Modelling Displacement Fields of Wood in Compression Loading Using Stochastic Neural Networks

Ling. H. [1], S. Samarasingle [1, 2] and G.D. Kulasiri [2]

[1]Natural Resource Engineering Group, Lincoln University, Canterbury, New Zealand
[2]Centre for Advanced Computational Solutions (C-fACS), Lincoln University, Canterbury, New Zealand
Email:lingh3@lincoln.ac.nz

***Keywords**: Karhunen-Loève theorem, stochastic processes, white noise, deterministic neural networks, stochastic neural networks, deformation profiles, wood*

EXTENDED ABSTRACT

Most environmental and biological phenomena, such as underground water flow and pollution and properties of wood, exhibit variability which can not be successfully simulated using deterministic approaches because many components of these systems could be inherently stochastic. However, these systems can be considered as a class of stochastic processes with arbitrarily inherent nature for modelling the system behaviour in space and time. Therefore, mathematical models based on stochastic calculus along with stochastic differential equations have been established to simulate these particular cases of environmental and natural systems.

Artificial neural networks (ANNs) are another approach used to model natural and biological systems on the basis of mimicking the information processing methods in the human brain. However, very limited work has been done on investigating the capability of current neural networks to learn and approximate stochastic processes in nature although most neural networks operate in a stochastic environment. As a result, it is necessary to develop a new class of neural network named Stochastic Neural Networks for simulating stochastic processes or stochastic systems. The aim of this research is to create a suitable mathematical model for developing stochastic neural networks and implementing the proposed stochastic neural networks for simulating displacement fields of wood in compression.

A stochastic neural network is based on the canonical representation of a class of non-stationary stochastic processes by means of Brownian motion or white noise. The reason is that Brownian motion and white noise, which are two basic stochastic processes, can enable a network to numerically estimate the stochastic integrals of the canonical representation. Depending on whether a stochastic process is represented by a random function or a set of realisations (data) of a stochastic system, different approaches are used to develop stochastic neural networks. This paper just focuses on how to develop a stochastic neural network based on a set of realisations of a stochastic system because this approach is suitable for real stochastic systems as the governing stochastic function is unknown. There are three main steps in developing a stochastic neural network: (a) finding a Canonical representation of the stochastic process by means of Brownian motion or White noise; (b) creating deterministic input-output mappings from the stochastic process and then developing deterministic neural networks; (c) developing the stochastic neural network by adding Brownian motion or White noise into the developed deterministic neural networks. The most important step in the whole process is creating deterministic input-outputs mappings from the stochastic process. The purpose of this step is to develop deterministic neural networks for defining all values of weights and parameters in the stochastic neural network. Karhunen-Loève theorem plays an important role in defining deterministic input-output mappings for developing deterministic neural networks as well as stochastic neural networks.

Two successful examples are shown in this paper in order to confirm the validity of the proposed stochastic neural network. One example is to use the proposed stochastic neural network approach for simulating Sine function with random noise. This example is used to check whether the developed stochastic neural network can successfully model stochastic processes. From outputs of the proposed neural network, we will show that the developed stochastic neural network has a high accuracy in simulating stochastic processes or stochastic systems. The other example is to implement the proposed neural network in a real stochastic system: analysing and simulating localised displacement fields of wood in compression. Furthermore, this paper also provides a deep view of the network outcomes and internal workings of the network.

1 INTRODUCTION

In the past, some stochastic mathematical models have been developed to display the arbitrarily inherent nature in a stochastic system and simulate the system behaviour over time (Bear et al., 1969; Wiest et al., 1969; Kulasiri and Verwoerd, 2002). A serious problem with these models is the difficulty in solving them analytically or numerically. ANNs have a high capability in approximating input-output mappings that are complex and nonlinear to arbitrary degree of precision. The incremental learning approaches used in ANNs make it possible for them to approximate all internal parameters iteratively and they solve some problems that cannot be solved analytically. These capabilities of neural networks make them suitable to address some of the problem related to stochastic models and develop neural networks that approximate random processes. However, current ANNs only focus on approximating deterministic input-output mappings. In fact, most ANNs operate in a stochastic environment where all signal could be inherently stochastic. Thus, it is necessary to develop a neural network which has the ability to learn stochastic processes or stochastic systems. Turchetti (2004) proposed a new class of neural networks called stochastic neural networks (SNNs) as a universal approximator of stochastic processes. His book (Turchetti, 2004) entitled “Stochastic Models of Neural Networks” has presented theoretical developments and a brief demonstration on the development of stochastic neural networks for a limited number of cases, but these cases have not used them to model natural, biological and environmental systems.

In Turchetti’s book on SNNs, there are two different approaches to incorporate stochastic properties into a network: Brownian motion and white noise, which are two fundamental stochastic processes. Furthermore, Brownian motion is used to simulate continuous stochastic processes while white noise is used to simulate discrete stochastic processes. Turchetti (2004) has already given more information on how to use SNNs to simulate stochastic processes by means of Brownian motion. However, white noise is mainly used to develop SNNs for simulating a real stochastic process or stochastic system. For most real stochastic systems or stochastic processes, we can only collect a limited number of realisations from the system as the governing stochastic functions are unknown. These collected realizations are just recorded values at each discrete time or location of stochastic processes or stochastic systems.

2 OBJECTIVES

This study aims to develop and implement a stochastic neural network for representing natural real stochastic systems. The specific objectives are: to explore a method for developing a stochastic neural network based on white noise; confirm the validity of the proposed stochastic neural network by simulating a Sine function with random noise, and implement the proposed neural network for modeling noisy displacement fields of wood in compression.

3 RELATED MATHEMATICAL BACKGROUND

The Karhunen-Loève (KL) theorem is given a central role in exploring deterministic input-output mappings from stochastic processes or stochastic systems (Turchetti, 2004). Now let us consider a stochastic process $\xi(t)$ and the covariance function of the stochastic process $B(t,s)$ for different times t and s. The KL expansion is a representation of a stochastic process as a linear combination of a finite number of orthogonal functions determined by the covariance function of this stochastic process (Gihman and Skorohod, 1974). As a result, a stochastic process can be expanded as the following equation (Eq. [1]) as well as the covariance function (Eq. [2])

$$\xi(t) = \sum_{\lambda \in \Lambda} \zeta(\lambda)\varphi(t,\lambda), \qquad [1]$$

$$B(t,s) = \sum_{\lambda \in \Lambda} \lambda\varphi(t,\lambda)\overline{\varphi(s,\lambda)}, \qquad [2]$$

where $\zeta(\lambda)$ is an orthogonal sequence of random variables, the variance of $\zeta(\lambda)$ is equal to the eigenvalues (λ) of the covariance function of the stochastic process ($E\{|\zeta(\lambda)|^2\} = \lambda$) and the bar denotes the conjugate complex quantity. $\varphi(t,\lambda)$ are the eigenfunctions of the covariance function. Since $\varphi(t,\lambda)$ are deterministic functions, they can be modeled by neural networks. These networks can be linearly combined with noise $\zeta(\lambda)$ as in Eq. [1] to develop stochastic neural networks. Thus, the first step in this method is developing deterministic input-output mappings from stochastic processes.

4 METHODS

4.1 Creating Input-output Mapping

A stochastic process can also be viewed as a set or a bundle of realisations in finite domain. Furthermore, we can only collect some finite number of realisations from real stochastic processes or systems. For example, Figure 1 contains six realisations from the Sine function with random noise.

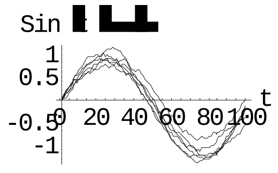


Figure 1. A set of realisations from the Sine function corrupted by noise

In this figure, all these realisations represent the behavior of the same Sine function but they also represent random fluctuations (y axis shows values of the Sine function). It is easy to see that the randomness becomes an inherent characteristic of this stochastic process. Following is a discussion on how to create deterministic input-output mapping using the KL theorem. The purpose of this is to develop deterministic neural networks.

Now we already have a dataset with a bundle of realisations of a stochastic process $\xi(t)$ and we use $\xi^k(t)$ as the k^{th} realisation. For each realisation, there are n different discrete values corresponding to each discrete time t and the value of n depends on the time interval Δt as well as the total time T of the whole realisation ($n = \frac{T}{\Delta t}$). The focus of the following step is to find the relationship between any two discrete times for all realisations and get the covariance matrix of this stochastic process. If we define that $\xi(t)$ at each discrete time t is viewed as an input variable for the covariance function, each realisation $\xi^k(t)$ at each discrete time t will be viewed as an element of each input variable. So we denote the whole group of input variables on the dataset by vector $\xi(t) = \{\xi(t_1), \xi(t_2), \cdots \xi(t_n)\}$ where $\xi(t_i)$ contains all values of realisations at the i^{th} discrete time i.e., $\xi(t_i) = \{\xi^1(t_i), \xi^2(t_i), \cdots \xi^k(t_i)\}$. In this vector representation, the mean and variance of all realisations at a particular discrete time t_i and the covariance of all realisations between any two different discrete times t_i and t_j can be efficiently calculated by using the following equations (Samarasinghe,2006):

$$\overline{\xi(t)} = \frac{1}{K}\sum_{k=1}^{K} \xi^k(t) \ , \qquad [3]$$

$$COV = \frac{1}{K-1}\sum_{k=1}^{K} (\xi^k(t) - \overline{\xi(t)})(\xi^k(t) - \overline{\xi(t)})^T \ . \qquad [4]$$

In Eq. [3], $\overline{\xi(t)} = \{\overline{\xi(t_1)}, \overline{\xi(t_2)}, \overline{\xi(t_3)}, \cdots \overline{\xi(t_n)}\}$ is a vector which contains all mean values of realisations at each discrete time t_i and K is the number of realisations in the dataset. In Eq. [4], COV is the covariance matrix which contains all variances and covariances. COV is a symmetric matrix with size $n \times n$ where n is the number of time intervals on the whole time domain. The diagonals of COV represent variances and off-diagonals represent covariances between any two different discrete times.

According to KL theorem, we can transform the COV matrix into a new matrix with new scaled variables. In this new matrix, all variables are independent of each other and all variables have their own variance. Therefore, the covariance between any two new variables is equal to zero. The COV matrix can be represented by using the Karhumen-Loève theorem as

$$COV = \sum_{j=1}^{n} \lambda_j \varphi_j(t)\overline{\varphi_j(t)} \ , \qquad [5]$$

where n represents the total number of variables in the new matrix; λ_j represents the variance of the j^{th} rescaled variable and λ_j is also called eigenvalues of the COV matrix; and $\varphi_j(t)$ is called eigenfunctions or eigenvectors of the COV matrix. The number of eigenfunctions depends on the number of discrete time intervals. We call this decomposition of the COV matrix the eigenvalue decomposition method. It is easy for us to get eigenvalues and eigenfunctions of the COV matrix using mathematical or statistical software.

A stochastic process can be represented by the function

$$\xi(t) = \sum_{j=1}^{n} \varphi_j(t)\zeta(\lambda_j) \ , \qquad [6]$$

where $\varphi_j(t)$ is eignefunctions of the COV matrix; $\zeta(\lambda_j)$ is a stochastic measure defined on a second order random field. The property of this stochastic measure $\zeta(\lambda_j)$ depends on its mean and variance. It is not possible for us to simulate this stochastic measure because of its randomness. Therefore, we first create a number of deterministic neural networks to simulate eigenfunctions of the COV matrix. The number of deterministic neural networks is decided by the number of eigenvalues which play a significant role in the KL representation of these real realisations. The stochastic measure is then represented by White noise that is embedded into the network when an output for a set of inputs is generated by the network. White noise $\zeta(\lambda_j)$ is an element of a Gaussian distribution with zero mean and variance

σ^2 which in this case is equal to the eigenvalue λ_j .Thus, the input and output mappings of deterministic neural networks are eigenfucntions from the decomposition of the COV matrix when we use data collected from a real stochastic environment.

4.2 Modelling Deterministic Neural Networks

After we define the input-output mapping of deterministic neural networks, the next step is to develop and model suitable neural networks to represent or mimic the patterns in the desired process $\varphi_j(t)$. There are three main deterministic neural networks for function approximation: Multilayer Perceptron Networks (Samarasinghe, 2006; Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe & White, 1989), Radial Basis Function Neural Networks (Park & Sandberg, 1991) and Approximate Identity Neural Networks (Conti and Turchetti, 1994). All of them have powerful capability in approximating arbitrary deterministic input-output mapping. In this case, we develop a series of Approximate Identity Neural Networks (AINNs) to learn these significant eigenfunctions $\varphi(t)$ obtained from the KL expansion of the COV matrix.

The following three factors are the focus of modelling AINNs: the number of neurons needed, the structure of network and the learning algorithm. The number of neurons depends on the input-output mappings. All these AINNs have the same structure: one input and one output with three layers. We use approximate identity functions $\omega(x) = \tanh\left(\frac{v(x-\vartheta)+\sigma}{2}\right) - \tanh\left(\frac{v(x-\vartheta)-\sigma}{2}\right)$ as our activation function. This function has the form of Gaussian distribution with special properties. The backpropagation algorithm, which is used to minimize the network's global error between the actual network outputs and their corresponding desired outputs, is used as the learning algorithm in this case. The backpropagation leaning method is based on gradient descent that updates weights through partial derivative of the network's global error with respect to the weights. When the learning step is completed, the weights of the network converge on the optimal values. As presented in Section 5, the proposed AINNs efficiently simulate all significant eigenfucntions.

4.3 Modelling Stochastic Neural Networks

After the training of deterministic neural networks is completed, the adjustment of weights of the stochastic neural network has been completed. Now we need to obtain stochastic properties of the network by adding white noise processes into deterministic neural networks as shown in Figure 2. From the KL expansion of the covariance matrix for a stochastic process, it can be seen that the whole stochastic process can be regarded as the linear combination of the product of these independent eigenfunctions and their corresponding stochastic measure $\zeta(\lambda)$ defined on the second order field. For these stochastic measures, their mean is equal to zero and their variances are equal to the corresponding eigenvalues. As a result, we can use white noise process to achieve the stochastic behavior of the corresponding networks because white noise processes have the same attributes of these stochastic measures. Figure 2 shows the structure of SNNs based on eigenfunctions and their corresponding white noise. In this figure, $\overline{\xi(t)}$ is the mean of the stochastic process at the time t . Now we have successfully developed a stochastic neural network. The next step is to choose some example stochastic processes or stochastic systems for confirming the validity of the proposed stochastic neural network.

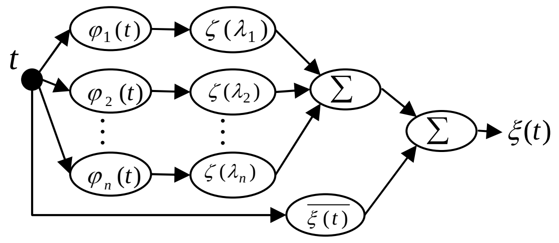


Figure 2. The structure of SNNs

5 RESULTS AND DISCUSSION

Based on the mathematical developments for a stochastic neural network, some realistic examples are used to explain in detail every step of development of deterministic neural networks as well as stochastic neural networks. The first example is about using a stochastic neural network to simulate the stochastic Sine function model. Let us use the six extracted realisations from the stochastic Sine function model which is shown in Figure 1 as our data set.

The first step in developing a stochastic neural network for this data is to calculate the covariance function of these six realisations in order to create input-output mappings for the networks from the real dataset by using KL theorem. We use Eq. [4] to calculate the covariance function or covariance matrix. The behavior of the covariance function for these six realisations is shown in Figure 3.

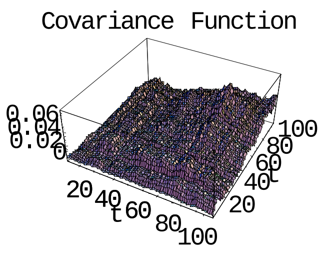


Figure 3. The covariance function of six realisations extracted from the stochastic Sine function model

According to the KL theorem, the covariance function can be decomposed into a series of eigenvalues and the corresponding eigenfunctions. Figure 4 shows only the first five eigenvalues in the KL representation of the covariance function as the left 95 eigenvalues are zero. In this figure, it can be seen that only four eigenvalues are significant and together capture the total variance in the original data. As a result, we just focus on these four significant eigenvalues as well as their corresponding eigenfunctions.

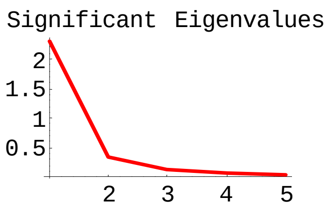


Figure 4. The significant eigenvalues in the KL representation of the covariance function

The number of AINNs depends on the number of these significant eigenvalues. As a result, four individual AINNs are used to simulate the four eigenfuctions. Figure 5 shows the values of the four eigenfucntions as well as their corresponding AINN's approximations. In the figure, the red lines represent eigenfucntion values determined by KL theory while the black lines represent the approximated outputs from the networks. Each AINN has a high accuracy of learning their input-output mappings (the range of R^2 is between 0.96-0.97).

The next step is to achieve stochastic properties of a neural network. According to KL expansion, a stochastic process can be viewed as the linear combination of the product of eignfuctions and corresponding stochastic measures. In terms of these stochastic measures, they have zero mean and their variance is equal to eigenvalues

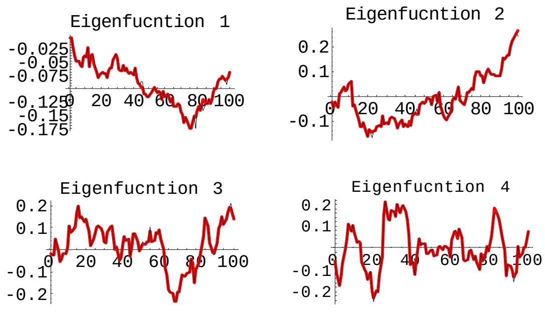


Figure 5. The four approximated eigenfucntions from AINN superimposed on eigenfunction values from KL expansion

corresponding to the eigenfucntions. These stochastic measures have the same property of White Noise so we can use White Noise to represent these stochastic measures. As a result, the proposed neural network can be considered as the linear combination of the product of eigenfunctions and their corresponding White Noise. Figure 6 represents 10 realisations obtained from the developed stochastic neural network. They are remarkably similar to the realisations from the original function shown in Figure 1. In order to confirm the validity of the proposed stochastic neural network, we need to compare the covariance function of the proposed stochastic neural network with that from realistic realisations. Figure 7 displays the predicted covariance function for 200 realisations extracted from the proposed network. It can be seen that there is no difference between the predicted covariance function and the actual covariance function (Figure 3) from the real realisations of the stochastic Sine function model (R^2 is 0.9).

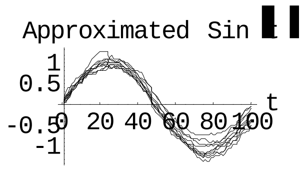


Figure 6. Ten realisations obtained from the stochastic neural network

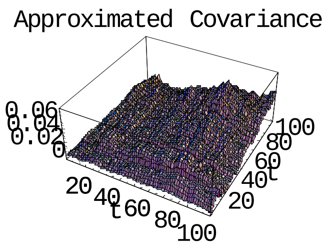


Figure 7. The covariance function of the approximated Sine function as obtained from 200 realisations of the stochastic neural network

Based on the first example, it can be seen that the proposed stochastic neural network has a high accuracy in simulating stochastic processes. Now we apply this idea to simulate and analyse localised displacement fields of wood in compression.

Wood like most biological and cellular materials has a very complicated internal structure that leads to variability in properties. In a recent study, image processing methods have been used to obtain displacement fields in a very small area on the surface of a loaded wood specimen in compression parallel-to-grain (Samarasinghe & Kulasiri, 2000). This means that we only know the behaviour of this small area from each image. However, the stochastic memory of the stochastic neural network can help us to recall more realisations of the behaviour of this wood specimen based on the existing images in order to deeply analyse the structural influence on mechanical behaviour of wood. The wood specimen ($41 \times 44 \times 136mm$) in this project was cut from kiln-dried structural grade New Zealand radiate pine (pinus radiata) boards obtained from a local timber yard in Christchurch (Samarasinghe & Kulasiri, 2000). These were tested on a computer controlled material testing facility that measured the applied load while a camera took images of the specimen at various load levels. By comparing the displaced images to the initial undisplaced images for an area around $20 \times 20mm$ using Digital Image Correlation (DIC), displacements of a large number of points were determined. The data determined this way contains two different displacements: vertical and horizontal displacements. Now we use the proposed stochastic neural network to analyse the structural influence on mechanical behaviour displacement of wood.

When a 2kN compression load is applied parallel-to-grain, vertical displacement (u) measures the amount of contraction in the same direction of loading while horizontal displacement (v) measures the amount of expansion in the perpendicular direction to loading. Figure 8 shows both vertical and horizontal displacement obtained from images using the DIC method. Here, one vertical displacement realisation corresponds to points along one column and one horizontal displacement realisation to one row of the image. Furthermore, x axis shows the total number of uniform interval ($0.67mm$) on the analytical area. In this figure, it can be seen that the influence of structure in loading parallel-to-grain on horizontal displacement is more complex and fluctuating than on vertical displacement.

According to KL expansion, we decompose the covariance function of both vertical and horizontal displacements in order to create deterministic input-output mappings. In terms of vertical displacement, we only got three significant eigenvalues from the distribution of all eigenvalues (30) from the KL expansion of the covariance function. However, we found that eight significant eigenvalues capture the variance in the original data for the horizontal displacements (the total number of eigenvalues is 30). The number of significant eigenvalues also display that there is a lot of noise or complexity in the horizontal displacement. Thus, we need to develop three AINNs for the vertical displacement and eight AINNs for the horizontal displacement to approximate their corresponding eigenfunctions. When the learning step is completed, most components of the proposed stochastic neural network have already been determined and we need to add the relevant white noise into their corresponding AINNs in order to achieve stochastic properties of the network as depicted in Figure 2.

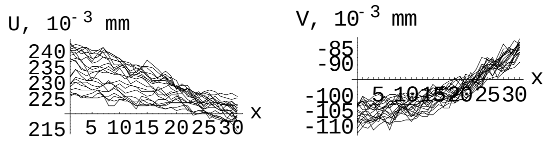
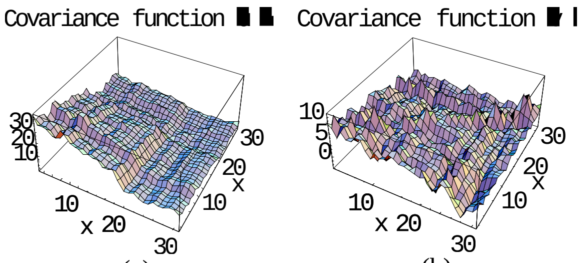


Figure 8. Vertical and horizontal displacement profiles for a wood specimen loaded 2kN in compression

Figure 9 displays the covariance function of vertical displacement as well as horizontal displacement.

Figure 10 displays some realisations obtained from the developed stochastic neural network for the vertical displacement as well as the horizontal displacement. They are remarkably similar to the actual realisations shown in figure 8.



(a) (b)

Figure 9. (a) Covariance function of the vertical displacement (u); (b) Covariance function of the horizontal displacement (v)

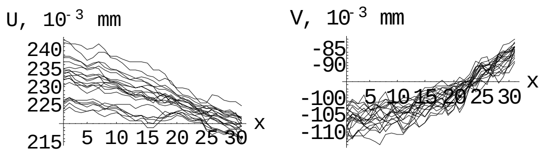


Figure 10. The approximated vertical and horizontal displacements from the stochastic neural networks

Figure 11 shows the approximated covariance functions from the stochastic neural networks. Compared to Figure 9, it can be seen that the stochastic neural networks have a high accuracy in approximating the experimental dataset (the range of R^2 is between 0.91-0.92).

Furthermore, through analysis of internal workings of SNNs, we get the following properties of SNNs:

a. The output of SNNs is a linear combination of the mean value at each discrete position and the summation of the product of each eigenfunction and their corresponding white noise at the same discrete time;
b. Values of white noise are constant in the same realisation. The difference between each different position depends on their corresponding mean values as well as eigenfunctions;
c. Values of white noise are stochastic between any two different realisations. This is the reason why each realisation is different.

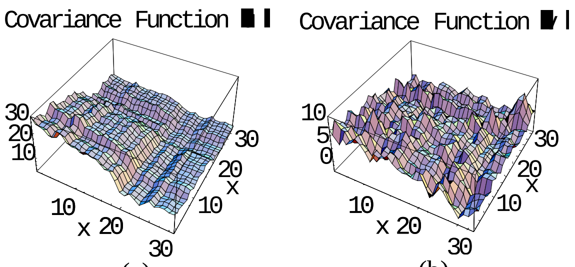


Figure 11. (a) The approximated covariance function of the vertical displacement (u); (b) The approximated covariance function of the horizontal displacement (v)

6 CONCLUSION

The purpose of this research is to present the mathematical developments and implementations of stochastic neural networks based on a realistic dataset from a stochastic system. Two examples in this paper provided enough evidence to confirm the validity of the theoretical results as well as give us more confidence in developing a stochastic neural network to simulate some real stochastic systems and stochastic processes. The proposed stochastic neural network can be viewed as a suitable tool to capture the complexity of system behaviour and simulate natural and biological phenomena.

7 REFERENCES

Bear, J. (1969). Hydrodynamic dispersion. In: Flow through porous media. *Academic Press,* New York.

Conti, M., & Turchetti, C. (1994). Approximation of dynamical systems by continuous-time recurrent approximate identity neural network. *Neural, Parallel & Sci. Computations*, 2, 299-322.

Cybenko, G. (1989). Approximation by superposition of sigmoidal function. Math. *Control, Systems, Signals*, 2, 303–314.

Funahashi, K. (1989). On the approximate realisation of continuous mappings by neural networks. *Neural Net*., 2, 183–192.

Gilman, I. I., & Skorohod, A.V. (1974). The theory of stochastic processes. Berlin, Germany: Springer-Verlag.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Net.* 2, 395–403.

Kulasiri, D. & Verwoerd, W. (2002). Stochastic dynamics modeling solute transport in porous media. *North Holland Applied Mathematics and Mechanics Series*. Vol. 44. Elsevier Science, Amsterdam. 250 pages, 2002.

Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function network. *Neural Computation*,3, 246-257.

Samarasinghe, S. (2006). Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition. *CRC Prem,* USA, 570pp.

Samarasinghe, S. & Kulasiri, G.D. (2000), Displacement fields of wood in tension based on image processing: Part 1. Tension parallel- and perpendicular- to- grain and comparison with isotropic behaviour. *Silva Fennica* 34(3):251-259.

Turchetti, C. (2004). Stochastic models of neural networks. *IOS Press,* Amsterdam.

Wiest, R. J. M. (1969). Fundamental principles in groundwater flow. In: Flow through porous media. *Academic Press,* New York.