ARTIFICIAL NEURAL NETWORKS FOR OPTIMIZATION PROCEDURES OF LAZY WAVE FLEXIBLE RISER CONFIGURATION

Ana Paula Benete
Bruno da Fonseca Monteiro
Carl Horst Albrecht
Breno Pinheiro Jacob
Mauro Henrique Alves de Lima Junior
anaoliveira@poli.ufrj.br
bruno.monteiro@poli.ufrj.br
carl@poli.ufrj.br
breno@lamcso.coppe.ufrj.br
mhljr@poli.ufrj.br
Laboratório de Métodos Computacionais e Sistemas Offshore – LAMCSO/COPPE/UFRJ
Ilha do Fundão, Avenida Pedro Calmon, CEP 21941-596, RJ, Rio de Janeiro, Brasil.

Abstract. Risers are among the most expensive and complex components of an offshore production system, because they suffer the action of a series of dynamic and environmental loads when connecting the seabed to the floating unit. An optimal design of such structures can bring more security and cost savings, which can be achieved by optimization techniques. In this context, this work presents the development and implementation of Artificial Neural Network (ANN) in optimization procedures of Lazy Wave flexible riser configuration as an alternative to Finite Element simulations, which has a high computational cost. For the network selection, a parametric analysis has been done considering the mean square error, accuracy of two different types of training algorithms and different amounts of neurons layers. A case study is presented comparing the results of the ANN optimization process with a simulation by the Finite Element Method (FEM). The results indicate a significant reduction of the computational cost of all optimization process was achieved using ANN, which was able to predict with accuracy the magnitudes involved in this type of problem.

Keywords: Artificial Neural Network, Riser, Optimization.

1 Introduction

Oil and gas production have migrated to deep and ultra-deep waters over the years, reaching water depths of more than 2000 meters, as in the Brazilian Pre-Salt fields. This type of production requires a series of complex equipment and new technologies to deal with the challenges of Floating Units, which are mostly FPSO's (Floating Production Storage and Offloading) and Semi-submersible platforms which are connected to the seabed by mooring lines (responsible for supporting the efforts and movements of the platform, usually caused by environmental loads) and a number of flexible lines designed for transferring oil, gases and other essential chemical products, risers. These lines need to be resistant and watertight, without compromising its flexibility. Therefore, risers are designed in independent layers of different materials. Each layer has a different purpose, such as reducing friction wear and resistance to high temperatures, stresses and corrosion. This number of requirements makes the riser one of the most expensive and complex equipment of an offshore system.

Because of great water depths in recent offshore production systems, the structural behavior, integrity and configuration of the riser has become important issues. In this context they are subject to heavier loads, due to factors such as a bigger water depth, floating unit movements, type and number of lines and location. With the need to balance cost and safety, optimization methods become useful for setting the parameters that define this configuration.

Previous works have studied the application of optimization techniques for the design of offshore oil production risers. In [1], the authors used Particle Swarm Optimization (PSO) method, tailoring the algorithm to the problem. In [2], an assessment of enhanced variants of three different groups of bio-inspired methodologies genetic algorithms, particle swarm optimization, and artificial immune system, was presented. A study to automate the design of compliant vertical access risers by applying Elitist non-dominated sorting genetic algorithm together with a design of experiments (DOE) was made by [3]. The outputs of these optimizations are usually obtained by Finite Element Method with high computational cost.

This motivates the studies of Artificial Neural Networks for optimization procedures in order to reduce this cost without losing a significant amount of precision on these parameters. In addition, Particle Swarm Optimization was chosen as optimization method to explore the search space. Previous works [4] have demonstrated that the application of ANN and other meta-models in mooring lines and risers presenting high levels of accuracy and low computational time, giving room for our research and application.

2 Problem Modeling

2.1 Lazy wave Riser Configuration

Since we are working in a context of ultra-deep waters, there is a need for more structured risers (comparing with smaller depths), causing a great submerged weight. Lazy wave configuration act to decrease the tensions at the riser bottom and top, using the buoys to generate buoyancy in the intermediate section of the line. Figure 1 presents a schematic model with some of the parameters that are important for a riser evaluation: Length of top riser segment (L1), Length of segments with distributed floaters (L2), Length of lower segment of the riser (L3), Buoy diameter (DFLUT), Buoy length (DLFLUT) and Spacing between buoys (ESPFLUT).

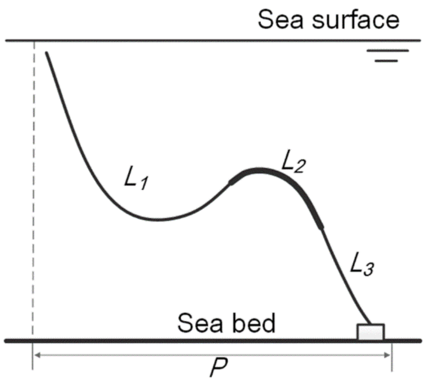

Figure 1. Lazy wave riser configuration and its measures

Any candidate solution must attend other constraints, parameters and limits previously defined [4] related to the riser structural response. These parameters are: value of maximum top angle variation (angle between the top riser segment axis and a vertical direction at the riser connection); tension limit on the riser top and bottom; maximum von Mises stress on riser sections.

By respecting these parameters and applying them to generate a dataset used to train the Artificial Neural Network, the optimizer guarantees the structural integrity of the riser.

2.2 Fitness Function

In order to relate the 6 riser variables and compare each evaluated solution, an unconstrained fitness function is calculated as follows:

$$F = \frac{C_{min}}{\left(\sum_{i=1}^{n} CI_i \cdot L_i\right)+(CI_{buoy} \cdot V_{buoy})} \tag{1}$$

Where, C_{min} is the lowest possible cost, used to normalize F in the interval [0,1]; $i = 1 \dots n$ represents the index of the riser segment; CI_i is a cost associated to each segment; L_i is the segment length (L1, L2 or L3); V_{buoy} is the buoy volume (calculated with DFLUT and DLFLUT) and CI_{buoy} is a cost associated to V_{buoy} [4].

It is important to note that this optimization problem consists of minimizing the fitness function if FEM procedure is used to compute the above parameters. In this work, a dataset is used to train a metamodel to return the fitness value making possible the use of any optimization algorithm even considering maximization characteristics.

2.3 Artificial Neural Network

Artificial Neural Networks are computational models based on mathematical regression capable to predict outputs, given a certain input. It relates inputs to outputs by changing parameters and function weights of each of its neurons. ANN neurons are key parts of the algorithm, formed by mathematical functions that receive one or more inputs and return an output, which may be the input of other neurons. These are organized in layers, where each layer has the same connections on its neurons and the same type of output. Figure 2 represents an artificial model of a neuron [5]. On the shaping of a neural network, the number of layers and neurons on these layers affects the quality and precision of the output.

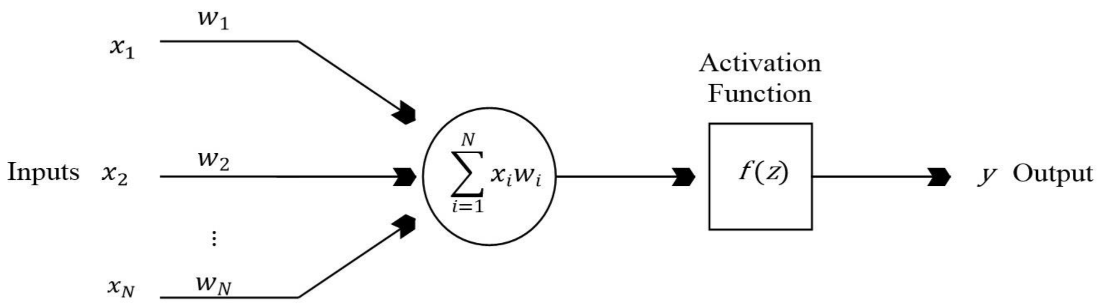

Figure 2. McCulloch-Pitts neuron

As shown in the Figure 2, the neuron calculates a linear combination of the inputs with their respective weights, forming the new scalar z, and uses z as the new input for the activation function $f(z)$:

$$y = f(z) = \frac{1}{1+e^{-x}} \tag{2}$$

When the output of a neuron can be received by other neurons without restrictions, recurrent networks are formed. Moreover, when only the next layer neuron on the network can receive the output, a feed-forward neural network, is formed as illustrated in Figure 3 [6].

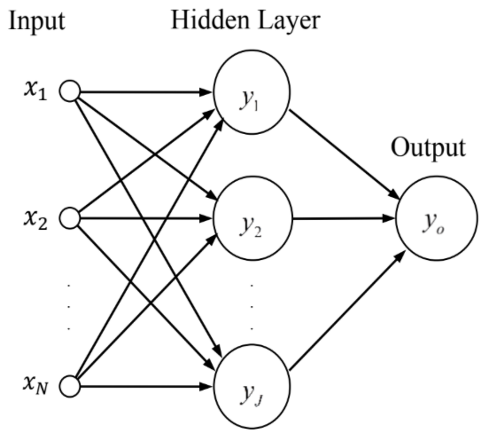

Figure 3. General structure of a two layers feed-forward artificial neural network

Many types of activation functions can be applied, depending on the purpose of each network. In this work, as implemented in MATLAB©, the sigmoid function, Eq. (2), is used to take the network outputs to values from 0 to 1 in the hidden layer of the network. However, in the output layer, responsible for returning the network output vector, a linear transfer function is used for the new function approximation. Therefore, this optimization works with a one hidden layer feed-forward network with sigmoid hidden neurons and linear output neurons.

The ability to learn correctly comes from a training process using a significant amount of data: examples of inputs. This learning can occur in two ways: Supervised and Unsupervised. Unsupervised learning requires only the inputs of a problem to train the neural network and it is usually used for cases where the network must learn to identify patterns in the data, often not identified by users, such as clustering problems [7]. Supervised learning requires both the desired inputs and outputs of the problem [8]. In this case, training of the network has the objective of mapping and finding relations between the inputs and outputs of the problem so that, when presenting a new input, the network can produce the desired output type.

Training an ANN on supervised learning is essentially the process of changing the weights related to each input that enters each neuron. Back-propagation method, the most common for multilayer neural networks, consists on repeating two steps: analyzing the errors of the network outputs and change the weights, from the end to the beginning of the network. These steps are repeated until the errors are within an acceptable range, previously defined, according to the purpose of the network [4].

MATLAB© provides three different algorithms for fitting optimization: Levenberg-Marquardt, Bayesian Regularization, Scaled Conjugate Gradient, which are analyzed and compared in the Case Study (3) (together with the number of neurons in the inner layer of the network) in relation to the Mean Squared Error between the network outputs and targets.

2.4 Particle Swarm Optimization

PSO, that was developed by Kennedy and Eberhardt in 1995 [9], consists in a computational method bio-inspired by the social behavior of some animals, for example, a flock of birds, where the information acquired by an individual about the search space influences the analysis of the entire population.

This method maintains a swarm of candidate solutions, called particles, starting with an initial population randomly generated. Throughout the evolution process, each particle remembers the best position it has found, and the best position found by all particles. This algorithm has a few parameters to be adjusted, population size, inertia coefficient, individual and social terms. Trelea [10] parameters that have been shown to present good results for general applications were used as constants of the problem.

Figure 4 presents the steps of the basic PSO. Equations and more details about PSO method can be found in [9].

```
Randomly initialize population
For each particle, calculate “fitness” from objective function
Set personal best = fitness
Compute global best
Repeat
    Compute new velocities of each particle
    Update positions
    Calculate fitness
    Update personal best and global best
Until stopping criterion reached (usually sufficiently good fitness or fitness evaluations)
```

Figure 4. Pseudo-code of standard PSO

2.5 Latin Hypercube Sampling

The key for a successful Artificial Neural Network is the dataset chosen for the training process. Dataset should have a considerable level of randomness (within the limits of each parameter) to cover as many cases as possible. Latin hypercube sampling method [11], usually used for multidimensional problems, can guarantee such randomness.

This method started with two dimensions (the Latin square), organizing the sample space formed by two variables in a square / "chessboard" $n_x n$ and selecting points from this square so that two points are not in the same row or column , Figure 5.

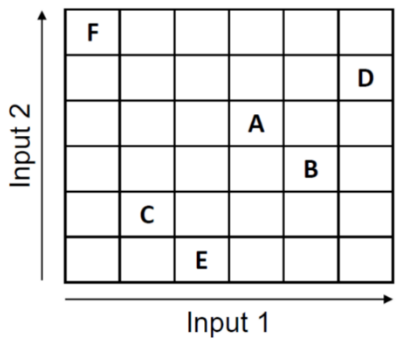

Figure 5. Latin square general example

Latin hypercube is the generalization of this method to a greater number of variables. In our particular problem, each one of the 6 variables (see section 2.1) are divided, within their limits, as can be seen in Table 3 on the next section, in n individuals (given a sample of size n) and a point is chosen at each of these intervals. This process generates the final sample of n vectors of 6 dimensions.

3 Case Study

3.1 Model Description

The riser example that we use to model the problem and to optimize the design is installed in a Lazy-Wave configuration between a floating unit and the seabed at 2,400 meters. Tables 1 and 2 show the geometrical and physical properties of this riser.

Table 1. Riser geometrical properties

Property	Value
Internal Diameter	0.3048m
External Diameter	0.4064m
Floater external diameter	0.5588m
Floater weight	2.0 kN/m
Floater buoyancy	8 kN/m
Thickness	0.0508m

Table 2. Riser physical properties

Property	Value
Elastic Modulus	203,851.8 MPa
Yield stress	413.0 MPa
Allowable stress	330.4 MPa
Density	7,800 kg/m^3
Specific weight	77 kN/m^3

Table 3 presents the free variables of the optimization process.

Table 3. Limits of the free variables of the problem

Free Variables	Max [m]	Min [m]
L1	1,200	2,000
L2	500	1,500
L3	2,000	3,000
DFLUT	1.5	3.5
DLFLUT	1.5	3.5
ESPFLUT	1.0	3.5

It is worth remembering that in the calculation of the Fitness Function (Eq. **Error! Bookmark not defined.**, described in section 2.2), the F value of segments with Floats (L2) are worth twice as many standard segments (L1 and L3), as already mentioned in previous studies that used the same riser model [4].

The data used to train the Artificial Neural Network was generated from Finite Element analysis provided by the SITUA-PROSIM, a non-commercial software used for the analysis of offshore systems [12]. It returns as output the result of the Fitness Function. By using the supervised learning method, the network is trained with both the inputs and outputs from FEM analyzes.

3.2 Parametric studies of the ANN Training

This section presents the parametric studies performed to choose the main parameters that directly influence the training time and Mean Square Error (MSE) on ANN. Such parameters are training algorithm, number of neurons and dataset size. In order to find the best network, some variations of these characteristics using MATLAB© ANN toolbox was compared.

First comparisons were made between training algorithms Levenberg Marquardt and Bayesian Regularization well as well the number of neurons. For each configuration, 10 independent networks were trained, the average of their MSEs and time (in minutes) was presented in Table 4. These rounds were performed with a large sample number (89,058) to avoid errors and deviations possibly caused by the dataset. The uneven number of samples is caused by corrections and filtering that were made in the file released by SITUA, excluding some nonconverging individuals from the sample. This study was done using a notebook with Intel Pentium processor with 2.16 GHz and 4GB of memory. For these two reasons (large sample number and PC configuration), the training time of the networks in this study is higher than the following comparisons.

In addition, all networks of this work were trained in the following proportion: 80% of data for training, 15% for validation and 5% for test. These proportions have been chosen to ensure the learning of as many examples as possible.

Table 4: Parametric studies of the ANN training algorithm and number of neurons

Name	Algorithm	Neurons	Average MSE	Average Time [min]
LM10	Levenberg Marquardt	10	2.3766E-07	8
LM30	Levenberg Marquardt	30	1.9216E-07	30
LM50	Levenberg Marquardt	50	1.8154E-07	79
B10	Bayesian regularization	10	2.5069E-07	7
B30	Bayesian regularization	30	1.9711E-07	24
B50	Bayesian regularization	50	1.9485E-07	52

Although LM50 configuration has lower MSE, the LM30 configuration was chosen, because it presents the MSE in the same order of magnitude as LM50 and it has a significantly shorter training time. Therefore, the configuration selected was: A network trained by the Levenberg Marquardt algorithm with 30 neurons in its layer.

Despite having already trained ANN with a low error in the previous study, it was decided to do one more study to identify the smallest dataset size where the ANN could be satisfactorily trained to achieve a shorter training time. From this parametric study to the following results, the analysis and training were done on a PC with Intel i7 processor with 3.40 GHz and 8 GB of memory.

Also, for these comparisons, 10 independent networks were trained for each dataset size, already using the previously defined configurations. Table 5 shows the average MSE of these 10 runs for each dataset size, average training time (in seconds) and the MSE for the best run.

Table 5: Parametric studies of the dataset size

Dataset size	Average MSE	Average Time [s]	Best Run MSE
200	4.12E-05	0	1.64E-05
500	5.60E-05	0.4	2.68E-07
1000	3.14E-07	1.8	2.36E-07
3000	2.45E-07	5.6	2.07E-07
5000	2.28E-07	10.7	2.07E-07
7000	2.19E-07	18.0	1.90E-07
10000	2.20E-07	22.0	2.06E-07

Analyzing the average MSE and comparing with those found in the parametric study of the algorithms, it is noticed that when increasing the dataset, above 3,000 samples, MSE tends to converge to values close to 2E-07, even though still varying around the core value, caused by random neural network responses in a few individuals. The same convergence process still happens with large datasets such as more than 80,000 samples. This shows that in this range of training samples, all networks will have outputs with similar accuracies. Therefore, the best network of the dataset size of 7,000 samples was used. Table 6 shows the configuration of the artificial neural network used in the optimization and the data division applied in training process:

Table 6: Artificial Neural Network final configuration

Algorithm	Levenberg Marquardt
Neurons	30
Dataset size	7,000
Training set (80%)	5,600
Validation set (15%)	1,050
Test set (5%)	350

3.3 Results: Optimization with the trained Artificial Neural Network.

In the optimization process, fitness value of each candidate solution is computed by the previously trained ANN. PSO algorithm was performed with a population size of 30 individuals along 100 generations. PSO coefficients were chosen according to Trelea [10], type 1, $\omega = 0.6$ and $C_1 = C_2 = 1.7$.

Figure 6 shows the optimization process evolution using ANN to return the fitness value. One can notice that the objective of this process consists in minimize the fitness value, as explained in section 2.2.

We can see that the algorithm converged around 2,500 function evaluations, which indicates that the number of evaluations (3000) was more than satisfactory.

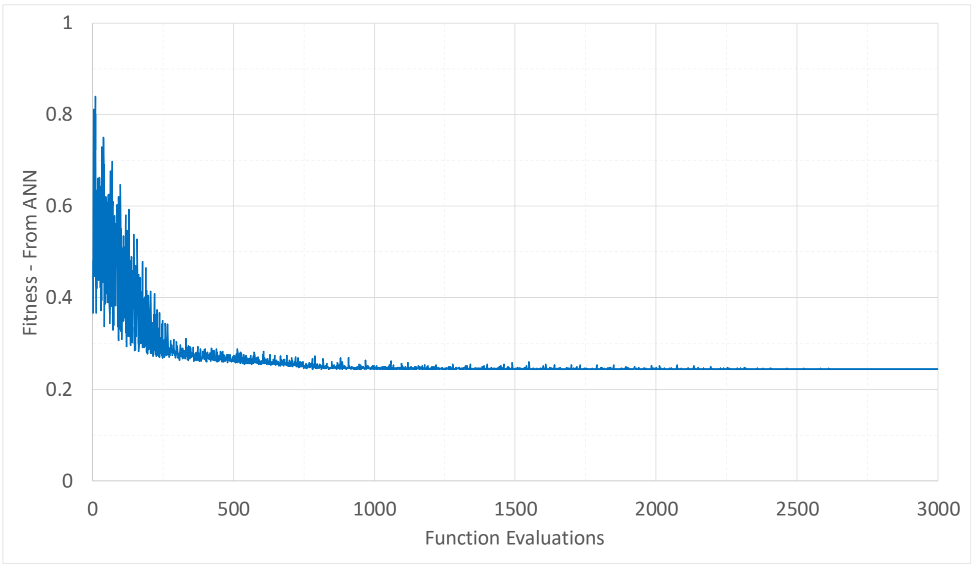

Figure 6: Evolution of the PSO algorithm

3.4 Results: Assessment of ANN

In order to assess the accuracy of the fitness calculation by ANN instead of the analysis by FEM, all individuals generated during the optimization process (by the PSO algorithm) were saved and subsequently calculated via FEM, so that we could obtain the correlation between the data. Figure 7 shows this correlation between ANN and FEM.

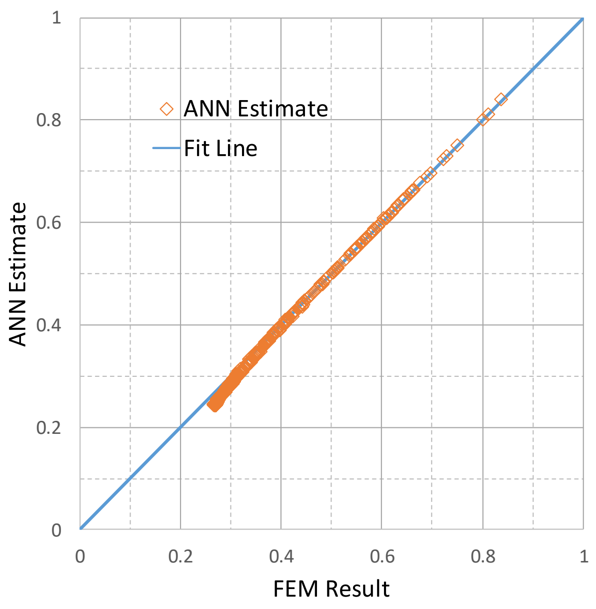

Figure 7: FEM Output and ANN Estimate

The maximum error between ANN and FEM data was 2.6E-02. This shows good accuracy of the trained ANN and that the replacement of the FEM analysis by the ANN was successful.

It can be observed that the region with lower fitness values is the one with greater error. This is due to the fitness function at this location is probably influenced by the constraints of the problem.

Another important comparison is about the total time in optimization process. Using ANN, the optimizer spent only 6.7s to complete the whole process while using FEM the optimizer spent 6240s or 1h 44min to perform the complete process using static analyses. This computational cost reduction may favor several other studies such as comparing other optimization algorithms, for instance.

4 Conclusion

This study aimed to apply Artificial Neural Network in the optimization process of configuration of Lazy-Wave risers installed in deep waters, being able to replace analyzes made by Finite Elements, reducing the computational costs. Finite Element analysis data were used to train the network and verify its outputs. These data are composed of the parameters of the riser and the result of the calculation of the fitness function. Besides showing the use of the networks in the optimization, parametric studies were done to define the characteristics of this network and the dataset used to improve its performance and its training time. The neural network used in the optimization process had as training mean error 2.19E-07, taking 18 seconds to be trained.

From the results, in the parametric studies, Levenberg Marquardt training algorithm with 30 neurons in its layer presented better performance. In addition, it can be noted that the replacement of FEM analysis by ANN throughout the optimization process was satisfactorily achieved.

Regarding computational cost, the optimization process with ANN spent only 6.7s while using FEM to compute the objective function spent 6240s considering static analyses.

In future works, the parametric studies elaborated in this research can be used to apply this type of ANN within optimization programs training it during the optimization processes, with new datasets in each iteration.

5 References

[1] de Pina AA, Albrecht CH, de Lima BSLP, Jacob BP. 2011. Tailoring the Particle Swarm Optimization Algorithm for the Design of Offshore Oil Production Risers. Optimization and Engineering, 12:215-235. https://doi.org/10.1007/s11081-009-9103-5.

[2] Vieira IN, de Lima BSLP, Jacob BP. 2012. Bio-Inspired Algorithms for the Optimization of Offshore Oil Production Systems. International Journal for Numerical Methods in Engineering, 91:1023–1044. https://doi.org/10.1002/nme.4301.

[3] Martins MAL, Lages EN, Silveira ESS. 2013. Compliant vertical access riser assessment: DOE analysis and dynamic response optimization. Applied Ocean Research, 41:28-40. https://doi.org/10.1016/j.apor.2013.02.002.

[4] Monteiro, B. F., Baioco, J. S., Delgado, E. R., Albrecht, C. H., de Lima, B. S. L. P., & Jacob, B. P. (2018). *Studies on Meta-Modeling for Lazy-Wave Steel Catenary Risers*. Proceedings of the 37st International Conference on Offshore Mechanics and Arctic Engineering, paper OMAE 2018-78181, Madrid, Spain.

[5] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activities. Bull Math Biophys 1943;5:115–33.

[6] Delgado, E., De Pina, A., Monteiro, B., Albrecht, C., & Jacob, B. (2016). Artificial neural networks meta-models for prediction of vessel offsets and mooring lines tensions of floating production systems. Proceedings of the XXXVI Iberian Latin American Congress on Computational Methods in Engineering.

[7] Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

[8]A Shapiro. The inner workings of neural networks and genetic algorithms ARCH, 1 (1999), pp. 415-426.

[9] Kennedy J, Eberhardt R. Particle swarm optimization. In: Proc IEEE conference on neural networks, 1995, pp 1942–1948.

[10] Trelea IC. 2003. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. Information Processing Letters, 85(6):317–325. https://doi.org/10.1016/S0020-0190(02)00447-7.

[11] Kleijnen JPC. An overview of the design and analysis of simulation experiments for sensitivity analysis. European Journal of Operational Research. 2005;164(2):287–300. https://doi.org/10.1016/j.ejor.2004.02.005

[12] Jacob, B.P., Bahiense, R.A., Correa, F.N., Jacovazzo, B.M., 2012. Parallel implementations of coupled formulations for the analysis of floating production systems, part I:coupling formulations, Ocean Eng. 55 206–218. http://dx.doi.org/10.1016/j.oceaneng.2012.06.019