Validated Numerics for Algebraic Path Tracking

Alexandre Guillemot, Pierre Lairez

Validated Numerics for Algebraic Path Tracking

Alexandre Guillemot
Inria, Université Paris-Saclay, Palaiseau, France

Pierre Lairez
Inria, Université Paris-Saclay, Palaiseau, France

ABSTRACT

Using validated numerical methods, interval arithmetic and Taylor models, we propose a certified predictor-corrector loop for tracking zeros of polynomial systems with a parameter. We provide a Rust implementation which shows tremendous improvement over existing software for certified path tracking.

CCS CONCEPTS

• **Mathematics of computing → Interval arithmetic**.

KEYWORDS

numerical algebraic geometry, certified path tracking

ACM Reference Format:
Alexandre Guillemot and Pierre Lairez. 2024. Validated Numerics for Algebraic Path Tracking. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '24), July 16–19, 2024, Raleigh, NC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3666000.3669673

1 INTRODUCTION

Path tracking, or homotopy continuation, is the backbone of numerical algebraic geometry [29]. Given a polynomial map $(t, x) \mapsto F_t(x)$, from $\mathbb{C} \times \mathbb{C}^n$ to $\mathbb{C}^n$, and a regular zero $\zeta \in \mathbb{C}^n$ of F_0, we want to compute for $t \in [0, 1]$ the continuation ζ_t, which is, assuming well-posedness, the unique continuous function of t such that $F_t(\zeta_t) = 0$ and $\zeta_0 = \zeta$. Heuristic approaches to path tracking are sometimes enough for solving polynomial systems, since we may have the possitiblity to certify zeros of the target system F_1 *a posteriori*. However, for computing monodromy actions – which applies to irreducible decomposition [28] or Galois group computation [12] – and finer invariants – such as braids [26] – it is not sound to use heuristic continuation methods.

Contributions. Building upon Moore's interval arithmetic criterion to isolate a zero of a polynomial system [16, 21, 27], and following ideas from van der Hoeven [31], also developed independently by Duff and Lee [9], we propose a new path tracking algorithm. The originality of this result lies in the model of computation which rather than idealizing interval arithmetic accounts for what a real software library like MPFI provides. There is no discrepancy between what is presented and what is implemented. We prove correction and termination in this model. In particular, we provide an algorithm (Section 4) to refine isolating boxes provided by Moore's criterion. We use this algorithm to formulate the path tracking algorithm (Section 5). The main difficulty is the balance of the working precision, which should be as low as possible for performance, but large enough to ensure appropriate convergence properties and termination.

The use of Taylor models enables predictors (Section 6), such as the tangent predictor, or the cubic Hermite predictor, as observed by van der Hoeven [31], leading to tremendous improvement over previous methods based on Smale's α-theory. We provide a Rust implementation that we compare with existing software for path tracking, both certified and noncertified (Section 7). We find that the number of iterations performed using the Hermite predictor is in the same order of magnitude as noncertified approaches.

Related work. The method of path tracking received vast attention in the context of numerical multivariate polynomial system solving. It is the method of choice for most state-of-the-art software: PHCpack [32], Bertini [2] or HomotopyContinuation.jl [6], for example. Despite their effectiveness (or perhaps *because* of their effectiveness), and despite recent work which brings robustness to unprecedented levels [30], this software does not guarantee correctness, that is the consistency of the result with the definition of the continuation. We obtain zeros of the target system which we may certify independently, but we are not certain that the paths were tracked correctly, without swapping for example.

There are countless certified path tracking algorithms based on Shub and Smale's α-theory. For the most part, they were developed for complexity analyses (for a review, see Cucker [8]) and their implementation is difficult. Beltrán and Leykin [3, 4] took on the challenge within the Macaulay2 package for numerical algebraic geometry [19]. In a specific case of "Newton homotopies" (where the system has the form $F_t(x) = F_1(x) - (1 - t)F_1(v)$ for some constant $v \in \mathbb{C}^n$), Hauenstein et al. [10], Hauenstein and Liddell [11] managed to incorporate a tangent predictor into the α-theory and obtained significant improvement. In the univariate case ($n = 1$), methods are much more diverse, even though path tracking is not the method of choice for solving in this case. We note in particular working implementations by Kranich [15], Marco-Buzunariz and Rodríguez [20], Xu et al. [33].

2 MOORE'S CRITERION

Let V be a finite dimensional linear space over $\mathbb{R}$ with a norm $\|-\|$. We denote by $|||-|||$ the associated operator norm on $\mathrm{End}(V)$. Let B denote the closed unit ball. (We will later choose $V = \mathbb{C}^n \simeq \mathbb{R}^{2n}$ and $\|-\|$ will be the real ∞-norm, so B will be a box.)

Theorem 2.1. *Let $f : V \to V$ be a continuously differentiable map and let $\rho \in (0, 1)$. Let $x \in V$, $r > 0$, and let $A : V \to V$ be a linear map. Assume that for any $u, v \in rB$,*

$$-Af(x) + \big[\,\mathrm{id}_V - A \circ \mathrm{d}f(x + u)\big](v) \in \rho r B.$$

Then there is a unique $\zeta \in x + rB$ such that $f(\zeta) = 0$. Moreover,

(i) $\|x - \zeta\| \leq \rho r$;

and for any $y \in x + rB$,

ISSAC '24, July 16–19, 2024, Raleigh, NC, USA

ACM ISBN 979-8-4007-0696-7/24/07
https://doi.org/10.1145/3666000.3669673

(ii) A *and* $\mathrm{d}f(y)$ *are invertible;*
(iii) $|||\mathrm{d}f(y)^{-1}A^{-1}||| \leq (1-\rho)^{-1}$*;*
(iv) $|||A||| \leq (1+\rho)|||\mathrm{d}f(y)^{-1}|||$*;*
(v) $(1-\rho)\|y-\zeta\| \leq \|Af(y)\| \leq (1+\rho)\|y-\zeta\|$.

The operator mapping a compact convex set E containing 0 in its interior to $K(E) = -Af(x) + [\mathrm{id}_V - A \circ \mathrm{d}f(x+E)]\,(E)$, has been introduced by Krawczyk [16] to refine isolating boxes. Moore [21] showed that the inclusion $K(E) \subseteq E$ implies the existence of a zero of f in $x+E$, by a reduction to Brouwer's fixed-point theorem. Then Rump [27, Theorem 7.4] proved unicity of the root if $K(E) \subseteq \mathring{E}$ (which implies $K(E) \subseteq \rho E$ for some $\rho \in (0,1)$ by compactness of E).[1] In the variant above, we assume that $K(E) \subseteq \rho E$ for some $\rho \in (0,1)$, and in addition we assume that E is a centered ball with respect to some norm (which is always the case if $E = -E$, in addition to E being compact, convex and a neighborhood of 0). In exchange for this extra assumption we can prove the theorem with Banach's fixed-point theorem, which is more elementary than Brouwer's, work with the operator norm induced by the norm instead of the spectral radius, and obtain the quantitative statements (i)–(v) which will prove useful later.

Proof. Consider the function $g(y) = y - A \circ f(y)$. The assumption on f rewrites as

$$\| \pm Af(x) + \mathrm{d}g(y)(v)\| \leq \rho r, \quad \forall y \in x + rB, \forall v \in rB, \qquad (1)$$

where the $\pm$ sign comes from changing v into $-v$, using $-B = B$. The triangle inequality then implies

$$2\|\mathrm{d}g(y)(v)\| \leq \|\mathrm{d}g(y)(v) - Af(x)\| + \|\mathrm{d}g(y)(v) + Af(x)\|,$$

which shows that $|||\mathrm{d}g(y)||| \leq \rho$, so g is ρ-Lipschitz continuous. Since $\rho < 1$, g is a contracting map. Moreover, it is well known [18, Thm. 18.2.1] that $|||\mathrm{d}g(y)||| \leq \rho < 1$ implies the invertibility of the operator $\mathrm{id} - \mathrm{d}g(y)$, that is $A \circ \mathrm{d}f(y)$, which implies (ii). The bounds (iii) and (iv) are also easy consequences.

Let $u \in E$ and let $u_t = x + tu$, for $t \in [0,1]$. By integrating the derivative, we compute

$$g(x+u) = x + \int_0^1 \big(- Af(x) + \mathrm{d}g(u_t)(u)\big)\mathrm{d}t,$$

which shows, using (1), that $g(x+rB) \subseteq x + \rho rB \subset x + rB$. By Banach's theorem, g has a unique fixed point ζ in $x + rB$. Since A is invertible, ζ is a zero of f. Inequality (i) follows from ζ belonging to $g(x+rB) \subseteq x + \rho rB$ and inequalities (v) follow from the ρ-Lipschitz continuity of g. □

3 DATA STRUCTURES

3.1 Arithmetic circuits

We represent polynomial functions $\mathbb{C}^n \to \mathbb{C}^m$ as *arithmetic circuits*, also known as *straight-line programs*. Briefly, an arithmetic circuit with input space $\mathbb{C}^n$, is a directed acyclic graph, multiple edges allowed, with four types of nodes: (1) input nodes, with no incoming egdes and labelled with an integer in $\{1,\dots,n\}$; (2) constant nodes, with no incoming edges and labelled with an element of $\mathbb{C}$; (3) addition nodes +, with exactly two incoming edges; (4) multiplication nodes ×, with exactly two incoming edges. We associate in the obvious way to each node v of a circuit a polynomial function $P_v : \mathbb{C}^n \to \mathbb{C}$ [see 7, for more details]. To a tuple of m nodes of a circuit, we associate a polynomial function $\mathbb{C}^n \to \mathbb{C}^m$.

This data structure is useful in that it represents not only a polynomial but also a scheme for evaluating it, over $\mathbb{C}$ or more general objects, for example interval numbers. Moreover we may use automatic differentiation (in forward or backward mode) to transform a circuit to another which also computes the derivative of some nodes with respect to some of the input variables.

3.2 Intervals

Checking Moore's criterion requires more than a point evaluation, but information on the image of a set by polynomial map. Interval arithmetic provides an effective approach to this issue. There are many ways to model and implement interval arithmetic. In short, we choose a set $\mathbb{F} \subset \mathbb{R}$ of representable numbers, and we define $\square\mathbb{R}$ (read "box $\mathbb{R}$" or "interval $\mathbb{R}$") to be the set of all nonempty compact intervals of $\mathbb{R}$ with end points in $\mathbb{F}$. Lastly, we assume effective binary operations $\boxplus$ and $\boxtimes$ on $\square\mathbb{R}$ such that for any $I, J \in \square\mathbb{R}$ and any $x \in I$ and $y \in J$, $x + y \in I \boxplus J$ and $xy \in I \boxtimes J$. For example, we can choose $\mathbb{F} = \mathbb{Q}$ and define

$$[a,b] \boxplus [c,d] = [a+c, b+d], \text{ and}$$
$$[a,b] \boxtimes [c,d] = [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)].$$

These are the usual formula for interval arithmetic, but they are seldom used in this exact form because of the unbearable swell of the binary size of the interval endpoints they produce in any nontrivial computation. In practice, $\mathbb{F}$ is often the set of finite IEEE-754 64-bits floating-points numbers and the formula above are implemented with appropriate rounding of the interval endpoints. (And we usually extend $\square\mathbb{R}$ with unbounded intervals to cope with overflows.) We may also choose $\mathbb{F} = \{a2^b \mid a, b \in \mathbb{Z}\}$, the set of dyadic numbers and round the endpoints of the interval at a given relative precision, which may change in the course of the computation. This models, for example, the behavior of multiple precision libraries such as MPFI [25] or Arb [13].

We define $\square\mathbb{C}$ as pairs of elements of $\square\mathbb{R}$, representing real and imaginary parts, endowed with the obvious extensions of $\boxplus$ and $\boxtimes$. We consider also vectors of boxes, denoted $\square\mathbb{C}^n$.

3.3 Interval extensions, adaptive precision

Let f be a function $\mathbb{C}^n \to \mathbb{C}^m$. An *interval extension* of f is a function $\square f : \square\mathbb{C}^n \to \square\mathbb{C}^m$ such that for any $X \in \square\mathbb{C}^n$ and any $x \in X$, $f(x) \in \square f(X)$. If f is a polynomial function represented by a circuit with constants in $\mathbb{F}$, then we obtain naturally an extension of f by replacing any constant c in the circuit by the singleton interval $[c,c]$ and evaluating the circuit using interval operations $\boxplus$ and $\boxtimes$.

The correctness of our algorithm does not depend on any hypothesis on interval arithmetic and extensions other than the basic requirements on interval extensions. As for termination, we need stronger hypotheses. As pointed out above, interval arithmetic is usually not implemented in exact arithmetic. It is also clear that finitely many representable numbers, as provided by the IEEE-754

[1] In fact Rump merely requires the strict inclusion $K(E) \subset E$ but this is not enough to obtain unicity, as shown by a simple linear projection $f(x,y) = (x,0)$. Alefeld and Mayer [1, Theorem 11] provide a correct version with a slightly weaker premise than $K(E) \subseteq \mathring{E}$ but stronger than $K(E) \subset E$.

arithmetic model, will not be enough to express termination arguments based on topology and convergence. So we introduce an adaptive precision model which can be implemented using any multiple precision interval library. The operations in this model depend on a parameter $u_{\text{prec}} \in (0,1)$, the *unit roundoff*, which can be raised or lowered at will. We require that for any $M \geq 1$ and any $[a,b], [c,d] \in \square\mathbb{R}$ included in $[-M,M]$,

$$[a,b] \boxplus [c,d] \subseteq [a+c-Mu_{\text{prec}}, b+d+Mu_{\text{prec}}], \text{ and} \quad (2)$$

$$[a,b] \boxtimes [c,d] \subseteq [\min(ac,ad,bc,bd) - M^2 u_{\text{prec}}, \max(ac,ad,bc,bd) + M^2 u_{\text{prec}}]. \quad (3)$$

Interval arithmetic implemented with IEEE-754 arithmetic satisfies these contraints, with $u_{\text{prec}} \sim 2^{-53}$, unless overflow occurs [23].

If we have a circuit $f : \mathbb{C}^n \to \mathbb{C}^m$, its interval extension $\square f$ depends on the working precision u_{prec}. To formulate a useful property, we need some metrics. Let $\|-\|$ be the real ∞-norm on $\mathbb{C}^n$ seen as $\mathbb{R}^{2n}$, that is

$$\|(z_1,\ldots,z_n)\| = \max_{1\leq i\leq n} \max\left(|\operatorname{Re}(z_i)|, |\operatorname{Im}(z_i)|\right). \quad (4)$$

For $X, Y \in \square\mathbb{C}^n$, we define the *width* (or *diameter*) and the *magnitude* $\|X\|_\square$ as

$$\operatorname{width}(X) = \sup_{x,y\in X} \|x-y\|, \text{ and } \|X\|_\square = \sup_{x\in X} \|x\|,$$

and we define the Hausdorff distance by

$$\operatorname{dist}(X,Y) = \max\left\{ \sup_{x\in X}\inf_{y\in Y}\|x-y\|, \sup_{y\in Y}\inf_{x\in X}\|x-y\| \right\}.$$

Lastly, $\operatorname{mid}(X)$ denotes the *midpoint* of X, which is some representable element of X (and we usually choose one that is as close as possible to the mathematical center).

Proposition 3.1. *Let $f : \mathbb{C}^n \to \mathbb{C}^m$ be a circuit with representable constants and let $\square f$ be its natural interval extension in the adaptive precision model. For any compact $K \subseteq \mathbb{C}^n$, there is some $L \geq 0$, independent of the unit roundoff, such that for any $X, Y \in \square\mathbb{C}^n$ included in K:*

(i) $\operatorname{width}(\square f(X)) \leq L\,(\operatorname{width}(X) + u_{\text{prec}})$,
(ii) $\operatorname{dist}(\square f(X), \square f(Y)) \leq L\,(\operatorname{dist}(X,Y) + u_{\text{prec}})$.

Proof. These two properties are stable under composition, so it is enough to check them for $\boxplus$ and $\boxtimes$, and this follows directly from (2) and (3). □

3.4 Moore boxes

Let $B \subseteq \mathbb{C}^n$ be the closed unit ball for the real ∞ norm $\|-\|$. Given a polynomial map $f : \mathbb{C}^n \to \mathbb{C}^n$ and some $\rho \in (0,1)$, a *ρ-Moore box for f* is a triple $(x,r,A) \in \mathbb{C}^n \times \mathbb{R}_> \times \mathbb{C}^{n\times n}$ such that

$$\left\| -r^{-1}Af(x) + (I_n - A\cdot \mathrm{d}f(x+rB))\cdot B \right\| \leq \rho.$$

We say that ρ is the *contraction factor*. By Theorem 2.1, if (x,r,A) is a Moore box, then there is a unique zero of f in the set $x+rB$, called the *associated zero*. We use Moore boxes (with representable x, r and A) as a data structure to represent a regular zero of f.

Given interval extensions $\square f$ and $\square \mathrm{d}f$, if it holds in interval arithmetic that

$$\left\| -r^{-1}A\cdot \square f(x) + (I_n - A\cdot \square \mathrm{d}f(x+rB))\cdot B \right\|_\square \leq \rho,$$

then (x,A,r) is a ρ-Moore box for f. Note that the magnitude is always a representable number, so we can indeed check this inequality accurately. This leads to Algorithm 1.

Algorithm 1 Interval certification of a Moore box

input $\square f : \square\mathbb{C}^n \to \square\mathbb{C}^n$; $\square \mathrm{d}f : \square\mathbb{C}^n \to \square\mathbb{C}^{n\times n}$;
$x \in \mathbb{C}^n$; $r \in \mathbb{R}_>$; $A \in \mathbb{C}^{n\times n}$; $\rho \in (0,1)$
output a boolean

```
1 def M(□f, □df, x, r, A, ρ):
2     K ← −r^{-1} A · □f(x) + (I_n − A · □df(x + rB)) · B
3     return ‖K‖_□ ≤ ρ
```

Lemma 3.2 (Correctness of Algorithm 1). *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a polynomial function, and let $\square f : \square\mathbb{C}^n \to \square\mathbb{C}^n$ and $\square \mathrm{d}f : \square\mathbb{C}^n \to \square\mathbb{C}^{n\times n}$ be interval extensions of f and $\mathrm{d}f$ respectively. For any $x \in \mathbb{C}^n, A \in \mathbb{C}^{n\times n}, r > 0$ and $\rho \in (0,1)$, if $M(\square f, \square \mathrm{d}f, x, r, A, \rho)$ returns* True, *then (x,r,A) is a ρ-Moore box for f.*

4 REFINEMENT OF MOORE BOXES

Following the original idea of Krawczyk [16], who introduced his operator to refine isolating boxes, we can refine a Moore box (x,r,A) for a polynomial map $f : \mathbb{C}^n \to \mathbb{C}^n$ by computing the intersection

$$(x+rB) \cap (x - A\cdot \square f(x) + (I_n - A\cdot \square \mathrm{d}f(x+rB))).$$

Unfortunately, this cannot always work, because the interval arithmetic may be too gross. It is possible that (x,r,A) is a Moore box, but the intersection above is just $x+rB$, providing no useful refinement. In other words, it is possible that (x,r,A) is a ρ-Moore box but $M(\square f, \square \mathrm{d}f, x, r, A, \rho)$ returns *False*.

4.1 Algorithm

We are given (a circuit representing) a polynomial map $f : \mathbb{C}^n \to \mathbb{C}^n$, a ρ-Moore box (x,r,A) with an associated zero ζ of f, and some $\tau \in (0,1)$. We want to compute another Moore box with same associated zero and contraction factor τ.

Algorithm 2 proceeds as follows. The main loop (Line 3) maintains a triple (y,s,U) and stops when the interval arithmetic criterion certifies that (y,s,U) is a τ-Moore box. After the first iteration, the matrix U is always $\mathrm{d}f(y)^{-1}$, give or take some roundoff errors. The computation of U need not be performed in interval arithmetic (this is the practical appeal of Krawczyk's operator), the correctness of the algorithm does not depend on the accuracy of this computation, but to ensure termination, the distance from U to $\mathrm{d}f(y)^{-1}$ must go to zero as the working precision increases (that is $u_{\text{prec}} \to 0$). After the main loop, we know that (y,s,U) is a τ-Moore box. Before returning it, we check if, by chance, $(y,2s,U)$ is also a τ-Moore box. We double the radius until it is not the case (or the radius exceeds 1) and then return the Moore box.

When Moore's criterion cannot be checked, we try to improve the triple (y,s,U), either by using a quasi-Newton iteration $y \to y - A\cdot f(y)$, or by shrinking the box with $s \to \frac{1}{2}s$. The choice depends on $\|A\cdot f(y)\|$, the size of the quasi-Newton step, compared to τs, with the goal of balancing the two terms in Moore's criterion.

We want to run the computation with standard double precision as much as possible, but the algorithm may warn that the working

precision is not large enough, or equivalently, u_{prec} is not small enough (Lines 8 and 10). The computational model assumes that we can increase the working precision, but on the practical side, if only double precision is available (this is the case in the implementation we propose), we abort the computation on a precision warning. In this view, it is important to avoid undue warnings.

In Algorithm 2, two checks may trigger a precision warning. First, after shrinking the box, when $\frac{s}{\tau r}$ drops below some threshold. This is because we expect τr to be the radius of the τ-Moore box that we are looking for (simply by considering a degree 2 approximation of f around x). So when $\frac{s}{\tau r}$ is too small we may suspect that Moore's criterion failed because of roundoff errors. In this case, we want to increase working precision in such a way that u_{prec} goes to 0 faster than s. Second, when performing a quasi-Newton iteration, we check that the roundoff error, that is $\operatorname{width}(y-\delta)$, is significantly smaller than the size of the quasi-Newton step. This ensures the convergence of y (Lemma 4.4).

Remark 4.1. Algorithm 2 features some arbitrary constants for which we picked explicit values. Let us name them: $\rho = \frac{7}{8}$, the contraction factor of the input; $\alpha = \frac{1}{64}$, used to compare $\|\delta\|_\square$; $\lambda = \frac{1}{2}$, used to shrink s; $\beta = \frac{1}{40}$, used in the precision check.

Naturally, there is some flexibility in the choice of these constants. To keep the algorithm correct, we need to have $\lambda < 1$ obviously. Values closer to 1 will produce bigger boxes with more iterations. The smaller α, the closer y is to the exact root. The quasi-Newton iteration converges rapidly, so there is little performance penalty in lowering α, but it may cause the precision check to fail earlier. For correctness, we need $\alpha + \rho < 1$. (cf. proofs of Lemmas 4.5 and 5.2). The parameter ρ can be close to 1, it improves performance to allow for Moore boxes with a larger contraction factor. We do not have to worry about the speed of convergence of the quasi-Newton iteration, because interval arithmetic is typically pessimistic. So if we can check a Moore box with contraction factor $\frac{7}{8}$, then it probably has a much lower actual contraction factor. However, other constants degrade if ρ is too close to 1. For correctness, we need $\frac{\rho+\beta}{1-\beta} < 1$. This number is the geometric ratio in Lemma 4.4.

Remark 4.2. Checking Moore's criterion is costly. From the practical point of view, with the path tracking algorithm in mind, it is beneficial for performance to perform first one or two quasi-Newton iterations (with the appropriate precision check) before entering the main loop.

4.2 Analysis

Theorem 4.3. *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a circuit, let (x, r, A) be a $\frac{7}{8}$-Moore box for f and let $\tau \in (0,1)$. On input x, r, A and τ, Algorithm 2 terminates and outputs a τ-Moore box for f with same associated zero as the input box.*

The algorithm does *something* until $M(\square f, \square \mathrm{d}f, y, r, U, \tau)$ holds. Then a second loop does *something* that preserves this property. So it is obvious, by design, that the algorithm outputs a τ-Moore box. It remains to check that the associated zeros of the input and output are the same, and that the algorithm terminates.

Algorithm 2 Refinement of a Moore box

input $f : \mathbb{C}^n \to \mathbb{C}^n$; (x, r, A), a $\frac{7}{8}$-Moore box; $\tau \in (0,1)$
output a τ-Moore box with same associated root as (x, r, A)

```
1  def refine(f, x, r, A, τ):
2      y ← x;   s ← r;   U ← A;
3      while not M(□f, □df, y, s, U, τ):
4          δ ← A · □f(y)
5          if ‖δ‖□ ≤ (1/64)τs:
6              s ← (1/2)s
7              if s < (1/16)τr:
8                  increase precision enough so that u_prec = o(s)
9          elif width(y − δ) > (1/40)‖δ‖□:    # precision check
10             increase working precision
11         else:
12             y ← mid(y − δ)
13         U ← mid(□df(y))^{-1}      # unchecked arithmetic
14     while 2s ≤ 1 and M(□f, □df, x, 2s, U, τ):
15         s ← 2s
16     return y, s, U
```

We first study the sequence $y_0 = x, y_1, \dots$ where y_k is the value of y after the kth quasi-Newton iteration. In principle, the quasi-Newton iteration converges, but roundoff errors could take over. The precision check (Line 9) ensures that it does not happen.

Lemma 4.4. *For any $k \geq 0$, $\|y_k - x\| \leq r$ and $\|y_k - \zeta\| \leq \frac{7}{8}\left(\frac{12}{13}\right)^k r$.*

Proof. We prove the claim by induction. First, $y_0 = x$, so we have $\|y_0 - x\| = 0$ and $\|y_0 - \zeta\| \leq \frac{7}{8}r$, by Theorem 2.1(i). Then, by definition, $y_{k+1} = \operatorname{mid}(y_k - \delta_k)$, where $\delta_k = A \cdot \square f(y_k)$ computed in interval arithmetic. So there is some $\varepsilon_k \in \mathbb{C}^n$ with $\|\varepsilon_k\| \leq \operatorname{width}(y_k - \delta_k)$ such that

$$y_{k+1} = y_k - Af(y_k) + \varepsilon_k = g(y_k) + \varepsilon_k. \tag{5}$$

Recall that $g : y \mapsto y - Af(y)$ is a $\frac{7}{8}$-Lipschitz continuous function on $x + rB$ such that $g(\zeta) = \zeta$ and $g(x + rB) \subseteq x + \frac{7}{8}B$. In particular,

$$\|y_{k+1} - \zeta\| \leq \tfrac{7}{8}\|y_k - \zeta\| + \|\varepsilon_k\|, \quad \text{and} \tag{6}$$

$$\|y_{k+1} - x\| \leq \tfrac{7}{8}r + \|\varepsilon_k\|. \tag{7}$$

The precision check (Line 9) implies that

$$\|\varepsilon_k\| \leq \operatorname{width}(y_k - \delta_k) \leq \tfrac{1}{40}\|\delta_k\|_\square. \tag{8}$$

Moreover $\operatorname{width}(\delta_k) \leq \operatorname{width}(y_k - \delta_k)$ so,

$$\|\delta_k\|_\square \leq \|Af(y_k)\| + \operatorname{width}(\delta_k) \leq \|Af(y_k)\| + \tfrac{1}{40}\|\delta_k\|_\square,$$

and by Theorem 2.1(v) it follows

$$\|\delta_k\|_\square \leq \tfrac{40}{39}\|Af(y_k)\| \leq \tfrac{25}{13}\|y_k - \zeta\|. \tag{9}$$

In combination with (8), we obtain

$$\|\varepsilon_k\| \leq \tfrac{5}{104}\|y_k - \zeta\|. \tag{10}$$

It follows easily from (6) and (10) that

$$\|y_{k+1} - \zeta\| \leq \tfrac{12}{13}\|y_k - \zeta\|. \tag{11}$$

From the induction hypothesis, we have $\|y_k - \zeta\| \leq \frac{7}{8}r$ and it follows from (7) and (10) that $\|y_{k+1} - x\| \leq r$, which proves the induction step. □

Lemma 4.5. *At any point of the algorithm, $\|y - \zeta\| \leq s$.*

Proof. The inequality obviously holds at the start of the algorithm. It remains to check that it still holds when y is moved by a quasi-Newton iteration or when s is halved. The norm $\|y - \zeta\|$ decreases when a quasi-Newton iteration is performed, by (11). So the inequality is preserved in this case. In the second case, we note that Line 6 is only reached when $\|\delta\|_\square \leq \frac{1}{64}\tau s$. Since $\|Af(y)\| \leq \|\delta\|_\square$, this implies, together with Theorem 2.1(v), that $\|y - \zeta\| \leq 8\|Af(y)\| \leq \frac{1}{8}s$. So after $s \leftarrow \frac{s}{2}$, we have $\|y - \zeta\| \leq \frac{1}{4}s$, and the inequality holds. □

Since $\zeta \in y + sB$, it is clear that the Moore box output by the algorithm is associated to ζ, by unicity of the associated root. It only remains to settle termination.

Lemma 4.6. *Considering the values of y, s, U and u_{prec} in an infinite run of Algorithm 2, we have: (i) $s \to 0$; (ii) $u_{\text{prec}} = o(s)$; (iii) $y \to \zeta$; (iv) $U \to \mathrm{d}f(\zeta)^{-1}$; (v) $\|Uf(y)\| \leq \frac{1}{2}\tau s$ eventually.*

Proof. First, we show that $s \to 0$. Assume it does not, implying that after a certain point, s is never halved, so we always have

$$\|\delta\|_\square > \tfrac{1}{64}\tau s. \tag{12}$$

Since s is not halved, every iteration of the main loop fails the "if" condition, so it reaches reaches the "elif" condition: the precision check. It may fail it (and the working precision is raised) or pass it (and a quasi-Newton iteration is performed). The precision check cannot fail for ever. Indeed, every fail causes the working precision to increase so $\mathrm{width}(y - \delta) \to 0$, by Proposition 3.1(i). By (12), $\|\delta\|_\square$ is bounded below, so we have $\mathrm{width}(y - \delta) \leq \frac{1}{40}\|\delta\|_\square$ and the precision check passes. This implies that infinitely many quasi-Newton iterations are perfomed. By (9), this implies that $\|\delta\|_\square \to 0$, in contradiction with (12). Therefore, $s \to 0$, proving (i). When $s \to 0$, Line 8 ensures that $u_{\text{prec}} = o(s)$. This checks (ii).

The radius s is only halved when $\|\delta\|_\square \leq \frac{1}{64}\tau s$. Since s is repeatedly halved, this implies that $\|\delta\|_\square \to 0$, Therefore $Af(y) \to 0$, and, by Theorem 2.1(v), $y \to \zeta$, proving (iii). Since $u_{\text{prec}} \to 0$ and $U = \mathrm{d}f(y)^{-1}$, up to roundoff errors, we also have $U \to \mathrm{d}f(\zeta)^{-1}$, proving (iv). So it only remains to check (v). We decompose

$$Uf(y) = \underbrace{\left(U - \mathrm{d}f(y)^{-1}\right)}_{=O(u_{\text{prec}})} \cdot \underbrace{f(y)}_{\to 0} + \underbrace{\left(\mathrm{d}f(y)^{-1}A^{-1}\right)}_{\|\|-\|\|\leq 8} \cdot \underbrace{Af(y)}_{\|-\|\leq \frac{1}{64}\tau s}, \tag{13}$$

using Theorem 2.1(iii). This proves that $\|Uf(y)\| \leq \frac{1}{8}\tau s + o(s)$. So eventually $\|Uf(y)\| \leq \frac{1}{2}\tau s$. This concludes the proof. □

It is now easy to prove that Algorithm 2 terminates. Let $e = s^{-1}U \cdot \square f(y)$. Both U and y converges (Lemma 4.6(iii) and (iv)), in particular they are bounded. By Proposition 3.1, $\mathrm{width}(U \cdot \square f(y)) = O(u_{\text{prec}})$, and by Lemma 4.6(ii), this is $o(s)$. So after division by s, we have $\mathrm{width}(e) = o(1)$. Since $s^{-1}Uf(y) \in e$, we have

$$\|e\|_\square \leq s^{-1}\|Uf(y)\| + \mathrm{width}(e) \leq \tfrac{1}{2}\tau + o(1). \tag{14}$$

Moreover, $y + sB \to \{\zeta\}$ in the Hausdorff metric because $y \to \zeta$ and $s \to 0$ (Lemma 4.6(iii) and (i)). Since $u_{\text{prec}} \to 0$, Proposition 3.1 implies that

$$(I_n - U \cdot \square \mathrm{d}f(y + sB)) \cdot B \to \left(I_n - \mathrm{d}f(\zeta)^{-1} \cdot \mathrm{d}f(\zeta)\right) \cdot B = 0.$$

It follows that $\|-e + (I_n - U \cdot \square \mathrm{d}f(y + sB)) \cdot B\|_\square \leq \|e\|_\square + o(1)$. By (14), this is eventually less than τ, which means that Moore's criterion $M(\square f, \square \mathrm{d}f, y, s, U, \tau)$ holds, and the main loop terminates. Due to the condition $s \leq \frac{1}{2}$, the second loop, that tries growing s, also terminates.

5 PATH TRACKING

5.1 Setting

We are given an arithmetic circuit $F : \mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$. The first argument is the parameter and put in subscript, so that F_t denotes the map $\mathbb{C}^n \to \mathbb{C}^n$ obtained from F by specialization of the parameter. It also denotes the circuit obtained by replacing the input nodes #1 (the index of the parameter variable), with a constant node t.

Let $\zeta \in \mathbb{C}^n$ be a regular zero of F_0, that is $F_0(\zeta) = 0$ and assume that the $n \times n$ matrix $\mathrm{d}F_0(\zeta)$ is invertible. There is a unique open interval $I \subseteq \mathbb{R}$ containing 0 and a unique continuous function $Z : I \to \mathbb{C}^n$ such that:

(i) $Z_0 = \zeta$;
(ii) $F_t(Z_t) = 0$ for any $t \in I$;
(iii) if $b \in \overline{I} \setminus I$, then either
  (a) $\lim_{t \to b} \|Z_t\| = \infty$; or
  (b) $\lim_{t \to b} \det(\mathrm{d}F_t(Z_t)) = 0$.

This follows from the study of the differential equation

$$\tfrac{\mathrm{d}}{\mathrm{d}t}Z_t = -\mathrm{d}F_t(Z_t)^{-1} \cdot \dot{F}_t(Z_t) \tag{15}$$

obtained from the equation $F_t(Z_t) = 0$ by differentiation, and where $\dot{F}_t$ denote the partial derivative of F_t with respect to t. The existence and uniqueness of a local solution is given by the Picard-Lindelöf Theorem [18, Theorem 19.1.1]. There is a unique maximal solution interval, this is I, and the condition (iii) reflects the behavior of the solution at the boundary of the maximal interval of definition

Algorithm 3 Path tracking

input $F_\bullet$, a circuit $\mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$; (x, r, A), a $\frac{7}{8}$-Moore box for F_0
output a Moore box for F_1

```
1  def track(F_•, x, r, A):
2      t ← 0;   h ← 1
3      while t < 1:
4          x, r, A ← refine(F_t, x, r, A, 1/8)
5          h ← 2h;   T ← [t, t + h]
6          while not M(□F_T, □dF_T, x, r, A, 7/8):
7              h ← (1/2)h;   T ← [t, t + h]
8              if u_prec > h:
9                  increase working precision
10         t ← sup T
11     return x, B
```

[18, Theorem 19.2.4]: the solution diverges or leaves the domain of definition of the differential equation.

In what follows, we assume that ζ is given by a $\frac{7}{8}$-Moore box, and we aim at computing a Moore box for Z_1, as a zero of F_1, assuming that $1 \in I$.

5.2 Algorithm

Algorithm 3 performs the path tracking operation, as defined above, using the *refine* algorithm. The main ingredient is the use of Algorithm 1 with interval functions $\mathcal{F}$ and $\mathrm{d}\mathcal{F}$ that are exensions of F_t and $\mathrm{d}F_t$ respectively for a range of values of t.

More precisely, let $T \in \square\mathbb{R}$. In the circuit representing $F_\bullet$ and $\mathrm{d}F_\bullet$, replace the input nodes #1 (the index of the parameter variable) by constant nodes containing the interval value T. We obtain circuits that can be evaluated over $\square\mathbb{C}^n$. We denote $\square F_T$ and $\square\mathrm{d}F_T$ these circuits. The fundamental property of interval arithmetic guarantees that the interval functions defined by $\square F_T$ and $\square\mathrm{d}F_T$ are interval extensions of F_t and $\mathrm{d}F_t$ respectively, for any $t \in T$. In particular, if $M(\square F_T, \square\mathrm{d}F_T, x, r, A, \frac{7}{8})$ returns *True*, then (x, r, A) is a $\frac{7}{8}$-Moore box for F_t for any $t \in T$. This follows from Lemma 3.2, applied with $f = F_t$ and the interval extensions $\square f = \square F_T$ and $\square\mathrm{d}f = \square\mathrm{d}F_T$.

Based on this idea, given a $t \in [0, 1]$ and a $\frac{7}{8}$-Moore box for F_t, we refine it into a $\frac{1}{8}$-Moore box (x, r, A) then we search for a interval $T = [t, t + \delta]$ such that $M(\square F_T, \square\mathrm{d}F_T, x, r, A, \frac{7}{8})$ holds. If the search is successful, we know that (x, r, A) is a $\frac{7}{8}$-Moore box for $F_{t+\delta}$ and we can repeat the process.

The correctness of the algorithm is glaring, but termination is not. Can we find a positive δ at each step? Does the process eventually reach $t = 1$? Or may it converge to a lower value of t?

5.3 Analysis

Theorem 5.1. *Let $F_\bullet : \mathbb{C}\times\mathbb{C}^n \to \mathbb{C}^n$ be a circuit. Let (x, r, A) be a $\frac{7}{8}$-Moore box for F_0 with associated zero ζ. Let $I \subseteq \mathbb{R}$ and $Z : I \to \mathbb{C}^n$ be defined as in Section 5.1.*

Algorithm 3 terminates if and only if $1 \in I$. In this case, it outputs a $\frac{7}{8}$-Moore box for F_1 with associated zero Z_1.

We first prove the termination of the inner "while" loop. Assume it does not terminate. Let $K_T \in \square\mathbb{C}^n$ denote the box vector computed in $M(\square F_T, \square\mathrm{d}F_T, x, r, A, \frac{7}{8})$ and let K_t denote the one that would be computed in $M(\square F_t, \square\mathrm{d}F_t, x, r, A, \frac{1}{8})$. Because the loop does not terminate, we always have $\|K_T\|_\square > \frac{7}{8}$. However, the triple (x, r, A) comes from *refine*, with contraction factor $\frac{1}{8}$. This procedure checks $M(\square F_t, \mathrm{d}\square F_t, x, r, A, \frac{1}{8})$, so it is guaranteed that $\|K_t\|_\square \le \frac{1}{8}$. Again because the loop does not terminate, both h and u_{prec} go to 0. So $T \to t$ in the Hausdorff metric, while A, x and r are fixed. In particular, $K_T \to K_t$, by Proposition 3.1(ii), which makes the two inequalities above contradict each other.

We now consider the iterations of the main loop. Let x_k, r_k, A_k, t_k, and T_k be the value of the respective variables at the end of the kth iteration. Moreover, let $T_0 = \{0\} \in \square\mathbb{R}$ and let (x_0, r_0, A_0) denote the input Moore box. Let N be the total number of iterations, perhaps infinite. Recall that I is the maximal interval of definition of Z.

Lemma 5.2. *For any $0 \le k \le N$, $T_k \subseteq I$ and for any $t \in T_k$, (x_k, r_k, A_k) is a $\frac{7}{8}$-Moore box for F_t with associated root Z_t.*

Proof. We proceed by induction on k. The base case $k = 0$ is simply the input assumption. Assume the statement holds for $k - 1$. Let s be the supremum of T_{k-1}, which is also the infimum of T_k. By induction hypothesis, $(x_{k-1}, r_{k-1}, A_{k-1})$ is a $\frac{7}{8}$-Moore box for F_s with associated zero Z_s. By definition,

$$(x_k, r_k, A_k) = \mathit{refine}(F_s, x_{k-1}, r_{k-1}, A_{k-1}, \tfrac{1}{8}).$$

(We now drop the index k everywhere.) The correctness of *refine* implies that (x, r, A) is a $\frac{1}{8}$-Moore box of F_s with same associated zero. By construction, $M(\square F_T, \square\mathrm{d}F_T, x, r, A, \frac{7}{8})$ holds. By Lemma 3.2, this implies that (x, r, A) is a $\frac{7}{8}$-Moore box for F_t for any $t \in T$.

It remains to prove that $T \subseteq I$ and that for any $t \in T$, the zero of F_t associated to (x, r, A) is Z_t. Let $J = \{t \in T \cap I \mid \|Z_t - x\| \le r\}$. This is, by definition, a closed set in $T \cap I$. It is not empty: $s \in J$. Moreover, for any $t \in J$, Z_t is the zero of F_t associated to (x, r, A), because there is a unique zero of F_t in the ball $x + rB$. Therefore, by Theorem 2.1(i), we also have $\|Z_t - x\| \le \frac{7}{8}r$. It follows that J is also open in $T \cap I$. Since $T \cap I$, is an interval, we have $J = T \cap I$.

It only remains to prove that $T \subseteq I$. Recall that $s = \inf T \in I$ and let $b = \sup(T \cap I)$. By definition, $\|Z_t - x\| \le r$ for any $t \in T \cap I$, and it remains true as $t \to b$. Moreover, by Theorem 2.1(iii), $\mathrm{d}F_t(Z_t)^{-1}$ stays bounded, and it remains true as $t \to b$. This shows that Z_t does not come close to a critical point of F_t as $t \to b$. This prevents b from being a boundary point of I (see Section 5.1). Therefore $T \subseteq I$. □

This proves a part of Theorem 5.1: if Algorithm 3 terminates, then $1 \in I$ (because $1 \in T_N \subseteq I$) and Z_1 is the associated zero of the output. It remains to prove that if $1 \in I$, then the algorithm terminates. For contradiction, assume it does not. The value of the variable t converges to some $s \in [0, 1]$. The step size h goes to 0, and $T \to \{s\}$ in the Hausdorff metric. By construction, we also have $u_{\mathrm{prec}} \to 0$. Lastly, Lemma 5.2 and Theorem 2.1(i) imply that x stays in a bounded set. Indeed, we always have $\|x - Z_t\| \le r$ for some $t \in [0, 1]$, and $r \le 1$, by construction of *refine*. Since $t \mapsto Z_t$ is continuous on $[0, 1]$, it is bounded. Similarly Theorem 2.1(iv) implies that A stays in a bounded set, because $t \mapsto \mathrm{d}F_t(Z_t)^{-1}$ is continuous for $t \in [0, 1]$.

By contruction of *refine*, we always have

$$\left\|-r^{-1}A \cdot \square F_t(x) + (I_n - A \cdot \square\mathrm{d}F_t(x + rB)) \cdot B\right\|_\square \le \tfrac{1}{8}, \quad (16)$$

after Line 4 of each iteration.

We first consider the case where r stays away from 0: there is some $r_0 > 0$ such that $r \ge r_0$ at every iteration. Since $h \to 0$, it is divided infinitely many times, and at least one $M(\dots)$ check fails when this happens. So infinitely often, we have

$$\left\|-r^{-1}A \cdot \square F_T(x) + (I_n - A \cdot \square\mathrm{d}F_T(x + rB)) \cdot B\right\|_\square > \tfrac{1}{2}. \quad (17)$$

As established above, A and r are bounded, so we may assume that they converge (and r converges to a positive value). Since t and T have the same limit, in the Hausdorff metric, and $u_{\mathrm{prec}} \to 0$, the two left-hand sides in (16) and (17) also have the same limit (Propositon 3.1(ii)), which contradicts the inequalities.

So we assume that r does not stay away from zero, and considering a subset of the iterations, we may assume that $r \to 0$. In this case, $x \to Z_s$. By construction of *refine*, we also have $A \to \mathrm{d}F_s(Z_s)^{-1}$.

Moreover, when some $r < 1$ is returned, this means that the box could not be grown, that is

$$\left\|-\tfrac{1}{2}r^{-1}A \cdot \square F_t(x) + (I_n - A \cdot \square \mathrm{d}F_t(x + 2rB)) \cdot B\right\|_\square > \tfrac{1}{8}.$$

The part $I_n - A \cdot \square \mathrm{d}F_t(x+2rB)$ converges to 0, and after multipliying by 2, this leads to

$$\left\|-r^{-1}A \cdot \square F_t(x)\right\|_\square \geq \tfrac{1}{4} + o(1), \tag{18}$$

in contradiction with (16) which shows that this magnitude is at most $\frac{1}{8} + o(1)$. This concludes the proof of termination.

6 PREDICTORS

6.1 Rationale

The performance of Algorithm 3 can be greatly improved by incorporating Taylor models. Consider an iteration of the main loop of *track*. After the *refine* step, we have t x, r, and A such that

$$\left\|-r^{-1}A \cdot F_t(x) + (I_n - A \cdot \mathrm{d}F_t(x + rB)) \cdot B\right\| \leq \tfrac{1}{8},$$

and we want a $\delta > 0$ as large as possible such that for all $\eta \in [0, \delta]$,

$$\Big\| \underbrace{-r^{-1}A \cdot F_{t+\eta}(x)}_{V} + \underbrace{(I_n - A \cdot \mathrm{d}F_{t+\eta}(x + rB)) \cdot B}_{\Delta} \Big\| \leq \tfrac{7}{8}.$$

For an informal analysis, we may consider r and δ as infinitesimally small and compute with first order expansions. We obtain that $\|\Delta\| = O(r + \eta)$ and

$$\|V\| = r^{-1}\|v\|\eta + O(r^{-1}\eta^2) = O(r^{-1}\eta),$$

where $v = -A \cdot \dot{F}_t(x)$ is the *speed vector* related to variation of the zero that we track as the parameter changes (see Equation 15). So the V term is likely to be the main obstruction in raising δ and this suggests that we may expect $\delta \simeq r$.

In the special case where the speed vector v vanishes, we may expect the much better $\delta \simeq \sqrt{r}$. In principle, it is easy to perform the first order correction and reduce to the stationary case by introducing the auxilliary system $G_\eta(x) = F_{t+\eta}(x - \eta v)$ which is made to satisfy $\dot{G}_0(x) = 0$. If we can rigorously track a zero of G_η as η moves from 0, we can certainly transfer this information to the original sytem $F_{t+\eta}$. Higher order corrections are also possible, we can enforce the cancellation of more terms in V, which will increase the step size until the Δ term takes over. However, this idea does not combine nicely with interval arithmetic which never cancels anything. This phenomenom, well known as the *dependency problem*, obliterates all ideas based on the cancelations of some dominant terms. Taylor models is a classical way to circumvent it.

6.2 Taylor models

Let $I \in \square\mathbb{R}$ be an interval containing 0. A *Taylor model of order ν on I* is a polynomial $P(\eta) = a_0 + a_1\eta + \cdots + a_{\nu+1}\eta^{\nu+1}$ of degree at most $r + 1$ with coefficients in $\square\mathbb{C}$. A Taylor model P on I encloses a function $f : \mathbb{R} \to \mathbb{C}$ if for any $t \in I$, there are $\bar{a}_0, \ldots, \bar{a}_{\nu+1} \in \mathbb{C}$ such that $\bar{a}_i \in a_i$ and $f(t) = \bar{a}_0 + \bar{a}_1 t + \cdots + \bar{a}_{\nu+1}t^{\nu+1}$. If $J \in \square\mathbb{R}$ is a subinterval of I then the interval computation of $P(J)$ contains $f(J)$ for any function f enclosed by P.

Since the variable t is bound to I, we can squeeze a Taylor model of order r into one of order $\nu - 1$ by replacing the last two terms $a_\nu\eta^\nu + a_{\nu+1}\eta^{\nu+1}$ by the single term $(a_\nu \boxplus (a_{\nu+1} \boxtimes I))\,\eta^\nu$. If P encloses a function f, then the reduced model still does. We define on order-ν Taylor models an addition by the componentwise addition of intervals $\boxplus$. We define also a multiplication by the usual polynomial multiplication formula, but using $\boxplus$ and $\boxtimes$, which leads to a Taylor model of order $2\nu + 1$, followed by repeated sequeezing to reduce to order ν. These operations are naturally compatible with the enclosure of functions.

In general, the first coefficients $a_0, \ldots, a_\nu$ of a Taylor model of order ν are narrow intervals enclosing the Taylor expansion of an enclosed function, their width reflect only roundoff errors. The last term $a_{\nu+1}\eta^{\nu+1}$ reflects the $O(\eta^{\nu+1})$ term of an order-ν Taylor expansion. For more details on Taylor models, we refer to Berz and Hoffstätter [5], Joldes [14], Moore et al. [22], Neumaier [24].

6.3 Path tracking with a predictor

We consider the same setting as in Section 5. Suppose that for some $t \in [0, 1]$, we have a $\frac{1}{8}$-Moore box (x, r, A) for F_t. Suppose also that we have a vector $\mathcal{X}(\eta)$ of polynomials such that $\mathcal{X}(0) = x$. Naturally, we will choose $\mathcal{X}(\eta)$ to approximate the zero $Z_{t+\eta}$ of $F_{t+\eta}$ the best we can, but we assume nothing. $\mathcal{X}$ is called the *predictor*. Let also $h > 0$ be what we think is a good step size.

Using the arithmetic of order-ν Taylor models on the domain $[0, h]$ (we will typically choose $\nu = 3$), we compute

$$\mathcal{K} = -r^{-1}A \cdot \square F_{t+\eta}(\mathcal{X}) + \left[I_n - A \cdot \square \mathrm{d}F_{t+\eta}(\mathcal{X} + rB)\right] \cdot B,$$

which is a vector of Taylor models. Then we compute $\mathcal{K}([0, h])$ and check if it is included in $\frac{7}{8}B$. If it is, then $(\mathcal{X}(e), r, A)$ is a $\frac{7}{8}$-Moore box for F_{t+e} for any any $e \in [0, h]$. This follows from the compatibility of the Taylor model arithmetic with enclosures.

If $\mathcal{K}([0, h])$ is not included in $\frac{7}{8}B$, we can try with $\mathcal{K}([0, \frac{h}{2}])$, or maybe $\mathcal{K}([0, \frac{h}{4}])$, we do not need to restart the computation of $\mathcal{K}$ from scratch with a lower step size. In principle, we may assume that $\mathcal{K}(0) \subseteq \frac{1}{8}B$, so there should be some $j \in (0, h]$ such that $\mathcal{K}([0, j]) \subseteq \frac{7}{8}B$, but if we need j to be very small, it makes more sense instead to recompute $\mathcal{K}$ over a smaller domain.

What predictor can we choose? Ideally, we would choose $\mathcal{X}$ to be a truncated Taylor expansion of $Z_{t+\eta}$ around $\eta = 0$. But only the first term is easy to get: by Equation (15), we have

$$Z_{t+\eta} = Z_t - \mathrm{d}F_t(Z_t)^{-1} \cdot \dot{F}_t(Z_t)\eta + O(\eta^2).$$

Since x approximates Z_t and A approximates $\mathrm{d}F_t(Z_t)^{-1}$, we may choose $\mathcal{X}_{\text{tangent}} = x - A \cdot \dot{F}_t(x)\eta$. We compute $\dot{F}_t(x)$ by automatic differentiation, similarly to $\mathrm{d}F_t(x)$. This is the *tangent predictor* and it leads to Algorithm 4.

Theorem 6.1. *Algorithm 4 is correct and terminates, in the sense of Theorem 5.1.*

Mutatis mutandis, the proof is the same as the one of Theorem 5.1. The quality of the predictor does not matter much, as long as it stays bounded.

If we record the previous values of x and the tangent vector, and the previous step size, we can compute the *Hermite predictor*

$$\mathcal{X}_{\text{Hermite}} = x + v\eta + (2w - 3\Delta x)\frac{\eta^2}{h_{\text{prev}}} + (w - 2\Delta x)\frac{\eta^3}{h_{\text{prev}}^2},$$

where $w = v + v_{\text{prev}}$ and $\Delta x = h_{\text{prev}}^{-1}(x - x_{\text{prev}})$. It is the unique polynomial with $\mathcal{X}(0) = x$, $\mathcal{X}'(0) = v$, $\mathcal{X}(-h_{\text{prev}}) = x_{\text{prev}}$ and $\mathcal{X}'(-h_{\text{prev}}) = v_{\text{prev}}$. This predictor, with order-3 Taylor models,

Algorithm 4 Path tracking with the tangent predictor

input $F_\bullet$, a circuit $\mathbb{C} \times \mathbb{C}^n \to \mathbb{C}^n$; (x, r, A), a $\frac{7}{8}$-Moore box for F_0
output a Moore box for F_1

def *track*$(F_\bullet, x, r, A)$:
  $t \leftarrow 0$; $h \leftarrow \frac{1}{2}$;
  while $t < 1$:
    $x, r, A \leftarrow \mathit{refine}(F_t, x, r, A, \frac{1}{8})$
    $h \leftarrow \frac{5}{4}h$ # try growing the step size
    $v \leftarrow \mathrm{mid}(-A \cdot \square\dot{F}_t(x))$
    $\mathcal{X} \leftarrow x + v\eta$ # η is the variable of Taylor models
    # compute $\mathcal{K}$ with order-2 Taylor model arithmetic on $[0, h]$
    $\mathcal{K} \leftarrow -r^{-1}A \cdot \square F_{t+\eta}(\mathcal{X}) + \left[I_n - A \cdot \square \mathrm{d}F_{t+\eta}(\mathcal{X} + rB)\right] \cdot B$
    if $\|\mathcal{K}([0, h])\|_\square > \frac{7}{8}$:
      $h \leftarrow \frac{h}{2}$
      $u_{\mathrm{prec}} \leftarrow \min(u_{\mathrm{prec}}, h)$
      if $\|\mathcal{K}([0, \frac{h}{2}])\|_\square > \frac{7}{8}$:
        # unsuccessful, restart the iteration with smaller h
        continue
    $t \leftarrow t + h$
    $x \leftarrow \mathrm{mid}(\mathcal{X}(h))$
  return x, r, A

gave us very good results, on which we report in Section 7. Compared to uncertified method, the complexity of the predictor has a larger impact on the computation time: the predictor is not only used as a guess, we need to validate it. So the balance between the complexity of the predictor and the number of iterations it saves does not favor high order methods.

By design of Algorithm 4, from one Moore box (x, r, A), at a given t, to the next point x' at time t', it is necessary that $(I_n - A \cdot \mathrm{d}F_{t'}(x')) \cdot B \subseteq \frac{7}{8}B$. This provides a theoretical maximum step length allowed by Moore's criterion, independent of the quality of the predictor, or overestimation issues in interval arithmetic. In experimentation, we observed that the Hermite predictor brings us very close to this theoretical limit. So we do not expect that higher order predictors may improve performance.

7 EXPERIMENTS

7.1 Implementation

We propose a Rust implementation of Algorithm 4, using the Hermite predictor. Only fixed precision is implemented: all intervals have 64-bits floating-point endpoints, and when the algorithm warns about precision, the computation is aborted. Interval arithmetic is implemented using the AVX instruction set for the x86-64 platform, following Lambov [17]. For example, the multiplication of two order-3 Taylor models, that is 100 real interval number multiplications and 40 additions, is performed with 638 SIMD instructions in less than 300 CPU cycles, according to the analysis tool *llvm-mca*. The source code is distributed under the GPLv3 license and available at

https://gitlab.inria.fr/numag/algpath.

7.2 Timings

We compared our implementation *algpath* with the Macaulay2 package *NumericalAlgebraicGeometry* [3]. As a state-of-the-art implementation of noncertified path tracking, we also benchmarked the Julia package *HomotopyContinuation.jl* [6]. We used a computer with an Intel Xeon E3-1220v3 CPU and 16GB of RAM. Path tracking algorithms uses very little memory, but we report some out-of-memory errors from Julia (to be investigated) and Macaulay2 (which tries to expand structured polynomials). The benchmarking data, scripts and raw results are available at

https://gitlab.inria.fr/numag/algpath-bench.

7.2.1 Data set. We considered linear homotopies, $F_t(x) = tf(x) + (1-t)g(x)$ between a start polynomial system g and a target system f. For the target, we considered several families. First, dense systems with random standard independent complex coefficients. Second, structured random systems, with low Waring rank, with components of the form $\pm 1 + \sum_{i=1}^{5} \left(\sum_{j=1}^{n} a_{i,j}x_j\right)^d$ with random coefficients $a_{i,j}$, independent and uniformly distributed in $\{-1, 0, 1\}$. Third, we considered the classical benchmark family Katsura n (available in Sagemath with `sage.rings.ideal.Katsura`), which is a polynomial system in $n+1$ variables with 2^n solutions.

For the start system, we considered total degree homotopies, with $g_i(x) = \gamma_i(x_i^d - 1)$ and $\gamma_i \in \mathbb{C}$ random, as well as Newton homotopies, with $g(x) = f(x) - f(x_0)$ for some random x_0.

7.2.2 Results analysis. Table 1 shows that the number of steps per second performed by *Macaulay2* is comparable to that of *HomotopyContinuation.jl*[2]. By strongly decreasing the number of steps required to track a path compared to *Macaulay2*, this work is able to solve problems of a much bigger scale. The gap between certified and noncertified methods is significantly narrowed.

Comparing the median number of steps performed by *HC.jl* and *algpath* on each problem suggests that, typically, the latter performs only 2 or 3 times more steps. Figure 1 inspects this relation more precisely. The correlation between the number of steps of *HC.jl* and *algpath* is strong on some examples (such as random dense or structured dimension-8 degree-3 systems), but weaker on some others (difficult paths in the degree 500 univariate polynomial, or Katsura examples).

Finally, we reach the limits of 64-bits floating-point arithmetic sooner than *HomotopyContinuation.jl*, as shown by the number of failures for high degree univariate instances, or Katsura-40. Note that we did not check systematically the absence of path swapping in *HC.jl*, there may be silent failures.

ACKNOWLEDGMENTS

We are grateful to Florent Bréhard, and Mioara Joldes for useful discussions, and to the referees for thoughtful reports. This work

[2] Rigorous benchmarking of Julia code is difficult because of run-time compilation (JIT). Following common practice, we run twice `HomopotopyContinuation.solve` with exactly the same arguments. The first run suffers from compilation overheads, while the second does not. However, some compilation overheads are input dependent. For fair comparison, it makes no sense to only time the second run (because it is too easy to compute something faster after we did it once). So the *total time* in Table 1 is the time of the first run. To lessen the compilation overheads in this first run, we perform a warmup run with a polynomial system of degree 1 and same dimension as the input system. In constrast, the *number of steps per second* is obtained from the time of the second run, divided by the total number of iterations.

				circuit size		HomotopyContinuation.jl					algpath					Macaulay2				
name	dim.	max deg	# paths	f	df	fail.	med.	max.	ksteps/s	time (s)	fail.	med.	max.	ksteps/s	time (s)	fail.	med.	max.	ksteps/s	time (s)
dense	1	10	10	88	96		6	10	30	1.8		11	31	55	< 0.1		629	2146	55	0.2
dense	1	20	20	168	202		13	23	53	1.8		29	134	42	< 0.1		40 k	183 k	46	20
dense	1	30	30	248	314		10	25	41	2.0		23	372	25	< 0.1		830 k	3478 k	30	18 min
dense	1	40	40	328	416		14	30	45	2.0		34	197	24	< 0.1			> 1 h		
dense	1	50	50	408	520		12	61	37	1.9		30	5567	13	0.7			> 1 h		
dense	1	100	100	808	1054		13	51	23	1.9		38	5289	7.4	1.4			> 1 h		
dense	1	500	500	4008	5466		14	59	3.8	3.9	2	60	1121	2.3	17	500				4.0
dense	1	1000	1000	8008	10952		15	100	1.7	12	35	74	976	1.1	82	1000				29
dense	2	5	25	316	368		23	56	57	2.3		50	95	25	< 0.1		2850	6736	53	1.4
dense	2	10	100	1016	1280		22	74	33	2.6		53	307	9.2	0.7		33 k	301 k	28	158
dense	2	20	400	3616	4612		25	63	13	3.1		74	401	2.9	12			> 1 h		
dense	2	30	900	7816	9952		24	127	5.8	6.4		85	690	1.4	72			> 1 h		
dense	2	40	1600	13616	17284		25	95	3.4	14		100	998	0.81	268			> 1 h		
dense	2	50	2500	21016	26624		27	84	2.3	33		117	1675	0.53	12 min			> 1 h		
katsura	5	2	16	192	98		49	74	41	3.8		74	136	26	< 0.1		3833	7903	38	1.9
katsura	7	2	64	310	158		59	99	59	3.9		100	203	15	0.5		5963	15 k	26	16
katsura	9	2	256	448	228		82	132	54	4.2		148	286	9.5	4.2		12 k	59 k	18	186
katsura	11	2	1024	606	308		100	179	41	6.7		177	359	6.3	30		21 k	88 k	13	30 min
katsura	16	2	32768	1090	548		153	303	22	235		304	1847	2.7	1 h			> 50 h		
katsura	21	2	1048576	1696	844		209	469	13	4 h	483	427	8798	1.4	101 h		not benchmarked			
katsura *	26	2	100	2430	1202		305	466	6.9	8.8	1	800	2930	0.73	125			> 1 h		
katsura *	31	2	100	3286	1614		382	538	4.9	12	1	852	5021	0.47	219			> 1 h		
katsura *	41	2	100	5376	2618		554	787	2.7	24	9	1371	5182	0.19	13 min			> 1 h		
dense *	4	3	100	1080	1318		39	67	41	2.4		66	127	8.3	0.9		3384	9936	35	10
dense *	6	3	100	4092	5384		54	96	9.0	3.3		112	224	2.3	5.1		11 k	24 k	18	62
dense *	8	3	100	11120	15242		73	124	2.1	6.3		157	354	0.86	19		21 k	74 k	9.5	243
structured *	4	3	100	244	418		40	78	92	4.0		75	199	24	0.4		4531	8925	41	11
structured *	6	3	100	426	778		66	101	59	3.9		130	254	13	1.1		18 k	61 k	23	85
structured *	8	3	100	670	1252		81	121	40	3.9		182	283	7.9	2.3		36 k	97 k	13	305
structured N	5	5	1	302	545		42	42	4.9	3.1		99	99	18	< 0.1		252 k	252 k	12	22
structured N	10	10	1	1034	2024		53	53	0.18	3.1		123	123	4.9	< 0.1			> 1 h		
structured N	15	15	1	2366	5079			> 8 GB				628	628	2.0	0.4			> 8 GB		
structured N	20	20	1	3554	6721			> 8 GB				1591	1591	1.2	1.5			> 8 GB		
structured N	25	25	1	5466	10541			> 8 GB				1734	1734	0.69	2.9			> 8 GB		
structured N	30	30	1	7788	15239			> 8 GB				1989	1989	0.43	5.2			> 8 GB		
dense N	4	3	1	792	1038		18	18	0.71	2.5		50	50	12	< 0.1		21 k	21 k	27	0.8
dense N	6	3	1	3072	4376		33	33	0.12	2.7		90	90	3.6	0.1		22 k	22 k	13	1.8
dense N	8	3	1	8464	12602		10	10	< 0.01	5.0		35	35	1.1	0.4		8775	8775	5.5	1.6
structured N	4	3	1	200	296		32	32	4.9	3.2		66	66	28	< 0.1		8559	8559	34	0.3
structured N	6	3	1	350	516		32	32	2.9	2.9		79	79	15	< 0.1		31 k	31 k	19	1.7
structured N	8	3	1	566	876		21	21	1.0	2.9		73	73	8.8	< 0.1		19 k	19 k	8.0	2.3

Table 1: *Comparison of* HomotopyContinuation.jl, algpath *(this work) and* Macaulay2. Dim.*: number of variables;* deg.*: maximum degree of the equations;* # paths*: number of paths to track;* circuit size*: size of the circuit evaluating the parametric system and its derivative;* fail.*: number of reported failures;* med.*: median number of iterations over all paths;* max.*: maximum number of iterations in a single path;* ksteps/s*: number of steps per second (thousands);* time*: total time to track all the paths.* * *A hundred randomly picked starting zeros of a total degree homotopy.* N *Newton homotopy.*

has been supported by the Agence nationale de la recherche (ANR), grant agreement ANR-19-CE40-0018 (De Rerum Natura); and by the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme, grant agreement 101040794 (10000 DIGITS).

REFERENCES

[1] G. Alefeld and G. Mayer. 2000. Interval Analysis: Theory and Applications. *J. Comput. Appl. Math.* 121, 1 (2000), 421–464.
[2] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. 2013. *Numerically Solving Polynomial Systems with Bertini.* Software, Environments, and Tools, Vol. 25. SIAM, Philadelphia, PA.
[3] C. Beltrán and A. Leykin. 2012. Certified Numerical Homotopy Tracking. *Exp. Math.* 21, 1 (2012), 69–83. https://doi.org/10/ggck73
[4] C. Beltrán and A. Leykin. 2013. Robust Certified Numerical Homotopy Tracking. *Found. Comput. Math.* 13, 2 (2013), 253–295. https://doi.org/10/ggck74
[5] M. Berz and G. Hoffstätter. 1998. Computation and Application of Taylor Polynomials with Interval Remainder Bounds. *Reliab. Comput.* 4, 1 (1998), 83–97. https://doi.org/bqmsdm
[6] P. Breiding and S. Timme. 2018. HomotopyContinuation.Jl: A Package for Homotopy Continuation in Julia. In *Int. Congr. Math. Softw.* 458–465. https://doi.org/10/ggck7q
[7] P. Bürgisser. 2000. *Completeness and Reduction in Algebraic Complexity Theory.* Springer-Verlag. https://doi.org/10/d9n4
[8] F. Cucker. 2021. Smale 17th Problem: Advances and Open Directions. *N. Z. J. Math.* 52 (2021), 233–257. https://doi.org/gtc2rc
[9] T. Duff and K. Lee. 2024. Certified Homotopy Tracking Using the Krawczyk Method. arXiv:2402.07053
[10] J. D. Hauenstein, I. Haywood, and A. C. Liddell, Jr. 2014. An a Posteriori Certification Algorithm for Newton Homotopies. In *Proc. ISSAC 2014.* ACM, 248–255. https://doi.org/10/ggck7h
[11] J. D. Hauenstein and A. C. Liddell. 2016. Certified Predictor–Corrector Tracking for Newton Homotopies. *J. Symb. Comput.* 74 (2016), 239–254. https://doi.org/10/ggck7j
[12] J. D. Hauenstein, J. I. Rodriguez, and F. Sottile. 2017. Numerical Computation of Galois Groups. *Found. Comput. Math.* (2017), 1–24. https://doi.org/gd2rw6
[13] F. Johansson. 2017. Arb: Efficient Arbitrary-Precision Midpoint-Radius Interval Arithmetic. *IEEE Trans. Comput.* 66, 8 (2017), 1281–1292. https://doi.org/10/gbn9sm
[14] M. Joldes. 2011. *Rigorous Polynomial Approximations and Applications.* Ph. D. Dissertation. École normale supérieure de lyon. https://theses.hal.science/tel-00657843
[15] S. Kranich. 2015. An Epsilon-Delta Bound for Plane Algebraic Curves and Its Use for Certified Homotopy Continuation of Systems of Plane Algebraic Curves. arXiv:1505.03432

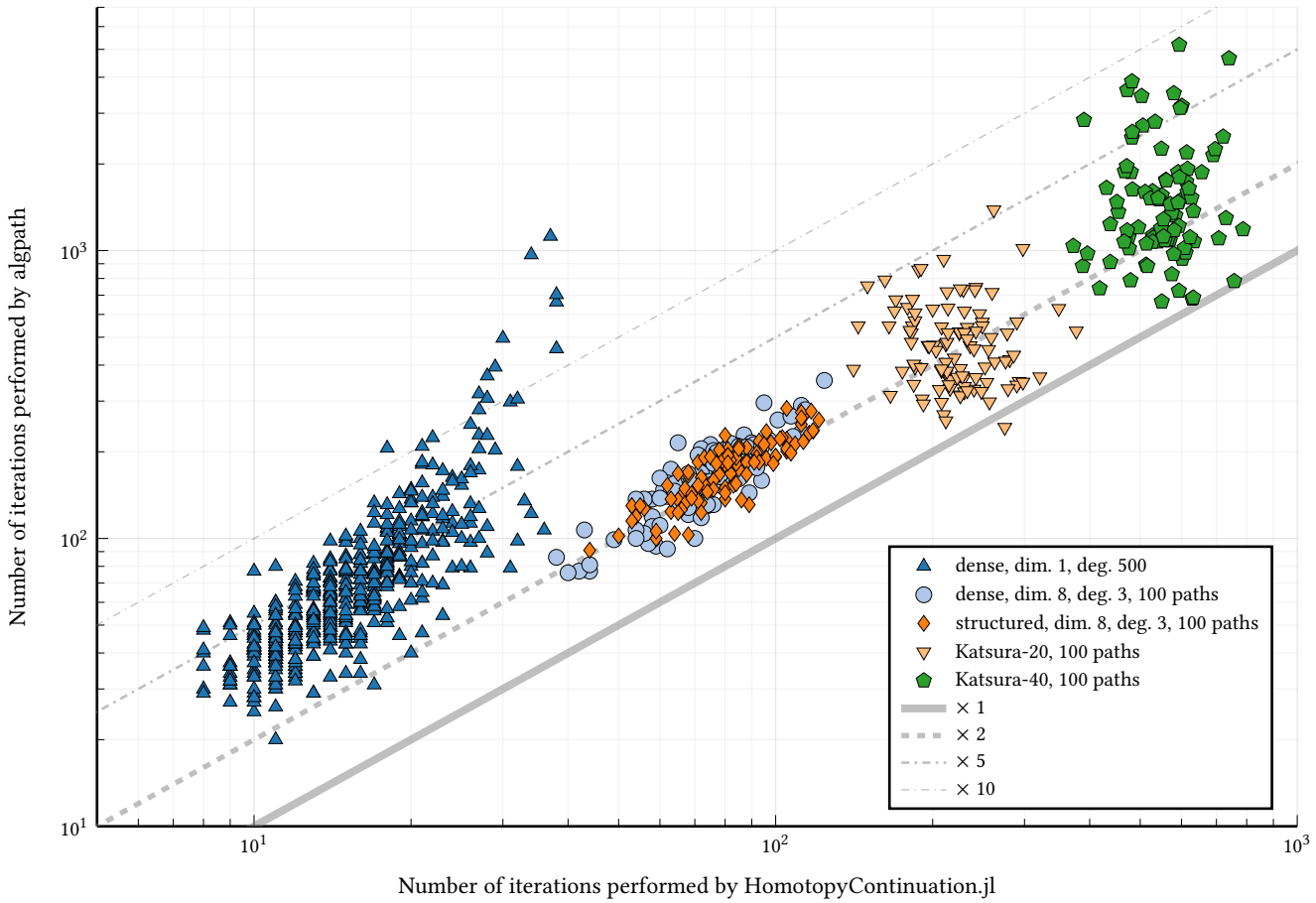

Figure 1: *Number of iterations performed by* algpath *(this work) and* HomotopyContinuation.jl *(noncertified path tracking) in four path tracking problems. We observe that* algpath *performs typically no more than 5 times more iterations than a state-of-the-art noncertified numerical solver. The ratio is close to 2 on well-conditioned examples (○ and ♦) but there is much more variability on poor conditioning (▲, ▽ and ⬟).*

[16] R. Krawczyk. 1969. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing* 4, 3 (1969), 187–201. https://doi.org/10/css7z9

[17] B. Lambov. 2008. Interval Arithmetic Using SSE-2. In *Reliab. Implement. Real Number Algorithms (Lecture Notes in Computer Science)*, P. Hertling, C. M. Hoffmann, W. Luther, and N. Revol (Eds.). Springer, 102–113. https://doi.org/c7vvrk

[18] S. Lang. 1997. *Undergraduate Analysis* (2 ed.). Springer. https://doi.org/gtcznr

[19] A. Leykin. 2011. Numerical Algebraic Geometry. *J. Softw. Algebra Geom.* 3, 1 (2011), 5–10. https://doi.org/ggck9r

[20] M. Á. Marco-Buzunariz and M. Rodríguez. 2016. SIROCCO: A Library for Certified Polynomial Root Continuation. In *Proc. ICMS 2016 (LNCS)*. Springer, 191–197. https://doi.org/10/grqk32

[21] R. E. Moore. 1977. A Test for Existence of Solutions to Nonlinear Systems. *SIAM J. Numer. Anal.* 14, 4 (1977), 611–615. https://doi.org/10/c66n76 jstor:2156481

[22] R. E. Moore, R. B. Kearfott, and M. J. Cloud. 2009. *Introduction to Interval Analysis*. SIAM. https://doi.org/c8ctwd

[23] J.-M. Muller, N. Brunie, F. De Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. 2018. *Handbook of Floating-Point Arithmetic* (2 ed.). Springer. https://doi.org/gtdkwj

[24] A. Neumaier. 2003. Taylor Forms—Use and Limits. *Reliable Computing* 9, 1 (2003), 43–79. https://doi.org/c48bk4

[25] N. Revol and F. Rouillier. 1999/2023. MPFI. https://gitlab.inria.fr/mpfi/mpfi

[26] J. I. Rodriguez and B. Wang. 2017. Numerical Computation of Braid Groups. arXiv:1711.07947

[27] S. M. Rump. 1983. Solving Algebraic Problems with High Accuracy. In *A New Approach to Scientific Computation*, U. W. Kulisch and W. L. Miranker (Eds.). Academic Press, 51–120. https://doi.org/10/kh8k

[28] A. J. Sommese, J. Verschelde, and C. W. Wampler. 2001. Numerical Decomposition of the Solution Sets of Polynomial Systems into Irreducible Components. *SIAM J. Numer. Anal.* 38, 6 (2001), 2022–2046. https://doi.org/10/fv5jzk

[29] A. J. Sommese, J. Verschelde, and C. W. Wampler. 2005. Introduction to Numerical Algebraic Geometry. In *Solving Polynomial Equations*, M. Bronstein, A. M. Cohen, H. Cohen, D. Eisenbud, B. Sturmfels, A. Dickenstein, and I. Z. Emiris (Eds.). Springer, 301–337. https://doi.org/10/bzsc24

[30] S. Telen, M. Van Barel, and J. Verschelde. 2020. A Robust Numerical Path Tracking Algorithm for Polynomial Homotopy Continuation. *SIAM J. Sci. Comput.* 42, 6 (2020), A3610–A3637. https://doi.org/10/grqm6n

[31] J. van der Hoeven. 2015. Reliable Homotopy Continuation. https://hal.science/hal-00589948v4

[32] J. Verschelde. 1999. Algorithm 795: PHCpack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation. *ACM Trans. Math. Softw. TOMS* 25, 2 (1999), 251–276. https://doi.org/10/fncfxj

[33] J. Xu, M. Burr, and C. Yap. 2018. An Approach for Certifying Homotopy Continuation Paths: Univariate Case. In *Proc. ISSAC 2018*. ACM, 399–406. https://doi.org/10/ggck7k