COUPLING MOHID WITH OPTIMIZATION ALGORITHMS: PERSPECTIVES ON THE DEVELOPMENT OF AUTOMATIC CALIBRATION TOOLS

E. B. Lima • P. G. Watts Rodrigues • A. J. Silva Neto • J. Lugon Jr. • M. Irízar Mesa • O. Llanes Santiago

CHAPTER SYNOPSIS

Background
The mathematical formulation adopted in MOHID includes parameters that need to be defined or estimated This chapter presents the coupling of stochastic optimization methods and neural networks with the MOHID platform, in order to solve an inverse problem, here formulated as the better estimative of bed roughness height and turbulent diffusion coefficient, parameters that can be present in the mathematical formulation of hydrodynamic and transport models. The direct problem focused on here was the simulation of the hydrodynamics and salinity transport in an estuarine environment, the Macaé estuary, Brazil. The results presented here are part of ongoing investigation line, which has been conducted by researchers of the IPRJ/UERJ since 2008.

Results
All optimization methods tested here showed satisfactory performances, leading to error estimates lower than measurement errors. Whenever these errors were added to the experimental data, a longer simulation period was required to estimate the desired parameters. Regarding the neural network, the results highlighted the necessity to define a better training strategy.

Conclusions
The coupling of MOHID with optimization algorithms can considerably give agility to calibration procedures. Future researches should focus on water quality simulation, which are strongly dependent on a quite large number of parameters.

1 INTRODUCTION

The diagnosis of a given water body is based on the concentration analyses of a number of parameters. The water resources management can also include the simulation of scenarios, which could emerge as a consequence of man's interference in the environment. In this particular regard, mathematical models are useful tools, reducing time, costs and risks in the environmental impact evaluation. The mathematical formulation of these models includes a number of parameters that can assume a range of reasonable values, and a better estimative is crucial to the simulation accuracy. This estimative can be done through trial and error, what, besides being highly time consuming, can lead to values that allow to model calibration for a single and particular scenario only. Further, such strategy can lead to multiple and acceptable solutions. Alternatively, this calibration can be performed by algorithms, as those based on inverse problems techniques. These techniques have been used in different science fields. For water resources modelling purposes, however, not many applications can be found in the literature.

For example, [1] used a modified Newton method to estimate the nonpoint sources of fecal coliform, in order to establish allowable load for the Wye River, USA. Other studies have made use of the Gradient Method, Conjugate Direction Method and Variational Method to estimate parameters usually present in groundwater modelling [2-3]. On the other hand, instead of parameters, the salinity boundary conditions was estimated by [4] for a three-dimensional tidal hydrodynamic and salinity transport model. In a similar way, [5] proposed a novel algorithm for the estimation of open boundary conditions in rivers systems where tidal forcing is present.

The solution of an inverse problem can be made through optimization algorithms, which attempt to minimize (or maximize) the value of a functional through the modification of its variables. Methods of optimization can be of two types: deterministics, which move with the setting in the sense of optimal, making use of particularities of the functional; and stochastic type, choosing random configuration with the goal of sweeping, stochastically, the entire search space, in the search for the optimum. These methods are usually compared regarding the computational time required to achieve the desired solution. From this point of view, the stochastic methods show advantages if compared with the deterministic ones. Another important feature of optimization methods is that they can be applied to any objective function, regardless the initial value assumed for it, beside the capability of avoid local minimums.

This chapter presents the coupling of optimization methods and neural networks with the MOHID platform, in order to solve an inverse problem, here formulated as the better estimative of bed roughness height and turbulent diffusion coefficient, parameters that can be present in the mathematical formulation of hydrodynamic and transport models. The direct problem focused on here was the simulation of the hydrodynamics and salinity transport in an estuarine environment, the Macaé estuary, Brazil.

1.1 MOHID simulator and the Direct Problem

In this work, the MOHID platform was used to simulate the hydrodynamics and salt transports of Macaé estuary, located at the Brazilian southeast coast (Figure 1). The Macaé basin, which has a population of around 141,000 people, has been submitted to some severe environmental impacts, especially in its estuarine zone. This has motivated the authorities to implement some actions, which includes the computational modelling of estuarine waters. The modelled domain included an extension of approximately 20 km, from the head to the outer region of the estuary, at the coast. It was adopted a spatial discretization of 40 m, based on quadratic cells. The bathymetry data of the coastal region was taken from the nautical chart 1507, edited by the Brazilian Navy in 1974, while the upper region bathymetry was obtained from [6]. Figure 2 shows the discretized domain and the adopted bathymetry implemented in MOHID.

Two boundary conditions were prescribed for the hydrodynamic model: (i) in the riverine boundary it was set a river discharge of 7.8 $\mathrm{m^3 s^{-1}}$, typical of the dry season for the Macaé river close to estuarine region [6]; (ii) in the marine boundary it was simulated an astronomic tide with 17 components. A salinity of 0.037 PSU was set for the riverine boundary, and 36 PSU for the sea boundary. As the initial condition, a salinity of 20 PSU was set for the whole domain.

1.2 Formulation of the Inverse Problem

In most of the scientific disciplines, and particularly in engineering, there are problems characterized by differential equations with associated initial and boundary conditions. When

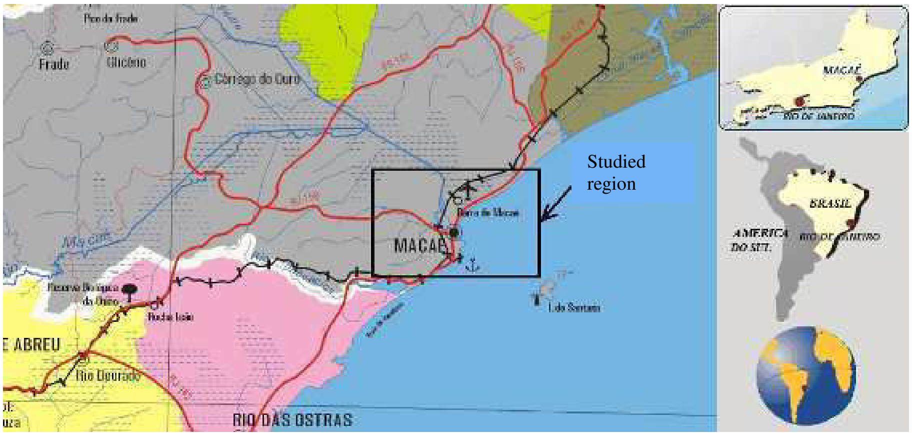

Figure 1. *Estuary location and detail of the simulated region.*

these problems are solved in a direct way, the result is generally a functional relationship or a system of equations, which can be used to calculate values of the dependent variable for given values of the independent variable. The inverse solution of systems of partial differential equations constitutes a complex problem, for which there are no universally accepted methods. Given an applicable direct solution to a system of partial differential equations, it is possible to propose an inverse problem as a problem of optimization. An algorithm to achieve such objective is [7]:

- Suppose a solution θ to the inverse problem. This can include the assumption of an initial or boundary condition, or a typical parameter for a given problem.
- Feed the supposed condition to the direct solution of the partial differential equation system, calculating in this way values of the dependent variable y. Here the output of the direct solution is a vector of values corresponding to the times in which the values of y are measured. This vector of solutions will be denoted as calculated and it will be represented as y'.
- Compare the calculated values y' with the observed ones, measured in consistent times.

The success of this approach is the mechanism through which the supposed condition is improved in the subsequent invocations of the first step. Optimization is the procedure used to upgrade the estimates for the unknown conditions in order to minimize the objective function, given by:

$$SSE(y',\theta) = \sum_{t_i=1}^{N_T} (y(t_i) - y'(t_i,\theta))^2 \tag{1}$$

where θ represents the parameters to be estimated and N_T is the total number of experimental data.

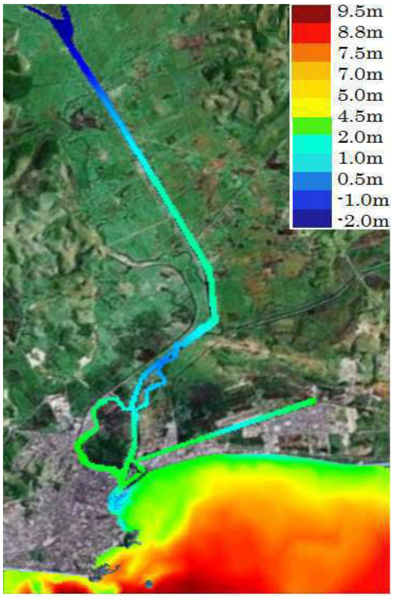

Figure 2. *Simulated domain and model bathymetry used.*

1.3 Parameter estimation in a model of estuarine hydrodynamics

The hydrodynamic and transport models built in the MOHID can be calibrated, for example, by adjusting, respectively, the bed roughness height (R) and the turbulent diffusion coefficient (D). On the other hand, the calibration procedure depends essentially on the field measurements. Regarding this aspect, there are two variables that are more easily obtained, the water level and salinity. Here a sensitivity analyses was performed to identify which of these variables would be more sensitive to the parameters of interest. The Macaé estuary bed is essentially dominated by sand sediments. According to [8], the bed roughness height in this case, must situate between 0.007 m and 0.05 m. On the other hand, the model configuration adopted here (time and space discretization) imposed a restriction regarding the turbulent diffusion coefficient, which might set between 0.01 m^2s^{-1} and 5.0 m^2s^{-1}, a quite large interval, anyway. Figure 3a compares the sensitivity coefficient of parameters in relation to the water level, where a much higher sensitivity in relation to R can be observed. Figure 3b shows the same results for salinity. Unlike water level, this variable is sensitive to both parameters. These results show that salinity simulation should be used in the parameters estimation.

1.4 Optimization Methods

The strategies described in Sub-sections 1.4.1 to 1.4.3 were employed to solve the inverse problem considered here. The Artificial Neural Network, described in 1.4.4 was also used for the inverse problem formulation and solution.

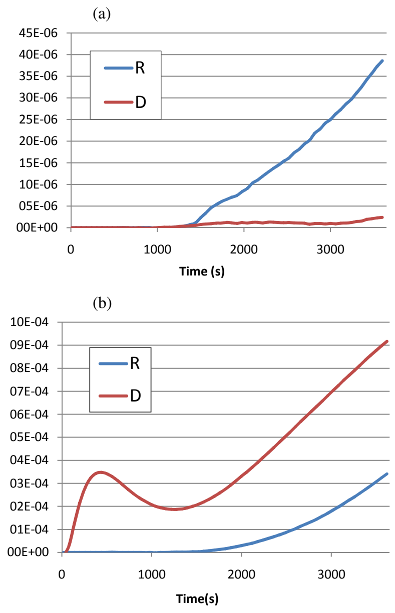

Figure 3. *Sensitivity coefficient of parameters in relation to water level (a) and salinity (b).*

1.4.1 Genetic Algorithm (GA)

The Genetic Algorithm (GA) is a method of Evolutionary Computation with several practical applications. The pseudocode 1 describes the general structure of an Evolutionary Algorithm.

```
Pseudocode 1: GA
$t = 0$;
Create an initial population $P(t)$;
WHILE no stopping criterion is satisfied DO
Evaluate the fitness of population $P(t)$;
Select reproducers from $P(t)$;
Generate a new population $P(t+1)$;
$t = t+1$;
END WHILE
```

The GA method is based in three basic operators, namely selection, crossover or recombination and mutation, detailed below:

Selection: selects chromosomes in the population for reproduction. The fitter a chromosome, more likely it is to be selected to reproduce;

Crossover: randomly chooses a locus in the string, which represents the parameters to be estimated. A new and different generation can be created from the exchange between two different strings. The crossover operator roughly mimics biological recombination between two single organisms chromosomes;

Mutation: randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. As pointed out by [9], mutation can occur at any position in the string, with a given probability, usually very small (e.g., 0.001).

These algorithms should work in a wide interval range for the parameters values, but with differences in the efficiency, highlighting the importance of the programmer strategy. In the GA, the fitness function evaluates the accuracy of the solution candidates. This function together with execution parameters defines the algorithm completely. Selection, recombination and mutation processes form a generation in the execution of a GA, and are executed until a satisfactory solution – defined by a stopping criterion or a specified number of generations - is reached.

1.4.2 Particle Collision Algorithm (PCA)

The PCA structure resembles the Simulated Annealing structure: first an initial configuration is chosen, then, there is a modification of the old configuration into a new one. The accuracy of these two configurations is compared [10]. A decision is then made on whether the new configuration is ‘’acceptable” or not. If that is the case, it serves as the old configuration for the next step. If it is not acceptable, the algorithm proceeds with a new change of the old configuration. PCA can also be considered a Metropolis algorithm, as a trial solution can be accepted with a certain probability. This acceptance may avoid the convergence to local optima. The main steps of the PCA are shown in the pseudocode 2 to 6.

Pseudocode 2: **Particle Collision Algorithms**

```
Generate an initial solution Old_Config;
Best_Fit ⟵ Fit(Old_Config);
FOR $n = 0$ to number of iterations DO
    Perturbation;
    IF Fit(New_Config) < Fit(Old_Config) THEN
        IF Fit(New_Config) <Best_Fit THEN
            Best_Fit ⟵ Fit(New_Config);
            Best_Config ⟵ New_Config;
        END IF
        Old_Config ⟵ New_Config;
        Exploration;
    ELSE
        Scattering;
    END IF
END FOR
```

Pseudocode 3: **Particle Collision Algorithms** (Cont.)
Function **Exploration**

```
FOR $n = 0$ untill the number of iterations DO
    Small_Perturbation;
    IF Fit(New_Config) < Fit(Old_Config)  THEN
        IF Fit(New_Config) <Best_Fit THEN
            Best_Fit $\longleftarrow$ Fit(New_Config);
            Best_Config $\longleftarrow$ New_Config;
        END IF
        Old_Config $\longleftarrow$ New_Config;
    END IF
END FOR
```

Pseudocode 4: **Particle Collision Algorithms** (Cont.)
Function **Scattering**

```
$P_{scattering} \longleftarrow 1 - \frac{Best\_Fit}{Fit(New\_Config)}$;
IF $P_{scattering}$ <Random(0:1) THEN
    Old_Config $\longleftarrow$ Aleatory solution;
ELSE
    Exploration;
END IF
```

Pseudocode 5: **Particle Collision Algorithms** (Cont.)
Function **Perturbation**

```
FOR $i = 0$ Untill (dimension of problem $-1$) DO
    Upper $\longleftarrow$ Upper limit [i];
    Lower $\longleftarrow$ Lower limit [i];
    Rnd $\longleftarrow$ Random(0:1);
    New_config[i] $\longleftarrow$ (3.old_config[i]+ (Lower + (Upper-Lower)Rnd))/4;
    IF New_Config[i] >Upper THEN
        New_Config[i] $\longleftarrow$ Upper limit [i];
    ELSE
        IF New_Config[i] <Lower THEN
            New_Config[i] $\longleftarrow$ Limite_Inferior[i];
        END IF
    END IF
END FOR
```

```
Pseudocode 6: **Particle Collision Algorithms** (Cont.)
Function **Small_Perturbation**
FOR $i = 0$ Untill (dimension of problem $-1$) DO
    Upper ⟵ Random(1,0:1,2)*Old_Config[i];
    IF Upper> Upper limit [i] THEN
        Upper ⟵ Limite_Superior [i];
    END IF
    Lower ⟵ Random(0,8:1,0)*Old_Config[i];
    IF Lower<Upper limit [i] THEN
        Lower ⟵ Lower limit [i];
    END IF
    Rnd ⟵ Random(0:10);
    New_config[i] ⟵ old_config[i]+((Upper-old_config[i]).Rnd);
    New_config[i] ⟵ New_config[i] - ((old_config[i]-lower).(1-Rnd));
END FOR
```

If the new configuration is better than the old one, the ''particle" is ''absorbed" and there is an exploration of the neighborhood, searching for an even better solution. Function "Exploration" performs this local search, generating a small stochastic perturbation of the solution inside a loop (inner loop). In the PCA canonical version it is a one-hundred-iteration loop. (see "Small Perturbation" in pseudocode 6). Otherwise, the "particle" is "scattered", which means that the new configuration receives random values within a given acceptable range. The scattering probability (pscattering) is inversely proportional to its quality. A high fitness particle will have a lower scattering probability.

1.4.3 Luus-Jaakola (LJ)

This method was introduced by [11] and its central idea consists on considering an initial broad search region for the parameters to be estimated. Random solutions are generated, whereas the search region becomes smaller as the iterations proceed. Pseudocode 7 describes the main steps of the method, where n_{out} and n_{in} are the numbers of iterations in the external and internal loops, respectively, chosen beforehand. R_j is a diagonal matrix of random numbers between [-0.5, 0.5] and ϵ is the coefficient of contraction of the search space.

1.5 Artificial Neural Network (ANN)

Artificial Neural Networks mimics biological neural networks, concerning their capability to learn and generalize. This is done through the basic structures of neural networks, the neurons. Eq. 2 shows the mathematical representation of a neuron:

$$out = f\left(\sum_{i=1}^{n} w_i d_i\right) \tag{2}$$

Where *out* is the output signal and d_i, is the neuron signals inputs, which are multiplied by weights, w_i, representing the synapses, and *f* is the activation function. The weights definition is done by a process called training. In a supervised training, two sets are used, one for data entry, called pattern, and another to output, called targets. In this case, the patterns are passed over the network and the weights are adjusted, in a way that the result is the expected target. The most common type of neural network using this type of training is the perceptron [12], which consists of a layer of inputs connected by paths, with weights associated with each neuron, fixed. The training makes use of an iterative tuning, which is repeated until the convergence to the weights is reached, i.e., until the network returns the expected target, defined a *priori* within a given tolerance.

```
Pseudocode 7: LJ
Algorithm Luus-Jaakola

Choose an initial search space $r^0$
Generate an initial solution $X^*$
FOR $i=1$ TO $n_{out}$ DO
    FOR $j = 1$ TO $n_{in}$ DO
        $X_j \leftarrow X^* + R_j r_{i-1}$
        IF $Fit(X_j) < Fit(X^*)$ THEN
            $X^* \leftarrow X_j$
        END IF
    END FOR
    $r_i = (1-\varepsilon) r_{i-1}$
END FOR
```

2 RESULTS

Estimation of the parameters of interest was performed using synthetic data generated with MOHID, adopting D and R, respectively, as 2.5 m^2s^{-1} and 0.025 m. The objective function minimization was done by the GA, PCA and LJ methods.

2.1 Genetic Algorithm (GA)

As experimental data, a total of 60 salinity values generated over a period of one hour of simulation were considered. In the GA, 30 generations were used, with a population of five individuals and crossover and mutation probability of, respectively, 50% and 4%. The estimated values for D and R were respectively of 2.4921 m^2s^{-1} and 0.0259 m. In order to evaluate de results accuracy, the relative error – $E(t)$ – between simulation and “measured“ data was adopted. Figure 4a presents the time evolution of this error. This time profile motivated a 12 hours (real time) simulation, assuming the same values for the parameters. Figure 4b shows that, even for this larger period of time, the error did not exceed 1%.

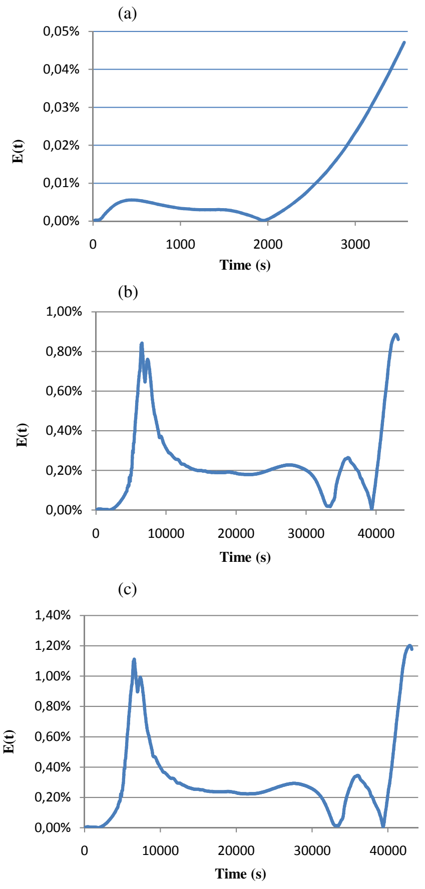

Figure 4. *Salinity relative error from GA estimative for: (a) 1 hour of simulation and considering experimental data acquired in the first hour; (b) 12 hours of simulation, considering experimental data acquired in the first hour; (c) 12 hours of simulation and considering noisy (4%) experimental data acquired in the first two hours.*

To assess the robustness of the method, a noise of 4% was added to the “experimental“ data. Firstly, the estimation was done considering the first hour of “experimental“ data, and poor estimates were obtained, of 1.7976 m^2s^{-1} and 0.0439 m, respectively for D and R. To improve these estimates, a period of 2 hour was considered, yielding to 2.4921 m^2s^{-1} and 0.0262 m, respectively. Figure 4c shows that, although the relative error trends to be larger than that observed with noiseless “experimental“ data, it did not exceed 1.20%.

2.2 Particle Collision Algorithm (PCA)

The same test case (considered in the previews section) was performed with the PCA, using 30 and 7 iterations, respectively for the outer and inner (exploration) loops. This roughly corresponds to a simulation with the same number of objective function evaluations of those performed in the GA estimates. Figure 5a shows the time evolution of the relative error, considering noiseless "experimental" data and one hour for estimation period. The worst, best and average results obtained with ten runs of the PCA are compared. It can be observed that even for the worst case, the maximum salinity relative error is of the order of 4.5%. Figure 5b presents the same comparison, with 4% of noise level and two hours for the estimation period. Here, much higher values for $E(t)$ can be observed.

2.3 Luus-Jaakola method (LJ)

In order to keep an equivalent number of objective function evaluations, the Luus-Jaakola method was run adopting internal and external loops of, respectively, 30 and 5, with a contraction coefficient of 0.05 for each test. Figure 6a shows the salinity relative error (worst, best and average) in 12 hours of simulation, considering "experimental" noiseless data, one hour for estimation period and adopting the estimates after ten runs of LJ. Figure 6b shows the results for another test, when it was considered "experimental" data with 4% of noise level acquired along the first two hours. The errors in both cases are less than 2.5%

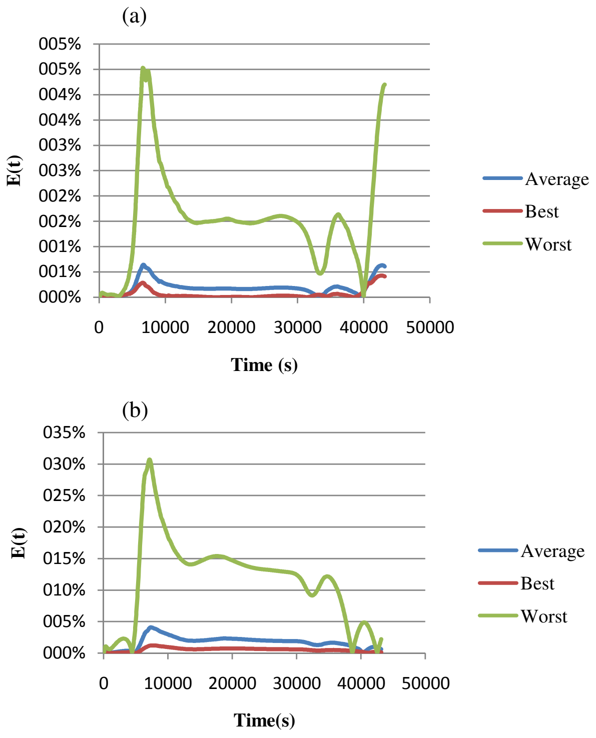

Figure 5. *Salinity relative error from PCA estimative, considering (a) noiseless experimental data and one hour for estimation period and (b) noisy (4%) experimental data and two hours for the estimation period.*

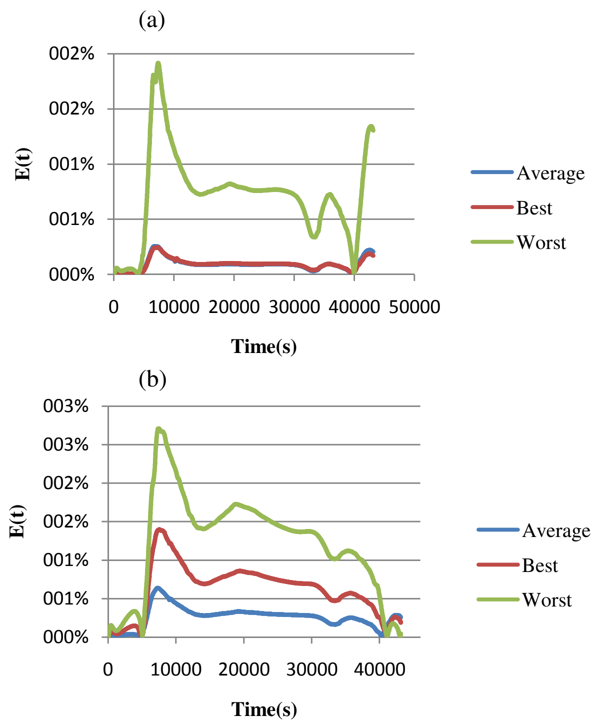

Figure 6. *Salinity relative error from LJ (ten runs) estimative in 12 hour of simulation considering (a) noiseless experimental data and one hour of estimation period and (b) noisy (4%) experimental data and two hours of estimation period.*

2.4 Artificial Neural Network (ANN)

Two ANNs were developed, one with three layers (one hidden), and the other with five layers (three hidden). For both ANNs, the input layer had 60 neurons, and the output layer had two neurons. The input data consisted of a 2 hours salinity time series "measured" in a given station at low tide. Here the expected ANNs output should be estimated values for mean R and D. Table 1 gives details of test cases.

Table 1. *Adopted test cases for the inverse problem solution using ANNs.*

Test cases	R (m)	D $(\mathrm{m}^2\mathrm{s}^{-1})$
1	0.007	0.1
2	0.007	5.0
3	0.05	0.1
4	0.05	5.0
5	0.025	2.5

The ANNs were trained with 600 patterns. Ten discrete values for R and D, respectively between 0.007 and 0.05 m, and between 0.1 $\mathrm{m^2s^{-1}}$ up to 5.0 $\mathrm{m^2s^{-1}}$ were considered, generating the first set of 100 training patterns. The other 500 patterns were generated adding noise to the original patterns with respectively following levels: 1%, 2%, 3%, 4% and 5%. The ANNs were trained using 20,000 epochs and the Gradient descent with momentum backpropagation optimization method for weight updating and bias values. In all layers the hyperbolic tangent sigmoid activation function was used. Tables 2 and 3 show the ratios between estimated and exact values for the mean roughness height, R'/R, and the turbulent diffusion coefficient, D'/D, for the five test cases listed in Table 1, respectively, with and without noise (4%) addition. For all these tests an ANN with three layers was used. For comparison, the maximum error between simulation and "observation", after a 12 hour simulation period, is showed.

The results show poor estimates for the mean R and/or D, with underestimation for both parameters. A marked improvement was observed with the application of a five layers ANN. Regardless this improvement, as a whole, the ANN approach method generated poor estimates for the parameters.

Table 2. *Three layers without noise.*

Test cases	R (m)	D $(\mathrm{m^2s^{-1}})$
1	0.007	0.1
2	0.007	5.0
3	0.05	0.1
4	0.05	5.0
5	0.025	2.5

Table 3. *Three layers with noise of 4%.*

Test cases	R (m)	D $(\mathrm{m^2s^{-1}})$
1	0.007	0.1
2	0.007	5.0
3	0.05	0.1
4	0.05	5.0
5	0.025	2.5

3 CONCLUSIONS

The present work dealt with a complex problem of estimating parameters in an estuarine simulation. The sensitivity analysis demonstrated the convenience of using the salinity as the observable variable. The inverse problem was then solved with three stochastic optimization methods, the Genetic Algorithm, the Particle Collision Algorithm and the Luus-Jaakola Method, coupled to hydrodynamic and transport models solved by MOHID. Artificial Neural

Networks were also used for the solution of the inverse problem. The optimization methods were implemented in a way to guarantee for all of them the same number of evaluations of the objective function, which is critical to define the computation time. With this regard, both GA and LJ had 150 evaluations to the objective function, whereas for the PCA this quantity was controlled by the probability associated with the exploration function calls, resulting in 163 evaluations in this present study.

Generally speaking, the optimization methods showed quite similar performances, with LJ presenting the best results with the lowest computational complexity. Although the GA method showed high efficiency as well, if compared to the other methods, it was disadvantageous due its binary approach, which imposes a limit to the search range (128 discrete values for each parameter in this study). It was also observed that, for all methods, noisy "measurements" required longer periods for estimation. The MOHID coupling with optimization algorithms seems to be a quite promising research area that can aid in the development of automatic calibration procedures.

REFERENCES

1. Shen, J., Jiab, J., Sissona and G. M., 2006. Inverse estimation of nonpoint sources of fecal coliform for establishing allowable load for Wye River, Maryland, Water Research (40): 3333-3342.
2. Sun, N.Z., 1994. Inverse Problems in Groundwater Modeling, Kluwer Academic Publishers, Dordrecht, The Netherlands.
3. Yeh, W.W.G., 1986. Review of parameter identification procedures in ground hydrology: The inverse problem, Water Resour. Res. (22): 95-108.
4. Yang, Z. and Hamrick, J. M., 2004. Optimal control of salinity boundary condition in a tidal model using a variational inverse method, Estuarine, Coastal and Shelf Science (62): 13-24.
5. Strub, I. S., Percelay, J., Stacey, M. T. and Bayen, A. M., 2009. Inverse estimation of open boundary conditions in tidal channels, Ocean Modelling (29): 85-93.
6. Amaral, K., 2003. Macaé River Estuary Computational Modeling as a Tool for the Integrated Management of Water Resources., COPPE/UFRJ /Rio de Janeiro. In Portuguese.
7. Karr, C.L., Yakushin, I. and Nicolosi, K., 2000. Solving inverse initial-value, boundary-value problems via genetic algorithm, Engineering Applications of Artificial Intelligence (13): 625-633.
8. Abbot, M. B. and Basco, D. R., 1989 Computational Fluid Dynamics, an Introduction for Engineers., Logan Group, UK Limited.
9. Melanie, M., 1999. An Introduction to Genetic Algorithms, A Bradford Book The MIT Press Cambridge, Fifth edition.
10. Sacco, W. F., Oliveira, C. R. E. and Pereira, C. M. N. A., 2006. Two stochastic optimization algorithms applied to nuclear reactor core design, Progress in Nuclear Energy (48): 525-539.
11. Luus, R. and Jaakola, T. H. I., 1973. Optimization by Direct Search and Systematic Reduction of the Size of Search Region, AIChE Journal (19): 760-766.
12. Fausett, L., 1994. Fundamentals of Neural Networks, Prentice-Hall, Englewood Cliffs, NJ.