Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

EA006 – TRABALHO DE FIM DE CURSO

Development of a graphical user interface tool for modelling and simulation of photovoltaic modules

Aluno:
Julen Lizarrondo Ostiz
Engenharia Elétrica-Estudante Especial
RA: 161321

Orientador:
Prof. Dr. Marcelo G. Villalva
FEEC/DSE

Campinas
07/05/2015

INDEX

1.SUMMARY

The main of this work is to develop a mathematical program to model and simulate photovoltaic modules. The tool will be provided with a graphical interface allowing the usage by users with little knowledge and without having a high-level program installed on their computers

In the next figure one can see the graphical interface of the program designed:

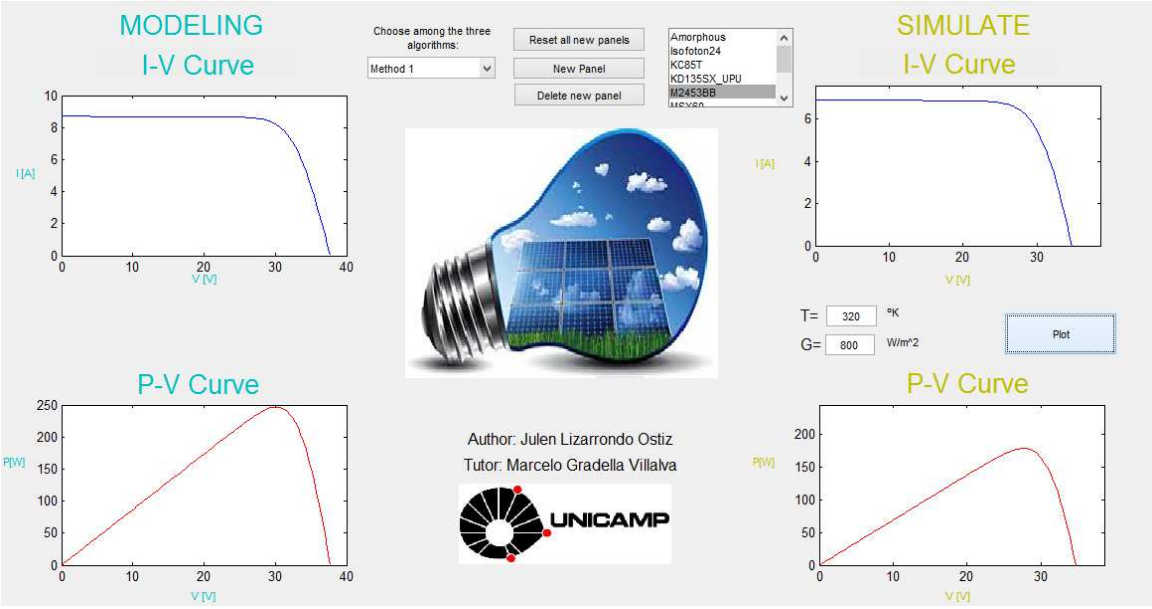

Figure 1. Graphical interface of the program designed**.**

To begin the program, the user must select a method among the three available that appear on the screen. Once a method is selected, the user must select a photovoltaic panel, so when the user clicks, two graphs appear on the left of the Figure 1. These graphics will be obtained after modelling the PV panel. From these graphs, the user gets conclusions about the modelling of the photovoltaic panel.

Then on the right side one can enter actual temperature and irradiance which will affect the panels when installed. This way one can compare two results.

There is also an option that allows the introduction of new photovoltaic panels. The interface will ask for the parameters of this new panel and the user data will be saved for modelling or simulation. Similarly, one can delete introduced new panels, pressing their set button to delete all new panels or you can press the "delete new panel" to delete only one of new panels.

2.INTRODUCTION

2.1 Introduction in renewable energies.

At present, the energy system is based on the use of a number of resources. These resources can be finite (coal, gas, oil, uranium) or infinite (renewable energy: sun, air, water).

It is clear that finite resources come a time will run out and you will become more expensive as the cost of finding or purchasing will be increasingly expensive. (Is not the same maximize fuel 100 meters underground than 1 km underground, build platforms are much more expensive).Continuing with the energy of finite resources, is the contamination. Some of the contaminants generates in the combustion hydrocarbons are responsible for numerous lung diseases, whereas others, such as carbon monoxide at high concentrations increases the likelihood of heart attack. These pollutants not only affect humans directly, they are also doing indirectly through the dreaded gases, which are already starting to show its effects on global climate (hurricanes, droughts, typhoons, floods)

By contrast, energy by infinite resources, known as renewable energy has the great advantage that these resources if they are national and not dependent on any other state to generate energy. Besides these resources are natural, so it has no cost, since the air (wind), the sun (solar, thermal) or water (hydro, tidal) are obviously resources that have not cost.

In the early nineteenth century, 95% of primary energy consumed in the world came from renewable sources. A century later, this percentage was 38%, and at the beginning of this century was only 16% [1]. However, the trend seems to be changing, as in many industrialized countries the proportion of renewable energy has grown considerably in the last two decades. Total worldwide investment in renewable energy in 2004 was 22.000 billions of dollars. In 2008 this number increased to 130.000 billion in 2009 to 160.000 billion and in2010 to 211.000 billion of dollars. In 2011 at least 118 countries support renewable energies with some kind of support or subvention. You'll data is well above the 55 countries that in some way supported the renewable energy in 2005.[2]

Renewable energies have been replaced to some extent in the market, nuclear energy and fossil fuels in four distinct markets such as power generation, thermal applications, fuel for transport and energy services without network connections in areas with difficult access.

The extensive interest of countries for this type of energy is caused by reducing of green house gases and other pollution substances, besides decreasing the energetic dependence and contribute to employment generation wing and energetic development.

The increasing cost of energies generated from fossil elements and environmental impact are suggesting that renewable energies can have a very good reception in the market with in the production sector. Photovoltaic energy is one of the most promising renewable energy. According to the International Technology Agency (IEA), in 2030, 5% of world electricity consumption will be created from photovoltaic panels and will grow to 11 % in 2050 [3]. Besides the cost of complete photovoltaic systems in the last six years, has been cut by 5 and the photovoltaic modules in 3. So, with the development of technology, solar energy will be very important the next few years. In the figure below we can see the global production of energy generated by solar panels and the prediction for the coming years.[4]

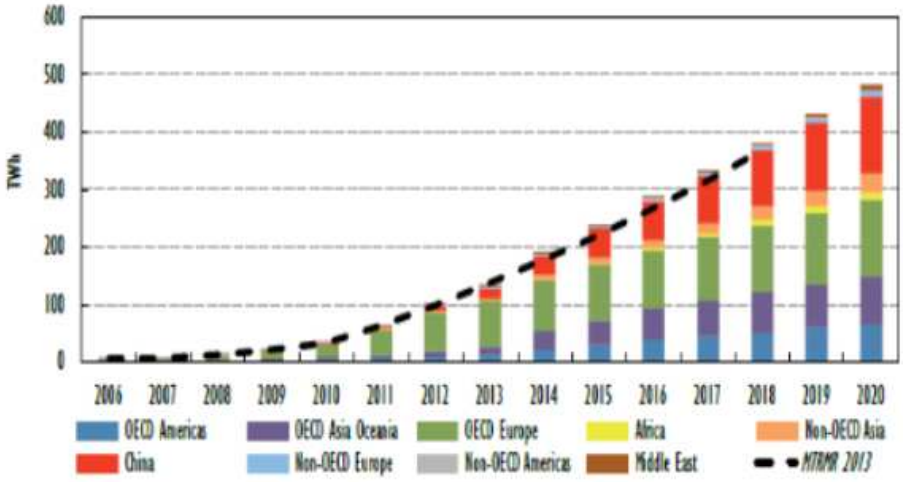

Figure 1. Solar PV generation and projection by region.

2.2 Introduction to photovoltaic panels

A solar cell is a device which converts solar energy into electrical energy through the photovoltaic effect. This is a device in which their components change when exposed to sunlight, change its resistors, its intensity and its voltage. Here one has some definitions to understand better the construction of PV strings.

- The photovoltaic module is a set of solar cells, normally connected in series.
- A photovoltaic panel is a set of modules connected together and mounted on a support.
- PV strings, is normally a set of panels. If we connect this PV string in parallel we get increased power.

Each photovoltaic module comprises a bypass diode placed in parallel with the module. This diode is triggered if the current generated is less than the current generated by other modules. Normally all modules connected in series and it try that all modules generate the same current value and if anyone generates less, the diode start working and eliminate momentarily this module. If we have several PV strings connected in parallel, diodes are also placed in parallel for the same reason.

The aim of these photovoltaic strings with bypass diodes, is to create the maximum possible power, so, we want to have high values of intensity and tension because:

$$P = V*I \qquad [2.1]$$

So if one increases the intensity of the tension, we can increase the power but we must be careful to not override any value of two. Figure 2 and figure 3 illustrate the behaviour of homogeneous PV array. V_{mpp} and I_{mpp} represent the Voltage and the current when the panels work at the maximum power point (MPP). These are the best point to work with the panels. One has other points, for example, $(0,I_{sc})$, or $(V_{0c},0)$ but that point we are not interested because poweris0. I_{sc} and V_{oc} we used to denominate the short circuit current and open circuit voltage.

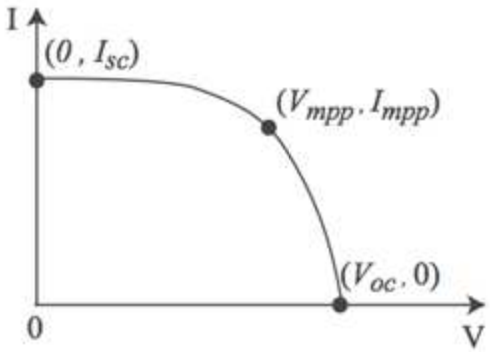

Figure 2. Current-voltage curve.

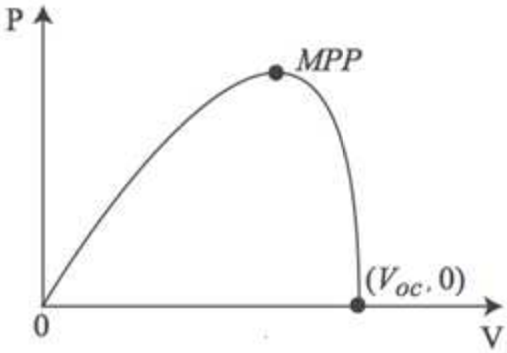

Figure 3. power-voltage curve.

The IV and PV curves of Figures 2 and 3 depend on the temperature and irradiance to which undergo. As seen in Figure 4,the irradiance is a fundamental aspect in the generation of power. By seeing the graph one can deduce that the value of the current may be proportional to the change of irradiance.

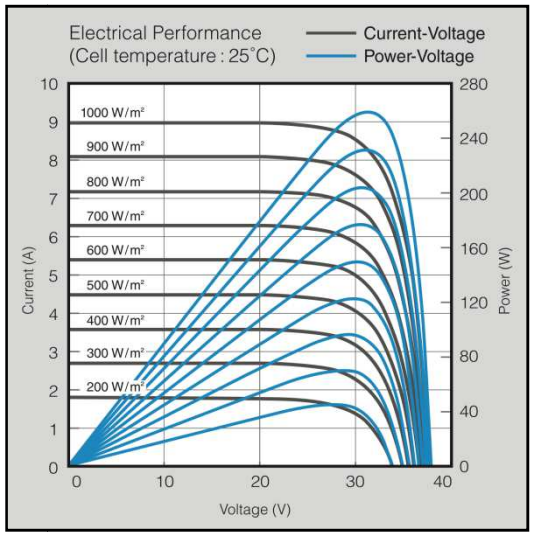

Figure 4.I-V, P-V, for different values of irradiance. Source: Mitsubishi Electric.

3. OBJECTIVES

The objectives of this work are to study the theory of photovoltaic devices and learn the techniques for the extraction the parameters of these devices. The aim is to create a computational tool with graphical interface for the extraction of commercial PV model parameters using the information we can subtract from commercial catalogues.

Once created this interface, the intention is to create an executable file to distribute this program to any computer without need for any type of special program installed.

This work was develop along the period of nine months with the following monthly tasks:

1. Study the theory of photovoltaic devices
2. Study methods and algorithms for obtaining parameters for photovoltaic devices
3. Study methods to simulate the photovoltaic devices
4. Create a high level program using the Matlab language to model and simulate the photovoltaic device
5. Elaborate a graphic interface for the program created
6. Elaborate a data bank with models of most commercial photovoltaic panels
7. Elaborate an executable file to distribute the program
8. Write the final work

Table 1. Schedule of tasks during the project development.

	2014				2015				
	September	October	November	December	January	February	March	April	May
1	x								
2	x								
3		x							
4		x	x						
5			x	x		x			
6						x	x		
7							x		
8								x	

4. METHODOLOGY

In the preparation of this work we have used three different methods to model the photovoltaic panels. The first model is "The Double-Diode and Single-Diode Models". Its name in interface is Method 1 or PVM1. The second model is the same model assuming that R_p is ∞ (a very big resistance). Its name in interface is Method 2 or PVM2. And the last model is same but with a explicit solution using Lambert W function. Its name in interface is Method 3 or PVM3.[4] These methods are to model the photovoltaic panels. To simulate panels, we use a program called evaluation.

To use these methods, in all cases, we need to have some parameters of photovoltaic panels that we want to model.

Panels information:

- **Iscn**: Nominal short-circuit voltage [A]
- **Vocn:** Nominal array open-circuit voltage [V]
- **Imp**: Array current maximum power point [A]
- **Vmp**: Array voltage @ maximum power point [V]
- **Pmax_e** = Vmp*Imp;
- **Kv**: Voltage/temperature coefficient [V/K]
- **Ki**: Current/temperature coefficient [A/K]
- **Ns**: Number of series cells

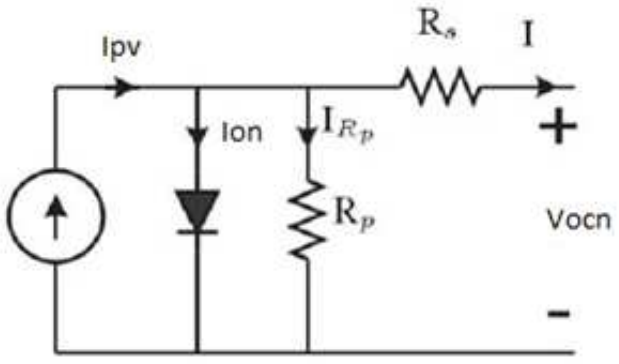

Figure 5. Single-diode model accounting for ohmic losses.

4.1 Method 1

To perform this method, a Matlab program was developed which implements several formulations that will be described below. You can see the program in the append of this work.

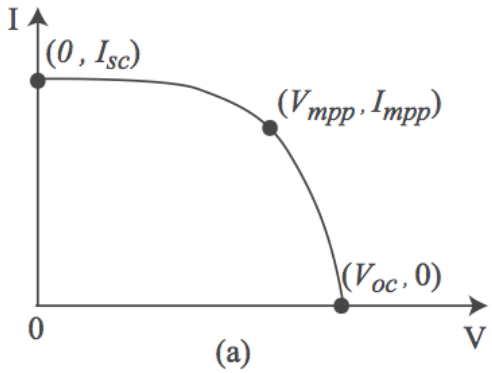

Figure 6. Current-voltage curve.

Seeing the figure4 and figure 5, it follows that the maximum resistance connected in series and the minimum resistance connected in parallel will be:

$$Rs_{max} = \frac{V_{ocn} - V_{mp}}{I_{mp}} \quad [4.1]$$

$$Rp\_min = \frac{V_{mp}}{I_{scn} - I_{mp}} - Rs\_max \quad [4.2]$$

Thermal junction voltage (nominal and current temperature) saving constant of Boltzmann (k = 1.3806503e-23 [J/K]) and Electron charge (q = 1.60217646e-19 [C]):

$$V_{tn} = \frac{k * Tn}{q} \quad [4.3]$$

$$V_t = \frac{k * T}{q} \quad [4.4]$$

The Current-Voltage curve of a PV array is obtained by subtracting a current of the diode which is not linearly dependent on the voltage from a constant current value. The ideal equivalent circuit shown in Figure3.0. Is made of a current generator in parallel with a diode. The diode takes into account the physical effects that occur in the silicon p-n junction cell. The current generator is photo-induced current which depends on the characteristics of the semiconductor material used for the cell, and especially, is linearly dependent Cell area, irradiance level, and temperature. Equation (4.5) gives the dependence in the past two exogenous variables:

$$I_{pv} = I_{pvn*\frac{G}{Gn}*[1+\propto_1*(T-T_n)]} \quad [4.5]$$

$$I = I_{pv} - I_{0n} * \left(e^{\frac{V}{Vt*a}} - 1\right) - \frac{V+I*R_s}{R_p} \quad [4.6]$$

When we calculated currents of equivalent circuit, we need to recalculate the series resistance and parallel resistance values. As shown in the appendix, all this comes within a loop in which all values are recalculated. After a large number of iterations (500.000 iterations), we obtain the values of desired Tension and desired Current to draw the corresponding graphics. In the appendix, you can see the equations and in what context they are used for obtaining the results.

4.2 Method 2

The method 2 is similar that method 1 but we assumed that the parallel resistance is a very big number, ∞, so we can neglected it and I_{Rp} will be 0. If Rp=∞ The equation of I [4.6] will be:

$$I = I_{pvn} - I_{on} * \left(e^{\frac{V}{V_{tn}*a}} - 1\right) \quad [4.7]$$

Now, we can see that the:

$$I_{pvn} = I_{sc} \quad [4.8]$$

If we have open circuit (OC):

$$0 = I_{pvn} - I_{on} * (e^{\frac{\frac{V_{ocn}}{n_s}}{a*V_{tn}}} - 1) \cong I_{pvn} - I_{on} * e^{\frac{\frac{V_{ocn}}{n_s}}{a*V_{tn}}} \quad [4.9]$$

$$V_{OCn} = -a * n_s * V_{tn} * \ln\left(\frac{I_{on}}{I_{pn}}\right) \quad [4.10]$$

The coefficient Voltage/ Temperature (Kv) we can calculated by:

$$K_v = \frac{dV_{ocn}}{dTn} = \frac{d}{dT}[n_s * a * V_{tn} * \ln(I_{pvn}) - n_s * a * V_{tn} * \ln(I_{on})] \quad [4.11]$$

Then, making the derivative,

$$K_v = \frac{V_{ocn}}{Tn} + n_s * a * V_{tn} * \left(\frac{k_i}{I_{pvn}} - \frac{3}{T_n} - \frac{E_{gap}}{k*T_n^2}\right) \quad [4.12]$$

And if we clearance the ideality factor (a),

$$a = \frac{K_v - {V_{ocn}}/{T_n}}{n_s * V_{tn} * (\frac{k_i}{I_{pvn}} - \frac{3}{T_n} - \frac{E_{gap}}{k * T_n^2})} \quad [4.13]$$

To calculate Ion, we can use (4.9) and:

$$I_{on} = I_{pvn} * e^{\frac{-E_{gap}}{a * n_s * V_{tn}}} \quad [4.14]$$

Now, we can calculate the temperature coefficient:

$$c = \frac{I_{on}}{T_n^3 * e^{\frac{E_{gap}}{K * T_n}}} \quad [4.15]$$

And the series resistance will be:

$$Rs = \frac{a * n_s * V_{tn} * \log\left(\frac{1 - I_{mp}}{I_{pv}}\right) + V_{ocn} - V_{mp}}{I_{mp}} \quad [4.16]$$

With this parameters, we can calculate the values of desired Tension and desired Current to draw the corresponding graphics as we can see in Appendix.

4.3 Method 3

In the method 3, we use the equations 4.3 , 4.4 , 4.13 , 4.14 , and 4.15 of method 2. Is similar but in this method we use parameter identification including Rp (explicit solution).

Whit these equations, to star this method, we create these variables:

$$alfa = \frac{V_{mp}*(V_{mp}-2*a*V_{tn}*n_s)}{a^2*V_{tn}^2*n_s^2} \quad [4.17]$$

$$beta = 2 * I_{mp} - I_{pv} - I_{on} \quad [4.18]$$

$$delta = a * I_{on} * V_{tn} * n_s \quad [4.19]$$

$$gama = 2 * \frac{V_{mp}}{a*V_{tn}*n_s} - \frac{V_{mp}^2}{a^2*V_{tn}^2*n_s^2} \quad [4.20]$$

We used the variables to create other variables:

$$y = \frac{V_{mp}*beta*e^{alfa}}{delta} \quad [4.21]$$

$$x = LambertW(y) + gama \quad [4.22]$$

Whit these values, we can calculate Rs and Rp:

$$Rs = \frac{x*a*V_{tn}*n_s*-V_{mp}}{I_{mp}} \quad [4.23]$$

$$Rp = \frac{x*a*V_{tn}*n_s}{I_{pv}-I_{mp}-I_{on}*(e^x-1)} \quad [4.24]$$

With this parameters, we can calculate the values of desired Tension and desired current to draw the corresponding graphics as we can see in Appendix.

4.4 Evaluation

To run the program evaluation, it is necessary run any of the three methods described above, since the calculated parameters are used. This program takes into account the temperature and irradiance field, so this program, simulates the behaviour of the panels in these conditions. Like the above methods can view the program in Appendix.

We can see the temperature and irradiation effect on the current:

$$dT=T-Tn \quad [4.25]$$

"Tn" be the nominal temperature (273.25 K) and "T" the actual temperature where we installed the panel. So,

$$I_{pvn} = \frac{Rs+Rp}{Rp} * I_{scn} \quad [4.26]$$

$$I_{pv} = \left(I_{pvn} + k_i * dT\right) * \frac{G}{Gn} \quad [4.27]$$

$$I_{sc} = (I_{scn} + k_i * dT) * \frac{G}{Gn} \quad [4.28]$$

Whit these parameters and "Io" we can calculate the values of desired Tension and desired Current to draw the corresponding graphics as we can see in Appendix.

5. RESULTS

In this paper, photovoltaic panels are modelled and simulated and we obtain IV and PV graphs as results. Using both graphs the user can see the result made after the modelling and simulation. So, the user can compare the two graphs, see its theoretical performance and compare it, to its actual operation at a given temperature and irradiation. For example, the following figure shows the modelling of method 1 for KC85Tpanel.

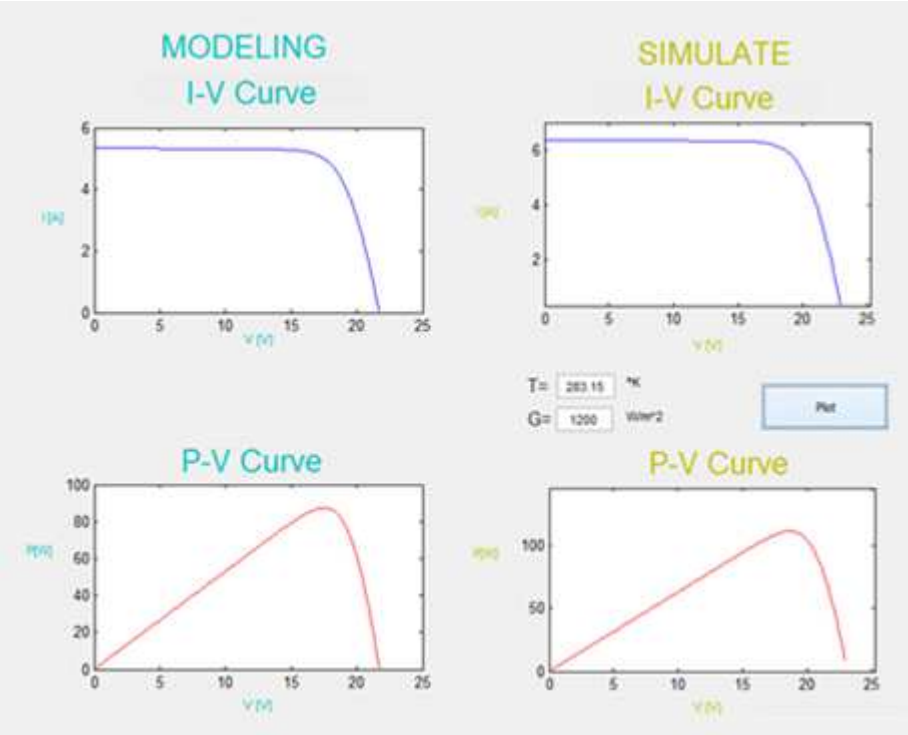

Figure 7. Results of the KC85T solar panel (35ºC and 1200W/m^2) with method 1.

So, the user is able to have the voltage and current to operate at maximum power with the panel that we put. Furthermore, in the part of modelling we have the option to view your behaviour and in the part of simulation have the option to see their actual behaviour after that we have introduced the actual temperature and the actual irradiance of land on which the panels are installed.

Similarly, we can have these results for any of the three methods and any photovoltaic panel. Also in the simulation part, we try to introduce different levels of temperature and irradiance and watch their behaviour.

In the next section, we will be discussed the results obtained by this graphical interface.

6. DISCUSSION

This type of program, provides for very many tests, and according to what the user goal, you can get different results and discussions.

For example, in the example of the previous figure, we modelling for panel KC85T by the method 1. It can be seen that changing to the method 2 or method 3 out similar results. It seems logical, since all three methods should give similar results and this situation occurs in most of the panels. In contrast, in Isofoton24 panel, in MSX60_single and in ONYX cell or not as expected result. In this case the panels have difficulty to modelled by Method 2.This, may be because such panels are not adequately modelled by the method of ∞ Rp.

As already discussed above in the part of simulation we can change the temperature and irradiance which will subject the panels. If we put a temperature of 298.15 K and an irradiance of1000 W /m^2, we get the same graphics on both sides of the interface. But if we change any of the two values will have small differences.

It is thus clear that if we increase their radiance, intensity and consequently the maximum power generated is also increased. In contrast ,if we decrease the irradiance will decrease the intensity and power generated.

Some thing similar happens changing the temperature. If we decrease, increase the maximum power voltage so that the maximum power increases (the current remains more or less constant). Conversely if we increase the temperature, we decrease the maximum power voltage so that the maximum power decrease.

In the two upcoming figures we can see the two examples discussed above.

Figure 8 presents simulation results of the KC85T solar panel with 1500W / m^2 of irradiance and 298 K of temperature and comparing with Figure 7we can show that the maximum power increases.

Figure 9, represents simulation of the panel KC85T with 1000W / m^2 of irradiance and 278 K and a comparison with Figure 7 shows that the maximum power decreases.

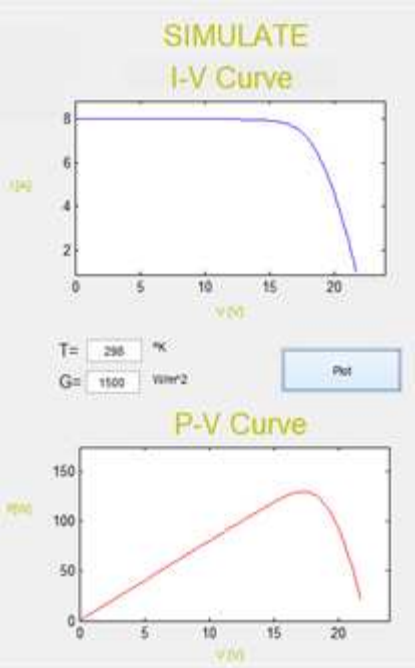

Figure 8. Simulation results at 1500W/m^2

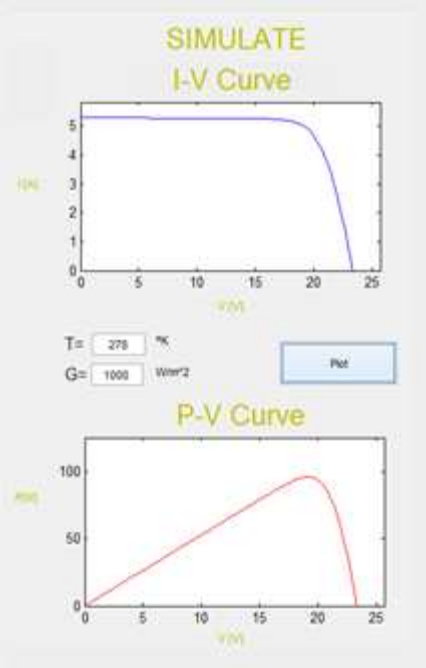

Figure 9. Simulation results 278 K

As already discussed above, we see in the figures (figure 7, figure 8 and figure 9) more clearly that increasing irradiance and decreasing temperature, the maximum power generated by the system increases.

7. CONCLUSION

The purpose of this work was to create a graphical interface for modelling and simulating different photovoltaic panels. Modelling algorithms were created using the Matlab language and a graphical interface was created.

The methodology of researching about algorithms and programming allowed a good understanding of PV panels and the creation of the program with a graphical interface was a very challenging goal.

The graphical tool developed in this work permits the simulation of any kind of single-junction PV module at any condition of temperature and irradiance. The program has been tested with several models of PV panels from different manufacturers and the results were excellent.

As conclusion of this work one can say that a powerful graphical tool for modelling and simulation has been created and the work offered a good opportunity of acquiring knowledge about the behaviour and functioning of PV panels and programming as well.

The program can be further improved with the addition of other functionalities and with the creation of other simulation options that can bring more flexibility for users.

8. Bibliography

[1] FOuQuEt,R.(2009):«A brief history of energy», en J. Evans and L.C.Hunt (eds.), In-ternational Handbook of the Economics of Energy, Cheltenham, UK and Northampton, MA, USA: Edward Elgar Publishing, p. 1-19.

[2] Univesity of Madrid. "Las energías renovables en el ámbito internacional". Prof. Francisco Javier André, Luis Miguel de Castro and Emilio Creda. [Online]. Available: http://www.revistasice.com/CachePDF/CICE_83__810091ECBB9FFCF682FDFE12C77FAB6D.pdf

[3] IEA "Technology Roadmap, Solar Photovoltaic Energy" International Energy Agency, Tech. Rep., 2014 [Online]. Available:

https://www.iea.org/publications/freepublications/publication/TechnologyRoadmapSolarPhotovoltaicEnergy_2014edition.pdf

[4] Internacional Energy Agency. "About solar photovoltaics"[Online]. Available: https://www.iea.org/topics/renewables/subtopics/solar/

[5] Power Electroncis and Control Techniques for maximum energy harvesting in photovoltaic sistems. Nicola Femia, Giocanni Petrone, Giovanni Spagnuolo and Massimo Vitelli. CRC Press (Taylor & Francis Group)

[6] Modeling and diagnosis of photovoltaic arrays. Doctoral Thesis Supervised by Dr. Edinson Franco, Dr. Giovanni Spagnuolo and Dr. Carlos Andr´es Ramos-Paja Departamento de Energ´ıa El´ectrica y Autom´atica - Universidad Nacional de Colombia

[7] Comprehensive Approach to Modeling andSimulation of Photovoltaic Arrays. Marcelo Gradella Villalva, Jonas Rafael Gazoli, and Ernesto Ruppert Filho. IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 24, NO. 5, MAY 2009

[8] Modeling and circuit-based simulation of photovoltaic arrays.M. G. Villalva, J. R. Gazoli, E. Ruppert F.University of Campinas - UNICAMP, Brazil. Brazilian journal of power electronics, 2009 vol. 14, no 1, pp. 35-45, ISSN 1414-8862

[9] Aprenda matlab 6.5. Javier garcia de Jalon. Madrid. AGosto 2004. Escuela Tecnica superior de ingenieros industriales. Universidad politecnica de Madrid.

9. APPENDIX

Method 1

```
% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil

%% PV model calculation

% Warning: do not change any values here unless you know what you are
% doing. You will have the opportunity to use any value of temperature or
% irradiance when you evaluate the model using the PV_model_eval program.

Gn = 1000;              %Nominal irradiance [W/m^2] @ 25oC
Tn = 25 + 273.15;       %Nominal operating temperature [K]

% Parameters used to control how the algorithm will work
%Increment of Rs
Rsinc = 0.001;

% Rsinc controls the precision and the algorithm speed. Small values are
% better for improved precision and helps the program to converge
%Maximum tolerable power error
tol = 0.0001; %Defines the model precision

%Number of "a" values to try
%namax = 100; %This will be used in a future version of the program

%Increment of "a"
%ainc = 0.01; %This will be used in a future version of the program

%Voltage points in each iteraction
nv = 100; %Defines how many points are used for obtaining the IxV curve

%Maximum number of iteractions for each value of "a"
%Avoids  program stall in case of non convergence
%This will be useful in a future version of the program
nimax = 500000;

%used for debugging
plott = 0; %1 = Enables plotting during algorithm execution
%0 = Disables plotting (better)
%% PROGRAM STARTS HERE

% Modeling algorithm - here we are obtaining the PV model parameters

% Reference values of Rs and Rp
% These values are not used in the modeling proces but they will
% be displayed at the end.
Rs_max = (Vocn - Vmp)/ Imp;
Rp_min = Vmp/(Iscn-Imp) - Rs_max;

% Initial guesses of Rp and Rs
Rs = 0;
Rp = Rp_min;

% The model is adjusted at the nominal condition
T = Tn;
G = Gn;
```

```
k = 1.3806503e-23;%Boltzmann [J/K]
q = 1.60217646e-19;  %Electron charge [C]

Vtn = k * Tn / q;           %Thermal junction voltage (nominal)
Vt  = k * T  / q;           %Thermal junction voltage (current temperature)

perror = Inf; %dummy value
ni = 0; %counter
a = 1; %Initial value of a

% Iterative loop executed untin Pmax,model = Pmax,experimental
while (perror>tol) && (Rp > 0) && (ni < nimax)

       ni = ni + 1;

       %  Temperature and irradiation effect on the current
              dT = T-Tn;
       Ipvn = (Rs+Rp)/Rp * Iscn;         % Nominal light-generated current
       Ipv = (Ipvn + Ki*dT) *G/Gn;       % Actual light-generated current
       Isc = (Iscn + Ki*dT) *G/Gn;       % Actual short-circuit current

       Ion = (Ipv - Vocn/Rp)/(exp(Vocn/Vt/a/Ns)-1);
       Io = Ion;

              % Increments Rs
       Rs  = Rs + Rsinc;
       Rp_ = Rp;

       % Egap = 2.72370016e-19;  % Bandgap do silício amorfo em J (=1.7 eV)
       Egap = 1.8e-19;         % Bandgap do silício cristalino em J (=1.124 eV)

       a = (Kv - Vocn/Tn) / ( Ns * Vtn * ( Ki/Ipvn - 3/Tn - Egap/(k*Tn^2) ) );


% a = 1; % Read the comments above.
% You can try other values of the ideality factor if your results are not
% satisfactory or if you find convergence problems.

       Rp = Vmp*(Vmp+Imp*Rs)/(Vmp*Ipv-Vmp*Io*exp((Vmp+Imp*Rs)/Vt/Ns/a)+Vmp*Io-
       Pmax_e);

       % Solving the I-V equation for several (V,I) pairs
       clearV
       clearI

       V = 0:Vocn/nv:Vocn;        % Voltage vector
       I = zeros(1,size(V,2));    % Current vector



       for j = 1 : size(V,2) %Calculates for all voltage values

       % Solves g = I - f(I,V) = 0 with Newton-Raphson

       g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);

       while (abs(g(j)) > 0.001)

       g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);
       glin(j) = -Io*Rs/Vt/Ns/a*exp((V(j)+I(j)*Rs)/Vt/Ns/a)-Rs/Rp-1;
       I_(j) = I(j) - g(j)/glin(j);
       I(j) = I_(j);

       end

       end% for j = 1 : size(V,2)
```

```
        % Calculates power using the I-V equation
        P = (Ipv-Io*(exp((V+I.*Rs)/Vt/Ns/a)-1)-(V+I.*Rs)/Rp).*V;

        Pmax_m = max(P);

        perror = (Pmax_m-Pmax_e);

        end% while (error>tol)

if (Rp<0) Rp = Rp_
end
```

Method 2

```
% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil



%% PROGRAM STARTS HERE

% PV MODEL calculation

% The theory used in this program for modeling the PV device is found
% in many sources in the litterature and is well explained in Chapter 1 of
% "Power Electronics and Control Techniques" by Nicola Femia, Giovanni
% Petrone, Giovanni Spagnuolo and Massimo Vitelli.

Gn = 1000;              %Nominal irradiance [W/m^2] @ 25oC
Tn = 25 + 273.15;       %Nominal operating temperature [K]

%Egap = 2.72370016e-19;% Bandgap do silício amorfo em J (=1.7 eV)
Egap = 1.8e-19;         % Bandgap do silício cristalino em J (=1.124 eV)

ns = Ns; % for compatibility

Ipvn = Iscn;

G = Gn;
T = Tn;
Ipv = Ipvn * G/Gn * (1 + Ki * (T-Tn));

k = 1.3806503e-23;   %Boltzmann [J/K]
q = 1.60217646e-19;  %Electron charge [C]

Vt =  k*T /q;
Vtn = k*Tn/q;

a = (Kv - Vocn/Tn) / ( ns * Vtn * ( Ki/Ipvn - 3/Tn - Egap/(k*Tn^2) ) );

Ion = Ipvn * exp( - Vocn/(a*ns*Vtn) );

C = Ion /  (Tn^3 * exp (-Egap / (k * Tn)));

Io = C * T^3 * exp(-Egap/k/T);

Rs = (a * ns * Vtn * log (1-Imp/Ipvn)+Vocn - Vmp)/Imp;

Rp = 9999999999; % Rp = infinite

% PROGRAM ENDS HERE


%% I-V and P-V CURVES of the calculated model at STC

% In this part of the program we are solving the I-V equation for several
% (V,I) pairs and ploting the curves using the model previously calculated

G =  1000;              %Irradiance for evaluating the model
T =  25 + 273.15;       %Temperature for evaluating the model

clear V
clear I
```

```
nv = 50; % número de pontos da curva

V = 0:Vocn/nv:Vocn;  % Voltage vector
I = zeros(1,size(V,2));    % Current vector

for j = 1 : size(V,2) %Calculates for all voltage values

% Solves g = I - f(I,V) = 0 with Newton-Raphson

g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);

while (abs(g(j)) > 0.001)

  g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);
glin(j) = -Io*Rs/Vt/ns/a*exp((V(j)+I(j)*Rs)/Vt/ns/a)-Rs/Rp-1;
I_(j) = I(j) - g(j)/glin(j);
  I(j) = I_(j);

end

end
```

Method 3

```
% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil

%% PROGRAM STARTS HERE

% PV model calculation

Gn = 1000;               %Nominal irradiance [W/m^2] @ 25oC
G =  1000;               %Irradiance for evaluating the model after the
parameter extraction
Tn = 25 + 273.15;        %Nominal operating temperature [K]
T =  25 + 273.15;        %Temperature for evaluating the model after the
parameter extraction

%Egap = 2.72370016e-19;  % Bandgap do silício amorfo em J (=1.7 eV)
Egap = 1.8e-19;          % Bandgap do silício cristalino em J (=1.124 eV)

% The theory used in this program for modeling the PV device is found
% in many sources in the litterature and is well explained in Chapter 1 of
% "Power Electronics and Control Techniques" by Nicola Femia, Giovanni
% Petrone, Giovanni Spagnuolo and Massimo Vitelli.

% This is straightforward way to calculate the PV model parameters but I
% have found some convergence problems for some devices.

Ipvn = Iscn; % We are assuming Ipv = Isc, which is a simplification

Ipv = Ipvn * G/Gn * (1 + Ki * (T-Tn));

k = 1.3806503e-23;   %Boltzmann [J/K]
q = 1.60217646e-19;  %Electron charge [C]

Vt =  k*T /q;
Vtn = k*Tn/q;

a = (Kv - Vocn/Tn) / ( Ns * Vtn * ( Ki/Ipvn - 3/Tn - Egap/(k*Tn^2) )  );

Ion = Ipvn * exp( - Vocn/(a*Ns*Vtn) );

C = Ion /  (Tn^3 * exp (-Egap / (k * Tn)));

Io = C * T^3 * exp(-Egap/k/T);

alfa = (Vmp*(Vmp-2*a*Vtn*Ns)/(a^2*Vtn^2*Ns^2));

beta = 2*Imp-Ipv-Ion;


delta = a * Ion*Vtn*Ns;

gama = 2 * Vmp/(a * Vtn*Ns) - Vmp^2/(a^2*Vtn^2*Ns^2);

y = ( Vmp*(beta)*exp (alfa) ) / (delta);

x = lambertw(y) + gama;

Rs = ( x * a * Vtn*Ns - Vmp)/Imp;
Rp = ( x * a * Vtn*Ns )/(Ipv - Imp - Ion * (exp(x)-1));
```

```
% PROGRAM ENDS HERE

%% I-V and P-V CURVES of the calculated model at STC

% Solving the I-V equation for several (V,I) pairs
clear V
clear I

nv = 50; % número de pontos da curva

V = 0:Vocn/nv:Vocn;  % Voltage vector
I = zeros(1,size(V,2));    % Current vector

for j = 1 : size(V,2) %Calculates for all voltage values

% Solves g = I - f(I,V) = 0 with Newton-Raphson

g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);

while (abs(g(j)) > 0.001)

  g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);
glin(j) = -Io*Rs/Vt/Ns/a*exp((V(j)+I(j)*Rs)/Vt/Ns/a)-Rs/Rp-1;
I_(j) = I(j) - g(j)/glin(j);
  I(j) = I_(j);

end

end
```

Evaluation

```
% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil
%This program evaluates the PV model for any T or G.
%You must run the PV_Modeling algorithm first


clc
%%

% Inputs
%T = 25 + 273.15;    %Temperature [K]
%G = 1000;  %Irradiance [W/m^2]

% Constants
k = 1.3806503e-23;%Boltzmann [J/K]
q = 1.60217646e-19;  %Electron charge [C]

% Thermal voltages
Vtn = k * Tn / q;    %Thermal junction voltage (nominal)
Vt = k * T / q;      %Thermal junction voltage (actual temperature)


%% Method of calculating Io

% Chose 1 to use the original method with (T/Tn)^3
% Chose 2 to use the alternative method with KV and KI

method = 1;


%% Calculation of Io (method 1)

%This is the original Io equation generally found in the literature.
%This requires finding the optimal value of Eg.
%See details in:
%"Modeling and circuit-based simulation of photovoltaica arrays"

if method == 1,

%Calculation of Eg and Io

Tmax = 75 + 273.15;
dT_ = Tmax - Tn;
Isc_ = (  Iscn + Ki*dT_  );
Voc_ = (  Vocn + Kv*dT_  );
Vt_ = k * Tmax / q;

Eg  = log(Isc_*Tn^3/Ion/Tmax^3/(exp(Voc_/a/Ns/k/Tmax*q)-
1))*a*k*Tn*Tmax/q/(Tmax-Tn);
%or:
%Eg = -log((Iscn+Ki*(Tmax-Tn))*Tn^3/Ion/Tmax^3/(exp((Vocn+Kv*(Tmax-
Tn))/a/Ns/k/Tmax*q)-1))*a*k*Tn*Tmax/q/(-Tmax+Tn);

Io  = Ion *(T/Tn)^(3) * exp(  q * Eg/a/ k * (1/Tn-1/T) );

end


%% Calculation of Io (method 2)
```

```
%This is the alternative Io equation suggested in:
%"Comprehensive approach to modeling and simulation of photovoltaic arrays"



if method == 2,

dT = T - Tn;
Isc_ = (  Iscn + Ki*dT  );
Voc_ = (  Vocn + Kv*dT  );

%Io = Isc_/(exp(Voc_/a/Ns/Vt)-1); %% OLD %%

Ipv_ = (Rs+Rp)/Rp * Isc_; %% NEW %% UPDATED ON JUNE/2010 %%

Io = (Ipv - Voc_/Rp)/(exp(Voc_/Vt/a/Ns)-1); %% NEW %% UPDATED ON JUNE/2010 %%

end


%%  Temperature and irradiation effect on the current
dT = T-Tn;
Ipvn = (Rs+Rp)/Rp * Iscn;         % Nominal light-generated current
Ipv = (Ipvn + Ki*dT) *G/Gn;       % Actual light-generated current
Isc = (Iscn + Ki*dT) *G/Gn;       % Actual short-circuit current


%% Solving the I-V equation for several (V,I) pairs
clearV
clearI

nv = 100;

V = 0:(Vocn+Kv*dT)/nv:(Vocn+Kv*dT);  % Voltage vector
I = zeros(1,size(V,2));    % Current vector

for j = 1 : size(V,2) %Calculates for all voltage values

% Solves g = I - f(I,V) = 0 with Newton-Raphson

g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);

while (abs(g(j)) > 0.001)

  g(j) = Ipv-Io*(exp((V(j)+I(j)*Rs)/Vt/Ns/a)-1)-(V(j)+I(j)*Rs)/Rp-I(j);
glin(j) = -Io*Rs/Vt/Ns/a*exp((V(j)+I(j)*Rs)/Vt/Ns/a)-Rs/Rp-1;
I_(j) = I(j) - g(j)/glin(j);
  I(j) = I_(j);


end

end
```

Main program

```
% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil

function varargout = interface_M1(varargin)
% INTERFACE_M1 M-file for interface_M1.fig
%      INTERFACE_M1, by itself, creates a new INTERFACE_M1 or raises
%      the existing singleton*.
%
%      H = INTERFACE_M1 returns the handle to a new INTERFACE_M1 or
%      the handle to the existing singleton*.
%
%      INTERFACE_M1('CALLBACK',hObject,eventData,handles,...) calls the
%      local function named CALLBACK in INTERFACE_M1.M with the given
%      input arguments.
%      INTERFACE_M1('Property','Value',...) creates a new INTERFACE_M1 or
%      raises the existing singleton*.  Starting from the left, property
%      value pairs are applied to the GUI before interface_M1_OpeningFcn
%      gets called.  An unrecognized property name or invalid value makes
%      property application stop.  All inputs are passed to
%      interface_M1_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help interface_M1

% Last Modified by GUIDE v2.5 27-Apr-2015 21:31:25

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @interface_M1_OpeningFcn, ...
'gui_OutputFcn',  @interface_M1_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


%PROGRAMMING THE PROGRAM STARTS:

% --- Executes just before interface_M1 is made visible.
function interface_M1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to interface_M1 (see VARARGIN)
```

```
%%IMAGEN
        axes(handles.axes8) %Carga la imagen en background
handles.imagen=imread('unicamp.jpg');
imagesc(handles.imagen)
axisoff

axes(handles.axes7) %Carga la imagen en background
        handles.imagen=imread('bombilla.jpg');
imagesc(handles.imagen)
axisoff

% to reset the name of new panels
        t=get(handles.listbox1,'string');
save ('cont','t')



% load the name of the new panels
        k=0;
val=0;

        p=1;
number=0;
save ('cont','number','-append')
        data_panels


% Choose default command line output for interface_M1
        handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes interface_M1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = interface_M1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%PROGRAMMING TO CHOOSE THE ALGORITHM:

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
%        popupmenu2 contents as cell array
%contents{get(hObject,'Value')} returns selected
%item from popupmenu2
number=get(hObject,'Value');
save ('cont','number','-append')
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all
%            CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


% PROGRAMMING BUTTON 'RESET ALL PANELS':

% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject    han9dle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% x is variable to count the number of new panels
        z=0;
save ('memoria','z')

% to reset the name of the new panels

set(handles.listbox1,'value',1)
load ('cont','t')
set(handles.listbox1,'string',t);
clearw


% PROGRAMMING BUTTON NEW PANEL

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

        k=1;
val=0;
        p=0;
        data_panels;

%PROGRAMING BOTTON DELETE ONE OF THE NEW PANEL

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load ('cont','val')
if val<10

warndlg('You can´t delete this panel','sorry')
end

load ('cont','val')
if val>9
load ('memoria','w')
load ('cont','val')
fin=length (w)
if val==fin
                u=w(1:(val-1));
z=val-10;
save ('memoria','z','-append')
                w=u;
save ('memoria','w','-append')
set(handles.listbox1,'value',val-1)
set(handles.listbox1,'string',w)
else
```

```
                u=w(1:(val-1));
                u(val:fin-1)=w(val+1:fin);
w=u;
save ('memoria','w','-append')
set(handles.listbox1,'string',w)
                z=fin-10;
save ('memoria','z','-append')
load ('memoria','M')
load ('cont','val')
for d=0:1:(fin-val-1)
                    load ('cont','val')
                    M(val+d,1)=M(val+d+1,1);
                    M(val+d,2)=M(val+d+1,2);
                    M(val+d,3)=M(val+d+1,3);
                    M(val+d,4)=M(val+d+1,4);
                    M(val+d,5)=M(val+d+1,5);
                    M(val+d,6)=M(val+d+1,6);
                    M(val+d,7)=M(val+d+1,7);
M(val+d,8)=M(val+d+1,8);
d=d+1;
end
for d=1:8
M(z+10,d)=0;
end
save ('memoria','M','-append')

end

end


%PROGRAMMING TO CHOOSE THE PANEL


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
%        contents as cell array contents{get(hObject,'Value')}
%        returns selected item from listbox1
%contenido=get(hObject,'String');
val=get(hObject,'Value');
        k=0;
        p=0;
save ('cont','val','-append')
            data_panels;

if val<11

                Iscn= M(val,1);
Vocn= M(val,2);
                Imp= M(val,3);
                Vmp= M(val,4);
                Pmax_e= M(val,5);
                Kv= M(val,6);
                Ki= M(val,7);
                Ns= M(val,8);

end
load ('cont','number')
if number==1
                data_PVM1new
plot(handles.axes4,V,I);
plot(handles.axes5,V,V.*I,'r');
                handles.Plot=[Iscn Ki Vocn Kv Ion a Ns Rs Rp];
```

```
elseif number==2
                data_PVM2
for a=1:size(I,2)
I(a)=norm(I(a))
end
plot(handles.axes4,V,I);
plot(handles.axes5,V,V.*I,'r');
                handles.Plot=[Iscn Ki Vocn Kv Ion a Ns Rs Rp];
elseif number==3
                data_PVM3
axes(handles.axes4)
plot(handles.axes4,V,I);
ylim([0 max(I)*1.1]);

axes(handles.axes5)
plot(handles.axes5,V,V.*I,'r');
ylim([0 max(I)*max(V)*1.01]);
handles.Plot=[Iscn Ki Vocn Kv Ion a Ns Rs Rp];
else
warndlg('You need to choose one method of algorithm')
end

guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%            called

% Hint: listbox controls usually have a white background on Windows
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


%PROGRAMMING THE TEMPERATURE OF ENTRY

function T_Callback(hObject, eventdata, handles)
% hObject    handle to T (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of T as text
%        str2double(get(hObject,'String')) returns contents of T
%        as a double
        NewStrVal=get(hObject,'String');
        NewVal = str2double(NewStrVal);
        handles.T=NewVal;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function T_CreateFcn(hObject, eventdata, handles)
% hObject    handle to T (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%            called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

```
%PROGRAMMING THE IRRADIANCE OF ENTRY

function G_Callback(hObject, eventdata, handles)
% hObject    handle to G (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of G as text
%        str2double(get(hObject,'String')) returns contents of G as
%        a double
        NewStrVal=get(hObject,'String');
        NewVal = str2double(NewStrVal);
        handles.G=NewVal;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function G_CreateFcn(hObject, eventdata, handles)
% hObject    handle to G (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%            called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

%PROGRAMMING BUTTON PLOT

% --- Executes on button press in Plot.
function Plot_Callback(hObject, eventdata, handles)
% hObject    handle to Plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        T=handles.T;
        G=handles.G;
        Iscn=handles.Plot(1);
        Ki=handles.Plot(2);
        Vocn=handles.Plot(3);
        Kv=handles.Plot(4);
Ion=handles.Plot(5);
        a=handles.Plot(6);
        Ns=handles.Plot(7);
        Rs=handles.Plot(8);
        Rp=handles.Plot(9);
Tn=25+273.15;
        Gn=1000;
        data_eval

if min(I)<0

%for a=1:size(I,2)
%    I(a)=norm(I(a))
%end

axes(handles.axes1)
plot(handles.axes1,V,I);
xlim([0 (Vocn+Kv*dT)*1.1]);
                ylim([0 max(I)*1.1]);

axes(handles.axes10)
plot(handles.axes10,V,V.*I,'r');
xlim([0 (Vocn+Kv*dT)*1.1]);
                ylim([0 max(V)*max(I)*1.01]);
```

```
elseif max(I)<0

for a=1:size(I,2)
                    I(a)=norm(I(a))
end

                axes(handles.axes1)
plot(handles.axes1,V,I);
xlim([0 (Vocn+Kv*dT)*1.1]);
                ylim([0 min(I)*1.1]);


axes(handles.axes10)
plot(handles.axes10,V,V.*I,'r');
xlim([0 (Vocn+Kv*dT)*1.1]);
                ylim([0 max(V)*max(I)*1.01]);

elseif max(V)<0

for a=1:size(V,2)
V(a)=norm(V(a))
end

axes(handles.axes1)
plot(handles.axes1,V,I);
xlim([0 max(V)*1.1]);
                ylim([0 max(I)*1.1]);

axes(handles.axes10)
plot(handles.axes10,V,V.*I,'r');
xlim([0 max(V)*1.1]);
                ylim([0 max(V)*max(I)*1.01]);

elseif min(V)<0

for a=1:size(V,2)
V(a)=norm(V(a))
end

axes(handles.axes1)
plot(handles.axes1,V,I);
xlim([0 max(V)*1.1]);
                ylim([0 max(I)*1.1]);

axes(handles.axes10)
plot(handles.axes10,V,V.*I,'r');
xlim([0 max(V)*1.1]);
                ylim([0 max(V)*max(I)*1.01]);

else
axes(handles.axes1)
plot(handles.axes1,V,I);
xlim([0 max(V)*1.1]);
                ylim([min(I)/1.1 max(I)*1.1]);

axes(handles.axes10)
plot(handles.axes10,V,V.*I,'r');
xlim([0 max(V)*1.1]);
                ylim([0 max(V)*max(I)]);
end
```

Panels

```
% DIFFERENTS PHOTOVOLTAIC PANELS

% Author: Julen Lizarrondo Ostiz
% Email: julenlizarrondo@hotmail.com

% Tutor: Marcelo Gradella Villalva
% Email: mvillalva@gmail.com

% University of Campinas, Brazil

% If you are interested in introducing a new panel, read the end of this
% document. If you have any questions do not hesitate to send me an email.

%% AMORPHOUS.m

%PV panel formed of 29 cells TPS-102 2.4W (amorphous silicon)

M(1,1)= 0.170;          %Iscn    Nominal short-circuit voltage [A]
M(1,2)= 21;             %Vocn    Nominal array open-circuit voltage [V]
M(1,3)= 0.137;          %Imp     Array current @ maximum power point [A]
M(1,4)= 17.5;           %Vmp     Array voltage @ maximum power point [V]
M(1,5)= M(1,3)*M(1,4);  %Pmax_e  Array maximum output peak power [W]
M(1,6)= -0.123;         %Kv      Voltage/temperature coefficient [V/K]
M(1,7)= 3.18e-3;        %Ki      Current/temperature coefficient [A/K]
M(1,8)= 29;             %Ns      Nunber of series cells

%% DATA_ISOFOTON24.m

M(2,1)= 8.7;            %Iscn   Nominal short-circuit voltage [A]
M(2,2)= 22.6;           %Vocn   Nominal array open-circuit voltage [V]
M(2,3)= 8.12;           %Imp    Array current @ maximum power point [A]
M(2,4)= 18.5;           %Vmp    Array voltage @ maximum power point [V]
M(2,5)= M(2,3)*M(2,4);  %Pmax_e Array maximum output peak power [W]
M(2,6)= -0.454*M(2,2);  %Kv     Voltage/temperature coefficient [V/K]
M(2,7)= 0.042*M(2,1);   %Ki     Current/temperature coefficient [A/K]
M(2,8)= 36;             %Ns     Nunber of series cells

%% DATA_KC85T.m

%%Information from the KC85T solar array datasheet

M(3,1)= 5.34;           %Iscn   Nominal short-circuit voltage [A]
M(3,2)= 21.7;           %Vocn   Nominal array open-circuit voltage [V]
M(3,3)= 5.02;           %Imp    Array current @ maximum power point [A]
M(3,4)= 17.4;           %Vmp    Array voltage @ maximum power point [V]
M(3,5)= M(3,3)*M(3,4);  %Pmax_e Array maximum output peak power [W]
M(3,6)= -8.21e-2;       %Kv     Voltage/temperature coefficient [V/K]
M(3,7)= 2.12e-3;        %Ki     Current/temperature coefficient [A/K]
M(3,8)= 36;             %Ns     Nunber of series cells

%% DATA_KD135SX_UPU.m

%%Information from the KD135SX_UPU solar array datasheet

M(4,1)= 8.37;           %Iscn   Nominal short-circuit voltage [A]
M(4,2)= 22.1;           %Vocn   Nominal array open-circuit voltage [V]
M(4,3)= 7.63;           %Imp    Array current @ maximum power point [A]
M(4,4)= 17.7;           %Vmp    Array voltage @ maximum power point [V]
M(4,5)= M(4,3)*M(4,4);  %Pmax_e Array maximum output peak power [W]
M(4,6)= -8e-2;          %Kv     Voltage/temperature coefficient [V/K]
M(4,7)= 5.02e-3;        %Ki     Current/temperature coefficient [A/K]
M(4,8)= 36;             %Ns     Nunber of series cells
```

```
%% data_M2453BB.m

%%Information from the solar array datasheet

M(5,1)= 8.7;              %Iscn   Nominal short-circuit voltage [A]
M(5,2)= 37.7;             %Vocn   Nominal array open-circuit voltage [V]
M(5,3)= 8.2;              %Imp    Array current @ maximum power point [A]
M(5,4)= 30.1;             %Vmp    Array voltage @ maximum power point [V]
M(5,5)= M(5,3)*M(5,4);    %Pmax_e Array maximum output peak power [W]
M(5,6)= -0.32/100*M(5,2); %Kv     Voltage/temperature coefficient [V/K]
M(5,7)= -0.032/100*M(5,1);%Ki     Current/temperature coefficient [A/K]
M(5,8)= 60;               %Ns     Nunber of series cells

%% data_MSX60.m

%%Information from the MSX60 solar array datasheet

M(6,1)= 3.8;            %Iscn   Nominal short-circuit voltage [A]
M(6,2)= 21.1;           %Vocn   Nominal array open-circuit voltage [V]
M(6,3)= 3.5;            %Imp    Array current @ maximum power point [A]
M(6,4)= 17.1;           %Vmp    Array voltage @ maximum power point [V]
M(6,5)= M(6,3)*M(6,4);  %Pmax_e Array maximum output peak power [W]
M(6,6)= -80e-3;         %Kv     Voltage/temperature coefficient [V/K]
M(6,7)= .003;           %Ki     Current/temperature coefficient [A/K]
M(6,8)= 36;             %Ns     Nunber of series cells


%% data_MSX60_single_cell.m

%%Information from the MSX60 solar array datasheet

M(7,1) = 3.8;           %Iscn   Nominal short-circuit voltage [A]
M(7,2)= 21.1/36;        %Vocn   Nominal array open-circuit voltage [V]
M(7,3)= 3.5;            %Imp    Array current @ maximum power point [A]
M(7,4)= 17.1/36;        %Vmp    Array voltage @ maximum power point [V]
M(7,5)= M(7,3)*M(7,4);  %Pmax_e Array maximum output peak power [W]
M(7,6)= -80e-3;         %Kv     Voltage/temperature coefficient [V/K]
M(7,7)= .003;           %Ki     Current/temperature coefficient [A/K]
M(7,8)= 1;              %Ns     Nunber of series cells



%% data_ONYX.m

%%Information from the KD135SX_UPU solar array datasheet

M(8,1)= 1.15;           %Iscn   Nominal short-circuit voltage [A]
M(8,2)= 48;             %Vocn   Nominal array open-circuit voltage [V]
M(8,3)= 1.04;           %Imp    Array current @ maximum power point [A]
M(8,4)= 36;             %Vmp    Array voltage @ maximum power point [V]
M(8,5) = M(8,3)*M(8,4); %Pmax_e Array maximum output peak power [W]
M(8,6)= -0.28*M(8,2);   %Kv     Voltage/temperature coefficient [V/K]
M(8,7)= 0.09*M(8,1);    %Ki     Current/temperature coefficient [A/K]
M(8,8)= 30;             %Ns     Nunber of series cells

%% data_Q6LM.m

%%Information from the Q6LM solar cell datasheet (typical curve)

M(9,1) = 7.61;          %Iscn   Nominal short-circuit voltage [A]
M(9,2)= 0.6118;         %Vocn   Nominal array open-circuit voltage [V]
M(9,3)= 7.1147;         %Imp    Array current @ maximum power point [A]
M(9,4)= 0.5119;         %Vmp    Array voltage @ maximum power point [V]
M(9,5)= M(9,3)*M(9,4);  %Pmax_e Array maximum output peak power [W]
M(9,6)= -.0037*M(9,2);  %Kv     Voltage/temperature coefficient [V/K]
M(9,7)= .0005*M(9,1);   %Ki     Current/temperature coefficient [A/K]
M(9,8)= 1;              %Ns     Nunber of series cells
```

```
%% PROGRAMMING TO CREATE NEW PANEL


if k==1

load ('memoria','z')
if z>0
load ('memoria','M')
end

prompt = {'Enter the name of panel:'};
               dlg_title = 'Enter the name of panel:';
name  = inputdlg(prompt,dlg_title);
titulo=name;
save ('cont','titulo','-append')


load ('memoria','z')

prompt = {'Enter the value of Iscn in A'};
               dlg_title1 = 'Enter the value of Iscn in A';
               b = inputdlg(prompt,dlg_title1);             %Iscn M(1,1);
M(10+z,1) = str2num(b{1});

prompt = {'Enter the value of Vocn in V'};
               dlg_title2 = 'Enter the value of Vocn in V';
               b = inputdlg(prompt,dlg_title2);
M(10+z,2) = str2num(b{1});                    %Vocn M(1,2);

prompt = {'Enter the value of Imp in A'};
               dlg_title3 = 'Enter the value of Imp in A';
               b = inputdlg(prompt,dlg_title3);
M(10+z,3) = str2num(b{1});                    %Imp  M(1,3);

prompt = {'Enter the value of Vmp in V'};
               dlg_title4 = 'Enter the value of Vmp in V';
               b = inputdlg(prompt,dlg_title4);
M(10+z,4) = str2num(b{1});                    %Vmp M(1,4);

               M(10+z,5)=M(10+z,3)*M(10+z,4);               %Pmax_e M(1,5);

prompt = {'Enter the value of Kv in V/K'};
               dlg_title5 = 'Enter the value of Kv in V/K';
               b = inputdlg(prompt,dlg_title5);
M(10+z,6) = str2num(b{1});                    %Kv M(1,6);

prompt = {'Enter the value of ki in A/K'};
               dlg_title6 = 'Enter the value of ki in A/K';
               b = inputdlg(prompt,dlg_title6);
M(10+z,7) = str2num(b{1});                    %Ki M(1,7);

prompt = {'Enter the value of Ns'};
               dlg_title7 = 'Enter the value of Ns';
               b = inputdlg(prompt,dlg_title7);
M(10+z,8) = str2num(b{1});                    %Ns M(1,8);

z=z+1;

save ('memoria','M','-append')
save ('memoria','z','-append')
               k=0;

if z==1
load ('cont','t')
                    w=t; %get(handles.listbox1,'string');
```

```
            else
load ('memoria','w')

end

if ischar(w)
                    w=cellstr(w);
end


load ('cont','titulo')

                w=[w;titulo];
save ('memoria','w','-append')
set(handles.listbox1,'string',w)

end


 %% DATA TO LOAD THE NEW PANELS

if val>9
load ('memoria','z')
load ('memoria','M')
load ('cont','val')
Iscn= M(val,1);
                Vocn= M(val,2);
Imp= M(val,3);
                Vmp= M(val,4);
                Pmax_e= M(val,5);
                Kv= M(val,6);
                Ki= M(val,7);
                Ns= M(val,8);

end

 %% DATA TO LOAD THE NAME OF THE NEW PANELS

if p==1
               p=0;

load ('memoria','z')
if z>0
load ('memoria','w')
set(handles.listbox1,'string',w)
else
load ('cont','t')
set(handles.listbox1,'string',t)

end
end
```