Estimation of Wind Turbine Generator Model Parameters using Artificial Intelligence Methods

R.Y. Kazakova, S.I. Nedelcheva, R.K. Popov

***Abstract*— Modelling (in a broad sense) is an essential tool for research in all areas and represents a scientifically based method for assessing the performance of systems and processes used for making engineering decisions. This applies in particular to the field of management systems, where the foundation is making decisions based on the information received.**

The existing and newly designed systems effectively examined by using the mathematical models (analytical and spoofing) which allows identifying some constant parameters that are involved in the differential equations representing the dynamics of the system analyzed. Such systems may come from a broad scientific spectrum, for example from economics and biology from communication and weather forecasting.

The present paper investigates some Artificial Intelligence (AI) methods identifying the parameters of a dynamical system. Two types of methods are compared - 'evolution' and 'particle swarm' intelligence. First, for this purpose, a system simulation model generating data (for the two methods of identification in order to compare afterwards the results) is used. After that, Genetic (GA) and Particle Swarm Optimization (PSO) algorithms are applied to estimate the wind turbine generator model parameters. The results of both methods are compared in terms of their accuracy and performance. The software for the simulation and AI process has been developed using MATLAB™.

Index Terms— ***Artificial intelligence, system parameter estimation, genetic algorithm, particle swarm optimization, wind turbine generator system model***

I. Introduction

THE main focus in system identification is on the parameter estimation process. Well-developed techniques such as least-square (LS), instrumental variable and maximum likelihood exist for parameters estimation of models. However, these techniques often fail in search for the global optimum if the search space is not differentiable or linear in the parameters.

To date, artificial intelligence (AI) techniques have become potential candidates to many control applications. One of the most powerful AI techniques is genetic algorithm (GA), which has been widely used and applied to control systems [1]. Genetic algorithms are very good for optimization problems with several local minima where conventional search algorithms fail. GA techniques can be effectively applied to system identification problem to estimate the model parameters. A GA simultaneously evaluates many points in the parameters space and converges toward the global solution. It does not require the search space to be differentiable or continuous [3,4]. Many researchers have applied the GA techniques to identify linear and non-linear systems. The wind turbine generator system is a complex nonlinear system with parameters that are difficult to identify using standard LS techniques.

R. KAZAKOVA is with the Department of Optoelectronics and Laser Engineering, Technical University of Sofia, Branch Plovdiv, Bulgaria (e-mail: rrrrosi@abv.bg).

S.I NEDELCHEVA is with the Department of Electrical Engineering, Electronics and Automation in Faculty of “Engineering and Pedagogy-Sliven” at Technical University of Sofia. (e-mail: stefned@abv.bg).

R. POPOV is with the Department of Electronics, Communications and Information Technologies, Plovdiv University “Paisii Hilendarski”, Bulgaria (e-mail: rum_pop@yahoo.com).

II. Estimation Methods

A. Genetic algorithm overview

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. We can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- *Selection rules* select the individuals, called *parents*, which contribute to the population at the next generation.
- *Crossover rules* combine two parents to form children for the next generation.
- *Mutation rules* apply random changes to individual parents to form children.

GA was pioneered in 1975 by Holland [5], and its concept is to mimic the natural evolution of a population by allowing solutions to reproduce, creating new solutions, which then compete for survival in the next iteration. The fitness improves over generation sand the best solution is finally achieved. The initial population, *P(0)*, is encoded randomly by strings. In each generation, *t*, the more fit elements are selected for the mating pool; and then processed by three basic genetic operators, reproduction, crossover, and mutation, to generate new offspring. On the basis of the principle of survival of the fittest, the best chromosome of a candidate solution is obtained. The pseudo code of GA illustrates the procedure of the computation as follows:

```
Procedure GA
begin
    t=0
    initialize P(t)
    evaluate P(t)
    while not satisfy stopping rule do
        begin

            t= t+1
            select P(t) from P(t1)
            alter P(t)
            evaluate P(t)
        end
    end
```

The power of GA lies in its simultaneous searching a population of points in parallel, not a single point. Therefore GA can find the approximate optimum quickly without falling in to a local optimum. In addition GA does not have the limitation of differentiability, as do the mathematical techniques. These characteristics of GA are the reasons it is used here for the problem of model identification in ARIMA models.

B. *Concepts of the PSO approach*

The neural network is a simplified model of human brain; genetic algorithm is inspired by the human evolution. Here we discuss another type of biological system - social system, more specifically, the collective behaviors of simple individuals interacting with their environment and each other. Someone called it as swarm intelligence. All of the simulations utilized local processes, such as those modeled by cellular automata, and might underlie the unpredictable group dynamics of social behavior.

There are two popular swarm inspired methods in computational intelligence areas: Ant colony optimization (ACO) and particle swarm optimization (PSO). ACO was inspired by the behaviors of ants and has many successful applications in discrete optimization problems [6].

The particle swarm concept originated as a simulation of simplified social system. The original intent was to graphically simulate the choreography of bird of a bird block or fish school. However, it was found that particle swarm model can be used as an optimizer [7].

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart [8] and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as GA. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. It is called "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called p_{best}. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called g_{best}. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called l_{best}.

After finding the two best values, the particle updates its velocity and positions with following equations (1) and (2).

$$v[\,] = v[\,] + c1 * rand(\,)*(p_{best}[\,] - present[\,]) + \\ +c2*rand()*(g_{best}[\,] - present[\,]) \quad (1)$$

$$present[\,] = persent[\,] + v[\,] \quad (2)$$

v[ ] is the particle velocity, *persent[ ]* is the current particle (solution). $p_{best}[\,]$ and $g_{best}[\,]$ are defined as stated before. *Rand ( )* is a random number between *(0,1). c1, c2* are learning factors. Usually *c1 = c2 = 2*. The pseudo code of the procedure is as follows:

```
for each particle
  Initialize particle
end
do
    for each particle
        Calculate fitness value
        if the fitness value is better than the best fitness value
           (pbest) in history
        set current value as the new pbest
    end
    Choose the particle with the best fitness value of all the
       particles as the gbest
    for each particle
       Calculate particle velocity according equation (1)
       Update particle position according Equ. (2)
    end
while maximum iterations or minimum error criteria is not
attained
```

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, which is a parameter specified by the user, then the velocity on that dimension is limited to V_{max}.

DOI: 10.17694/bajece.52662

III. Model Description

The system, modeled in this article represents a wind turbine generator. It consists of three main subsystems: wind turbine; synchronous generator and mechanical part.

A. Mathematical description

The mechanical subsystem is represented by the Simulink-blocks which correspond to the equation:

$$M_T - M_G = J\frac{d\omega}{dt} \qquad (3)$$

For a description of processes in the synchronous generator using the model of generalized electric machine axes d and q (d axis coincides with the axis of the poles and q-axis with geometric neutral line). The axes d and q are considered connected with the poles of the rotor. Windings of the armature are replaced with two equivalent in effect windings the axes d and q with respective electromotive forces - voltage, resulting from the relative rotation of the actual stator windings in the coordinate system of the rotor.

The mathematical description of the dynamic processes in the synchronous electric drive is obtained by the equations write mechanical properties at axes d and q, associated with the rotor, on which is disposed exciting coil:

$$\begin{aligned}
&u_{id} = R_1 i_{1d} + p\Psi_{1d} - \omega_{1\text{ел}}\Psi_{1q}\\
&u_{iq} = R_1 i_{1q} + p\Psi_{1q} + \omega_{1\text{ел}}\Psi_{1d}\\
&u_{\text{в}} = R_{\text{в}} i_{\text{в}} + p\Psi_{\text{в}}\\
&p_n\left(\Psi_{1d} i_{1q} - \Psi_{1q} i_{1d}\right) + \beta(\omega_0 - \omega_1) - M_{12} - M_{c1} = J_1 \mathrm{p}\omega_1\\
&M_{12} - M_{c2} = J_2 \mathrm{p}\omega_2,
\end{aligned} \qquad (4)$$

where:

$$\begin{aligned}
&\Psi_{1d} = L_{1d} i_{1d} + L_{12d} i_{\text{в}};\\
&\Psi_{1q} = L_{1q} i_{1q};\\
&\Psi_{\text{в}} = L_{\text{в}} i_{\text{в}} + L_{12d} i_{1d};\\
&M_{12} = c_{12}(\varphi_1 - \varphi_2)
\end{aligned} \qquad (5)$$

$\beta = 2M_k/\omega_0 s_k$ is the modulus of firmness of the asynchronous component to a moment determined by the operation of the snubber winding. In our case the snubber winding absent and the component $\beta(\omega_0 - \omega_1)$ is equal to zero.

The output power of the aerodynamic the wind turbine is described by the following equation [2]:

$$\mathrm{Pm} = C_p(\lambda,\beta)\frac{\rho A}{2}v_{wind}^3\,, \qquad (6)$$

where:

P_m	- Mechanical output power of the turbine (W);
c_p	- Performance coefficient of the turbine;
ρ	- Air density (kg/m^3);
A	- Turbine swept area (m^2);
υ	- Wind speed (m/s);
λ	- Tip speed ratio of the rotor blade tip speed to wind speed
β	- Blade pitch angle (deg).

Equation (6) can be normalized. In the per unit (pu) system we have:

$$P_{m\_pu} = k_p c_{p\_pu} v_{wind\_pu}^3, \qquad (7)$$

where :

P_{m_pu} - Power in pu of nominal power for particular values of ρ and A;

c_{p_pu} - Performance coefficient in pu of the maximum value of c_p;

υ_{wind_pu} - Wind speed in pu of the base wind speed. The base wind speed is the mean value of the expected wind speed in m/s;

k_p - Power gain for $c_{p\_pu} = 1$ and $\upsilon_{wind\_pu} = 1$ pu, k_p is less than or equal to 1

A generic equation is used to model $c_p(\lambda,\beta)$. This equation, based on the modeling turbine characteristics of [1], is:

$$c_p(\lambda,\beta) = c_1\left(\frac{c_2}{\lambda_i} - c_3\beta - c_4\right)e^{\frac{-c_5}{\lambda_i}} + c_6\lambda \qquad (8)$$

with

$$\frac{1}{\lambda_i} = \frac{1}{\lambda + 0.08\beta} - \frac{0.035}{\beta^3 + 1} \qquad (9)$$

The coefficients c_1 to c_6 are: $c_1 = 0.5176$, $c_2 = 116$, $c_3 = 0.4$, $c_4 = 5$, $c_5 = 21$ and $c_6 = 0.0068$.

The mechanical power P_m as a function of generator speed, for different wind speeds and for blade pitch angle β = 4 degree and base wind speed = 12 m/s , is illustrated below – Fig. 1.

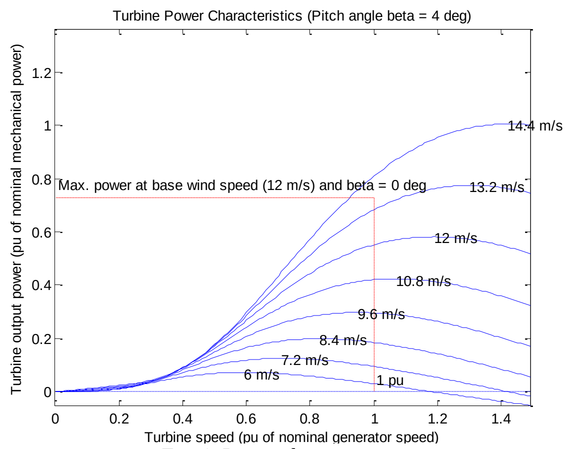


Fig. 1. Power characteristics

B. The model structure

The structural scheme of the mathematical model of the wind generator is shown in Fig.2. The system consists of a generator, mechanical subsystem and a wind turbine.

The mechanical subsystem is represented by Simulink-blocks Sum, Gain2 and Int1, which correspond to equation (3).

The scheme of the model of the synchronous generator is shown in Fig.3. It is made up on the basis of the system of equations (4) and (5).

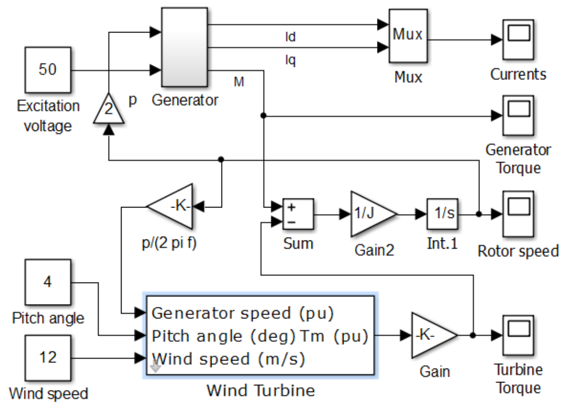


Fig. 2. The structural scheme of the mathematical model of the wind generator

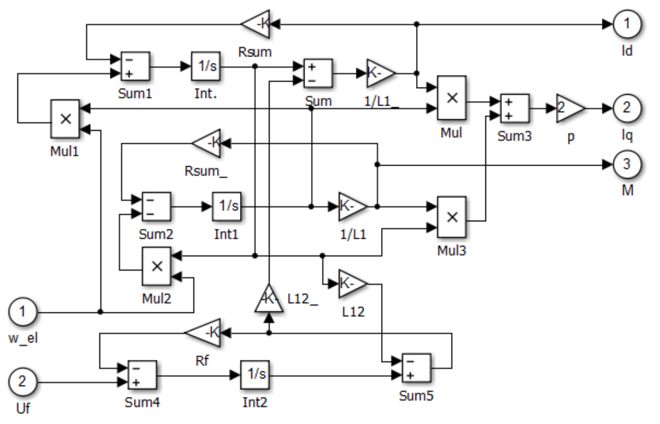


Fig. 3. The structural scheme of the synchronous generator mathematical model

For a description of the wind turbine a library model of Matlab [Simulink-Sim Power Systems], [2] has been used

IV. Simulations

The basic simulation was performed at the following parameter values:
Estimated parameters:
Rsum = 0.894 - Sum of armature and load resistance (Ohm);
R_f = 2.524; - Excitation armature resistance (Ohm);
L_{12} = 0.152 - Mutual inductance (H);
L_1 = 0.0117 - Own synchronous inductance (H);
J = 0.02 - Total inertia ($kg.m^2$)
Other input parameters:
Generator excitation voltage = 50 V;
Initial rotor speed 100 rad/s;
Wind Speed = 12 m/s;
Pitch Angle = 4 deg;
The simulation results are illustrated in Fig.4

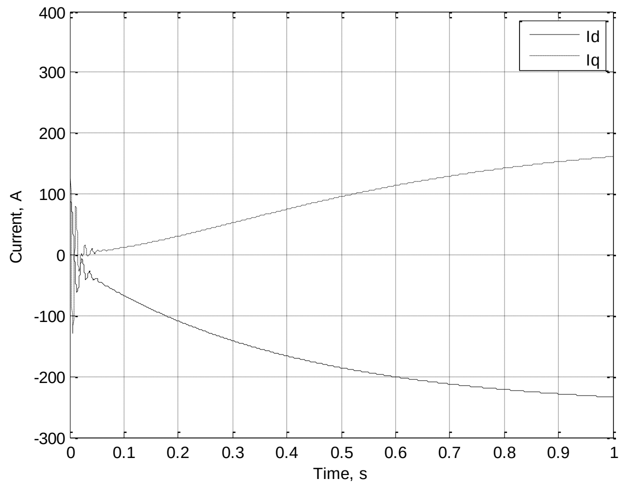


Fig. 4. Simulation of change of currents along the axes d and q

Experiment Design

System, with parameters have to be estimated is represented by the wind turbine generator system model, produced in Simulink environment as a block diagram. This model was described in details above. There are five parameters in system model to be identified: Rsum , Rf , L12, L1, J . Monte Carlo simulation planning using central composite design is performed. Bounds for each parameter are defined in the parameter space. Two algorithms: real coded GA and PSO are applied and tested. Firstly, we have to choice input factors to be varied. In GA algorithm they are:

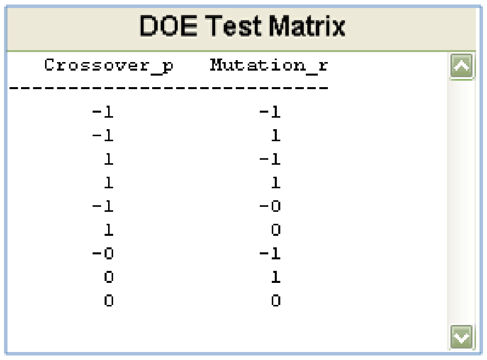
- Crossover_p – Crossover probability parameter;
- Mutarion_r – Mutation rate.

Varried parameters in PSO algorithm are chosen to be:

- Correction_factor- is the correction (learning) factor;
- Inertia- is the inertia factor.

In the next step central composite design is performed. Only one central point is used in case to obtain uniform values distribution in the parameter space. For this purpose, is used Matlab function **ccdesign.** Design results are showed on Fig. 5. Function **plotGaSim.m** is used here to display results.
After that parameter variance bounds are chosen:

- Crossover_p = 0 … 1;
- Mutation_r = 0.05 … 0.5



DOE Test Matrix

```
  Crossover_p   Mutation_r
--------------------------
      -1           -1
      -1            1
       1           -1
       1            1
      -1           -0
       1            0
      -0           -1
       0            1
       0            0
```

Fig. 5. Two factors central composite design test matrix

The position of the factors in the space of the parameters in GA is shown in Fig.6.

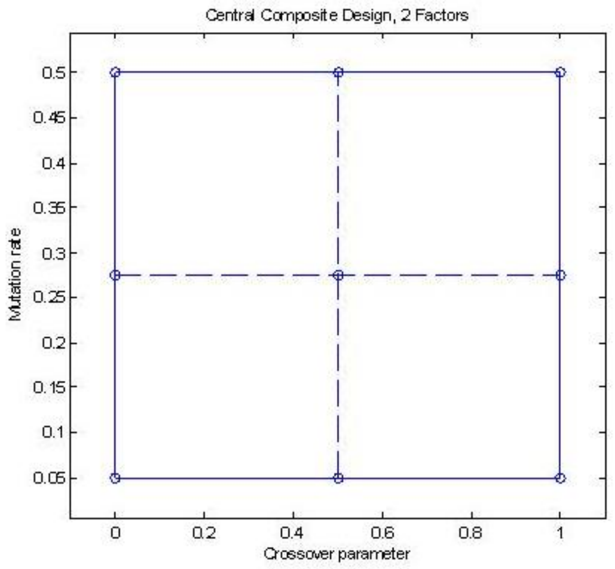


Fig.6. Positions of the input variables in the parameter space in GA tests.

Central composite design for the PSO – test parameters variation in Monte Carlo simulation are presented on fig 7.

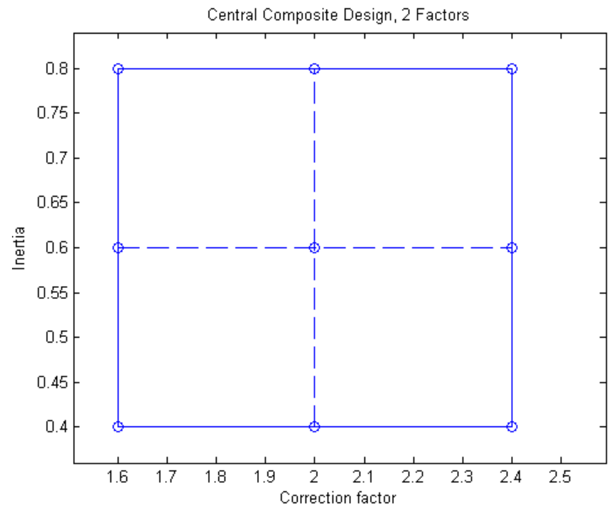


Fig.7. Positions of the input variables in the parameter space in PSO tests

The real coded GA search algorithm uses BLX-α crossover operation and two varied parameters: crossover parameter and mutation rate. It is written as a Matlab – function with the next syntax:

[A B AvSAE]=ga_proc(err, popsize, crossover_p, mutation_r, n_generations), where:

A is a vector, containing evaluated 'a' parameters;
B is a vector, containing evaluated 'b' parameters (for future use);
AvSAE is a vector of average sum of absolute error of the model for all generations;
err is an error tolerance;
popsize is the size of population;
crossover_p is the crossover parameter
mutation_r is the mutation rate;
n_generations is the number of generations.

The two parameters in PSO-based search algorithm may vary too: correction factor and inertia. The related code is written as a Matlab – function with the next syntax:

[A B AvSAE]=pso_proc(swarm_size, correction_factor, inertia, iterations, err), where:

A, B, AvSAE and err are the same as in function ga_proc;
swarm_size is the size of the swarm;
correction_factor is the correction (learning) factor;
inertia is the inertia factor
iterations is the number of iterations.

Each algorithm was evaluated 90 times with varied parameters.

Evaluation function setting of the model is the same for GA and PSO. It is based on a comparison of the outputs of the model and evaluation model (changes in the values of the currents Id and Iq in time). First configure the evaluation model on which to set the value of the current set of genes (model parameters). Then he started end to give the current realization of output functions. Finally, calculate the average of the sum of absolute error AvSAE the current configuration of the evaluated parameters. The syntax of the function is as follows:

fit = fitness5(gene),

where the gene is a vector containing the current set of genes (model parameters).

V. Results

Test results for first 5 estimations, using real coded GA and PSO are reported in Table 1 and Table 2 respectively.

TABLE I
MONTE CARLO SYSTEM PARAMETER ESTIMATION TEST RESULTS USING GA

Run Number	Crossover probability	Mutation rate	AvSAE final value	R_{sum},	R_f,	L_{12},	L_1,	J
				Ohm	Ohm	H	H	Kg.m^2
1	0.6	0.05	8.7812	0.83969	2.1095	0.14288	0.01199	0.0812
2	0.2	0.5	25.5266	0.98713	3.0567	0.31685	0.0236	0.01458
3	1	0.275	46.705	0.53758	0.9729	0.074		0.01351
4	1	0.5	37.1226	0.69762	1.4955	0.07375	0.00768	0.00523
5	1	0.05	57.3622	0.48262	0.6502	0.05	0.00609	0.0095

TABLE II
MONTE CARLO SYSTEM PARAMETER ESTIMATION TEST RESULTS USING PSO

Run Number	Correction factor	I nertia	AvSAE final value	R_{sum},	R_f,	L_{12},	L_1,	J
				Ohm	Ohm	H	H	Kg.m^2
1	2	0.8	5.493	1.0016	2.5279	0.61985	0.0481	0.01592
2	1.6	0.6	70.32	3.1422	19.9406	0.99357	0.01348	0.2
3	2	0.8	4.102	0.9912	2.5451	0.40782	0.0316	0.01449
4	2.4	0.6	7.983	1.0482	2.5577	0.8053	0.06334	0.005
5	2.4	0.8	15.566	0.9133	2.4368	0.54215	0.04386	0.15934

VI. Discussion

The significance analyzes for the varied parameters was preformed too. Combined scatter plot for GA and PSO parameters significance analyzes is showed on Fig.8 and Fig.9 respectively. Best values for the crossover parameter and mutation rate are 0,6 and 0,5. Best values for the correction factor and inertia are 2,0 and 0,8.

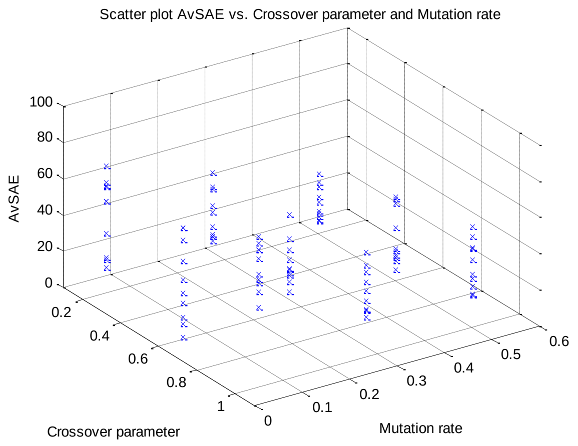


Fig. 8 Scatter plot of AvSAE vs. Crossover parameter and Mutation rate in GA tests.

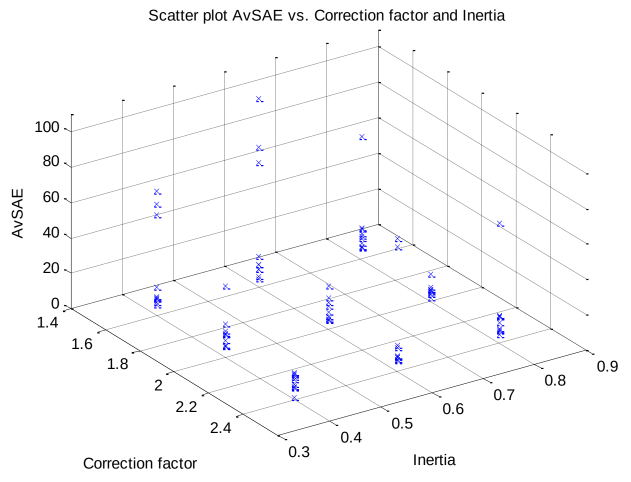


Fig.9. Scatter plot of AvSAE versus Correction factor and Inertia in PSO tests

Provided null hypothesis F-test for each of tested algorithms returns the result as listed below:

GA: F > P for all varied factors
Crossover parameter is significant: F = 7.5913, P = 0,1764.
Mutation rate is significant: F = 13.5861, P = 0,1764

PSO: F > P for all varied factors
Correction Factor is significant: F = 1.4571, P = 0,1764
Inertia is significant: F = 0.1955, P = 0,1764

The average values of the parameters calculated by GA and PSO algorithm are showed in the following Table 3. It may be noted that in this case PSO gives more accurate results. Are taken over the values of the rows of tables with minimum error AvSAE (Average Value of Sum of Absolute Error) - respectively 79 rows of PSO Table 2 and row 27 of GA of Table 1.

TABLE III
MEAN VALUES OF THE ESTIMATED PARAMETERS BY GA AND PSO ALGORITHMS

	Rsum	Rf	L_{12}	L_1	J
Original	0.894	2.524	0.152	0.0117	0.02
PSO	0.9362	2.4347	0.18024	0.01439	0.02089
GA	0.88925	2.5581	0.15901	0.01212	0.0195
Error PSO, %	- 4.72	- 3.53	-18.57	- 22,99-	- 4.45
Error GA, %	- 0.53	- 1.35	- 4.61	- 3.58	2.5

The next Fig.10 is showed sorted values of AvSAE for both algorithms PSO (in red) and GA (in blue).

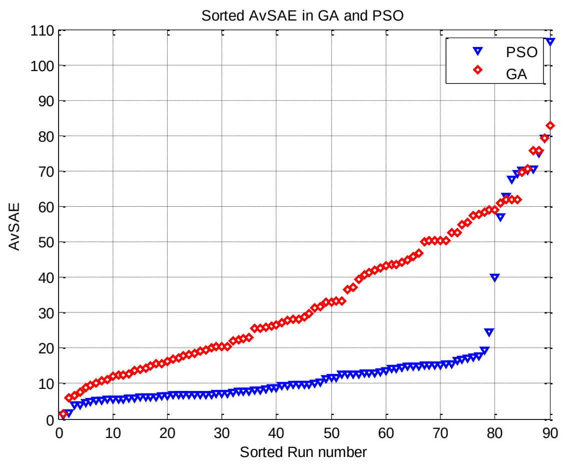


Fig.10. Sorted by values of AvSAE for both algorithms PSO (in red) and GA (in blue)

Test duration for all 90 estimations in each case (GA and PSO) was measured to check algorithms' performance and the reported results are:

GA test duration – 2 h, 32 min, 50.7 s
PSO test duration – 2 h, 37 min, 18.6 s

VII. CONCLUSIONS

1. Both GA and PSO algorithm are suitable for use in the evaluation process the parameters of the wind turbine generator system.
2. Table 3 and Fig.10 showed, that PSO procedure reports much more accurate estimation of the system parameters for the wind turbine generator system model.
3. Best value for the crossover parameter in GA is 0,6. For the mutation rate this value is 0,5.
4. Best value for the correction factor in PSO is 2.0. For the inertia this value is 0,8
5. GA reports a bit better efficiency, but the difference in test time duration is very small. This is due to the fact that the main time in carrying out the tests is spent for the simulation model of the wind turbine and to evaluate the setting of the model - 2,700 times for each of the cases. Procedures themselves take up very little processor time.

REFERENCES

[1] Bedwani, W, A., Ismail, O. M. Genetic optimization of variable structure PID Control System, In: ACS/IEEE International Conference on Computer Systems and Applications, 2001, pp. 27–30.
[2] Siegfried Heier, "Grid Integration of Wind Energy Conversion Systems," John Wiley&Sons Ltd, 1998, ISBN 0-471-97143-X
[3] Kargupta, H., Smith, R. E., System identification with evolving polynomial networks, Proceeding of the 4th International Conference on Genetic Algorithm, University of California, San Diego, USA, 1991, pp. 370-376.
[4] Kristinsson, K,, Dumont, G, System identification and control using Genetic Algorithms, Ieee Transactions on Systems, Man and Cybernetics, 1992 22 (5), pp, 1033–1046,
[5] Holland J.H., Adaptation in natural and artificial system, Ann Arbor, The University of Michigan Press, 1975.
[6] http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html
[7] http://www.engr.iupui.edu/~shi/Coference/psopap4.html
[8] http://www.engr.iupui.edu/~eberhart/web/PSObook.html

BIOGRAPHIES

ROSITSA KAZAKOVA was born in Belozem, Bulgaria, in 1957. She received the M.S degree (Automatic Control Systems) in University of Food Technologies of Plovdiv in 1983. Since 1988 she is an assistant professor in Technical University of Sofia, branch Plovdiv.
She is research interest are in Artificial Intelligence, Measurement and Automation.

STEFKA NEDELCHEVA is Professor, PhD Eng. and from 2001 she is Head of Department “Electrical Engineering, Electronics and Automation” in Faculty of “Engineering and Pedagogy- Sliven” at Technical University of Sofia. She becomes a Ch. Assistant Professor in the Department of "Electrical Engineering" in 1990, then Associate Professor in 1995 and Head of Department "Electrical Engineering" (1996-1997).
Lectures on “Electrical Networks”, “Electrical Energy Production”, “Mathematical Methods in Electrical Energy Production”, “Decentralized Energy Sources”, “Ecology and Renewable Energy Sources”, “Power line Protection Design” and “Smart grids”.
Interests: Study and modeling modes of operation in electrical networks and systems, application of mathematical methods in electricity, decentralized electricity sources.

RUMEN POPOV was born in Plovdiv, Bulgaria, in 1964. He received the M.S. degree (Automatic Control Systems) in Technical University of Tula, Russia, in 1990 and the Ph.D. in Technical University of Ruse, Bulgaria in 2009.
From 1990 to 1991, he was a Research Engineer with the Aviotechnics SA, Plovdiv at Radio Electronic Systems department. Since 1991, he has been an Assistant Professor with the Control Systems Department, Technical University of Sofia, branch Plovdiv. He became an associated professor at the Technical University of Sofia, Bulgaria in 2011. At present hi is an associated professor at Department of Electronics, Communications and Information Technologies in Plovdiv University “Paisii Hilendarski” He is the author of more than 62 articles. His research interests include Measurement and Automation, Renewable Energy Systems, Sun - Tracking systems. His main courses are in Measurement Systems, SCADA Systems, PLC and Signals and Systems. Assoc. Prof. Popov is a member of the Bulgarian Geothermal Association since 2011.