COMPUTING TENSOR Z-EIGENVECTORS WITH DYNAMICAL SYSTEMS

AUSTIN R. BENSON* AND DAVID F. GLEICH†

Abstract. We present a new framework for computing Z-eigenvectors of general tensors based on numerically integrating a dynamical system that must converge to a Z-eigenvector. Our motivation comes from our recent research on spacey random walks, where the long-term dynamics of a stochastic process are governed by a dynamical system that must converge to a Z-eigenvectors of a given transition probability tensor. Here, we apply the ideas more broadly to general tensors and find that our method can compute Z-eigenvectors that algebraic methods like the higher-order power method cannot compute.

1. Tensor eigenvectors. Computing matrix eigenvalues is a classic problem in numerical linear algebra and scientific computing. Given a square matrix $\boldsymbol{A}$, the goal is to find a vector-scalar pair $(\mathbf{x}, \lambda)$ with $\mathbf{x} \neq 0$ satisfying

$$\boldsymbol{A}\mathbf{x} = \lambda\mathbf{x}. \tag{1.1}$$

The pair $(\mathbf{x}, \lambda)$ is called the eigenpair, $\mathbf{x}$ the eigenvector, and λ the eigenvalue. After several decades of research and development, we have, by and large, reliable methods and software for computing *all* eigenpairs of a given matrix $\boldsymbol{A}$. (Experts will, of course, be aware of exceptions, but we hope they would agree with the general sentiment of the statement.)

In numerical *multilinear* algebra, there are analogous eigenvector problems (note the plurality). For example, given a three-mode cubic tensor $\underline{\boldsymbol{T}}$ (here meaning that $\underline{\boldsymbol{T}}$ is a multi-dimensional $n \times n \times n$ array with entries $\underline{T}_{i,j,k}$, $1 \leq i, j, k, \leq n$[1]), the two most common tensor eigenvector problems are:

Z-eigenvectors [24]	H-eigenvectors [24]
l^2-eigenvectors [18]	l^k-eigenvectors [18]
$\sum_{jk} \underline{T}_{i,j,k} x_j x_k = \lambda x_i,\ 1 \leq i \leq n$	$\sum_{jk} \underline{T}_{i,j,k} x_j x_k = \lambda x_i^2,\ 1 \leq i \leq n$
$\|\mathbf{x}\|_2 = 1$	$\mathbf{x} \neq 0$

We use the "Z" and "H" terminology instead of "l^2" and "l^k". Both Z- and H-eigenvectors are defined for tensors with the dimension equal in all modes (such a tensor is called *cubic* [7]), and the definitions can be derived by showing that the eigenpairs are KKT points of a variational form for a generalization of a Rayleigh quotient to tensors [18]. One key difference between the types is that H-eigenvectors are scale invariant, while Z-eigenvectors are not—this is why we put a norm constraint on the vector. Specifically, if we ignore the norm constraint and scale $\mathbf{x}$ by a constant, the corresponding eigenvalue would change; for H-eigenpairs it is easy to see that this is not the case. If $\underline{\boldsymbol{T}}$ is symmetric, then it has a finite set of Z-eigenvalues and moreover, there must be a real eigenpair where the tensor order is odd [6].

This paper presents a new framework for computing Z-eigenpairs. Tensor Z-eigenvectors show up in a variety of applications, including evolutionary biology [5, 20], low-rank factorizations and compression [1, 12, 16, 21], signal processing [11, 15],

*Department of Computer Science, Cornell University (arb@cs.cornell.edu).
†Department of Computer Science, Purdue University (dgleich@purdue.edu).

[1]In true pure mathematical parlance, this is the definition of a hypermatrix, not a tensor (see the book chapter by Lim for the precise definitions [19].) However, "tensor" has become synonymous with a multi-dimensional array of numbers [13], and we adopt this terminology.

quantum geometry [10, 27], medical imaging [25], and data mining [3, 9, 28]. These eigenpairs can be computed with a Lassere type semidefinite programming hierarchy [8, 21, 22], although the scalability of such methods is limited. Thus, we still lack robust and scalable general-purpose methods for computing these eigenvectors.

We introduce two special cases of tensor contractions that will be useful:

1. The *tensor apply* takes a cubic tensor and a vector and produces a vector:

$$\begin{array}{lll} \text{three-mode tensor} & \mathbf{y} = \underline{\boldsymbol{T}}\mathbf{x}^2 & y_i = \sum_{j,k} \underline{T}_{i,j,k} x_j x_k \\ m\text{-mode tensor} & \mathbf{y} = \underline{\boldsymbol{T}}\mathbf{x}^{m-1} & y_i = \sum_{i_2,\ldots,i_n} \underline{T}_{i_1,\ldots,i_m} x_{i_2} \cdots x_{i_m} \end{array}$$

2. The *tensor collapse* takes a cubic tensor and a vector and produces a matrix:

$$\begin{array}{lll} \text{three-mode tensor} & \boldsymbol{Y} = \underline{\boldsymbol{T}}[\mathbf{x}]^1 & \boldsymbol{Y} = \sum_k \underline{T}_{:,:,k} x_k \\ & & Y_{ij} = \sum_k \underline{T}_{i,j,k} x_k \\ m\text{-mode tensor} & \boldsymbol{Y} = \underline{\boldsymbol{T}}[\mathbf{x}]^{m-2} & \boldsymbol{Y} = \sum_{i_3,\ldots,i_m} \underline{T}_{:,:,i_3,\ldots i_m} x_{i_3} \cdots x_{i_m} \\ & & Y_{ij} = \sum_{i_3,\ldots,i_m} \underline{T}_{i,j,i_3,\ldots,i_m} x_{i_3} \cdots x_{i_m}. \end{array}$$

For the tensor collapse operator, the ":" symbol signifies taking all entries along that index, so $\underline{T}_{:,:,k}$ is a square matrix. We note that the tensor may not be symmetric, but we are always contracting onto the first mode (tensor apply) or first and second modes (tensor collapse). We assume that $\underline{\boldsymbol{T}}$ has been permuted in the appropriate manner for the problem at hand. With this notation, it is easy to see that the tensor Z-eigenvector problem reduces to the following analog of (1.1):

$$\underline{\boldsymbol{T}}\mathbf{x}^{m-1} = \lambda\mathbf{x}, \quad \|\mathbf{x}\|_2 = 1. \tag{1.2}$$

The crux of our computational method is based on the following observation that relates tensor and matrix eigenvectors.

Observation 1.1. A *tensor Z-eigenvector* $\mathbf{x}$ of an m-mode tensor must be a *matrix eigenvector* of the collapsed matrix $\underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}$, i.e.,

$$\underline{\boldsymbol{T}}\mathbf{x}^{m-1} = \lambda\mathbf{x} \iff \underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}\mathbf{x} = \lambda\mathbf{x}. \tag{1.3}$$

The trick, of course, is that the matrix itself depends on the tensor eigenvector we want to compute. Therefore, we still have a nonlinear problem.

2. A new method for computing tensor Z-eigenvectors. Observation 1.1 provides a different perspective on the tensor Z-eigenvector problem. Specifically, the tensor Z-eigenvector problem is actually a matrix eigenvector problem, but for some unknown matrix. Our computational approach is based on the following continuous-time dynamical system, for reasons that we will make clear in Subsection 2.2:

$$\frac{d\mathbf{x}}{dt} = \Lambda(\underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}) - \mathbf{x}, \tag{2.1}$$

where Λ is some fixed map that takes as input a matrix and produces as output some prescribed eigenvector of the matrix with unit norm. For a matrix $\boldsymbol{M}$, Λ could be defined to compute several objects:

1. the eigenvector of $\boldsymbol{M}$ with smallest/largest magnitude eigenvalue
2. the eigenvector of $\boldsymbol{M}$ with smallest/largest algebraic eigenvalue
3. the eigenvector of $\boldsymbol{M}$ with kth smallest/largest magnitude eigenvalue
4. the eigenvector of $\boldsymbol{M}$ with kth smallest/largest algebraic eigenvalue

```julia
function tensor_apply(T::Array{Float64,3}, x::Vector{Float64})
    n = length(x)
    y = zeros(Float64, n)
    for k in 1:n; y += T[:, :, k] * x * x[k]; end
    return y
end

function tensor_collapse(T::Array{Float64,3}, x::Vector{Float64})
    n = length(x)
    Y = zeros(Float64, n, n)
    for k in 1:n; Y += T[:, :, k] * x[k]; end
    return Y
end

function Z_evec_dynsys_forward_euler(T::Array{Float64,3},
                                     h::Float64, niter::Int64=20)
    function f(u::Vector{Float64})  # Derivative
        Y = tensor_collapse(T, u)
        d, V = eig(Y)
        ind = sortperm(abs.(real(d)))[1]
        v = V[:, ind]
        return sign(v[1]) * v - u  # sign consistency
    end
    x = normalize(ones(Float64, size(T, 1)))  # starting point
    lam_hist = [x' * tensor_apply(T, x)]
    for _ = 1:niter
        x += h * f(x)
        push!(lam_hist, x' * tensor_apply(T, x))
    end
    return (x, lam_hist)  # guess at evec and history of evals
end
```

FIG. 1. *Julia implementation of the dynamical system for a 3-mode tensor based on computing the largest magnitude real eigenvalue and numerical integration with the forward Euler method. Code snippet is available at https://gist.github.com/arbenson/f28d1b2de9aa72882735e1be24d05a7f.*

We resolve the ambiguity in the sign of the eigenvector by picking the sign based on the first element. In the case of multiple eigenvectors sharing an eigenvalue, we propose using the closest eigenvector to $\mathbf{x}$, although we have not evaluated this technique.

PROPOSITION 2.1. *Let Λ be a prescribed map from a matrix to one of its eigenvectors. Then if the dynamical system in (2.1) converges to a non-zero solution, it must converge to a tensor Z-eigenvector.*

Proof. If the dynamical system converges, then it converges to a stationary point. Any stationary point has zero derivative, so

$$\frac{d\mathbf{x}}{dt} = 0 \iff \Lambda(\underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}) = \mathbf{x} \iff \underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}\mathbf{x} = \lambda\mathbf{x} \text{ for some } \lambda \text{ that depends on } \Lambda$$
$$\iff \underline{\boldsymbol{T}}\mathbf{x}^{m-1} = \lambda\mathbf{x}. \qquad \square$$

One must be a bit careful with the input and output values of Λ. If $\underline{\boldsymbol{T}}$ is not symmetric, then $\underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}$ might not be diagonalizable; thus we may have to deal with complex eigenvalues. To keep the dynamical system real-valued, one could always modify the map Λ to output the real part. However, the tensor need not be symmetric (nor $\underline{\boldsymbol{T}}[x]^{m-2}$ normal for all $\mathbf{x}$) for the dynamical system to maintain real values. In fact, our motivation for this dynamical system comes from a tensor that is not necessarily symmetric, which we will discuss in Subsection 2.2.

Proposition 2.1 leads to a broad framework for computing Z-eigenvectors:

1. Choose a map Λ and a numerical integration scheme.
2. Numerically integrate (2.1).

Different choices of Λ may converge to different Z-eigenvectors and different numerical integration schemes may lead to different convergence properties. Figure 1 shows a concrete example, where Λ picks the eigenvector corresponding to the largest eigenvalue in magnitude and the integration scheme is the forward Euler method.

2.1. Forward Euler convergence with diagonal tensors. As an illustrative example, we consider the special case of using the forward Euler numerical integration scheme for computing the tensor eigenvalues of an n-dimensional, m-mode diagonal tensor $\underline{\boldsymbol{T}}$. Without loss of generality, assume that the diagonal entries of $\underline{\boldsymbol{T}}$ are decreasing so that ordered so that $\underline{T}_{i,\ldots,i} < \underline{T}_{j,\ldots,j}$ if $i > j$. This tensor has n Z-eigenpairs: $(\mathbf{e}_i, \underline{T}_{i,\ldots,i})$ for $1 \leq i \leq n$, where $\mathbf{e}_i$ is the ith standard basis vector. Suppose that we want to compute the ith eigenvector and set Λ to map $\underline{\boldsymbol{T}}[\mathbf{x}]^{m-2}$ to the i largest algebraic eigenvalue (with unit norm). Let $\mathbf{r}_k = \mathbf{x}_k - \mathbf{e}_i$ be the residual at the kth iteration. If the step size is h, then

$$\|\mathbf{r}_{k+1}\| = \|\mathbf{x}_{k+1} - \mathbf{e}_i\| = \|\mathbf{x}_k + h(\mathbf{e}_i - \mathbf{x}_k) - \mathbf{e}_i\| \tag{2.2}$$

$$= (1-h)\|\mathbf{x}_k - \mathbf{e}_i\| = (1-h)\|\mathbf{r}_k\| = (1-h)^k\|\mathbf{r}_0\|. \tag{2.3}$$

Thus, the forward Euler scheme converges if $h \leq 1$ and converges in one step if $h = 1$.

2.2. Motivating the dynamical system with spacey random walks. The motivation for the dynamical system comes from our previous analysis of a stochastic process known as the "spacey random walk" that relates tensor eigenvectors of a particular class of tensors to a stochastic process [4]. Specifically, the class of tensors are irreducible *transition probability tensors* (any tensor $\underline{\boldsymbol{P}}$ with $\sum_{i_1=1}^{n} \underline{P}_{i_1,i_2,\ldots,i_m} = 1$ for $1 \leq i_2, \ldots, i_m \leq n$). For simplicity, we will now discuss a three-mode transition probability tensor $\underline{\boldsymbol{P}}$, where the entries can be interpreted as coming from a second-order Markov chain; the entry $\underline{P}_{i,j,k}$ is the probability of transitioning to state i given that the last two states were j and k. Due to the theory of Li and Ng [17], there exists a tensor Z-eigenvector $\mathbf{x}$ with eigenvalue 1 satisfying

$$\underline{\boldsymbol{P}}\mathbf{x}^2 = \mathbf{x}, \quad \textstyle\sum_i x_i = 1, \quad x_i \geq 0. \tag{2.4}$$

The vector $\mathbf{x}$ is stochastic, but it does *not* represent the stationary distribution of a Markov chain. Instead, we showed that $\mathbf{x}$ is the limiting distribution of a non-Markovian, generalized vertex-reinforced random walk [2] that we call the *spacey random walk* [4]. In the nth step of a spacey random walk, after the process has visited states $X_1, \ldots, X_n$, it *spaces out* and forgets its second last state (that is, the state X_{n-1}). It then invents a new history state Y_n by randomly drawing a past state $X_1, \ldots, X_n$. Finally, it transitions to X_{n+1} via the second-order Markov chain represented by $\underline{\boldsymbol{P}}$ as if its last two states were X_n and Y_n, i.e., it transitions to X_{n+1} with probability $\underline{P}_{X_{n+1},X_n,Y_n}$. (In contrast, a true second-order Markov chain would transition with probability $\underline{P}_{X_{n+1},X_n,X_{n-1}}$.)

Using results from Benaïm [2], we showed that the long-term dynamics of the spacey random walk for an m-mode transition probability tensor are governed by the following dynamical system [4]:

$$\frac{d\mathbf{x}}{dt} = \Pi(\underline{\boldsymbol{P}}[\mathbf{x}]^{m-2}) - \mathbf{x}, \tag{2.5}$$

where Π is a map that takes a column-stochastic transition matrix and maps it to the Perron vector of the matrix. In other words, if the spacey random walk converges, it must converge to an attractor of the dynamical system in (2.5). The dynamical system in (2.5) is a special case of the more general system in (2.1), where the map Λ picks the eigenvector with largest algebraic eigenvalue (the Perron vector), and the tensor has certain structural properties (it is a transition probability tensor).

To summarize, our prior work studied a specific case of the general dynamical system in (2.1) to understand the stochastic process behind principal Z-eigenvectors of transition probability tensors. The general dynamical system provides a new framework for computing general tensor eigenvectors—*if* the dynamical system in (2.1) converges, then it converges to a tensor Z-eigenvector. The dynamical system may not converge [23], but it usually does in practice, as we will see in section 3.

2.3. Relationship with the shifted higher-order power method. The shifted higher-order power method [14] can be derived by noticing that

$$(1+\gamma)\lambda\mathbf{x} = \underline{\boldsymbol{T}}\mathbf{x}^{m-1} + \gamma\lambda\mathbf{x} \tag{2.6}$$

for any eigenpair. This yields the iteration

$$\mathbf{x}_{k+1} = \frac{\frac{1}{1+\gamma}\left(\underline{\boldsymbol{T}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right)}{\|\frac{1}{1+\gamma}\left(\underline{\boldsymbol{T}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right)\|} \tag{2.7}$$

for any shift parameter γ (the case where $\gamma = 0$ is just the "higher-order power method" [12, 16, 26]). Kolda and Mayo showed that when $\underline{\boldsymbol{T}}$ is symmetric, the iterates in (2.7) converge monotonically to a tensor eigenvector given an appropriate shift γ.

If $\underline{\boldsymbol{T}} = \underline{\boldsymbol{P}}$ for some transition probability tensor $\underline{\boldsymbol{P}}$ and we are interested in the case when $\lambda = 1$, then one can also derive these iterates by the dynamical system

$$\frac{d\mathbf{x}}{dt} = \underline{\boldsymbol{P}}\mathbf{x}^{m-1} - \mathbf{x} \tag{2.8}$$

(c.f. (2.5)). If this dynamical system converges ($d\mathbf{x}/dt = 0$), then $\mathbf{x} = \underline{\boldsymbol{P}}\mathbf{x}^{m-1}$, and $\mathbf{x}$ is a tensor Z-eigenvector with eigenvalue 1. If we numerically integrate (2.8) using the forward Euler method with step size $h = 1/(1+\gamma)$ and any starting vector $\mathbf{x}_0$ satisfying $\mathbf{x}_0 \geq 0$ and $\|\mathbf{x}_0\|_1 = 1$, then the iterates are

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{1+\gamma}\left(\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} - \mathbf{x}_k\right) \tag{2.9}$$

$$= \frac{1}{1+\gamma}\left(\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right) = \frac{\frac{1}{1+\gamma}\left(\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right)}{\|\frac{1}{1+\gamma}\left(\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right)\|_1}, \tag{2.10}$$

which are the same as the shifted higher-order power method iterates in (2.7). The last equality comes from the fact that $\|\mathbf{x}_k\| = 1$ and $\mathbf{x}_k \geq 0$ by a simple induction argument: the base case holds by the initial conditions and

$$\|\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\|_1 = \|\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1}\|_1 + \gamma = 1 + \gamma \tag{2.11}$$

since $\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1}$ and $\mathbf{x}_k$ are both stochastic vectors.

For a general tensor $\underline{\boldsymbol{T}}$, we can use the dynamical system

$$\frac{d\mathbf{x}}{dt} = \underline{\boldsymbol{T}}\mathbf{x}^{m-1} - \|\mathbf{x}\|_2\mathbf{x}. \tag{2.12}$$

If $d\mathbf{x}/dt = 0$, then $\|\mathbf{x}\|_2\mathbf{x} = \underline{\boldsymbol{T}}\mathbf{x}^{m-1}$, so $\mathbf{x}$ is a Z-eigenvector of $\underline{\boldsymbol{T}}$ with eigenvalue $\|\mathbf{x}\|_2$. Now suppose that we numerically integrate the dynamical system in (2.12) by

1. taking a forward Euler step to produce the iterate $\mathbf{x}'_{k+1}$; and
2. projecting $\mathbf{x}'_{k+1}$ onto the unit sphere by $\mathbf{x}_{k+1} = \mathbf{x}'_{k+1}/\|\mathbf{x}'_{k+1}\|_2$.

λ	Type	S-HOPM	SS-HOPM	V1	V2	V3	V4	V5
0.0180	Neg. stable	0	18	0	25	0	100	0
0.4306	Neg. stable	38	29	38	0	45	0	0
0.8730	Neg. stable	62	40	62	0	47	0	0
0.0006	Pos. stable	0	13	0	19	8	0	0
0.0018	Unstable	0	0	0	25	0	0	32
0.0033	Unstable	0	0	0	35	0	0	37
0.2294	Unstable	0	0	0	0	0	0	31

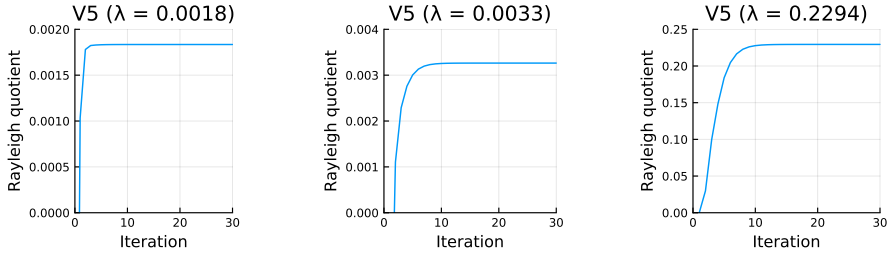

FIG. 2. ***(Top)*** *The 7 eigenvalues of the test tensor from Kolda and Mayo [14, Example 3.6] and the number of random trials (out of 100) that converge to the eigenvalue for (i) the symmetric higher-order power method; S-HOPM [12, 16, 26], (ii) the shifted symmetric higher-order power method; SS-HOPM [14]), and (iii) 5 variations of our dynamical systems approach. V1 selects the largest magnitude eigenvalue, V2 selects the smallest magnitude eigenvalue, V3 selects the largest algebraic eigenvalue, V4 selects the smallest algebraic eigenvalue, and V5 selects the second smallest algebraic eigenvalue. Our algorithm is the only one that is able to compute all of the eigenvalues, including those which are "unstable", the eigenvectors of which SS-HOPM and S-HOPM will not be converge to [14].* ***(Bottom)*** *Convergence plots for the three unstable eigenvalues from variation 5 of our algorithm in terms of the Rayleigh quotient* $\mathbf{x}_k^T \underline{\boldsymbol{T}} \mathbf{x}_k^{m-1}$*, where* $\mathbf{x}_k$ *is the kth iterate.*

If the step size of the forward Euler method is $h = 1/(1+\gamma)$, then

$$\mathbf{x}'_{k+1} = \mathbf{x}_k + \frac{1}{1+\gamma}(\underline{\boldsymbol{T}}\mathbf{x}_k^{m-1} - \|\mathbf{x}_k\|_2\mathbf{x}_k) = \frac{1}{1+\gamma}\left(\underline{\boldsymbol{P}}\mathbf{x}_k^{m-1} + \gamma\mathbf{x}_k\right) \tag{2.13}$$

since $\|\mathbf{x}_k\|_2 = 1$. The projection onto the unit sphere then gives the shifted higher-order power method iterates in (2.7).

3. Numerical examples. We now demonstrate our method on two tensors used in prior work. Subsection 3.1 shows that our approach can compute all eigenvalues of a specific tensor, while the (shifted) higher-order power method cannot compute all of the eigenvalues. Subsection 3.2 shows verifies that our approach can compute all eigenvalues of a tensor whose eigenvalues were found with semi-definite programming.

3.1. Example 3.6 from Kolda and Mayo [14]. Our first test case is a $3\times3\times3$ symmetric tensor from Kolda and Mayo [14, Example 3.6]:

$$\underline{T}_{:,:,1} = \begin{bmatrix} -0.1281 & 0.0516 & -0.0954 \\ 0.0516 & -0.1958 & -0.179 \\ -0.0954 & -0.179 & -0.2676 \end{bmatrix}, \quad \underline{T}_{:,:,2} = \begin{bmatrix} 0.0516 & -0.1958 & -0.179 \\ -0.1958 & 0.3251 & 0.2513 \\ -0.179 & 0.2513 & 0.1773 \end{bmatrix}$$

$$\underline{T}_{:,:,3} = \begin{bmatrix} -0.0954 & -0.179 & -0.2676 \\ -0.179 & 0.2513 & 0.1773 \\ -0.2676 & 0.1773 & 0.0338 \end{bmatrix}.$$

The tensor has 7 eigenvalues, which Kolda and Mayo classify as "positive stable", "negative stable", or "unstable" (see Figure 2, top), corresponding to positive definiteness,

λ	SDP	V1	V2	V3	V4	V5
9.9779	✓	94	0	0	100	0
4.2876	✓	6	0	100	0	0
0.0000	✓	0	100	0	0	100

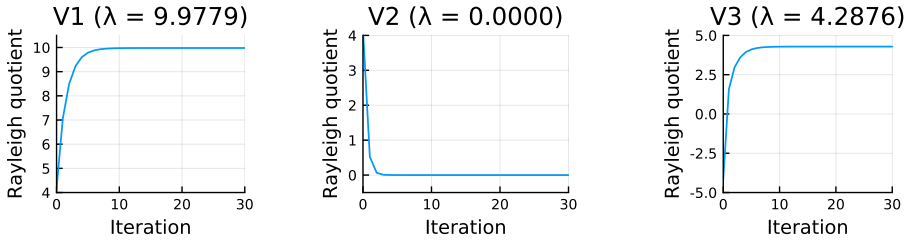

FIG. 3. ***(Top)*** *The 3 eigenvalues of the test tensor from from Cui, Dai, and Nie [8, Example 4.11] and the number of random trials (out of 100) that converge to the eigenvalue for 5 variations of our dynamical systems approach. Variation 1 selects the largest magnitude eigenvalue, variation 2 selects the smallest magnitude eigenvalue, variation 3 selects the largest algebraic eigenvalue, variation 4 selects the smallest algebraic eigenvalue, variation 5 selects the 2nd smallest algebraic eigenvalue. Our algorithm is able to compute all of the eigenvalues, which the SDP approach is guaranteed to compute [8].* ***(Bottom)*** *Convergence plots for the three eigenvalues from different variations of our algorithm in terms of the Rayleigh quotient* $\mathbf{x}_k^T \underline{\boldsymbol{T}} \mathbf{x}_k^{m-1}$*, where* $\mathbf{x}_k$ *is the kth iterate.*

negative definiteness, or indefiniteness of the projected Hessian of the Lagrangian of their optimization function [14]. (Note that since the tensor has odd number of modes, eigenvalues are only defined up to sign.) Kolda and Mayo showed that their shifted symmetric higher-order power method (SS-HOPM), a generalization of the symmetric higher-order power method (S-HOPM) [12, 16, 26], only converges to eigenvectors of the positive or negative stable eigenvalues.

Of the 7 eigenpairs for the above tensor, 3 are unstable. Our dynamical systems approach can compute all 7 eigenpairs, using 5 variations of the dynamical system:

1. Λ maps $\boldsymbol{M}$ to the eigenvector with largest magnitude eigenvalue;
2. Λ maps $\boldsymbol{M}$ to the eigenvector with smallest magnitude eigenvalue;
3. Λ maps $\boldsymbol{M}$ to the eigenvector with largest algebraic eigenvalue;
4. Λ maps $\boldsymbol{M}$ to the eigenvector with smallest algebraic eigenvalue; and
5. Λ maps $\boldsymbol{M}$ to the eigenvector with second smallest algebraic eigenvalue.

We used the forward Euler method with step size set to 0.5 in order to compute the eigenvalues. Empirically, convergence is fast, requiring 10 or fewer iterations (see the bottom row of Figure 2). We note that one can also also compute these eigenvectors with a Lassere type semidefinite programming hierarchy [8, 21, 22], although the scalability of such methods is limited. We provide numerical results from a tensor in this literature in the next section.

3.2. Example 4.11 from Cui, Dai, and Nie [8]. Our second test case is a $5 \times 5 \times 5$ symmetric tensor from Cui, Dai, and Nie [8, Example 4.11]:

$$\underline{T}_{i,j,k} = \frac{(-1)^i}{i} + \frac{(-1)^j}{j} + \frac{(-1)^k}{k}, \; 1 \leq i, j, k \leq 5.$$

The tensor has 3 eigenvalues (again, the tensor has an odd number of modes, so the eigenvalues are only defined up to sign). We use the same 5 variations of our algorithm to compute the eigenpairs (Figure 3). Again, we are able to compute all of the eigenvalues of the tensor, and convergence is rapid.

4. Discussion. Scalable methods for computing tensor eigenvectors remain a challenge. Our new framework for computing Z-eigenvectors offers insights through two observations. First, a tensor Z-eigenvector is a matrix eigenvector of *some* matrix. However, the matrix is obtained by applying the tensor collapse operator with the Z-eigenvector itself. Second, for a certain class of tensors where eigenvectors have a stochastic interpretation, the dynamical system in (2.1) is the one that governs the long-term dynamics of the stochastic process. Nevertheless, the same dynamical system seems to work for more general tensors. In fact, we can compute tensor eigenvectors that the (shifted) higher-order power method cannot. However, one major challenge is knowing what map Λ to choose—different choices lead to different eigenvectors and there is no immediate relationship between them for general tensors. The structure of the tensor can offer some guidance; for the special case of a transition probability tensor $\underline{\boldsymbol{P}}$ and eigenvalue $\lambda = 1$, the right map is the one that outputs the Perron vector of a column stochastic matrix. Finally, our method is not immediately applicable to H-eigenvectors because Observation 1.1 no longer holds. Adapting our methodology to this class of eigenvectors is an area for future research.

Our framework came from relating tensor eigenvectors to stochastic processes. This is quite different from the core ideas in the tensor literature, which are firmly rooted in algebraic generalizations. We hope that our ideas encourage the use of stochastics and other areas of applied mathematics in solving difficult tensor problems.

Acknowledgments. We thank Brad Nelson for providing valuable feedback. ARB is supported by NSF TRIPODS Award #1740822. DFG is supported by NSF award CCF-1149756, IIS-1422918, IIS-1546488, the NSF Center for Science of Information STC, CCF-0939370, DARPA SIMPLEX, and the Sloan Foundation.

REFERENCES

[1] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[2] M. Benaïm. Vertex-reinforced random walks and a conjecture of Pemantle. *The Annals of Probability*, 25(1):361–392, 1997.

[3] A. R. Benson, D. F. Gleich, and J. Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126, 2015.

[4] A. R. Benson, D. F. Gleich, and L.-H. Lim. The spacey random walk: a stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017.

[5] D. A. Bini, B. Meini, and F. Poloni. On the solution of a quadratic vector equation arising in markovian binary trees. *Numerical Linear Algebra with Applications*, 18(6):981–991, 2011.

[6] D. Cartwright and B. Sturmfels. The number of eigenvalues of a tensor. *Linear Algebra and its Applications*, 438(2):942–952, 2013.

[7] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279, 2008.

[8] C.-F. Cui, Y.-H. Dai, and J. Nie. All real eigenvalues of symmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1582–1601, 2014.

[9] D. F. Gleich, L.-H. Lim, and Y. Yu. Multilinear PageRank. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1507–1541, 2015.

[10] S. Hu, L. Qi, and G. Zhang. Computing the geometric measure of entanglement of multipartite pure states by means of non-negative tensors. *Physical Review A*, 93(1), 2016.

[11] E. Kofidis and P. A. Regalia. Tensor approximation and signal processing applications. *Contemporary Mathematics*, 280:103–134, 2001.

[12] E. Kofidis and P. A. Regalia. On the best rank-1 approximation of higher-order supersymmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(3):863–884, 2002.

[13] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[14] T. G. Kolda and J. R. Mayo. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1095–1124, 2011.

[15] L. Lathauwer. *Signal processing based on multilinear algebra*. PhD thesis, Katholieke Universiteit Leuven, 1997.

[16] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the Best Rank-1 and Rank-(R_1, R_2, ..., R_N) Approximation of Higher-Order Tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[17] W. Li and M. K. Ng. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra*, Online:1–24, 2013.

[18] L.-H. Lim. Singular values and eigenvalues of tensors: a variational approach. In *1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 129–132, 2005.

[19] L.-H. Lim. Tensors and hypermatrices. In *Handbook of Linear Algebra, Second Edition*, chapter 15, pages 231–260. Chapman and Hall/CRC, 2013.

[20] B. Meini and F. Poloni. A perron iteration for the solution of a quadratic vector equation arising in markovian binary trees. *SIAM Journal on Matrix Analysis and Applications*, 32(1):248–261, 2011.

[21] J. Nie and L. Wang. Semidefinite relaxations for best rank-1 tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 35(3):1155–1179, 2014.

[22] J. Nie and X. Zhang. Real eigenvalues of nonsymmetric tensors. *Computational Optimization and Applications*, 70(1):1–32, 2017.

[23] J. Peterson. Personal communication, 2018.

[24] L. Qi. Eigenvalues of a real supersymmetric tensor. *J. Symb. Comput.*, 40(6):1302–1324, 2005.

[25] L. Qi, Y. Wang, and E. X. Wu. d-eigenvalues of diffusion kurtosis tensors. *Journal of Computational and Applied Mathematics*, 221(1):150–157, 2008.

[26] P. A. Regalia and E. Kofidis. The higher-order power method revisited: convergence proofs and effective initialization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2000.

[27] T.-C. Wei and P. M. Goldbart. Geometric measure of entanglement and applications to bipartite and multipartite quantum states. *Physical Review A*, 68(4), 2003.

[28] T. Wu, A. Benson, and D. F. Gleich. General tensor spectral co-clustering for higher-order data. In *Advances in Neural Information Processing Systems*, pages 2559–2567, 2016.