CONVERGENCE OF ROOT LOCATION ALGORITHMS WITH RANDOM INITIAL POINTS

JOSHUA SIKTAR

ABSTRACT. An introductory course in numerical methods typically devotes substantial attention to analyzing algorithms for finding roots of functions $f \in C^0(\mathbb{R})$. One of the most frequently asked questions is how to find an optimal starting point for your algorithm of choice. In this article we explore a related question: if we don't know how to find a good starting point, then how many iterations of an algorithm are we expected to make in order to guarantee our error is below a certain threshold? We explore the answer to this question for linearly convergent algorithms, using a method that we aptly call the Random Initial Points Technique.

CONTENTS

1. INTRODUCTION AND MOTIVATION

We discuss a new means of answering the question: how quickly we can expect to converge upon roots of a function using some predetermined algorithm? One can run the same algorithm on the same function many times and only change the starting point of the algorithm. However, a perhaps more insightful way to get the same information is to select the starting point at random and then compute the "expected" runtime of the algorithm where we seek some predetermined accuracy. That is, the technique is a unification of the fundamentals of numerical analysis and probability theory, and we call it the **Random Initial Points Technique**.

The Random Initial Points Technique works as follows: we consider some function $f \in C^0([a,b],\mathbb{R})$ where $f(a)$ and $f(b)$ have opposite sign and choose some method that we know a priori converges to a root of f for any starting point $x_0 \in [a,b]$. We also fix some accuracy we wish to reach, i.e. some $\epsilon > 0$. In this article we also assume the root of f is unique, that $f(a) > 0$, and $f(b) < 0$, which are all just simplifying assumptions. Of these assumptions, the first is most noteworthy, as it assures us that the root to which the algorithm converges is not dependent on the starting point.

Now we consider what is computed when using the Random Initial Points Technique. We assume for some choice of starting point [of the algorithm] $x_0 \in [a,b]$ that the algorithm converges as slowly as possible while still converging either linearly, quadratically, etcetera. Then under that assumption we compute how many non-integral iterations are required to achieve the desired accuracy. By "non-integral," I mean that we do not round the number of iterations to the nearest integer despite the fact that this would make physical sense when actually running algorithms; this helps preserve the continuity of the number of iterations when treated as a function of the algorithm's starting point.

In Section 2 we analyze the scope of the Random Initial Points Technique by first showing when the technique is *not* relevant. In Section 3 we perform a detailed analysis of the Random Initial Points Technique on linearly convergent algorithms; I stress now that the end result we obtain is not nearly as important as the demonstration of how the technique works. Finally in Section 4 we discuss additional extensions and generalizations of this technique.

2. Why the Bisection Method is an Irrelevant Example

Before we get into great detail into how to apply the Random Initial Points Technique to root-finding algorithms, it is worth the time to deepen our understanding of the technique by demonstrating when it is *not* applicable.

Recall that the bisection method is used on some function $f \in C^0([a,b],\mathbb{R})$. Assume without loss of generality that $f(a) > 0$ and $f(b) < 0$. Then the bisection method finds a point $r \in [a,b]$ for which $f(r) = 0$ by continuously splitting the interval $[a,b]$ in half based on the sign of the midpoint of the interval [Ch]. The bisection method, by definition, requires that our initial point x_0 is the midpoint of the interval; that is, $x_0 := \frac{a+b}{2}$. We then seek to split the interval enough times to estimate r with error at most some predetermined $\epsilon > 0$.

There lies the problem: the bisection method is beyond the scope of the Random Initial Points Technique because x_0 is not randomly determined; rather, it is fixed. In section 4 we briefly discuss how this can be circumvented by modifying the traditional bisection algorithm.

3. Expected Runtime for Linearly Convergent Algorithms

This section, the core of the paper, discusses how to apply the Random Initial Points Technique to linearly convergent algorithms for locating a root. Namely, the formulation developed in this section will be useful when applying Newton's Method to locate a non-simple root [Ch]. Before proceeding we more concretely define the class of functions we analyze.

Definition 3.1 (Linearly Convergent Algorithms)**.** *Consider an $f \in C^0([a,b],\mathbb{R})$ for which $f(a) > 0$, $f(b) < 0$, and $f(r) = 0$ for some $r \in [a,b]$. Let $\{x_n\}_{n=0}^{\infty} \subset [a,b]$ be a sequence of points representing the convergence of a root-locating algorithm to r. The algorithm is said to be **linearly convergent** if $\exists\, c \in (0,1)$ such that*

$$|x_n - r| \le c|x_{n-1} - r| \tag{3.1}$$

for all $n \in \mathbb{N}^+$.

It is worth noting that an immediate consequence of (3.1) is the assurance of the inequality

$$|x_n - r| \le c^n|x_0 - r| \tag{3.2}$$

for all $n \in \mathbb{N}^+$. In the arguments that follow we will use (3.2) directly. In particular, this identity will be useful in the proof of the main theorem.

Theorem 3.2. *Assume $f \in C^0([a,b],\mathbb{R})$ where $f(a) > 0$ and $f(b) < 0$. Let $r \in [a,b]$ be fixed such that $f(r) = 0$, and fix $\epsilon > 0$. Assume also that the value of r is unique. Now consider some linearly convergent algorithm to identify the root r, with starting point $x_0 \in [a,b]$ defined uniformly at random. Let $c \in [0,1]$ be some constant for which* (3.2) *holds $\forall n \in \mathbb{N}^+$. Let N be a random variable representing the non-integral number minimal number of steps assuring r is approximated with error at most ϵ. Then*

$$E[N] = \frac{1}{\ln\left(\frac{1}{c}\right)(b-a)}\left((r-a)\left(\ln\left(\frac{r-a}{\epsilon}\right)-1\right)+(b-r)\left(\ln\left(\frac{b-r}{\epsilon}\right)-1\right)\right). \tag{3.3}$$

In order to prove (3.3), we first state and prove a lemma pertaining to the value of N.

Lemma 3.3. *Let a, b, r, and ϵ be defined as in 3.2 and fix the starting point $x_0 \in [a,b]$ of some linearly convergent algorithm. Let $N(x_0)$ refer to the non-integral minimal number of iterations of our chosen linearly convergent algorithm to ensure that the error after completing that number of iterations is at most ϵ. Then*

$$N(x_0) = \log_{\frac{1}{c}}\left(\frac{|x_0 - r|}{\epsilon}\right). \tag{3.4}$$

Proof of Lemma 3.3. Fix $x_0 \in [a,b]$ and the choice of linearly convergent algorithm. To ensure we are computing the non-integral minimal number of iterations, we assume that the algorithm converges as slowly as possible. That is, we assume

$$|x_n - r| \;=\; c|x_{n-1} - r| \tag{3.5}$$

for all $n \in \mathbb{N}^+$. It follows immediately that

$$|x_n - r| \;=\; c^n|x_0 - r| \tag{3.6}$$

for all $n \in \mathbb{N}^+$. In assuring we compute the minimal value N, we must also assume the iteration has error exactly ϵ after $N(x_0)$ iterations (as discussed in Section 1). That is,

$$\epsilon \;=\; c^{N(x_0)}|x_0 - r|. \tag{3.7}$$

Solving equation (3.7) for $N(x_0)$ by taking logarithms gives the desired result. □

With the lemma proven, we can now prove the main result (3.3), which quickly reduces to another, far more technical calculation.

Proof of Theorem 3.2. By the definition of the expected value of a random variable defined over $[a, b]$,

$$\mathbb{E}[N] \;=\; \frac{1}{b-a}\int_a^b N(x)dx. \tag{3.8}$$

Applying (3.4) to (3.8) yields

$$\mathbb{E}[N] \;=\; \frac{1}{b-a}\int_a^b \log_{\frac{1}{c}}\left(\frac{|x-r|}{\epsilon}\right)dx. \tag{3.9}$$

For ease of computation we change the base of the logarithm to e and split (3.9) into two integrals based on the absolute values, resulting in

$$\mathbb{E}[N] \;=\; \frac{1}{\ln\left(\frac{1}{c}\right)(b-a)}\left(\int_a^r \ln\left(\frac{r-x}{\epsilon}\right)dx + \int_r^b \ln\left(\frac{x-r}{\epsilon}\right)dx\right). \tag{3.10}$$

To evaluate (3.10) we first focus on computing the integral $\int \ln\left(\frac{r-x}{\epsilon}\right)dx$ when $x < r$. Using the fact that $\int \ln y dy \;=\; y\ln y - y + C$ combined with the substitution $u \;=\; \frac{r-x}{\epsilon}$ we see that

$$\int \ln\left(\frac{r-x}{\epsilon}\right)dx \;=\; (x-r)\left(\ln\left(\frac{r-x}{\epsilon}\right) - 1\right) + C. \tag{3.11}$$

In a similar vein, when $x > r$,

$$\int \ln\left(\frac{x-r}{\epsilon}\right)dx \;=\; (x-r)\left(\ln\left(\frac{x-r}{\epsilon}\right) - 1\right) + C. \tag{3.12}$$

From here, in order to evaluate (3.10), we must compute the improper integrals which correspond to (3.11) and (3.12). We first demonstrate the computation of $\int_a^r \ln\left(\frac{r-x}{\epsilon}\right)dx$ using (3.11). By the definition of an improper integral,

$$\int_a^r \ln\left(\frac{r-x}{\epsilon}\right)dx \;=\; \lim_{y\to r}\int_a^y \ln\left(\frac{r-x}{\epsilon}\right)dx \;=\; \lim_{y\to r}\left[(x-r)\left(\ln\left(\frac{r-x}{\epsilon}\right) - 1\right)\right]_a^y. \tag{3.13}$$

We can expand (3.13) as follows:

$$\begin{aligned}&\lim_{y\to r}\left[(z-r)\ln\left(\frac{r-z}{\epsilon}\right) + (r-z) + (r-a)\ln\left(\frac{r-a}{\epsilon}\right) + (a-r)\right]\\ &\quad=\; (r-a)\ln\left(\frac{r-a}{\epsilon}\right) + (a-r) + \lim_{z\to r}\left((z-r)\ln\left(\frac{r-z}{\epsilon}\right)\right).\end{aligned} \tag{3.14}$$

The limit $\lim_{z\to r}\left((z-r)\ln\left(\frac{r-z}{\epsilon}\right)\right)$ is an $\frac{\infty}{\infty}$ indeterminate form and can be evaluated with L'Hopital's Rule once we rewrite it as $\lim_{z\to r}\frac{\ln\left(\frac{r-z}{\epsilon}\right)}{\frac{1}{z-r}}$. This limit will evaluate to 0 and it follows that

$$\int_a^r \log_{\frac{1}{c}}\left(\frac{r-x}{\epsilon}\right) dx \;=\; \frac{(r-a)\left(\ln\left(\frac{r-a}{\epsilon}\right)-1\right)}{\ln\left(\frac{1}{c}\right)}. \tag{3.15}$$

Substituting (3.15) into (3.10) gives

$$\mathbb{E}[N] \;=\; \frac{1}{\ln\left(\frac{1}{c}\right)(b-a)}\left(\frac{(r-a)\left(\ln\left(\frac{r-a}{\epsilon}\right)-1\right)}{\ln\left(\frac{1}{c}\right)} + \int_r^b \ln\left(\frac{x-r}{\epsilon}\right) dx\right). \tag{3.16}$$

We leave as an exercise to the reader performing a similar computation to prove the following when $x > r$:

$$\int_r^b \log_{\frac{1}{c}}\left(\frac{x-r}{\epsilon}\right) dx \;=\; \frac{(b-r)\left(\ln\left(\frac{b-r}{\epsilon}\right)-1\right)}{\ln\left(\frac{1}{c}\right)}. \tag{3.17}$$

□

The main difference in this computation from the one shown to prove (3.15) is that the limit generated from the indefinite integral is applied to the lower limit of the integral rather than the upper limit. Substituting (3.17) into (3.16) immediately yields the desired result.

4. Future Extensions and Questions

This article focused predominantly on applying the Random Initial Points Technique to linearly convergent algorithms, but most root-locating algorithms are not of this type. Different variants of Newton's method have different rates of convergence depending on the behavior of the function f and its derivatives [Ch]. This makes it natural to continue our study by performing similar analyses on quadratically and cubically convergent methods. In a similar vein, considering functions with multiple roots in some interval is a complication worth considering because then the root to which the algorithm converges depends on the starting point.

Section 2 discussed the relationship between the Bisection Method and the Random Initial Points Technique. It bears mentioning that if we modified the Bisection Method so that the point of partitioning the interval at each iteration was randomly determined, then the Random Initial Points Technique would apply. An interesting problem would be to compare the expected runtime of this algorithm to the standard Bisection Method.

Finally, the Random Initial Points technique may allow for the derivation of classical inequalities. If two distinct algorithms are used to locate a root of f with the Random Initial Points Technique, and one algorithm is known a priori to converge to the root faster than the other, then the expected runtime of the former algorithm is lower than the latter. Computing two integrals along the lines of deriving (3.3) would thus let us immediately derive an inequality that could be manipulated algebraically as we see fit.

References

[Ch] W. Cheney, D. Kincaid, *Numerical Mathematics and Computing*, Thomson Brooks/Cole, 2008.

E-mail address: jsiktar@andrew.cmu.edu

Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213