Time-Varying Matrix Eigenanalyses via Zhang Neural Networks and Look-Ahead Finite Difference Equations

Frank Uhlig
Department of Mathematics and Statistics,
Auburn University, AL 36849-5310, USA (uhligfd@auburn.edu)

Yunong Zhang
School of Information Science and Technology,
Sun Yat-sen University (SYSU),
Guangzhou 510006, China (zhynong@mail.sysu.edu.cn)

Abstract: This paper adapts look-ahead and backward finite difference formulas to compute future eigenvectors and eigenvalues of piecewise smooth time-varying symmetric matrix flows $A(t)$. It is based on the Zhang Neural Network (ZNN) model for time-varying problems and uses the associated error function $E(t) = A(t)V(t) - V(t)D(t)$ or $e_i(t) = A(t)v_i(t) - \lambda_i(t)v_i(t)$ with the Zhang design stipulation that $\dot{E}(t) = -\eta E(t)$ or $\dot{e}_i(t) = -\eta e_i(t)$ with $\eta > 0$ so that $E(t)$ and $e(t)$ decrease exponentially over time. This leads to a discrete-time differential equation of the form $P(t_k)\dot{z}(t_k) = q(t_k)$ for the eigendata vector $z(t_k)$ of $A(t_k)$. Convergent look-ahead finite difference formulas of varying error orders then allow us to express $z(t_{k+1})$ in terms of earlier $A$ and $z$ data. Numerical tests, comparisons and open questions complete the paper.

Subject Classifications: 65H17, 65L12, 65F15, 65Q10, 92B20

Key Words: matrix eigenvalues, matrix computations, symmetric matrix flows, Zhang neural network, time-varying equations, time-varying computations, 1-step ahead discretization formula, error function, numerical analysis, numerical linear algebra

1 Introduction

Many numerical methods in most every branch of applied mathematics employ matrices, vectors and spatial or time-dependent models to solve specific problems. Static-time and time-varying problems sometimes behave very differently as far as their numerics are concerned. Therefore time-invariant and time-varying problems may require different approaches when they deal with different challenges and their own inherent computational limitations.

Time-varying numerical matrix matrix and ZNN methods are a relatively new subject and differ greatly from our classical numerical canon and textbooks. The very words neural networks have multiple uses in Numerics and simply refer to the natural propagation of impulses that are passed along nervous systems. ZNN methods differ substantially from every other so called neural network, see e.g. [18], and from any other standard ODE based method, see e.g. [10], or from decomposition methods, see e.g. [5].

ZNN methods have two main and differing ingredients: the first is the error equation and the ZNN stipulation that the error equation is to decay exponentially fast to zero over time. Once this new, quite unusual and nowhere else used error function ODE has been discretized in ZNN, the second novel part of ZNN is its subsequent reliance on 1-step ahead convergent finite difference equations, rather than on standard ODE IVP solvers. Such look-ahead difference schemes have never occurred in any of our finite difference uses or the literature. The first ones were in fact constructed by hand on scratch paper within the last couple of years, see [12] and [14]. These first tries achieved rather low truncation error orders. A formal process to obtain higher error order 1-step ahead convergent difference schemes appears in [17].

There are several hundred papers with time-varying Zhang type methods in engineering journals, but relatively few, not even a handful, in the Numerical Linear Algebra literature.

For over a dozen years now, a special class of dynamic methods has been built on the idea of Yunong Zhang and Jun Wang [25] from 2001. These are so called Zhang dynamics (ZD), or zeroing dynamics; see [23] [24] [6] [19] and
Zhang neural networks (ZNN), or zeroing neural networks; see [28, 7, 20]. ZD methods are specifically designed for time-varying problems and have proven most efficient there. They use 1st-order time derivatives and have been applied successfully to – for example – solve time-varying Sylvester equations [22, 9, 8], to find time-varying matrix inverses [21, 7, 3, 12] (see also [4]), and to optimize time-varying matrix minimization problems [26, 11], all in real-time. These algorithms generally use both, 1-step ahead and backward differentiation formulas and run in discrete time with high accuracy. From given time-varying matrix or vector valued problem, all ZNN methods form a problem specific error function \( e(t) \) that is stipulated to decrease exponentially to zero, both globally and asymptotically [24, 28, 29] by the ODE \( \dot{e}(t) = -\eta e(t) \) for a decay constant \( \eta > 0 \). This discretized time-varying error matrix ODE problem is then solved successfully through finite difference formulas and simple recursive vector additions without the use of any standard numerical matrix factorizations, local iterative methods, or ODE solvers, except possibly for computing the necessary starting values.

The ZNN method is a totally new branch of Numerical Analysis. It is essential for robot control, self-driving vehicles, and autonomous airplanes et cetera where its ability to predict future systems states accurately is an essential ingredient for engineering success. ZNN’s numerical findings and open challenges are many and most of these have not even been recognized or even named in their own right.

In this paper we extend the ZD method for the recently studied time-varying matrix eigenvalue problem [27] that dealt with convergence and robustness issues of the DE based ZNN model solution. Here we propose a new symmetric discrete-time ZD model that computes all eigenvalues and eigenvectors of time-varying real symmetric matrix flows \( A(t) \), be they repeating or not. Our model uses convergent look-ahead and standard backward finite difference formulas instead of an ODE solver as was done in [27] for example.

Previous efforts to find the eigenstructure of time-varying matrix flows, mostly via DE solvers and path following continuations go back at least to 1991 [1] where the smoothness of the eigendata and SVD data of a time-dependent matrix flow was explored in detail. A few years later, Mailybeav [13] studied problems with near identical eigenvalues for time-varying general matrix computations. Matrix flows with coalescing eigenvalues pose crucial problems for these methods. These problems were observed and further studied by Dieci and Eirola in 1999 [5] and more recently by Špirkić and Kressner [15] in 2016 and they likewise occur for a related parameter-varying matrix eigen problem of Loisel and Maxwell in 2018 [10]. Our ZNN eigenvalue algorithm is impervious to these restrictions as it handles repeated eigenvalues in symmetric matrix flows \( A(t) \) without any problems, see Figures 1, 5, and 8.

The new ZD model of this paper is a discrete dynamical system with the potential for practical real-time and on-chip implementation and computer simulations. We include results of computer simulations and numerical experiments that illustrate the usefulness and efficiency of our real-time discrete ZD matrix eigenvalue algorithm, both for smooth data inputs and also for piecewise smooth time-varying matrix flows \( A(t) \) that might occur naturally when sensors fail or data lines get disrupted in the field.

2 A Zhang Neural Network Formulation for the Time-Varying Matrix Eigenvalue Problem

Computing the eigenvalues and eigenvectors of matrices with known fixed entries, i.e., of static matrices, has presented mathematicians with difficult problem from the early 1800s on. In the early 20th century, engineers such as Krylov thought of vector iteration and built iterative methods for finding matrix eigenvalues. Iterative matrix methods increased in sophistication with the advent of computers in the 1950s and are now very successful to solve static matrix problems for a multitude of dense or large structured and sparse matrices. For dense matrices the eigenvalue problem was essentially left unsolved for 150 years until John Francis and Vera Kublanovskaja independently created the QR algorithm around 1960.

While it is nowadays quite easy to solve the eigenvalue problem \( Ax = \lambda x \) for fixed entry matrices \( A \) and find their eigenvectors \( x \) and eigenvalues \( \lambda \), what happens if the entries of \( A \) vary over time? Naively, one could solve the associated time-varying eigenvalue equation \( A(t)x(t) = \lambda(t)x(t) \) for multiple times \( t_k \) and take the computed eigendata output for \( A(t_k) \) as an approximate solution of the time-varying eigen problem at time \( t_{k+1} \). This might work if \( A(t) \) is explicitly known for all \( t_0 \leq t \leq t_{	ext{end}} \) and if we can wait for the end of the computations for \( A(t_k) \) to see the results for the past time \( t_k \). But if we need to know the eigenstructure of \( A(t_{k+1}) \) from the previous be-
Behavior of $A(t)$ before time $t_{k+1}$ has arrived, such as for robot optimization processes etc, the naive method would compute and deliver the ’exact’ solution for an instance in the past, one that may have little or no relation to the current time situation. And besides, how would the delayed solution maneuver discontinuities in $A(t)$ in real-time or when large entry value swings occur in $A(t)$? Therefore to solve time-varying matrix eigenvalue problems reliably in real-time we need to learn how to predict the future value of the eigenvalues and eigenvectors of $A(t)$ for the time instance $t_{k+1}$ solely from earlier and past $A(t_j)$ data instances $t_j$ with $t_j < t_{k+1}$. This challenge is rather new and has not been much explored in numerical analysis.

Our best static matrix eigenvalue methods use orthogonal matrix factorizations and these are backward stable. Backward stability gives us the exact solution to an adjacent problem whose distance from the given problem is bounded by the eigenvalue condition numbers of the given matrix. For time-varying matrix problems, we apparently need to relax on backward stability and instead work on accurate forward predictions. Here is how this can be done by using Zhang Neural Network ideas for time-varying matrix eigenvalue problems.

**Statement of the Problem and the Zhang Neural Network Error Equation**

For a real symmetric flow of matrices $A(t) \in \mathbb{R}^{n,n}$ and $0 \leq t \leq t_f$, we consider the problem of finding non-singular real matrices $V(t) \in \mathbb{R}^{n,n}$ and real diagonal matrices $D(t) \in \mathbb{R}^{n,n}$ so that

$$A(t)V(t) = V(t)D(t) \quad \text{for all } t.$$

(1)

Since we assume $A(t)$ to be real symmetric at all times $t$, such matrices $V(t)$ and $D(t)$ will exist for all $t$. Our aim is to find them accurately and predictively in real-time. To solve (1), the Zhang Neural Network approach looks at the time-varying homogeneous error equation which has the form

$$e(t) = A(t)V(t) - V(t)D(t) = O_n \in \mathbb{R}^{n,n}.$$

(2)

Next, the ZNN approach stipulates exemplary behavior for $e(t)$ by asking for exponential decay of $e(t)$ as a function of time, or

$$\dot{e}(t) = -\eta e(t)$$

(3)

with a decay constant $\eta > 0$. Equation (3) can be written out explicitly as

$$\dot{e}(t) = \dot{A}(t)V(t) + A(t)\dot{V}(t) - \dot{V}(t)D(t) - V(t)\dot{D}(t) = -\eta A(t)V(t) + \eta V(t)D(t) = -\eta e(t),$$

(4)

$$A(t)\dot{V}(t) - \dot{V}(t)D(t) - V(t)\dot{D}(t) = -\eta A(t)V(t) + \eta V(t)D(t) - \dot{A}(t)V(t)$$

(5)

where in (5) we have gathered all terms with the derivatives $\dot{V}(t)$ of the unknown eigenvector matrix $V(t)$ and $\dot{D}(t)$ of the eigenvalue matrix $D(t)$ on the left hand side.

When we specify equation (5) for one eigenvalue/eigenvector pair $x_i(t)$ and $\lambda_i(t)$ of $A(t)$ and $i = 1,\ldots,n$ we obtain

$$A(t)x_i(t) - \lambda_i(t)x_i(t) = -\eta(A(t) - \lambda_i(t))x_i(t) + \lambda_i(t)x_i(t) - \dot{A}(t)x_i(t)$$

(6)

since scalars and vectors always commute, which was not the case for the $n$ by $n$ matrix equation (5).

Note that we do not know how to solve the full system eigenvalue equation (5) via ZNN directly. Upon rearranging terms in (6) we finally have

$$(A(t) - \lambda_i(t)I_n)x_i(t) - \dot{\lambda}_i(t)x_i(t) = (-\eta(A(t) - \lambda_i(t))I_n) - \dot{A}(t))x_i(t)$$

(7)

where $I_n$ is the identity matrix of the same size $n$ by $n$ as $A(t)$.

For each $i = 1,\ldots,n$ the last equation (7) is a differential equation in the unknown eigenvector $x_i(t) \in \mathbb{R}^n$ and the unknown eigenvalue $\lambda_i(t) \in \mathbb{R}$ which we rewrite in augmented matrix form by further rearrangement as

$$\begin{pmatrix}
A(t) - \lambda_i(t)I_n & -x_i(t) \\
2x_i(t)x_i^T(t) & 0
\end{pmatrix}
\begin{pmatrix}
x_i(t) \\
\dot{\lambda}_i(t)
\end{pmatrix}
= 
\begin{pmatrix}
(-\eta(A(t) - \lambda_i(t))I_n) - \dot{A}(t))x_i(t) \\
-\mu x_i^T(t)x_i(t) - 1
\end{pmatrix}$$

(8)

Here the second block row in (8) has been added below equation (7) by expanding the exponential decay differential equation for $e_2(t) = x_i^T(t)x_i(t) - 1$ that is meant to ensure unit eigenvectors $x_i(t)$ throughout for a separate decay
constant $\mu$.

Note that the leading $n$ by $n$ block matrix $A(t) - \lambda_i(t)I_n$ of the system matrix in equation (8) is symmetric for symmetric matrix flows $A(t)$. To help speed up our computations we transform the error differential equation $2x_i(t)^T\dot{x}_i(t) = -\mu \cdot (x_i^T(t)x_i(t) - 1)$ for $e_2$ slightly by dividing it by $-2$. That leads to the equivalent error function differential equation $-x_i(t)^T\dot{x}_i(t) = \mu/2 \cdot (x_i^T(t)x_i(t) - 1)$ for $e_2$. Thus for all symmetric matrix flows $A(t)$, we can replace the non-symmetric ZNN model (8) by its symmetric version

\[
\begin{pmatrix}
A(t) - \lambda_i(t)I_n & -x_i(t) \\
-x_i^T(t) & 0
\end{pmatrix}
\begin{pmatrix}
\dot{x}_i(t) \\
\lambda_i(t)
\end{pmatrix}
= \begin{pmatrix}
-\eta(A(t) - \lambda_i(t)I_n) - \dot{A}(t)x_i(t) \\
\mu/2 \cdot (x_i^T(t)x_i(t) - 1)
\end{pmatrix}.
\] (9)

The full eigenvalue and eigenvector problem (1) of time-varying symmetric matrix flows $A(t) \in \mathbb{R}^{n,n}$ can now be solved for each of its $n$ eigenpairs separately by using the matrix differential equation (9) with a symmetric system matrix for $i = 1, \ldots, n$ in turn.

In this paper we study and solve (1) with the help of convergent look-ahead difference schemes of varying truncation error orders for discrete, smooth and non-smooth symmetric time-varying matrix $A(t_k)$ inputs.

3 Solving the Zhang Neural Network Error Equation via Look-ahead and Backward Discretization Formulas

Our model (8) has recently been solved in real-time by using the standard ODE solver ode15s of MATLAB, see (27). In (27), model (8) was shown to be convergent and robust against data perturbations. ODE based solvers can unfortunately not be adapted easily to real-world sensor driven applications since they rely on intermediate data that may not be available with discrete-time sensor data. They work best for model testing of function valued time-varying $A(t)$ inputs. Thus there is a need to develop alternate discrete-time solvers such as ZNN methods that go beyond what was first explored in continuous-time in (27).

For discretized symmetric input data $A(t_k) = A(t_k)^T \in \mathbb{R}^{n,n}$ and $k = 0, 1, 2, \ldots$ it is most natural to discretize the differential equation (9) in sync with the sampling gap $\tau = t_{k+1} - t_k$ which we assume to be constant for all $k$. With choosing $\mu = \eta$ we have

\[
P(t_k) = \begin{pmatrix}
A(t_k) - \lambda_i(t_k)I_n & -x_i(t_k) \\
-x_i^T(t_k) & 0
\end{pmatrix} \in \mathbb{R}^{n+1,n+1}, \quad z(t_k) = \begin{pmatrix}
x_i(t_k) \\
\lambda_i(t_k)
\end{pmatrix} \in \mathbb{R}^{n+1},
\]

and

\[
q(t_k) = \begin{pmatrix}
-\eta(A(t_k) - \lambda_i(t_k)I_n) - \dot{A}(t_k)x_i(t_k) \\
\eta/2 \cdot (x_i^T(t)x_i(t) - 1)
\end{pmatrix} \in \mathbb{R}^{n+1}
\]

our model (9) becomes the set of matrix differential equations

\[
P(t_k)\dot{z}(t_k) = q(t_k)
\] (11)

for $k = 0, 1, 2, \ldots$, each equidistant discrete time step $0 \leq t_k \leq t_f$, and each eigenpair $x_i(t_k)$ and $\lambda_i(t_k)$ of $A(t_k)$. Note that $P(t_k)$ is always symmetric if the input matrix $A(t_k)$ is.

The Discretization Process

The differential equations (9) and (11) are equivalent. They each contain two derivatives: that of the unknown eigendata vector $z(t_k)$ and that of the input function $A(t_k)$. In our discretization we replace the derivative $\dot{A}(t_k)$ by the backward discretization formula

\[
\dot{A}_k = \frac{11A_k - 18A_{k-1} + 9A_{k-2} - 2A_{k-3}}{6\tau}
\] (12)

of error order $O(\tau^3)$ (2) p. 355] for example, where we have abbreviated $A(t_j)$ by writing $A_j$ for each $j$ and $\tau$ is the constant sampling gap from instance $t_j$ to $t_{j+1}$. Formula (12) is a four-instant backward difference formula.
Our look-ahead finite difference formula for \( z_k \) is the five-instant forward difference formula 5-IFD that we have adapted from \([12]\) for \( \dot{z}_k \) as

\[
\dot{z}_k = \frac{8z_{k+1} + z_k - 6z_{k-1} - 5z_{k-2} + 2z_{k-3}}{18\tau}.
\] (13)

It also has the truncation error order \( O(\tau^3) \). By replacing \( \dot{z}_k \) in \([11]\), multiplying equation (13) by \( 18\tau \) and reordering terms we obtain

\[
18\tau \cdot \dot{z}_k = 8z_{k+1} + z_k - 6z_{k-1} - 5z_{k-2} + 2z_{k-3} = 18\tau(P\backslash q).
\] (14)

where we have expressed the solution \( x \) of the linear system \( Px = q \) by the MATLAB symbol \( P\backslash q \). By multiplying equation (13) by \( 18\tau \), we have raised the truncation error order of the resulting difference equation (14) by one power of \( \tau \) to \( O(\tau^4) \). Solving equation (14) for \( z_{k+1} \) gives us the following look-ahead finite difference equation for the discretized time-varying matrix eigenvalue problem

\[
z_{k+1} = \frac{9}{4} \tau(P\backslash q) - \frac{1}{8}z_k + \frac{3}{4}z_{k-1} + \frac{5}{8}z_{k-2} - \frac{1}{4}z_{k-3}.
\] (15)

Completely written out in terms of the block entries of \( P \) and with \( q \) as defined in \([10]\), the last equation becomes

\[
z_{k+1} = -\frac{9}{4\tau} \left( \begin{pmatrix} A_k - \lambda_k I_n & -\dot{z}_k \\ -\dot{z}_k^T & 0 \end{pmatrix} \right) \left( \begin{pmatrix} \eta(A_k - \lambda_k I_n) + \frac{11A_k - 18A_{k-1} + 9A_{k-2} - 2A_{k-3}}{6\tau} & \eta/2 \cdot (\dot{z}_k^T \dot{z}_k - 1) \\ -\eta/2 \cdot (\dot{z}_k^T \dot{z}_k - 1) & 1/8z_k + \frac{3}{4}z_{k-1} + \frac{5}{8}z_{k-2} - \frac{1}{4}z_{k-3} \end{pmatrix} \right) \right)
\] (16)

Here \( \dot{z}_k = z^{(1, \ldots, n)} \) is the leading \( n \) entries of \( z_k \) \( \in \mathbb{R}^{n+1} \) and contains the current approximate unit eigenvector \( x_i(t_k) \) for the \( i \)th eigenvalue \( \lambda_i(t_k) \) of \( A(t_k) \). The five-instant forward difference formula 5-IFD \([13]\) was discovered, developed and analyzed extensively in \([12]\) sections 2.3, 2.4. There is was shown to be zero stable and consistent and hence convergent when used in multistep recursion schemes.

The same truncation error order as in \([13]\) is achieved by the following convergent six-instant forward difference formula 6-IFD of \([14]\) Sections III A and Theorems 2 and 3

\[
\dot{z}_k = \frac{13}{24\tau} z_{k+1} - \frac{1}{4\tau} z_k - \frac{1}{12\tau} z_{k-1} - \frac{1}{6\tau} z_{k-2} - \frac{1}{8\tau} z_{k-3} + \frac{1}{12\tau} z_{k-4}
\] (17)

with truncation error order \( O(\tau^3) \). But it gives slightly better results when combined with the five-instance backward difference formula, see \([2]\) p. 356,

\[
\dot{A}_k = \frac{25A_k - 48A_{k-1} + 36A_{k-2} - 16A_{k-3} + 3A_{k-4}}{12\tau}
\] (18)

of similar error order as our earlier discretization formula \([12]\).

Very few convergent look-ahead finite difference formulas were known until quite recently. The ones of higher truncation error orders than Euler’s method at \( O(\tau^2) \) have been found by lucky and haphazard processes described in \([12]\) section 2.3 or \([14]\) Appendix A for example. Our five- and six-instant look-ahead difference formulas 5-IFD \([13]\) and \([17]\) with truncation error orders \( O(\tau^4) \) serve rather well here with \([17]\) and \([18]\) giving us one more accurate digit than \([13]\) and \([12]\). In the meantime, higher truncation order look-ahead finite difference formulas have been developed \([17]\) with orders up to \( O(\tau^8) \) and we will use some of these for comparisons in the next section with numerical examples.

4 Numerical Implementation, Results and Comparisons

The ZNN algorithm that we detail in this section finds the complete eigendata of time-varying symmetric matrix flows \( A(t) \) \( \in \mathbb{R}^{n,n} \) by using the five-instance forward difference formula \([13]\) and predicts the eigendata for time \( t_{k+1} \) while using the four instance backward formula \([12]\) for approximating \( \dot{A}(t_k) \). Using other difference formula
pairs such as (17) and (18) or higher error order ones instead would require finding additional starting values and adjusting the code lines that define $A_k$ and $z_{k+1}$ accordingly.

To start the 5-IFD iteration (16) requires knowledge of four start-up matrices $A(t_k) \in \mathbb{R}^{n \times n}$ for $k = 1, \ldots, 4$ and their complete eigendata, i.e., knowledge of all $n$ eigenvectors and associated eigenvalues of $A(t)$ at the four time instances $t_1, \ldots, t_4$. In this paper’s ZNN eigen algorithms we always start from $t_1 = 0$ with $A(0)$ and then gather the complete eigendata of $A(0), A(t_2), A(t_3),$ and $A(t_4)$ in an $n + 1 \times 4$ eigendata matrix $Z$ using Francis’ QR algorithm as implemented by MATLAB’s `eig` m-file. To evaluate $z(t_{k+1}) \in \mathbb{R}^{n+1}$ with an eigenvector preceding the respective eigenvalue for any $k \geq 4$ via formula (16), we always rely on the four immediately time-preceding eigendata sets.

We repeat this ZNN process in a double do loop, in an outer loop for increasing time instances $t_k$ from $t_5$ on until the final time $t_f$ is reached, and in an inner, a separate loop to predict each of the $n$ eigenvalues and the associated eigenvectors of $A(t_{k+1})$ from earlier eigendata for the next time instance $t_{k+1}$ by using the look-ahead finite difference formula (13).

This works well for smooth data changes in $A(t)$ and all eigenvalue distributions of the flow. Note that ZNN succeeds even for symmetric matrix flows with repeated eigenvalues, where both, decomposition and ODE path following methods suffer crucial breakdowns, see e.g. [5] and [10]. When the computed approximate derivative $\dot{A}(t_k)$ computed by a discretization formula such as (12) has an unusually large norm above 300, we judge that the input data $A(t_k)$ has undergone a discontinuity, such as from sensor or data transmission failure. In this case we restart the process anew from time $t_k$ on, just as we did at the start from $t_1 = 0$ on and again use MATLAB’s `eig` function for four consecutive instances to produce the necessary start-up eigendata. Thereafter we compute recursively again from $t_{k+1}$ onwards via (16) as done before for $t_1 = 0, t_2, t_3,$ and $t_4$, but now for the jump-modified input data flow $\dot{A}(t)$. This proceeds until another data jump occurs and the original data flow $A(t)$ continues as input till the final time $t_f$. In that way we can find the complete eigendata vectors $z_m(t)$ and $j > k + 3$ for all affected indices $m$ until we reach the final specified time instance $t_f$. Our method thus allows for piecewise smooth data flows $A(t)$ and renders it adaptive to real-world occurrences and implementations.

In the following we show some low dimensional test results for the symmetric time-varying eigenvalue problem with the 7 by 7 seed matrices

$$
A_s(t) = \begin{pmatrix}
\sin(t) + 2 & e^{\sin(t)} & 0 & -e^{\sin(t)} & 1/2 & 1 + \cos(t) & 0 \\
e^{\sin(t)} & \cos(t) - 2 & 0 & 1 & \cos(2t) & e^{\cos(t)} & 0 \\
0 & 0 & -0.12t^2 + 2.4t - 7 & 0 & 0 & 0 & 0 \\
-e^{\sin(t)} & 1 & 0 & 1/(t + 1) & \arctan(t) & \sin(2t) & 0 \\
1/2 & \cos(2t) & 0 & \arctan(t) & 1 & e^{\cos(t)} & 0 \\
1 + \cos(t) & 1 & 0 & \sin(2t) & e^{\cos(t)} & 1/(t + 2) & 0 \\
0 & 0 & 0 & 0 & 0 & -0.15t^2 + 3t - 6 & 0
\end{pmatrix}
$$

and the data-jump perturbed symmetric time-varying matrices

$$
A_{sj}(t) = \begin{pmatrix}
\sin(t) + 2 & e^{\sin(t)} & 0 & -e^{\sin(t)} & 0 & 1 + \cos(t) & 0 \\
e^{\sin(t)} & 0 & 0 & 1 & \cos(2t) & 1 & 0 \\
0 & 0 & 1.3t - 15 & 0 & 0 & 0 & 0 \\
-e^{\sin(t)} & 1 & 0 & 1/(t + 1) & 1 & 2\cos(2t) & 0 \\
0 & \cos(2t) & 0 & 1 & -3 & e^{\cos(t)} & 0 \\
1 + \cos(t) & 1 & 0 & 2\cos(2t) & e^{\cos(t)} & 6/(t + 2) & 0 \\
0 & 0 & 0 & 0 & 0 & 14.05 - t & 0
\end{pmatrix}
$$

whose entries we randomize via the same random orthogonal similarity $U \in \mathbb{R}^{7 \times 7}$ to become our test matrices $A(t) = U^T A_s(t)U$ and $A_{sj}(t) = U^T A_{sj}(t)U$. In our tests, $A_j(t)$ takes over from $A(t)$ between two-user set discontinuous or jump instances. $A_{sj}(t)$’s data differs from that of $A_j(t)$ in positions (1,5), (2,2), (3,3), (4,5), (4,6), (5,1), (5,4), (5,5), (6,4), (6,6), and (7,7). These entries are marked in red in $A_j(t)$.

Our first example uses the sampling gap $\tau = 1/200$ sec and the decay constant $\eta = 4.5$. The computations run from time $t = 0$ sec to $t = t_f = 20$ sec with data input discontinuities at two jump points $t = 8$ sec and $t = 14.5$ sec. The computed eigendata satisfies the eigenvalue equation $A(t)V(t) = V(t)\lambda(t)$ with relative errors in the $10^{-5}$ to $10^{-7}$ range in Figure 2 except for a sharp glitch near the end of the run when two eigenvalues of $A(t)$ take sharp
turns and nearly cross paths. Figure 3 shows the behavior of the norm of the derivative matrix \( \dot{A}(t) \) as computed via (12) over time. This norm varies very little between 2 and 4 unless the input data becomes discontinuous at the chosen jump points where it spikes to around 2,000. Figure 4 examines the orthogonality of the computed eigenvectors over time. They generally deviate from exact orthogonality by between \( 10^{-4} \) and \( 10^{-7} \) except for spikes when two computed eigenvalues make sharp turns, sharper than the chosen sampling gap \( \tau \) allows us to see clearly. The troublesome behavior of ‘coalescing’ eigenvalues in matrix flows was noticed earlier by Mailybeav [13] and studied more recently by Dieci and Eirola [5]. Without graphing, our algorithm’s typical run times for this data set and duration until \( t = t_f = 20 \) sec are around 0.5 seconds and take about 2.5 % of the total simulation time. This leaves the processor idle for around 97 % of our 20 second time interval and indicates a very efficient real-time realization.

![Figure 1](image1.png)

**Figure 1**

![Figure 2](image2.png)

**Figure 2**
Figure 3

Figure 4
To reach higher accuracies with the sampling gap shortened to \( \tau = 1/1,000 \) sec, the results are much improved, see Figures 5 through 7 below. Here we have raised the value of \( \eta \) to 80. When using its previous value of 4.5 with \( \tau = 0.001 \) sec, the plots would look much the same, but the achieved accuracy would suffer a wee bit. On the other hand for \( \tau = 0.005 \) in Figures 1 – 4, increasing \( \eta \) from our chosen value would not have worked well. With \( \tau = 0.001 \) sec and for any almost any \( \eta \gg 1 \), the average eigendata computation time for a run without graphing averages at around 2.5 sec. This allows the processor to be idle for around 87% of the total process time and again shows that our ZNN based discretized eigendata algorithm is well within real-time feasibility.

![Figure 5](image1.png)

**Figure 5**

Note that in Figure 5 the eigenvalue glitch of Figure 1 near \(-2\) and \(t = 19\) has been smoothed out. There are 31 incidences of eigenvalue crossings or repeated eigenvalues in this example that were handled by ZNN without any problems at all.

In Figure 6 below, the eigenvalue equation errors have been lowered by a factor of around \(10^3\) by decreasing the sampling gap by a factor of 5 from Figure 2 and \(5^4 = 625 \approx 1,000\), validating the 5-IFD error order \(O(\tau^4)\) here.

![Figure 6](image2.png)

**Figure 6**
The seven plots above were obtained by using the 5-IFD 1-step ahead difference formula (13) in conjunction with the 4 instance backward difference formula (12). The 6-IFD (17) and the 5 instance backward formula (18) pair have the same error order as the former difference formula pair (13) and (12) but they achieve superior results as the following plots indicate.

This plot shows no glitches near –2 and \( t = 19 \) sec when compared with Figure 1 and it shows a clear 32nd eigenvalue crossing at that former glitch point.
When using the 6-IFD in our Matlab code Zmatrixeig2_3sym.m [16], the computed results generally have lower relative errors of between a half to one digit than with the 5-IFD method and Zmatrixeig2_2sym.m in [16]. Both methods have the same truncation error order 4 and run equally fast.

Finally we investigate another way to try and predict the eigendata of $A_{k+1} = A(t_{k+1})$ from that of earlier eigendata for $A_j = A(t_j)$ with $j \leq k$. If the sampling gap $\tau$ is small such as $\tau = 0.001$ sec, how would the eigendata of $A(t_k)$ fare as a predictor for that of $A(t_{k+1})$? To generate the plot below we have computed the eigenvalues of $A(t)$ 20,000 times using the static Francis matrix eigenvalue algorithm that is built into Matlab in eig.

Clearly this naive method does not reliably predict future eigendata at all since it generates past eigendata whose average relative errors of magnitudes between $10^{-4}$ and $8 \cdot 10^{-4}$ exceed those of our two tested ZNN algorithms significantly. This has been corroborated in general for static methods in [21]. Static method appropriations for time-varying problems suffer from sizable residual errors. The discretized ZNN methods of this paper are reliable predictors for time-varying matrix eigenvalue problems. ZNN can achieve high accuracy. It is predictive and discretizable, as well as capable of handling discontinuous and non-smooth input data flows $A(t)$. 
Outlook

In the course of this research we have come upon a number of interesting observations and intriguing open questions which we post here.

A: Orthogonal Eigenvectors for Time-varying Symmetric Matrix Flows $A(t)$

Our ZNN based methods with low truncation error orders do not seem to compute nearly orthogonal eigenvectors for time-varying real symmetric matrix flows $A(t)$ as depicted in Figures 4 and 7. In theory, the set of eigenvectors of every symmetric matrix $A(t)$ can be chosen as orthonormal and the static Francis QR algorithm computes them as such by its very design of creating a number of orthogonal similarities on $A(t)$ that then converges to a diagonal eigenvalue matrix for the symmetric matrices $A(t)$. But ZNN dynamics have no way of knowing that each $A(t)$ is symmetric and besides they pay no heed to backward stable orthogonal static methods. We have tried to force orthogonality on the eigenvectors of each $A(t)$ by adjoining the following exponential decay differential equation as an additional error function

$$ee(t) = \left( \begin{array}{cccc} -x_1 & - & \cdots & - \\ \vdots & \ddots & \ddots & \vdots \\ -x_{k-1} & - & \cdots & - \end{array} \right) \left( \begin{array}{c} \vdots \\ x_k(t) \end{array} \right) = o_{k-1}$$

to the $P(t_k)\dot{z}(t_k) = q(t_k)$ formulation in equation (11). But this made for worse orthogonalities than occur naturally with our simpler method. However, the new higher truncation error order convergent look-ahead difference schemes of [17] seem to put this problem with discretized ZNN methods to rest. See Figure 13 below where we have made use of the new 5th error order convergent 7-IFD finite difference method, coded as Zmatrixeig3_3bsym.m in [16] to give us

$$z_{k+1} = \frac{-80z_k + 182z_{k-1} + 206z_{k-2} - z_{k-3} - 110z_{k-4} + 40z_{k-5}}{237}.$$ 

Figure 13 shows only moderate deviations from orthogonality of around $10^{-15}$ for the computed 7 by 7 eigenvector matrices except for the deliberately chosen discontinuities at 4.5 and 13. Otherwise there only a couple of relatively small glitches with orthogonality in the $10^{-9}$ to $10^{-12}$ range plus one major one in the $10^{-5}$ range near $t = 19$. 
B : General Time-varying Matrix Flows $A(t) \in \mathbb{R}^{n,n}$ or $\mathbb{C}^{n,n}$ and their Eigendata

How will the Zhang Neural Network method work for general square matrix flows $A(t)$, independent of their entry structure and eigenvalues? Its main task is updating the eigendata vector $z(t_k)$ by a rule such as the 5-IFD based formula below with truncation error order 4, coded as 2matrixeig2.2sym.m in [16]

$$z_{k+1} = \frac{9}{4} r(P \setminus q) - \frac{1}{8} z_k + \frac{3}{4} z_{k-1} + \frac{5}{8} z_{k-2} - \frac{1}{4} z_{k-3}.$$  \hfill (15)

This task is relatively simple: create the matrix $P$ and the right hand side vector $q$ such as indicated in formulas (10) and (16), for example; compute a prescribed linear combination of previous $z$ vectors and solve a linear equation. By all signs, a generalization of our method should be able to work with arbitrary matrix flows, not just symmetric flows $A(t) \in \mathbb{R}^{n,n}$. But we have not checked.

C : Matrix Flows $A(t) \in \mathbb{R}^{n,n}$ for Large Dimensions $n$, both Dense or Sparse, and their Eigendata Computations

How will an iteration rule such as [15] work with large dense or huge sparse matrix flows $A(t)$? Its main work consists of solving the linear system $P \setminus q$ for large dimensions $n$. Here all of our known static linear equation solvers for sparse and dense system matrices, be they Gauss, Cholesky, Conjugate Gradient, GMRES or more specific Krylov type methods should work well. Future tests will tell.

D : Matrix Flow $A(t)$ Eigendata Computations with Singular System Matrices $P$ in Equation (11)

Must ZNN based discrete methods break down if the DE system matrix $P(t)$ becomes singular at some time $t$? If $n$ is relatively small and we fail with Gaussian elimination to find $P \setminus q$, i.e. if $P(t)$ is singular, then we can append one row of $n+1$ zeros to $P_{n+1,n+1}$ and a single zero to $q$, and the backslash operator \ of Matlab will not use elimination methods but solve the augmented non-square linear system

$$\begin{pmatrix} P & 0 & \ldots & 0 \\ 0 & \cdots & 0 & q \\ \end{pmatrix}_{n+2,n+1} \setminus \begin{pmatrix} 0 \\ 0 \\ \end{pmatrix}_{n+2}$$  \hfill (11a)

instead by using the SVD for the augmented equation (11b). This is a little more expensive but delivers the least squares solution. We wonder what could be done for singular $P$ for sparse and huge time-varying matrix eigenvalue problems in the singular $P(t)$ case.

E : Is it more advantageous to perform $n$ separate Iterations, each with one $n+1$ by $n+1$ DE System Matrix $P$ as was done here, or use just one $n^2 + n$ by $n^2 + n$ comprehensive DE System at each timestep $t_k$?

We have tried both approaches and the outcome depends. Further tests are warranted.

Time-varying matrix problems form a new and seemingly different branch of Numerical Analysis. ZNN Neural Network based algorithms rely on a few known stable 1-step ahead discretization formulas for their simplicity, speed and accuracy, and on solving linear equations. ZNN method sensitivities such as the accuracy problems with sharply turning eigenvalue paths as depicted in Figures 1, 2, 4, 5, 6, 7, 9 and 11 near $t = 19$ are new and differ from what we have learned in static matrix numerical analysis. This gives us fertile ground for in-depth numerical explorations of ZNN methods with time-varying matrix problems in the future.

Matlab codes of our programs for time-varying matrix eigenvalue problems are available inside the folder [http://www.auburn.edu/~uhligfd/m_files/T-VMatrixEigenv]. The 5-IFD version is called 2matrixeig2.2sym.m (via 5-IFD), the 6-IFD version is 2matrixeig2.3sym.m (via 6-IFD) and the 7-IFD method is 2matrixeig3.3bsym.m in [16].

Acknowledgements : We thank Françoise Tisseur and Peter Benner and his group for valuable suggestions and additional references.
References

[1] A. Bunse-Gerstner, R. Byers, V. Mehrmann, N. K. Nichols, Numerical computation of an analytic singular value decomposition of a matrix valued function, Numerische Mathematik, 60 (1991), 1 - 39.

[2] G. Engeln-Müllges, F. Uhlig, Numerical Algorithms with C, Springer (1996).

[3] K. Chen, Recurrent implicit dynamics for online matrix inversion, Applied Mathematics and Computation, 219 (20) (2013), 10218 - 10224.

[4] K. Chen, C. Yi, Robustness analysis of a hybrid of recursive neural dynamics for online matrix inversion, Applied Mathematics and Computation 273, (2016), 969 - 975.

[5] Luca Dieci and Timo Eirola, On smooth decompositions of matrices, SIAM J. Matrix Anal. Appl., 20 (1999), p. 800-819 (1999).

[6] D. Guo, Z. Nie, L. Yan, Theoretical analysis, numerical verification and geometrical representation of new three-step DTZD algorithm for time-varying nonlinear equations solving, Neurocomputing, 214 (2016), 516 - 526.

[7] L. Jin, S. Li, B. Hu, RNN models for dynamic matrix inversion: a control-theoretical perspective, IEEE Transactions on Industrial Informatics, 14 (1) (2018), 189 - 199.

[8] S. Li, S. Chen, B. Liu, Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function, Neural Processing Letters, 37 (2) (2013), 189 - 205.

[9] S. Li, Y. Li, Nonlinearly activated neural network for solving time-varying complex Sylvester equations, IEEE Transactions on Cybernetics, 44 (8) (2014), 1397 - 1407.

[10] Sébastien Loisel and Peter Maxwell, Path-following method to determine the field of values of a matrix at high accuracy, SIAM J. Matrix Analysis, Appl., 39 (2018), p. 1726 - 1749. https://doi.org/10.1137/17M1148608

[11] J. Li, M. Mao, F. Uhlig, Y. Zhang, Z-type neural-dynamics for time-varying nonlinear optimization under a linear equality constraint with robot application, Journal of Computational and Applied Mathematics, 327 (2018), 155 - 166.

[12] J. Li, M. Mao, F. Uhlig, Y. Zhang, A 5-instant finite difference formula to find discrete time-varying generalized matrix inverses and scalar reciprocals, 17 p., submitted

[13] A. A. Mailybaev, Computation of multiple eigenvalues and generalized eigenvectors for matrices dependent on parameters, Numerical Linear Algebra with Applications 13 (2006), 419-436.

[14] B. Qiu, Y. Zhang, Z. Yang, New Discrete-Time ZNN Models for Least-Squares Solutions of Dynamic Linear Equation Systems With Time-Varying Rank-Deficient Coefficients, IEEE Trans Neural Networks and Learning Systems, to appear (2019), 12 p.

[15] P. Sirković, D. Kressner, Subspace acceleration for large-scale parameter-dependent Hermitian eigenproblems, SIAM J. Matrix Anal. Appl. 37 (2016), 695-718. [Zbl 1382.65104]

[16] F. Uhlig, Matlab codes for time-varying matrix eigenvalue problems are at [http://www.auburn.edu/~uhligfd/m_files/T-VMatrixEigenV]. This paper’s codes are called Zmatrixeig2_2sym.m when using formulas (13) and (14), Zmatrixeig2_3sym.m, when using formulas (17) and (18) and Zmatrixeig3_3bsym.m.

[17] F. Uhlig, The construction of high order convergent look-ahead finite difference formulas for Zhang neural networks, submitted, 9p.
[18] Xuezhong Wang, Maolin Che, Yimin Wei, Recurrent neural network for computation of generalized eigenvalue problem with real diagonalizable matrix pair and its applications, Neurocomputing, 216 (2016), p. 230-241.

[19] L. Xiao, B. Liao, A convergence-accelerated Zhang neural network and its solution application to Lyapunov equation, Neurocomputing, 193 (2016), 213 - 218.

[20] L. Xiao, B. Liao, S. Li, Z. Zhang, L. Ding, L. Jin, Design and analysis of FTZNN applied to real-time solution of non-stationary Lyapunov equation and tracking control of wheeled mobile manipulator, IEEE Transactions on Industrial Informatics, 14 (1) (2018), 98 - 105.

[21] Y. Zhang, S. S. Ge, Design and analysis of a general recurrent neural network model for time-varying matrix inversion, IEEE Transactions on Neural Networks, 16 (6) (2005), 1477 - 1490.

[22] Y. Zhang, D. Jiang, J. Wang, A recurrent neural network for solving Sylvester equation with time-varying coefficients, IEEE Transactions on Neural Networks, 13 (5) (2002), 1053 - 1063.

[23] Y. Zhang, Z. Ke, P. Xu, C. Yi, Time-varying square roots finding via Zhang dynamics versus gradient dynamics and the former's link and new explanation to Newton-Raphson iteration, Information Processing Letters, 110 (24) (2010), 1103 - 1109.

[24] Y. Zhang, F. Li, Y. Yang, Z. Li, Different Zhang functions leading to different Zhang-dynamics models illustrated via time-varying reciprocal solving, Applied Mathematical Modeling, 36 (9) (2012), 4502 - 4511.

[25] Yunong Zhang and Jun Wang, Recurrent neural networks for nonlinear output regulation, Automatica, vol. 37 (2001), p. 1161 - 1173.

[26] Y. Zhang, M. Yang, J. Li, L. He, S. Wu, ZFD formula 4IgSFD Y applied to future minimization, Physics Letters A, 381 (19) (2017), 1677 - 1681.

[27] Y. Zhang, M. Yang, C. Li, F. Uhlig, H. Hu, New continuous ZD model for computation of time-varying eigenvalues and corresponding eigenvectors, submitted, 17 p.

[28] Y. Zhang, C. Yi, Zhang Neural Networks and Neural-Dynamic Method, Nova Science Publishers Inc., UK, 2011.

[29] Y. Zhang, C. Yi, D. Guo, J. Zheng, Comparison on Zhang neural dynamics and gradient-based neural dynamics for online solution of nonlinear time-varying equation, Neural Computing and Applications, 20 (1) (2011), 1 - 7

\[ \ldots \text{/latex/t-vMatrixEvviaDiffEq.tex} \] April 25, 2019

tveig005.pdf
tvglrer005.pdf
tvdernorm005.pdf
tvorthdev005.pdf
tveig001.pdf
tvglrer001.pdf
tvorthdev001.pdf
tv9eig005.pdf
tv9glrer005.pdf
tvglrererrFrQR.pdf
tveig33symm.pdf