Thesis for the degree of Master of Science

Convergence of Newton's Method in Solving Quadratic Matrix Equation with Artificial Neural Network

Garam Kim

Department of Mathematics
The Graduate School
Pusan National University

August 2020

Convergence of Newton's Method in Solving Quadratic Matrix Equation with Artificial Neural Network

A Thesis submitted to the graduate school of
Pusan National University in partial fulfillment of
the requirements for the degree of Master of Science
under the direction of Hyun-Min Kim

The thesis for the degree of Master of Science
by Garam Kim
has been approved by the committee members.

July 7th, 2020

Chair	Sangil Kim	_____
Member	Young-Jun Choi	_____
Member	Hyun-Min Kim	_____

Contents

Convergence of Newton's Method in Solving Quadratic Matrix Equation with Artificial Neural Network

Garam Kim

Department of Mathematics
The Graduate School
Pusan National University

Abstract

In this study, we combine numerical method and Artificial Neural Network into one idea. Since the convergence of Newton's method depends on initial matrix, Artificial Neural Network is adopted to compensate its limitation. Hence, we introduce new iteration which is combined with Artificial Neural Network to promise the convergence, and the convergence of new iteration and pure Newton's iteration is compared as a result.

Chapter 1

Introduction

Artificial Neural Network(ANN) is a biologically inspired computer designed to simulate the way in which the human processes information. ANN gathers their knowledge by detecting the patterns and relationships in data and learn(or trained) through experience. An ANN formed from hundreds of single units, artificial neurons connected with coefficients(weights), which constitute the neural structure and are organized in layers [1]. They have the capability to implement massively parallel computations for mapping, function approximation, classification, and pattern recognition processing. ANN can capture the highly nonlinear associations between inputs(predictors) and target(responses) variables and can adaptively learn the complex functional forms [16]. Hence, we attempt to apply ANN as a solver to get the solution of the quadratic matrix equation.

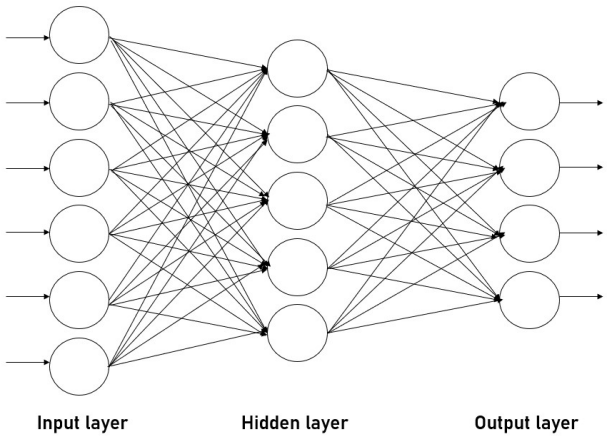

Figure 1.1: Feedforward network which does not have a connection back from the output to input neurons.

The main purpose of this paper is to solve quadratic matrix equation (QME),

$$F(X) \equiv AX^2 + BX + C = 0 \tag{1.0.1}$$

where $A, B, C \in \mathbb{R}^{m\times m}$. If $S \in \mathbb{R}^{m\times m}$ satisfies $F(S) = 0$, then we call that S is a solvent. For a given matrix equation $F(X) = 0$ and a given initial value $X_0 \in \mathbb{R}^{m\times m}$, Newton's iteration is defined by

$$X_{i+1} = X_i - D'^{-1}_{X_i} F(X_i) \tag{1.0.2}$$

where D' denotes the Fréchet derivative of F. To find a numerical solution of Matrix equation, Newton's method is a very natural approach, whose convergence highly relies on the starting matrix. In other words, these simplified Newton's method has poor numerical stability. Moreover, determining the initial matrix which guarantees the convergence of iteration is a highly interesting problem [12]. Therefore, giving a proper X_0 is mightily important and thus we purpose a new algorithm to compute the solvent of QME, which has good numerical stability.

We first recall some notations, definitions as well as theorems related to Newton's method, and introduce pure Newton's algorithm in Chapter 2. Definition, properties and mathematical framework of ANN are explained, and new iteration with ANN is purposed in Chapter 3. A large number of problems in the type of QME arises in several applicants, such as dynamic systems as well as Markov Chain and so on. In numerical experiments, Chapter 4, we consider the following special QME (1.0.3), which is motivated by noisy Wiener-Hopf problem for Markov chains:

$$Q(X) \equiv AX^2 + BX + C = 0, \tag{1.0.3}$$

$$\begin{aligned} \text{where } A &\in \mathbb{R}^{n\times n} \text{ is an identity matrix,} \\ -B &\in \mathbb{R}^{n\times n} \text{ is a diagonal matrix,} \\ -C &\in \mathbb{R}^{n\times n} \text{ is a nonsingular M-matrix,.} \end{aligned}$$

A real square matrix is called a Z-matrix, if all its off-diagonal elements are nonpositive. Suppose that every real eigenvalue of A is positive, then such a matrix is called an M-matrix [11]. More theorems and proofs are in the excellent papers, see [7, 18]. In this chapter, we try two experiments; how the output is close to the solution that comes form trained neural network, and how new iteration with ANN has better convergence performance. More specific, we calculate the average of the frobenius norm of

the difference between predicted solution and the numerical solution, and the mean of the value of QME at the predicted solution. Moreover, we will calculate the number of convergence out of a hundreds of different equations QME, and compared the number of convergence. By the comparison, we can easily check a benefit of Newton's iteration with ANN.

Chapter 2

Numerical Method for Finding Solution

In this chapter, we recall the definition of Fréchet derivative of matrix equation, and introduce the pure Newton's iteration for QME which is one of the basic iteration procedures approximating a solution, as well as its algorithm.

2.1 The Derivatives of Matrix Functions

Definition 2.1.1. [10] The sensitivity of matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ to small perturbation is governed by the Fréchet derivative. The Fréchet derivative at a point $A \in \mathbb{C}^{n\times n}$ is a linear mapping,

$$\begin{aligned} \mathbb{C}^{n\times n} &\xrightarrow{L(A)} \mathbb{C}^{n\times n} \\ E &\longmapsto L(A,E) \end{aligned} \tag{2.1.1}$$

such that for all small $E \in \mathbb{C}^{n\times n}$,

$$f(A+E) - f(A) - L(A,E) = \sigma(\|E\|) \tag{2.1.2}$$

and it therefore describes the first-order effect on f of perturbations in A.

In this paper, we denote $L(A,E)$ as $F'(A)E$ in (2.1.3) and as $F_A(E)$ in the rest of paper. (2.1.2) can be denoted as follows

$$\lim_{E\to 0} \frac{\|F(X+E) - F(X) - F'(X)E\|}{\|E\|} = 0. \tag{2.1.3}$$

For example, recall the quadratic matrix equation $F(X)$ (1.0.1)

$$F(X) \equiv AX^2 + BX + C = 0,$$

where A, B, C and X are $m \times m$ matrices. Then the Fréchet derivative of F is calculated as

$$\begin{aligned}
&\lim_{E\to 0} \frac{\|F(X+E) - F(X) - F'(X)E\|}{\|E\|} = 0 \\
&= \lim_{E\to 0} \frac{\|A(X+E)^2 + B(X+E) + C - AX^2 - BX - C - F'(X)E\|}{\|E\|} = 0 \\
&= \lim_{E\to 0} \frac{\|A(XE + EX + E^2) + BE - F'(X)E\|}{\|E\|} = 0
\end{aligned}$$

$$\Rightarrow F'(X)E = A(XE + EX) + BE$$

We explain $F'(X)E$ as the Fréchet derivative of F at X in the direction E.

2.2 Newton's Method

Let a matrix function $F : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ and matrix equation $F(X) = 0$ be given. For the given initial value X_0, the pure Newton's iteration $\{X_k\}_{k=0}^{\infty}$ of F is defined by

$$\begin{cases} F'_{X_k}(H_k) = -F(X_k), \\ X_{k+1} = X_k + H_k. \end{cases} \qquad k = 0, 1, 2, \ldots \tag{2.2.1}$$

For each $k = 0, 1, 2, \ldots$, $F'_{X_k} : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ is a linear operator. Thus, there is a unique linear representation $\mathbf{F}'_{X_k} : \mathbb{C}^{mn} \to \mathbb{C}^{mn}$ such that $\mathbf{F}'_{X_k}\mathrm{vec}(H_k) = -\mathrm{vec}(F(X_K))$.

Definition 2.2.1. The `vec` operator creates a column vector from a matrix A by stacking column vector of $A = [a_1\ a_2\ \cdots\ a_n]$ below one another:

$$\mathrm{vec}(A) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}.$$

We now introduce pure Newton's method for QME (1.0.1) $F(X)$ according to the idea developed in [3], [4], [9], and it can be stated as

$$\begin{cases} AH_kX_k + (AX_k + B)H_k = -F_k(X_k), \\ X_{k+1} = X_k + H_k. \end{cases} \qquad k = 0, 1, 2, \ldots \tag{2.2.2}$$

The Fréchet derivative of QME (1.0.3) at X in the direction H is given by

$$F'_X(H) = AHX + (AX + B)H.$$

The general approach for solving 2.2.2 is to solve $m^2 \times m^2$ linear system deriven by vec operator and kronecker product [5, 8] such as

$$\mathbf{F}'_{X_k}\mathrm{vec}(H_k) = -\mathrm{vec}(F(X_K)), \tag{2.2.3}$$

where

$$\mathbf{F}'_X = [\,(X^T \otimes A + I \otimes AX) + I \otimes B\,]$$

Also suppose that F'_{X_k}, which is the Fréchet derivative of F at X_k in the direction H_k, is nonsingular [9] then Newton's iteration (2.2.1) can be rewritten as

$$X_{k+1} = X_k + {F'_{X_k}}^{-1}(-F(X_k))$$

which is equivalent to

$$AX_kX_{k+1} + AX_{k+1}X_k + BX_k = AX_k^2 - C.$$

To compute the solvent of $F(X)$, natural approach is to apply Newton's method to $F(X)$. The algorithm of pure Newton's method for QME $Q(X)$ in (1.0.3) is as follows.

Algorithm 2.1: Newton's iteration for equation (1.0.1)

Given X_0, ε *and* $i = 0$
while $\varepsilon < \delta$ **do**
    Solve H_i in the equation of (2.2.3)
    $D_{X_i}(H_i) = -Q(X_i)$
    $X_{i+1} \leftarrow X_i + H_i$
    $i \leftarrow i + 1$
    Calculate δ
end
$X \leftarrow X_i$

Now the natural question is, under which condition does Newton's iteration converge? The next theorem explains the answer to our question.

Theorem 2.2.2 (Local convergence of Newton's method)**.** Assume that $F : \mathbb{R}^n \to \mathbb{R}^n$ is differentiable at each point of an open neighborhood of a solution $\mathbf{x}^*$ of $F(\mathbf{x}) = 0$, that F' is continuous at $\mathbf{x}^*$, and that $F'(\mathbf{x}^*)$ is nonsingular. Then $\mathbf{x}^*$ is a point of attraction of the iteration (2.2.2) and

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 0. \tag{2.2.4}$$

Moreover, if

$$\|F'(\mathbf{x}) - F'(\mathbf{x}^*)\| \leq \alpha \|\mathbf{x} - \mathbf{x}^*\| \tag{2.2.5}$$

for all $\mathbf{x}$ in some open neighborhood of $\mathbf{x}^*$, then there is constant $C < \infty$ such that

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq C\|\mathbf{x}_k - \mathbf{x}^*\|^2 \tag{2.2.6}$$

for all $k \geq k_0$, where k_0 depends on $\mathbf{x}_0$.

The property (2.2.4) is known as superlinear convergence while (2.2.6) is called quadratic convergence. We note that (2.2.5) is ensured if the component function f_i of all F are all twice continuously differentiable in a neighborhood of $\mathbf{x}^*$. Hence, under these mild differentiability assumptions together with the nonsingularity of $F'(\mathbf{x}^*)$ the content of (2.2.2) is that Newton's iteration must converge to $\mathbf{x}^*$. However, despite of intrinsic

theorem (2.2.2) of the iteration method considered, they are generally useless when one tries to ascertain whether an iterative process will converge starting from a given $\mathbf{x}_0$, and what is the error if the process is stopped with the kth iterate. The wonderful book [17] explains various theorems regards error bounds and starting matrix under the specific condition.

Chapter 3

Artificial Neural Network as a Solver

An Artificial Neural Network (ANN) is an information processing mechanism that is inspired by the biological nervous system, such as the brain. The first step toward ANN came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. The key paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements, called neurons, working in harmony to solve specific problem [15]. Let first us look over the elements in the structure of ANN, and understand how does neural network work and learn.

3.1 Introduction to ANN

There are many types of neural networks designed by now and new ones are created every week but all can be described by activation function of their neurons, by the learning rule and by the connecting formula.

3.1.1 Neurons

The artificial neuron is the building component of the ANN designed to simulate the function of the biological neuron. The arriving signals, called inputs, multiplied by the connection weights (adjusted) are first summed and then passed through an activation function to produce the output for the neuron [1].

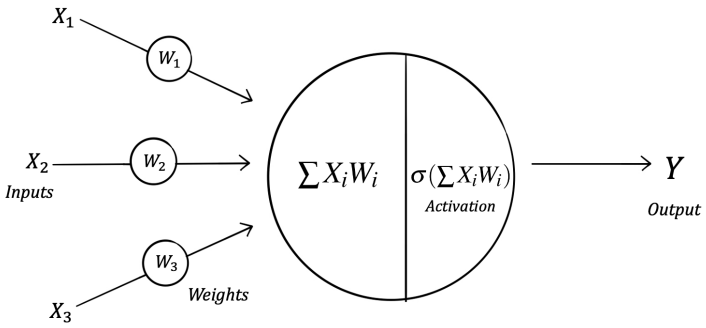

Figure 3.1: Model of an artificial neuron.

3.1.2 Activation Function

Activation function can be linear, threshold or sigmoid function. Sigmoid activation function is usually used for hidden layer because it combines nearly linear behavior, curvilinear behavior and nearly constant behavior depending on the input value [13]. Input to sigmoid is any value between negative infinity and positive infinity number while the output can only be a number between 0 and 1 [20].

3.1.3 Architecture

Figure (1.1) represents the architecture of a simple neural network. It is made up of input, output, and one or more hidden layers. Each node from the input layer is connected to a node from the hidden layer and every node from the hidden layer is connected to nodes in the output layer. The input layer represents the raw information that is fed into the network. Every single input to the network is duplicated and send down to the nodes in the hidden layer. The hidden layer accepts data from the input layer. It uses input values and modifies them using some weight value, this new value is then send to the output layer but it will also be modified by some weight from the connection between the hidden and output layer. Output layer process information received from the hidden layer and produces an output [2].

3.1.4 Learning Rule

A neural network is trained to map a set of input data by iterative adjustment of the weights. Information from inputs is fed forward through the network to optimize the weights between neurons. Optimization of the weights is made by backward propaga-

tion of the error during training or learning phase. The ANN reads the input and output values in the training data set and changes the value of the weighted links to reduce the difference between the predicted and target values. The error in prediction is minimized across may training cycles until the network reaches a specified level of accuracy. If a network is left to train for too long, however, it will overtrain and will lose the ability to generalize [1].

3.2 Mathematical View of ANN

We start by introducing the necessary notations for neural networks which we utilize in this paper.

Notation 3.2.1.

- $L \in \mathbb{N}$ *is the number of layers of the network,*
- $N_l \in \mathbb{N}$*, is the dimension of the output of* l*-th layer for* $l = 1, \ldots, L$*,*
- $\sigma : \mathbb{R} \to \mathbb{R}$ *is the activation function which acts pointwise*
  i.e., $\sigma(y) = [\sigma(y^1), \ldots, \sigma(y^m)]$ *for* $y = [y^1, \ldots, y^m] \in \mathbb{R}^m$*.*

We consider a feedforward architecture where only neurons from neighboring layers can be connected.

Definition 3.2.2. [6] Let $d, L \in \mathbb{N}$.
(1) A neural network Φ with input dimension d and L layers is a sequence of matrix-vector tuples

$$\Phi = ((W_1, b_1), (W_2, b_2), \ldots, (W_L, b_L)),$$

where $N_0 = d$ and $N_1, N_2, \ldots, N_L \in \mathbb{N}$, and where each W_l is an $N_l \times N_{l-1}$ matrix and $b_l \in \mathbb{R}^{N_l}$. Then we call Φ **a standard neural network**.
(2) Then, we define the associated realization Φ with activation function $\sigma : \mathbb{R} \to \mathbb{R}$ as the map $R_\sigma(\Phi) : \mathbb{R}^d \to \mathbb{R}^{N_L}$ such that

$$R_\sigma(\Phi)(x) = x_L,$$

where x_L results from scheme given by

$$\begin{aligned} x_0 &:= x, \\ x_l &:= \sigma(W_l x_{l-1} + b_l), l = 1, \ldots, L-1, \\ x_L &:= W_l x_{L-1} + b_L. \end{aligned}$$

3.3 Newton's Iteration with ANN

In this section, we introduce the idea and algorithm of Newton's iteration with ANN. We trained Neural Network with input as coefficients E, F of equation (1.0.3), and target as the solution X of equation (1.0.3). Then, for arbitrary QME (1.0.3), the network gives output which acts as a starting matrix X_0 for Newton's iteration. Figure 3.2 shows the procedure of Algorithm 3.1.

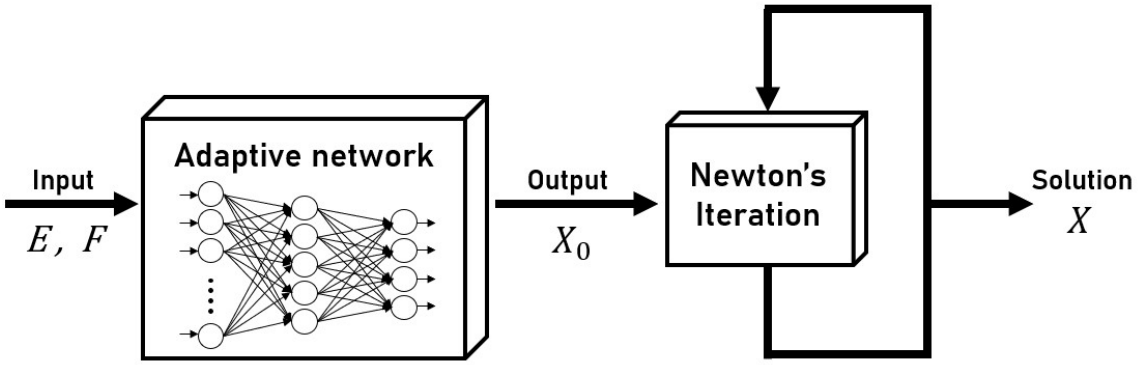

Figure 3.2: The diagram of Newton's iteration with ANN.

Algorithm 3.1: Newton's iteration with ANN for equation (1.0.1)

Input *E, F*
Output X
Given $R_\sigma(\Phi)$, ε *and* $i = 0$
input $\leftarrow$ E, F
$X_0 \leftarrow R_\sigma(\Phi)(input)$
while $\varepsilon < \delta$ **do**
    Solve H_i in the equation of (2.2.3)
    $D_{X_i}(H_i) = -Q(X_i)$
    $X_{i+1} \leftarrow X_i + H_i$
    $i \leftarrow i + 1$
    Calculate δ
end
$X \leftarrow X_i$

The ANN is used to effectively train a neural network through a method called backpropagation which computes the gradient of the loss function with respect to the weights of the network for a single input–output example and does so efficiently. Hence, the output X_0 of ANN is close to the solution, and we can guess that it satisfies the condition "sufficiently close to the solution". Moreover, we expect that Newton's iteration with starting matrix X_0 converges to the solution. For more on backpropagation see Rumelhart et al. [19], and LeCun et al. [14].

Chapter 4

Numerical Experiments

In this chapter, we compare the convergence rate between the pure Newton's iteration (2.1) and the Newton's iteration with ANN (3.1). All of computations are performed by using MATLAB/version R2020a with 2.4 GHz Intel Core i5 CPU and 8GB memory.

We adopted $L = 2$ and $d = n^2 + n$, where $Q(X) \in \mathbb{R}^{n\times n}$. The activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoid, i.e., $\sigma(x) = (1 + exp(-x))^{-1}$. In other words, we introduce a shallow feedforward network with $n^2 + n$ inputs in one hidden layer. Moreover, we set tolerance to $\varepsilon = 10^{-14}$ for break each iteration of numerical experiments. All iterations are terminated with the absolute error δ, when

$$\delta = \|\mathrm{vec}(Q(X_i))\|_2 = \|Q(X_i)\|_F \leq \varepsilon$$

when the absolute error δ is less than the tolerance, where $\delta = \|Q(X_i)\|_F$. Recall the equation (1.0.3), $Q(X) \equiv AX^2 + BX + C = 0$. Let $N_1 = 5$, and $n = 2, 3, 4, 7, 10$.

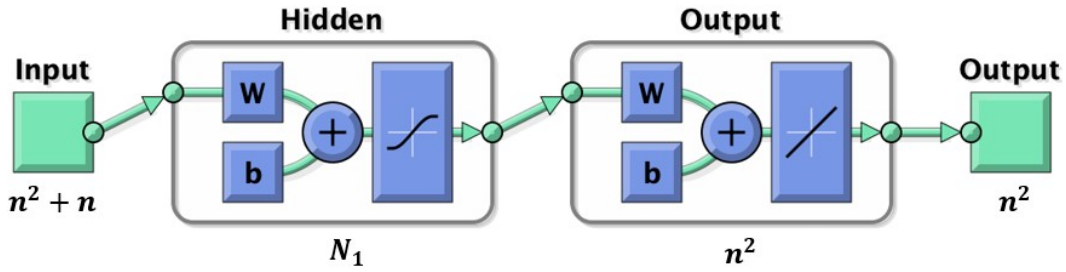

Figure 4.1: The diagram of the shallow feedforward network in MATLAB.

For Algorithm 2.1, random starting matrix X_0 is given. In MATLAB code,

$$
\begin{aligned}
A &= \mathsf{eye(n)},\\
B &= -\mathsf{diag(rand(n,1))},\\
C &= -\mathsf{makeC(n)},\\
X_0 &= \mathsf{rand(n)}.
\end{aligned}
$$

The algorithm for making nonsingular M-matrix C is as follows,

Algorithm 4.1: makeC

Given n
while *1* **do**
  $B_0 \leftarrow rand(n)$
  for $k \leftarrow 1$ **do**
    $B_0 \leftarrow rand \times 10$
  end
  if $eig(B_0) > 0$ **then**
    Break
  end
end
$C \leftarrow B_0$

We trained a shallow feedforward network with coefficients and numerical solutions of various equations, and all matrices are vectorized. The input data is divided randomly into seventy percent of training data, fifteen percent of validation data, and the rest fifteen percent of testing data. We use Mean Square normalized Error (MSE) to measure the network's performance.

Prior to applying Algorithm 2, we first utilize the trained shallow feedforward network as a single solver for QME. The figure (4.2) depicts Frobenius norm of $\|X_{ANN} - X^*\|$ at each dimension, where X_{ANN} is given by trained neural network and X^* is a numerical solution. We repeat five times to get the data as well as the results.

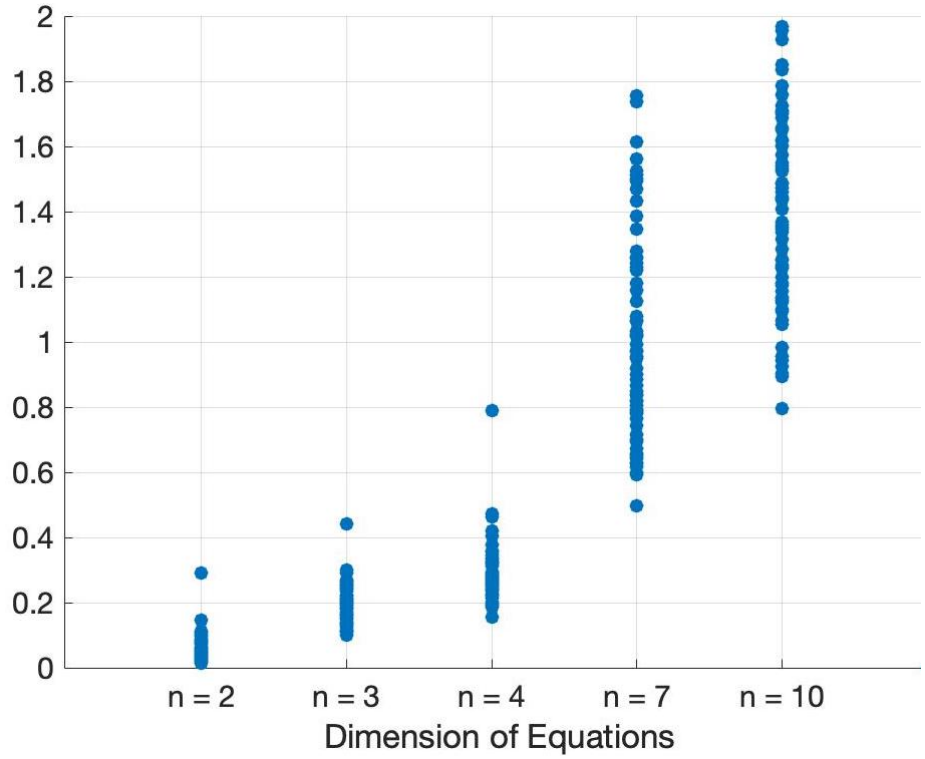

Figure 4.2: The difference between X_{ANN} and X^* at each dimension.

	dimension of the equation				
	2	3	4	7	10
$\|X_{ANN} - X^*\|_F$	0.0609	0.1950	0.2891	0.9703	1.3965
$\|Q(X_{ANN})\|_F$	0.2359	0.7465	1.1114	3.9200	6.1046

Table 4.1: The average of $\|X_{ANN} - X^*\|_F$ and $\|Q(X_{ANN})\|_F$.

Table 4.1 shows the average of the Frobenius norm of the difference between the predicted solution and the numerical solution. Plus, it gives the mean of the value of QME at the predicted solution. X_{ANN} is somewhat close to the numerical solution X^*, but they are numerically different and the mean of the values $Q(X_{ANN})$ far from zero. In other words, the accuracy of the solution is remarkably row. Therefore, it is necessary to improve the structure of the network or take other complementary algorithms to improve it. Based on the result that X_{ANN} is close to the numerical solution, and that Newton's method converges when the initial value and the solvent are sufficiently close, a new iteration with ANN as the initial value is purposed.

We repeat this process for five times and compute the average number for each dimension.

n	Algorithm 2.1	Algorithm 3.1
	Pure Newton's iteration	Newton's iteration with ANN
2	1.4	0
3	5.2	0
5	12.6	0
7	45.6	0
10	98.8	0

Table 4.2: The number of equations does not converge among 100 equations

Table 4.2 shows two explanations and several comments are worth making. First of all, the larger n is, the less Algorithm 2.1 converges with random starting matrices. It shows that as the size of the matrix increases, Newton's iteration converges less. Moreover, in Algorithm 3.1, the iteration converges in all cases, even for larger matrix dimension. It means that neural network gives adequate starting matrices which promises the convergence of iteration.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this paper, we addressed the problem of the numerical instability of Newton's method. One of the main contributions of our work is to combine a numerical method to Artificial Neural Network, which to our idea have never been proposed before, and to purpose the new iteration to overcome the numerical instability. The basic method, Newton's iteration and together with relevant definition and theorem have been presented. Introducing fundamental concepts and mathematical expressions of Neural Network, Newton's iteration with ANN has been designed.

The main focus of our thesis is to apply a trained neural network to give an initial guess of Newton's iteration so that it makes converge. From the numerical example, we show that Newton's iteration with ANN gives the solution of quadratic matrix equation much effectively than the pure Newton's iteration.

5.2 Future Work

Much different research, tests, and experiments have been left before for the future. Future work concerns deeper analysis or new proposals to try different methods. There are some ideas that I would like to try:

1. The way the neural network could be changed or more complicated: instead of using a shallow feedforward network, it could be a deep neural network, in order to improve the accuracy of the network. That way, the neural network would have

significant accuracy so it would be able to use as a single solver.

2. It could be interesting to consider the smallest radius with respect to the convergence of Newton's method by using a neural network. Furthermore, considerable research on local/ global convergence theorem has been taken, and new approach that the initial guess form the mathematical framework of Neural Network has a certain boundary would be a further step of numerical analysis, which is essential to be proved.

Bibliography

[1] Agatonovic-Kustrin, S., and Beresford, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Journal of pharmaceutical and biomedical analysis, 22(5), 717-727, (2000).

[2] Cilimkovic, M. Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 15, (2015).

[3] Davis, G. J. Algorithm 598: an algorithm to compute solvent of the matrix equation AX 2+ BX+ C= 0. ACM Transactions on Mathematical Software (TOMS), 9(2), 246-254, (1983).

[4] Dennis Jr, J. E., and Schnabel, R. B. Numerical methods for unconstrained optimization and nonlinear equations. Society for Industrial and Applied Mathematics, (1996).

[5] Fiedler, M., and Ptak, V. On matrices with non-positive off-diagonal elements and positive principal minors. Czechoslovak Mathematical Journal, 12(3), 382-400, (1962).

[6] Gühring, I., Kutyniok, G., and Petersen, P. Complexity bounds for approximations with deep ReLU neural networks in Sobolev norms, (2019).

[7] Guo, C. H. On a quadratic matrix equation associated with an M-matrix. IMA Journal of Numerical Analysis, 23(1), 11-27, (2003).

[8] Higham, N. J. Accuracy and stability of numerical algorithms. Society for industrial and applied mathematics, (2002).

[9] Higham, N. J., and Kim, H. M. Solving a quadratic matrix equation by Newton's method with exact line searches. SIAM Journal on Matrix Analysis and Applications, 23(2), 303-316, (2001).

[10] Higham, N. J., and Relton, S. D. Higher order Fréchet derivatives of matrix functions and the level-2 condition number. SIAM Journal on Matrix Analysis and Applications, 35(3), 1019-1037, (2014).

[11] Horn, R. A., and Johnson, C. R. Matrix analysis. Cambridge university press, (2012).

[12] Kim, H. M. Convergence of Newton's method for solving a class of quadratic matrix equations. Honam Math. J, 30(2), 399-409, (2008).

[13] Larose, D. T. Discovering Knowledge in Data. New Jersey: John Willey and Sons, (2005).

[14] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551, (1989).

[15] Majumder, M. Artificial neural network. In Impact of Urbanization on Water Shortage in Face of Climatic Aberrations(pp. 49-54). Springer, Singapore, (2015).

[16] Okut, H. Bayesian regularized neural networks for small n big p data. Artificial neural networks-models and applications, (2016).

[17] Ortega, J. M. Numerical analysis: a second course. Society for Industrial and Applied Mathematics, (1990).

[18] Rogers, L. C. Fluid models in queueing theory and Wiener-Hopf factorization of Markov chains. The Annals of Applied Probability, 4(2), 390-413, (1994).

[19] Rumelhart, D. E. Hinton G. E and Williams RJ,". Learning Representations by Backpropagating Errors, 533-536, (1986).

[20] Smith, S. W. The scientist and engineer's guide to digital signal processing, (1997).

인공신경망을 이용한 행렬 이차 방정식의 수렴

김 가 람

부산대학교 대학원 수학과

요약

이 연구에서 우리는 이차 행렬방정식의 해를 구하기 위하여 수치적인 방법과 인공 신경망을 하나의 아이디어로 결합한다. 뉴턴의 반복법은 초기 행렬에 따라 수렴이 결정되므로, 그 한계를 보완하기 위하여 인공신경망을 도입한 새로운 반복법을 소개한다. 그리고 뉴턴 반복법과 인공신경망이 도입된 새로운 반복법의 수렴 결과를 비교한다.