension of the Horner scheme given by the ordering $z_1, \ldots, z_n$ of the variables.

[8] http://nemocas.org/links.html

	$\mathbf{g} = \mathbf{0}$			$\mathbf{h} = \mathbf{0}$		
$log_2(\epsilon)$	#Sols	t (s)	(m,M)	#Sols	t (s)	(m,M)
-53	$0 + 30 \times 10$	0.79	(-212,- 424)	$200 + 0 \times 10 + 1 \times 100$	0.92	(-212, -212)
-106	$200 + 10 \times 10$	1.01	(-212,- 424)	$200 + 0 \times 10 + 1 \times 100$	0.95	(-212, -424)
-212	$300 + 0 \times 10$	9.09	(-424,- 848)	$200 + 10 \times 10 + 0 \times 100$	1.03	(-212, -848)
-424	$300 + 0 \times 10$	9.36	(-848,-1696)	$300 + 0 \times 10 + 0 \times 100$	9.24	(-848, -848)

Table 1. Clustering the solutions of systems in Eq. 3 and Eq. 4 with $d_1 = 30$, $c = 10$, $\delta = 128$ and $d_2 = 10$ for four values of ϵ in polybox $\mathbf{B}$ centered in $\mathbf{0}$ with width 10^{40}.

are included in the polybox $\mathbf{B}$ centered in $\mathbf{0}$ with width 10^{40}. We computed clusters of solutions for the two systems with `tcluster` in the initial box $\mathbf{B}$ for four values of ϵ. Table. 1 gives: the cluster structure of the solutions in columns #Sols ($c_1 + c_2 \times 10 + c_3 \times 100$ means c_1 clusters with sum of multiplicities 1, c_2 clusters with sum of multiplicities 10 and c_3 clusters with sum of multiplicities 100), the solving time in seconds in columns t and the minimum and maximums precision required on clusters of f_1 in columns M and m (*i.e.* the log_2 of the radius of the disk isolating the clusters).

The system ($\mathbf{g} = \mathbf{0}$) has $d_1 - c = 20$ clusters of $d_2 = 10$ solutions above each root in S_2; the pairwise distance between solutions in such clusters is about $\gamma^{-1} = 2^{-64}$. It has also $c = 10$ clusters of $d_2 = 10$ solutions above the cluster of roots S_1; the pairwise distance between solutions in such clusters is less than $2^{-b} \simeq 2^{-397}$. This cluster structure is the one found by `tcluster` with $\epsilon = 2^{-53}$. When $\epsilon = 2^{-106}$, the 20 clusters above roots in S_2 are split, while the ones above roots in S_1 are preserved. When $\epsilon = 2^{-212}$, the clusters above roots in S_1 are split even if the pairwise distances between solutions in these clusters are far smaller than 2^{-212}; this happens because isolating roots of g_2 with error less than $\epsilon = 2^{-212}$ requires more precision on roots of f, as shown is column (m,M). When $\epsilon = 2^{-424}$, all the clusters are split.

The system ($\mathbf{h} = \mathbf{0}$) has $(d_1 - c)d_2 = 200$ solutions above roots in S_2 and a cluster containing $cd_2 = 100$ simple solutions above roots in S_1. The first components of the solutions in this cluster are in a complex disc of radius less that $2^{-b} \simeq 2^{-397}$ and the second components are in a complex disc of radius less than $\hat{\gamma} = 2^{-127}$. This cluster structure is found by `tcluster` with $\epsilon = 2^{-53}$ and $\epsilon = 2^{-106}$. When $\epsilon = 2^{-212}$, the cluster of 100 solutions is split in 10 clusters of 10 solutions. When $\epsilon = 2^{-424}$, the cluster is split in 100 solutions.

4.3 Benchmarks with random dense systems

We present benchmarks for randomly generated triangular systems without and with multiple solutions. We compare the efficiency and the robustness of `tcluster` and two homotopy solvers.

Homotopy solvers. Homotopy solving is a two steps proccess. First, an upper bound D on the number of solutions (counted with multiplicities) of the sys-

tem is computed. D can be the Bézout's bound, or obtained with the so-called *polyhedral homotopy* (see [12]) in which case it is sharper. In a second step, D paths are followed until they end up at a finite solution of the system, or they are decided to go to a solution at infinity.

`HOM4PS-2.0`[9] and `Bertini`[10] (see [2]) are two homotopy solvers. `HOM4PS-2.0` is known for being very fast but not robust (*i.e.* in certain cases, it can miss some solutions). It does not compute the multiplicity structure of the solutions. It implements polyhedral homotopy. `Bertini` is in general slower than `HOM4PS-2.0` but much more robust; it can use an Adaptive Multi-Precision (AMP) arithmetic and computes the multiplicities of solutions. Below `Bertini AMP` refers to `Bertini` with AMP. It takes D as the Bézout's bound.

Systems. We follow the approach of [7] to generate triangular systems with and without multiple solutions. The *type* of a triangular system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ where $\mathbf{f} = (f_1, \ldots, f_n)$ is the list $(d_1, \ldots, d_n)$ where $d_i = deg_{z_i}(f_i)$.

A random dense polynomial f_i in $\mathbb{C}[z_1, \ldots, z_i]$ of degree d_i in z_i is generated as follows. If $i > 1$, $f_i = \sum_{j=0}^{d_i} g_j z_i^j$ where $g_j \in \mathbb{C}[z_1, \ldots, z_{i-1}]$ is a random dense polynomial of degree $d_i - j$ in z_{i-1}. If $i = 1$, f_i is a dense polynomial in $\mathbb{C}[z_1]$ of degree d_1. A random dense triangular system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ of type $(d_1, \ldots, d_n)$ is obtained by generating successively random dense polynomials f_i of degrees d_i in z_i.

A triangular system $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ of type $(d_1, \ldots, d_n)$ with multiple solutions is generated as follows: f_1 is a random dense polynomial in $\mathbb{C}[z_1]$ of degree d_1. For $i = 2, \ldots, n$, $f_i = a_i^2(b_i z_i + c_i)^{\lfloor \frac{d_i+1}{2} \rfloor - \lfloor \frac{d_i}{2} \rfloor}$ where a_i is a random dense polynomial in $z_1, \ldots, z_i$ with degree $\lfloor \frac{d_i}{2} \rfloor$ in z_i and b_i, c_i are random dense polynomials in $z_1, \ldots, z_{i-1}$ of degrees d_i in z_{i-1}.

Benchmarks. In Table 2, we compare the two homotopy solvers and `tcluster` on triangular systems of random dense polynomial equations randomly generated as explained above, with integer coefficients in $[-2^8, 2^8]$, without and with multiple solutions. In both cases, we generated 5 systems of each type (see column type) and solved the systems with `HOM4PS-2.0`, `Bertini AMP` and `tcluster` with $\epsilon = 2^{-53}$. `tcluster` is first called with an initial polybox centered in $\mathbf{0}$ and width 2 (columns `tcluster` local), then with width 10^6 (columns `tcluster` global). For the systems we tested, `tcluster` with initial polybox of width 10^6 found all the solutions but in general this is not guaranteed. The columns #Sols give the average number of solutions found by each solver and the columns t(s) the average sequential times in seconds. For `tcluster`, the columns #Clus give the average number of clusters found. All the systems we generated have $d_1 \times \ldots \times d_n$ solutions (where $(d_1, \ldots, d_n)$ is the type of the system) counted with multiplicities, which is the Bézout's bound of the system; therefore the homotopy solvers have to follow this number of paths.

[9] http://www.math.nsysu.edu.tw/~leetsung/works/HOM4PS_soft.htm
[10] https://bertini.nd.edu/

	`tcluster` local		`tcluster` global		`HOM4PS-2.0`		`Bertini AMP`	
type	#Sols, #Clus	t (s)	#Sols, #Clus	t (s)	#Sols	t (s)	#Sols	t (s)
Systems with only simple solutions								
(6,6,6)	34.2, 34.2	0.07	216, 216	0.51	0	0.06	216	1.17
(9,9,9)	149, 149	0.42	729, 729	2.11	713	0.47	729	29.3
(6,6,6,6)	63.4, 63.4	0.26	1296, 1296	3.86	1274	1.37	1296	24.2
(9,9,9,9)	559, 559	2.94	6561, 6561	26.8	6036	111	6560	1605
(6,6,6,6,6)	155, 155	1.61	7776, 7776	36.6	7730	28.6	7776	318
(9,9,9,9,9)	1739, 1739	27.6	59049, 59049	416	-	-	?	>3600
Systems with multiple solutions								
(6,6)	10.8, 5.40	0.02	36, 18	0.08	36	0.00	18	3.63
(9,9)	23.8, 13.6	0.05	81, 45	0.24	67.4	0.06	45	218
(6,6,6)	35.2, 8.80	0.06	216, 54	0.33	210	0.16	54	47.9
(9,9,9)	113, 37.6	0.37	729, 225	1.72	357	18.9	?	>3600

Table 2. Solving random dense triangular systems with `tcluster`, `HOM4PS-2.0` and `Bertini AMP`. `tcluster` is called with $\epsilon = 53$ and an initial polybox centered in $\mathbf{0}$. For `tcluster` local (resp. global), the initial box has width 2 (resp. 10^6).

Systems with only simple solutions. For type (9,9,9,9,9), `Bertini AMP` has been stop after 1 hour and `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 solutions could be missing. `HOM4PS-2.0` returns incorrect results without warnings. In contrast, `tcluster` global always finds the correct number of solutions. Regardless of the correction of the results, `tcluster` global is in average faster than `Bertini AMP` for all the types of systems we tested, and is faster than `HOM4PS-2.0` for systems of types $(9, 9, 9, 9)$. Using `tcluster` to find solutions in a small initial polybox (`tcluster` local) is significantly faster than the other approaches when one is only interested in the solutions in this polybox.

Systems with multiple solutions. For a multiple solution that is well isolated from the others, `tcluster` finds a cluster containing it and reports the sum of multiplicities in the cluster (*i.e.* the multiplicity of the solution). For each system, the number of clusters found by `tcluster` global is the number of distinct complex solutions of the system. `HOM4PS-2.0` fails in finding all the solutions. `Bertini AMP` computes the multiplicity of solutions and its output is correct for all the systems we tested. For type (9,9,9), `Bertini AMP` has been stopped after 1 hour. `tcluster` global is in average faster than `Bertini AMP` for the systems we tested, and faster than `HOM4PS-2.0` for systems of type $(9, 9, 9)$.

4.4 Systems obtained by triangularization

In this subsection, we report on using `tcluster` for clustering the solutions of triangular systems $\mathbf{f}(\mathbf{z}) = \mathbf{0}$ obtained from a non-triangular system $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ with Regular Chains (RC, see for instance [1,8]). Algorithms for triangularizing

systems with RC produce a set of triangular systems $\{\mathbf{f}_1(\mathbf{z}) = \mathbf{0}, \ldots, \mathbf{f}_l(\mathbf{z}) = \mathbf{0}\}$ having distinct solutions whose union is the set of distinct solutions of $\mathbf{f}(\mathbf{z}) = \mathbf{0}$. Notice that the multiplicities of solutions are not preserved by this process. We used the `RegularChains` package of `Maple` implementing the RC algorithm described in [16].

Systems. We consider non-triangular systems $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ both classical and randomly generated. We generate sparse random systems $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ where $\mathbf{g} = (g_1, \ldots, g_n)$ and each g_i has the form $g_i(\mathbf{z}) = z_i^{d_i} - g_i'(\mathbf{z})$ where g_i' is a polynomial in $\mathbb{Z}[\mathbf{z}]$ having total degree $d_i - 1$, integers coefficients in $[-2^8, 2^8]$ and 5 monomials. The *type* of such a system is the tuple $(d_1, \ldots, d_n)$. The classical systems come from [6]. The set of all the examples can be found at https://cims.nyu.edu/~imbach/IPY19/IPY19.txt.

The benchmark For each of the types $(4,4,4)$, $(5,5,5)$, $(4,3,3,3)$, $(4,4,3,3)$ we generated randomly a system as described above, and computed a triangular system with the `Maple` function `Triangularize` of the `RegularChains` package, using the option `'probability'=0.9`. For the classical systems, we used no option. In table 3, the time required to compute the regular chains is reported in column RC. We solved all systems with `Bertini AMP` (see group of columns `Bertini AMP` in table 3) and report the number of solutions with the multiplicity structure found by `Bertini` ($c_1 + c_2 \times m_2 + c_3 \times m_3$ means c_1 solutions with sum of multiplicities 1, c_2 solutions with sum of multiplicities m_2 and c_3 solutions with sum of multiplicities m_3), the number of paths followed (column #Paths) and the solving time. We solved all systems with the function `RootFinding[Isolate]` of `Maple` with the options `digits=15, output=interval, method='RC'` (*i.e.* using regular chains) and reported the number of real solutions of the system and the solving time (see group of columns Isolate RC in table 3). For all systems, we used `tcluster` with $\epsilon = 2^{-53}$ and an initial polybox centered in $\mathbf{0}$ with width 10^6. For all the systems we tested, `tcluster` is faster than the numerical solver of `RootFinding[Isolate]` (when using RC) that finds only real solutions. We also tested `HOM4PS-2.0` for these systems: the number of paths followed for each system is about the number of solutions of the system, and the running time is always less than 0.05s, However the number of solutions reported is wrong for systems $(5,5,5)$ (97 instead of 125) and *5-body-homog* (93 instead of 99).

Random systems in table 3. For these systems, the number of solutions is equal to the Bézout's bound; therefore `Bertini AMP` follows one path per solution. Homotopy solving in these cases is much more efficient than triangularizing the system with RC and solving it with `tcluster`. For these systems, the RC algorithm produces a triangular system of type $(d, 1, \ldots, 1)$ where d is the Bézout's bound, with a huge bitsize. For the system of type $(3,3,4,4)$, the triangular system has type $(144,1,1,1)$ and each equation has bitsize about 738. In that case, `tcluster` has to isolate some solutions of the first equation in disks of radius less than 2^{-424}. Solving the first equation with `ccluster` and $\epsilon = 2^{-424}$ takes 26.13s: `tcluster` spends most of the time in isolating roots of the first polynomial. Any

	Bertini AMP			Isolate RC		RC	tcluster global	
type/name	#Sols	#Paths	t (s)	#Sols	t (s)	t (s)	#Sols	t (s)
Random systems								
(4,4,4)	64	64	0.06	6	7.53	3.82	64	1.48
(5,5,5)	125	125	0.30	?	>1000	24.2	125	18.7
(3,3,3,4)	108	108	0.13	?	>1000	52.4	108	7.25
(3,3,4,4)	144	144	0.26	?	>1000	68.7	144	25.7
Classical systems with only simple solutions								
Arnborg-Lazard	20	120	0.80	8	3.09	0.08	20	0.22
Czapor-Geddes-Wang	24	720	28.6	2	1.87	0.17	24	0.59
cyclic-5	70	120	0.35	10	1.92	0.55	70	0.81
Classical systems with multiple solutions								
5-body-homog	$45+2\times3+2\times24$	224	7.63	11	8.30	0.16	49	0.80
Caprasse	$24+8\times4$	144	0.25	18	1.49	0.24	32	0.27
neural-network	$90+18\times2$	162	0.36	22	5.82	0.13	108	0.89

Table 3. Solving systems of polynomial equations with regular chains and `tcluster`, and `Bertini AMP`. `tcluster` is called with $\epsilon = 53$ and an initial polybox centered in $\mathbf{0}$ with width 10^6.

improvement of `ccluster` will directly benefit to `tcluster`. For three of these systems, `RootFinding[Isolate]` has been stopped after 1000s.

Classical systems with only simple solutions in table 3. These systems have few finite solutions compared to their Bézout's bounds, and `Bertini AMP` wastes time in following paths going to infinity. In contrast, `tcluster` is sensitive to the number of solutions in the initial solving domain. This explains why computing triangular systems and solving it with `tcluster` is faster than `Bertini AMP` for systems *Arnborg-Lazard*, *Czapor-Geddes-Wang*.

Classical systems with multiple solutions in table 3. For these systems, `Bertini AMP` reports the multiplicity structure of the solutions. The triangularization step removes the multiplicity, and the RCs obtained are easier to solve; `tcluster` finds only clusters with one solution counted with multiplicity.

References

1. Aubry, P., Lazard, D., Maza, M.M.: On the theories of triangular sets. Journal of Symbolic Computation **28**(1), 105 – 124 (1999). https://doi.org/10.1006/jsco.1999.0269
2. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu. https://doi.org/10.7274/R0H41PB5
3. Becker, R., Sagraloff, M., Sharma, V., Xu, J., Yap, C.: Complexity analysis of root clustering for a complex polynomial. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. pp. 71–78. ISSAC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2930889.2930939

4. Becker, R., Sagraloff, M., Sharma, V., Yap, C.: A near-optimal subdivision algorithm for complex root isolation based on the pellet test and newton iteration. Journal of Symbolic Computation (2017). https://doi.org/10.1016/j.jsc.2017.03.009
5. Beltrán, C., Leykin, A.: Certified numerical homotopy tracking. Experimental Mathematics **21**(1), 69–83 (2012). https://doi.org/10.1080/10586458.2011.606184
6. Boulier, F., Chen, C., Lemaire, F., Moreno Maza, M.: Real root isolation of regular chains. In: Feng, R., Lee, W.s., Sato, Y. (eds.) Computer Mathematics. pp. 33–48. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43799-5˙4
7. Cheng, J.S., Gao, X.S., Yap, C.K.: Complete numerical isolation of real roots in zero-dimensional triangular systems. Journal of Symbolic Computation **44**(7), 768–785 (2009). https://doi.org/10.1016/j.jsc.2008.04.017
8. Dahan, X., Maza, M.M., Schost, E., Wu, W., Xie, Y.: Lifting techniques for triangular decompositions. In: Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation. pp. 108–115. ISSAC '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1073884.1073901
9. Dayton, B.H., Zeng, Z.: Computing the multiplicity structure in solving polynomial systems. In: Proceedings of the 2005 international symposium on Symbolic and algebraic computation. pp. 116–123. ACM (2005). https://doi.org/10.1145/1073884.1073902
10. Dickenstein, A., Emiris, I. (eds.): Solving Polynomial Equations: Foundations, Algorithms, and Applications. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/B138957
11. Giusti, M., Lecerf, G., Salvy, B., Yakoubsohn, J.C.: On location and approximation of clusters of zeros: Case of embedding dimension one. Foundations of Computational Mathematics **7**(1), 1–58 (2007). https://doi.org/10.1007/s10208-004-0159-5
12. Huber, B., Sturmfels, B.: A polyhedral method for solving sparse polynomial systems. Mathematics of computation **64**(212), 1541–1555 (1995). https://doi.org/10.1090/S0025-5718-1995-1297471-4
13. Imbach, R., Pan, V.Y., Yap, C.: Implementation of a near-optimal complex root clustering algorithm. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) Mathematical Software – ICMS 2018. pp. 235–244. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96418-8˙28
14. Johansson, F.: Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. IEEE Transactions on Computers **66**, 1281–1292 (2017). https://doi.org/10.1109/TC.2017.2690633
15. Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials ... and now for real! In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. pp. 303–310. ISSAC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2930889.2930937
16. Maza, M.M.: On triangular decompositions of algebraic varieties. Tech. rep., Citeseer (2000)
17. Mignotte, M.: On the distance between the roots of a polynomial. Applicable Algebra in Engineering, Communication and Computing **6**(6), 327–332 (Nov 1995). https://doi.org/10.1007/BF01198012
18. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval analysis. Siam (2009)
19. Wampler, I.C.W., et al.: The Numerical solution of systems of polynomials arising in engineering and science. World Scientific (2005)

20. Xu, J., Burr, M., Yap, C.: An approach for certifying homotopy continuation paths: Univariate case. In: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation. pp. 399–406. ISSAC '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3208976.3209010
21. Zhang, Z., Fang, T., Xia, B.: Real solution isolation with multiplicity of zero-dimensional triangular systems. Science China Information Sciences **54**(1), 60–69 (2011). https://doi.org/10.1007/s11432-010-4154-y