



## A New Neurodynamic Model with Adam Optimization Method for Solving Generalized Eigenvalue Problem

Ebrahim Ganjalipour<sup>1</sup>, Khadijeh Nemati<sup>1</sup>, Amir Hosein Refahi Sheikhi<sup>1\*</sup>, Hashem Saberi Najafi<sup>2</sup>

<sup>1</sup>Department of Mathematics and Computer Sciences, Faculty of Mathematical Sciences, Lahijan Branch, Islamic Azad University, Lahijan, Iran; Ibrahim.ganjali@gmail.com; khadijehnemati64@gmail.com; Ah\_refahi@yahoo.com.

<sup>2</sup>Department of Applied Mathematics, Ayandegan Institute of Higher Education, Tonekabon, Iran; hnajafi@guilan.ac.ir.

### Citation:



Ganjali, E., Nemati, Kh., Refahi Sheikhi, A. H., & Saberi Najafi, H. (2021). A new neurodynamic model with adam optimization method for solving generalized eigenvalue problem. *Big data and computing visions*, 1 (2), 83-95.

Received: 11/02/2021

Reviewed: 22/03/2021

Revised: 19/04/2021

Accept: 03/05/2021


### Abstract

In this paper we proposed a new neurodynamic model with recurrent learning process for solving ill-condition Generalized Eigenvalue Problem (GEP)  $Ax = \lambda Bx$ . our method is based on Recurrent Neural Networks (RNN) with customized energy function for finding smallest (largest) or all eigenpairs. We evaluate our method on collected structural engineering data from Harwell Boeing collection with high dimensional parameter space and ill-conditioned sparse matrices. The experiments demonstrate that our algorithm using Adam optimizer, in comparison with other stochastic optimization methods like gradient descent works well in practice and improves complexity and accuracy of convergence.

**Keywords:** Recurrent Neural Network (RNN), Eigenpairs, Adam optimizer, Positive definite matrix, Ill-condition.

## 1 | Introduction

Generalized Eigenvalue Problem (GEP), is one of the most important problems in numerical linear algebra. The generalized eigenvalue is a handy and versatile mathematical tool that provides information about the correlation of linear transformations [1]. It's importance is because it is often used in a lot of engineering problems, for example in structural engineering, most of machinery vibration problems is GEP of mass and stiffness matrices. In these cases, we come across a symmetric positive definite GEP of the form  $Ax = \lambda Bx$ , where A and B are symmetric, and at least one of them is positive definite. One elementary method for finding eigenvalue and eigenvectors of GEP is using characteristic equation ( $\det(A - \lambda B) = 0$ ) but except special cases it is not desired because we can not obtain the coefficients of characteristic equation by computing determinant or other numerical methods specially for large matrices. There are solutions with acceptable results, like Krylov subspace method that projects matrices to low dimension space for achieving reduced computation cost and better performance [2]. Inner-outer iteration method in [3] presented an algorithm with remarkable fast convergence. The method is based on using inverse vector iteration method and solving the resulting system with the help of Krylov subspace methods.

 Licensee Big Data and Computing Vision. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).



Corresponding Author: [ah\\_refahi@yahoo.com](mailto:ah_refahi@yahoo.com)



<https://doi.org/10.22105/bdcv.2021.142589>

Neural network approach for eigenvalue problem employs a Recurrent Neural Network (RNN) and it exploits the continuous time error backpropagation learning algorithm [4]. In [5], [6] and [7] neurodynamical methods are proposed for solving constrained complex-variable convex and pseudo convex optimization. Neural dynamical method is a possible and promising approach to solve optimization problems with high dimension and complex structure in real time [8]. Mathematical interpretation of the neural network method for the optimization is usually transformed into an Ordinary Differential Equation (ODE) and called the neurodynamic optimization approach [9]. Cichocki and Unbehauen [4] supposed nonlinear dynamic system for solving Ordinary Eigenvalue Problem (OEP) with high computation performance and parallel processing ability. They find all the eigenvalues and the associated eigenvectors simultaneously by training the network to make some target patterns. the optimization process in [4] must be repeated in many times with different initial conditions for finding all eigen pairs. In [4] the performance and convergence behavior of the proposed neural networks architectures are investigated by extensive computation simulations.

Many RNNs for  $Ax = \lambda Bx$  problem have been reported which used neurodynamic optimization approach. [10] proposed the following continuous RNN to solve GEP

$$dx/dt = x^T A x B x - x^T B x A x. \quad (1)$$

Liu et al. [10] first introduced some assumptions, such as  $A, B$  are real symmetric matrices,  $B$  is positive definite, and both the algebraic and geometrical multiplicities of the largest and smallest generalized eigenvalue equal to 1. Feng et al. [11] proved that the proposed neural network in [10] is capable of computing all generalized eigenvalues of GEP. Based on more general assumption, they not only proved that the limit of state solution of the RNN exists, but also proved that the state solution converges to a generalized eigenvector. Moreover, it was shown that the related generalized eigenvector depends on the choice of the initial point. There is a lot of works that focused on computing largest or smallest eigenvalue of symmetric positive definite matrix. Yi et al. [12] studied this problem and developed to compute eigenvectors of any real symmetric matrix. The proposed model of neural network described by differential equations. It is a model of RNNs that have asynchronous parallel processing ability and can achieve high computing performance. The dynamics of the proposed neural network model is described by

$$\dot{x}(t) = -x(t) + f(x(t)), \quad (2)$$

$$f(x) = [x^T A x + (1 - x^T A x)I]x \quad \text{where } A \text{ is symmetric.}$$

$x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$  represents the state of the network,  $I_n$  is the  $n \times n$  identity matrix. Clearly, this is a class of RNNs. Dynamic behavior of *Eq. (1)* plays a crucial and important part to its applications. equilibrium point for the model satisfies that  $-x + f(x) = 0$  i.e.

$$x^T A x - x^T A x x = 0. \quad (3)$$

Since  $A$  is a symmetric matrix, then there exists an orthonormal basis of  $\mathbb{R}^n$  composed by eigenvectors of  $\mathbb{R}^n$ . Supposing  $\lambda_i$  ( $i = 1, \dots, n$ ) as eigenvalues of  $A$  and  $V_i$  ( $i = 1, \dots, n$ ) as the corresponding eigenvectors that compose an orthonormal basis of  $A$ . Then, for any  $x \in \mathbb{R}^n$  it can be represented as

$$x = \sum_{i=1}^n z_i V_i. \quad (4)$$

$$x(t) = \sum_{i=1}^n \sqrt{\frac{x(0)^T x(0)}{\sum_{j=1}^n z_j^2(0)}} e^{2x(0)^T x(0)(\lambda_j - \lambda_i)t} z_i(0) V_i. \quad (5)$$

where  $z_i$  ( $i = 1, \dots, n$ ) are some constants. Yi et al. [12] proved *Eq. (1)* dynamic model converges to eigenvector and eigenvalue of matrix.

Most of introduced methods in [3], [4], [10], [11], [12], [13] and [14] didn't provide performance verifications for solving GEP arise in real engineering applications and didn't check results in sparse large and ill condition data. In This paper we use ill condition matrices for verifying results. We propose nonlinear dynamic system using RNN and Schur decomposition for solving GEP. We use Adam method for optimization of the RNN energy function and results are investigated for structural engineering data of Harwell Boeing Collection [15].

The rest of this paper organized as follows. In Section 2 we explain RNN of [4] for finding eigenpairs of OEP. In Section 3 we present our method for solving GEP using Adam optimizer. In Section 4 we explain how to change neurodynamic model for computing largest (smallest) eigenpair. In Section 5 we indicate results of executing RNN on matrices of structural engineering and Section 6 summarizes some conclusions.

## 2 | RNN for Finding Eigenvalues and Eigenvectors of $Ax = \lambda x$

In this section we present neurodynamic model in [4] which we use it in our algorithm. Assume that  $\lambda_i$  is eigenvalue and  $V_i = [v_{1i}, v_{2i}, \dots, v_{ni}]^T \neq 0$  is associated eigenvector (referred to as an eigenpair) of a real matrix  $A \in \mathbb{R}^{n \times n}$ . We can find the solution of the set of nonlinear algebraic  $(A - \lambda_i I)V_i = 0$  ( $i = 1, 2, \dots, n$ ) equations by using the penalty method that formulate the cost function as follow:

$$E(V_i, \lambda_i) = \frac{1}{2} \sum_{k=1}^n e_k^2 + \frac{k}{2} \left( \sum_{j=1}^n v_{ji}^2 - 1 \right)^2, \quad (6)$$

$$e_k = \sum_{j=1}^n a_{kj} v_{ji} - \lambda_i v_{ki}. \quad (7)$$

Where  $K > 0$  is the penalty parameter. Cichocki and Unbehauen [4] used gradient descent algorithm for minimization of the cost function in *Eq. (6)* with respect to the variables  $\lambda_i$  and  $v_{ji}$  ( $j = 1, 2, \dots, n$ ).

$$\frac{d\lambda_i}{dt} = -\mu \frac{\partial E}{\partial \lambda_i},$$

And

$$\frac{dv_{ji}}{dt} = -\mu \frac{\partial E}{\partial v_{ji}}.$$

So, we have

$$\frac{d\lambda_i}{dt} = \mu \sum_{k=1}^n e_k v_{ki}, \quad (8)$$

$$\frac{dv_{ji}}{dt} = -\mu \left[ \sum_{k=1}^n e_k a_{kj} - \lambda_i e_j + k v_{ji} \left( \sum_{j=1}^n v_{ji}^2 - 1 \right) \right]. \quad (9)$$

## 3 | Computing All Eigenpairs of (A, B) by RNN and Adam Optimization

In this Section we propose a new method for solving GEP problem, in this sense We change the architecture of RNN and replace gradient descent method with Adam (adaptive moment estimation) optimizer in [16] which works well with sparse gradients and non-stationary settings. In the Adams method, magnitudes of parameter updates are invariant to rescaling of gradient, its step sizes are approximately bounded by the step size hyper parameter, it does not require a stationary objective, it works with sparse gradients and it naturally performs a form of step size annealing [16].

Our method find eigenpairs without computing inverse of B in such a way that works with a new RNN based on combining Cholesky decomposition of B and Schur form of B. In the following we explain details of method.

We can define the GEP problem for symmetric matrices  $A \in \mathbb{R}^{n \times n}$  and  $B \in \mathbb{R}^{n \times n}$ .

$$Ax_i = \lambda_i Bx_i \quad \forall i \in \{1, 2, \dots, n\}.$$

Matrix form of this equation is

$$AX = \Delta BX.$$

Columns of  $X = \{x_1, x_2, \dots, x_n\}$ ,  $X \in \mathbb{R}^{n \times n}$  are eigenvectors and diagonal elements of  $\Delta = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ ,  $\Delta \in \mathbb{R}^{n \times n}$  are eigenvalues such that  $\lambda_i \in \mathbb{R}$ ,  $x_i \in \mathbb{R}^n$ . It is clear that eigenvalue problem is special case of GEP with  $B = I$ .

Now we recall a theorem that we use it later in our proposed method for finding eigenpairs of (A, B).

**Theorem 3.1.** The symmetric GEP problem has real eigenvalues and corresponding linear independent eigenvectors [2].

Since B is symmetric positive definite, it admits Cholesky decomposition:

$$B = MM^T.$$

So, from

$$AX = \Delta BX.$$

We have

$$AX = \Delta MM^T X.$$

Which gives

$$M^{-1} A (M^T)^{-1} M^T X = \Delta M^T X.$$

So, we have

$$CV = \Delta V, \text{ where } V = M^T X. \quad (10)$$

The matrix  $C = B^{-1}A = M^{-1} A (M^T)^{-1}$  is symmetric therefore  $\lambda$  is real. The assertion about the eigenvectors is obvious, because a symmetric matrix has a set of n independent eigenvectors [2]. We should note that the computation of  $M^{-1}$  is not easily possible, we use following method for finding  $M^{-1}$  and  $(M^T)^{-1}$ .

We compute the real Schur form of B and order the eigenvalues of B from smallest to largest

$$U^T B U = D = \text{diag}(d_1, \dots, d_n). \quad (11)$$

Then Form

$$M = U D^{\frac{1}{2}} = U \text{diag}(\sqrt{d_1}, \dots, \sqrt{d_n}). \quad (12)$$

Since  $U$  is orthogonal that means  $U^{-1} = U^T$  so we have

$$M^{-1} = \left(UD^{\frac{1}{2}}\right)^{-1} = \left(D^{\frac{1}{2}}\right)^{-1} U^{-1} = \left(D^{\frac{1}{2}}\right)^{-1} U^T. \quad (13)$$

$$(M^T)^{-1} = \left(\left(D^{\frac{1}{2}}\right)^{-1} U^T\right)^T = U \left(D^{\frac{1}{2}}\right)^{-1}. \quad (14)$$

According to *Eq. (10)* deduction for  $\forall i \in \{1, 2, \dots, n\}$  it is clear that  $\lambda_i$  as eigenvalues of  $Cv_i = \lambda_i v_i$  are equal to the eigenvalues of  $(Ax_i = \lambda_i Bx_i)$  and eigenvectors of pair  $(A, B)$  can achieve by following relation:

$$x_i = \left(D^{\frac{1}{2}}U^T\right)^{-1} v_i = UD^{\frac{1}{2}} v_i.$$

Now we define a RNN with Adam optimization method for solving  $Cy = \Lambda y$  which finds all the eigenvalues and associated eigenvectors on-line. For  $C = \left(D^{\frac{1}{2}}\right)^{-1} U^T A \left(\left(D^{\frac{1}{2}}\right)^{-1} U^T\right)^T$ , suppose matrices  $V$  and  $\Delta$  which fulfil the eigenvalue decomposition  $C = V\Delta V^{-1}$  where  $\Delta = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  is the diagonal matrix so that the elements are the eigenvalues of  $C$  and the columns  $v_i$  ( $i = 1, 2, \dots, n$ ) of the matrix  $V = (v_1, v_2, \dots, v_n)$  are the corresponding eigenvectors. the matrix  $V$  is orthogonal, i.e.,  $V^{-1} = V^T$  or  $V^T V = V V^T = I$  [9]. In order to compute the desired matrices  $V$  and  $\Delta$  we extend presented neurodynamic model in [1] which is RNN with two branches. One branch realizes the eigenvalue decomposition  $C = V\Delta V^{-1}$  and the other branch fulfils the requirement  $V^{-1} = V^T$  or  $V^T V = V V^T = I$ . The network is driven by the excitation vector  $\hat{x} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)^T$  which contains a set of stochastic independent input signals (e.g., orthogonal signals or mutually uncorrelated random (noise) sources). The error vectors of energy function are:

$$\bar{\tau} = \bar{f} - p. \quad (15)$$

$$\hat{\tau} = \hat{f} - s. \quad (16)$$

Where

$$C = \left(D^{\frac{1}{2}}\right)^{-1} U^T A \left(\left(D^{\frac{1}{2}}\right)^{-1} U^T\right)^T. \quad (17)$$

$$\bar{f} = C\hat{x}, p = Vr = V\Delta g = V\Delta V^T \hat{x},$$

$$\hat{f} = \hat{x}, s = Vg = VV^T \hat{x}.$$

The energy vector function is

$$E = [E_1, E_2, \dots, E_n],$$

$$\Delta = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n),$$

$$V = (v_1, v_2, \dots, v_n).$$

$$E_i(v_i, \lambda_i) = \frac{1}{2} \left( \sum_{j=1}^n \bar{\tau}_j^2 + k \sum_{j=1}^n \hat{\tau}_j^2 \right) \quad (i = 1, 2, \dots, n). \quad (18)$$

Where

$$\bar{\tau}_i = \bar{f}_i - p_i = \sum_{j=1}^n c_{ij} \hat{x}_j - \sum_{j=1}^n v_{ij} r_j. \quad (19)$$

$$\hat{\tau}_i = \hat{f}_i - s_i = \hat{x}_i - \sum_{j=1}^n v_{ij} g_j. \quad (20)$$

$$g_j = \sum_{i=1}^n v_{ij} \hat{x}_i. \quad (21)$$

For proposed energy function in Eq. (17) following differential Eqs. (21) and (22) are gradient of energy function w.r.t eigenvalues and eigenvectors.

$$\frac{\partial E_i}{\partial \lambda_i} = (\sum_{k=1}^n \bar{\tau}_k v_{ki}) g_i. \quad (22)$$

$$\frac{\partial E_i}{\partial v_{ij}} = \left[ \sum_{k=1}^n \bar{\tau}_k r_j - k \hat{\tau}_j g_j + k \left( \sum_{k=1}^n \hat{\tau}_k v_{kj} \right) \hat{x}_i \right] \quad (j = 1, 2, \dots, n). \quad (23)$$

Now we are ready for the main algorithm of this paper, which is combination of constructing C matrix by Eq. (17) and Adam optimizer based on Eqs. (21) and (22) as gradients of energy function w.r.t to eigenvalues and eigenvectors, this resulting algorithm is:

---

**Algorithm 3.1:** Finding all eigenpairs of  $Ax = \lambda Bx$  based on Schur form of B and Adam Optimization method. default settings are  $\alpha = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . All operations on vectors are element-wise. With  $\beta_1^t$  and  $\beta_2^t$  we denote  $\beta_1$  and  $\beta_2$  to the power t.

---

Require: A, B matrices of GEP

Require: U, D from Schur decomposition of B ( $U^T B U = D$ )

Require:  $\alpha$  (Step size)

Require:  $\beta_1, \beta_2 \in [0,1)$  (Exponential decay rates for the moment estimates)

Require:  $E_i(v_i, \lambda_i)$  energy function of Eq. (17) (with parameters  $v_i, \lambda_i$  ( $i = 1, \dots, n$ ))

Require:  $v_0 = (v_{01}, v_{02}, \dots, v_{0n})$  as eigenvector and  $\lambda_0$  as eigenvalue (Initial parameters)

---

$$C \leftarrow \left( D^{\frac{1}{2}} \right)^{-1} U^T A \left( \left( D^{\frac{1}{2}} \right)^{-1} U^T \right)^T$$

$M_0 \leftarrow 0$  (Initialize 1st moment vector)

$S_0 \leftarrow 0$  (Initialize 2nd moment vector)

$t \leftarrow 0$  (Initialize timestep)

In parallel for i'th element of energy vector function  $E_i(v_i, \lambda_i)$  do

While  $v_i, \lambda_i$  not converged do

$t \leftarrow t + 1$

$d_{\lambda_i} \leftarrow \frac{\partial E_i}{\partial \lambda_i}$  (Get gradients w.r.t. stochastic objective at timestep t)

$M_{\lambda_i} \leftarrow \beta_1 \cdot M_{\lambda_i} + (1 - \beta_1) \cdot d_{\lambda_i}$  (Update biased first moment estimate)

$S_{\lambda_i} \leftarrow \beta_2 \cdot S_{\lambda_i} + (1 - \beta_2) \cdot d_{\lambda_i}^2$  (Update biased second raw moment estimate)

$\hat{M}_{\lambda_i} \leftarrow M_{\lambda_i} / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)

$\hat{S}_{\lambda_i} \leftarrow S_{\lambda_i} / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)

$\lambda_i \leftarrow \lambda_i - \alpha \cdot \hat{M}_{\lambda_i} / (\sqrt{\hat{S}_{\lambda_i}} + \epsilon)$  (Update parameters)

In parallel for j'th element of eigenvector  $v_i$  do

$d_{v_{ij}} \leftarrow \frac{\partial E_i}{\partial v_{ij}}$  (Get gradients w.r.t. stochastic objective at timestep t)

$M_{v_{ij}} \leftarrow \beta_1 \cdot M_{v_{ij}} + (1 - \beta_1) \cdot d_{v_{ij}}$  (Update biased first moment estimate)

$S_{v_{ij}} \leftarrow \beta_2 \cdot S_{v_{ij}} + (1 - \beta_2) \cdot d_{v_{ij}}^2$  (Update biased second raw moment estimate)

$\hat{M}_{v_{ij}} \leftarrow M_{v_{ij}} / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)

$\hat{S}_{v_{ij}} \leftarrow S_{v_{ij}} / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)

$v_{ij} \leftarrow v_{ij} - \alpha \cdot \hat{M}_{v_{ij}} / (\sqrt{\hat{S}_{v_{ij}}} + \epsilon)$  (Update parameters)

end

end

end

$\Delta \leftarrow \{\lambda_i\}_{i=1}^n$  (eigenvalues of  $Ax = \lambda Bx$ )

$X \leftarrow \left\{ x_i = \left( D^{\frac{1}{2}} U^T \right)^{-1} v_i \right\}_{i=1}^n$  (eigenvectors of  $Ax = \lambda Bx$  s.t  $\{v_i\}_{i=1}^n$  are eigenvectors of  $Cv = \lambda v$ )

return ( $\Delta, X$ )

---

The algorithm updates exponential moving averages of the gradient ( $M_{\lambda_i}, M_{v_{ij}}$ ) and squared gradient ( $S_{\lambda_i}, S_{v_{ij}}$ ) where hyper-parameters  $\beta_1, \beta_2 \in [0,1)$  control the exponential decay rates of these moving average.

## 4 | Computing the Largest (Smallest) Eigenpair for (A, B)

In some applications like signal processing, it is satisfying that we find largest or smallest eigenpair of GEP. We use generalized Rayleigh quotient in our neurodynamic model and it changes the form of optimization problem [17]. Following formula is defined for Rayleigh quotient of (A, B) pair

$$\lambda = \frac{v^T A v}{v^T B v}. \quad (24)$$

Since  $\frac{v^T A v}{v^T B v} B v = \lambda B v$ , Eq. (26) shows that  $\lambda_n \leq \frac{v^T A v}{v^T B v} \leq \lambda_1$ . Values of  $\lambda_n$  and  $\lambda_1$  are smallest and largest eigenvalues of (A, B) pair. According to above definition, finding the smallest and largest eigenvalue for  $Ax = \lambda Bx$  can done by minimizing function in Eq. (30).

$$Rq(v) = \pm \frac{\langle Av, v \rangle}{\langle Bv, v \rangle} = \pm \frac{v^T A v}{v^T B v}. \quad (25)$$

So,

we can suppose following optimization model. From previous section we have that

$$\begin{aligned} C &= \left(D^{\frac{1}{2}}\right)^{-1} U^T A \left(\left(D^{\frac{1}{2}}\right)^{-1} U^T\right)^T. \\ \text{minimize } \Gamma(v) &= \pm \frac{1}{2} v^t C v, \\ \text{st. } (C - \lambda I)v &= 0, \\ v^t v - 1 &= 0. \end{aligned} \quad (26)$$

Where  $v = (v_1, v_2, \dots, v_n)$ ;

Now we reconstruct recurrent network of Section 3:

$$E(v, \lambda) = \frac{1}{2} \left[ \pm \theta v^t C v + \sum_{k=1}^n e_k^2 + \frac{w}{2} \left( \sum_{k=1}^n v_k^2 - 1 \right)^2 \right]. \quad (27)$$

$$e_k = \sum_{j=1}^n c_{kj} v_j - \lambda v_k. \quad (28)$$

Where  $w > 0$  and  $\theta > 0$  are penalty parameters and we achieve following differential equation for RNN:

$$\frac{\partial E}{\partial \lambda} = \sum_{k=1}^n e_k v_k. \quad (29)$$

$$\frac{\partial E}{\partial v_j} = - \left[ \pm \theta \sum_{k=1}^n c_{jk} v_k + \sum_{k=1}^n e_k c_{kj} - \lambda e_j + w v_j \left( \sum_{k=1}^n v_k^2 - 1 \right) \right] \text{ with } v_j(0) \neq 0. \quad (30)$$

Negative sign of  $\theta$  is for largest eigenvalue. For preventing large values for  $w$  and obtaining high accuracy in final result,  $\theta$  parameter is decreased gradually. Smallest (largest) eigenvalue of  $C$  is equal to Smallest (largest) eigenvalue of (A, B) and corresponding eigenvector of (A, B) can achieve by multiplication of  $\left(D^{\frac{1}{2}} U^T\right)^{-1}$  to converged eigenvector. We changed algorithm 4.1 and resulting algorithm for finding smallest (largest) eigenpairs of  $Ax = \lambda Bx$  is :

---

**Algorithm 4.1:** Finding smallest (largest) eigenpairs of  $Ax = \lambda Bx$  based on Schur form of B and Adam Optimization method. default settings are  $\alpha = 0.001$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . All operations on vectors are element-wise. With  $\beta_1^t$  and  $\beta_2^t$  we denote  $\beta_1$  and  $\beta_2$  to the power  $t$ .

---



Require: A, B matrices of GEP

Require: U, D from Schur decomposition of B ( $U^T B U = D$ )

Require:  $\alpha$  (Step size)

Require:  $\beta_1, \beta_2 \in [0,1)$  (Exponential decay rates for the moment estimates)

Require:  $E_i(v_i, \lambda_i)$  energy function of Eq. (17) (with parameters  $v_i, \lambda_i$  ( $i = 1, \dots, n$ ))

Require:  $v_0 = (v_{01}, v_{02}, \dots, v_{0n})$  as eigenvector and  $\lambda_0$  as eigenvalue (Initial parameters)

$$C \leftarrow \left( D^{\frac{1}{2}} \right)^{-1} U^T A \left( \left( D^{\frac{1}{2}} \right)^{-1} U^T \right)^T$$

$M_0 \leftarrow 0$  (Initialize 1st moment vector)

$S_0 \leftarrow 0$  (Initialize 2nd moment vector)

$t \leftarrow 0$  (Initialize timestep)

While  $v_i, \lambda_i$  not converged do

$t \leftarrow t + 1$

$d_\lambda \leftarrow \frac{\partial E_i}{\partial \lambda}$  (Get gradients w.r.t. stochastic objective at timestep t)

$M_\lambda \leftarrow \beta_1 \cdot M_\lambda + (1 - \beta_1) \cdot d_\lambda$  (Update biased first moment estimate)

$S_\lambda \leftarrow \beta_2 \cdot S_\lambda + (1 - \beta_2) \cdot d_\lambda^2$  (Update biased second raw moment estimate)

$\widehat{M}_\lambda \leftarrow M_\lambda / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)

$\widehat{S}_\lambda \leftarrow S_\lambda / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)

$\lambda \leftarrow \lambda - \alpha \cdot \widehat{M}_\lambda / (\sqrt{\widehat{S}_\lambda} + \epsilon)$  (Update parameters)

In parallel for j'th element of eigenvector  $v_i$  do

$d_{v_j} \leftarrow \frac{\partial E_i}{\partial v_j}$  (Get gradients w.r.t. stochastic objective at timestep t)

$M_{v_j} \leftarrow \beta_1 \cdot M_{v_j} + (1 - \beta_1) \cdot d_{v_j}$  (Update biased first moment estimate)

$S_{v_j} \leftarrow \beta_2 \cdot S_{v_j} + (1 - \beta_2) \cdot d_{v_j}^2$  (Update biased second raw moment estimate)

$\widehat{M}_{v_j} \leftarrow M_{v_j} / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)

$\widehat{S}_{v_j} \leftarrow S_{v_j} / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)

$v_j \leftarrow v_j - \alpha \cdot \widehat{M}_{v_j} / (\sqrt{\widehat{S}_{v_j}} + \epsilon)$  (Update parameters)

end

end

$\lambda_{\max} \leftarrow \{\lambda_i\}_{i=1}^n$  (eigenvalues of  $Ax = \lambda Bx$ )

$x \leftarrow \left( D^{\frac{1}{2}} U^T \right)^{-1} v$  (eigenvectors of  $Ax = \lambda Bx$  s.t.  $v$  is smallest (largest) eigenvectors of  $Cv = \lambda v$ )

return  $(\lambda_{\max}, x)$

## 5 | Computer Simulation Results

For investigating performance of presented algorithms we use different collected data from Harwell Boeing collection. We evaluate our method for GEP of structural engineering problems like lumped mass, transmission tower, TV studio and buckling of a hot washer. In the following reports, we indicate results of executing RNN on GEPs with high condition number matrices. The ability of changing hyper parameters of RNN makes better performance and accuracy in convergence to eigenpairs of various GEPs. Exemplary computer simulation results are shown in following.

Table 1 to 4 give matrix statistic items of selected ill-condition GEPs. Fig. 1 to 8 show the convergence of algorithm for each of the problems. We apply the algorithm with random initial condition and adjusted hyper parameters ( $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ ) to find corresponding eigenpairs. Number of required iterations to achieve a desired accuracy, i.e.  $10^{-7}$ , is shown in Table 5. In order to our results, as we can see in Table 5, presented algorithm is able to find a more accurate solution in a much shorter computing (optimization) time in comparison with Stochastic Gradient Descent (SGD).



Table 1. Statistics of lumped mass problem matrices in  $Ax=\lambda Bx$ .

Eigen Pair Matrices	Matrix A: BCSSTK06 from Harwell Boeing collection	Matrix B: BCSSTM06 from Harwell Boeing collection																				
Size	420 x 420, 4140 entries, real symmetric positive definite	420 x 420, 420 entries, real symmetric positive definite																				
Non-Zeros	<table><tr><th>total</th><th>diagonal</th><th>below diagonal</th><th>above diagonal</th><th>A-A'</th></tr><tr><td>7860</td><td>420</td><td>3720</td><td>3720</td><td>0</td></tr></table>	total	diagonal	below diagonal	above diagonal	A-A'	7860	420	3720	3720	0	<table><tr><th>total</th><th>diagonal</th><th>below diagonal</th><th>above diagonal</th><th>A-A'</th></tr><tr><td>420</td><td>420</td><td>0</td><td>0</td><td>0</td></tr></table>	total	diagonal	below diagonal	above diagonal	A-A'	420	420	0	0	0
total	diagonal	below diagonal	above diagonal	A-A'																		
7860	420	3720	3720	0																		
total	diagonal	below diagonal	above diagonal	A-A'																		
420	420	0	0	0																		
Conditioning	<table><tr><th>Frobenius norm</th><td>2.1e+10</td><th>condition number (est.)</th><td>1.2e+07</td></tr><tr><th>2-norm (est.)</th><td>3.5e+09</td><th>diagonal dominance</th><td>no</td></tr></table>	Frobenius norm	2.1e+10	condition number (est.)	1.2e+07	2-norm (est.)	3.5e+09	diagonal dominance	no	<table><tr><th>Frobenius norm</th><td>6.7e+04</td><th>condition number (est.)</th><td>3.5e+06</td></tr><tr><th>2-norm (est.)</th><td>7.6e+03</td><th>diagonal dominance</th><td>yes</td></tr></table>	Frobenius norm	6.7e+04	condition number (est.)	3.5e+06	2-norm (est.)	7.6e+03	diagonal dominance	yes				
Frobenius norm	2.1e+10	condition number (est.)	1.2e+07																			
2-norm (est.)	3.5e+09	diagonal dominance	no																			
Frobenius norm	6.7e+04	condition number (est.)	3.5e+06																			
2-norm (est.)	7.6e+03	diagonal dominance	yes																			

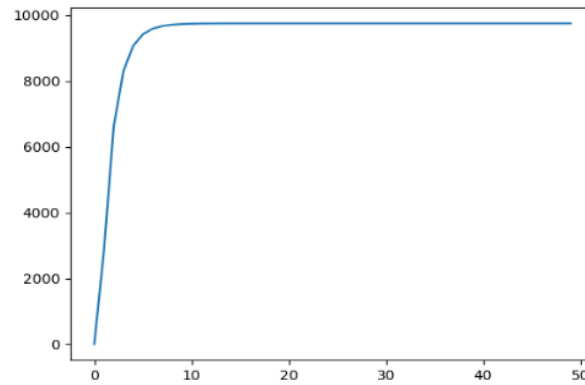


Fig. 1. Eigenvalue convergence of lumped mass problem.

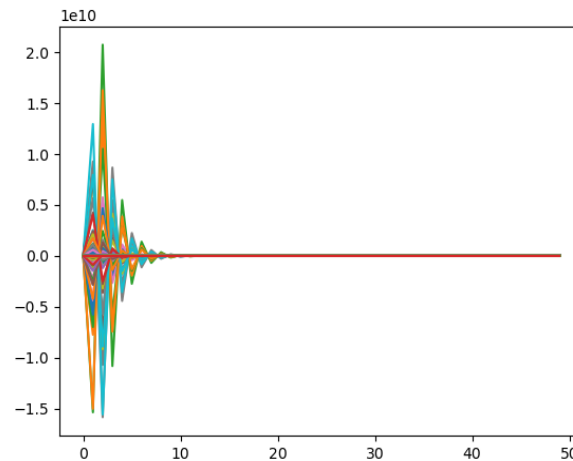


Fig. 2. Eigenvector convergence of lumped mass problem.

Table 2. Statistics of transmission tower problem matrices in  $Ax=\lambda Bx$ .

Eigen Pair Matrices	Matrix A: BCSSTK05 from Harwell Boeing collection	Matrix B: BCSSTM05 from Harwell Boeing collection																				
Size	153 x 153, 1288 entries, real symmetric positive definite	153 x 153, 153 entries, real symmetric positive semi-definite																				
Non-Zeros	<table><tr><th>total</th><th>diagonal</th><th>below diagonal</th><th>above diagonal</th><th>A-A'</th></tr><tr><td>2423</td><td>153</td><td>1135</td><td>1135</td><td>0</td></tr></table>	total	diagonal	below diagonal	above diagonal	A-A'	2423	153	1135	1135	0	<table><tr><th>total</th><th>diagonal</th><th>below diagonal</th><th>above diagonal</th><th>A-A'</th></tr><tr><td>153</td><td>153</td><td>0</td><td>0</td><td>0</td></tr></table>	total	diagonal	below diagonal	above diagonal	A-A'	153	153	0	0	0
total	diagonal	below diagonal	above diagonal	A-A'																		
2423	153	1135	1135	0																		
total	diagonal	below diagonal	above diagonal	A-A'																		
153	153	0	0	0																		
Conditioning	<table><tr><th>Frobenius norm</th><td>2.2e+07</td><th>condition number (est.)</th><td>3.5e+04</td></tr><tr><th>2-norm (est.)</th><td>6.2e+06</td><th>diagonal dominance</th><td>no</td></tr></table>	Frobenius norm	2.2e+07	condition number (est.)	3.5e+04	2-norm (est.)	6.2e+06	diagonal dominance	no	<table><tr><th>Frobenius norm</th><td>4.7</td><th>condition number (est.)</th><td>13</td></tr><tr><th>2-norm (est.)</th><td>0.93</td><th>diagonal dominance</th><td>yes</td></tr></table>	Frobenius norm	4.7	condition number (est.)	13	2-norm (est.)	0.93	diagonal dominance	yes				
Frobenius norm	2.2e+07	condition number (est.)	3.5e+04																			
2-norm (est.)	6.2e+06	diagonal dominance	no																			
Frobenius norm	4.7	condition number (est.)	13																			
2-norm (est.)	0.93	diagonal dominance	yes																			

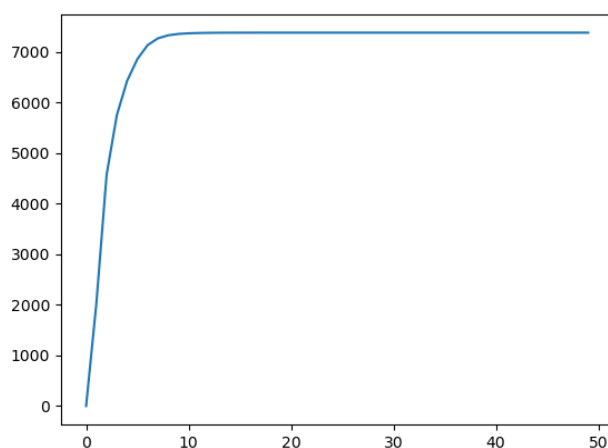


Fig. 3. Eigenvalue convergence of transmission tower problem.

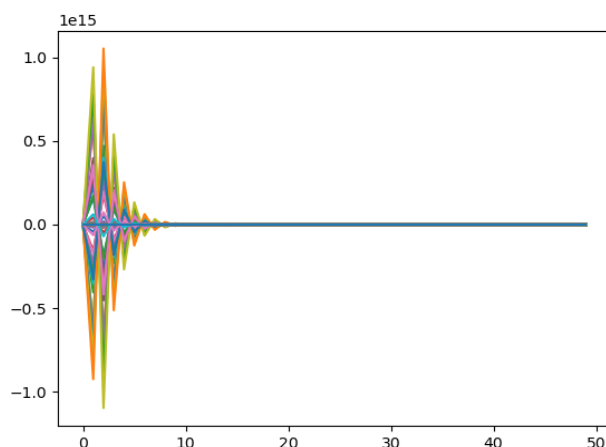


Fig. 4. Eigenvector convergence of transmission tower problem.

Table 1. Statistics of TV studio problem matrices in  $Ax=\lambda Bx$ .

Eigenpair Matrices	Matrix A: BCSSTK08 from Harwell Boeing collection					Matrix B: BCSSTM08 from Harwell Boeing collection				
Size	1074 x 1074, 7017 entries, real symmetric positive definite					1074 x 1074, 1074 entries, real symmetric positive definite				
Non-Zeros	total	diagonal	below diagonal	above diagonal	A-A'	total	diagonal	below diagonal	above diagonal	A-A'
	12960	1074	5943	5943	0	1074	1074	0	0	0
Conditioning	Frobenius norm	1e+11	condition number (est.)	4.7e+07		Frobenius norm	2.3e+06	condition number (est.)	8.3e+06	
	2-norm (est.)	7.7e+10	diagonal dominance	no		2-norm (est.)	1.4e+06	diagonal dominance	yes	

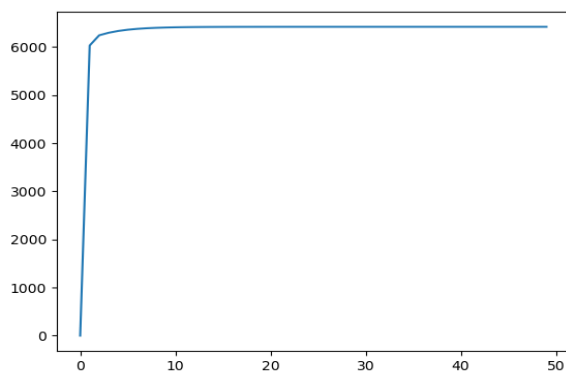


Fig. 5. Eigenvalue convergence of TV studio problem.

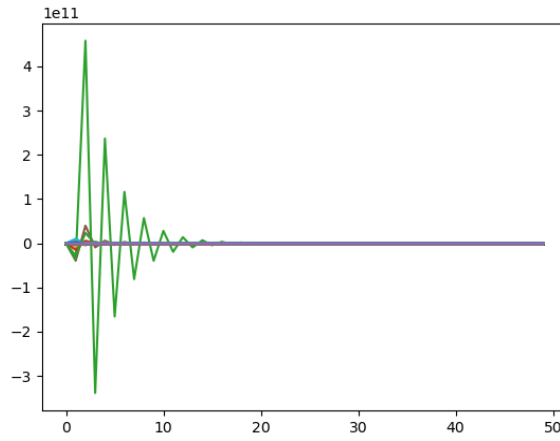


Fig. 6. Eigenvector convergence of TV studio problem.

Table 4. Statistics of buckling of a hot washer problem matrix in  $Ax=\lambda Bx$ .

Eigenpair Matrices	Matrix A: BCSSTK10 from Harwell Boeing collection					Matrix B: BCSSTM10 from Harwell Boeing collection				
Size	1086 x 1086, 11578 entries, real symmetric positive definite					1086 x 1086, 11589 entries, real symmetric positive semi-definite				
Non-Zeros	total	diagonal	below diagonal	above diagonal	A-A'	total	diagonal	below diagonal	above diagonal	A-A'
	22070	1086	10492	10492	0	22070	1086	10492	10492	0
Conditioning	Frobenius norm	3e+08	condition number (est.)	1.3e+06		Frobenius norm	3e+08	condition number (est.)	1.3e+06	
	2-norm (est.)	4.5e+07	diagonal dominance	no		2-norm (est.)	4.5e+07	diagonal dominance	no	

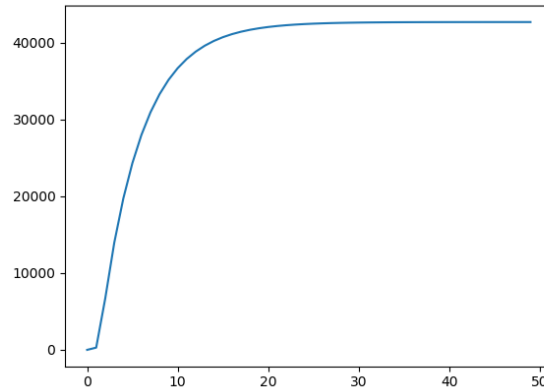


Fig. 7. Eigenvalue convergence for buckling of a hot washer problem.

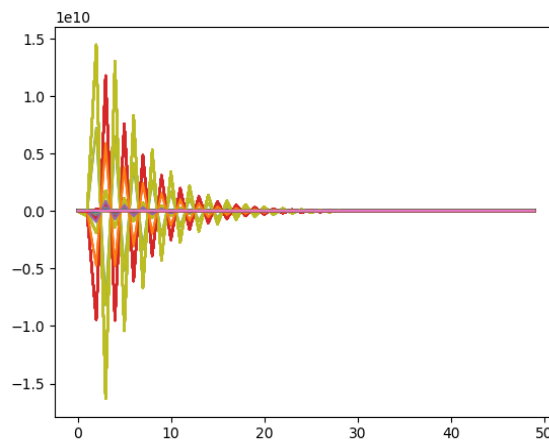


Fig. 8. Eigenvector convergence for buckling of a hot washer problem.

Table 5. Summary of results with adjusted optimization hyper parameters.

Problem	Optimizer	Hyper parameters	Error and Iteration
	Adam	$\alpha = 0.001$	$e = 10^{-7}$
		$\beta_1 = 0.9$	
	Gradian Descent	$\beta_2 = 0.999$	iteration = 21
		$\epsilon = 10^{-8}$	
	Gradian Descent	$M=0.001$	$e = 10^{-7}$
		$K = 40$	iteration = 38
	Adam	$\alpha = 0.001$	$e = 10^{-7}$
		$\beta_1 = 0.9$	
	Gradian Descent	$\beta_2 = 0.999$	iteration = 18
		$\epsilon = 10^{-8}$	
	Gradian Descent	$M=0.001$	$e = 10^{-7}$
		$K = 40$	iteration = 29
	Adam	$\alpha = 0.001$	$e = 10^{-7}$
		$\beta_1 = 0.9$	
	Gradian Descent	$\beta_2 = 0.999$	iteration = 27
		$\epsilon = 10^{-8}$	
	Gradian Descent	$M=0.001$	$e = 10^{-7}$
		$K = 40$	iteration = 50
	Adam,	$\alpha = 0.001$	$e = 10^{-7}$
		$\beta_1 = 0.9$	
	Gradian Descent	$\beta_2 = 0.999$	iteration = 44
		$\epsilon = 10^{-8}$	
	Gradian Descent	$M=0.001$	$e = 10^{-7}$
		$K = 40$	iteration = 63

## 6 | Conclusions

In this paper we used new neurodynamic model for solving GEP. We used Schur QR decomposition algorithm merged with presented neurodynamic method. Resulting new RNN computes all eigenpairs or largest(smallest) eigenvalue and corresponding eigenvector. We evaluate our method on collected structural engineering data from Harwell Boeing collection with high dimensional parameter space and ill condition sparse matrices. Our results demonstrate that Adam optimizer, with increased penalty parameter of energy function, in compare to other stochastic optimization methods like SGD works well in practice and improves complexity and accuracy of convergence.

## References

- [1] Erkan, U. (2021). A precise and stable machine learning algorithm: eigenvalue classification (EigenClass). *Neural computing and applications*, 33(10), 5381-5392.
- [2] Datta, B. N. (2010). *Numerical linear algebra and applications* (Vol. 116). Siam.
- [3] Najafi, H. S., & Refahi, A. (2007). FOM-inverse vector iteration method for computing a few smallest (largest) eigenvalues of pair (A, B). *Applied mathematics and computation*, 188(1), 641-647.
- [4] Cichocki, A., & Unbehauen, R. (1992). Neural networks for computing eigenvalues and eigenvectors. *Biological cybernetics*, 68(2), 155-164.
- [5] Feng, J., Chai, Y., & Xu, C. (2021). A novel neural network to nonlinear complex-variable constrained nonconvex optimization. *Journal of the Franklin institute*, 358(8), 4435-4457.
- [6] Xu, C., Chai, Y., Qin, S., Wang, Z., & Feng, J. (2020). A neurodynamic approach to nonsmooth constrained pseudoconvex optimization problem. *Neural networks*, 124, 180-192.
- [7] Qin, S., Feng, J., Song, J., Wen, X., & Xu, C. (2016). A one-layer recurrent neural network for constrained complex-variable convex optimization. *IEEE transactions on neural networks and learning systems*, 29(3), 534-544.

- [8] Qin, S., Fan, D., Wu, G., & Zhao, L. (2015). Neural network for constrained nonsmooth optimization using Tikhonov regularization. *Neural networks*, 63, 272-281.
- [9] Liao, L. Z., Qi, H., & Qi, L. (2004). Neurodynamical optimization. *Journal of global optimization*, 28(2), 175-195.
- [10] Liu, L., Shao, H., & Nan, D. (2008). Recurrent neural network model for computing largest and smallest generalized eigenvalue. *Neurocomputing*, 71(16-18), 3589-3594.
- [11] Feng, J., Yan, S., Qin, S., & Han, W. (2019). A neurodynamic approach to compute the generalized eigenvalues of symmetric positive matrix pair. *Neurocomputing*, 359, 420-426.
- [12] Yi, Z., Fu, Y., & Tang, H. J. (2004). Neural networks-based approach for computing eigenvectors and eigenvalues of symmetric matrix. *Computers & mathematics with applications*, 47(8-9), 1155-1164.
- [13] Wang, X., Che, M., & Wei, Y. (2016). Recurrent neural network for computation of generalized eigenvalue problem with real diagonalizable matrix pair and its applications. *Neurocomputing*, 216, 230-241.
- [14] Najafi, H. S., & Khaleghi, E. (2004). A new restarting method in the Arnoldi algorithm for computing the eigenvalues of a nonsymmetric matrix. *Applied mathematics and computation*, 156(1), 59-71.
- [15] Boisvert, R. F., Pozo, R., Remington, K., Barrett, R. F., & Dongarra, J. J. (1997). Matrix Market: a web resource for test matrix collections. *Quality of numerical software* (pp. 125-137). Springer, Boston, MA.
- [16] Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. <https://arxiv.org/abs/1412.6980>
- [17] Ghogh, B., Karray, F., & Crowley, M. (2019). Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*. <https://arxiv.org/abs/1903.11240>