Funciones desarrolladas en MATLAB

****

```
function [I,V,P] = R_solar_modulo(RL, G, T, model)
% I_solar_modulo.m --  modelo para generadores solares fotov.
%
%
% Entradas:
%   V, Tensión de entrada
%   G, irradiancia (Wm^-2)
%   T, temperatura en grados Celsius
%   model, parametros de la célula solar
% Salidas:
%   I, vector de corrientes
% RL, resistencia de carga
eps=1e-4;  %Precision
k = 1.3803e-23;     % Constante de Boltzmann (J/K)
q = 1.602e-19;      % Carga del electrón (Cul)
% Parámetros del modelo
eval(model);


% Variables
A=A; % factor de idealidad de la union PN


Tr=Tr; % Temperatura de referencia en ºC
TrK = 273 + Tr;     % Temperatura de referencia en K
TK=273+T;   % Temperatura de trabajo del array en K.
```

```
Gref=Gr;  %Radiación de referencia


Voc_ref=Vocr;   % Tensión de circuito abierto en las condiciones de referencia

Isc_ref=Iscr;   % Corriente de corto circuito  en las condiciones de referencia

Vmpp_ref=Vmpp;  % Tensión del MPP en las condiciones de referencia

Impp_ref=Impp;  % Corriente del MPP en las condiciones de referencia

Rs=Rs;          % Resistencia serie del generador

Rp=Rp;          % Resistencia paralelo del generador


kv=kv; %coeficiente de temperatura para Voc a nivel de módulo

ki=ki; %coeficiente de temperatura para Isc a nivel de módulo

m=A*ns;   % factor de idealidad a efectos de módulo


VTx=k*TK/q;  % Potencial Termico a la temperatura de trabajo


%%Traslación de los valores corregidos de Isc y Voc


Vocx=Voc_ref+m*k*TK/q*log(G/Gref)+kv*(T-Tr)  ;   % Vocx= Voc en las condiciones de trabajo;

%Iscx=Isc_ref*(G/Gref)+ki*(T-Tr)    ;           % Iscx=Isc en las condiciones de trabajo;

Iscx=(Isc_ref +ki*(T-Tr))*(G/Gref)    ;         % Iscx=Isc en las condiciones de trabajo;



% Determinación de los parámetros del modelo analítico

I0x=(Iscx*(1+Rs/Rp)-Vocx/Rp)/(exp(Vocx/VTx/m)-exp(Iscx*Rs/VTx/m));   % I0x= I0 (corriente
inversa de sat)  en las condiciones de trabajo;


ILx=I0x*(exp(Vocx/VTx/m)-1)+Vocx/Rp;          % ILx= IL (fotocorriente)  en las condiciones de
trabajo;
```

```
%determinación de la corriente por el método de Newton-Rapshon

I = zeros(size(RL));
epsilon=10;

while epsilon >eps;
Ia_ant=I;

f= I-ILx+I0x*(exp((I*RL+I*Rs)/VTx/m)-1)+(I*RL+I*Rs)/Rp;
df=1+I0x*exp((I*RL+I*Rs)/VTx/m)*(Rs+RL)/VTx/m + (Rs+RL)/Rp;
I= I-f/df;
 epsilon=abs(I-Ia_ant);
end

% Evitamos corrientes negativas (emulamos el diodo de bloqueo)
if I <0;
I =0;
end

V=I*RL;
P=V.*I;
**********************************************************************
```

```
**********************************************************************

function I = I_solar_modulo(V, G, T, model)
% I_solar_modulo.m --  modelo para generadores solares fotov.
%
%
% Entradas:
%   V, Tensión de entrada
%   G, irradiancia (Wm^-2)
%   T, temperatura en grados Celsius
%   model, parametros de la célula solar
% Salidas:
%   I, vector de corrientes

eps=1e-4;  %Precision

k = 1.3803e-23;     % Constante de Boltzmann (J/K)
q = 1.602e-19;      % Carga del electrón (Cul)
% Parámetros del modelo
eval(model);

% Variables
A=A; % factor de idealidad de la union PN

Tr=Tr; % Temperatura de referencia en ºC
TrK = 273 + Tr;     % Temperatura de referencia en K
TK=273+T;   % Temperatura de trabajo del array en K.
```

```
Gref=Gr;  %Radiación de referencia


Voc_ref=Vocr;   % Tensión de circuito abierto en las condiciones de referencia

Isc_ref=Iscr;   % Corriente de corto circuito  en las condiciones de referencia

Vmpp_ref=Vmpp;  % Tensión del MPP en las condiciones de referencia

Impp_ref=Impp;  % Corriente del MPP en las condiciones de referencia

Rs=Rs;          % Resistencia serie del generador

Rp=Rp;          % Resistencia paralelo del generador


kv=kv; %coeficiente de temperatura para Voc a nivel de módulo

ki=ki; %coeficiente de temperatura para Isc a nivel de módulo



m=A*ns;   % factor de idealidad a efectos de módulo


VTx=k*TK/q;  % Potencial Termico a la temperatura de trabajo


%%Traslación de los valores corregidos de Isc y Voc


Vocx=Voc_ref+m*k*TK/q*log(G/Gref)+kv*(T-Tr)  ;   % Vocx= Voc en las condiciones de trabajo;

%Iscx=Isc_ref*(G/Gref)+ki*(T-Tr)    ;          % Iscx=Isc en las condiciones de trabajo;

Iscx=(Isc_ref +ki*(T-Tr))*(G/Gref)    ;          % Iscx=Isc en las condiciones de trabajo;


% Determinación de los parámetros del modelo analítico

I0x=(Iscx*(1+Rs/Rp)-Vocx/Rp)/(exp(Vocx/VTx/m)-exp(Iscx*Rs/VTx/m));   % I0x= I0 (corriente
inversa de sat)  en las condiciones de trabajo;
```

```
ILx=I0x*(exp(Vocx/VTx/m)-1)+Vocx/Rp;          % ILx= IL (fotocorriente)  en las condiciones de trabajo;


%determinación de la corriente por el método de Newton-Rapshon


I = zeros(size(V));

epsilon=10;


while epsilon >eps;

Ia_ant=I;


f= I-ILx+I0x*(exp((V+I*Rs)/VTx/m)-1)+(V+I*Rs)/Rp;

df=1+I0x*exp((V+I*Rs)/VTx/m)*Rs/VTx/m + Rs/Rp;

I= I-f/df;


 epsilon=abs(I-Ia_ant);

end


% Evitamos corrientes negativas (emulamos el diodo de bloqueo)

if I <0;

I =0;

End


**********************************************************************
```

```
**************************************************************************

function z=ponderado(u,x,Uexp,Uhip)

z=u*(Uexp(1)+Uexp(2)*exp(-x/Uexp(3)))+(1-u)*(Uhip(1)+Uhip(2)./x);

**************************************************************************

**************************************************************************

function z=polinomico3(u,x)

z=u(1)+u(2)./x+u(3)./x.^2+u(4)./x.^3;

**************************************************************************

function z=polinomico2(u,x)

z=u(1)+u(2)./x+u(3)./x.^2;

**************************************************************************

**************************************************************************

function z=polinomico2(u,x)

z=u(1)+u(2)./x+u(3)./x.^2;

**************************************************************************

**************************************************************************

function z=hiperbolico(u,x)

z=u(1)+u(2)./x;

**************************************************************************
```

```
**************************************************************************
function z=fobj_ponderado(u,x,ym,Uexp,Uhip)
ycal=ponderado(u,x,Uexp,Uhip);
z=sum((ym-ycal).^2);
**************************************************************************

**************************************************************************

function z=fobj_polinomico3(u,x,ym)
ycal=polinomico3(u,x);
z=sum((ym-ycal).^2);
**************************************************************************

**************************************************************************
function z=fobj_polinomico2(u,x,ym)
ycal=polinomico2(u,x);
z=sum((ym-ycal).^2);
**************************************************************************

**************************************************************************
function z=fobj_oeh(u,x,ym)
ycal=oeh(u,x);
z=sum((ym-ycal).^2);
**************************************************************************
```

```
**************************************************************************
function z=fobj_hiperbolico(u,x,ym)
ycal=hiperbolico(u,x);
z=sum((ym-ycal).^2);
**************************************************************************


**************************************************************************
function z=fobj_exponencial(u,x,ym)
ycal=exponencial(u,x);
z=sum((ym-ycal).^2);
**************************************************************************
function z=exponencial(u,x)
z=u(1)+u(2)*exp(-x/u(3));
**************************************************************************
```