Adaptive multiprecision path tracking

Daniel J. Bates* Jonathan D. Hauenstein†

Andrew J. Sommese‡ Charles W. Wampler II§

March 8, 2007

Abstract

This article treats numerical methods for tracking an implicitly defined path. The numerical precision required to successfully track such a path is difficult to predict a priori, and indeed, it may change dramatically through the course of the path. In current practice, one must either choose a conservatively large numerical precision at the outset or re-run paths multiple times in successively higher precision until success is achieved. To avoid unnecessary computational cost, it would be preferable to adaptively adjust the precision as the tracking proceeds in response to the local conditioning of the path. We present an algorithm that can be set to either reactively adjust precision in response to step failure or proactively set the precision using error estimates. We then test the relative merits of reactive and proactive adaptation on several examples arising as homotopies for solving systems of polynomial equations.

2000 Mathematics Subject Classification. Primary 65H10; Secondary 65H20, 65G50, 14Q99.

Key words and phrases. Homotopy continuation, numerical algebraic geometry, polynomial systems, Bertini.

---

*Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55122 (dbates1@nd.edu, www.nd.edu/∼dbates1) This author was supported by the Duncan Chair of the University of Notre Dame, the University of Notre Dame, NSF grant DMS-0410047, the Arthur J. Schmitt Foundation, and the Institute for Mathematics and its Applications in Minneapolis (IMA)

†Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556 (jhauenst@nd.edu, www.nd.edu/∼jhauenst). This author was supported by the Duncan Chair of the University of Notre Dame, the University of Notre Dame, NSF grant DMS-0410047, and the Institute for Mathematics and its Applications in Minneapolis (IMA)

‡Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556 (sommese@nd.edu, www.nd.edu/∼sommese). This author was supported by the Duncan Chair of the University of Notre Dame, the University of Notre Dame, NSF grant DMS-0410047, and the Institute for Mathematics and its Applications in Minneapolis (IMA)

§General Motors Research and Development, Mail Code 480-106-359, 30500 Mound Road, Warren, MI 48090 (Charles.W.Wampler@gm.com, www.nd.edu/∼cwample1) This author was supported by NSF grant DMS-0410047 and the Institute for Mathematics and its Applications in Minneapolis (IMA)

1 Introduction

Path tracking is the task of tracing out a one-real-dimensional solution curve described implicitly by a system of equations, typically n equations in $n+1$ variables, given an initial point on, or close to, the path. This can arise in many ways, but our motivation is the solution of systems of polynomials via homotopy continuation (see [1, 10, 11, 14, 15, 22]). In this method, to find the isolated solutions of the system $f(z) = 0$ for given polynomials $f : \mathbb{C}^n \to \mathbb{C}^n$, one constructs a homotopy, $H(z,t)$, $H : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ such that $H(z,0) = f(z)$ is the target system to be solved while $H(z,1)$ is a starting system whose isolated solutions are known. There is a well-developed theory on how to construct such homotopies to guarantee, with probability one, that every isolated solution of $f(z) = 0$ is the endpoint in the limit as $t \to 0$ of at least one smooth path $z_i(t)$, where $H(z_i(t),t) = 0$ on $t \in (0,1]$ and where $z_i(1)$, $i = 1,2,\ldots$, are the known isolated solutions of $H(z,1) = 0$. Similar constructions arise in other contexts, where the existence of a path leading to the desired solutions may or may not be guaranteed. Even when there is no guarantee, experience shows that in some application domains continuation techniques yield solutions more reliably than Newton's method, especially when good initial guesses are not available. While in our applications the path is just a means of arriving at the endpoint, in other applications one may desire to accurately trace out the path itself, such as when plotting the response of a mathematical model as one of its parameters is varied.

The most common path tracking algorithms are predictor-corrector methods: from an approximate solution point on the path, a predictor gives a new approximate point a given step size along the path, then a corrector brings this new point closer to the path. For example, one may use an Euler predictor, which steps ahead along the tangent to the path, or a higher order predictor that uses one or more recent points and possibly derivatives of the homotopy function at them to extrapolate to the predicted point. Typically, the prediction is then used as the initial point for correction by Newton's method. Since the solution set is one-dimensional, an extra constraint is introduced to isolate the target of the correction. For general homotopies, a useful constraint is to find where the solution path intersects the hyperplane normal to the last computed tangent direction. In the more restrictive setting of polynomial systems, the homotopy can be designed such that the paths advance monotonically with t, that is, there are no turning points, in which case it is acceptable (and simpler) to perform corrections by holding t fixed. The adaptive precision algorithm we discuss here is compatible with any of these prediction and correction schemes.

For good results, the predictor step size must be chosen appropriately. Too large a step size may result in a prediction outside the zone of convergence of the corrector, while too small a step size means progress is slow and computationally costly. Consequently, it has long been recognized that adaptive control of the step size is crucial for obtaining good reliability without undue computational cost.

While step size control is well established, less attention has been paid to efficient handling of precision. With wider availability of software packages for higher precision arithmetic, along with faster computers to execute the software, it becomes interesting to consider how adjustable precision might be deployed to improve the performance of path tracking algorithms. The issue at stake is analogous to step size control: without enough precision, path tracking might fail, but the use of excessive precision is inefficient. To address this tradeoff, this paper proposes an algorithm that dynamically adjusts the number of digits used in computations according to the evolution of the numerical conditioning of the homotopy function.

In our primary application of interest, the solution of polynomial systems, there are several factors driving the need for higher precision. It is well known that high degree polynomials often lead to ill-conditioned problems. When treating polynomial systems in several variables, the total degree of the system, being the product of the degrees of the individual equations, quickly becomes large even for low degree polynomials, which can also lead to ill-conditioning. Thus, one driving force is the desire to solve larger systems of higher total degree.

A second motivation is that our systems often have some, or possibly many, singular solutions, and thus, the solution paths leading to these solutions are necessarily ill-conditioned near the end. While endgame methods exist for enhancing the accuracy with which such endpoints can be estimated, for singularities of high enough multiplicity, more precision is required.

A third motivation is that many polynomial systems from applications come with natural parameters. These systems with the parameters included typically have exact coefficients, which are relatively small integers. For "general values" of the parameters the structure of the solution set of the systems are the same [16], i.e., the number of multiplicity k points for $k = 1, 2, 3, \dots$ remains constant. If the coefficients of the polynomials are computed to sufficient accuracy using the given choice of parameter values, the solutions of the system will typically have a clean structure for multiple points, i.e., will not have a cluster of solutions near each other unless the points computed more precisely converge to the same point. Here it is important that the coefficients are computed accurately as functions of the parameters. For example, in §5.5, the polynomial system has coefficients computed accurately to 308 digits. The coefficients are polynomial functions of certain lengths, e.g., offsets and link lengths, and of cosines and sines of what are called twist angles. If only eight digits are used for the coefficients, as printed in the table in [15], then the algebraic relationships that should exist between the coefficients fail to hold to more than 8 digits. For example, some coefficients depend on $\cos\alpha$ and $\sin\alpha$, where α is a twist angle, and when these are rounded to 8 digits, the algebraic relation $\cos^2(\alpha) + \sin^2(\alpha) = 1$ no longer holds at higher precision. The effect of this is that the multiple points, e.g., the ones of multiplicity 24, form clusters that do not come together as precision is increased.

Finally, although the homotopy constructions guarantee, with probability one, that no path passes exactly through a singularity before reaching its endpoint, there is always a chance that a near-singular condition can be encountered. To obtain the highest reliability possible, we need to detect this and allocate sufficient digits to successfully track past such obstructions.

The paper is organized as follows. In section 2, we review the behavior of Newton's method in floating point, revealing how its accuracy and convergence properties depend on precision. In section 3, we discuss path tracking with adaptive step size control and identify how it fails when precision is insufficient. This leads, in section 4, to a novel technique for path tracking using adaptive precision. This new adaptive precision path tracking algorithm has been implemented in a software package, Bertini [2], currently under development by the authors. Several examples are presented in section 5 to illustrate the usefulness of adaptive precision. Section 6 contains a brief discussion of the value of adaptive precision in the presence of endgames. Finally, in section 7, a few related ideas that would make interesting studies are discussed.

We would like to thank the referees for their insightful comments, which have resulted in this being a much better article.

2 Background: Newton's method

The core numerical process in the path tracker is the corrector, which in our case is Newton's method. A good predictor speeds up the path tracker by allowing a large step while still supplying an initial guess within the convergence region of the corrector. However, it is the loss of convergence that causes path tracking to fail. In exact arithmetic, as long as the path remains nonsingular, there must be a region surrounding the path within which Newton's method converges quadratically. With a small enough step Δt in t, we can be assured of advancing along the path, although possibly very slowly. This holds even if we use only a zero-th order predictor, i.e., if the point from the last value t_k is used to initialize the corrector for the new value $t_{k+1} = t_k + \Delta t$. In contrast, in inexact floating point arithmetic, the convergence region can disappear, thus halting the path tracker. Short of this, an unacceptably slow linear rate of convergence might dominate, causing the step size to plummet. It can also happen that the corrector appears to converge, judging by a diminishing corrector step, but since the limit point is a zero of the inexactly evaluated function, it may be further than the desired tolerance away from the exact root. Consequently, success requires enough precision to produce a convergence region and to ensure that the limit point is within the prescribed tolerance of the exact root.

Due to these considerations, an analysis of Newton's method in floating point is of interest and will help us derive rules for setting the precision used in our path tracker. The following analysis resembles that of [23]. Let $F(z) : \mathbb{C}^n \to \mathbb{C}^n$ be continuously differentiable, and denote its Jacobian matrix of partial derivatives as $J(z)$. To solve $F(z) = 0$ by Newton's method given an initial guess z_0, one iteratively updates z_i, $i = 1, 2, \ldots$, as

$$\begin{aligned} &\text{Solve } J(z_i)\Delta z_i = -F(z_i) \text{ for } \Delta z_i, \\ &z_{i+1} = z_i + \Delta z_i. \end{aligned} \tag{1}$$

In the following analysis, we will assume a submultiplicative matrix norm, i.e., one that satisfies $\|AB\| \leq \|A\|\|B\|$. In particular, the 2-norm suffices. When we apply the formulas, we may save computation by using other norms with an adjustment factor. For example, for an $n \times n$ matrix A, $\|A\|_2 \leq \sqrt{n}\|A\|_p$ for $p = 1$ or $p = \infty$.

Suppose that we work in floating point with unit roundoff u. In other words, if we compute with a mantissa of b bits in binary or with P digits in decimal, $u = 2^{-b} = 10^{-P}$. We may consider evaluating the residuals $F(z_i)$ in higher precision, say $\bar{u} \leq u$ before rounding back to working precision u to compute the Newton correction. Let $\hat{F}(z)$ be the floating point output of the procedure that evaluates $F(z)$. We assume that there exists a function ψ depending on z, u, and $\bar{u}$ such that the error $e(z) = \hat{F}(z) - F(z)$ obeys

$$\|e(z)\| \leq u\|F(z)\| + \psi(z, u, \bar{u}). \tag{2}$$

By definition, at a solution point z_*, we have $F(z_*) = 0$, so it is clear that the function ψ drives the final error. To determine ψ, one must examine the function F and the program that implements it. We will give approximations to ψ later for the systems we treat.

In solving Eq. 1 for the correction Δz_i, there is error in evaluating $J(z_i)$ and in solving the linear system. Both errors can be absorbed into an error matrix E_i such that the computed correction is

$$\Delta z_i = (J(z_i) + E_i)^{-1}(F(z_i) + e(z_i)). \tag{3}$$

We assume this error is bounded by

$$\|E_i\| \leq \mathcal{E}\left(u\|J(z_i)\| + \phi(z_i, u)\right), \tag{4}$$

for some constant $\mathcal{E} > 1$ and positive function ϕ. We expect the first term because of roundoff of the Jacobian, whereas ϕ accounts for errors in evaluating J that do not vanish even when J does. The constant $\mathcal{E}$ accounts for the subsequent growth in the error during the linear solve.

For simplicity of notation, let $v = z_i$ be the current guess, $\bar{v} = z_{i+1}$ the new guess after a single iteration, and let v_* be the solution point near v. Also, let's use the shorthand notations $F = F(v)$, $J = J(v)$, $J_* = J(v_*)$, $\Delta = \|v - v_*\|$ and $\bar{\Delta} = \|\bar{v} - v_*\|$. In the next paragraph, we will establish a bound on $\bar{\Delta}$ in terms of Δ. Whenever $\bar{\Delta} < \Delta$, the Newton step successfully reduces the error in the estimate of the root v_*.

Since $F(v_*) = 0$, the Taylor series of $F(z)$ at v_* gives

$$F(z) = J_* \cdot (z - v_*) + H.O.T.$$

where the higher order terms, $H.O.T.$, are quadratic or higher in $z - v_*$. Similarly,

$$J(z) = J_* + H.O.T.$$

where the higher order terms are linear in $z - v_*$. Consequently, in a ball $B = \{z : \|z - v_*\| \leq R\}$ centered at v_* with $v \in B$, there exist positive constants α and β such that

$$\|F(z)\| \leq \|J_*\|\|z - v_*\| + \alpha\|z - v_*\|^2, \tag{5}$$

$$\|F(z) - J_*(z - v_*)\| \leq \alpha\|z - v_*\|^2 \tag{6}$$

$$\|J_*\| \leq \|J\| + \beta\|z - v_*\|, \qquad \|J - J_*\| \leq \beta\|z - v_*\|. \tag{7}$$

From these, we may conclude that

$$\|F(z)\| \leq \|J\|\Delta + (\alpha + \beta)\Delta^2. \tag{8}$$

In Newton's method, we solve

$$(J + E)d = -(F(v) + e) \tag{9}$$

for d and take the step

$$\bar{v} = v + d + \varepsilon, \tag{10}$$

where ε is the error in forming the sum. The standard model of round-off error in floating point addition [27] gives

$$\|\varepsilon\| \leq u(\|v\| + \|d\|) \leq u(\Delta + \|v_*\| + \|d\|), \tag{11}$$

so subtracting v_* from both sides of Eq. 10, we have

$$\bar{\Delta} \leq \|v - v_* + d\| + u(\Delta + \|v_*\| + \|d\|). \tag{12}$$

If J is nonsingular and $\|J^{-1}\|\|E\| < 1$, then $(J + E)$ is nonsingular, the Newton step is well defined, and

$$\|(J + E)^{-1}\| \leq K\|J^{-1}\|, \qquad K = \frac{1}{1 - \|J^{-1}\|\|E\|}. \tag{13}$$

Accordingly, from Eq. 9 we have

$$\|d\| \leq K\|J^{-1}\|(\|F\| + \|e\|). \tag{14}$$

Also, after adding $(J + E)(v - v_*)$ to both sides of Eq. 9 and simplifying using Eqs. 5–8,13, we have

$$\|v - v_* + d\| \leq K\|J^{-1}\|(\|E\|\Delta + (\alpha + \beta)\Delta^2 + \|e\|). \tag{15}$$

Substituting from Eqs. 14,15 into Eq. 12 and using Eqs. 2,4,8, we obtain the bound

$$\begin{aligned}\bar{\Delta} \leq &K\|J^{-1}\|(1 + u)^2(\alpha + \beta)\Delta^2 + \\ &\left(K\|J^{-1}\|\left[(2 + \mathcal{E} + u)u\|J\| + \mathcal{E}\phi\right] + u\right)\Delta + K\|J^{-1}\|(1 + u)\psi + u\|v_*\|.\end{aligned} \tag{16}$$

This relation holds as long as $\|J^{-1}\|\|E\| < 1$, so that the linear solve in the Newton step is well defined, and v is in the ball B, so that Eqs. (5–7) hold.

If $\bar{\Delta} < \Delta$, the Newton step reduces the error in the approximation of the root. In exact arithmetic, we have $u = \phi = \psi = 0$ and $K = 1$, so $\bar{\Delta} \leq \|J^{-1}\|(\alpha+\beta)\Delta^2$. The error contracts if the initial guess is accurate enough so that $\Delta < 1/(\|J^{-1}\|(\alpha+\beta))$. If we also have $\Delta < 1/(\|J_*^{-1}\|(\alpha+\beta))$, it is clear that all subsequent iterates are nonsingular and contractive, from which one has the well-known result that Newton's method converges quadratically to a nonsingular solution for a sufficiently accurate initial guess. One sees that the more singular the Jacobian is at the root, the slower the convergence and the smaller the convergence zone.

In floating point arithmetic, we cannot expect the error to converge to zero. From Eq. 16, one may expect the error to contract until

$$\Delta \approx (1+u)K\|J^{-1}\|\psi + u\|v_*\| \approx K\|J_*^{-1}\|\psi + u\|v_*\|. \tag{17}$$

The second term is the error inherent in representing v_* in floating point. The first term depends on the accuracy, ψ, with which the function is evaluated. This can be reduced by using higher precision, $\bar{u}$, in the function evaluation per Eq. 2. The precision of the Jacobian and the linear solve do not affect the final error.

On the other hand, the precision of the Jacobian and the linear solve do affect convergence. Without enough precision, $\|J^{-1}\|\|E\|$ may approach or surpass 1, which means that the linear solve may fail due to singularity or may yield such an inaccurate step that the error diverges. Notice that $\|J^{-1}\|\|E\| = u\|J^{-1}\|\|J\| + \|J^{-1}\|\phi = u\kappa + \|J^{-1}\|\phi$, where $\kappa = \mathrm{cond}(J)$. The first term, $u\kappa$, reflects the well-known result that in solving a linear system, floating point roundoff is magnified by the condition number of the matrix.

3 Step length control and failure

To produce an improved path tracking algorithm, it is useful to first examine a standard predictor/corrector algorithm to see why adaptive step size control generally succeeds when conditioning is mild and why it may fail when conditioning is adverse.

A simple and effective approach for step-size control is to adjust the step size up or down according to the success or failure of a complete prediction/correction cycle. Suppose the homotopy function $H(z,t) = 0$ defines a one-dimensional nonsingular path $z(t)$. We are given a start point approximately on the path, $z_0 \approx z(t_0)$, an ending value of t, and a tolerance to which points on the path are to be found. Then, in brief, a predictor/corrector path tracker with adaptive step size control may be constructed as follows.

Initialize Select: an initial step size, s; the number of corrector iterations allowed per step, $N \geq 1$; the step adjustment factor, $a \in (0,1)$; the step expansion integer, $M \geq 1$; and a minimum step size $s_{\min}$.

Predict Estimate a new point near the path whose distance from the current point is the step size s.

Correct Iteratively improve the new path point, constraining its distance from the prior path point. Allow at most N iterations to reach the specified tolerance.

On success If the tolerance is achieved:

- Update the current path point to be the newly found path point.
- If we have reached the final value of t, exit with *success*.
- If there have been M successes in a row, expand the step size to $s = s/a$.

On failure If the tolerance is not achieved:

- Cut the step size to $s = as$.
- If $s < s_{\min}$, exit with *failure*.

Loop Go back to **Predict**.

The key is to allow only a small number of iterations in the corrector, typically only $N = 2$ or $N = 3$. This forces the prediction to stay within a good convergence region surrounding the path. If a large number of iterations is allowed, a bad prediction might ultimately converge, but it may wander first and become attracted to a completely different path in the homotopy. Keeping N small, the step-size adaptation slows down to negotiate sharp turns in the path and accelerates whenever the path is relatively straight. Properly implemented, this results in a robust and efficient path tracking algorithm.

We can be a bit more precise. Let us do so by specifically considering an Euler predictor with a Newton corrector. Both of these derive from the linearized local model of the path. The Taylor series at (z_1, t_1) is

$$H(z_1+\Delta z, t_1+\Delta t) = H(z_1,t_1) + \frac{\partial H}{\partial z}(z_1,t_1)\Delta z + \frac{\partial H}{\partial t}(z_1,t_1)\Delta t + H.O.T. \quad (18)$$

where the higher order terms, $H.O.T.$, are quadratic or higher in $(\Delta z, \Delta t)$. Ignoring the higher order terms and setting $H(z_1 + \Delta z, t_1 + \Delta t) = 0$, one has the basic Euler predictor and Newton corrector relations. These are a system of n equations in $n+1$ unknowns; as long as the combined matrix $[\partial H/\partial z \; \partial H/\partial t]$ is rank n, there is a well-defined tangent direction and tracking may proceed. The predictor adds a constraint on the length of the step along the tangent, whereas corrector steps are constrained to move transverse to the tangent. The extra constraints are particularly simple in the case where $\partial H/\partial z$ is rank n, for then the path progresses monotonically in t, and the step can be controlled via the advance of t. Accordingly, one has a linear system to be solved for Δz:

$$\left[\frac{\partial H}{\partial z}(z_1, t_1)\right] \Delta z = -\left(H(z_1, t_1) + \frac{\partial H}{\partial t}(z_1, t_1)\Delta t\right). \tag{19}$$

For prediction, we set $\Delta t = s$, the current step size, and for correction, we set $\Delta t = 0$.

Since the neglected terms are quadratic, the prediction error is order $\mathcal{O}(s^2)$. Thus, in the case of a failed step, cutting the step size from s to as reduces the prediction error by a factor of a^2. In this way, cuts in the step size quickly reduce the prediction error until it is within the convergence region of the corrector. With a kth order predictor, the prediction error scales as a^{k+1}, potentially allowing larger step sizes. In any case, the adaptive approach quickly settles to a step size s just small enough so that the corrector converges, while the next larger step of s/a fails. With $a = 1/2$ and $M = 5$, the step size adapts to within a factor of 2 of its optimum, with an approximate overhead of 20% spent checking if a larger step size is feasible.

Failure of path tracking with an adaptive step size can be understood from the discussion of Newton's method in § 2. For small enough initial error and exact arithmetic, the Newton corrector gives quadratic convergence to a nonsingular root. Near a singularity, $\|J_*^{-1}\|$ is large, which can lead to a small quadratic convergence zone and a slower rate of quadratic convergence. Inexact arithmetic can further shrink the convergence zone, degrade the convergence rate from quadratic to linear, and introduce error into the final answer. From these considerations, we see that there are two ways for the adaptive step size path tracker to halt prematurely near a singularity.

1. The predictor is limited to a tiny step size to keep the initial guess within the convergence zone of the corrector. If this is too small, we may exceed the total computational cost allotted for the path.

2. The path may approach a point where the final error of the corrector is larger than the requested path tracking tolerance.

The first mode of failure can occur even with infinite precision, but degradation of the convergence properties with too low a precision increases the occurrence of this failure. The second mode of failure is entirely a consequence of lack of precision. By allocating enough precision, we can eliminate the second mode of failure and reduce the occurrence of the first mode. It is important to note that in some applications there is flexibility in the definition of the homotopy, which can be used to enlarge convergence zones and thereby speed up path tracking. For example, re-scaling of the equations and variables can sometimes help. However, such maneuvers are beyond the scope of this paper, which concentrates only on tracking the path of a given homotopy.

4 Adaptive precision

The use of high precision can largely eliminate both types of path tracking failure identified above. However, high precision arithmetic is expensive, so it must be employed judiciously. One might be tempted to ratchet precision up or down in response to step failures as in the adaptive step size algorithm. This presents the difficulty that there is just one stimulus, step failure, and two possible responses, cut the step size or increase precision. In the following paragraphs, we outline two possible approaches for adapting both step size and precision.

Both our "reactive" adaptation algorithm and our "proactive" one are encompassed in the same flowchart, Figure 4.1. One may see that the heart of the method is still the adaptive step size tracker described in § 3. In the diagram, P is the number of digits of precision and $P_{\max}$ is the largest precision we allow. The step size of the tracker is Δt, and $\varepsilon(P)$ is a lower limit for it, depending on P, as discussed further below. The magnitude of the most recent Newton step in the corrector is $\|d\|$ and $10^{-\tau}$ is the accuracy to which we wish to track the path. The final value of t to which we wish to track is $t_{\rm f}$. Limits on P and on the total number of steps guarantee that the algorithm always terminates, declaring failure if too much computation has occurred before reaching $t_{\rm f}$.

The algorithm as presented in Figure 4.1 assumes that the final value $t_{\rm f}$ can be reached without encountering an exact singularity. The homotopies used for solving polynomial systems are constructed such that, with probability one, no singularities will occur until possibly at the end, say $t = t^*$, where the homotopy function $H(z, t^*) = F(z)$ is the target system. In this situation, one should not use the corrector at $t = t^*$. Instead, one may predict the endpoint from values of t near t^* and accept the predictions as accurate when two successive ones agree to the desired precision. Both the prediction to t^* and the prediction to the next point on the path near t^* benefit from the use of a more sophisticated predictor than Euler's method, accounting for the possible existence of a singularity at the endpoint. For example, endgames that estimate the winding number of the root and use it to compute a fractional power series can be very effective [17, 19]. Such endgames still use a path tracker to approach t^*. In this paper, the endgame itself is out of scope: we discuss only the tracking algorithm and we consider the final value $t_{\rm f} \neq t^*$ to be the last point on the path required by the endgame.

4.1 Reactive Adaptation

Reactive adaptation changes precision in response to the observed behavior of the tracking process. In particular, if path tracking fails at a given precision, one may try again at a higher precision. The simplest approach is to run the entire path in a fixed precision with adaptive re-runs. That is, if the path tracking fails, one re-runs it in successively higher precision until the whole path is tracked successfully or until limits in computing resources force termination. The advantage of this approach is that adaptation is completely external to the core path tracking routine. Thus, this strategy can be applied to any path tracker that enables requests for higher precision. For example, in the polynomial domain, the package PHC [24] offers multiple precision, although the precision must be set when calling the program.

The path re-run strategy has two main disadvantages. First, when too low a precision is specified, the tracker wastes computation near the point of failure before giving up and initiating a re-run in higher precision. Second, and more important, the whole path is computed in high precision when it may be needed only in a small section of the path, often near the end in the approach to a singular solution.

A slightly more sophisticated treatment avoids re-computing the segment of the path leading up to the failure point. Upon failure at one precision, higher precision is initiated from the last successfully computed point on the path. In this approach, the adaptive step size module may still waste time cutting the step size before finally triggering a needed increase in precision. Moreover, the minimum step size needs to be set appropriately so that the precision is increased correctly. Once the precision has been increased, it may take many steps before the step size is increased again to a reasonable size. There is also a theoretical possibility that even though more precision is needed to accurately track the path, the tracking module may wander instead of failing, jumping to another solution path in the homotopy.

Figure 4.1 presents a flowchart that encompasses both the reactive adaptation method just described and the proactive one of the following section. For the reactive method, the checks labeled A, B, and C are skipped. There are three branches where the reactive method increases the precision, P: the prediction or correction steps may fail or the step size, Δt, may become too small. Failure in the predictor or corrector steps means that the linear solve of Eq. 19 has aborted early due to singularity. Using the magnitude of the largest entry in J as $\|J\|$, Gaussian elimination with row pivoting may declare such a failure when the magnitude of the largest available pivot is smaller than $u\mathcal{E}\|J\|$, for then the answer is meaningless.

The check for a step size that is too small is to compare Δt to $\varepsilon(P)$, a predetermined minimum step size for each level of precision. It is important to note that making $\varepsilon(P)$ too large may cause a runaway condition where precision is rapidly increased to its maximum, thus causing the path to declare failure. Setting it too small will result in temporary stalling of the path, wasting computation until the precision increase is finally triggered and wasting more computation afterwards as the algorithm has by then adopted an unnecessarily small step size. Our settings for $\varepsilon(P)$ are given in § 5.1. We determined by experimentation that values of $\varepsilon(P) = 10^{a-P}$ for $2 \leq a \leq 4$ seem to work well.

A path may have a difficult spot somewhere other than near the end of the path. Approaching this spot, the step size will decrease and the precision may increase. After passing the difficulty, the algorithm needs a means of readjusting back to a larger step size and smaller precision. For the box "Consider increase

Δt," we use the criterion of the classic step-size control method of § 3, that is, we increase Δt if several successive steps have succeeded. It is not clear at this time how to implement decreases in precision in a purely reactive manner. In contrast, the proactive approach of the following section gives a clear signal for when precision can be decreased.

4.2 Proactive adaptation

Instead of waiting for the adaptive step size method to fail before initiating higher precision, we propose to proactively monitor the conditioning of the homotopy to judge the level of precision needed at each step. In this way, the computational burden of higher precision is incurred only as needed, adjusting up and down as the tracker proceeds. If done properly, this avoids the computational cost incurred in the reactive approach of unnecessarily cutting the step size when lack of precision is the root cause. Of course, monitoring the numerical properties of the homotopy introduces a new computational burden of its own. We will explore whether the trade-off is advantageous with respect to total computation, but even if the trade-off is in favor of the reactive approach, the proactive approach could still be preferable to avoid wandering due to lack of precision.

To decide how much precision is needed, we turn to the analysis of Newton's method from § 2. We wish to ensure that the achievable accuracy is within the specified tolerance and that convergence is fast enough.

In what follows, we need to evaluate $\|J\|$ and $\|J^{-1}\|$. These do not need to be very accurate, as we will always include safety margins in the formulas that use them. $\|J\|$ is readily available in the max norm, i.e., the matrix norm induced by the infinity norm for vectors. $\|J^{-1}\|$ is more difficult, as we do not wish to compute the full inverse of the matrix. This issue has been widely studied in terms of estimating the condition number $\kappa = \|J\|\|J^{-1}\|$. A relatively inexpensive method, suggested in [26] and elsewhere, is to choose a unit vector b at random and solve $Jy = b$ for y. Then, we use the estimate $\|J^{-1}\| \approx \|y\|$. Although this underestimates $\|J^{-1}\|$, tests of matrices up to size 10×10 show the approximation to be reliably within a factor of 10 of the true value, which is easily absorbed into our safety margins.

One requirement is that $\|J^{-1}\|\|E\|$ should be small enough to ensure that the error-perturbed Jacobian is nonsingular. Minimally, we require $\|J^{-1}\|\|E\| < 1$, but by requiring it to be a bit smaller, say $\|J^{-1}\|\|E\| < 10^{-\sigma_1}$ for some $\sigma_1 \geq 1$, we force $K \approx 1$, as defined in Eq. 13. This removes the growth of K as one possible source of failure. Suppose that the error function ϕ in Eq. 4 is of the form $\phi = \Phi u$. Then, our first rule is to require

$$\|J^{-1}\|\mathcal{E}\left(\|J\| + \Phi\right) u < 10^{-\sigma_1}. \tag{20}$$

Using P decimal digits of arithmetic results in precision $u = 10^{-P}$, so we may

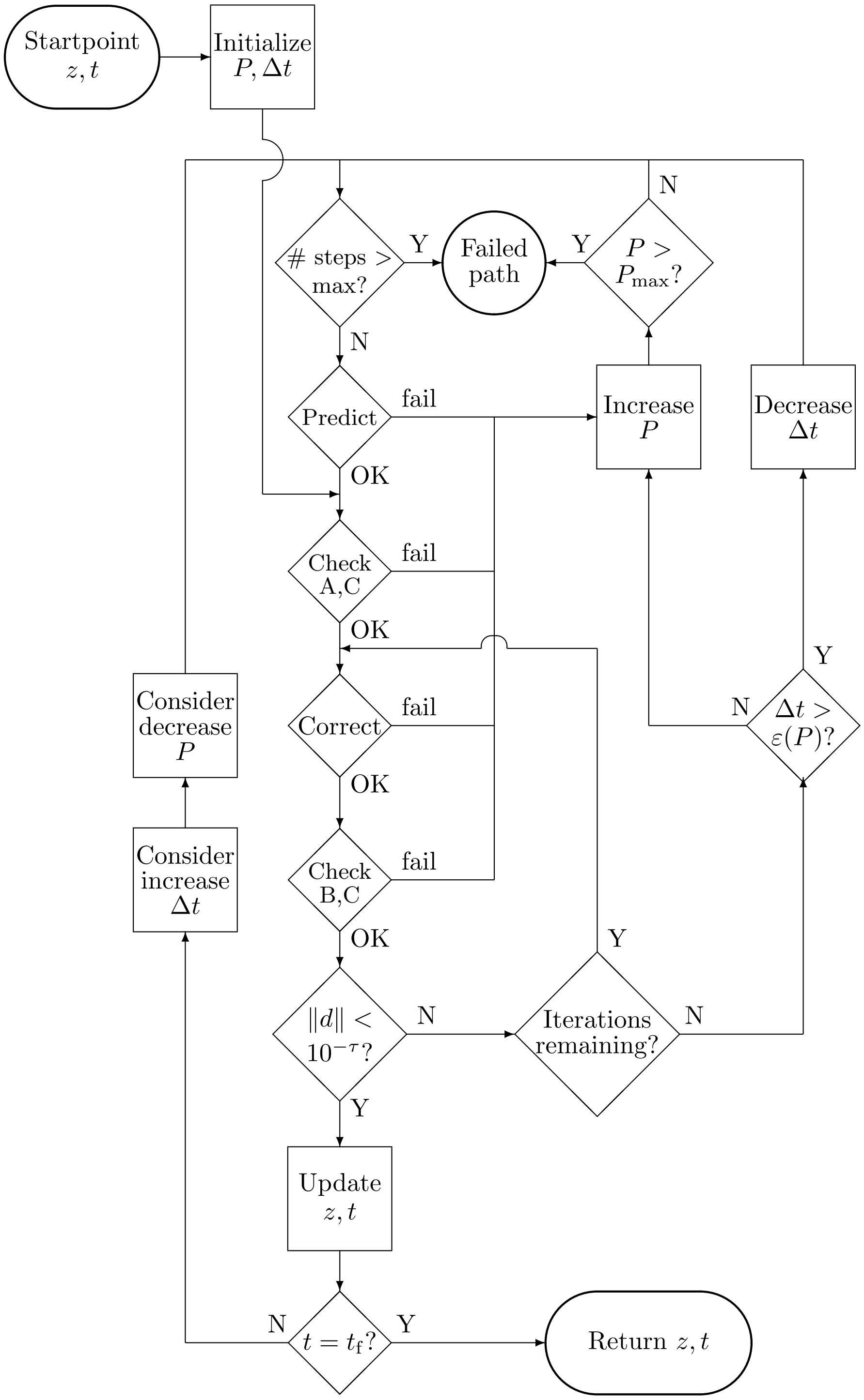

Figure 1: Adaptive precision path tracker with proactive precision checks. The purely reactive algorithm skips all checks of conditions A, B, and C. The purely proactive approach sets $\varepsilon(P) \equiv 0$.

restate this rule as

$$P > \sigma_1 + \log_{10}[\|J^{-1}\|\mathcal{E}\,(\|J\| + \Phi)]. \qquad (\mathbf{A})$$

A second requirement is that the corrector must converge within N iterations, where we keep N small as in the usual adaptive step size algorithm, typically 2 or 3. Let us say that the tolerance for convergence is $\Delta = \|v - v_*\| < 10^{-\tau}$. Recall that in each step of Newton's method, we compute d and take the step $\bar{v} = v + d$. The best estimate available of the accuracy is $\Delta \approx \|d\|$, so we declare success when $\|d\| < 10^{-\tau}$. Suppose that after $i < N$ iterations this is not yet satisfied. We still have $N - i$ iterations to meet the tolerance, and we would like to be sure that a lack of precision does not prevent success. Pessimistically, we assume that the linear factor in Δ in Eq. 16 dominates the quadratic one and that the rate of convergence does not improve with subsequent iterations. We force $K \approx 1$, and we have $u \ll 1$. Including the same safety margin as before, 10^{σ_1}, the requirement becomes

$$\left[10^{\sigma_1}\|J^{-1}\|\,((2 + \mathcal{E})u\|J\| + \mathcal{E}\phi) + u\right]^{N-i}\|d\| < 10^{-\tau}. \qquad (22)$$

As before, let's assume $\phi = \Phi u$. Taking logarithms, the number of decimal digits of precision must satisfy

$$P > \sigma_1 + \log_{10}\left[\|J^{-1}\|\,((2 + \mathcal{E})\|J\| + \mathcal{E}\Phi) + 1\right] + (\tau + \log_{10}\|d\|)/(N - i). \quad (\mathbf{B})$$

Since we only apply this formula when the tolerance is not yet satisfied, we have $\|d\| > 10^{-\tau}$, or equivalently, $\tau + \log_{10}\|d\| > 0$. This implies that between corrector iterations, requirement **B** is always more stringent than Eq. **A**. However, we still use Eq. **A** outside the corrector, because $\|d\|$ is not then available.

Our third requirement is that the precision must be high enough to ensure that the final accuracy of the corrector is within the tolerance at full convergence. For this, Eq. 17 is binding, so including a safety margin of $10^{-\sigma_2}$ and using the norm of the current approximate solution, $\|v\|$, as the best available estimate of $\|v_*\|$, we require

$$\|J^{-1}\|\psi + u\|v\| < 10^{-\tau-\sigma_2}. \qquad (24)$$

Suppose the error in evaluating the homotopy function is given by $\psi = \Psi\bar{u}$. If the function is evaluated in the same precision as the rest of the calculations, i.e., $\bar{u} = u$, we have the requirement

$$P > \sigma_2 + \tau + \log_{10}(\|J^{-1}\|\Psi + \|v\|). \qquad (\mathbf{C})$$

If instead we evaluate the function to higher precision, say $\bar{u} = 10^{-P'} < u = 10^{-P}$, we have the dual criteria

$$P > \sigma_2 + \tau + \log_{10}\|v\|, \qquad P' > \sigma_2 + \tau + \log_{10}\|J^{-1}\| + \log_{10}\Psi. \qquad (\mathbf{C'})$$

The effect of adding the two errors is absorbed into the safety factor σ_2.

Conditions A, B, and C (or C′) allow one to adjust the precision as necessary without waiting for the adaptive step size to fail. If necessary, the precision can even be increased between corrector iterations. An algorithm using these criteria is described by the flowchart in Figure 4.1. With the proactive checks in place, it is no longer useful to trigger reactive increases in precision due a to small step size. Accordingly, we may remove the step-size check, which is equivalent to setting $\varepsilon(P) \equiv 0$.

Unlike in the reactive method, the proactive method has a clear signal for when it is safe to decrease precision after passing a locally ill-conditioned point on the path. In the "consider decreasing P" box of Figure 4.1, one may safely do so if the checks of criteria A, B, and C in the last tracking step indicate that a lower precision would have sufficed.

4.3 Error estimates

To use the foregoing procedures, we need the function evaluation error, ψ, and the errors contributing to E, namely, $\mathcal{E}$ and ϕ. There is a trade-off between using rigorously safe bounds for highest reliability or using less stringent figures reflecting typical behavior to avoid the overuse of high precision. Rough figures are acceptable as this is just a means of setting the precision.

While ψ and ϕ concern the errors in evaluating the function and its Jacobian, the factor $\mathcal{E}$ concerns the stability of the linear solve. Round-off errors can accumulate through each stage of elimination. When Gaussian elimination with partial pivoting is used, the worst-case error bound grows as $\mathcal{E} = 2^n$ for solving an $n \times n$ system [3]. However, as indicated in [3], $\mathcal{E}$ rarely exceeds n with the average case around $n^{\frac{2}{3}}$ or $n^{\frac{1}{2}}$. Setting $\mathcal{E} = n^2$ should therefore be sufficient for almost all cases.

It is important to note that the error in the function depends on the error in its parameters. For example, consider the simple function $f(x) = x^2 - 1/3$. If this is rounded off to $g(x) = x^2 - 0.333$ before we use high precision to solve $g(x) = 0$, we will obtain an accurate value of $\sqrt{0.333}$ but we will never get an accurate value of $1/\sqrt{3}$. Although this is an obvious observation, it can easily be forgotten in passing a homotopy function from some application to the adaptive precision path tracking algorithm. If coefficients in the function are frozen at fixed precision, the algorithm tracks the solutions of the frozen function, not the exact problem that was intended. Whether the difference is significant depends on the nature of the application and the sensitivity of the function.

In the remainder of this section, we discuss how to obtain error estimates for several classes of functions.

4.3.1 General functions

A user of the path tracker will not usually wish to expend a lot of effort in developing error bounds, so an automated error analysis is desirable. A rigorous and automated way of establishing error bounds for almost any continuous function is to use interval arithmetic. Software packages such as IntBis [7], IntLab [21], Alias [13], and MPFI [20] provide interval evaluation for any real-valued function composed from a wide range of continuous elementary functions and arithmetic operations. Among these packages IntLab and MPFI currently support multiple precision, although IntLab's multiple precision is slow while MPFI does not support complex arithmetic. While a relatively efficient complex multiprecision interval arithmetic could be built on top of MPFI, it is beyond our scope to pursue this.

One could go further and employ interval techniques to obtain a path tracker with fully rigorous step-size control, as in [8], guaranteeing that path crossing never occurs. However, this can be expensive, due partially to the cost of interval arithmetic but more significantly due to the cost of over-conservative error bounds, which slow the algorithm's progress by driving the step size smaller than necessary. Still, when rigorous results are desired, it may be worth the computational cost. The method of [8] does not explicitly include adaptive precision, so something along the lines discussed here could be useful in modifying that approach.

Since our main area of interest is in systems of polynomial equations, we devote the rest of this article to that important subclass.

4.3.2 Straight-line polynomials

Suppose $f : \mathbb{C}^n \to \mathbb{C}$ is a polynomial whose terms are indexed by the set $\mathcal{I}$, that is,

$$f(z) = f(z_1, \ldots, z_n) = \sum_{i \in \mathcal{I}} c_i z_1^{d_{1i}} \cdots z_n^{d_{ni}}. \tag{27}$$

We wish to estimate the error in evaluating $f(z)$ using floating-point arithmetic with unit roundoff $\bar{u}$. Evaluating the fully expanded form of a polynomial, that is, evaluating each term of Eq. 27 and then summing, is not the most efficient approach in terms of operation counts. For example, to evaluate a dense polynomial in one variable, e.g., $f(x) = c_0x^3 + c_1x^2 + c_2x + c_3$, Horner's rule, $f(x) = (((c_0x) + c_1)x + c_2)x + c_3$, uses the fewest multiplications. See [6, §4.6] for more discussion. Different evaluation procedures can result in different floating point round-off errors as well, for example, in floating-point arithmetic distributivity, $a(b + c) = ab + ac$, and associativity, $(a + b) + c = a + (b + c)$, do not hold exactly [6, §4.2].

Suppose the program to evaluate $f(z)$ has been parsed into a straight-line program, that is, a sequence of unary and binary operations free of branches or loops. Then, we may approximate errors by accumulating their effects across successive operations. Interval analysis by circular arithmetic [4] is a convenient method for this. Let $\langle a, r\rangle$, $a \in \mathbb{C}$, $r \in \mathbb{R}^+$, denote the closed disk in the complex plane with center a and radius r. For two disks A_1, A_2 and an operation "$\circ$," let

$$A_1 \circ A_2 = \{x_1 \circ x_2 | x_1 \in A_1, x_2 \in A_2\}.$$

We are interested in the cases when "$\circ$" is addition or multiplication. Then, one may confirm the inclusions [4]

$$\begin{aligned} \langle a_1, r_1 \rangle + \langle a_2, r_2 \rangle &= \langle a_1 + a_2, r_1 + r_2 \rangle, \\ \langle a_1, r_1 \rangle \times \langle a_2, r_2 \rangle &\subset \langle a_1 a_2, |a_1| r_2 + |a_2| r_1 + r_1 r_2 \rangle. \end{aligned} \tag{28}$$

This shows that if a_1 and a_2 are the computed values of the operands whose exact values lie in the respective disks, then the true sum or product must lie within the disks on the right-hand side of these relations. However, when we evaluate $a_1 \circ a_2$ in floating point, round-off error must be taken into consideration. By appropriately increasing the radius of the output disk, we can guarantee that it still contains the exact answer. Since the roundoff for any point in the disk is bounded by $|x|u$ where $|x|$ is the largest magnitude of any point in the disk, we have

$$\begin{aligned} \langle a_1, r_1 \rangle + \langle a_2, r_2 \rangle &\subset \langle a_1 + a_2, r_1 + r_2 + (|a_1 + a_2| + r_1 + r_2)u \rangle, \\ \langle a_1, r_1 \rangle \times \langle a_2, r_2 \rangle &\subset \langle a_1 a_2, \rho + (\rho + |a_1 a_2|)u \rangle, \\ &\text{where } \rho = |a_1| r_2 + |a_2| r_1 + r_1 r_2. \end{aligned} \tag{29}$$

These relations can be used to bound the accumulated floating-point error in the straight-line evaluation of a polynomial. In computing the radius in either of these formulae in floating point, one should consistently use upward rounding to be sure of obtaining inclusion.

To apply proactive adaptation based on criteria A, B, and C (or C′), we would like to develop a linearized model of how the error depends on precision. That is, we would like to express the evaluation error $\psi(z,u)$ as $\psi(z,u) \leq \Psi(z)u + H.O.T.$, where the higher order terms depend on quadratic and higher powers of u. To this end, suppose that the error radii of all the initial data are expressed in terms of the unit roundoff. For example, the exact number $2+3i$ is $\langle 2+3i, 0u \rangle$ whereas the floating-point representation $(1/3)_{\mathrm{fp}}$ of $1/3$ is inexact: $(1/3) \in \langle (1/3)_{\mathrm{fp}}, (1/3)_{\mathrm{fp}} u \rangle$. Applying Eq. 29 to each operation of the straight-line program and keeping u as an indeterminant, the result after each operation is a disk of the form $\langle a, r(u) \rangle$, where $a \in \mathbb{C}$ and $r(u)$ is a polynomial in u with nonnegative real coefficients. At the end of the computation, one obtains a disk $f(z) \in \langle \hat{f}(z), r_1(z)u + r_2(z)u^2 + H.O.T. \rangle$, where $\hat{f}(z)$ is the floating point evaluation of $f(z)$, and r_1 and r_2 are numeric coefficients. For a given unit roundoff u_0, if $r_2 u_0 << r_1$, then the linear term dominates the quadratic one and presumably all higher powers of u as well. Then, we may safely use $\psi(z,u) \approx r_1(z)u$, i.e., $\Psi = r_1(z)$, in the adaptive precision routines.

The foregoing methodology computes the function value and the error bound simultaneously. Unfortunately, the error bound is at least as expensive to compute as the function value, which may be an unacceptable computational burden. A less burdensome approach is to compute an error bound over a region and update this only when the path tracking algorithm leaves that region. To do so, we again just apply Eq. 29, but we begin with a disk for each coordinate z_i of z, say $z_i \in \langle a_i, \rho_i \rangle$, $\rho_i \in \mathbb{R}^+$, $i = 1, \ldots, n$. The method proceeds as above, with the only difference being that the radii of the results of each operation have a constant coefficient. At the end, we have a disk $\langle \hat{f}(z), r_0(z,\rho) + r_1(z,\rho)u + r_2(z,\rho)u^2 + H.O.T. \rangle$ that contains all possible values that the polynomial may take on the region defined by the initial disks.

Throughout this section, we have spoken only of how to bound the roundoff error in function evaluation. It should be clear that the same approach applies to bound the errors in the entries of the Jacobian matrix, since each of these is also a polynomial. After estimating the error in each entry, one may estimate the error $\phi(z,u)$ in Eq. 4 using the relations $\|A\|_2 \leq \sqrt{n}\,\|A\|_p$ for $p = 1$ or $p = \infty$.

4.3.3 Approximations for expanded polynomials

To avoid program complexity and save computation time, it is preferable not to perform a complete interval analysis of errors. In many cases a rough analysis is sufficient and easily derived. This is indeed possible for the case of expanded polynomials.

Referring to Eq. 27, we denote the degree of the i-th term as $d_i = \sum_{j=1}^{n} d_{ji}$. If the coefficient c_i is represented in floating point with unit roundoff $\bar{u}$, then the error in the floating-point evaluation of the i-th term is $\bar{u}(d_i+1)|c_i||z_1|^{d_{1i}} \cdots |z_n|^{d_{ni}}$, ignoring quadratic and higher terms in $\bar{u}$. (If c_i has an exact floating point representation, such as if it is an integer of magnitude smaller than $1/\bar{u}$, then a slightly smaller bound applies: $\bar{u}d_i|c_i||z_1|^{d_{1i}} \cdots |z_n|^{d_{ni}}$.) Summing the terms and rounding off to working precision u gives the error between the computed value $\hat{f}(z)$ and the true $f(z)$ of

$$e(z) = |\hat{f}(z) - f(z)| \leq u|f(z)| + \bar{u} \sum_{i \in \mathcal{I}} (d_i + 1)|c_i||z_1|^{d_{1i}} \cdots |z_n|^{d_{ni}}. \tag{30}$$

Thus, comparing to Eq. 2, we have

$$\psi(z, \bar{u}) = \bar{u} \sum_{i \in \mathcal{I}} (d_i + 1)|c_i||z_1|^{d_{1i}} \cdots |z_n|^{d_{ni}}. \tag{31}$$

Equation 31 is as computationally expensive to evaluate as $f(z)$. However, this cost is usually small compared to evaluating the Jacobian matrix and solving for the Newton correction. Even so, one might prefer to replace $\psi(z,\bar{u})$ with an upper bound that is cheaper to compute. Suppose $D = \max_{i \in \mathcal{I}} d_i$ and $|z|_\infty = \max_{j \in [1,n]} |z_j|$. Then, clearly

$$\psi(z, \bar{u}) \leq \bar{u}|z|_\infty^D \sum_{i \in \mathcal{I}} (d_i + 1)|c_i| = \bar{u}|z|_\infty^D S, \tag{32}$$

where S is the sum indicated in the intermediate expression. Since S does not depend on z, it can be computed once at the beginning of the path tracking algorithm.

For a system of polynomials, $F : \mathbb{C}^n \to \mathbb{C}^m$, we obtain an error estimate for each polynomial in the system, i.e., $\psi_j(z, \bar{u})$, $j = 1, \ldots, m$. Then, the 2-norm error of the system is simply $\psi(z, \bar{u}) = \|[\psi_j(z, \bar{u})]\|_2$, which if we use Eq. 32 becomes

$$\psi(z, \bar{u}) = \bar{u}|z|_\infty^D (S_1^2 + \cdots + S_m^2)^{1/2},$$

where S_j is the sum appearing in Eq. 32 applied to the j-th polynomial.

At first glance, it may seem that errors could be reduced by simply scaling the functions, and thereby scaling their coefficients, by some small factor. But $\|J_*^{-1}\|$ will scale oppositely, so the error predicted by Eq. 17 is unchanged.

4.3.4 Homogeneous polynomials

Homogeneous polynomials, in which every term has the same degree $D = d_i$, $i \in \mathcal{I}$, are a special case. Some problems are naturally homogeneous, but for those that are not, it is often convenient and numerically advantageous to homogenize the equations before solving the system. Any inhomogeneous polynomial $f(z_1, \ldots, z_n)$ can be easily homogenized to obtain a related function $F(x_0, x_1, \ldots, x_n)$, with

$$F(1, x_1, \ldots, x_n) = f(x_1, \ldots, x_n).$$

If f is as in Eq. 27, then

$$F(x) = \sum_{i \in \mathcal{I}} c_i x_0^{D-d_i} x_1^{d_{1i}} \cdots x_n^{d_{ni}}, \tag{33}$$

where D and d_i are as above. Hence, for any solution z_* of $f(z) = 0$ there is a corresponding solution $x_* = (1, z_*)$ of $F(x) = 0$.

One advantage of homogenization is that we can re-scale any solution x_* of $F(x) = 0$ to make $\|x_*\| = 1$, which often helps numerical conditioning. More precisely, since $F(\lambda x) = \lambda^D F(x)$, we have that if $F(x) = 0$, then also $F(\lambda x) = 0$. Consequently, the solution set of a system of homogeneous polynomials can be said to lie in projective space $\mathbb{P}^n$, the set of lines through the origin in $\mathbb{C}^{n+1}$. (Similarly, the solutions of multihomogeneous polynomials lie in a product of projective spaces, see [22].) To numerically solve for a point $x_* \in \mathbb{P}^n$, we intersect the corresponding line in $\mathbb{C}^{n+1}$ with an inhomogeneous hyperplane $a^T x - 1 = 0$, where $a \in \mathbb{C}^{n+1}$ can be chosen arbitrarily excepting a such that $a^T x_* = 0$. In other words, $a^T x - 1 = 0$ defines a $\mathbb{C}^n$ patch on $\mathbb{P}^n$, and $a^T x = 0$ is the hyperplane at infinity for that patch. In practice, we may choose a at random, with a zero probability of picking it such that the one-real-dimensional homotopy path intersects the hyperplane at infinity. However, during path tracking, if $\|x\|$ should grow, one may reassign $a = x'/\|x\|_2$, where x' denotes the complex conjugate of x, so that the new representative of the same projective point is $x/\|x\|_2$. In this way, the magnitudes of both a and x can be kept near one throughout the homotopy without changing the underlying path in projective space.

Suppose we use homogenization to force each homogenized coordinate x_i, $i = 0, 1, \ldots, n$ to lie in a unit disk at the origin: $x_i \in \langle 0, 1\rangle$. Then, we may obtain in one initial computation an error bound that will hold throughout the entire path tracking procedure by applying the method of § 4.3.2. Similarly, since $\|x\|_\infty \leq 1$, Eq. 32 becomes

$$\psi(x, u) \leq u(D+1) \sum_{i \in \mathcal{I}} |c_i|, \tag{34}$$

or just $uD \sum_{i \in \mathcal{I}} |c_i|$ if the coefficients are exact. Similarly, the derivatives have an approximate error bound of

$$\phi(x, u) \approx uD(D+1) \sum_{i \in \mathcal{I}} |c_i|. \tag{35}$$

The downside is that this bound may be overly conservative.

The effect of homogenization on the interpretation of the desired tolerance should be noted. The homogeneous coordinates $x = [x_0, x_1, \ldots, x_n] \in \mathbb{P}^n$ correspond to dehomogenized coordinates $z = (z_1, \ldots, z_n) = (x_1/x_0, \ldots, x_n/x_0)$. Suppose each coordinate of x is computed to accuracy $\epsilon << 1$ and consider what this implies about the accuracy of z. Dehomogenization of the result approximates z_i as $\hat{z}_i = (x_i + \delta_i)/(x_0 + \delta_0)$, where $|\delta_i| \leq \epsilon$, $i = 0, \ldots, n$. If $\|z\|_\infty < 1$, our homogenization will compute it with $|x_0| \approx 1$ and so $|\hat{z}_i - z_i| \approx \delta_i + \delta_0 \leq 2\epsilon$. On the other hand, if $\|z\|_\infty > 1$, the largest coordinate of x, say x_k will be computed with magnitude 1, and dehomogenization gives $|\hat{z}_k - z_k| \approx |z_k|\delta_i + |z_k|^2\delta_0 \leq (|z_k| + |z_k|^2)\epsilon$. So small solutions are found to absolute error 2ϵ whereas large ones are found to absolute error $\epsilon\|z\|_\infty^2$. This implies a total loss of precision for $\|z\|_\infty \geq 1/\epsilon$, as one must expect since $[\epsilon, 1] \in \mathbb{P}$ and $[0, 1] \in \mathbb{P}$ become indistinguishable to accuracy ϵ, yet they dehomogenize to $1/\epsilon$ and ∞, respectively.

5 Computational results

This section contains a brief discussion of the implementation details for multiprecision arithmetic and for evaluating the rules for adapting precision. We then discuss the application of the adaptive precision methods to several polynomial systems.

5.1 Implementation details

Bertini [2] is a software package for computation in numerical algebraic geometry currently under development by the authors. Bertini is written in the C programming language and makes use of straight-line programs for the representation, evaluation, and differentiation of polynomials. All the examples discussed here were run on an Opteron 250 processor running Linux, using a version of Bertini similar to the beta release.

	IEEE double	MPFR						
bits	52	64	96	128	192	256	512	1024
decimal digits	16	19	28	38	57	77	154	308

Table 1: Number of digits for mantissas at selected levels of precision

P	52	64	96	128	160	192	224
$\varepsilon(P)$	10^{-14}	10^{-16}	10^{-25}	10^{-34}	10^{-43}	10^{-52}	10^{-61}

Table 2: Values of $\varepsilon(P)$ for reactive adaptive precision

The adaptation rules, A, B, and C (or C′), leave some implementation choices open. For the runs reported here, we chose to evaluate function residuals to the same precision as the computation of Newton corrections, so rule C applied, not rule C′. Also, in rules A and B, we chose to use $\mathcal{E} = n^2$, where n is the number of variables, which is somewhat conservative for typical cases but underestimates the worst pathological cases. (See section 4.3 for more on this issue.) The rules require formulas for evaluating the error bounds $\psi(x, u) = \Psi u$ and $\phi(x, u) = \Phi u$. These are problem dependent, so we report our choices for each of the example problems below. Also, since the size of the first step is problem and path dependent, Bertini automatically adjusts this based on the results of the Euler prediction of the first step rather than relying on the user to provide an appropriate size.

To adaptively change precision, Bertini relies on the open source MPFR library for multiprecision support. Bertini has data types and functions for double precision (based on the IEEE double standard) and higher precision (using MPFR). Although the program would be simpler if MPFR data types and functions were used exclusively, the standard double precision types and functions in C are more efficient, so Bertini uses these whenever the adaptation rules indicate that double precision is sufficient. Additional details regarding the use of multiple precision may be found on the Bertini website. Since the use of adaptive precision variables is highly implementation-specific, no other details are described here.

MPFR requires the addition of precision to the mantissa in packets of 32 bits. Since the discussion of the examples below involves both binary and decimal digits, Table 1 provides the conversion between the two for selected levels of precision.

The reactive adaptation method requires a minimum step size, $\varepsilon(P)$, for each level of precision, P. The values we used in our experiments are shown in Table 2.

5.2 Numerical error for a Wilkinson polynomial

Before testing adaptive precision in a path tracking application, let's consider Newton's method applied to a Wilkinson polynomial and compare its results to the predictions of our error analysis. The kth Wilkinson polynomial, $w_k(z)$, is defined as $w_k(z) = \prod_{i=1}^{k}(z-k)$, and we report here on the specific case of $k = 11$. To solve $w_{11}(z) = 0$ numerically, we use straight-line programs to evaluate $w_{11}(z)$ and $J(z) = w_{11}'(z)$ and apply Newton's method. Let us consider two different straight-line programs for $w_{11}(z)$:

- the product form $f_1(z) = (z-1)(z-2)\cdots(z-11)$ evaluated left to right as written,
- the expanded form $f_2(z) = z^{11} - 66z^{10} + 1925z^9 - \cdots - 150917976z^2 + 120543840z - 39916800$, where powers of z are formed by multiplication, then the terms are summed from left to right.

In the expanded form, the coefficient of largest magnitude is that of the quadratic term, which is large, but still small enough to be represented exactly in double precision. (For large enough k, an error analysis in double precision would have to include roundoff error in the coefficients.)

Consider Newton's method in the vicinity of the root $z = 7$. For $z \in \langle 7, .1\rangle$ with exact coefficients, error analysis of the product form gives

$$f_1(\langle 7, .1\rangle) \in \langle 0, 2682 + 52492u + 488015u^2\rangle$$

while the expanded form gives

$$f_2(\langle 7, .1\rangle) \in \langle 0, 1.2\times 10^{11} + 9.4\times 10^{12}u + 2.6\times 10^{13}u^2\rangle.$$

In both cases, the quadratic term in u is negligible for all u under consideration. Now, the Jacobian at $z = 7$ is $J(7) = 17280$, so Eq. 17 predicts that Newton's method will converge to $\Delta \approx (7 + \Psi/17000)u$, where $\Psi = 53000$ for the product form and $\Psi = 9.4\times 10^{12}$ for the expanded form. For double precision, $u \approx 2.2\times 10^{-16}$, so we have $\Delta \approx 2.2\times 10^{-15}$ for the product form and $\Delta \approx 1.2\times 10^{-7}$ for the expanded form. As Newton's method converges towards the root, the bound Ψ may improve. For example, consider the tighter disk $z \in \langle 7, 1\times 10^{-7}\rangle$ for which the error analysis gives

$$f_1(\langle 7, 1\times 10^{-7}\rangle) \in \langle 0, 0.00173 + 0.035u + 0.33u^2\rangle$$

while the expanded form gives

$$f_2(\langle 7, 1\times 10^{-7}\rangle) \in \langle 0, 1.2\times 10^{5} + 8.7\times 10^{12}u + 2.3\times 10^{13}u^2\rangle.$$

Due to the fact that $z = 7$ has an exact representation in floating point (this will not be the case for the roots of most polynomials one would solve numerically!), a better estimate of the final error for Newton's method is in this case just

$\Delta \approx u\Psi/17000$. One finds that as the disk tightens around $z = 7$, $\Psi \to 0$ for the product form and so we expect Newton's method to converge to the exact root. In contrast, for the expanded form of $w_{11}(z)$, Ψ remains at 8.7×10^{12}, implying that one cannot expect a final answer better than $\Delta \approx 1.1 \times 10^{-7}$ using double precision.

This is a good estimate of the behavior one observes by running Newton's method with initial guesses in the vicinity of $z = 7$. Using the product form to evaluate the polynomial and starting with an initial guess of $z = 7.03 + 0.07i$, one obtains errors $|z_i - 7|$ in the first five iterates of 1.96×10^{-3}, 1.42×10^{-6}, 7.36×10^{-13}, 1.54×10^{-25}, 0, at which point the exact root has been found. For the expanded form and the same initial guess, the errors are 1.96×10^{-3}, 1.41×10^{-6}, 4.22×10^{-10}, 2.40×10^{-9}, 2.43×10^{-10}. The iterates dither around the exact root, staying within $|z_i - 7| < 3.6 \times 10^{-9}$ for $3 \le i \le 50$. This is somewhat better than the predicted residual of 1×10^{-7}, which is not surprising because the estimate of Ψ is conservative.

5.3 High-degree Chebyshev polynomials

To illustrate our method's applicability to high-degree polynomials, let's consider the Chebyshev polynomials, which have been studied extensively and are known to have many interesting properties [3]. There is one Chebyshev polynomial in each degree, and scaled so that the leading coefficient is 1, they may be defined recursively by

$$\begin{aligned}
&T_0(x) := 2,\\
&T_1(x) := x, \text{ and}\\
&T_i(x) := xT_{i-1}(x) - T_{i-2}(x)/4, \text{ for } i \ge 2.
\end{aligned}$$

The roots of $T_{n+1}(x)$ are all nonsingular and given by

$$\cos\left(\frac{(2n+1-2k)\,\pi}{2n+2}\right),$$

for $k = 0, 1, ..., n$.

Several of these polynomials, with degrees ranging from 10 to 150, were solved using Bertini. The systems were run with a linear homotopy with a total-degree start system and without homogenizing. Bounds on Φ and Ψ were set using the method of Section 4.3.2 and are displayed in Table 3. Since the solutions of the Chebyshev polynomials are known to be non-singular, the paths were tracked to $t_{\mathrm{f}} = 0$ using basic homotopy continuation with a tolerance of 10^{-8}, i.e., $\tau = 8$, and both safety digits set to 1.

With this setup, double precision is sufficient for path tracking with $n = 10$. This allows for the demonstration of the current cost of higher precision relative to double precision as shown in Table 4. Notably, increasing from double precision (52 bits) to 64-bit multiple precision increases the computational time by a factor of 13.3, while increasing from 64 bits to 1024 bits causes only a

n	Ψ	Φ
10	54	230
50	520,000	$1.4 \cdot 10^{7}$
100	$1.3 \cdot 10^{10}$	$7.2 \cdot 10^{11}$
150	$2.3 \cdot 10^{14}$	$2.0 \cdot 10^{16}$

Table 3: Estimates of Ψ and Φ for the Chebyshev problems of varying degrees.

double (52 bits)	64 bits	96 bits	128 bits	256 bits	512 bits	1024 bits
2.447	32.616	35.456	35.829	50.330	73.009	124.401

Table 4: Average time, in seconds, of 10 runs of the Chebyshev polynomial of degree 10 with $\tau = 8$, for different levels of fixed precision.

factor of 3.8 increase. This clearly shows the high computational cost to make the first move from double precision to higher precision.

Table 5 compares the computational times of fixed precision with those of proactive and reactive adaptive precision. For fixed precision, we report the time for the minimum precision that successfully tracked all paths. For $n = 10$, which runs successfully in double precision, one can see in the proactive run time the overhead cost of checking conditions A, B, and C to see if an increase in precision is needed. Since higher precision is not needed for $n = 10$, the proactive method is the worst performer. For larger n, however, increases in precision are needed, so both adaptive methods outperform fixed precision. As n increases, so does the fraction of the path which must be tracked in higher precision, so relative differences in the times decrease. However, since one does not know a priori how much fixed precision is needed, the real cost of a fixed precision re-run strategy would be much higher.

In these runs, the reactive method uses less time than the proactive approach. This is because the proactive error bounds are conservative, causing an increase in precision somewhat earlier than the reactive method. The high cost of leaving double precision outweighs any savings in avoiding failed steps, thus giving the reactive method the advantage.

n	minimum fixed precision		proactive	reactive
10	52 bits	2.447	2.937	2.447
50	96 bits	1698.350	797.064	319.251
100	128 bits	9763.147	9739.978	3730.438
150	192 bits	32709.488	28944.029	21121.190

Table 5: Average time of 10 runs of Chebyshev polynomials of degree n with $\tau = 8$, in seconds.

τ	minimum fixed precision		proactive	reactive
8	96 bits	6.018	1.565	0.368
14	128 bits	203.67	198.67	86.68

Table 6: Average time of 10 runs of the chemical system, in seconds.

5.4 A problem from chemistry

To illustrate the effect of tightening the tracking tolerance (i.e., increasing τ) on adaptive precision path tracking, we will consider a polynomial system coming from chemistry. To determine chemical equilibria, one may pass from a set of reaction and conservation equations to polynomials, as described in [14, 22]. One such polynomial system, discussed in [18], is the following:

$$f = \begin{bmatrix} 14z_1^2 + 6z_1z_2 + 5z_1 - 72z_2^2 - 18z_2 - 850z_3 + 0.000000002 \\ 0.5z_1z_2^2 + 0.01z_1z_2 + 0.13z_2^2 + 0.04z_2 - 40000 \\ 0.03z_1z_3 + 0.04z_3 - 850 \end{bmatrix}.$$

Bertini homogenized this system using one variable group and found the solutions by using a linear homotopy with a compatible total-degree start system, that is, one whose polynomials have degrees 2, 3, and 2, respectively. The solution set of this system consists of eight nonsingular finite solutions and a multiplicity four solution at infinity.

Using the method of Section 4.3.2, the error bounds were set to $\Psi = 120,030$ and $\Phi = 240,008$. Both safety digits were set to 1. To exhibit the effect of adaptive precision tracking for tighter tolerances, the paths were tracked from $t = 1$ to $t_{\mathrm{f}} = 10^{-30}$ using $\tau = 8$ and $\tau = 14$. Each setting was run with 10 different choices of random numbers, and the average run time for the entire system is provided in Table 6.

With $\tau = 8$, the paths leading to the infinite solution started to fail around $t = 10^{-7}$ when using double precision due to the inadequacy of double precision to complete the linear algebra calculations. With $\tau = 14$, in each of the ten tests performed, one of the paths leading to the infinite solution failed before $t = 0.9$ when using double precision due to Newton iterations not converging to the required tolerance. By rerunning the system with increased (fixed) precision, the path tracking completed successfully.

Both the proactive and reactive methods tracked all of the paths successfully in both tolerances. With $\tau = 8$, both methods started tracking all of the paths in double precision. For the paths in which precision needed to be increased, this first occurred around $t = 10^{-2}$ and $t = 10^{-7}$ for the proactive and reactive method, respectively. With $\tau = 14$, the proactive method increased precision immediately and started tracking all paths in 96 bit precision, while the reactive method was able to start tracking in double precision. These differences of when the precision was increased directly caused the difference in average tracking time between the proactive and reactive method.

5.5 A multivariate system from robotics

When using homotopy continuation methods to solve polynomial systems, one should always make use of an endgame. The purpose of the following example is to compare how the adaptive precision tracking methods of this paper perform when solving a typical, nonpathological example by simple tracking and with the added power of an endgame. In particular, the fractional power series endgame of [19] was employed in solving the following polynomial system [15], an inverse kinematics problem for a general six-revolute serial-link robot.

$$f = \begin{bmatrix} x_1^2 + x_2^2 - 1 \\ x_3^2 + x_4^2 - 1 \\ x_5^2 + x_6^2 - 1 \\ x_7^2 + x_8^2 - 1 \\ \begin{array}{l} -0.24915068x_1x_3 + 1.6091354x_1x_4 + 0.27942343x_2x_3 + 1.4348016x_2x_4 + 0.40026384x_5x_8 - \\ 0.80052768x_6x_7 + 0.074052388x_1 - 0.083050031x_2 - 0.38615961x_3 - 0.75526603x_4 + 0.50420168x_5 - \\ 1.0916287x_6 + 0.40026384x_8 + 0.04920729 \end{array} \\ \begin{array}{l} 0.12501635x_1x_3 - 0.68660736x_1x_4 - 0.11922812x_2x_3 - 0.71994047x_2x_4 - 0.43241927x_5x_7 - \\ 0.86483855x_6x_8 - 0.03715727x_1 + 0.035436896x_2 + 0.085383482x_3 - 0.039251967x_5 - \\ 0.43241927x_7 + 0.013873010 \end{array} \\ \begin{array}{l} -0.63555007x_1x_3 - 0.11571992x_1x_4 - 0.66640448x_2x_3 + 0.11036211x_2x_4 + 0.29070203x_5x_7 + \\ 1.2587767x_5x_8 - 0.62938836x_6x_7 + 0.58140406x_6x_8 + 0.19594662x_1 - 1.2280342x_2 - \\ 0.079034221x_4 + 0.026387877x_5 - 0.057131430x_6 - 1.1628081x_7 + 1.2587767x_8 + 2.162575 \end{array} \\ \begin{array}{l} 1.4894773x_1x_3 + 0.23062341x_1x_4 + 1.3281073x_2x_3 - 0.25864503x_2x_4 + 1.1651720x_5x_7 - \\ 0.26908494x_5x_8 + 0.53816987x_6x_7 + 0.58258598x_6x_8 - 0.20816985x_1 + 2.6868320x_2 - 0.69910317x_3 + \\ 0.35744413x_4 + 1.2499117x_5 + 1.4677360x_6 + 1.1651720x_7 + 1.1076340x_8 - 0.69686809 \end{array} \end{bmatrix}.$$

The exact formulation of this problem includes nonalgebraic functions as coefficients, which Bertini does not currently accept. As a result, the coefficients were calculated in 1024-bit precision, available at [2], with the first 8 digits matching the coefficients given above. Using a homotopy similar to that of Section 5.4, there are 256 paths, 32 leading to finite regular solutions, 64 leading to thirty-two multiplicity two infinite solutions, 64 leading to four multiplicity sixteen infinite solutions, and 96 leading to four multiplicity twenty-four infinite solutions. Even though this problem is 2-homogeneous with the variable groups $\{x_1, x_2, x_5, x_6\}$ and $\{x_3, x_4, x_7, x_8\}$, this was not used. Not taking advantage of the special structure of the system makes it a bit more representative of systems that come up in practice, where, even taking account of all special structure, the number of singular solutions greatly outnumber the actual physical solutions, as in [25].

We ran this system using two different endgames: the first tracking until the path stabilizes and the second using a power series endgame. In both cases, a geometric series of sample values for t was set: $t = 10^{-k}$, for $k = 1, 2, \ldots$ The paths were tracked between the sample points with a tolerance of 10^{-6}, and sample points for the endgame were refined to full precision. The endgame stops when successive predictions of the endpoint at $t = 0$ agree to within 10^{-12}. In the first test, the prediction to $t = 0$ is taken as just the current path point, whereas in the second test, a fractional power series was used following the method in [19]. In both tests, using the method of section 4.3.2, we set

	minimum fixed precision		proactive	reactive
Test 1	192 bits	300.169	85.404	176.591
Test 2	96 bits	196.143	47.393	37.630

Table 7: Average time of 10 runs of the IPP system for the two tests, in seconds.

Φ to 14.5 and Ψ to 42.2, and both safety digits to 1. The average times are provided in Table 7. One observes a marked reduction in run time using the more advanced endgame.

In both tests, the proactive adaptive precision tracking used higher precision and larger step size than the reactive method to track between the sample points. In the first test, the proactive method was faster since taking fewer steps in higher precision was advantageous over taking more steps in lower precision utilized by the reactive method. In the second test, fewer sample points were needed to generate an accurate prediction which kept the tracking sufficiently far away from the endpoints that have high multiplicity. In this case, the reactive method was faster since taking more steps in lower precision was advantageous over using higher precision with less steps utilized by the proactive method.

6 Discussion regarding singular endgames

Multiple precision becomes necessary when numerical stability is poor, such as at a point having a near-singular Jacobian matrix. In the case of an exact singularity at the end of a continuation path, an endgame based on fractional power (Puiseux) series or Cauchy integration [19, 22] can increase the accuracy that can be obtained at a given precision, thereby reducing the computational burden of higher precision, especially in cases where a double precision endgame suffices. For the endgame to succeed, the homotopy path must be sampled in an “endgame operating zone” inside the branchpoint closest to the endpoint but before the path becomes too ill-conditioned at the end [19, 22]. Without enough precision, this operating zone is empty. Adaptive precision tracking will ensure that enough precision is allocated to succeed. Likewise, endgames based on deflating the system to derive a related nonsingular one [9] may need higher than double precision to make a correct decision regarding the rank of the Jacobian at each stage of deflation. The method of [12] may lower the precision needed to make a correct rank decision, but there is no guarantee that double precision, or any prespecified precision, will suffice. So while either type of endgame can help reduce computation and sometimes can avoid higher precision, ultimately an adaptive precision method is needed to ensure success in difficult cases. Among the causes leading to difficult cases, the most common are high degree polynomials, badly scaled functions, endpoints having high multiplicity, and a tight tolerance on the final accuracy.

7 Conclusion

We have presented an adaptive precision predictor-corrector path tracking algorithm that automatically allocates enough precision to allow convergence of Newton's method in the corrector phase. The adaptation can be driven reactively in response to failures in convergence or proactively using rules based on a numerical analysis of Newton's coupled to bounds established on the floating-point error incurred in evaluating the homotopy function. The reactive approach is simpler in that no error bound is required and no computation time is spent evaluating the adaptation rules. It does, however, require the careful setting of the minimum step size, $\varepsilon(P)$, to ensure that the need for higher precision is triggered appropriately. We determined good settings for $\varepsilon(P)$ by experimentation.

The proactive approach requires an error bound. Floating point errors depend not only on the mathematical function being evaluated but also on the sequence of operations used, for example, $a(b+c)$ and $ab+ac$ may have different errors. We provide rough formulas for polynomials evaluated in expanded form, based on just the degree and the size of the coefficients, and we give a more precise method using circular arithmetic, a form of interval analysis, to bound the error in any straight-line polynomial function. To apply the proactive approach to non-polynomial systems, one must extend the interval analysis to handle whatever new elementary functions arise, such as trigonometric functions and the like. Although we have not carried this out in our experiments, there appears to be no fundamental barrier to such extensions of the approach.

In focusing on Newton's method in the corrector phase, we have neglected the prediction phase. While the corrector dominates the convergence to the path, we recognize that a careful study of predictor methods is certainly warranted. The use of different predictor schemes, e.g., Adams-Bashforth rather than Euler, is well worth considering. Moreover, step size control in the current algorithm is entirely reactive, i.e., based on the history of success and failure of the predictor-corrector steps. A careful analysis of the predictor might be combined with the convergence criteria of the corrector to proactively determine a safe step size, possibly making decisions balancing the computational cost of a small step size against that of higher precision. Along this line, the method presented in [8] presents a rigorous step size control algorithm based on interval arithmetic, but it does not consider adaptive precision. Instead, if not enough precision is used, the safe step size goes to zero. An approach like ours may remove this mode of failure. The main intent of [8] is to prevent path crossing when two or more paths in the homotopy pass close to each other. We have not addressed this important issue here.

In conclusion, the adaptive precision approach presented here appears to be highly effective for tracking numerically difficult homotopy paths. Open questions remain about prediction methods and step-size control, and additional work is needed to extend the error analysis to non-polynomial functions. While our error analysis goes far in explaining the conditions that drive an increase in precision, at present the conservative nature of the bounds causes the proactive method to increase precision earlier than necessary. Due to the high cost of leaving double precision, the reactive method, which uses double precision as long as possible, takes less computational time than the proactive method in all our standard tracking experiments. When endgames are utilized and the tracking between the sample points needs to be completed in precision higher than double, the reactive method may become slower than the proactive method due to the increased number of steps needed to track between the sample points.

References

[1] E.L. Allgower and K. Georg. *Numerical Continuation Methods, an Introduction*, volume 13 of *Springer Ser. in Comput. Math.* Springer–Verlag, Berlin Heidelberg New York, 1990. Reprinted in 2003 by SIAM as volume 45 in the Classics in Applied Mathematics series.

[2] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Bertini: Software for Numerical Algebraic Geometry. Available at www.nd.edu/∼sommese/bertini.

[3] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[4] I. Gargantini and P. Henrici, Circular arithmetic and the determination of polynomial zeros. *Numer. Math.*, 18:305–320, 1971/72.

[5] A. Griewank and M.R. Osborne. Analysis of Newton's method at irregular singularities. *SIAM J. Numer. Anal.*, 20(4): 747–773, 1983.

[6] D.E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1969.

[7] R.B. Kearfott and M. Novoa. Algorithm 681: INTBIS, a portable interval Newton/bisection package. *ACM Trans. Math. Softw.*, 16:2:152–157, 1990.

[8] R.B. Kearfott and Z. Xing. An interval step control for continuation methods. *SIAM J. Numer. Anal.*, 31(3):892–914, 1994.

[9] A. Leykin, J. Verschelde, and A. Zhao. Evaluation of jacobian matrices for Newton's method with deflation for isolated singularities of polynomial systems. In *SNC 2005 Proceedings. International Workshop on Symbolic-Numeric Computation.* Xi'an, China, July 19-21, 2005. Edited by Dongming Wang and Lihong Zhi. pp. 19-28.

[10] T.Y. Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica* 6:399–436, 1997.

[11] T.Y. Li. Numerical solution of polynomial systems by homotopy continuation methods. In *Handbook of Numerical Analysis. Volume XI. Special Volume: Foundations of Computational Mathematics*, edited by F. Cucker, pp. 209–304, 2003.

[12] T.Y. Li and Z. Zeng. A rank-revealing method with updating, downdating, and applications. *SIAM J. Matrix Anal. Appl.* 26(4):918–946, 2005.

[13] J.-P. Merlet. A parser for the interval evaluation of analytical functions and its applications to engineering problems. J. Symbolic Computation, 31:475–486, 2001.

[14] A.P. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[15] A.P. Morgan and A.J. Sommese. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput.*, 24(2):115–138, 1987. Errata: *Appl. Math. Comput.*, 51:209, 1992.

[16] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989. Errata: *Appl. Math. Comput.* 51:207, 1992.

[17] A.P. Morgan, A.J. Sommese, and C.W. Wampler. Computing singular solutions to nonlinear analytic systems. *Numer. Math.*, 58(7):669–684, 1991.

[18] A.P. Morgan, A.J. Sommese, and C.W. Wampler. Computing singular solutions to polynomial systems. *Adv. Appl. Math.*, 13(3):305–327, 1992.

[19] A.P. Morgan, A.J. Sommese, and C.W. Wampler. A power series method for computing singular solutions to nonlinear analytic systems. *Numer. Math.*, 63(3):391–409, 1992.

[20] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliable Computing*, 11:4:275-290, 2005.

[21] S.M. Rump. INTLAB - INTerval LABoratory. In *Developments in reliable computing, Proc. of (SCAN-98), Budapest, September 22–25, 1998*, edited by T. Csendes, Kluwer Academic Publishers, Dordrecht, pp. 77–104, 1999.

[22] A.J. Sommese and C.W. Wampler. *Numerical solution of systems of polynomials arising in engineering and science.* World Scientific, Singapore, 2005.

[23] F. Tisseur, Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.* 22(4):1038–1057, 2001.

[24] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM T. Math. Software* 25(2): 251–276, 1999. Software available at www.math.uic.edu/∼jan.

[25] C.W. Wampler, A. Morgan, and A.J. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *ASME Journal of Mechanical Design* 114(1):153–159, 1992.

[26] D. Watkins. *Fundamentals of matrix computations.* John Wiley & Sons Inc., New York, 1991.

[27] J.H. Wilkinson. *Rounding errors in algebraic processes.* New York, Dover Publications Inc., 1994. Reprint of the 1963 original [Prentice-Hall, Englewood Cliffs, N.J.].