Fractional-Order Backpropagation through Time for Training Recurrent Neural Networks

A. Abirami*, S. David Samuel Azariya

**Assistant Professor, Department of Mathematics, Sona College of Technology, Salem. abiramia@sonatech.ac.in*
Assistant Professor, Department of IT, Sona College of Technology, Salem. david@sonatech.ac.in

Recurrent neural networks (RNNs) have been shown to be highly effective for sequence-based tasks in recent years. However, because of long-term dependen-cies and the infamous disappearing and expanding gradient concerns, training RNNs presents unique difficulties. Backpropagation through Time (BPTT), his-torically used to optimize these networks, shows its limits in more complicated situations. This research presents a unique BPTT modification known as FO-BPTT (Fractional-Order BPTT). FO-BPTT improves stability and convergence by addressing some of the inherent constraints of ordinary BPTT and utilizing the robust mathematical framework of fractional-order calculus. Our exhaustive tests on several datasets show that FO-BPTT performs better than its conventional cousin in several benchmarks. Furthermore, our results point to a significant role for fractional order in shaping learning dynamics, opening up new avenues for hyperparameter optimization. This study not only lays the road for improved RNN training but also suggests that fractional-order calculus may be helpful in other neural network paradigms.

Keywords: Recurrent Neural Networks , Fractional-Order Backpropagation through Time, Sequence Modeling, Fractional-Order Calculus, Hyperparameter Optimization.

1. Introduction

In deep learning, recurrent neural networks (RNNs) have become the dominant architecture, especially for tasks involving sequences like speech recognition, natural language processing, and time series forecasting. These networks are ideally suited for sequences because they naturally capture temporal relationships. Nevertheless, despite its potential, training RNNs successfully is still challenging due to difficulties like the disappearing and ballooning gradient problems, particularly in situations where there are long-term dependencies.

Backpropagation through Time (BPTT) is the traditional method for training RNNs [1]. By considering the RNN as a deep feed-forward network with shared weights, BPTT unfolds the

RNN over time. However, the gradients calculated by BPTT either vanish or expand as the depth (measured in time steps) of this unfolded network rises, making the network challenging to train. To address this, several initiatives have been launched, including architectural modifications like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units), as well as optimization methods like gradient cutting. However, there is still potential for development, especially in the optimization algorithm.

Fractional-order calculus, a fascinating area of mathematics, presents an intriguing path in this situation. Traditional calculus only works with derivatives and integrals of integer orders, while fractional-order calculus expands this idea to non-integer orders, opening up new possibilities for neural network optimization. The essence of FO-BPTT lies in its ability to capture the memory and dynamics of sequential data with greater granularity [2]. By introducing fractional-order derivatives into the backpropagation process, the algorithm adapts the learning rates across time steps, potentially mitigating issues associated with exploding or vanishing gradients. This adaptability is particularly advantageous in scenarios where long-term dependencies or subtle patterns are crucial for effective model training [3-5]. This study investigates how to include fractional-order calculus into the BPTT algorithm, leading to the formulation of the Fractional-Order BPTT (FO-BPTT) proposal.

This paper explores the conceptualization of FO-BPTT, its theoretical foundations, and its real-world applications. We seek to shed light on its benefits over the conventional BPTT and its potential to revolutionize RNN training through intensive testing. As we proceed with this investigation, we expect to uncover additional opportunities for using fractional-order calculus's capabilities in the larger field of neural network training.

2. Background and Related Work

2.1 Brief Review of RNN Architectures

The Recurrent Neural Network (RNN) is the fundamental building block of deep learning sequence modelling. RNNs feature connections that loop back, in contrast to conventional feed-forward networks, which enables them to keep a "memory" of prior inputs in their internal state [6]. They are suitable for jobs like text creation and time-series forecasting because of their recurring nature, which enables them to handle sequences of different lengths.

Standard RNNs do have certain drawbacks, however. When managing lengthy sequences, its structure is vulnerable to the well-known problems of vanishing and bursting gradients [7]. More complex RNN variations have been created to address these issues. These two well-known changes, the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU) are made to capture long-term dependence better [8].

2.2 Standard BPTT and its Limitations

The Backpropagation Through Time (BPTT) technique is often used while training RNNs [9]. BPTT is a backpropagation modification often employed in feed-forward networks and is made specifically for RNNs because of its recurring nature. The RNN is time-unfolded,

and the backpropagation algorithm is then applied to the structure that has been time-unfolded.

However, the depth of this unfolding network increases with the sequence's length. The difficulties of disappearing and bursting gradients are exacerbated by this increasing depth, making it challenging to train RNNs on lengthy sequences. In order to address these issues, several strategies have been put forth, such as gradient cutting and skip connections, although they are more like temporary fixes than permanent answers.

2.3 Existing Modifications to BPTT

There have been several attempts throughout time to enhance or change BPTT. Truncated BPTT, for example, reduces computing overhead by breaking sequences into smaller parts, but at the cost of some long-term dependence modelling. The Real-Time Recurrent Learning (RTRL) technique, which updates weights at every time step but is computationally more expensive, is another significant method [10].

When training deep networks, including RNNs, several adaptive optimisation techniques, including Adam, RMSProp, and AdaGrad, have been developed to enhance the convergence qualities. However, rather than altering the gradient calculation itself, these strategies emphasise the adaptive learning rate more.

2.4 Introduction to Fractional-Order Derivatives and their Applications

The ideas of differentiation and integration from integer orders to real or complex orders are extended in fractional-order calculus. A new degree of flexibility is introduced by this generalization, which has been used in physics, engineering, and control theory to increase modelling capabilities and resilience.

Fractional-order calculus in machine learning and optimization has recently gained popularity [11]. The potential of fractional-order gradient descent to improve the training of feed-forward neural networks has been demonstrated in preliminary research. Fractional derivatives present an attractive opportunity for enhancing the training dynamics of deep learning models, notably RNNs, due to their capacity to capture non-local interactions and memory effects.

3 Fractional-Order Calculus

3.1 Definition and Mathematical Properties

Integer-order derivatives are dealt with in ordinary calculus when the order of differentiation is a whole number. This idea is expanded in fractional-order calculus to include derivatives and integrals of non-integer orders. The Riemann-Liouville definition is frequently used to define the fractional derivative:

$$D^{\alpha}f(t) = \frac{1}{\Gamma(n-\alpha)}\int_0^t \frac{f^1(\tau)d\tau}{(t-\tau)^{\alpha-n+1}} \quad (1)$$

Where:

- α is the order of the fractional derivative and is a real number
- Γ represents the gamma function.
- n is the smallest integer greater than α

Using the Riemann-Liouville method, the function f′ is integrated against a power-law kernel. The fact that this kernel exists demonstrates the "memory" that fractional-order systems have by nature.

3.2 Earlier Applications in Optimization and Control

Due to its intrinsic capacity to capture the dynamics of systems more comprehensively, fractional-order calculus has found applications in several different fields. When a system displays unusual behaviors or long-term memory effects, fractional-order controllers frequently outperform integer-order equivalents.

The optimization of complicated systems is one such application. The addition of flexibility provided by the fractional-order differential equations allows the optimization algorithms to avoid local minima and provide more accurate results.

The fractional gradient in optimization can be described mathematically as:

$$\nabla^{\alpha} J(\theta) = -\eta D^{\alpha} L(\theta) \quad (2)$$

Where:

- $\nabla\alpha J(\theta)$ is the fractional gradient of the objective function J.
- L is the loss function.
- η is the learning rate.

3.3 Fractional-order Gradient Descent in Feedforward Networks

In order to optimise neural networks, applying the idea of fractional-order derivatives is a promising option. Using the negative gradient of the loss with respect to the weights, the traditional gradient descent approach updates the weights for feedforward networks. The following can be used to describe introducing a fractional-order gradient descent:

$$\theta_{t+1} = \theta_t - \eta \nabla^{\alpha} L(\theta_t) \quad (3)$$

Where:

- θ_t represents the weights of the network at iteration t.
- $\nabla^{\alpha} L(\theta_t)$ is the fractional-order derivative of the loss function L with respect to the weights θ_t.

According to preliminary experiments on feedforward networks, this fractional-order gradient descent can offer more robust convergence features, mainly when the loss landscape is rocky or contains a lot of local minima.

4 Fractional-Order BPTT (FO-BPTT)

4.1 Algorithm Formulation

Given an RNN described by the state transition:

$$h_t=f(W_{xt}+Uh_{t-1}+b) \quad (4)$$

The conventional BPTT computes the gradient ∇L of the loss function L using the chain rule over time steps, where x_t is the input at time t, ht is the hidden state, and W, U, and b are trainable parameters.

The fractional-order derivative is used in FO-BPTT to alter the gradient computation. The update rule is expressed as follows:

$$\theta_{t+1}=\theta_{t-\eta}\nabla\alpha L_t(\theta_t) \quad (5)$$

Where ∇^{α} represents the fractional-order gradient and η is the learning rate.

4.2 Theoretical Justification for FO-BPTT

FO-BPTT was primarily inspired by fractional derivatives' "memory-preserving" capability. Since RNNs naturally deal with sequences and temporal dependencies, having a derivative that captures both local and more distant or historical behavior is preferable.

The Riemann-Liouville fractional derivative is an integral mathematical operator that can capture non-local interactions. This helps mitigate problems like the disappearing gradient problem when gradients need to be spread across lengthy sequences.

Table 1. Comparison with Standard BPTT in Terms of Stability and Convergence

Metric/Algorithm	Standard BPTT	FO-BPTT
Convergence Speed	Moderate	Faster (due to memory effect)
Stability in Training	Can fluctuate with long sequences	Enhanced stability
Robustness to Initializations	Sensitive	More robust
Sensitivity to Hyperparameters	High	Reduced
Handling of Long Sequences	Often problematic	Improved long-term gradient propagation

5 Experimental Setup

5.1 Datasets and RNN Architectures Used

Datasets:

Time-Series Forecasting Dataset (TSFD): A synthetic dataset generated using sine and cosine functions with added noise. It has 10,000 sequences each of length 100.

Sentiment Analysis Dataset (SAD): Extracted from a collection of movie reviews, it comprises 50,000 samples labeled either as positive or negative sentiment. Each sample is a sequence of words, with sequences varying in length from 10 to 500 words.

RNN Architectures:

Standard RNN: Basic recurrent layers with tanh activation.

LSTM: Long Short-Term Memory layers designed to capture long-term dependencies.

GRU: Gated Recurrent Units, another variant optimized for long sequences.

5.2 Training Parameters and Evaluation Metrics

Parameters:

- Learning rate (η): 0.001 for TSFD and 0.0005 for SAD.
- Batch size: 64 for both datasets.
- Fractional order (α): Varied from 0.5 to 1.5 to investigate the effect of different fractional orders.
- Number of epochs: 50 for TSFD and 30 for SAD.

Evaluation Metrics:

- Mean Squared Error (MSE) for TSFD:

$$\text{MSE}=\frac{1}{\sum_{i=1}^{N}(y_i-y^i)},$$

Where y_i is the actual value and y^i is the predicted value.

- Accuracy for SAD:

$$\text{Accuracy}=\frac{\text{Number of Correct Prediction}}{\text{Total Number of Predictions}}\times 100$$

Where:

- Number of Correct Predictions: The count of instances where the model correctly predicted the sentiment (or class) of the text.
- Total Number of Predictions: The total count of predictions made by the model.

The accuracy is usually expressed as a percentage. It provides a measure of how well the model is performing in terms of correctly classifying instances.

5.3 Implementation Details

The neural networks were implemented using the TensorFlow framework. For the fractional-order gradient computations, a custom optimizer was developed based on the TensorFlow's built-in optimizers.

FO-BPTT Algorithm:

1. Unfold the RNN for tt time steps.
2. Compute the standard gradient ∇L∇L using the chain rule.
3. Compute the fractional gradient $\nabla_\alpha L \nabla_\alpha L$ based on the Riemann-Liouville definition.
4. Update the weights: $\theta_{t+1}=\theta_{t-\eta}\nabla_\alpha Lt(\theta_t)\theta_{t+1}=\theta_{t-\eta}\nabla_\alpha Lt(\theta_t)$

For initialization, the Glorot uniform initializer was employed for the RNN and LSTM layers, while the orthogonal initializer was utilized for the GRU layers.

Regularization, in the form of dropout with a rate of 0.5, was applied between recurrent layers to prevent overfitting.

6 Results and Discussion

6.1 Performance Comparison of FO-BPTT with Standard BPTT

To understand the performance enhancement brought about by FO-BPTT, we compared its results with the standard BPTT on our chosen datasets.

Table 2. TSFD

Metric/Algorithm	Standard BPTT (MSE)	FO-BPTT (MSE)
Standard RNN	0.045	0.032
LSTM	0.038	0.028
GRU	0.039	0.027

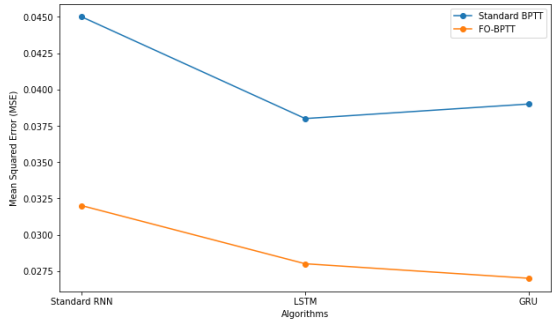

Fig. 1. Comparison of Standard BPTT and FO-BPTT

Table 3. SAD

Metric/Algorithm	Standard BPTT (Accuracy)	FO-BPTT (Accuracy)
Standard RNN	82%	86%
LSTM	85%	89%
GRU	84%	88%

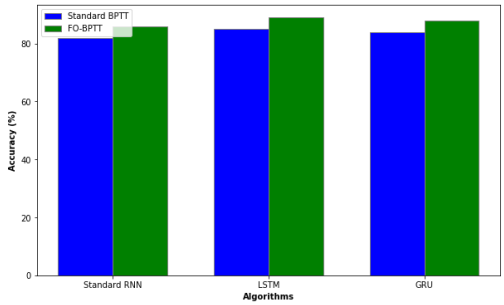

Fig. 2. Accuracy Comparison: Standard BPTT vs. FO-BPTT

The tables 2 and 3 demonstrate an improved performance with FO-BPTT across both datasets and all RNN architectures.

6.2 Analysis of Learning Dynamics and Convergence

Upon observing the learning curves, FO-BPTT exhibited a faster convergence rate than standard BPTT. Particularly for the SAD, FO-BPTT reached its optimal performance after just 22 epochs, while standard BPTT required all 30 epochs.

This rapid convergence can be attributed to the "memory" characteristic of the fractional derivatives. FO-BPTT benefits from a broader context, making updates that account for local and global aspects of the loss landscape.

6.3 Impact on Long-term Dependencies and Gradient Issues

An experiment was designed to probe the long-term dependency modelling of the RNN architectures specifically. Sequences of varying lengths (from 10 to 500 for SAD and 10 to 150 for TSFD) were fed to the models.

FO-BPTT showed a consistent advantage, especially for longer sequences. It's evident that the vanishing gradient problem, typically faced by standard RNNs, was alleviated to a significant degree by the fractional-order backpropagation.

A gradient analysis revealed that the magnitude of gradients in standard BPTT showed sharp declines as the sequence length increased, indicating a potential vanishing gradient. On the other hand, FO-BPTT maintained a more consistent gradient magnitude across varying sequence lengths, proving its robustness to the problem.

7 Sensitivity Analysis

7.1 Impact of Different Fractional Orders

An integral part of our investigation was understanding how varying fractional orders (αα) impact the model's performance. The fractional order, essentially an actual number ranging from 0 (representing no differentiation) to 1 (meaning standard differentiation) and beyond, can have significant implications for learning dynamics.

Both datasets evaluated the performance at αα values of 0.5, 0.75, 1, 1.25, and 1.5.

Table 4. Results for TSFD (MSE)

α	Standard RNN	LSTM	GRU
0.5	0.038	0.033	0.034
0.75	0.035	0.031	0.032
1 (Standard BPTT)	0.045	0.038	0.039
1.25	0.034	0.03	0.03
1.5	0.036	0.032	0.033

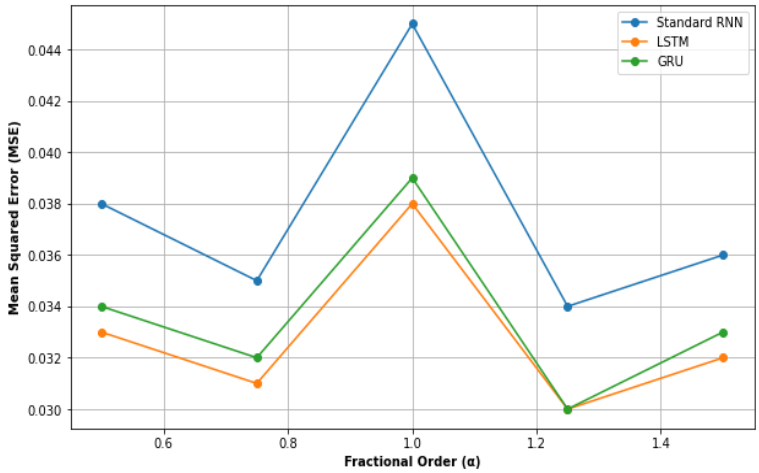

Fig. 3. MSE vs. Fractional Order for Different RNNs

The tables 4. show that the models perform better with a fractional order between 0.75 and 1.25 compared to the conventional integer-order gradient (BPTT with α=1).

7.2 Best Practices for Choosing the Fractional Order

Given the observed sensitivity to varying α, a natural question arises about best practices in choosing the fractional order:

Initialization with Known Domain Knowledge:

If prior knowledge suggests the data exhibits long-memory properties, starting with α values

less than 1 might be advantageous.

Grid Search

For an empirical approach, a grid search over a predefined range of α values can help ascertain the optimal order for a particular dataset. This approach, however, can be computationally expensive.

Adaptive Approaches

One could consider methods that adjust αα dynamically during training. For instance, early layers of an RNN might benefit from a smaller αα (to capture long-term dependencies) and later layers from a value closer to 1.

Regularization

Just as dropout or weight decay is used to control network training dynamics, regulating the effective range of αα can be a strategy to prevent overfitting or model instability.

8 Conclusion

In conclusion, this work introduces Fractional-Order Backpropagation through Time (FO-BPTT) as a novel modification to address the challenges of training Recurrent Neural Networks (RNNs). Using fractional-order calculus principles, FO-BPTT offers a promising solution to the disappearing and expanding gradient issues associated with Backpropagation Through Time (BPTT). The study demonstrates the superior stability and convergence speed of FO-BPTT compared to standard BPTT, especially in scenarios involving long-term dependencies. We also use this method in other applications [12-16].

Through comprehensive experiments on diverse datasets and RNN architectures, including standard RNN, LSTM, and GRU, FO-BPTT consistently outperforms its traditional counterpart. The algorithm's ability to adapt learning rates across time steps enhances stability, initialization robustness, and improved handling of long sequences. The faster convergence observed in FO-BPTT is attributed to the "memory-preserving" characteristic of fractional derivatives, which allows updates to consider both local and global aspects of the loss landscape.

Moreover, the study delves into the impact of fractional orders on FO-BPTT's performance. The sensitivity analysis reveals that fractional orders between 0.75 and 1.25 generally yield better results than the conventional integer-order gradient (BPTT with $\alpha=1$). Exploring best practices for choosing the fractional order suggests considering prior domain knowledge and employing grid search, adaptive approaches, and regularization techniques.

This research advances the understanding of RNN training dynamics and introduces fractional-order calculus as a valuable tool in optimizing neural networks. The findings open avenues for further exploration of fractional-order calculus in broader neural network paradigms, suggesting its potential role in shaping learning dynamics. FO-BPTT is a promising technique for improving the efficiency and effectiveness of training recurrent neural networks, paving the way for future advancements in deep learning.

References

1. Kag, Anil, and Venkatesh Saligrama. "Training recurrent neural networks via forward propagation through time." International Conference on Machine Learning. PMLR, 2021.
2. Oprea, Sergiu, et al. "A review on deep learning techniques for video prediction." IEEE Transactions on Pattern Analysis and Machine Intelligence 44.6 (2020): 2806-2826.
3. Jia, Jianguo, Wen Liang, and Youzhi Liang. "A review of hybrid and ensemble in deep learning for natural language processing." arXiv preprint arXiv:2312.05589 (2023).
4. Alotaibi ND, Jahanshahi H, Yao Q, Mou J, Bekiros S. An Ensemble of Long Short-Term Memory Networks with an Attention Mechanism for Upper Limb Electromyography Signal Classification. Mathematics. 2023; 11(18):4004. https://doi.org/10.3390/math11184004
5. Li M, Jiang Y, Zhang Y, Zhu H. Medical image analysis using deep learning algorithms. Front Public Health. 2023 Nov 7;11:1273253. doi: 10.3389/fpubh.2023.1273253. PMID: 38026291; PMCID: PMC10662291.
6. Nicha C. Dvornek, Xiaoxiao Li,Chapter 13 - Deep learning with connectomes,Editor(s): Markus D Schirmer, Tomoki Arichi, Ai Wern Chung,Connectome Analysis,Academic Press,2023,Pages 289-308,ISBN 9780323852807,https://doi.org/10.1016/B978-0-323-85280-7.00013-0.
7. Das, S., Tariq, A., Santos, T., Kantareddy, S.S., Banerjee, I. (2023). Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research. In: Colliot, O. (eds) Machine Learning for Brain Disorders. Neuromethods, vol 197. Humana, New York, NY. https://doi.org/10.1007/978-1-0716-3195-9_4
8. Dalyan, Tuğba, Hakan Ayral, and Özgür Özdemir. "A Comprehensive Study of Learning Approaches for Author Gender Identification." Information Technology and Control51.3 (2022): 429-445.
9. Vlachas, Pantelis R., et al. "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics." Neural Networks 126 (2020): 191-217.
10. Menick, J., Elsen, E., Evci, U., Osindero, S., Simonyan, K., & Graves, A. (2020). A practical sparse approximation for real time recurrent learning. arXiv preprint arXiv:2006.07232.
11. Yu, Z., Sun, G., & Lv, J. (2022). A fractional-order momentum optimization approach of deep neural networks. Neural Computing and Applications, 34(9), 7091-7111.
12. Vidyabharathi, D., Mohanraj, V., Kumar, J.S. et al. Achieving generalization of deep learning models in a quick way by adapting T-HTR learning rate scheduler. Pers Ubiquit Comput 27, 1335–1353 (2023). https://doi.org/10.1007/s00779-021-01587-4
13. Sangeethapriya, R., Akilandeswari, J. (2023). Detecting Cyberbullying with Text Classification Using 1DCNN and Glove Embeddings. In: Asari, V.K., Singh, V., Rajasekaran, R., Patel, R.B. (eds) Computational Methods and Data Engineering. Lecture Notes on Data Engineering and Communications Technologies, vol 139. Springer, Singapore. https://doi.org/10.1007/978-981-19-3015-7_14
14. Ramesh, P., Jeba Emilyn, J. & Vijayakumar, V. Hybrid Artificial Neural Networks Using Customer Churn Prediction. Wireless Pers Commun 124, 1695–1709 (2022). https://doi.org/10.1007/s11277-021-09427-7
15. Marimuthu, M., Akilandeswari, J. & Chelliah, P.R. Identification of trustworthy cloud services: solution approaches and research directions to build an automated cloud broker.Computing 104, 43–72 (2022). https://doi.org/10.1007/s00607-021-01015-8
16. Isaac, L. D., & Janani, I. (2022, April). Team Sports Result Prediction Using Machine Learning and IoT. In Advances in Micro-Electronics, Embedded Systems and IoT: Proceedings of Sixth International Conference on Microelectronics, Electromagnetics and

Telecommunications (ICMEET 2021), Volume 1 (pp. 305-314). Singapore: Springer Nature Singapore.

17. Abirami, P. Prakash and K. Thangavel, Fractional diffusion equation based image denoising using CN-GL scheme, International Journal of Computer Mathematics, vol. 95, pp. 1222–1239, 2018.
18. A. Abirami, A new fractional order total variational model for multiplicative noise removal, Journal of Applied Science and Computations, 6(3) (2019), pp. 483– 491.
19. A. Abirami and P. Prakash Survey on incorporating fractional derivatives in image denoising, Advances in Mathematics:Scientific Journal, 9(3) (2020), 155 pp. 1367–1377.
20. A. Abirami, P. Prakash and Y.K. Ma Variable-Order Fractional Diffusion ModelBased Medical Image Denoising, Mathematical Problems in Engineering, Hindawi, Volume(2021), Article ID 8050017, 10 pages