East Asian Math. J.
Vol. 38 (2022), No. 1, pp. 095–105
http://dx.doi.org/10.7858/eamj.2022.008

DEEP LEARNING APPROACH FOR SOLVING A QUADRATIC MATRIX EQUATION

Garam Kim and Hyun-Min Kim*

Abstract. In this paper, we consider a quadratic matrix equation $Q(X) = AX^2 + BX + C = 0$ where $A, B, C \in \mathbb{R}^{n\times n}$. A new approach is proposed to find solutions of $Q(X)$, using the novel structure of the information processing system. We also present some numerical experimetns with Artificial Neural Network.

1. Introduction

Over the past years, we focus on solving a wide range of linear systems and have attempted to find ways to solve the quadratic matrix equation. Many of the results that discuss some properties of quadratic matrix equation have been shown, and according to the results, many algorithms are introduced with some restrictions. There are no entirely acceptable methods; there are some efficient ones. Our subject here turns to the nonlinear quadratic matrix equation. Define a quadratic matrix equation by

$$Q(X) \equiv AX^2 + BX + C = 0 \tag{1.1}$$

where $A, B, C \in \mathbb{R}^{n\times n}$. Then we wish to have a generalized formula to (1.1), just like the usual rule for the roots of a scalar quadratic equation. Sad to say, however, it is established in particular case when $A = I$, B commutes with C and $B^2 - 4C$ has a square root. Besides, we should note that (1.1) could have no solution, a finite positive number, or infinitely many [11]. In other words, the existence and characterization of the solution are not straightforward, and computing a solution poses an interesting challenge [12]. There has been much effort to identify the sufficient condition for the existence of the solution: Eisenfeld [6] uses the contraction mapping principle to show that if A and B

Received December 24, 2021; Accepted January 12, 2022.
2010 *Mathematics Subject Classification.* 15A24, 65F10, 65H10.
Key words and phrases. quadratic matrix equation, artificial neural network, deep learning, Newton's method.

*Corresponding author.

This work was supported by a 2-Year Research Grant of Pusan National University.

are nonsingular and

$$4||B^{-1}A||\,||B^{-1}C|| < 1 \tag{1.2}$$

for a subordinate matrix form, then there exist at least two solutions. And a similar but more restrictive conditions as well as several sets of sufficient condition for the existence without restriction are introduced [18], [20]. Likewise, research on the complete theorem that guarantee the existence of solutions is still ongoing.

Another interesting part would be the way to finding a solution. One of the candidate is using a Symbolic Math ToolBox in MATLAB. Of course, it is powerful for $n = 2$, but this approach is able to find only for very special A, B and C when $n \geq 3$ [12]. Then a natural approach for solving the quadratic matrix equation is Newton's method, which has been investigated for this from a half century ago. While newton's method is very attractive for solving the quadratic matrix equation, it has some weakness.

In this work, we take a new approach to find solutions of $Q(X)$, using the novel structure of the information processing system. Artificial Neural Network solves problems that are difficult for human beings but relatively straightforward for computers [7]; the network comprises a large number of interconnected elements under solid mathematical rules. We review the noble numerical method, Newton's method with powerful but imperfect results, and investigate how neural networks learn. From the numerical experiments, we conclude by describing the relationship with the initial values that affect the convergence of Newton's method and its development.

2. Theory

The quadratic matrix equation can be solved under the special condition, when $A = I$, B commutes with C (i.e. $BC = CB$) and $B^2 - 4C$ has a square root. We can get the solution

$$X = -\frac{1}{2}B + \frac{1}{2}\sqrt{B^2 - 4C}\,,$$

where $\sqrt{M}$ denotes any square root of M.

Unfortunately, there is no generalization of the formula for the solution of quadratic matrix equation, with general A, B and C. The general information about the existence of solvents comes from the connection between the quadratic matrix equation and the quadratic eigenvalue problem. If $Q(\lambda)$ has a linearly independent eigenvectors, $v_1, \cdots, v_n$, then $Q(X)$ has a solvent [13]. However, Kim [16] explains that not all cases that eigenvectors corresponding to distinct eigenvalues are linearly independent, so constructing solvent is complicated.

The following result shows that all solvents of $Q(X)$ can be constructed by the computationally satisfactory generalized Schur decomposition. We first define

$$F = \begin{bmatrix} 0 & I \\ -C & -B \end{bmatrix} \qquad \text{and} \qquad G = \begin{bmatrix} I & 0 \\ 0 & A \end{bmatrix}.$$

Theorem 2.1. [12, Theorem 3] *All solvent of $Q(X)$ are of the form $X = W_{21}W_{11}^{-1} = W_{11}T_{11}S_{11}^{-1}W_{11}^{-1}$, where*

$$W^*FZ = T, \qquad W^*GZ = S$$

is generalized Schur decomposition with W and Z unitary and T and S upper triangular, and where all matrices are partitioned as block 2×2 matrices with $n \times n$ blocks.

Moreover, we are able to find all solvents using `solve` command of Symbolic Math ToolBox in MATLABbut symbolic solution is clearly impractical for large n.

A natural approach for solving the quadratic matrix equation is Newton's method, which has been investigated by [3]. The method can be first off derived by the derivative of $Q(X)$.

Definition 2.2. *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,*

$$\begin{aligned} \mathbb{C}^{n\times n} &\xrightarrow{D_A} \mathbb{C}^{n\times n} \\ E &\longmapsto D_A(E) \end{aligned} \tag{2.1}$$

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

$$f(A+E) - f(A) - D_A(E) = \sigma(\|E\|). \tag{2.2}$$

Such small perturbation affects in the original data on the solvents of $Q(X)$ and can arise from many sources, including errors in measurement and inaccuracy involved by generating matrices in a computer [3].

The Fréchet derivative of the quadratic matrix equation is obtained from the expansion

$$\left.\begin{aligned} AE_iX_i + (AX_i + B)E_i &= -Q(X_i) \\ X_{i+1} &= X_i + H_i \end{aligned}\right\} \quad i = 0, 1, 2, \ldots. \tag{2.3}$$

The general approach for solving (2.3) is to solve

$$D_X(E) = AEX + (AX + B)E$$

using $n^2 \times n^2$ linear system driven by vec operator and kronecker product,

$$\mathbf{D}'_{X_i}\operatorname{vec}(E_i) = -\operatorname{vec}(Q(X_i)), \tag{2.4}$$

where

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

Supposing $\mathbf{D}'_X$ is nonsingular, the newton's iteration can be rewritten as

$$X_{i+1} = X_i + {\mathbf{D}'_{X_i}}^{-1}(-Q(X_i))$$

which is equivalent to

$$AX_iX_{i+1} + AX_{i+1}X_i + BX_i = AX_i^2 - C.$$

Standard convergence results for Newton's method states that if $\mathbf{D}'_X$ is nonsingular, then for the starting matrix of the newton's method is sufficiently close to the solution, and the iteration converge quadratically to the solvent [17].

Theorem 2.3. [22] *Let f be Fréchet differentiable on D, and f' be Lipschitz continuous. Let x^* be a zero of f such that $f'(x^*)$ is nonsingular. Then there exists a radius r such that the Newton's method started from any x_0 with $||x - x_0|| < r$ converges to x^* quadratically.*

However, the inverse of the derivative must be evaluated at each iteration, which comes at a cost, and some modification is necessary to achieve convergence from initial values not very near a solution [5].

This analysis breaks down if $\mathbf{D}'_X$ is singular. A singular $\mathbf{D}'_X$ represents an extreme case of sensitivity in that small perturbations in the data can produce unbounded changes in the solvents. There is no question here of subroutine accuracy or of errors in computation; these are fundamental properties of the problem itself.

3. Deep Learning Approach of solving $Q(X)$

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. 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.

Notation 3.1.

- *$L \in \mathbb{N}$ 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., $\sigma(y) = [\sigma(y^1), \ldots, \sigma(y^L)]$ for $y = [y^1, \ldots, y^L] \in \mathbb{R}^L$.*

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

Definition 3.2. [10] *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

$$\Phi = ((W^1, b^1), (W^2, b^2), \dots, (W^L, b^L)),$$

where $N_0 = d_{\mathbf{x}}$ and $N_1, N_2, \dots, N_L \in \mathbb{N}$, 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 fully connected and feedforward neural network. The neural network Φ is said to be a deep neural network if $L \geq 2$.
(2) Then, we define the associated realization Φ with activation function $\sigma : \mathbb{R} \to \mathbb{R}$ as the map $R_\sigma(\Phi) : \mathbb{R}^{d_{\mathrm{x}}} \to \mathbb{R}^{N_L}$ such that

$$R_\sigma(\Phi)(x) = \mathrm{x}^L,$$

where x^L results from scheme given by

$$\begin{aligned}
&\mathrm{x}^0 := \mathbf{x}, \\
&\mathrm{x}^l := \sigma(W^l \mathrm{x}^{l-1} + b^l), l = 1, \dots, L-1, \\
&\mathrm{x}^L := W^L \mathrm{x}^{L-1} + b^L.
\end{aligned}$$

Here the activation function σ is a nonlinear function and popular choices of σ are the rectified linear unit (ReLU) function $\mathrm{ReLU}(x) = \max(x, 0)$ and the sigmoid function $\mathrm{Sigmoid}(x) = (1 + \exp(-x))^{-1}$.

Cybenko [2] claims that a shallow feedforward network which has a single hidden layer, with sigmoid activation function can approximate any Borel measurable function from one finite-dimensional space to another with almost zero amount of loss. The output units are always assumed to be linear. If the number of hidden layer is more than two, then we call the network as deep learning network.

Theorem 3.3 (Universal Approximation Theorem)**.** *Let I_n denote n-dimensional unit cube, $[0,1]^n$ and σ be any continuous sigmoid function. Then the finite sums of the form*

$$G(x) = \textstyle\sum_{j=1}^N \alpha_j \sigma(W_j x + b_j)$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \epsilon \quad \textit{for all} \quad x \in I_n.$$

Leshno et al. showed in 1993 [19] that it is not the specific choice of the activation function, but rather the multilayer feed-forward architecture itself which means the network has to be large enough to being approximated [14]. However, Barron [1] claims that the theorem does not say how large this network will be. The size of network is to be considered how depth the network is, or how many number of nodes are. From the experiment by Goodfellow et al. [7], empirical result shows that the deeper network generalize better. However, the result shows that in the fully connected layer, test accuracy of the network does not change even the number of parameter increases. In other words, deeper model tends to be perform better, but it is not merely because of larger model. Specially, it expresses a belief that the function should be consist of many simpler functions composed together [7].

Summarizing what we define in this section, for a given network with depth L, we denote the result of each layer as

$$\mathsf{x}^l = \sigma(W^l \mathsf{x}^{l-1} + b^l), \quad l = 1, 2, \cdots, L,$$

where weight, the 'connection strength' to each note to the next layer, be

$$W^l = \begin{bmatrix} W^l_{1.1} & \cdots & W^l_{1.N_{l-1}} \\ \vdots & \ddots & \vdots \\ W^l_{N_l.1} & \cdots & W^l_{N_l.N_{l-1}} \end{bmatrix} \begin{bmatrix} \mathsf{x}^{l-1}_1 \\ \vdots \\ \mathsf{x}^{l-1}_{N_{l-1}} \end{bmatrix} + \begin{bmatrix} b^l_1 \\ \vdots \\ b^l_{N_{l-1}} \end{bmatrix}.$$

Along the network's layer, the final function is denoted as

$$f(\mathbf{x}) = W^L(W^{L-1}\sigma(\cdots(\sigma(W^2\sigma(W^1\mathbf{x} + b^1) + b^2)\cdots) + b^{L-1}) + b^L.$$

3.2. Convergence of ANN

Section 3.1 introduces what types of function can be approximated by neural network. However, in a real-world, these networks are used to estimate functions which we do not know how to write down analytically [8]. Hence, we will outline the most common method of training neural network.

In the feedforward network, the input $\mathbf{x}$ provides the initial information that flows forward through to the hidden unit. Once the network weights and biases have been initialized, the network is ready for training. During training, the weights and biases of network are iteratively adjusted to minimize the network performance and finally produce $\hat{\mathbf{y}}$ (it is more general notation for the output x^L in the definition and its dimension denotes by $d_{\mathbf{y}}(= N_L)$). In order to determine how well a prediction by the iteration is, we may establish a performance function (loss function).

For a given training set, $(\mathbf{x}_i,\ \mathbf{y}_i)_{i=1}^m \subset \mathbb{R}^{d_{\mathbf{x}}} \times \mathbb{R}^{d_{\mathbf{y}}}$, we would like to learn a output $\hat{\mathbf{y}}$ from a parametric family $\mathcal{H} := \{\theta \mid \theta \in \{W^l_{i.j}, b^l_i\},\ l = 1, 2, \cdots, L\}$ by minimizing performance function. The most common function is Mean Square Error (MSE), which measures the networks performance according to the mean of squared errors:

$$\min_{\theta \in \mathcal{H}} L(\theta) := \frac{1}{L}\sum_{n=1}^{L} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2, \quad \text{where } \hat{\mathbf{y}} = f(\theta, \mathbf{x}).$$

We train all layers by Gradient Descent (GD) method, for $k = 1, 2, \cdots$ be the each iteration, and $l \in \{1, 2, \cdots, L\}$, then all weights and biases are updated as follows:

$$W^l(k) = W^l(k) - \eta \frac{\partial L(\theta)(k-1)}{\partial W^l(k-1)},$$

$$b^l(k) = b^l(k) - \eta \frac{\partial L(\theta)(k-1))}{\partial b^l(k-1)},$$

where $\eta > 0$ is the learning rate.

During training, the weights and biases of network are iteratively adjusted to minimize the network performance. More generalized notation for GD method is defined by the equation below:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \quad \text{where} \quad f : \mathbb{R}^N \to \mathbb{R}.$$

Here f is the form of the minimization problem $\min_{x \in \mathbb{R}^N} f(x)$, and $f(x) \in \mathcal{C}^2$.

GD is one of the well-known algorithm to attack optimization problem when the point x so that $\nabla f(x) = 0$, then we call critical point. The set of local minima of f is a subset of critical points of f. The algorithm may converge to a critical point which is not a local minimum, called a saddle point [23]. Hence, the interplay between the saddle points and the performance of GD is a critical and not well-understood aspect of non-convex optimization. Despite our incomplete theoretical understanding, in practice, the intuitive nature of the GD method makes it a basic tool for attacking non-convex optimization problem, where we have very little understanding of the geometry. The worst case, converging to saddle point [21], exists in theory; practitioners have fairly successful at applying these techniques across a wide variety of problems [25]. Moreover, many alternative methods have been introduced to be faster and minimizing a combination of squared error and weights, et cetera. Generally, the train stops when any of these conditions occurs; when the maximum number of epochs (repetitions) is reached; when the maximum amount of time is exceeded; when performance is minimized to the goal; when the performance gradient falls below given scalar [4];

4. Numerical Experiments and Concluding remarks

A large number of problems in the type of quadratic matrix equation arise in several applicants, such as dynamic systems as well as Markov Chain and so on. We consider the following special QME, which is motivated by noisy Wiener-Hopf problem for Markov chains:

$$A = I_n, \quad B = \operatorname{diag}(a_1, \cdots, a_n), \quad C = M - s\, I_n \tag{4.1}$$

where

$$s \geq \rho(M)$$

and ρ denotes the spectral radius. More theorems and proofs are in the excellent papers, see [9, 26]. In this chapter, we try two experiments; the accuracy of network according to the dimension of the equation, and how the output of network close to the numerical solution.

We adopted $L = 5$, $d_{\mathbf{x}} = n^2 + n$ and $(N_1, N_2, N_3, N_4, N_5) = (100, 70, 50, 10, n^2)$. The activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoid, i.e., $\sigma(x) = (1 + \exp(-x))^{-1}$. In other words, we make deep learning structure with 4 hidden layers, $n^2 + n$ inputs and n^2 outputs. We trained a network with coefficients and numerical solutions of thousands of equations, and all matrices are vectorized. The input data is divided randomly into seventy percent of training data, fifteen percent

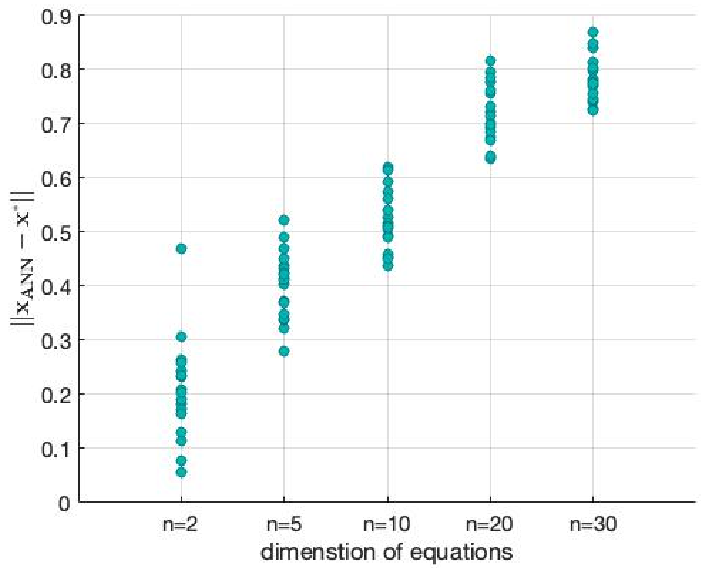

FIGURE 1. The difference between X_{ANN} and X^* at each dimension.

of validation data, and fifteen percent of testing data. We use Mean Square normalized Error (MSE) to measure the network's performance. The figure 1 depicts 2-norm of $\|X_{ANN} - X^*\|$ at each dimension, where X_{ANN} is given by trained neural network and X^* is the corresponding numerical solution.

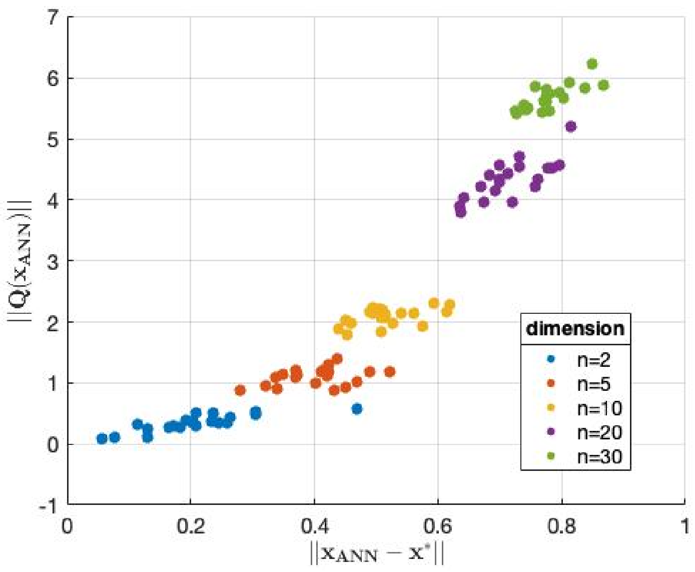

FIGURE 2. The relationship between $\|X_{ANN} - X^*\|$ and $\|Q(X^*)\|$ at each dimension.

Figure 1 shows how well our network predicts solutions. The larger the matrix is, the more parameters there are, the further away X_{ANN} is from the numerical solution X^*. Here we would empirically draw an assumption of the convergence radius of Newton's method, mentioned in Theorem 2.3, which we should prove in future work.

The possibility of using Newton's method for small problems has been investigated by Santosa [27]. When the matrix is large, Newton's algorithms involve the computation and inversion of large matrix [24]. Furthermore, we check the norm of the equation at X, making the dimension large. Even though X is not entirely functional for the true solution, it leaves a hypothesis that it could be utilized as a close initial guess.

References

[1] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[3] G. J. Davis. Numerical solution of a quadratic matrix equation. *SIAM J. Sci. Stat. Comput.*, 2(2):164175, June 1981.

[4] H. B. Demuth, M. H. Beale and M. T. Hagan. *Deep Learning Toolbox User's Guide.* Natick, Massachusetts, United State, March 2021.

[5] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Society for Industrial and Applied Mathematics, 1996.

[6] J. Eisenfeld. Operator equations and nonlinear eigenparameter problems. *Journal of Functional Analysis*, 12(4):475–490, 1973.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press.

[8] L. F. Guilhoto. An overview of artificial neural networks for mathematicians. 2018.

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

[10] I. Gühring, G. Kutyniok and P. Petersen. Error bounds for approximations with deep ReLU neural networks in $w^{s,p}$ norms. 2019.

[11] N. J. Higham. Computing real square roots of a real matrix. *Linear Algebra and its Applications*, 88-89:405–430, 1987.

[12] N. J. Higham and H.-M. Kim. Numerical analysis of a quadratic matrix equation . *IMA Journal of Numerical Analysis*, 20(4):499–519, 10 2000.

[13] N. J. Higham and H.-M. Kim. 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.

[14] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[15] T. Jaware, V. Patil and R. Badgujar. *Artificial Neural Network.* LAP Lambert Academic Publishing, 2019.

[16] H.-M. Kim. Convergence of Newtons method for solving a class of quadratic matrix equations. *Honam Mathematical Journal*, 30(2):399–409.

[17] W. Kratz and E. Stickel. Numerical Solution of Matrix Polynomial Equations by Newton's Method. *IMA Journal of Numerical Analysis*, 7(3):355–369, 1987.

[18] P. Lancaster and J. G. Rokne. Solutions of nonlinear operator equations. *SIAM Journal on Mathematical Analysis*, 8(3):448–457, 1977.

[19] M. Leshno, V. Ya. Lin, A. Pinkus and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867, 1993.

[20] J. E. McFarland. An iterative solution of the quadratic equation in banach space. *American Mathematical Society*, 9:824830, 1958.

[21] Y. Nesterov. *Introductory lectures on convex optimization: a basic course; 1st ed.* Applied optimization. Springer, Boston, 2004.

[22] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Society for Industrial and Applied Mathematics, 2000.

[23] I. Panageas and G. Piliouras. Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions, 2016.

[24] R. G. Pratt, C. Shin and G. J. Hick. GaussNewton and full Newton methods in frequencyspace seismic waveform inversion. *Geophysical Journal International*, 133(2):341–362, 05 1998.

[25] A. Ravindran, K. M. Ragsdell and G. V. Reklaitis. *Engineering Optimization: Methods and Applications, Second Edition.* John Wiley & Sons, Inc., 2006.

[26] L. C. G. Rogers. Fluid models in queueing theory and wiener-hopf factorization of markov chains. *The Annals of Applied Probability*, 4(2):390–413, 1994.

[27] F. Santosa. W. W. Symes and G. Raggio. Inversion of band-limited reflection seismograms using stacking velocities as constraints. *IOP Publishing Ltd*, 3(3):448–457, 1977.

Garam Kim
Department of Mathematics, Pusan National University
Busan 46241, Republic of Korea
E-mail address: garam.tolba@gmail.com

Hyun-Min Kim
Department of Mathematics, Pusan National University
Busan 46241, Republic of Korea
E-mail address: hyunmin@pusan.ac.kr