Using Directional Fibers to Locate Fixed Points of Recurrent Neural Networks

Garrett E. Katz and James A. Reggia

***Abstract*— We introduce mathematical objects that we call "directional fibers," and show how they enable a new strategy for systematically locating fixed points in recurrent neural networks. We analyze this approach mathematically and use computer experiments to show that it consistently locates many fixed points in many networks with arbitrary sizes and unconstrained connection weights. Comparison with a traditional method shows that our strategy is competitive and complementary, often finding larger and distinct sets of fixed points. We provide theoretical groundwork for further analysis and suggest next steps for developing the method into a more powerful solver.**

***Index Terms*— Directional fibers, fixed points, nonlinear dynamical systems, numerical traversal, recurrent neural networks.**

I. Introduction

ONE of the most basic properties of any dynamical system is the location of its fixed points. However, in nonlinear, high-dimensional dynamical systems, such as recurrent neural networks (RNNs), ascertaining this information can be very challenging. Fixed points of RNNs can represent many things, including stored content-addressable memories [1], solutions to combinatorial optimization problems [2], and unstable way-points of nonfixed dynamics [3]. Consequently, effective fixed point solvers can improve our understanding and engineering of RNNs in all of these use cases. In addition, such solvers can provide information useful in comparing the effects of different learning rules, and can reveal new strategies for solving other nonlinear systems of equations in general.

Both attractive and unstable fixed points are highly relevant to many neurocomputational phenomena, ranging from low-level motor control [4], [5] to high-level cognitive functions [3], [6], [7]. Hopfield [1] showed that through Hebbian learning, networks can emerge that possess fixed points relevant to a desired computation. However, the resulting networks often possess additional "spurious" fixed points that were not present during training, and whose locations are not known *a priori* [8]. Numerous empirical and theoretical studies have provided a solid understanding of the local and global stability of fixed points (surveyed in [9]), as well as their arrangement in phase space, given certain conditions on the connection weights, such as symmetry [10], [11], but not a method for ascertaining the precise locations of every fixed point for arbitrary connection weights. Zeng and Wang [12] derived remarkably fine-grained results: given arbitrary weights, their analysis partitions the phase space into exponentially many regions, and, for each region, provides sufficient conditions for the presence of a unique locally or globally stable attractor. However, short of a brute force approach that checks each region, which is infeasible on large networks, it is not obvious how to efficiently and precisely locate the attractors that are actually present, avoiding regions where they are absent.

In practice, fixed points are often found by repeated local optimization from random initial points [13]. This method can find many fixed points, but is not guaranteed to find them all. To the best of our knowledge, there is no efficient procedure that precisely locates every fixed point of RNNs with arbitrary connectivity. The closest works we have seen are generic global solvers that use bisection-based branch-and-bound search [14], [15], which may not scale to large RNNs.

Since global fixed point location remains an important open problem, it is worth developing new solvers that, if not global themselves, are complementary to existing solvers and provide new perspective on the global problem. Here, we present a novel strategy for locating fixed points in a broad class of dynamical systems, including RNNs with arbitrary size and no symmetry constraint on the weights. We first present our approach in general terms, and then describe its application to RNNs. Next, we show empirically that the method consistently locates many fixed points in many randomly sampled networks. Comparison with an existing method shows that our strategy is both competitive and complementary, often locating different and larger sets of fixed points. Finally, we discuss future directions for improving the solver's performance.

While our approach is *not* a global solver, it does present a new *strategy* toward the eventual goal of developing a global solver, and derives some preliminary theoretical groundwork. Whether or not this strategy can ultimately succeed depends on several theoretical questions that as yet remain open. In addition to proposing and evaluating our method, one of our main contributions is to pose and discuss these open questions and suggest future directions for improving the solver's performance.

II. Theoretical Groundwork

A. Notation

We denote the natural numbers by $\mathbb{N}$ and the real numbers by $\mathbb{R}$. We use $N \in \mathbb{N}$ to denote the dimensionality of a dynamical system and $\mathbf{0} \in \mathbb{R}^N$ to denote a column vector

Manuscript received December 13, 2016; revised May 3, 2017 and July 23, 2017; accepted July 25, 2017. This work was supported in part by ONR under Award N000141310597 and in part by Lockheed Martin Corporation. *(Corresponding author: Garrett E. Katz.)*

The authors are with the Department of Computer Science, University of Maryland at College Park, College Park, MD 20742 USA (e-mail: gkatz@cs.umd.edu).

Digital Object Identifier 10.1109/TNNLS.2017.2733544

containing all zeros. For any matrix M, we use $M_{i,j}$ to denote the (i, j)th entry, and $M_{i,:}$ to denote the ith row. We use the D prefix to denote multivariate differentiation. For example, if $f : \mathbb{R}^N \to \mathbb{R}^N$ is a differentiable function, then Df is its Jacobian, i.e., $(Df(v))_{i,j} = df_i(v)/dv_j$ for $v \in \mathbb{R}^N$. Commas and semicolons inside square brackets denote horizontal or vertical concatenation, respectively, of matrices and vectors. For example, $[A, B]$ is the horizontal concatenation of A and B. The vector and induced matrix 2-norm is denoted by $||\cdot||_2$.

B. Directional Fibers

Our proposed method is based on mathematical objects that we call *directional fibers*. To the best of our knowledge, the concept of directional fibers is new, and their utility for locating fixed points has not been recognized previously, either in RNNs or other dynamical systems. Whereas a standard mathematical fiber is the inverse image of some constant *point*, a directional fiber is the inverse image of some constant *direction*. This concept is well defined for any function whose codomain is a vector space. Directional fibers have several mathematical properties useful for locating fixed points, described in the following. This section presents directional fibers in general terms; Section III shows how our method can be applied to RNNs.

Definition 1: Given $f : \mathbb{R}^N \to \mathbb{R}^N$, and $c \in \mathbb{R}^N - \{\mathbf{0}\}$, the directional fiber of c under f, denoted by $\gamma^{(c)}$, is defined as

$$\gamma^{(c)} \overset{\text{def}}{=} \{v \in \mathbb{R}^N : f(v) \text{ is parallel to } c\}. \tag{1}$$

Consider a discrete-time dynamical system with states in $\mathbb{R}^N$, and state transitions given by

$$\Delta v = f(v) \tag{2}$$

where $v \in \mathbb{R}^N$ is the current state, and $f : \mathbb{R}^N \to \mathbb{R}^N$ is a function that specifies Δv, the change in state after one dynamical update. Given a direction vector c, the directional fiber $\gamma^{(c)}$ is the set of all states where the dynamical update moves in the direction of $\pm c$. Fig. 1 shows an example.[1]

If $f(v)$ is parallel to c, then $f(v) = \alpha c$ for some $\alpha \in \mathbb{R}$, and it is convenient to make α explicit. Let $F^{(c)} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ be the function defined as

$$F^{(c)}(v, \alpha) \overset{\text{def}}{=} f(v) - \alpha c. \tag{3}$$

Then, the directional fiber is equivalent to the following set:

$$\Gamma^{(c)} \overset{\text{def}}{=} \{(v, \alpha) \in \mathbb{R}^N \times \mathbb{R} : F^{(c)}(v, \alpha) = \mathbf{0}\}. \tag{4}$$

We sometimes write $(v, \alpha) \in \mathbb{R}^N \times \mathbb{R}$ as a point $x \in \mathbb{R}^{N+1}$.

If f is differentiable and Df satisfies certain rank conditions, detailed in the following, it turns out that a typical directional fiber is a 1-D manifold containing every fixed point. The practical significance of this property is that a directional fiber can be *numerically traversed* to locate fixed points. More formally, we have the following definition and propositions, which are proved in Appendix A. The main mathematical tools used in the proofs are Sard's theorem [16] and the inverse function theorem (IFT) [17]. Differentiability of f (and hence of $F^{(c)}$, for every c) is assumed throughout.

[1]Directional fibers may also be understood in relation to nullclines. If a point updates in the direction of c, then it is stationary along any direction orthogonal to c. So directional fibers can be thought of as intersections of $N - 1$ nullclines in a rotated coordinate system. Thanks to an anonymous reviewer for this observation.

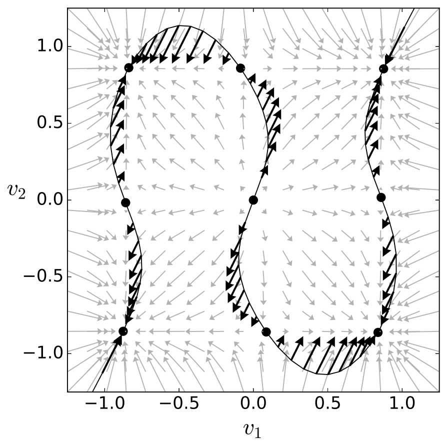


Fig. 1. Phase space for an arbitrary two-neuron network, with an arbitrary directional fiber superimposed. The network model is defined in Section III-A. Light gray arrows indicate $f(v)$ at regularly spaced points $v \in \mathbb{R}^2$. Solid black circles indicate fixed points. The black curve is a directional fiber: the set of all v that update along the same constant direction $\pm c$. The direction $\pm c$ is emphasized by additional arrows showing $f(v)$ at regularly spaced points along the fiber, colored black and scaled by a factor of 2.

Definition 2: A direction vector c is *regular* if $DF^{(c)}$ is full rank at every $x \in \Gamma^{(c)}$. Otherwise, c is *critical.*

Proposition 1: If Df is full rank at every fixed point, then the set of critical c has Lebesgue measure 0 in $\mathbb{R}^N$.

Proposition 2: For any regular c, $\Gamma^{(c)}$ is a 1-D manifold.

Proposition 3: Any directional fiber contains every fixed point: fixed points occur precisely when $\alpha = 0$.

In summary, on the assumption that Df is full rank at every fixed point, almost any directional fiber can be numerically traversed to locate fixed points. Based on our empirical experiments (Section IV), we conjecture that this assumption holds for almost every RNN in the family we study here, with respect to some reasonably defined measure. We leave this conjecture as an open question, although recent results may be relevant [18]. As a caveat, even if the set of RNNs where this assumption fails has measure zero, it still includes some important special cases, such as networks with line attractors, which necessarily contain fixed points where Df is not full rank [19].

Although a regular directional fiber is a 1-D manifold containing every fixed point, it is not necessarily *path connected*. Therefore, whether or not *every* fixed point can be located by traversing a *single* fiber is an open question and depends on the method for choosing c. It is entirely possible that sometimes *no* choice of c will result in a fully connected fiber, and determining when this occurs may be intractable or undecidable. For the RNNs we study here, these questions remain open, but we elaborate on the issue in Section III-C and provide relevant experimental results in Section IV-C.

C. *Fiber-Based Fixed Point Solver*

We now present our method for locating fixed points. The key idea is to choose a suitable directional fiber and numerically traverse it, adding fixed points to a running list as they are encountered. Choosing a suitable fiber, deriving a reliable numerical update scheme and identifying starting and stopping conditions are all system-dependent problems, that we revisit in Section III for the case of RNNs.

Algorithm 1 Fiber-Based Traversal Routine

```
TRAVERSE($f$, $c$)
1: Initialize $(v, \alpha) \in \Gamma^{(c)}$ with starting condition
2: $V^* \leftarrow \{\}$
3: loop
4:     $(\tilde{v}, \tilde{\alpha}) \leftarrow$ TAKE-STEP($f$, $c$, $(v, \alpha)$)
5:     if sign($\tilde{\alpha}$) $\neq$ sign($\alpha$) then
6:         Solve $f(v^*) = \mathbf{0}$ for $v^*$ seeded with $v$
7:         $V^* \leftarrow V^* \cup \{v^*\}$
8:     end if
9:     $(v, \alpha) \leftarrow (\tilde{v}, \tilde{\alpha})$
10:    if stopping condition is satisfied then
11:        return $V^*$
12:    end if
13: end loop
```

The traversal process is codified in algorithm TRAVERSE (Algorithm 1), which operates as follows. Line 1 initializes the traversal at a valid starting point $(v, \alpha) \in \Gamma^{(c)}$. Line 2 initializes V^*, a running list that starts out empty and accumulates fixed points over the course of traversal. At each iteration, line 4 invokes a numerical update scheme to compute a discrete step from the current point (v, α) to the next point $(\tilde{v}, \tilde{\alpha})$ along the fiber. The dedicated subroutine for this numerical update, called TAKE-STEP (line 4), is described in more detail in Section II-D. After the update, lines 5–8 check whether α has changed sign, in which case it has crossed 0, and the update is passing through a new fixed point, by Proposition 3. The precise location of the new fixed point, denoted v^*, is found via local optimization seeded with v on line 6 before it is added to the running list V^* on line 7. Finally, the current point is updated to the new position before the next iteration (line 9). Traversal continues until a stopping condition is satisfied on lines 10–12.

D. TRAVERSE *Update Scheme*

Numerical steps along $\Gamma^{(c)}$ use the unit tangent vector at the current point (v, α), which we denote by z. The proof of Proposition 2 (Appendix A) shows that z is the unique (up to sign) unit vector satisfying

$$DF^{(c)}(v, \alpha)z = \mathbf{0}. \tag{5}$$

In other words, the tangent spans the null space of $DF^{(c)}$, which is 1-D as long as $DF^{(c)}$ is full rank. Equation (5) can be solved for z with standard linear algebra routines.

While (5) can be used to compute a tangent vector, it does not prescribe exactly how that tangent vector should be used. One possibility is to use a numerical integration scheme, such as the Euler or Runge–Kutta methods, but the computed path may diverge from the true mathematical fiber over time. Since we have not only the tangent vector, but also the implicit equation $F^{(c)}(v, \alpha) = \mathbf{0}$, which defines $\Gamma^{(c)}$, we can go further. Using θ^* to denote the current step size, and $x^{(0)}$ and $x^{(\theta^*)}$ to denote (v, α) and $(\tilde{v}, \tilde{\alpha})$, respectively, we solve

$$G(x^{(\theta^*)}) = [\mathbf{0};\ \theta^*] \tag{6}$$

for $x^{(\theta^*)}$, seeded with $x^{(0)}$, where $G : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^{N+1}$ is defined by

$$G(x) \overset{\text{def}}{=} [F^{(c)}(x);\ z^T(x - x^{(0)})]. \tag{7}$$

Equation (6) simultaneously maintains $F^{(c)}(x^{(\theta^*)}) = \mathbf{0}$, which keeps $x^{(\theta^*)}$ in $\Gamma^{(c)}$, and enforces $z^T(x^{(\theta^*)} - x^{(0)}) = \theta^*$, which moves the traversal forward by a distance of θ^* in the tangent direction. This update scheme is a variant of *numerical path following*, for which other methods exist [20], but ours has several novel aspects discussed in Section V. The update is codified in the TAKE-STEP sub-routine (Algorithm 2).

Algorithm 2 Numerical Step Subroutine

```
TAKE-STEP($f$, $c$, $(v, \alpha)$)
1: Compute a suitable step size $\theta^*$
2: Solve Eq. 6 for $x^{(\theta^*)} = (\tilde{v}, \tilde{\alpha})$, seeded with $x^{(0)} = (v, \alpha)$
3: return $(\tilde{v}, \tilde{\alpha})$
```

The step size must be derived with care. If too large, the traversal can "leap" to a remote point on $\Gamma^{(c)}$ or inadvertently reverse direction. If small enough, one can guarantee that $x^{(\theta^*)}$ converges to the same point that would have resulted from the mathematically ideal traversal: that is, the traversal where $x^{(0)}$ flows continuously along $\Gamma^{(c)}$, by a distance of θ^*, in the direction of z. However, if θ^* is *too* small, the traversal can be very slow, so θ^* should be maximized as much as possible while maintaining correctness. This may require leveraging particular properties of the dynamical system at hand, which we did for the specific case of RNNs. Theorem 1 and its proof sketch in Appendix C outline the general derivation strategy. The theoretical and implementation details of our RNN-specific derivation are available online in an archival supplemental technical report [21].

Theorem 1: Given any regular c and any $x^{(0)} \in \Gamma^{(c)}$, one can construct a neighborhood $\mathcal{U}^*$ around $x^{(0)}$ and a $\theta^* > 0$, such that for each $\theta \in [0, \theta^*]$, there is a unique $x^{(\theta)} \in \mathcal{U}^*$ solving $G(x^{(\theta)}) = [\mathbf{0}; \theta]$, and Newton's method will converge to $x^{(\theta)}$ when seeded with $x^{(0)}$. Furthermore, the resulting bijection $\theta \mapsto x^{(\theta)}$ is continuous on $[0, \theta^*]$.

Fig. 2 shows the key quantities referenced by Theorem 1. Each $x^{(\theta)}$ solving $G(x^{(\theta)}) = [\mathbf{0}; \theta]$ is a point in $\Gamma^{(c)}$ that lies a distance of θ from $x^{(0)}$ in the direction of the tangent z. The fact that the map $\theta \mapsto x^{(\theta)}$ is a continuous bijection for all $\theta \in [0, \theta^*]$ guarantees that the same $x^{(\theta)}$ would result from the mathematically ideal traversal. It also guarantees that the transported tangent direction at $x^{(\theta)}$ must have nonnegative dot product with z, which can be used to avoid inadvertently reversing direction. θ^* is the largest step size for which Theorem 1 makes these guarantees. This is the step size used

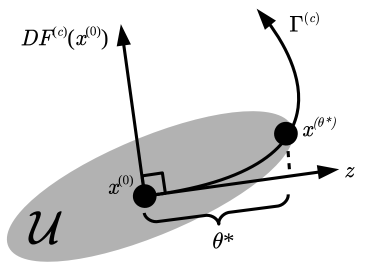


Fig. 2. Key quantities in Theorem 1. $x^{(0)}$ is the current point on the fiber $\Gamma^{(c)}$. z is the tangent vector, which is orthogonal to the rows of $DF^{(c)}(x^{(0)})$ (in higher dimensions, there are multiple row vectors). Steps in $\mathcal{U}$ are certified, θ^* is the maximal such step size, and $x^{(\theta^*)}$ is the new point after the step.

in the TAKE-STEP subroutine. The step size is adaptive, since it depends on the current point $x^{(0)}$.

III. Application to Recurrent Neural Networks

A. Neural Network Model

As a starting point, we have focused on a discrete-time version of Hopfield's analog model [22] with no external stimuli. The update rule for a network with N units is

$$v \mapsto \sigma(Wv) \tag{8}$$

where $v \in \mathbb{R}^N$ is a column vector of real-valued neural activations, $W \in \mathbb{R}^{N\times N}$ is a matrix of connection weights, and σ is the hyperbolic tangent function, applied coordinatewise. Hence, for any given W, the function f is given by

$$f(v) = \sigma(Wv) - v. \tag{9}$$

An ideal solver should locate every v satisfying $f(v) = \mathbf{0}$. The following example characterizes the complexity of this problem. Suppose W is diagonal. Then, $f(v) = \mathbf{0}$ reduces to N independent 1-D problems of the form $\sigma(W_{i,i}v_i) - v_i = 0$. If $W_{i,i} > 1$, then this equation has three solutions, which is apparent from plotting $\sigma(W_{i,i}v_i) - v_i$ as a function of v_i. Therefore, the full system has 3^N solutions. Consequently, any solver that enumerates every fixed point has worst case complexity at least exponential in N. However, we can still ask that a good solver has low *work complexity*, defined as *the time spent per fixed point found*. Section IV characterizes the work complexity of our solver both when the number of fixed points found is small and when it is large.

B. Applying TRAVERSE

To use TRAVERSE with f as in (9), Df must be full rank at every fixed point, as required by Propositions 1 and 2. While we were unable to formally verify this property in general, it was apparently satisfied by every randomly sampled W in our experiments (Section IV). The RNN-specific derivation of Theorem 1 in our technical report [21] also requires that W be invertible, but there are no other structural constraints (such as symmetry).

In addition to a suitable step size, TRAVERSE also requires RNN-specific starting and stopping conditions. For the starting condition, we note that $F^{(c)}(\mathbf{0}, 0) = \sigma(W\mathbf{0}) - \mathbf{0} - 0c = \mathbf{0}$, so the origin is always a valid initial point, for any W and c. Appendix B proves the following stopping condition.

Proposition 4: Given fixed W, suppose that c is regular and that $W_{i,:}c \neq 0$ for all i. For any $(v, \alpha) \in \Gamma^{(c)}$, with tangent vector $z = (\dot{v}, \dot{\alpha})$, if

$$|\alpha| > \frac{\sigma^{-1}\left(\sqrt{1 - \min\left\{1, \frac{1}{\|W\|_2}\right\}}\right) + \sum_j |W_{i,j}|}{|W_{i,:}c|} \tag{10}$$

for all i, then $\dot{\alpha} \neq 0$.

The significance of Proposition 4 is that, as long as $\dot{\alpha} \neq 0$, α cannot reverse direction. In particular, once $|\alpha|$ satisfies (10), α cannot return to 0, so no more fixed points will be encountered. Proposition 4 indicates at least one constraint on c: it must be chosen, so that $W_{i,:}c \neq 0$ for each i. Since we choose c randomly in our experiments, this will be true with probability 1.

Although the range of σ is $(-1, 1)$, f is well defined for any $v \in \mathbb{R}^N$, so directional fibers can leave and return to $(-1, 1)^N$ several times during traversal. However, since $\sigma(Wv) \in (-1, 1)^N$, any v solving $f(v) = \mathbf{0}$ must be in $(-1, 1)^N$. So there is no risk of encountering "fixed points" outside of the neural state space. Additionally, since σ is an odd function, directional fibers are always symmetric about $\mathbf{0}$. In particular, $-v$ is a fixed point whenever v is a fixed point. So traversal need only proceed in one direction from $\mathbf{0}$, and the negations of each fixed point found can be added afterward.

C. Topological Sensitivity of Directional Fibers

By Proposition 3, any directional fiber contains every fixed point. However, the main hurdle faced by our approach is that for some choices of c, the fiber $\Gamma^{(c)}$ is not necessarily *connected*. Traversal of one connected component will fail to identify fixed points contained in another connected component. This effect is shown in Fig. 3. As c is varied through a critical direction, the topology of $\Gamma^{(c)}$ can change: closed loops can form that are disconnected from the component of $\Gamma^{(c)}$ that contains the origin. Any fixed points on these closed loops will never be encountered by TRAVERSE when these c values are used.

In effect, the set of all possible choices of c can be viewed as the unit hypersphere $\mathbb{S}^{N-1}$, and the critical directions partition the sphere into disjoint open sets where c is regular. We refer to these disjoint open sets as the *regular regions*. In Fig. 3 (top), the critical directions are the black solid curves, and the regular regions are the open spaces on the surface of the sphere between the black solid curves. Choices of c in different regular regions can induce different topologies for $\Gamma^{(c)}$. If there is always one regular region in which $\Gamma^{(c)}$ is fully connected, and if there is an efficient algorithm that is always guaranteed to compute some c within this region, then the combination of this hypothetical algorithm with TRAVERSE would constitute a provably correct, global fixed point solver. Even if a result this strong cannot be obtained, it still may be possible that a *small subset* of regular regions can be identified, such that repeating TRAVERSE on a choice of c from each one, and taking the union of fixed points found by each repetition, can be guaranteed to locate *most* of the fixed points of the network. It may even be possible that a single random choice

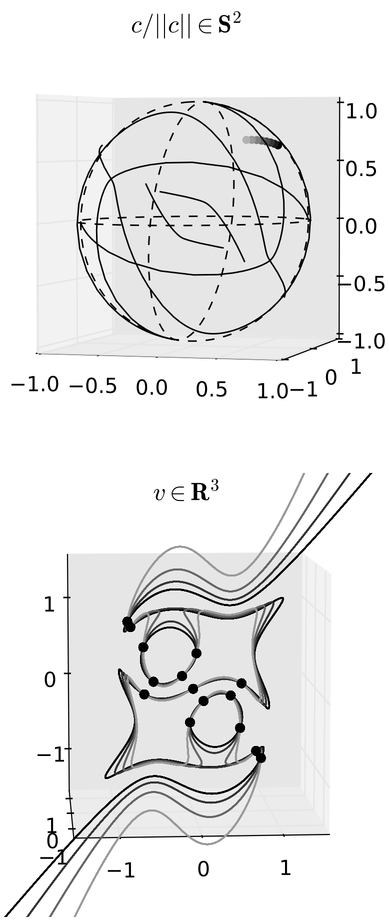


Fig. 3. Topology of $\Gamma^{(c)}$ changes when c passes through a critical direction. Here, an arbitrary three-unit network is used for illustrative purposes. Top: space of all choices of c, normalized to unit length ($\mathbf{S}^2$ denotes the unit sphere). The dashed lines are bad choices where $W_{i,:}c = 0$ for some i [i.e., where the stopping condition (10) is undefined]. The solid black curves are critical directions—bad choices for which $DF^{(c)}$ is singular at some $x \in \Gamma^{(c)}$. The critical sets were approximated using brute force methods on a finely sampled grid, which is not feasible in general, but possible in this low-dimensional example. The critical sets have zero measure, as expected from Proposition 1. The shaded circles varying from light to dark gray in the top-right indicate a particular sequence of c values that cross over the critical set. Bottom: corresponding $\gamma^{(c)}$ for several c values in that sequence, in the same shade of gray, superimposed on the neural phase space. As c crosses over the critical set, two closed loops disconnect from the main body of the directional fiber. Solid black circles indicate fixed points of the network, which are at risk of being isolated on disconnected components of $\gamma^{(c)}$ for some choices of c.

of c will always locate a relatively large subset of fixed points with relatively high probability. We pose the question of how and whether these possibilities can actually be realized in practice as an open problem for future study. Section IV-C provides a preliminary experiment that sheds some light on this question.

IV. Computer Experiments

We implemented a reference version of TRAVERSE for RNNs and ran a battery of computer experiments to gauge the efficacy of our method.[2] In the first set of experiments, we compare our approach with the commonly used baseline of repeating local optimization on a large number of randomly sampled seeds. In the second set of experiments, we compare the output of TRAVERSE using different choices of c.

A. Experimental Methods

1) Sampling Distribution for W: All experiments were performed on several randomly sampled networks, with network size N between 2 and 1024. At each N, we sampled several random W values: 50 at each $N \in \{2, 4, 7, 10, 13, 16\}$, 10 at each $N \in \{24, 32, 48, 64\}$, and 5 at each $N \in \{128, 256, 512, 1024\}$. Each W was constructed as follows. First, we sampled a random $N \times N$ matrix V, with uniform independent identically distributed (i.i.d.) entries in the interval $(-1, 1)$. Next, we set $W = \sigma^{-1}(V)V^{-1}$. Substituting this formula for W into (8) shows that each column of V is a fixed point of the resulting network. This property was useful for comparing the output of different solvers with a set of fixed points that was known *a priori*. In the following, we refer to these as the "known fixed points." Typically, these networks possess many other initially unknown fixed points, in addition to those known by construction.

2) Counting Unique Fixed Points: To accurately compare the outputs of the solvers, it is important to accurately count the number of unique fixed points found by each. Determining whether a point should be considered fixed, and whether two fixed points should be considered distinct, are nontrivial problems in finite-precision arithmetic. The computed values of $f(v)$ at "fixed points" were generally a few multiples of machine precision and rarely identically $\mathbf{0}$. Likewise, any pair of "identical" fixed points were generally a few multiples of machine precision apart, and rarely identically equal.

Using an error analysis of $f(v)$ in finite-precision arithmetic, we devised a test that could decide either "no" or "maybe" as to whether a true fixed point existed within machine precision of a floating-point approximation. As such, strictly speaking, our reported counts are upper bounds on the true counts. However, empirical evidence suggests that these bounds are tight. The details of this test and empirical evidence of its accuracy are provided in our technical report [21].

Given two points $v^{(1)}$ and $v^{(2)}$ both classified as fixed, we marked them as duplicates if $\max_i |v_i^{(1)} - v_i^{(2)}| < 2^{-21}$. While simple, this method proved reliable, as confirmed empirically by computing pairwise distances within the sets of points found by each solver on each network tested, before filtering for duplicates. Fig. 4 shows a histogram of these pairwise distances,[3] aggregated across all solvers and all networks; 2^{-21} clearly separates by a large margin those distances that are near machine precision from those that are not. Several thousand pairs also had distances $\sim 2^{-1024}$ (data not shown), when both $v^{(1)}$ and $v^{(2)}$ were within machine precision of $\mathbf{0}$.

B. Comparison With a Baseline Solver

As a baseline for comparison, we used the common approach of solving $f(v) = \mathbf{0}$ with repeated, randomly initialized local optimization. Our first implementation sampled

[2]The Python code for TRAVERSE, the computer experiments, and the figures are open-source and freely available at www.github.com/garrettkatz/rnn-fxpts.

[3]When a solver returned more than 1000 points, pairwise distances were computed within a random 1000-point subset.

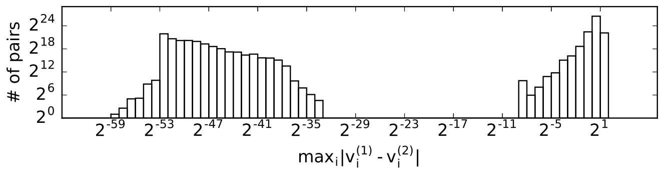


Fig. 4. Pairwise distances between fixed points before filtering for duplicates. Distances are computed within a large sample of pairs across all networks and solvers tested. $v^{(1)}$ and $v^{(2)}$ denote any such pair of fixed points.

the initial points uniformly, and then solved $f(v) = \mathbf{0}$ with Newton's method. However, on all but the smallest networks, almost every initial point converged to the trivial solution $v = \mathbf{0}$. In response, we adopted a more sophisticated technique used by Sussillo and Barak [13]. Their technique starts by running the network dynamics and randomly sampling initial points along the observed trajectories. Each sample is then used as an initial point for an independent run of a local optimization routine, with $(1/2)||f(v)||_2^2$ as the objective function.[4] This objective function attains a minimum value of 0 at any fixed point v. It can also attain nonzero local minima at so-called "slow points" that are not fixed.[5] The optimization routine used is the trust-region Newton conjugate gradient method, provided with the Jacobian and the Gauss–Newton approximation to the Hessian. We refer to this technique as the "baseline method."

The comparative study was conducted as follows. For each network in the test data, TRAVERSE was first run with a random choice of c. We sampled c with i.i.d. Gaussian entries and then normalized c to unit length, which results in a uniform distribution over the surface of the unit hypersphere. TRAVERSE was allowed to run either until the stopping condition was met or a maximum number of steps had been taken,[6] so that the run would terminate in a reasonable time frame. Next, the baseline method was applied to the same network. It was allowed to continue sampling and optimizing random points until the same amount of time had elapsed as had been spent by TRAVERSE. We denote the set of fixed points found by TRAVERSE as T and the set found by the baseline as B.

For each $v \in B$, its negative $-v$ was also added to B, for fair comparison with TRAVERSE, which does the same. B was then postprocessed by removing any slow points that were not fixed, and any fixed points that were duplicates.[7] Next, several set operations were performed on the processed outputs: we counted the fixed points found by both methods ($|T \cap B|$), by either method ($|T \cup B|$), and by one but not the other ($|T - B|$ and $|B - T|$). Fig. 5 shows the results. On average, for $N \leq 16$, $|B - T|$ was somewhat larger than $|T - B|$, indicating that the baseline was finding more fixed points. However, as N grew,

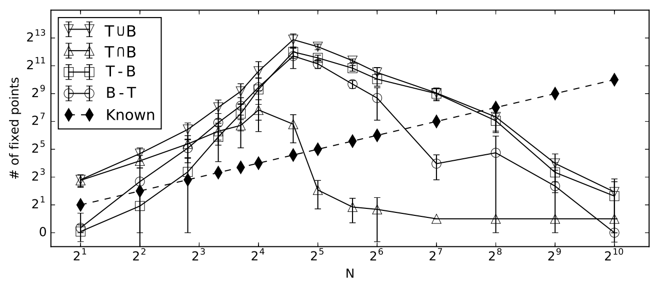


Fig. 5. Fixed points found by TRAVERSE as compared with the baseline. The x-axis indicates network size N and the y-axis indicates set cardinalities, both on a log scale. T and B denote the sets of fixed points found by TRAVERSE and the baseline, respectively. Each data point in the plot is the average cardinality of one set (either $|T \cap B|$, $|T \cup B|$, $|T - B|$, or $|B - T|$ as indicated), where the average is taken over all networks of a given size. Standard deviations of these cardinalities at each N are shown with error bars. The dashed line indicates the number of known fixed points at each N.

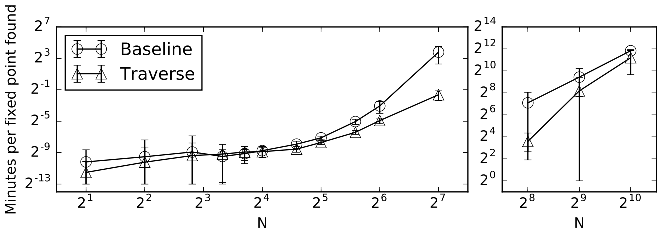


Fig. 6. Comparison of work complexity (time spent per fixed point found, including postprocessing) for each solver. Each data point is the average taken over all networks of a given size. Standard deviations are shown with error bars. Two y-axis scales are used for improved legibility.

$|T - B|$ was often significantly larger than $|B - T|$, indicating that TRAVERSE was finding more fixed points. In addition, $|T \cap B|$ approached 1, indicating that the fixed points found by each method were largely disjoint save for one common element, which turned out to be the trivial fixed point $\mathbf{0}$. Both methods found similar portions of the known fixed points, ranging from ~100% at $N = 2$ to ~0% at $N \geq 32$.

For $N \geq 32$, TRAVERSE consistently reached the maximum step count and terminated early. This effect was exaggerated after $N = 256$ when the step limit was reduced. We expect that the upward trend in fixed points found would have continued if TRAVERSE had run to completion, but elected not to test this because of computational cost. Both methods use many repetitions of matrix operations that are expensive for large N, and the trials for $N = 1024$ ran for several days. However, TRAVERSE often had lower *work* complexity, as shown in Figs. 6 and 7. In particular, Fig. 7 shows that when run to completion, TRAVERSE (and the baseline) had runtime roughly proportional to fixed point counts, whether few or many total fixed points were found. On the other hand, for larger N, early termination renders the runtime roughly constant, so in this case both methods must be scaled up further before any conclusions can be made about absolute work complexities. Even so, the *relative* work complexity of TRAVERSE appears favorable at this scale, at least when run time must be limited. The space requirements of TRAVERSE are also lower, since it records each unique point at most once as it proceeds along the fiber. In contrast, before postprocessing, the baseline had

[4]Strictly speaking, Sussillo and Barak [13] used a continuous-time network model, so they applied the minimization to the analogous continuous-time differential rather than a discrete-time difference.

[5]Slow points have proved useful in identifying nonfixed dynamical features, such as line attractors [23]. Points along a directional fiber where $|\alpha|$ achieves a nonzero local minimum can be viewed as candidate slow points, so our method may also prove relevant in this regard. Exploring this possibility is an important future research direction.

[6]2^{20} steps for $N \leq 256$, and 2^{17} or 2^{15} for $N = 512$ or 1024, respectively.

[7]T does not include any duplicates or slow points by design, but was postprocessed similarly as a sanity check.

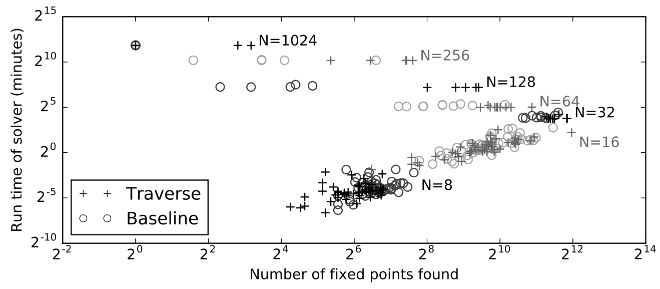


Fig. 7. Scatter plot of absolute run times and fixed point counts. Each data point shows the run time and number of fixed points found by one solver on one network. Data across several network sizes N are shown, with different N values labeled and shown in alternating shades of gray.

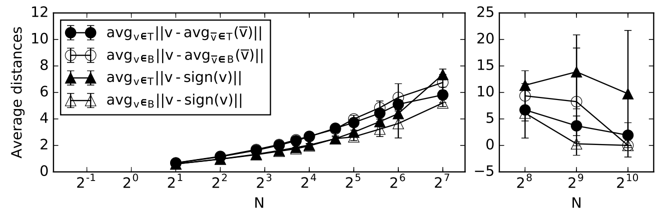


Fig. 8. Statistics on the spatial distributions of fixed points found by TRAVERSE ("T") and the baseline ("B"). Each data point is the average taken over all fixed points found for all networks of a given size. Standard deviations are shown with error bars. Two y-axis scales are used for improved legibility.

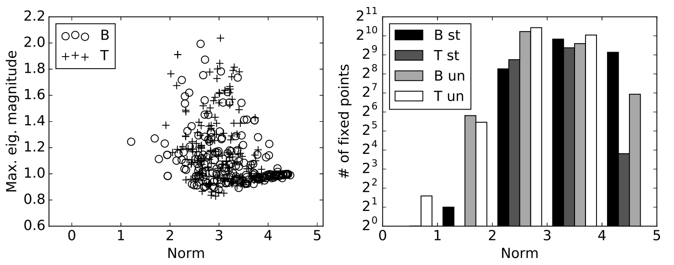


Fig. 9. Stability of fixed points found in one sample network. Left: each data point corresponds to a fixed point found by TRAVERSE ("T") or the baseline ("B"). The maximum eigenvalue magnitude of Df at that point is plotted versus the point's norm. A maximum eigenvalue magnitude less than 1 indicates a stable point. Right: histogram of the same data, showing the distribution of norms within the stable ("st") and unstable ("un") sets.

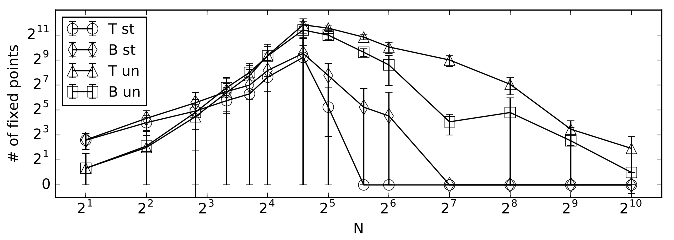


Fig. 10. Counts of stable ("st") and unstable ("un") fixed points found by TRAVERSE ("T") and the baseline ("B"), averaged over all networks of a given size. Standard deviations are shown with error bars.

found many duplicates of the same fixed points (when different seeds converged to the same local optimum), and also many nonfixed slow points. The baseline typically stored ~2–4 times as many points as TRAVERSE before postprocessing. While this could have been counteracted by screening each candidate point online, rather than postprocessing at the end, this would require additional time, and the baseline would find fewer total fixed points in the same timeframe. Since this approach could have been viewed as biasing our results toward TRAVERSE, we did not pursue it here.

The disjointedness of T and B at larger N raises the question of whether each is an essentially random subset of the network's fixed points, or if their respective distributions in phase space differ in some more ordered way. To explore this question, we calculated the average distance around the mean, and the average distance to the nearest corner of the state space $(-1, 1)^N$, across all points in T and in B. Fig. 8 shows the results. For $N \geq 32$, the baseline points were often farther from the means and closer to the corners. One might expect that this greater proximity to the corners is correlated with stability, which we checked using the eigenvalues of Df at each fixed point found. Eigenvalues with magnitude less than one (resp., greater than one) indicate stable (resp., unstable) directions. Fixed points where all eigenvalues have magnitude less than one are, therefore, stable. As expected, the fixed points with larger norms tended to be more stable, as shown for an example network in Fig. 9. Fig. 10 shows the general trend over all networks as a function of N. Both stable and unstable points are located by both methods. However, as N increases, both methods tend to find more unstable points than stable ones, and TRAVERSE tends to find as many or more unstable points than the baseline. We suspect the differences for $N \geq 32$ are not intrinsic properties of directional fibers, but rather because TRAVERSE was starting from $\mathbf{0}$ and consistently terminating early on these trials before reaching the outer extremities of the fiber. Although this might be viewed as an artifact, it is also an important practical consideration. If TRAVERSE tends to locate more unstable points more quickly than the baseline, it may be particularly useful in applications where the *nonfixed* dynamics are of primary concern.

C. Comparison of Different Directional Fibers

Our next experiment concerned the problem of choosing c. We hypothesized that for almost any network, there is a *single* choice of c with which TRAVERSE can locate *all* or at least *most* fixed points. For each network tested, we generated many choices of c (as detailed in the following), denoted $c^1, \ldots, c^k, \ldots, c^K$, and reran TRAVERSE on each one. Let $T(W, c)$ denote the set of fixed points found when using a direction c on a given set of network weights W. The union $\cup_k T(W, c^k)$ can be viewed as a first approximation to the full set of all fixed points of the network. If our hypothesis is true, we might expect to observe a single c^k whose individual $T(W, c^k)$ contains all or most points in the entire union $\cup_k T(W, c^k)$. Of course, this experiment is by no means conclusive, since not every possible c can be checked, and $\cup_k T(W, c^k)$ is only a first approximation to the full set of fixed points. Our goal was solely to collect some relevant preliminary results.

The choices of c^k we used in this experiment were motivated by (10), which requires that $W_{i,:}c \neq 0$ for each i. Visual inspection of low-dimensional examples, as in Fig. 3, suggests that c satisfying $W_{i,:}c = 0$ may be loosely correlated with the critical directions. As such, the regions of the sphere where all $W_{i,:}c \neq 0$ may be considered as rough proxies for the regular regions. There are 2^N such regions; one for each

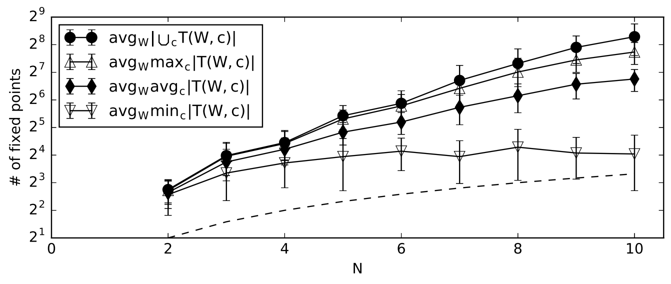


Fig. 11. Statistics on the number of fixed points found by TRAVERSE, using different choices of c. $T(W, c)$ denotes the set of points found when using direction c on network weights W. Multiple c were tested with each W, and multiple W were tested at each network size N. Statistics on $T(W, c)$ were calculated across all c for a fixed W. Then, averages and standard deviations of those statistics were calculated across all W for a fixed N. Those averages are plotted in the figure with standard deviations indicated by error bars. The dashed line indicates the number of fixed points known by construction.

possible choice of $\operatorname{sign}(Wc) \in \{-1, 1\}^N$. Let $s^k \in \{-1, 1\}^N$ denote the kth possibility. A corresponding c^k can be found by solving the linear system $Wc^k = s^k$. In practice, we add small random noise to ensure that c^k is in general position and hence most likely a regular direction. We computed each c^k in this way for each W that was tested, and then used TRAVERSE to compute the corresponding $T(W, c^k)$. To assess the results, we computed the following statistics for each W, shown in Fig. 11: $\min_k|T(W, c^k)|$, $\operatorname{mean}_k|T(W, c^k)|$, $\max_k|T(W, c^k)|$, and $|\cup_k T(W, c^k)|$. We suppress k in Fig. 11 for legibility. For this experiment, N was capped at 10, since $N = 10$ already results in 2^{10} possible c^k, each of which must be used for another run of TRAVERSE.

We found that even the worst choices of c^k (corresponding to $\min_k |T(W, c^k)|$) typically found more fixed points than were known by construction of W, and that the single best choice of c^k (corresponding to $\max_k |T(W, c^k)|$) consistently located a large portion of the entire union. Even on average, most individual choices of c^k (corresponding to $\operatorname{mean}_k|T(W, c^k)|$) were able to independently locate a nonnegligible portion of the entire union. These results suggest that even choosing c at random can lead to reasonable performance of TRAVERSE, and for almost any network, there *may* be a single nearly optimal choice of c. On the other hand, we are as yet unable to discern the pattern governing *which* choice of s^k produced the best c^k on any given network. So, short of enumerating every s^k (which is infeasible for large N), it remains unknown how to identify the optimal c for any particular weight matrix W.

V. Discussion

We have presented a new, general strategy for locating fixed points, based on directional fibers, and have applied it to RNNs. Compared with a traditional fixed point solver, using local search from random seeds, our strategy often finds more fixed points, particularly on larger networks. The points found also tend to have different distributions in phase space, making our approach complementary to the baseline. Furthermore, our method has lower space requirements, returning points that are already guaranteed to be fixed and unique, as opposed to the baseline, which may find many duplicated points.

Since TRAVERSE follows an implicitly defined curve, it is an example of numerical path following [20]. However, whereas the standard formulation of numerical path following involves a "predictor" step followed by a "corrector" step at every update, we formalize our update using a single step comprised of the numerical solution of (6). Moreover, in addition to a robust stopping condition, we provide a recipe for maximizing step size, which formally guarantees that the numerical traversal matches the mathematically ideal traversal up to machine precision. Finally, to the best of our knowledge, the notion of *directional fibers* in particular as the paths to be followed is new. Numerical path following has been used for fixed point location before, but in a rather different way via homotopy continuation [24]. In these methods, a separate path is traced for each fixed point, and each such path originates from a known fixed point in a simpler dynamical system that is continuously transformed into the target dynamical system. In contrast, our method uses a *single* path defined *within* the state space of the target dynamical system (i.e., a directional fiber), and which passes through *multiple* fixed points.

The main open question in our approach is how and when a suitable directional fiber can be identified. However, we have found that our method locates a substantial number of fixed points even with random choices of c. We have also found preliminary empirical evidence that for many networks, there may exist a single nearly optimal choice of c that locates most fixed points. However, the question remains of how to *instantiate* this c in practice. This open problem is ripe for further theoretical and empirical investigation, which is the subject of future work. One avenue worth investigating is a *combination* of the baseline and TRAVERSE: since the points found by each are largely complementary, it appears likely that baseline points often lie on disconnected components of the fiber, and can be used to initialize subsequent runs of TRAVERSE along those disconnected components.

In addition to locating many fixed points, other criteria for choosing c should also be considered. Different c values in the same regular region may find the same fixed points, but in a very different running time, depending on the shapes of the corresponding fibers. If one fiber has sharper "bends," it may require smaller step sizes and hence more steps. Therefore, the choice of c can also impact the work complexity of TRAVERSE.

Aside from choosing c, various other issues in our current work should be addressed. To begin with, the open question of whether almost every W satisfies Proposition 1 should be answered. In addition, our application of TRAVERSE to RNNs should be extended to handle external stimuli, and tested using other generative processes for W that are more representative of the networks used in modern machine learning. This includes W that are singular, sparse, larger, and/or trained (as in [13]).

Our method also relies on a finite-precision error analysis for accurately counting unique fixed points, which appears effective in practice but lacks full mathematical rigor. Aside from their bisection-based algorithms, the existing rigorous global solvers also use provably correct data types that fully account for roundoff errors [14], [15]. Incorporating these data types into our implementation of TRAVERSE could resolve the issue.

Future work could also explore the applicability of directional fibers in other nonneural dynamical systems, although system-specific starting conditions, stopping conditions, and step sizes would have to be derived. In addition, finding zeros of a gradient can be viewed as finding fixed points of a vector field. So directional fibers may have relevance to (potentially nonconvex) numerical optimization.[8] Relatedly, the prospects of TRAVERSE for NP-hard optimization are intriguing but uncertain, and hinge on a better understanding of directional fibers, both in terms of correctness (i.e., connectedness) and complexity (i.e., amenability to fast traversal).

Finally, future studies can put TRAVERSE to work in order to further deepen our understanding and engineering of fixed and nonfixed network dynamics. For example, although we contrast TRAVERSE with the baseline method as implemented by Sussillo and Barak [13], it was only a small part of their work. Once the fixed points were located, they used an additional analysis, based on linearization around those fixed points, to form compelling explanations for how the networks behaved and how they accomplished the tasks for which they were trained. Repeating this analysis on fixed points located by TRAVERSE could yield additional insights.

Appendix A
Properties of Directional Fibers

Proof of Proposition 1: Consider the function $h : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ defined by $h(v, \beta) = \beta f(v)$. Then, $Dh = [\beta Df, f]$. Let $\mathcal{H} = \{h(v, \beta) : \text{rank}(Dh(v, \beta)) < N\}$. By Sard's theorem, $\mathcal{H}$ has Lebesgue measure 0 in $\mathbb{R}^N$. Now, let c be any critical direction. By definition, there is some $(v, \alpha) \in \Gamma^{(c)}$, such that $\text{rank}(DF^{(c)}(v, \alpha)) < N$. Since $DF^{(c)} = [Df, -c]$, this implies that $\text{rank}(Df(v)) < N$ as well. If α values were 0, then $f(v)$ would be $\mathbf{0}$, since $F^{(c)}(v, \alpha) = f(v) - \alpha c = \mathbf{0}$ for all $(v, \alpha) \in \Gamma^{(c)}$. But then, Df would be less than full rank at a fixed point, contradicting the antecedent of the proposition. So, given $\alpha \neq 0$, we can rearrange $f(v) - \alpha c = \mathbf{0}$ to write $c = f(v)/\alpha = h(v, 1/\alpha)$. So $DF^{(c)}(v, \alpha) = [Df(v), -f(v)/\alpha]$, which has the same rank as any matrix obtained by nonzero rescaling of its columns, including $Dh(v, 1/\alpha) = [(1/\alpha)Df(v), f(v)]$. Hence, $Dh(v, 1/\alpha)$ has rank less than N, and so $c = h(v, 1/\alpha)$ is in $\mathcal{H}$. Therefore, the set of critical directions is a subset of a zero measure set, and so also zero measure. □

Proof of Proposition 2: Given that c is regular, $DF^{(c)}(x^{(0)})$ is full rank at any $x^{(0)} \in \Gamma^{(c)}$, and since $F^{(c)}$ maps $\mathbb{R}^{N+1}$ to $\mathbb{R}^N$, the full rank $DF^{(c)}(x^{(0)})$ must have a 1-D null space. Let z be a unit vector spanning that null space and let $G : \mathbb{R}^{N+1} \to \mathbb{R}^{N+1}$ be defined as in (7). Since z spans the null space of $DF^{(c)}(x^{(0)})$ and has unit length, $DG(x^{(0)}) = [DF^{(c)}(x^{(0)}); z^T]$ is invertible and z is also the $(N+1)$th column of $(DG(x^{(0)}))^{-1}$. By the IFT, G is a homeomorphism between a neighborhood $\mathcal{U}$ of $x^{(0)}$ and a neighborhood $\mathcal{V}$ of $G(x^{(0)})$. In particular, for $x \in \mathcal{U} \cap \Gamma^{(c)}$, $F^{(c)}(x)$ vanishes and $G(x) \in \mathcal{V}$ has the form $[\mathbf{0}; z^T(x - x^{(0)})]$, so $\mathcal{U} \cap \Gamma^{(c)}$ is homeomorphic to $\mathbb{R}$. Hence, $\Gamma^{(c)}$ is locally 1-D around $x^{(0)}$, and effectively parameterized by the $(N+1)$th coordinate of $G(x)$. The tangent to $\Gamma^{(c)}$ at $x^{(0)}$ is the derivative of G^{-1} with respect to this parameter, which by the IFT is precisely the $(N+1)$th column of $(DG(x^{(0)}))^{-1}$, namely, z. Given that c is regular, $DF^{(c)}$ is full rank at every such $x^{(0)} \in \Gamma^{(c)}$, and so $\Gamma^{(c)}$ is globally a 1-D manifold. □

Proof of Proposition 3: Given any c, and any fixed point v, we have $F^{(c)}(v, 0) = f(v) - 0c = \mathbf{0} - \mathbf{0} = \mathbf{0}$, so $(v, 0) \in \Gamma^{(c)}$. □

Appendix B
RNN-Specific Stopping Condition

Proof of Proposition 4: Let $(v, \alpha) \in \Gamma^{(c)}$ satisfy (10). Writing $DF^{(c)}$ more explicitly in (5), we have

$$\Sigma W \dot{v} - \dot{v} - \dot{\alpha} c = \mathbf{0} \tag{11}$$

where Σ is diagonal with $\Sigma_{i,i} = \sigma'(W_{i,:}v)$ and σ' denotes the derivative of σ with respect to its argument. If $\dot{\alpha} = 0$, then (11) implies that $||\Sigma W \dot{v}||_2 = ||\dot{v}||_2$. So

$$||\Sigma W \dot{v}||_2 < ||\dot{v}||_2 \text{ implies } \dot{\alpha} \neq 0. \tag{12}$$

By properties of $||\cdot||_2$, the antecedent of (12) is true whenever

$$\max_i \ \sigma'(W_{i,:}v) < 1/||W||_2 \tag{13}$$

is true. So it remains to show that (10) implies (13). Since $(v, \alpha) \in \Gamma^{(c)}$, for each i, we have

$$v_i = \sigma(W_{i,:}v) - \alpha c_i \tag{14}$$
$$|W_{i,:}v| \geq l||\alpha W_{i,:}c| - |W_{i,:}\sigma(Wv)|r| \tag{15}$$
$$\sigma'(W_{i,:}v) \leq \sigma'(|\alpha W_{i,:}c| - \textstyle\sum_j |W_{i,j}|) \tag{16}$$

where (14) follows by rearranging $F^{(c)}(v, \alpha) = \mathbf{0}$, (15) follows by multiplying both sides of (14) by W and using properties of $|\cdot|$, and (16) follows by applying σ' to both sides of (15) and using properties of σ. So (13) is satisfied if

$$\sigma'(|\alpha W_{i,:}c| - \textstyle\sum_j |W_{i,j}|) \leq \min\{1, 1/||W||_2\} \tag{17}$$

for all i. Using the fact that $\sigma' = 1 - \sigma^2$ and isolating $|\alpha|$ in (17) shows that (17) is equivalent to (10). □

Appendix C
Numerical Update Scheme

Proof of Theorem 1(Sketch): Define a norm $||\cdot||$ that will facilitate tight bounds, and given some θ, let $x^{(n)}$ denote the nth Newton iterate. Combining formulas for Newton iterations and Taylor's theorem derives a recurrence relation of the form

$$-DG(x^{(n)})(x^{(n+1)} - x^{(n)}) = R^{(n-1)}(x^{(n)} - x^{(n-1)}) \tag{18}$$

where $R^{(n-1)}(\cdot)$ is a second-order Taylor remainder. Taking norms on both sides of (18) obtains a bound of the form

$$||x^{(n+1)} - x^{(n)}|| \leq \rho^{(n)}||x^{(n)} - x^{(n-1)}||^2 \tag{19}$$

where $\rho^{(n)}$ is an expression whose contents depend on the particulars of $||\cdot||$, $DG(x^{(n)})$, and $R^{(n-1)}(\cdot)$. One can check that $x^{(1)} - x^{(0)} = \theta z$, which relates the recurrence to θ in the

[8]Thanks to an anonymous reviewer for this observation.

base case. If all the $\rho^{(n)}$ values can be bounded by a single ρ, then iterating (19) from this base case gives

$$||x^{(n+1)} - x^{(n)}|| \leq (\rho\theta||z||)^{2^n}/\rho. \tag{20}$$

If, in addition

$$\rho\theta||z|| < 1 \tag{21}$$

then (20) will imply that $x^{(n)}$ is a Cauchy sequence and hence convergent, and will also allow us to bound $x^{(n)}$ near $x^{(0)}$ as follows:

$$\begin{align} ||x^{(n)} - x^{(0)}|| &\leq \sum_{k=0}^{n-1} ||x^{(k+1)} - x^{(k)}|| \tag{22}\\ &\leq \frac{1}{\rho}\sum_{k=0}^{n-1} (\rho\theta||z||)^{2^k} \tag{23}\\ &\leq \frac{1}{\rho}\sum_{k=1}^{n} (\rho\theta||z||)^{k} \tag{24}\\ &\leq \frac{1}{\rho}\left(\frac{\rho\theta||z||}{1-\rho\theta||z||}\right) \tag{25} \end{align}$$

where (22) follows from the triangle inequality, (23) follows by substituting (20), (24) follows readily from (23), and (25) follows from the formula for geometric series.

Now, consider any $\delta > 0$, which determines a neighborhood $\mathcal{U} = \{x : ||x - x^{(0)}|| < \delta\}$. For $x \in \mathcal{U}$, since x is bounded near $x^{(0)}$, $DG(x)$ is bounded near $DG(x^{(0)})$. If δ is not too large, $DG(x)$ can also be kept full rank, since it will be near $DG(x^{(0)})$, which is evaluated at a point on $\Gamma^{(c)}$ and hence full rank. Using these bounds on $DG(x)$, one can determine a single ρ that bounds any $\rho^{(n)}$ for which $x^{(n-1)}$ and $x^{(n)}$ fall inside $\mathcal{U}$. With this ρ fixed, consider any θ that satisfies (21) as well as

$$\frac{1}{\rho}\left(\frac{\rho\theta||z||}{1-\rho\theta||z||}\right) < \delta. \tag{26}$$

Using (22)–(26), show by mathematical induction that $x^{(n)} \in \mathcal{U}$ and $\rho^{(n)} < \rho$ for *all* iterates, so that (20) is always satisfied and the $x^{(n)}$ values do indeed converge. Let $x^{(\theta)}$ denote their limit. To show that $x^{(\theta)}$ does, in fact, solve $G(x^{(\theta)}) = [\mathbf{0}; \theta]$, take norms in the Newton iteration formula to get

$$||[\mathbf{0}; \theta] - G(x^{(n)})|| \leq ||DG(x^{(n)})|| \cdot ||x^{(n+1)} - x^{(n)}||. \tag{27}$$

Given full rank DG in $\mathcal{U}$, we have $||DG(x^{(n)})|| > 0$, while $||x^{(n+1)} - x^{(n)}||$ approaches 0. So $G(x^{(n)})$ approaches $[\mathbf{0}; \theta]$.

Finally, consider all such δ, each of which determines a corresponding $\mathcal{U}$, ρ, and θ. If the expressions for ρ and θ in terms of δ are continuous, then the maximal θ can be taken as θ^*, and the corresponding neighborhood as $\mathcal{U}^*$. It remains to show that for any $\theta \in [0, \theta^*]$, $x^{(\theta)}$ is unique in $\mathcal{U}^*$, and a continuous function of θ. Consider $\theta_1, \theta_2 \in [0, \theta^*]$, and use Taylor's theorem, the full rank of DG in $\mathcal{U}^*$, and the fact that $||G(x^{(\theta_1)}) - G(x^{(\theta_2)})||_2 = |\theta_1 - \theta_2|$ to obtain

$$|\theta_1 - \theta_2| \geq \lambda||x^{(\theta_1)} - x^{(\theta_2)}||_2 \tag{28}$$

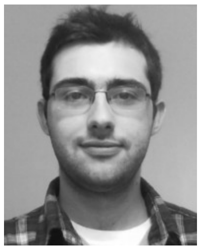
where λ is a nonzero lower bound on the least singular value of DG. Equation (28) shows that $\theta \mapsto x^{(\theta)}$ is continuous, and that if $x^{(\theta_1)} \neq x^{(\theta_2)}$, then $\theta_1 \neq \theta_2$. □

ACKNOWLEDGMENT

The authors would like to thank Dr. H. Elman and Dr. G. Forni for helpful discussions.

REFERENCES

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.*, vol. 79, no. 8, pp. 2554–2558, 1982.
[2] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, Jul. 1985.
[3] M. I. Rabinovich, R. Huerta, P. Varona, and V. S. Afraimovich, "Transient cognitive dynamics, metastability, and decision making," *PLoS Comput. Biol.*, vol. 4, no. 5, p. e1000072, 2008.
[4] M. F. Afzal and A. A. Minai, "Reliable storage and recall of aperiodic spatiotemporal activity patterns using scaffolded attractors," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2016, pp. 2047–2054.
[5] H. Wang, Q. Li, J. Yoo, and Y. Choe, "Dynamical analysis of recurrent neural circuits in articulated limb controllers for tool use," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2016, pp. 4339–4345.
[6] J. Sylvester and J. Reggia, "Engineering neural systems for high-level problem solving," *Neural Netw.*, vol. 79, pp. 37–52, Jul. 2016.
[7] P. Varona and M. I. Rabinovich, "Hierarchical dynamics of informational patterns and decision-making," *Proc. B Roy. Soc.*, vol. 283, no. 1832, p. 20160475, 2016.
[8] J. Bruck and V. P. Roychowdhury, "On the number of spurious memories in the Hopfield model," *IEEE Trans. Inf. Theory*, vol. 36, no. 2, pp. 393–397, Mar. 1990.
[9] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 7, pp. 1229–1262, Jul. 2014.
[10] D. J. Amit, *Modeling Brain Function: The World of Attractor Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[11] S. V. B. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 204–215, Jun. 1990.
[12] Z. Zeng and J. Wang, "Multiperiodicity of discrete-time delayed neural networks evoked by periodic external inputs," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1141–1151, Sep. 2006.
[13] D. Sussillo and O. Barak, "Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks," *Neural Comput.*, vol. 25, no. 3, pp. 626–649, 2013.
[14] A. N. Wittig, "Rigorous high-precision enclosures of fixed points and their invariant manifolds," Ph.D. dissertation, Michigan State Univ., East Lansing, MI, USA, 2012.
[15] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, vol. 13. Dordrecht, The Netherlands: Springer, 2013.
[16] A. Sard, "The measure of the critical values of differentiable maps," *Bull. Amer. Math. Soc.*, vol. 48, no. 12, pp. 883–890, 1942.
[17] S. G. Krantz and H. R. Parks, *The Implicit Function Theorem: History, Theory, and Applications*. New York, NY, USA: Springer, 2012.
[18] D. Soudry and Y. Carmon. (2016). "No bad local minima: Data independent training error guarantees for multilayer neural networks." [Online]. Available: https://arxiv.org/abs/1605.08361
[19] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky, "Theory of orientation tuning in visual cortex," *Proc. Nat. Acad. Sci. USA*, vol. 92, no. 9, pp. 3844–3848, 1995.
[20] E. L. Allgower and K. Georg, "Numerical path following," *Handbook Numer. Anal.*, vol. 5, no. 3, p. 207, 1997.
[21] G. E. Katz and J. A. Reggia, "Identifying fixed points in recurrent neural networks using directional fibers: Supplemental material on theoretical results and practical aspects of numerical traversal," Univ. Maryland, College Park, College Park, MD, USA, Tech. Rep. CS-TR-5051, Dec. 2016. [Online]. Available: http://hdl.handle.net/1903/18918
[22] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, no. 10, pp. 3088–3092, 1984.
[23] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," *Nature*, vol. 503, no. 7474, pp. 78–84, 2013.
[24] S. N. Chow, J. Mallet-Paret, and J. A. Yorke, "Finding zeroes of maps: Homotopy methods that are constructive with probability one," *Math. Comput.*, vol. 32, no. 143, pp. 887–899, 1978.



Garrett E. Katz received the B.A. degree in philosophy from Cornell University, Ithaca, NY, USA, in 2007, and the M.A. degree in mathematics from The City College of New York, New York City, NY, USA, in 2011. He is currently pursuing the Ph.D. degree in computer science with the University of Maryland at College Park, College Park, MD, USA.

His current research interests include neural computation, machine learning, causal inference, cognitive robotics, imitation learning, and complex adaptive systems.

James A. Reggia received the M.D. and Ph.D. degrees in computer science from the University of Maryland at College Park, College Park, MD, USA, in 1975 and 1981, respectively.

He is currently a Professor with the Department of Computer Science, with joint appointments in the Institute for Advanced Computer Studies and the Maryland Robotics Center, University of Maryland at College Park. His current research interests include neural computation, machine learning, causal inference, cognitive robots, genetic programming, and swarm intelligence.