Function Approximation of Seawater Density Using Genetic Algorithm

Abdulrahman Ahmed Bobakr Baqais, Moataz Ahmed, and Mostafa H. Sharqawy

Abstract— **Function Approximation is a popular engineering method used in system identification or equation optimization. Artificial Intelligence (AI) techniques have been used extensively to spot the best curves that match the real behavior of the system due to the wide spectrum of the search space. Genetic algorithm is well-known for its fast convergence and ability to find an optimal structure of the solution. In this paper, we propose using a genetic algorithm method as a function approximator to get a correlation for seawater density. We will use a polynomial form of the approximation. After implementing the algorithm, the results from the produced function are compared with the real data used in the algorithm.**

Index Terms— **Genetic Algorithm; function approximation; system identification, correlation.**

I. Introduction

FUNCTION approximation refers to finding an approximate relation for a set of input-output pairs of data that connects well with the data. Function approximation is an interesting method that has many applications in control, communication, and many engineering problems. In many of these problems the behavior of the unknown function is usually nonlinearl. Therefore, a basic regression or correlation may not lead to the best function that suit the data. The basis of modeling an unknown function from available data is strictly mathematical and it's beyond the scope of this paper. However, a brief description of some of the mathematical expressions that can be used as a function approximator is summarized here for introductory purposes.

Polynomial is a well-known approximator of many functions such as Legendre[1]. Different forms of polynomials have been used for approximation such as Taylor's expansion and Chebyshev polynomial. If the interval on which the data lies can be segmented into different regions and evaluated or approximated with different polynomials then this is called Splines.

Manuscript received March 18, 2013; revised April 03, 2013. This work was supported through the Center for Clean Water and Clean Energy at King Fahd University of Petroleum & Minerals (KFUPM) under Project Number R13-CW-10.

Abdulrahman Ahmed Bobakr Baqais is a PhD at the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia (corresponding author to provide e-mail: baqais@kfupm.edu.sa).

Moataz Ahmed is with the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia (e-mail: moataz@kfupm.edu.sa).

Mostafa H. Sharqawi is with the Department of Mechanical, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia (e-mail: mhamed@kfupm.edu.sa).

To find the best polynomial that approximates the behavior of certain data, many techniques have been used. Two of which are widely used which are statistical-based regression and Artificial Intelligence (AI) techniques. In this paper, the artificial intelligence (AI) technique is used because of the fast convergence and high accuracy results obtained from such a stochastic search mechanism of AI method. The importance of finding a function approximator lies in its ability to be generalized [2]. So according to Draelos and Hush [2], the ability of one model to be entitled as a function approximator is realized in its ability to find the function and recognize it as well.

In this paper, we will focus on using genetic algorithm only as a function approximator, but through the discussion we will explain the rationale behind this choice and compare it with other AI techniques. In section II, an introduction to genetic algorithm is given. In section III, previous approaches found in the literature are summarized, in section IV, our implementation design and environmental setup is given, and in section V and VI results and analysis are presented.

II. Evolutionary Computing (Genetic Algorithms)

Evolutionary computing refers to a computer algorithm which has the ability and capability to evolve through multiple runs of the algorithm. Evolution indicates that out of the population set, the algorithm can provide a solution space where optimized solutions are presented and inadequate solutions are removed and replaced with better ones. Hence, the evolutionary computing comes as it resembles the fundamental of evolutionary theory where the survival is only for the best. Gradually, different terminology creeps into the field such as evolutionary programming, genetic algorithms, and genetic programming. Genetic algorithms are extension to the concept of evolutionary computing. Evolutionary computing initially was invested only on the mutation operator through different generation. Genetic algorithm enhances the algorithm by adding crossover and inversion, both of which mimic biological functions. Genetic programming: provides a new presentation of the solution based on a tree-like encoding scheme.

• The process of genetic algorithm usually has generic sets of steps [3]

- Initialization of population randomly.
- Computing the fitness function
- Selection of solution from the solution pool based on

their fitness function

- Apply genetic algorithm operators
- Termination criteria

The population should be chosen carefully in term of size and heterogeneity. Size of the population may have a consequence on the complexity time of the algorithm. Heterogeneity must also be considered as the solution sets must be as different as possible to furnish for weaker solutions evolving and allows stronger solution to emerge and stabilize.

Each solution is represented as a chromosome and each chromosome includes genes where each gene represents a particular feature. This is very suitable in biology, but to represent that in computer science domain, a mechanism must be devised to serve for that purpose. One common technique for encoding is using binary numbers. Therefore, a chromosome may have 10 binary numbers (Genes) where each one of them can indicate the presence of that feature (could be labeled 1) or the absence of that gene from the organism (labeled value 0). This has its advantage in some problems like combinatorial problem [4]. The use of binary encoding has been the norm in genetic algorithm and sometimes is called pure genetic algorithm. However, designing different strategies to encode the genetic algorithm is possible. In fact, one of the main issues that contributes to generate better solution lie in the transformation phase of the problem. Transforming computer science problems to mimic the behavior of evolving biological organism is one of the hardest aspects for designing genetic algorithm. As a rule-of-thumb, the more efficient the designing of genetic algorithm, the better the solution would be. In addition, , a complex encoding strategy may result on cascading the complexity to other genetic algorithm operators like crossover and mutation. This consequently may result in various ways of applying genetic algorithm operators [5]. However, no general encoding rule can be applied for all problems [6] Moreover, the same problem can be encoded by more than one method.

Many of genetic algorithms are used for optimization problems because of the way fitness function is computed. Fitness function is used to quantify the optimality of a solution [7]. Obviously, when the problem represents numeric values, genetic algorithm can be used straightforwardly to find optimized values. However, non-numeric values may cause a problem. Thus, designing a fitness function may require a level of abstraction of the problem, an understanding of different solution layers, and recognition of optimized solutions.

The fitness function is a measurement indicator in a selection process. It is usually encoded as a function with range of values. The higher the fitness function, the more plausible the solution is.

Selection Operator is a mechanism used to transfer the successful candidates of solutions from one generation to another generation. This process should be chosen very carefully as it is important to get a clear final solution. Selection operator should strive to cover a wide enough set of potential promising solutions and leave out hopeless weaker individuals that have no or little impact on the final solution. Different strategies of selection operator have been suggested in the literature. The Elitist's method chooses the best individual whereas the Roulette's wheel method is an alternative strategy based on probability. A blended approach is possible. Ranking and Tournament could be used too.

In Elitist method, not all members have a chance of selection, only top members are shown and the remaining will be discarded. Afterwards, the genetic algorithms operation is preferred on the selected candidates. This operation must be performed in a sophisticated manner to ensure that the best properties of candidate solutions are mixed to produce a better solution. Roulette's wheel is an algorithm which uses probability theories to select a solution with higher fitness. In Ranking Selection: the rank or the ordered value of all individuals is considered while in Tournament, a set of individuals are compared in each run and the best are selected.

Genetic Algorithm Operators: In biology, strong offspring is a result of two partners who mate and reproduce. This process is called crossover. Different variation of crossover may apply. Another genetic algorithm operation is mutation, where part of the solution is flipped or inverted. Different variation of mutation exists whether to flip a bit, range of bits, or may increase/decrease the value of a bit. Both crossover and mutation could hold some probability values of their occurrence. These values obviously are more related to fitness function. Mutation and crossover may occur at one or different points in the gene encoding and sometimes are referred to as "crossover rate" [5]. Crossover technique can be divided into:

- One point crossover
- Two points crossover
- Multipoint crossover
- Uniform crossover

Mutation can introduce diversity to the population [7]. Different mutation operator may apply to the chromosome:

- Substitution
- Deletion
- Duplication
- Inversion
- Insertion

The final step in genetic algorithms is termination [8] Termination can be determined when an optimized solution with acceptable error is found or when the scheme of genetic algorithms is preserved. The last one is obvious since the main goal of genetic algorithms is to search for optimal solution as soon as all members of solution space become similar. In addition, genetic algorithms should be terminated when the search mechanism is unable to find a better solution.

III. Previous Works

There are many attempts to use various AI techniques to solve the problem of function approximation. Many of these techniques are either based on one model or incorporating two or more models together to achieve higher accuracy. The use of neural network as function approximator has been extensively investigated [2],[9]-[11] since this is the objective of inventing a neural network model. The issues of using neural network model to approximate the relations between input and outputs can be summarized in the following four points [12]:

- Difficulty in decoding the hidden layers functions and their number of nodes
- Demand for larger samples
- Strongly dependent
- Random initialization of weights leading to difficulty of reproducing

Tsai,, Chung, and Chang [9] argue that neural network models, are particular RBFF; is not efficient in approximating constant values in the output or constant values in some intervals of the outputs since they adopt Gaussian as their activation functions. In addition, Artificial Neural Network (ANN) has long computation times and is not conveniently adaptable [11].

Many researchers used hybrid models such as fuzzy neural network which was employed by Simpson and Jahns [13] to find all fuzzy sets embedded in the problem and to combine them linearly for approximation. The idea is based on the area bounded by minimum and maximum values. So for each cluster or fuzzy set, the set can be expanded as necessary to cover another value or pruned in the case of overlapping. Simpson and Jahns [13] tested their approach using the following function.

$$f(x) = 2x/(1+x^2) \tag{1}$$

They used 300 sample points and they got a very close approximation of the real function output.

Wang, Lee, Liu and Wang[14] employed the hybrid method of using fuzzy neural network model with a robust learning to approximate different known functions such as the sine and Gaussian and as well as to approximate the surface.

Kuo, Hu, and Chen [15] used a hybrid model incorporating Radial Basis Neural Network (RBNN) genetic algorithms, Particle Swarm Optimization (PSO), and Self Organizing Map (SOM). RBNN was used however as the core approximator where the genetic algorithm was employed to enhance the results. Kuo, Hu, and Chen [15] noted that blending genetic algorithm with RBNN in one model is complicating the model and extending the computation time though it performs better. So, they proposed to inject Self-Organizing map in the procedure to reduce the effect of the complexity of the model on the computation time and combine the evolving feature of the genetic algorithm with the memory preserving nature of PSO to ensure the diversity in the population. The implementation was carried on different functions based on standalone models and their proposed hybrid model. Their results brings the least percentages of errors.

Another possibility of a hybrid model is to combine genetics algorithm and fuzzy logic in one model as described by Ashrafzadeh, Nowicki, Mohamadian, and Salmon. [16] where the fuzzy system is used as the core of approximation and genetics algorithm were applied to search for the optimum member functions and fuzzy rules.

Kwon, Moon, and Hong [17] developed two genetic algorithms: one based on parametric model and the second is based on non-parametric model. On the other hand, Kumar, Chandra, and Kumar [18] approximated the fuzzy system based on Feed Forward Neural Network (FFNN).

Another technique that has shown interesting results is the clustering technique. Finding function approximation using clustering can be achieved when the output points lies in different intervals that can be replaced by clusters. Gonzalez, Rojas, Ortega, and Prieto [19] provided a detailed analysis of applying clustering technique for function approximation. They claim that their method can handle noise data efficiently by clustering them and it requires no prior knowledge of the structure of the input or output data.

IV. Problem Description

In this paper, we would like to find a correlation for the density of seawater as a function of three independent variables; temperature, salinity and pressure. Since the available data for this experiment considers the density at atmospheric pressure level which is constant, we eliminated the pressure variable. The data obtained contains 130 points of water density at different salinity and temperature measurements given by Sharqawy, Lienhard, and Zubair [20]. The temperature (t) range is from 0-90 °C in increments of 10 degrees, while salinity(s) ranges from 0-120 g/kg. in increments of 10 g/kg.

V. Research Methodology:

In the present work, Matlab[21] Global Optimization Toolbox is used to give a quick access to various parameters of the genetic algorithm such as selection, number of generation and for plotting purposes. However, the design of the chromosome and the fitness function is coded to be adjusted to our problem.

In function approximation, the most significant feature of the solution is to have a high accuracy of the obtained curve drawn by input-output pairs. This implies very low relative error in approximating each point to the actual data. Therefore, the fitness function is to minimize rather than to maximize. To ensure, that our fitness function is coded correctly, the errors must be decreasing through the generations until a convergence or an optimal solution is found. Mean Relative Absolute Error (MRAE) was used as a fitness function and it’s defined as:

MRAE = 1/n ∑ (|approximated-actual|/ actual)

The population of our solutions is represented as binary bits. This gives us the flexibility we need to manipulate the chromosome in a variety of ways as we are going to explain shortly. Each chromosome (or individual) contains many terms where each term is composed of three blocks:

- Coefficient
- Power of the temperature
- Power of the salinity

For simplicity we will assume that all terms are polynomial functions since polynomial functions are known to be a universal function approximator. So the chromosome will have the following shape as in Fig1.

Coefficient	*T Power*	*S Power*	*Coefficient*	*T Power*	*S Power*	…	*Coefficient*	*T Power*	*S Power*
Term 1			Term 2			…	Term N		

Fig1. Chromosome Design Blocks

In these polynomial functions, the power of each term may have a non-integer value. In normal regression approximation, the researcher is constrained to use only integer values in the power block. However, using a resolution concept, we are allowing both the coefficient and the power to have real values. Table I shows the range of coefficient and power values.

Another aspect that is considered is the probability of crossover and mutation and the range of bits that are applied to. Crossover and mutation can be implemented in various ways: either on the term boundary, on the block unit boundary, or within the individual bits. Each method has its merits and drawbacks based on the targeted problem. In this specific chromosome design, we changed the crossover and mutation probabilities within the bits themselves. This is done for two reasons: the first is because all of the terms are in polynomial shape and hence targeting the term boundary will not speed up the converging of the solution. If our chromosome design exhibits other function types such as trigonometric function, then targeting terms boundary may have better results. The second reason is that manipulating individual bits increases the chances of finding better solutions. Even though, it may increase the complexity of the problem and the search space to explore in general large problems. The simplicity of our method is that we allocate only 30 bits to the term which mitigate these issues and the extra time added will be negligible or of little significance as shown in the experiment sections.

TABLE I
VARIABLE RESOLUTION

Variable	Range
Coefficient	0-1024(in 1 increment)
T Power	0-10.24 (in 0.1 increment)
S Power	0-10.24 (in 0.1 increment)

I. EXPERIMENT DESCRIPTION AND SET UPS

The control parameters used in the genetic algorithm are as follows:

- Number of Individuals: 1000
- Number of Generations: 150
- Crossover operators: Two point.
- Crossover rate: 0.80
- Mutation rate: 0.01

II. EXPERIMENT RESULTS AND ANALYSIS

Using the above design and parameters, we run the algorithm for two cases: 2 terms, and 4 terms.

Case 1: Two terms:
Fitness Value: 0.015
Density =

$$940 + 32 * S^{0.2783} \tag{2}$$

Case 2: Four terms
Fitness Value: 0.0101
Density =

$$980 + 3 * S^{0.7168} \tag{3}$$

The previous two experiments show that the algorithm can converge very quickly to an acceptable accuracy. Only with 150 generations, the algorithm using two terms is able to find an equation with absolute average deviation of 1.5%. In the second case using four terms, the algorithm is able to find an equation with a higher equation of approximately absolute average deviation of 1.1%. Appendix shows a comparison between the measured data and the calculated data using (2) and (3) for the salinity ranges between 0-100 g/kg.

III. CONCLUSION

The previous two experiments show that genetic algorithms can successfully find a good polynomial approximator to the density data with fewer numbers of terms. By only using 4 terms we are able to find a good approximation with an average deviation of only 1.1%. Evidently, if we increase the number of terms then the accuracy will be higher and better.

APPENDIX

S [g/kg]	**T [C]**	**Density [kg/m3]**	**4 Terms Equation**	**2 Terms Equation**
0	0	999.788	980.000	940.000
0	10	999.652	980.000	940.000
0	20	998.158	980.000	940.000
0	30	995.602	980.000	940.000
0	40	992.17	980.000	940.000
0	50	987.991	980.000	940.000
0	60	983.154	980.000	940.000
0	70	977.728	980.000	940.000
0	80	971.761	980.000	940.000
0	90	965.291	980.000	940.000
10	0	1007.917	995.629	1000.737
10	10	1007.464	995.629	1000.737
10	20	1005.774	995.629	1000.737
10	30	1002.767	995.629	1000.737

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

S [g/kg]	T [C]	Density [kg/m3]	4 Terms Equation	2 Terms Equation
10	40	999.152	995.629	1000.737
10	50	994.949	995.629	1000.737
10	60	990.176	995.629	1000.737
10	70	984.851	995.629	1000.737
10	80	978.993	995.629	1000.737
10	90	972.62	995.629	1000.737
20	0	1015.938	1005.686	1013.659
20	10	1015.196	1005.686	1013.659
20	20	1013.192	1005.686	1013.659
20	30	1010.133	1005.686	1013.659
20	40	1006.478	1005.686	1013.659
20	50	1002.247	1005.686	1013.659
20	60	997.457	1005.686	1013.659
20	70	992.129	1005.686	1013.659
20	80	986.28	1005.686	1013.659
20	90	979.931	1005.686	1013.659
30	0	1023.958	1014.350	1022.458
30	10	1022.941	1014.350	1022.458
30	20	1020.654	1014.350	1022.458
30	30	1017.537	1014.350	1022.458
30	40	1013.836	1014.350	1022.458
30	50	1009.572	1014.350	1022.458
30	60	1004.762	1014.350	1022.458
30	70	999.426	1014.350	1022.458
30	80	993.584	1014.350	1022.458
30	90	987.255	1014.350	1022.458
40	0	1031.995	1022.216	1029.331
40	10	1030.715	1022.216	1029.331
40	20	1028.16	1022.216	1029.331
40	30	1024.978	1022.216	1029.331
40	40	1021.226	1022.216	1029.331
40	50	1016.923	1022.216	1029.331
40	60	1012.089	1022.216	1029.331
40	70	1006.743	1022.216	1029.331
40	80	1000.905	1022.216	1029.331
40	90	994.595	1022.216	1029.331
50	0	1040.472	1029.539	1035.055
50	10	1038.389	1029.539	1035.055
50	20	1035.71	1029.539	1035.055
50	30	1032.457	1029.539	1035.055
50	40	1028.647	1029.539	1035.055
50	50	1024.302	1029.539	1035.055
50	60	1019.44	1029.539	1035.055
50	70	1014.08	1029.539	1035.055
50	80	1008.243	1029.539	1035.055
50	90	1001.948	1029.539	1035.055
60	0	1048.274	1036.455	1040.002
60	10	1046.079	1036.455	1040.002
60	20	1043.305	1036.455	1040.002
60	30	1039.973	1036.455	1040.002
60	40	1036.1	1036.455	1040.002
60	50	1031.707	1036.455	1040.002
60	60	1026.813	1036.455	1040.002
60	70	1021.437	1036.455	1040.002
60	80	1015.598	1036.455	1040.002
60	90	1009.316	1036.455	1040.002
70	0	1056.137	1043.050	1044.386
70	10	1053.821	1043.050	1044.386
70	20	1050.945	1043.050	1044.386
70	30	1047.526	1043.050	1044.386
70	40	1043.585	1043.050	1044.386
70	50	1039.14	1043.050	1044.386
70	60	1034.209	1043.050	1044.386
70	70	1028.813	1043.050	1044.386
70	80	1022.97	1043.050	1044.386
70	90	1016.699	1043.050	1044.386
80	0	1064.063	1049.383	1048.338
80	10	1061.617	1049.383	1048.338
80	20	1058.629	1049.383	1048.338
80	30	1055.117	1049.383	1048.338
80	40	1051.101	1049.383	1048.338
80	50	1046.599	1049.383	1048.338
80	60	1041.629	1049.383	1048.338
80	70	1036.209	1049.383	1048.338
80	80	1030.359	1049.383	1048.338
80	90	1024.096	1049.383	1048.338
90	0	1072.052	1055.496	1051.948
90	10	1069.465	1055.496	1051.948
90	20	1066.357	1055.496	1051.948
90	30	1062.746	1055.496	1051.948
90	40	1058.649	1055.496	1051.948
90	50	1054.085	1055.496	1051.948
90	60	1049.071	1055.496	1051.948
90	70	1043.625	1055.496	1051.948
90	80	1037.764	1055.496	1051.948
90	90	1031.507	1055.496	1051.948
100	0	1080.103	1061.418	1055.279
100	10	1077.366	1061.418	1055.279
100	20	1074.129	1061.418	1055.279
100	30	1070.411	1061.418	1055.279
100	40	1066.229	1061.418	1055.279
100	50	1061.598	1061.418	1055.279

S [g/kg]	T [C]	Density [kg/m3]	4 Terms Equation	2 Terms Equation
100	60	1056.536	1061.418	1055.279
100	70	1051.06	1061.418	1055.279
100	80	1045.187	1061.418	1055.279
100	90	1038.933	1061.418	1055.279

REFERENCES

[1] M. A. Ahmed and K. A. DeJong, "Function approximator design using genetic algorithms," in , IEEE International Conference on Evolutionary Computation, 1997, 1997, pp. 519 –524.

[2] T. Draelos and D. Hush, "A constructive neural network algorithm for function approximation," presented at the , IEEE International Conference on Neural Networks, 1996, 1996, vol. 1, pp. 50 –55 vol.1.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[4] R. Curry and M. Heywood, "Towards Efficient Training on Large Datasets for Genetic Programming," in Advances in Artificial Intelligence, A. Y. Tawfik and S. D. Goodwin, Eds. Springer Berlin Heidelberg, 2004, pp. 161–174.

[5] O. Räihä, "Applying Genetic Algorithms in Software Architecture Design," 2008.

[6] S. Ronald, "Robust encodings in genetic algorithms: a survey of encoding issues," in , IEEE International Conference on Evolutionary Computation, 1997, 1997, pp. 43 –48.

[7] K.-L. Du and M. N. S. Swamy, Neural Networks in a Softcomputing Framework. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[8] Real Life Applications of Soft Computing - CRC Press Book. .

[9] J.-R. Tsai, P.-C. Chung, and C.-I. Chang, "A sigmoidal radial basis function neural network for function approximation," presented at the , IEEE International Conference on Neural Networks, 1996, 1996, vol. 1, pp. 496 –501 vol.1.

[10] T. Varshney and S. Sheel, "Approximation of 2D function using simplest neural networks #x2014; A comparative study and development of GUI system," presented at the 2010 International Conference on Power, Control and Embedded Systems (ICPCES), 2010, pp. 1 –4.

[11] H. Gao, "The probability characteristic of function approximation based on artificial neural network," presented at the 2010 International Conference on Computer Application and System Modeling (ICCASM), 2010, vol. 1, pp. V1–66 –V1–71.

[12] Y. Mizukami, Y. Wakasa, and K. Tanaka, "A proposal of neural network architecture for non-linear function approximation," presented at the Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004, 2004, vol. 4, pp. 605 – 608 Vol.4.

[13] P. K. Simpson and G. Jahns, "Fuzzy min-max neural networks for function approximation," presented at the , IEEE International Conference on Neural Networks, 1993, 1993, pp. 1967 –1972 vol.3.

[14] W.-Y. Wang, T.-T. Lee, C.-L. Liu, and C.-H. Wang, "Function approximation using fuzzy neural networks with robust learning algorithm," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 27, no. 4, pp. 740 –747, Aug. 1997.

[15] R. J. Kuo, T.-L. Hu, and Z.-Y. Chen, "Evolutionary Algorithm Based Radial Basis Function Neural Network for Function Approximation," presented at the 3rd International Conference on Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009, 2009, pp. 1 –4.

[16] F. Ashrafzadeh, E. P. Nowicki, M. Mohamadian, and J. C. Salmon, "Optimal approximation of nonlinear functions by fuzzy systems," presented at the IEEE 1997 Canadian Conference on Electrical and Computer Engineering, 1997. Engineering Innovation: Voyage of Discovery, 1997, vol. 2, pp. 781 –784 vol.2.

[17] Y.-K. Kwon, B.-R. Moon, and S.-D. Hong, "Critical heat flux function approximation using genetic algorithms," IEEE Transactions on Nuclear Science, vol. 52, no. 2, pp. 535 – 545, Apr. 2005.

[18] V. S. Kumar, S. V. Chandra, and C. S. Kumar, "Neuro-Fuzzy Function Approximations Using Feedforward Networks - An Application of Sigmoidal Signal," presented at the 2010 Ninth International Conference on Machine Learning and Applications (ICMLA), 2010, pp. 895 –898.

[19] J. Gonzalez, H. Rojas, J. Ortega, and A. Prieto, "A new clustering technique for function approximation," IEEE Transactions on Neural Networks, vol. 13, no. 1, pp. 132 –142, Jan. 2002.

[20] M.H. Sharqawy, J.H. Lienhard V, S.M. Zubair, Thermophysical Properties of Sea Water: A Review of Existing Correlations and Data, Desalination and Water Treatment, 16 (2010) 354 – 380.

[21] Matlab 7.7.0.471(R2008b), The MathWorks.Inc, 2008