01 Apr 2007

A Fuzzy-PSO Based Controller for a Grid Independent Photovoltaic System

Richard L. Welch

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Recommended Citation

R. L. Welch and G. K. Venayagamoorthy, "A Fuzzy-PSO Based Controller for a Grid Independent Photovoltaic System," *Proceedings of the IEEE Swarm Intelligence Symposium, 2007*, Institute of Electrical and Electronics Engineers (IEEE), Apr 2007.
The definitive version is available at https://doi.org/10.1109/SIS.2007.367942

Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

A Fuzzy-PSO Based Controller for a Grid Independent Photovoltaic System

Richard Welch, *Student Member, IEEE*, and Ganesh K. Venayagamoorthy, *Senior Member, IEEE*
Real-Time Power and Intelligent Systems (RTPIS) Laboratory
Department of Electrical and Computer Engineering
University of Missouri – Rolla, MO 65409.
rwelch@ieee.org & gkumar@ieee.org

Abstract – This paper presents a particle swarm optimization (PSO) method for optimizing a fuzzy logic controller (FLC) for a photovoltaic (PV) grid independent system consisting of a PV collector array, a storage battery, and loads (critical and non-critical loads). PSO is used to optimize both the membership functions and the rule set in the design of the FLC. Optimizing the PV system controller yields improved performance, allowing the system to meet more of the loads and keep a higher average state of battery charge. Potential benefits of a optimized controller include lower costs through smaller system sizing and a longer battery life.

I. INTRODUCTION

As the cost of fossil fuels continues to rise, the cost of electricity generated by traditional means also increases. However, as technology and manufacturing processes improve the cost of alternative energy sources such as solar and wind energy is decreasing [1]. This rising cost of traditional energy sources and lowered cost of renewable energy is driving demand growth for renewable energy to unprecedented levels. Even so, the difference in cost of electricity generated by wind (and especially solar) and that generated by the conventional method is not insignificant, thus making some optimal control of a renewable energy source a good way to make the overall system more economical. By using a smart or optimal controller, more of the load can be met than by using a traditional controller with the same sized system.

Traditionally, PV system controllers have been simple devices that do not assign priority to various loads. Instead, they attempt to power all loads all of the time, and if there is any excess energy, then they use that to charge the batteries. In this paper, an optimal fuzzy logic based controller is developed that prioritizes the system loads and is thus able to meet more of the critical loads than the non-critical loads. This method also keeps a higher battery state of charge in case of lengthy periods of unfavorable weather. The controller that is developed in this paper is not the only attempt at creating an optimal fuzzy logic controller. At least one previous attempt has been made [2], but it only focused on optimizing the membership functions and not the fuzzy rule set. Others have also explored fuzzy controllers for standalone PV systems [3, 4].

The rest of the paper is organized as follows. Section II describes the PV system model. Sections III, IV and V describe the PV priority controller, fuzzy logic controller design and PSO respectively. Section VI presents the results and finally, the conclusions are given in Section VII.

II. PV SYSTEM MODEL

In order to develop the optimized controller, a simulation was carried out using Matlab and data from the Total Meteorological Year 2 (TMY2) [5] database from the National Renewable Energy Laboratory (NREL) for Caribou, ME.

For this simulation, the system model shown in Fig. 1 was used to describe the interaction between components of the entire PV system.

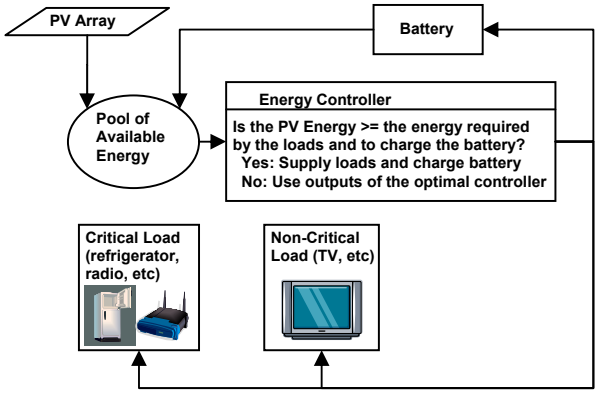

Fig. 1. Component interaction diagram.

Each of the system subcomponents was then modeled. For the battery, a simple bucket model was used. The maximum battery capacity was 34.56 kWh, and the minimum state of charge was 30%. The PV array was modeled as a simple 14.35 sq meter device with an efficiency of 11%. A plot of the output PV energy derived from the solar insolation from the TMY2 database for the Caribou, ME area is shown below in Fig. 2.

1-4244-0708-7/07/$20.00 ©2007 IEEE

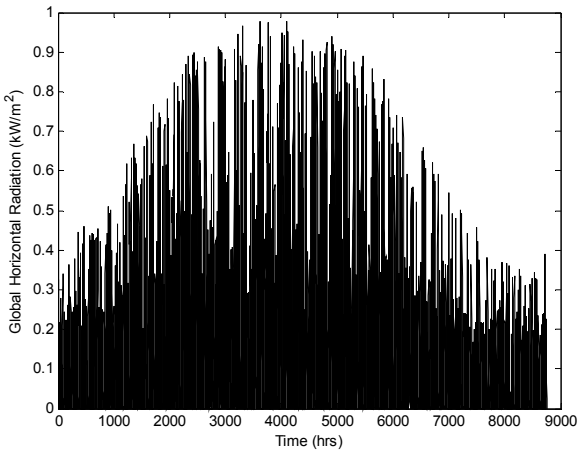

Fig. 2. Energy from PV array.

The loads were modeled using the repeating load profile shown below in Fig. 3. The critical (or base) load was taken as the constant valued load (as shown by the dotted line) whose value was a constant 0.124 kWh, and represents such items as essential lighting or small constant refrigeration loads. The non-critical load was taken as the remainder of the load and might represent the loads associated with typical morning and evening activities (television, extra lighting, etc).

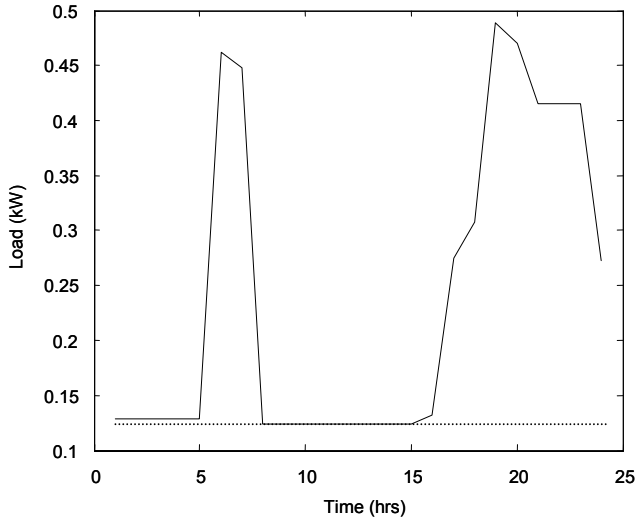

Fig. 3. Repeating daily load profile.

Each load is connected to the controller separately, and can be controlled independently. For simulation purposes, it is assumed that the loads can be switched on and off in arbitrary amounts. In practice however, loads are discrete values and would have to be turned fully on or off. The actual size of the devices used would depend on the application.

Once the system model had been finalized, the fuzzy logic controller was able to be developed. The fuzzy logic controller has 3 inputs: the energy from the PV array, the current state of charge of the battery, and the current loads. It has the following outputs: energy to the critical load, energy to the non-critical load, and energy to the battery.

The PV priority (or conventional) controller has 4 inputs: the energy from the PV array, the current state of charge of the battery, the current critical load, and the current non-critical load. It also has the following outputs: the energy to the critical load, energy to the non-critical load, and the energy to the battery.

III. PV PRIORITY CONTROLLER

The PV Priority controller is a very simple controller that is commonly deployed in conventional systems today [6]. This controller simply attempts to power all loads using energy from the PV array and if it is not able to satisfy the entire load then it uses any available energy from the batteries to completely satisfy the load. If there is more energy available from the PV array than is required by the load, then the excess PV energy is stored in the batteries.

IV. FUZZY LOGIC CONTROLLER

Fuzzy logic controller (FLC) consists of three main components:

- Fuzzification process
- Inference engine
- Defuzzification process

This can be seen in Fig. 4 below, which shows a block diagram of the fuzzy logic controller. Each of the main components is discussed below.

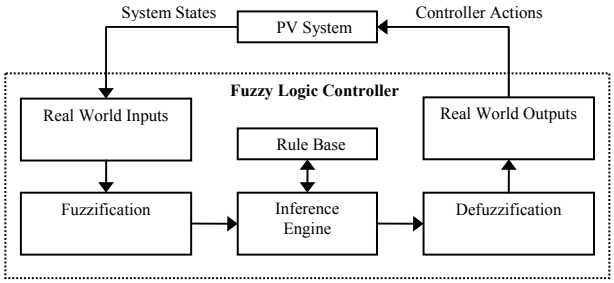

Fig. 4. Block diagram of fuzzy logic based PV controller.

Fuzzification Process

The input membership functions take the inputs to the controller (after they have been normalized by some value

suitable for the membership functions) and produce a degree of membership for each fuzzy set in the membership function. For each fuzzy set, this value is usually designated by the symbol μ. For example, the membership function shown in Fig. 5 takes as inputs the current loads and assigns to that a degree of membership for each fuzzy set in the graph. In this example, "Z" represents the "Zero" fuzzy set, "VS" represents "Very Small", "S" is "Small", "M" is "Medium", L" is "Large", and finally "VL" is "Very Large". In this case, an input of 0.5 would give the following degrees of membership for each fuzzy set:

$\mu(z)=0$, $\mu(vs)=0$, $\mu(s)\approx 0.7$, $\mu(m)\approx 0.2$, $\mu(l)=0$, $\mu(vl)=0$

For this study, all three input membership functions and two of the output membership functions used the above fuzzy sets. The third output (the output for determining the energy dispatched to the battery) used a membership function containing only five sets: "LD" for "Large Discharge", "SD" for "Small Discharge", "Z" for "Zero", "SC" for "Small Charge" and finally "LC" for "Large Charge".

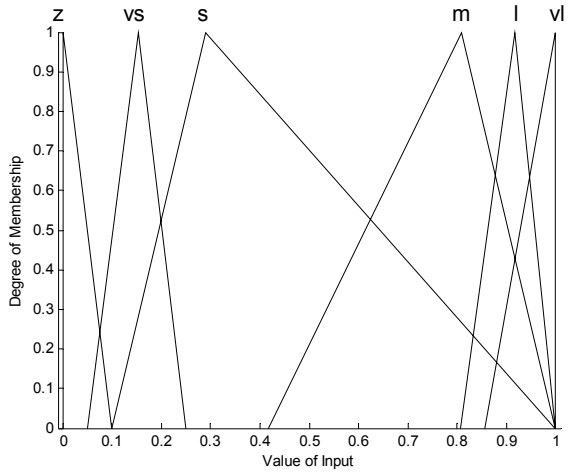

Fig. 5. Input member function for load input

Inference Engine

Once the degrees of membership for each fuzzy set have been determined for a particular input, they are presented to the inference engine. The inference engine takes these fuzzy set memberships and determines which rules should be evaluated. A typical fuzzy rule is of the form "If A, B, and C then D, E and F". As an example, one of the rules for this fuzzy controller might be: If (PV energy is "Large") and (Current state of charge of the battery is "Large") and (Current loads are "Large") then (Energy to the critical load is "Very Large") and (Energy to the non-critical load is "Medium") and (Energy to the battery is "Small Charge").

The rules to be evaluated are selected based on non-zero memberships of the input values. To extend the previous example, the shown rule would only be selected if all of the inputs had a membership value other than 0. Once this rule (and any others meeting these criteria) was found, the output degrees of membership would be asserted according to the membership's values of the inputs. Once the output degrees of membership are found, they are sent to the final stage of the fuzzy controller.

Defuzzification

Once the degrees of membership of the outputs have been found via the inference engine, the defuzzification process takes these values and translates them into an output value. This is done much like the fuzzification process but in reverse. In the defuzzification process, a trapezoid is formed using the asserted membership function, bounded below by the x-axis and bounded above by the degree of membership for that membership function. Once this trapezoid is found, its center of mass is found and wherever that point lies along the x-axis, this point is taken as the output value. In the case where multiple rules have been asserted (and hence multiple degrees of membership for the outputs), the center of mass of all of the asserted memberships' trapeziods is found (ignoring any overlapping effects, as per the "centroid" method). The value obtained is then taken as the output of the fuzzy logic controller and is then multiplied by a normalizing value to return it to the level of real world outputs.

In addition to the three main components above, a check on the outputs of the controller is made so that no excess energy was dispatched to any of the 3 outputs (i.e., the loads could not be over supplied nor could the battery be overcharged). Additionally, checks were put into place to verify that no more energy was being dispatched than was available at any given time. If more energy was being dispatched than energy available, the outputs were scaled back to meet this constraint.

V. PARTICLE SWARM OPTIMIZATION

In order to optimize the fuzzy logic controller presented in the previous section, the Particle Swarm Optimization (PSO) [7, 8, 9] technique was used to optimize both the membership functions and rules set of the inference engine. Using PSO to optimize a fuzzy logic controller has been done before [2, 9], and is an interesting way to give better performance to a fuzzy logic system.

PSO is an iterative algorithm that represents possible solutions to a given problem with a series of multidimensional vectors. Each vector is called a *particle* and contains one complete solution. Each dimension of each particle represents one parameter of a solution to be optimized. In this case, 30 particles are used and chosen to

represent each possible controller with the following parameters:

- Each fuzzy set in each membership function (besides the first and last fuzzy set in each membership function) was represented by 3 values: 1 for the left-most point, one for the right-most point, and a third for the middle point. The end values had a fixed leg (either the left or right, depending on which end of the membership function the occupied), so they were only specified by 2 values. Since there are 5 membership functions (3 inputs and 2 outputs) with 6 fuzzy sets and 1 membership function (1 output) with 5 fuzzy sets, this equates to 93 parameters just to represent the membership functions.
- Each rule was represented by 3 values (1 for each output). Since there are 3 inputs and each can take on as many as 6 values, there are 216 rules. Since each rule is represented by 3 values, this adds another 648 parameters.

Summing each of these values up, it can be seen that each solution is represented with 741 parameters, so each particle has 741 dimensions. PSO optimizes these values by using a process based on social interaction, much like a flock of birds or school of fish. In PSO, a collection of particles takes on values that represent a possible solution. As the swarm of particles moves about (according to a defined velocity determined by how well each is doing), the particles' values change. As they change, a record of each particle's best position (called *pbest*) is kept as well as the global overall best position (called *gbest*). The equations to determine velocity, and position updates are shown below in (1) to (2) respectively. In each, the index *i* ranges over the number of particles. In (1), up to 10% of the previous velocity was kept (so as to be used as momentum for the particle) and a weighting of 2 was chosen for the *pbest* and g*best* terms as it has been shown that the sum of these two weights should be about 4 for best performance. [10]

$$Velocity(i) = 0.1 * rand * Velocity(i) + 2 * rand * (pbest(i) - Position(i)) + 2 * rand * (gbest - Position(i)) \quad (1)$$

$$Position(i) = Position(i) + Velocity(i) \quad (2)$$

Equations (1) and (2) can be applied directly for continuous values (such as the membership functions), but for discrete values (such as the rules) there has to be some sort of quantization applied. In this case, a continuous valued variable was quantized to the appropriate number of levels for each rule in the PSO. For example, the output that dispatches energy to the battery can take on 5 states ranging from "Large Discharge" to "Large Charge". In this example, the continuous value ranged from -1 to 1. At each iteration when a controller is instantiated from a particle, this continuous value is quantized into 1 of 5 equal ranges, each representing one of the fuzzy sets.

The quality of solution for each particle is measured by the *fitness function* when evaluated at the particle's point. In this research, the following fitness function was used (where each term is a percentage calculated over the entire year, using a fuzzy logic controller instantiated from the current particle):

$$Fitness = (30/23) * (Critical\ Load Satisfied) + (15/30) * (Average\ Battery\ State\ of\ Charge) + (13/30) * (Non - Critical\ Load\ Satisfied) \quad (3)$$

A higher fitness function value (or just called the fitness) results from a better performing individual. As the algorithm progresses, it is expected that the best solution should continue to improve over time (which is shown by an improving fitness over time). In (3), the weights are determined by trial and error in this and past research. They are given their relative weighting in order to place a higher importance on first the critical load being satisfied, then keeping as high as possible the average battery charge and finally powering the non-critical load.

An outline of the basic overall process that was implemented for this paper is given below:

1. Set initial particles to a predetermined "naïve" solution (this solution was a "good guess" as a starting place for optimization).
2. Instantiate a solution for each particle.
3. Measure the fitness of each solution.
4. Check to see if the current fitness is better than the current *pbest* for each particle; if it is, update *pbest*
5. Check to see if any of the current *pbests* are better than the current *gbest*; if any is, update *gbes*t with the best one.
6. For the membership function portion (the first 93 dimensions) of the particles and then the rules portion (the last 648 dimensions), perform the following steps:
    a. Update the velocity of each particle and perform limit checks.
    b. Update the position of each particle according to the particle's velocity and perform limit checks and other rules.
7. Measure the fitness of each solution.
8. Check to see how many iterations have been performed and how well the best solution is performing. If the limits have been reached, then quit. Otherwise, loop back to step #4.

One addition to the standard PSO algorithm that was made was to add some limit checks on the instantiated individual, and its velocity through space. In the case of the instantiated individual, checks were also put in place to make sure that no

entry in any membership function spanned a distance of less than 0.1. If this case was found, then the membership function entry was widened to a width of 0.1. Also, some checks were put into place to verify that the entire width of the input space was mapped to a fuzzy set. That is to say that no possible input could fall outside of a membership function entry. This was implemented by assigning the minimum membership function to the lowest value and the maximum membership function to the upper value. Then it was verified that all remaining membership functions overlapped, thus spanning the entire width of the inputs space.

VI. SIMULATION RESULTS

The results from this study are encouraging, and surpass previous attempts at optimizing a fuzzy logic controller [2]. After running the PSO optimization for 50 iterations, the final gbest fitness is 2.086250. This can be seen from Fig. 6 showing how the best particle's (gbest) fitness increased throughout the simulation:

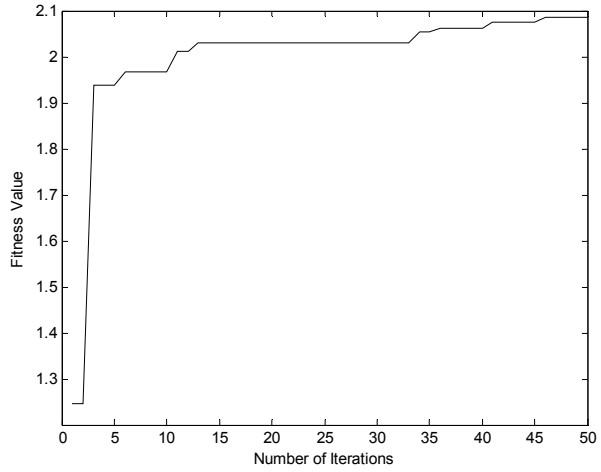

Fig. 6. Fitness of gbest particle over the entire simulation

The actual performance (as well as the performances of the PV priority and un-optimized (naïve) fuzzy logic controllers) is listed below in Table I. The *Total Score* row was calculated by evaluating the fitness function above using the 3 measures of performance (energy to the critical load, energy to the non-critical load, and energy to the battery). This gives an objective method of comparing controller performance.

These results show that optimizing the membership functions as well as the rule set allows the fuzzy logic controller to perform far better than the un-optimized fuzzy controller or even just optimizing the membership functions alone. Additionally, Figs. 7-9 show the membership functions for the naïve fuzzy logic controller before optimization for each of the inputs and outputs (the first graph actually represents the first 3 inputs, and the second represents the first 2 outputs, since they were initially all the same). Figs. 10 to 15 represent the optimized membership functions after application of the PSO algorithm. Finally, Table II shows the first 30 (of 216) rules before and after optimization via PSO (only the consequents are modified). It can be seen that the frequency of rule changes increases as the rule number increases. More modifications are observed in the full set of 216 rules.

While performance is increased using PSO to optimize the original FLC, it can be seen that many of the optimized membership functions have 1 or 2 fuzzy sets which span the entire membership function domain. In these cases, it may be possible to reduce the number of fuzzy sets used, thus reducing the dimensions needed for the PSO optimization.

VII. CONCLUSIONS

The design of an optimal fuzzy logic controller for a grid independent photovoltaic system has been presented using particle swarm optimization. PSO is able to optimize the membership functions and develop optimal rules for FLC. Results show more of the critical loads are met most of the time (95.51%). Additionally, the average state of charge of the battery is also kept at a higher level in case of time of extended poor solar insolation (75.3%). As a result, it is possible that a smaller (and cheaper) overall PV system utilizing such an optimal controller would be suitable for meeting the same loads as a larger, more expensive system not using an optimal controller.

TABLE I.

SUMMARY OF CONTROLLER PERFORMANCE SHOWING BOTH PERCENTAGE OF LOADS MET AND ACTUAL VALUES

	PV Priority	**Naïve Fuzzy**	**Optimal Fuzzy**
Percentage Critical Load Met	84.22 [914.80 kWh]	93.04 [1011.00 kWh]	95.51 [1038.00 kWh]
Percentage Non-Critical Load Met	77.21 [778.40 kWh]	32.16 [324.20 kWh]	61.8 [623.00 kWh]
Average Battery Charge	63.87 [22.07 kWh]	76.58 [26.47 kWh]	75.3 [26.02 kWh]
Total Score	1.95	1.89	2.09

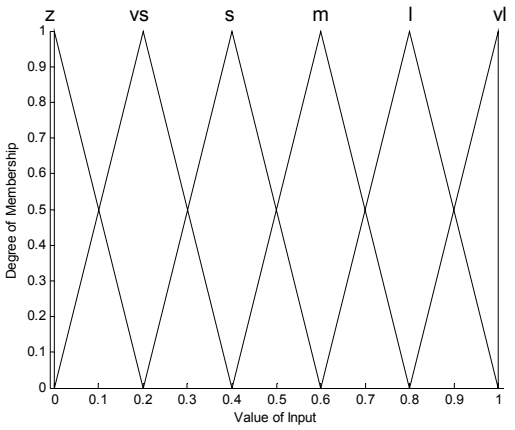

Fig. 7. Membership function for all three inputs: PV, PV energy, and current state of battery charge, as well as current load.

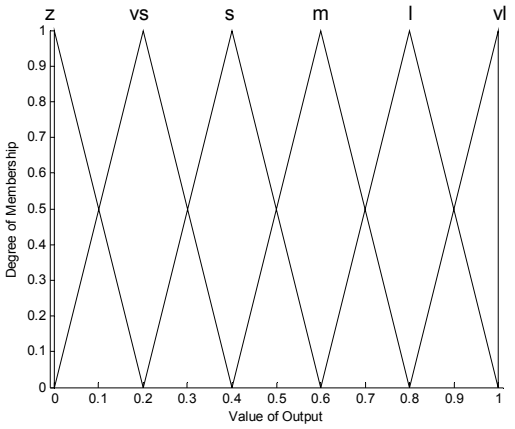

Fig. 8. Membership function for first two outputs: energy to critical load and energy to non-critical load.

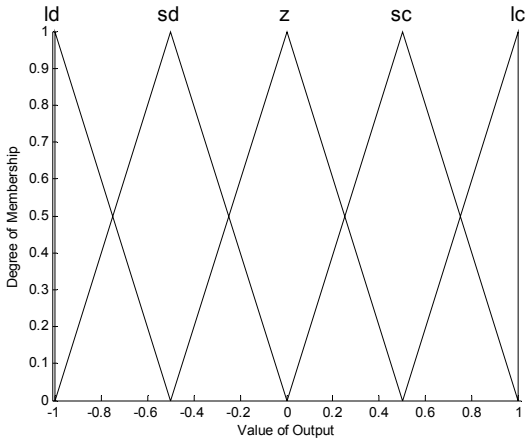

Fig. 9. Membership function of output for dispatching energy to the battery.

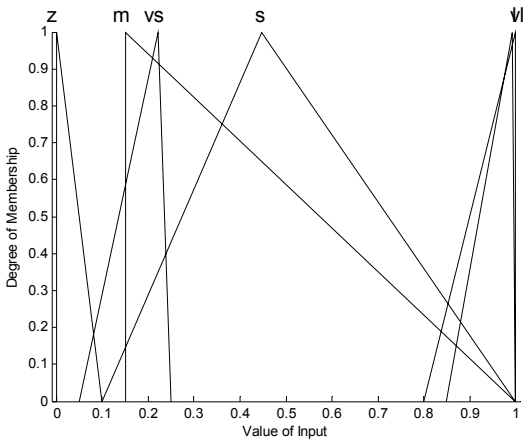

Fig. 10. Optimized membership function for PV energy input.

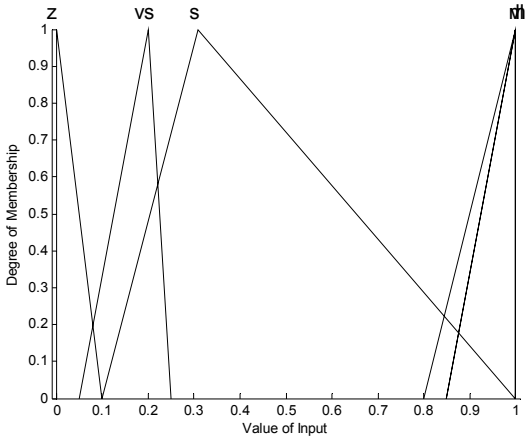

Fig. 11. Optimized membership function for state of battery charge input.

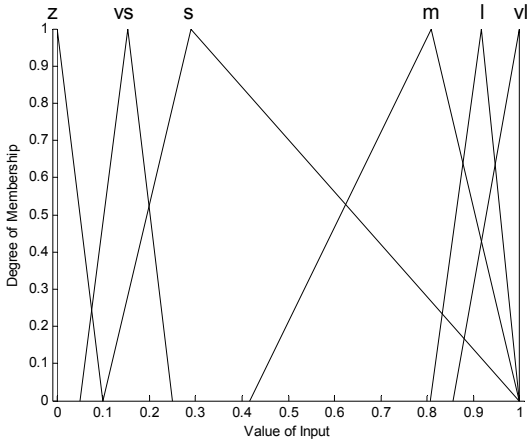

Fig. 12. Optimized membership function for current load input.

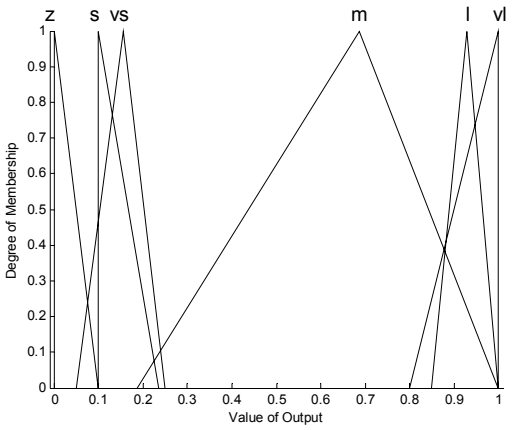

Fig. 13. Optimized membership function for energy to critical load output.

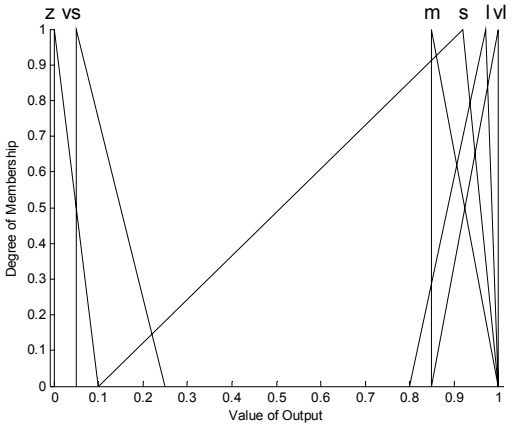

Fig. 14. Optimized membership function for non-critical load output.

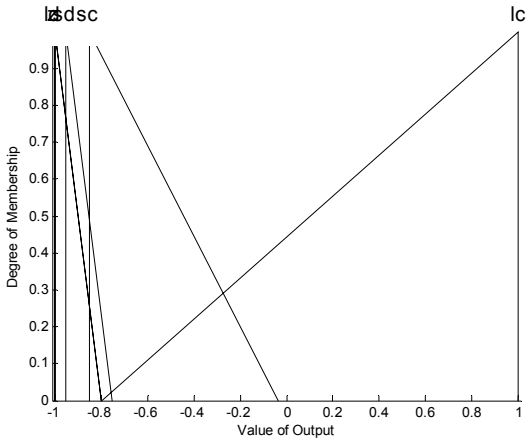

Fig. 15. Optimized membership function for energy to the battery output.

REFERENCES

[1] R. A. Messenger, J. Ventre, Photovoltaic System Engineering, CRC Press, 2004.

[2] R. L. Welch, G. K. Venayagamoorthy, "Comparison of Two Optimal Control Strategies for a Grid Independent Photovoltaic System", *IEEE Industry Applications Society*, October 2006.

[3] J.A. Momoh, A.R. Ofoli, "Load management and control of the photovoltaic (PV) system using fuzzy logic", *Large Engineering Systems Conference on Power Engineering (LESCOPE)*, pp. 184-188, 2001.

[4] A. Moreno, J. Julve, S. Silvestre and L. Castaner, "A Fuzzy Logic Controller For Stand Alone PV Systems", Photovoltaic Specialists Conference, 2000. Conference Record of the Twenty-Eighth IEEE

[5] "TMY2 User's Manual", Jun. 1995. National Renewable Energy Laboratory, Golden, Colorado. [Online] Available: http://rredc.nrel.gov/solar/old_data/nsrdb/tmy2/

[6] G. P. Henze, R. H. Dodier, "Adaptive Optimal Control of a Grid-Independent PV-System", *Trans. ASME Journal of Solar Energy Engineering*, vol. 125, Feb. 2003, pp. 34 – 42.

[7] A. P. Engelbrecht, *Computational Intelligenc An Introduction*, Wiley, 2003.

[8] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Morgan Kauffman Publishers, San Franscisco, CA. ISBN 1-55860-595-9, 2001.

[9] S. Doctor, V. K. Venayagamoorthy, "Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller", *IEEE Industry Applications Conference*, vol. 2, pp. 1200-1206, Oct. 2004.

[10] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation, vol 6, issue 1, pp. 58-73, Feb. 2002.

TABLE II.

FIRST THIRTY RULES OF RULE BASE BEFORE AND AFTER PSO OPTIMIZATION

If			Then		
PV Energy	**Battery State of Charge**	**Load**	**Energy to Critical Load**	**Energy to Non-Critical Load**	**Energy to the Battery**
z	z	z	z→z	z→z	z→z
z	z	vs	z→z	z→z	z→z
z	z	s	z→z	z→z	z→z
z	z	m	z→z	z→z	z→z
z	z	l	z→z	z→z	z→z
z	z	vl	vs→vs	z→z	z→z
z	vs	z	z→z	z→z	z→z
z	vs	vs	**vl→m**	z→z	sd→sd
z	vs	s	**vl→m**	z→z	sd→sd
z	vs	m	**vl→m**	z→z	sd→sd
z	vs	l	**vl→m**	z→z	sd→sd
z	vs	vl	**vl→m**	z→z	sd→sd
z	s	z	**z→m**	z→z	**z→sd**
z	s	vs	**vl→z**	z→z	**sd→z**
z	s	s	**vl→m**	z→z	sd→sd
z	s	m	**vl→m**	z→z	sd→sd
z	s	l	**vl→m**	z→z	sd→sd
z	s	vl	**vl→m**	z→z	sd→sd
z	m	z	**z→vl**	z→z	**z→sd**
z	m	vs	**vl→z**	z→z	**sd→z**
z	m	s	**vl→m**	z→z	sd→sd
z	m	m	**vl→m**	**vs→z**	sd→sd
z	m	l	**vl→m**	**s→vs**	sd→sd
z	m	vl	**vl→m**	s→s	sd→sd
z	l	z	**z→m**	**z→vl**	**z→sd**
z	l	vs	**vl→z**	z→z	**sd→z**
z	l	s	**vl→m**	z→z	sd→sd
z	l	m	**vl→m**	z→z	sd→sd
z	l	l	**vl→m**	**vs→z**	sd→sd
z	l	vl	**vl→m**	**s→vs**	sd→sd