A Tool For Automatic Determination Of Model Parameters Using Particle Swarm Optimization

Willy Nzale[*, a], Hossein Ashourian [a], Jean Mahseredjian [b], Henry Gras [a]

[a] *PGSTech, Montréal, QC H2K 1C3, Canada*
[b] *Polytechnique Montréal, Montréal, QC H3T 1J4, Canada*

***Abstract*--This paper presents a tool developed in EMTP to automatically determine model parameters for matching existing field measurements. The tool uses the particle swarm optimization (PSO) algorithm to calibrate or update existing models. To enhance the performance of the tool, a technique used to improve PSO efficiency is also proposed.**

Two test cases are presented. The first case aims to determine the parameters of the reactive power control loop in a PV park controller model. The second case finds the unknown parameters in an exciter model of a synchronous machine connected to a grid.

***Keywords*: Digital-twin, Electromagnetic Transients, EMTP, Particle Swarm Optimization, Renewables.**

1. INTRODUCTION

Modern simulation tools, such as EMTP [1], are capable of predicting network performance under various operating conditions and assist engineers in taking informed decisions. Accurate predictions and analysis require accurate models of simulated grids. Some models may require a large number of parameters (model data). Due to unavailability of data from equipment manufacturers, it is common to assign typical values to some unknown parameters. This process can be based on engineering judgement or typically available data. Model accuracy can be further improved through availability of measurements in the actual grid.

The concept is to use the response of the actual network component and tune its model parameters to accurately reproduce it. For example, the measured waveform of the reactive power generated by a PV, corresponding to a step change, can be utilized to calculate the settings of the reactive power control loop gains in the park controller model.

This paper presents a tool based on particle swarm optimization (PSO) for matching simulation and measurement results. PSO is an algorithm that finds the optimal solution of a problem using an iterative process in which a candidate solution is improved with respect to a quality criterion. PSO has been applied in [2]. Optimization enabled EMT simulations were initially presented in [3] and [4]. They have been also used in [5] for the design of power systems. Contrary to these works, the research in this paper focuses on the operational level of power systems. It presents a contribution for tuning parameters of existing power system components with limited availability of data. The implementation of PSO is also improved by proposing a recalibration approach for search intervals. It is also demonstrated in a real existing case context and can be easily extended to other real cases. Our work subscribes in the field of digital twin research where the model must be recalibrated to imitate field measurements.

This paper implements the PSO method for calculating unknown parameters in an EMT-type software for Inverter Based Resource models and synchronous generators. It can be effectively extended to other models in EMT-type simulations.

Section II of this paper recalls the concept of PSO, section III describes how this concept is implemented inside the EMTP tool, with proposed features, including a technique to improve the efficiency of the traditional PSO method. In sections IV and V, demonstration examples are presented to evaluate the proposed tool's performance.

2. PARTICLE SWARM OPTIMIZATION METHOD

2.1. Overview

Particle Swarm Optimization was introduced in [6],[7]. Initially, it was used to simulate social behavior [8]. The reference [9] describes some important aspects of PSO and swarm intelligence in general. A survey of PSO applications is presented in [10] and [11]. Reference [12] shows a review on some PSO based works.

Basically, the PSO technique uses a population of *n* particles. For each $j \in [1, n]$ and at each time *t*, the *j*-th particle $X_j(t)$, which is a candidate solution to the problem to be solved, is represented by an *m*-dimensional real-valued vector:

$$X_j(t) = \left[x_{j,1}(t), x_{j,2}(t), ..., x_{j,m}(t) \right] \quad (1)$$

where *m* is the number of optimized parameters, $x_{j,k}(t)$ is the value of the *k*-th parameter in the *j*-th candidate solution and $k \in [1, m]$. Each optimized parameter is constrained to vary in a predefined range:

$$x_{j,k}(t) \in \left[x_k^{\min}, x_k^{\max} \right] \quad (2)$$

where $x_k^{\min}$ and $x_k^{\max}$ are lower and upper limits of the *k*-th parameter to optimize.

In PSO, particles move in a multidimensional search space at given velocities. The velocity of the *j*-th particle is:

[*]Corresponding author.
Email address: willy-arnaud.nzale-mimbe@polymtl.ca (W. Nzale).

Paper submitted to the International Conference on Power Systems Transients (IPST2023) in Thessaloniki, Greece, June 12-15, 2023.

$$V_j(t) = \left[v_{j,1}(t), v_{j,2}(t), ..., v_{j,m}(t) \right] \quad (3)$$

where $v_{j,k}(t)$ is the velocity component of the *j*-th particle for the *k*-th parameter. $v_{j,k}(t)$ must always be limited to enhance the local exploration of the problem space by some maximum [8]. Therefore:

$$v_{j,k}(t) \in \left[-v_k^{\max}, v_k^{\max} \right] \quad (4)$$

$$v_k^{\max} = \frac{x_k^{\max} - x_k^{\min}}{N} \quad (5)$$

where N is a number of intervals selected for the *k*-th parameter. Particles change their positions after updating their velocities. Any velocity update involves three components: inertia, cognition and social [2], [8].

The contribution of the inertia component is controlled by a coefficient called the inertia weight $w(t)$. More explanations are given in [13]. It is customary to decrease the value of $w(t)$ over iterations (for better local exploration). One approach is to use the decrement function presented in (6) and coming from [2]:

$$w(t) = \alpha w(t-1) \quad (6)$$

where α is a constant smaller than 1.

The contributions of cognition and social components are evaluated with the knowledge of the individual best position $X_j^*(t)$ for any of the *j*-th particles in the population, and the knowledge of the global best position $X^{**}(t)$, which is the best position among all best positions found so far by each individual particle. The best position particle is identified using particle fitness value. The fitness value is calculated with the fitness function F (or objective function) [8]. This function is to be defined for each optimization problem. The fitness value associated to the *j*-th particle $X_j(t)$ is $F(X_j(t)) = F_j$. The fitness value associated to the global best particle position $X^{**}(t)$ is named F^{**} or simply F_{best}, whereas the fitness value associated to the individual best particle position $X_j^*(t)$ is named F_j^*.

2.2. PSO algorithm

The following steps describe the iterative searching process of the PSO algorithm [2],[6],[8]. For the optimization problem in hand, the formulation of the fitness function F must be done prior to the execution of the following steps.

1. Initialize particles at $t = 0$.
   1.1. Randomly generate *n* particles, $\{X_j(0), j = 1, 2, ..., n\}$, where $X_j(0) = \left[x_{j,1}(0), x_{j,2}(0), ..., x_{j,m}(0) \right]$. To get $x_{j,k}(0)$, simply select a random value inside the *k*-th parameter search space defined in (2). In the same way, randomly generate initial velocities, $\{V_j(0), j = 1, 2, ..., n\}$, where $V_j(0) = \left[v_{j,1}(0), v_{j,2}(0), ..., v_{j,m}(0) \right]$. To get $v_{j,k}(0)$, simply select a random value inside $\left[-v_k^{\max}, v_k^{\max} \right]$.
   1.2. Evaluate each particle using the fitness function F.
   1.3. For each particle, set the individual best $X_j^*(0) = X_j(0)$ and the associated fitness value $F_j^* = F_j$, $j = 1, 2, \ldots, n$.
   1.4. Search for the best fitness function value F_{best} and set the associated particle as the global best, $X^{**}(0)$ with a fitness value of $F^{**} = F_{best}$.
   1.5. Assign a value to the inertia weight coefficient $w(0)$.
2. Set $t = t + 1$, then use (6) to update the inertia weight.
3. For each particle candidate, update the velocity vector in (3) with the following formula:

$$\begin{aligned} v_{j,k}(t) = {} & w(t) v_{j,k}(t-1) \\ & + c_1 r_1 \left(x_{j,k}^*(t-1) - x_{j,k}(t-1) \right) \\ & + c_2 r_2 \left(x_{j,k}^{**}(t-1) - x_{j,k}(t-1) \right) \end{aligned} \quad (7)$$

   where c_1 and c_2 are constants (both positive); r_1 and r_2 are random numbers selected in [0,1]. Then, check the velocity limits (see (4)). If the velocity goes out of bounds, bring it back to its limit value. In (7), the first, second and third terms in the sum respectively stand for the inertia, the cognition and the social components.
4. For each particle, update the position vector in (1) with the following formula:

$$x_{j,k}(t) = v_{j,k}(t) + x_{j,k}(t-1) \quad (8)$$

5. Evaluate each particle's position using the fitness function, then update each particle individual best and the global best if necessary.
6. Go back to 2 and repeat the process iteratively until a stopping criterion is satisfied. The criterion may be a predefined maximum number of iterations or a threshold for fitness values. In each case, go to 7.
7. The particle position for which the fitness value is equal to the global best value is selected as the final solution of the optimization problem.

3. PSO IMPLEMENTATION FOR MODEL PARAMETER DETERMINATION

3.1. Overview and assumptions

This section describes how the PSO algorithm is implemented in the proposed tool in EMTP. The goal of the tool is to determine some unknown parameter values in the model of any component in a power network so that the model

response matches the corresponding physical response obtained by on-site measurements. The features coded in the tool include PSO parameter selections, fitness function definition, stopping criteria definition, and a proposed technique to improve efficiency of PSO algorithm under some conditions.

It is assumed that the operating conditions of the studied grid are replicated in the complete grid model. The targeted component (or device) model has the same operating settings as its physical twin in the field. The reference measurements obtained from the grid include sufficiently accurate recordings of active and reactive powers, voltages and/or currents in time-domain. It is also assumed that the parameters of the component model that have significant impact on its response are identified.

3.2. PSO particle definitions and parameter selections

In the proposed tool, the set of device parameter values to be determined represents a particle candidate. Each particle involved in the search process consists of *m* real values, where *m* is the number of parameters.

In our implementation, the parameters of the PSO algorithm are defined as follows: the value of the initial weight is $w(0)=1$, the value of α in (6) is 0.99. Also, $c_1=c_2=2$ and N in (5) is set to 5. As stated in [2] and [14], these PSO parameter values generally lead to better performance. Furthermore, in our implementation, the velocity update will be done using the constriction factor χ presented in [15]-[17]. χ is given by:

$$\chi=\frac{2\kappa}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|} \tag{9}$$

where $\phi=c_1+c_2$, and κ is an arbitrary constant in the range $[0,1]$. In our implementation, κ is equal to α defined in (6). Using χ, (7) is therefore replaced by the following:

$$\begin{aligned} v_{j,k}(t)=\chi\Big[& w(t)v_{j,k}(t-1) \\ & +c_1r_1\left(x^{*}_{j,k}(t-1)-x_{j,k}(t-1)\right) \\ & +c_2r_2\left(x^{**}_{j,k}(t-1)-x_{j,k}(t-1)\right)\Big] \end{aligned} \tag{10}$$

During the search process, if at any moment, the updated velocity $v_{j,k}(t)$ calculated using (10), gets out of the boundaries defined by (4), $v_{j,k}(t)$ is changed to become the exceeded boundary velocity, that is: $v_{j,k}(t)=v_k^{\max}$ (if $v_{j,k}(t)>0$), or $v_{j,k}(t)=-v_k^{\max}$, (if $v_{j,k}(t)<0$). The same solution is applied if the updated position of any particle (calculated using (8)) goes beyond the limits defined in (2). Furthermore, in this later case, it is important to re-initialize the velocity $v_{j,k}(t)$ to avoid the same phenomenon at the next iteration. In our implementation, we propose to re-initialize the velocity by a random value and in the opposite direction, that is:

$$v_{j,k}(t)=-R()\,v_{j,k}(t) \tag{11}$$

where $R()$ is a random number in $[0,1]$.

3.3. Fitness function definition

To evaluate the position of each particle candidate and determine individual and global bests after each iteration, a fitness function must be defined. In our implementation, the fitness function is built from sampled data in time-domain of both experimental and simulated waveforms. The experimental data, named reference waveform or experimental waveform $f_{exp}(t)$, is measured in the field at a specific location. The simulated data, named simulated waveform $f_{sim}(t)$, is captured at the same location but in the EMTP model (after running a simulation). The goal of the optimization is to make the simulated waveform match the reference one.

The fitness function can be defined in several ways. One simple approach which is implemented in the tool is the « *sum absolute error*» fitness function. Its formulation is:

$$F=\sum_{i=1}^{N_S}\left|f_{\exp}(i)-f_{sim}(i)\right| \tag{12}$$

where N_S is the number of sampled data values in the observation interval. After the update of a particle position, the obtained parameter values are entered in EMTP, a simulation is run to obtain $f_{sim}(t)$ and the fitness value is evaluated using (12).

3.4. Stopping criteria

PSO is an iterative process. In our implementation, the iterations are stopped when one of these conditions is met:

- the predefined total number of iterations $N_{\max iter}$ is reached;
- the fitting error e_{fit} gets less that a predefined threshold, named *convergence tolerance* in the tool.

e_{fit} is calculated for each particle position candidate right after its fitness value. e_{fit} is a number that expresses the relative mismatch error between the simulated waveform and the experimental waveform:

$$e_{fit}(\%)=\frac{\sum_{i=1}^{N_S}\left|f_{\exp}(i)-f_{sim}(i)\right|}{\sum_{i=1}^{N_S}\left|f_{\exp}(i)\right|}\times 100 \tag{13}$$

When PSO stops due to the first condition, the particle position with the lowest fitness value found so far, is taken as the final solution. When the optimization stops due to the second condition, the last particle position (which is the one

with e_{fit} less than the threshold) is taken as the final or the optimal solution.

3.5. Proposed technique to improve PSO efficiency

In most cases, the PSO algorithm can reach an accurate optimal solution. An accurate solution is a solution that gives a simulated waveform close enough to reference. However, if the searching ranges or velocity limits (defined by user) for some parameters are too large, PSO may have trouble to converge. The found optimal solution might not be sufficiently accurate. In our implementation, we propose a technique to avoid that and maintain the capability of PSO to find accurate optimal solution, even in cases with large search intervals.

The proposed technique consists of re-initializing and relaunching the iteration process, with improved searching range for each parameter to optimize. This is performed when the maximum number of iterations is reached and the best fitting error (calculated using (13)) is still higher than the convergence tolerance after running PSO once. The big challenge is to derive an approach to appropriately select the new and improved searching ranges.

It has been observed that during the iterative process, the search space covered by particles evolves. After some iterations, the search space contains the ideal solution, and the algorithm simply needs more iterations to reach that solution. However, due to the wide search range for each parameter and due to the relatively high velocity limits, that ideal solution may be overjumped and never reached. Therefore, we propose to use the search space covered at the last iteration as the new search space. That space is defined by a set of search intervals (one for each parameter). The updated search interval for each parameter *k* (initially defined in (2)) becomes:

$$x_{j,k}(t) \in \left[x_k^{new\min}, x_k^{new\max} \right] \quad (14)$$

where $x_k^{new\min}$ and $x_k^{new\max}$ are the new lower and upper limits of the *k*-th parameter to optimize.

$$x_k^{new\min} = \min\left\{ x_{j,k}(N_{\max iter}) \right\}_{j\in\{1,\ldots,n\}} \quad (15)$$

$$x_k^{new\max} = \max\left\{ x_{j,k}(N_{\max iter}) \right\}_{j\in\{1,\ldots,n\}} \quad (16)$$

$x_{j,k}(N_{\max iter})$ is the value of the *k*-th parameter in the *j*-th candidate solution at the last iteration number $N_{\max iter}$.

To improve the likelihood of the new search space to contain the ideal solution, this space is expanded to include the best particle candidates found so far. In our implementation, as PSO is run using *n* particles, the new search space is expanded to include the *n* best candidates found so far. That is, assuming $X_{i,k}^{best}$ is the *k*-th parameter in the *i*-th best candidate solution $\left(i \in [1,n]\right)$, for the search interval of parameter *k*, if for any *i*, $X_{i,k}^{best} < x_k^{new\min}$, then $x_k^{new\min}$ in (14) is updated to become $x_k^{new\min} = X_{i,k}^{best}$. In the same way, if $X_{i,k}^{best} > x_k^{new\max}$, then $x_k^{new\max}$ in (14) is updated to become $x_k^{new\max} = X_{i,k}^{best}$.

After updating the search intervals for each parameter, the velocity limits are recalculated using (5) with $N = 10$. Note that this new value differs from the previous one $(N = 5)$ to ensure slower displacement of particles and reduce the risk of overjumping better solution candidates. Also, the value 10 is selected in accordance with [2] where it was pointed out that optimal results are obtained with $N \in [5,10]$.

After resizing searching intervals and modifying velocity limits, the iterative loop is re-launched. This proposed technique is very efficient to reach more accurate solution candidates as will be shown in the demonstration example.

4. EXAMPLE 1: IBR PARAMETER DETERMINATION

4.1. Case Description

In this first example, the tool is used to determine the parameter values of the reactive power (Q) control loop in the model of an IBR, a PV park. The parameters must be determined so that the simulated waveform of the reactive power generated by the park model, matches the waveform of the recorded reactive power supplied by the actual plant during a step change in the reactive power reference. Fig. 1 shows the model of the park and the connected network.

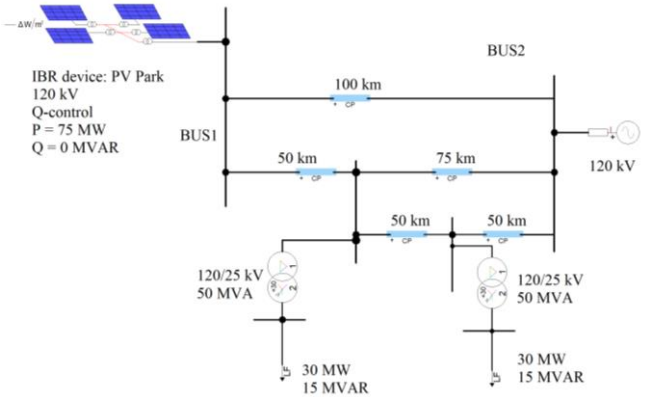

Fig. 1. PV park connected to a grid, EMTP model.

The searched parameter values are the proportional and integral constants K_p and K_i in the reactive power control loop, as shown in the PV park device mask (see Fig. 2).

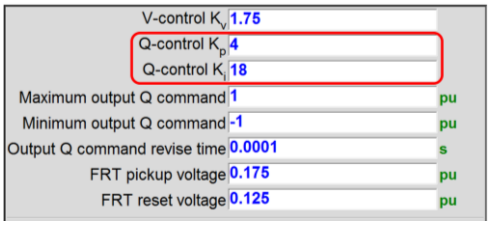

Fig. 2. Q-control parameters for the PV park model

The selection of K_p and K_i as optimization parameters is due to the structure of the controller in the software. However, the tool can work just as well in case of a different implementation structure (user would simply need to select all

parameters that have impact on reactive power response).

A step change is applied to the reactive power reference signal in the actual park. The reactive power generated by the park is recorded and shown in Fig. 3 (in blue). An initial setting (prediction) of the Q-control loop parameters (using values shown in Fig. 2) yields the model response shown in Fig. 3 (in red). The goal is to find the values of K_p and K_i to enter in Fig. 2, so that the red plot fits the blue one.

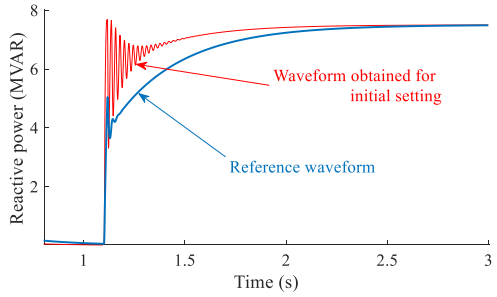

Fig. 3 Park reactive power: reference waveform in blue (from measurements on the real park) and simulated waveform (with initial values) in red.

To set up the EMTP optimization tool for this example, K_p and K_i are selected from the list of model parameters (see Fig. 4). The searching range is set to $[0, 20]$ for each parameter, as prescribed by (2).

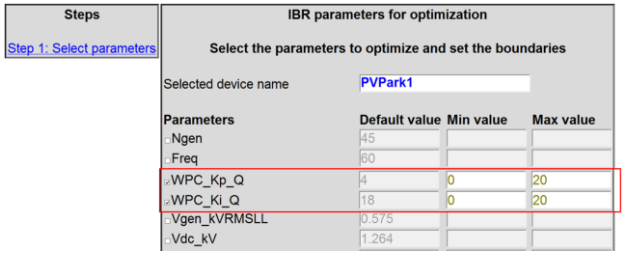

Fig. 4. Parameter selection window in the tool.

The observation interval is set to $[1, 2.6]$ in Fig. 5. This interval corresponds to the interval inside which the transient state is observed in the reference waveform (see Fig. 3).

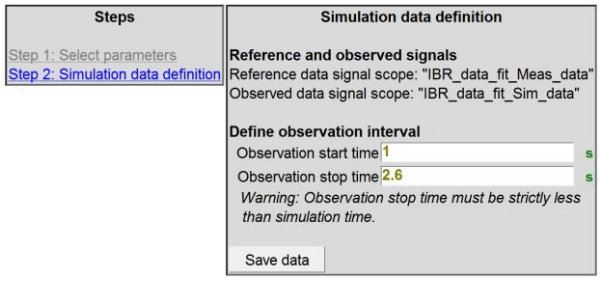

Fig. 5. Simulation data definition window in the tool

The optimization uses 10 particles with settings in Fig. 6.

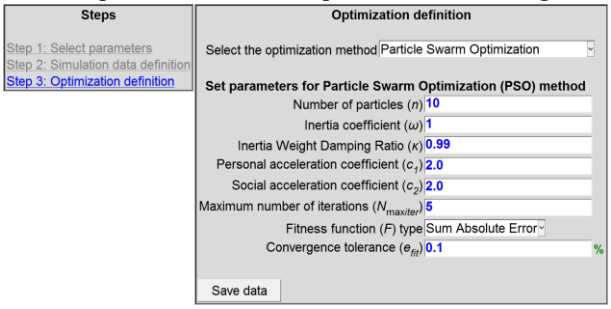

Fig. 6. Optimization definition window in the tool

4.2. Optimization Results

Once the settings are completed in the tool, the optimization process begins. Optimization starts by initializing the 10 particles with random values (within search intervals) for searched parameters. For each particle, a simulation is run, and the fitness value is evaluated. Then each particle position is updated in the iterative process described in Section 2.2 At the end of the process, the optimal particle candidate found by the tool has the following parameter values: $K_p = 1.26929$ and $K_i = 6.92825$. The final fitting error is 0.879%.

The simulated waveform for found optimal parameters, alongside with the reference is now shown in Fig. 7.

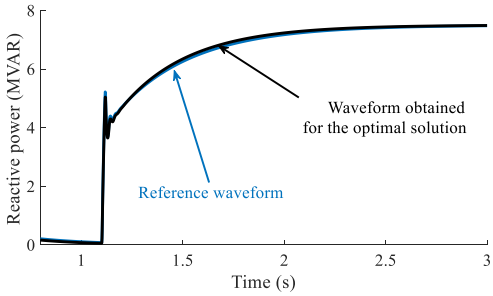

Fig. 7. Reference and optimal waveforms for park reactive power

Fig. 7 clearly shows that the tool can achieve an excellent match between the model and measurements. The total computing time was around 10 minutes for the 50 simulation runs (10 particles and 5 iterations). Time-step and simulation time in each run were respectively set to $100\,\mu s$ and 3seconds. This overall computing time is reasonable and could have been much higher if a random searching approach was used. In addition to that, a random searching approach might not find an acceptable solution.

Simulation results, not shown here, reveal that the use of more particles generally yields better optimization results. This is because more particles lead to more explorations in the search space and higher probability for catching better solution candidates. In contrary, the increase of the total number of iterations does not generally yield better results. In some cases, after few iterations, the particles fly within a search space that does contain better solution candidates, but the particles overjump these solutions. This is due to fixed and relatively large search interval and velocity limits for each parameter. This observation has motivated the development of the proposed technique to improve efficiency in PSO, as described in 3.5 The next example demonstrates the effectiveness of this technique.

5. Example 2: Synchronous Machine Exciter Parameter Determination

5.1. Case Description

In this second example, the tool is used to determine parameter values in the exciter model of a synchronous machine. The parameters must be determined so that the time-domain waveform of the excitation field voltage signal matches the waveform of the same signal recorded in the actual synchronous machine during a step change in reference

voltage. Contrary to the first example, in this example the range of variation for one of the searched parameters is quite wide. This example will demonstrate the effectiveness of the proposed technique to improve the efficiency of the PSO method in case of wide search ranges.

The circuit for this case is given in Fig. 8. The synchronous machine SM1, its exciter SEXS_1 and the connected load Load1, are modeled in EMTP. The goal is to determine the values of the gain K and time constant T_E in the exciter model. These parameters are available from the exciter device mask as shown in Fig. 9.

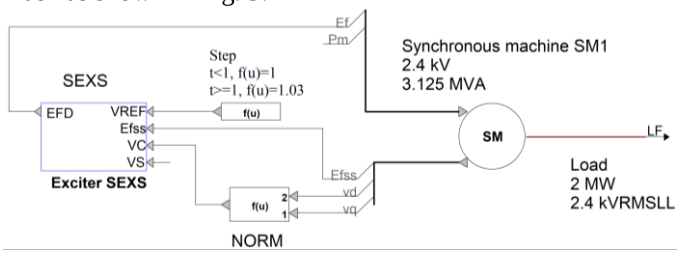

Fig. 8. Synchronous machine connected to a load

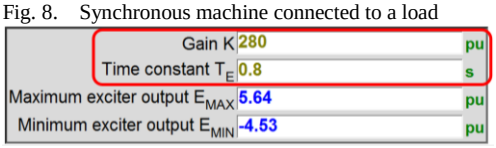

Fig. 9. Exciter parameters

The reference voltage input of the exciter is changed from 1 to 1.03 pu at the 1 s time-point, in the actual system and in the model. The excitation field voltage of the actual exciter is shown in Fig. 10 (in blue). The excitation field voltage generated by the exciter in the model with standard initial values of Fig. 9 is also shown in Fig. 10 (in red).

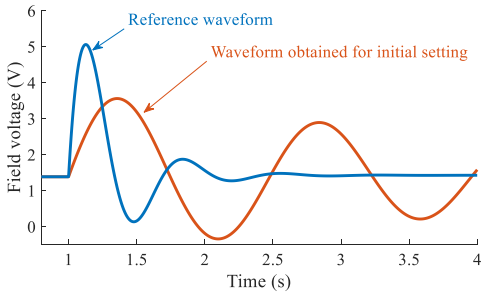

Fig. 10. Excitation field voltage

The tool is launched and PSO optimization is run with the settings shown in Fig. 6. The search intervals are $[100, 300]$ for the gain K and $[0,1]$ for the time constant T_E.

5.2. Optimization results and Discussion

After 5 iterations in the PSO algorithm, the first optimal solution is found: $K = 167.74$ and $T_E = 0.0734$ with a fitting error of 7.35%. As the fitting error is higher than the convergence tolerance, the technique proposed in this paper (see section 3.5) to improve efficiency is triggered. The technique reduces the search interval to $[142.854, 270.197]$ for K and $[0.000, 0.480]$ for T_E. Then the iteration loop of PSO is restarted. After 5 more iterations, the second optimal solution is: $K = 194.743$ and $T_E = 0.0968$ with a fitting error of 4.029%. Fig. 11 shows the waveforms of the field voltage for the first and second optimization solutions.

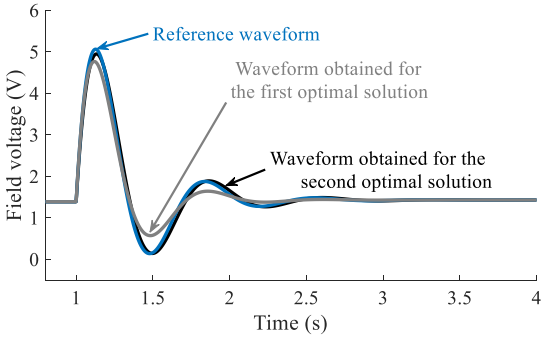

Fig. 11. Excitation field voltage waveform obtained after optimization

The first optimal solution (in gray) gives a result quite close to the reference (when compared to the initial setting of Fig. 10). However, the match is not good enough. Implementing the proposed technique yields a second optimal solution with a better match (see black plot in Fig. 11). This proves the efficiency of the proposed technique.

It is worth to point out that in this case, the proposed technique was applied only once. It is possible to apply it several times to improve accuracy of the final solution and therefore the quality of the match. However, one must keep in mind that each time this approach is applied, the iterative loop is re-executed in full, increasing computing time.

6. Conclusions

This paper presented a tool developed in EMTP and based on the PSO algorithm to determine values of unknown parameters in component models. Parameters are determined so that the time-domain response of the model matches the response of the actual component in the field. This work is within the digital-twin concept. The specifics of the tool include appropriate selection of PSO parameters, adequate definition of a fitness function and implementation of a technique to improve PSO efficiency. Demonstration examples have shown that the proposed tool can achieve very accurate parameter values to reproduce measured events. The performance of the PSO algorithm is generally jeopardized when the search ranges are too large, but in the presented implementation, the performance is not affected.

Acknowledgments

This work was supported by PGSTech company in Montreal, Canada.

References

[1] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian, and L. Gérin-Lajoie, "On a new approach for the simulation of transients in power systems," *Electr. Power Syst. Res.*, vol. 77, no. 11, pp. 1514-1520, Sep. 2007.

[2] M. A. Abido, "Optimal design of power-system stabilizers using particle swarm optimization," in *IEEE Transactions on Energy Conversion*, vol. 17, no. 3, pp. 406-413, Sept. 2002, doi: 10.1109/TEC.2002.801992.

[3] A. GoIe, S. Filizadeh, R. Menzies and P. Wilson, "Optimization-enabled electromagnetic transient simulation," IEEE Power Engineering Society

General Meeting, 2004., Denver, CO, USA, 2004, pp. 1133 Vol.1-, doi: 10.1109/PES.2004.1373019.

[4] A. M. Gole, S. Filizadeh and P. L. Wilson, "Inclusion of robustness into design using optimization-enabled transient simulation," in IEEE Transactions on Power Delivery, vol. 20, no. 3, pp. 1991-1997, July 2005, doi: 10.1109/TPWRD.2005.848722.

[5] B. Poudel, B. Bhandari, E. Amiri, P. Rastgoufard, T. E. Field and R. A. McCanne, "Interconnection Study and Optimization of Grid Connected Photovoltaic System Using Electromagnetic Transient Program," *2021 IEEE Kansas Power and Energy Conference (KPEC)*, 2021, pp. 1-6, doi: 10.1109/KPEC51835.2021.9446233.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of *ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[7] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 *IEEE International Conference on Evolutionary Computation Proceedings*. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 1998, pp. 69-73, doi: 10.1109/ICEC.1998.699146.

[8] J. Kennedy, "The particle swarm: social adaptation of knowledge," *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, 1997, pp. 303-308, doi: 10.1109/ICEC.1997.592326.

[9] Kennedy, J., Eberhart, R.C. *Swarm Intelligence*. Morgan Kaufmann, 2001, ISBN 978-1-55860-595-4.

[10] Poli, R. (2007). "An analysis of publications on particle swarm optimization applications". *Technical Report CSM-469*. Archived from the original on 2011-07-16. Retrieved 2010-05-03.

[11] Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimization". *Journal of Artificial Evolution and Applications*. 2008: 1–10. doi:10.1155/2008/685175.

[12] Mohammad Reza Bonyadi, Zbigniew Michalewicz. "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review", *Evol Comput* 2017; 25 (1): 1–54. doi: https://doi.org/10.1162/EVCO_r_00180.

[13] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization, " in *Proc. 7th Ann. Conf. Evolutionary Program., Mar*. 1998, pp. 591–600.

[14] Y. He, W. Jin Ma and J. Ping Zhang, "The Parameters Selection of PSO Algorithm influencing On performance of Fault Diagnosis," in *MATEC Web of Conferences* 63, 02019, 2016.

[15] J. Barrera, O. Alvarez-Bajo, J. J. Flores, C. A. Coello Coello, "Limiting the Velocity in the Particle Swarm Optimization Algorithm," *Computación y Sistemas*, Vol. 20, No. 4, 2016, pp. 635–645 doi: 10.13053/CyS-20-4-2505.

[16] Eberhart, R. C. & Shi, Y. (2000). "Comparing inertia weights and constriction factors in particle swarm optimization, " *Proceedings of the 2000 Congress on Evolutionary Computation* (CEC ’00), volume 1, pp. 84–88.

[17] Clerc, M. & Kennedy, J. (2002). "The particle swarm - explosion, stability, and convergence in a multi-dimensional complex space, " *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 58–73.