DEVELOPING A MATLAB CODE FOR PARAMETRIC ANALYSIS OF A MODELLED COUPLED PHOTOVOLTAIC/THERMAL CONCENTRATING SOLAR COLLECTOR FOR ELECTRICITY GENERATION

Mahdi Kiani
Ph.D. Student
Tulsa, OK, USA

ABSTRACT

There are two common methods to generate renewable energy by using the power of the sun including photovoltaic and solar thermal. By combining these two methods in one system, though more electrical energy can be generated but due to competing operating requirements new problems emerge. Experimental and theoretical studies are required to enhance the efficiency of this kind of hybrid systems. To optimize and analyze the combined efficiencies in a coupled photovoltaic (PV)/thermal concentrating solar collector systems a new approach has been introduced and published by Otanicar based on a coupled electrical/thermal model. Developing a Matlab code for this model has been described in this paper. The code takes solar irradiance, fluid temperature in/out and ambient temperature and returns PV efficiency and thermal efficiency as well as mass flow rate of the heat transfer fluid.

INTRODUCTION

To optimize the overall efficiency of hybrid PV/thermal systems at high concentration ratios and temperatures a coupled electro-thermal model has been innovated by Otanicar and his colleagues (1, 2).

In their proposed configuration losses in the PV cell due to reduced efficiencies at elevated temperatures and the incident solar energy below the PV band gap are both harnessed as heat (2).

In other words, they have presented a unique design strategy for a hybrid PV/thermal system that only has mild thermal coupling which can lead to enhanced efficiency. It has been claimed that by creating a fluid filter (Figure 1) that absorbs energy directly in the fluid below the band-gap and a PV cell with an active cooling strategy combined efficiencies greater than 38% can be achieved (1).

Based on the heat fluxes (Figure 2) and electrical efficiency of the PV an iterative numerical scheme that involves a coupled electro-thermal simulation of the solar energy conversion process has been introduced (1, 2).

Investigating the interaction between PV efficiency and the solar thermal collector's operating parameters is the purpose of this iterative numerical scheme (2).

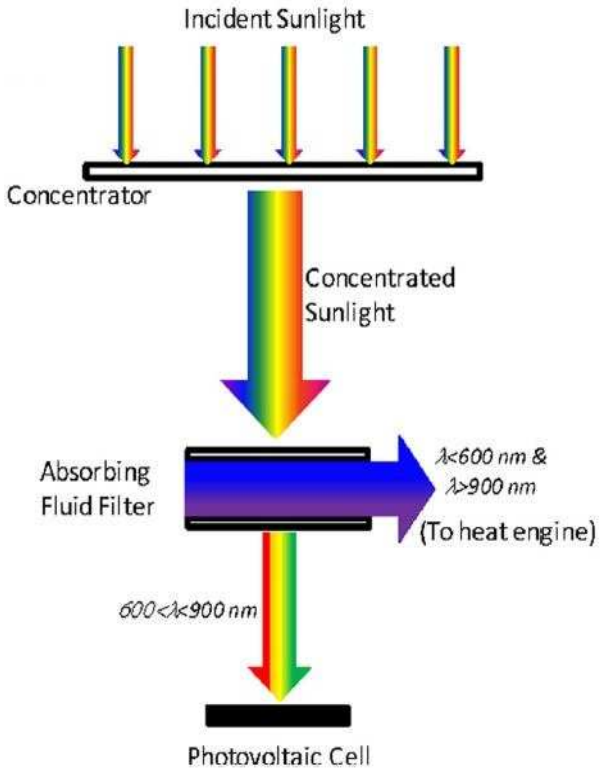

Figure 1: Using absorbing fluid filter to thermally decoupled PV/T hybrid concentrating solar collector (1).

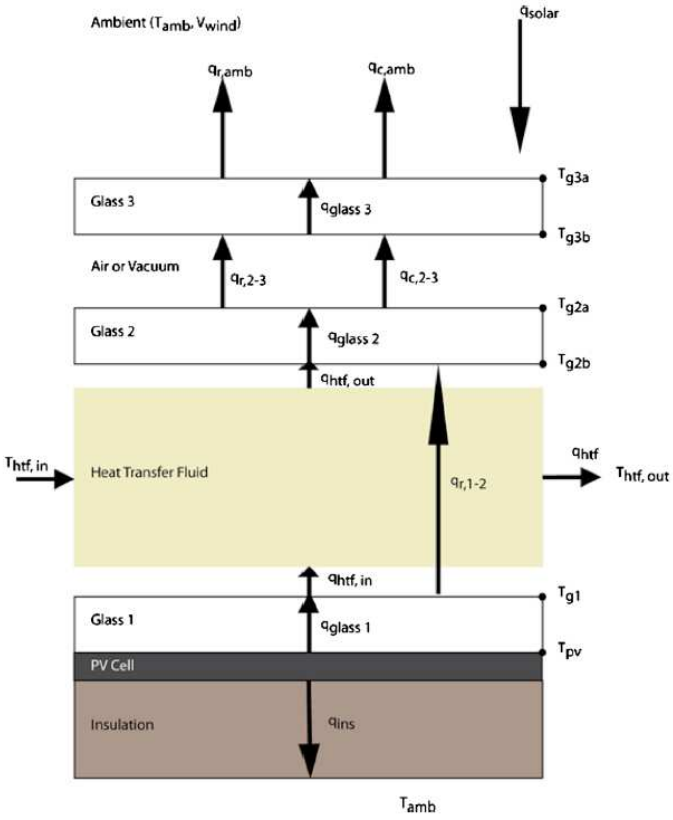

Figure 2: PV/thermal hybrid collector schematic for thermal modeling (2).

In this project based on the presented iterative numerical approach a Matlab code has been developed for this coupled photovoltaic/thermal concentrating solar collector model to make the analysis and optimizations easier and faster. Moreover, a new model for PV efficiency has been applied.

Electrical Model

There are correlations which represent the temperature dependence of the PV module's electrical efficiency, many of them assume a linear form, differing only in the numerical values of the relevant parameters which, as expected, are material and system dependent (3).

Equation 1 shows the traditional linear expression for the PV electrical efficiency.

$$\eta_{pv} = \eta_{Tref}[1 - \beta_{ref}(T_{pv} - T_{ref})] \quad (1)$$

The quantities η_{Tref} and β_{ref} are normally given by the PV manufacturer. However, they can be obtained from flash tests in which the module's electrical output is measured at two different temperatures for a given solar radiation flux (3).

A number of equations found in the literature for the efficiency of PV cells/modules are shown in Table 1 which contains values for the parameters of equation 1 (3).

Table 1: PV efficiency correlation coefficients (3)

T_{ref} (°C)	$\eta_{T_{ref}}$	β_{ref} (°C^{-1})	Comments
25	0.15	0.0041	Mono-Si
28	0.117 (average) (0.104–0.124)	0.0038 (average) (0.0032–0.0046)	Average of Sandia and commercial cells
25	0.11	0.003	Mono-Si
25	0.13	0.0041	PV/T system
		0.005	
20	0.10	0.004	PV/T system
25	0.10	0.0041	PV/T system
20	0.125	0.004	PV/T system
25		0.0026	a-Si
25	0.13	0.004	Mono-Si
	0.11	0.004	Poly-Si
	0.05	0.0011	a-Si
25	0.178	0.00375	PV/T system
25		0.005	Mono-Si
25	0.12	0.0045	Mono-Si
25	0.097	0.0045	PV/T system
25		0.0045	PV/T system
25	0.0968	0.0045	
		0.005	UTC/PV system
25	0.09	0.0045	PV/T system
25	0.12	0.0045	PV/T system
25		0.0045 c-Si 0.0020 a-Si	PV/T system
25	0.12	0.0045	PV/T system
25	0.12	0.0045	PV/T system
25	0.127	0.0063	PV/T system
25	0.127 unglazed 0.117 glazed	0.006	PV/T system
25		0.0054	PV/T system

Thermal Model

The heat transfer equations which have been driven from the model can be seen below through equations number 2 to 15 (1, 2).

$$q_{ins} = h_{ins}(T_{PV} - T_{amb}) \quad (2)$$

$$q_{r,PV-1} = \frac{1}{\frac{1}{\varepsilon_{glass}} + \frac{1}{\varepsilon_{PV}} - 1}\sigma({T_{PV}}^4 - {T_{g1b}}^4) \quad (3)$$

$$q_{c,PV-1} = h_{1-PV}(T_{PV} - T_{g1b}) \quad (4)$$

$$q_{glass,1} = \frac{-K_{glass}}{\delta_{glass,1}}(T_{g1} - T_{PV}) \quad (5)$$

$$q_{htf,in} = h_{conv}(T_{g1} - T_{htf,ave}) \quad (6)$$

$$q_{HTF} = \dot{m}c_p(T_{HTF,out} - T_{HTF,in}) \quad (7)$$

$$q_{htf,out} = h_{conv}(T_{htf,ave} - T_{g2b}) \quad (8)$$

$$q_{glass,2} = \frac{-K_{glass}}{\delta_{glass,2}}(T_{g2a} - T_{g2b}) \quad (9)$$

$$q_{r,1-2} = \frac{1}{\frac{1}{\varepsilon_{glass}} + \frac{1}{\varepsilon_{PV}} - 1}\sigma({T_{PV}}^4 - {T_{g2a}}^4) \quad (10)$$

$$q_{r,2-3} = \frac{\varepsilon_{glass}}{2-\varepsilon_{glass}}\sigma({T_{g2a}}^4 - {T_{g3b}}^4) \quad (11)$$

$$q_{c,2-3} = h(T_{g2a} - T_{g3b}) \quad (12)$$

$$q_{glass,3} = \frac{-K_{glass}}{\delta_{glass,3}}(T_{g3a} - T_{g3b}) \quad (13)$$

$$q_{r,amb} = \varepsilon_{glass,3}\sigma({T_{g3a}}^4 - T_{amb}^4) \quad (14)$$

$$q_{c,amb} = h_{wind}(T_{g3a} - T_{amb}) \quad (15)$$

The equations 16 to 23 represent the energy balance equations for the model.

$$\tau_{fluid}\tau_{sys}\alpha_{PV}(1-\eta_{PV})CG = q_{ins} + q_{r,PV-1} + q_{c,PV-1} \quad (16)$$

$$q_{r,PV-1} + q_{c,PV-1} = q_{glass,1} \quad (17)$$

$$q_{glass,1} = q_{HTF,in} + q_{r,1-2} \quad (18)$$

$$q_{HTF,in} + \tau_{g2}\tau_{g1}\alpha_{HTF}CG + \alpha_{HTF}q_{r,1-2} = q_{HTF,out} + q_{HTF} \quad (19)$$

$$q_{HTF,out} + (1-\alpha_{HTF})q_{r,1-2} = q_{glass,2} \quad (20)$$

$$q_{glass,2} = q_{r,2-3} + q_{c,2-3} \quad (21)$$

$$q_{r,2-3} + q_{c,2-3} = q_{glass,3} \quad (22)$$

$$q_{glass,3} = q_{r,amb} + q_{c,amb} \quad (23)$$

By substituting the equations of heat fluxes from equations 2 to 15 and PV efficiency from equation 1 into energy balance equations we get 8 equations and eight unknowns. Thus, there is a nonlinear system of equations with eight equations and eight unknowns which should be solved.

Solving Systems of Non-linear Equations

Consider the solution to a system of *n* non-linear equations in *n* unknowns given by

f1(x1, x2,…,xn) = 0
f2(x1,x2,…,xn) = 0
.
.
.
fn(x1,x2,…,xn) = 0

The system can be written in a single expression using vectors, i.e.

f(**x**) = 0,

Where the vector **x** contains the independent variables, and the vector **f** contains the functions *fi(***x***):*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, ..., x_n) \\ f_2(x_1, x_2, ..., x_n) \\ \vdots \\ f_n(x_1, x_2, ..., x_n) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}.$$

Newton-Raphson Method to Solve Systems of Non-linear Equations

A Newton-Raphson method for solving the system of linear equations requires the evaluation of a matrix, known as the Jacobian of the system, which is defined as:

$$\mathbf{J} = \frac{\partial(f_1, f_2, ..., f_n)}{\partial(x_1, x_2, ..., x_n)} = \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \cdots & \partial f_1/\partial x_n \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \cdots & \partial f_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \cdots & \partial f_n/\partial x_n \end{bmatrix} = [\frac{\partial f_i}{\partial x_j}]_{n\times n}$$

If **x** = **x**0 (a vector) represents the first guess for the solution, successive approximations to the solution are obtained from

$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}^{-1}\cdot\mathbf{f}(\mathbf{x}_n) = \mathbf{x}_n - \Delta\mathbf{x}_n$.

With:

$\Delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$

A convergence criterion for the solution of a system of non-linear equation could be, for example, that the maximum of the absolute values of the functions *fi(***x**n*)* is smaller than a certain tolerance ε, i.e.,

$$\max_i |f_i(\mathbf{x}_n)| < \varepsilon$$

Another possibility for convergence is that the magnitude of the vector **f**(**x**n) be smaller than the tolerance, i.e.,

$|\mathbf{f}(\mathbf{x}_n)| < \varepsilon$

We can also use as convergence criteria the difference between consecutive values of the solution, i.e.,

$$\max_i |(x_i)_{n+1} - (x_i)_n| < \varepsilon.$$

Or

$|\Delta\mathbf{x}_n| = |\mathbf{x}_{n+1} - \mathbf{x}_n| < \varepsilon$

The main complication with using Newton-Raphson to solve a system of non-linear equations is having to define all the functions *∂fi/∂xj*, for *i,j = 1,2, …, n*, included in the Jacobian. As the number of equations and unknowns, *n*, increases, so does the number of elements in the Jacobian, *n2* (4).

Matlab Code

A Matlab code (annex A) was developed based on the Newton-Raphson Method to solve the system of equations of the model. The Solar flux, concentration ratio and heat transfer fluid temperature at outlet and inlet are the inputs and the code solves the system of equations for unknown temperatures at different parts of the system as well as the mass flow rate of the heat transfer fluid.

The code returns the PV efficiency based on equations 1 and thermal efficiency through equations 24 as outputs.

$$\eta_{Thermal} = \frac{q_{HTF}}{C * G} \qquad (24)$$

RESULTS AND DISCUSSION

To make sure if the code is robust and fast enough to solve variety of cases, some potential cases were considered and the code was run (Table 2). Elapsed time after running the code is in the order of 0.0008 seconds that means the code is fast enough to solve the system of equations for input different conditions.

Table 2: PV and thermal efficiencies as well as mass flow rate at different input conditions (G=900.7 [W/m^2])

C	T_HTF_OUT	T_HTF_IN	PV Efficiency	Thermal Efficiency	Mass Flow Rate
1	302	300	0.1499	0.4680	0.1343
5	313	300	0.1494	0.4734	0.1035
10	326	300	0.1489	0.4742	0.1025
20	351	300	0.1477	0.4746	0.1024
50	421	300	0.1443	0.4748	0.1017
100	524	300	0.1385	0.4749	0.1013
200	698	300	0.1268	0.4754	0.1016

Effect of Initial Guess

To investigate if the initial guess affects the final results the code was run using different initial guesses for the last case in table 2. Table 3 shows the different initial guesses which have been applied to code for this case. Same final answers for each variable as well as PV and thermal efficiencies for each initial guess were obtained and just the number of iteration which leads to convergence is different for different initial guesses. It should be mentioned that the code does not converge for some initial guesses like sixth guess in table 3.

Table 3: Different Initial guesses and number of iteration to converge

Variable	T_{g3a}	T_{g3b}	T_{g2a}	T_{g2b}	m·	T_{g1a}	T_{g1b}	T_{PV}	# Iteration
First Guess	300	301	302	303	0.1	304	305	306	5
Second Guess	30	30	30	30	1	30	30	30	6
Third Guess	700	700	700	700	10	700	700	700	6
Forth Guess	1	700	100	7000	100	50	12	9	7
Fifth Guess	26	56	690	25	10	456	125	89	6
Sixth Guess	256	179	90	15	100	56	690	800	No
Seventh Guess	300	300	300	300	10	400	690	800	8

Convergence

As it can be seen in table 3 the code converges after 5 to 8 iterations dependent on initial guess. The used convergence criterion for this code is shown as below:

$$\max_i |f_i(\mathbf{x}_n)| < \varepsilon$$

The value of epsilon is 1×10^{-10}, which means the absolute value of each of the energy balance equations, after substituting the obtained value of each related variable, should be less than 1×10^{-10} to get convergence. Figure 3 illustrates the convergence of the nonlinear system of equations for each heat balance equation related to case seven in table 3.

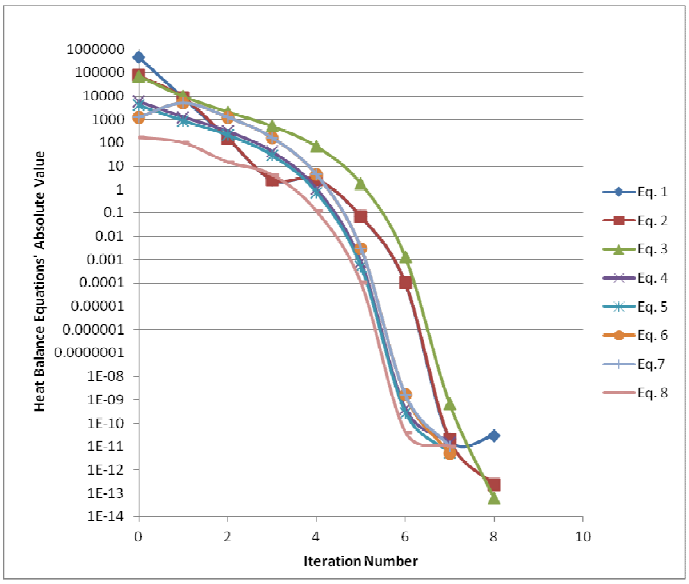

Figure 3: Convergence of heat balance equations.

Real Case Study

To investigate if the code is efficient and fast when the real conditions are applied; based on the US typical meteorological year data sets for insolation at Tulsa international airport some direct normal incident (DNI) radiations were selected to run the code. Table 4 shows the selected conditions as well as obtained data. For all cases the temperature of the heat transfer fluid at inlet was considered as equal as ambient temperature which is 298 K.

DNI (W/m^2)	C (-)	T-HTF-OUT (K)	PV Efficiency (%)	Thermal Efficiency (%)	Iteration (#)	Elapsed Time (s)
377	200	500	14.04	47.47	4	0.018995
364	200	500	14.07	47.45	4	0.019000
46	200	500	14.87	43.15	4	0.000638
46	50	500	14.96	28.42	4	0.019001
46	50	400	14.97	38.81	4	0.019251
415	200	500	13.95	47.52	4	0.019791
415	1	500	14.94	0	4	0.019052

CONCLUSIONS

1. The Newton-Raphson Method can be applied to solve the system of nonlinear equations which governs the coupled photovoltaic/thermal concentrating solar collector for electricity generation.
2. The developed code is fast and applicable in different input conditions.
3. The code returns the PV efficiency and thermal efficiency as well as the mass flow rate of the heat transfer fluid of the modeled system.

NOMENCLATURE

C_p = specific heat, J kg^{-1} K^{-1}
G = solar flux, W m^{-2}

$_m$ = mass flow rate, kg s^{-1}
C = concentration ratio, –
α = absorptivity, –
ε = emissivity, –
τ = transmissivity, -
τ_{sys} = system transmissivity, –
σ = Stefan-Boltzmann constant, W m^{-2} K^{-4}
δ = thickness, m
q = heat flux, W m^{-2}
T = temperature, K
k = thermal conductivity, W m^{-1} K^{-1}
h = heat transfer coefficient, W m^{-2} K^{-1}

ACKNOWLEDGMENTS

I would like to appreciate my professor Dr. Todd Otanicar for his instructions and recommendations during accomplishment of this project.

REFERENCES

1. Todd P. Ihtesham Chowdhury, Ravi Prasher and Patrick E. Phelan, "*Band-Gap Tuned Direct Absorption for a Hybrid Concentrating Solar Photovoltaic/Thermal System*", Journal of Solar Energy Engineering, NOVEMBER 2011, Vol. 133 / 041014-1.
2. Todd P. Ihtesham Chowdhury, Patrick E. Phelan and Ravi Prasher, "*Parametric analysis of a coupled photovoltaic/thermal concentrating solar collector for electricity generation*", JOURNAL OF APPLIED PHYSICS 108, 114907 _2010.
3. E. Skoplaki, J.A. Palyvos, "*On the temperature dependence of photovoltaic module electrical performance: A review of efficiency/power correlations*", Solar Energy 83 (2009) 614–624.
4. Gilberto E. Urroz, "*Solution of non-linear equations*", September 2004.

ANNEX A

```
tic
% Calculating the PV efficiency and thermal efficiency for the PV(C-Si)/Thermal systems.

clc
clear

% INPUTS:

% Ambient Temperature [K]
Tamb=298;
% Fluid  Temperature In [K]
Thtfin=298;
% Fluid  Temperature Out [K]
Thtfout=500;
%Concentration Ratio [-]
C=100;
% Solar Flux [W/m^2]
G=415;

% SOLAR CELL Efficeiny PARAMETERS:

%Reference Temperature [K]
Tref=298;
%Module's Electrical Efficiency at Reference Temperature [-]
Eref=0.15;
%Temperature Coefficient [1/K]
Bref=0.0041;

%Thermal/Radiative Properties:

%Wind Heat Transfer Coefficient [[W/( m^2 *K)]
hwind=10;
%Ambient Temperature [K]
Tamb=298;
%Emissivity of Glass 3 [ ]
eg3=0.9;
%Stefan-Boltzmann Constant [W/(m^2*K^4)]
s=5.6704E-08;
%Thermal Conductivity of Glass [W/(m*K)]
Kg=0.8;
%Thickness of Glass 3 [m]
dg3=0.003;
%Heat Transfer Coefficient Between Glass 2 and 3 [[W/( m^2 *K)]
h=0;
%Thickness of Glass 2 [m]
dg2=0.003;
%Nusselt Number [ ]
Nu=8.23;
%Thickness of Heat Transfer Fluid [m]
dhtf=0.2;
%Tranmissivity of Heat Transfer Fluid [ ]
```

```
tf=0.407;
%Absorptivity of Heat Transfer Fluid [ ]
ahtf=1-tf;
%Emissivity of Glass [ ]
eg=0.9;
%Emissivity of P.V. [ ]
ep=0.35;
%Tranmissivity of Glass 2 [ ]
tg2=0.9;
%Tranmissivity of Glass 3 [ ]
tg3=0.9;
%Heat Transfer Coefficient Between P.V. and Glass 1 [[W/( m^2 *K)]
h1pv=0;
%Thickness of Glass 1 [m]
dg1=0.003;
%Tranmissivity of system [ ]
tsys=tg2*tg3*tg3;
%Absorptivity of P.V. [ ]
apv=0.8;
%Heat Transfer Coefficient Between air and P.V. [[W/( m^2 *K)]
hins=1000;
%Thickness Between Glass 2 and 3 [m]:
dg23=0.025;


N=100;
epsilon=1e-10;
maxval=10000;

%First guess
x0=[300; 300; 300; 300; 1; 300; 300; 300];

x=x0;

while (N>0)

Tg3a=x(1,1);
Tg3b=x(2,1);
Tg2a=x(3,1);
Tg2b=x(4,1);
m=x(5,1);
Tg1a=x(6,1);
Tg1b=x(7,1);
Tpv=x(8,1);

%Defining the Function EBE:
ebe1=tf*tsys*apv*C*G*(1-(Eref*(1-Bref*(Tpv-Tref))))-1/(1/eg+1/ep-1)*s*(Tpv^4-Tg1b^4)-h1pv*(Tpv-Tg1b)-
hins*(Tpv-Tamb);
ebe2=1/(1/eg+1/ep-1)*s*(Tpv^4-Tg1b^4)+h1pv*(Tpv-Tg1b)+Kg/dg1*(Tg1a-Tg1b);
ebe3=Nu*(-0.0000819477*((Thtfout+Thtfin)/2-273)-0.000000192257*((Thtfout+Thtfin)/2-
273)^2+0.000000000025034*((Thtfout+Thtfin)/2-273)^3-0.0000000000000072974*((Thtfout+Thtfin)/2-
273)^4+0.137743)/dhtf*(Tg1a-(Thtfout+Thtfin)/2)+1/(1/eg+1/ep-1)*s*(Tpv^4-Tg2a^4)+Kg/dg1*(Tg1a-Tg1b);
ebe4=Nu*(-0.0000819477*((Thtfout+Thtfin)/2-273)-0.000000192257*((Thtfout+Thtfin)/2-
273)^2+0.000000000025034*((Thtfout+Thtfin)/2-273)^3-0.0000000000000072974*((Thtfout+Thtfin)/2-
273)^4+0.137743)/dhtf*(Tg1a-(Thtfout+Thtfin)/2)+tg2*tg3*(1-tf)*C*G+(1-tf)/(1/eg+1/ep-1)*s*(Tpv^4-Tg2a^4)-
```

```
Nu*(-0.0000819477*((Thtfout+Thtfin)/2-273)-0.000000192257*((Thtfout+Thtfin)/2-
273)^2+0.000000000025034*((Thtfout+Thtfin)/2-273)^3-0.0000000000000072974*((Thtfout+Thtfin)/2-
273)^4+0.137743)/dhtf*((Thtfout+Thtfin)/2-Tg2b)-m*1000*(0.002414*((Thtfout+Thtfin)/2-
273)+0.0000059591*((Thtfout+Thtfin)/2-273)^2-0.000000029879*((Thtfout+Thtfin)/2-
273)^3+0.000000000044172*((Thtfout+Thtfin)/2-273)^4+1.498)*(Thtfout-Thtfin);
ebe5=Nu*(-0.0000819477*((Thtfout+Thtfin)/2-273)-0.000000192257*((Thtfout+Thtfin)/2-
273)^2+0.000000000025034*((Thtfout+Thtfin)/2-273)^3-0.0000000000000072974*((Thtfout+Thtfin)/2-
273)^4+0.137743)/dhtf*((Thtfout+Thtfin)/2-Tg2b)+(1-ahtf)/(1/eg+1/ep-1)*s*(Tpv^4-Tg2a^4)+Kg/dg2*(Tg2a-
Tg2b);
ebe6=h*(Tg2a-Tg3b)+eg/(2-eg)*s*(Tg2a^4-Tg3b^4)+Kg/dg2*(Tg2a-Tg2b);
ebe7=h*(Tg2a-Tg3b)+eg/(2-eg)*s*(Tg2a^4-Tg3b^4)+Kg/dg3*(Tg3a-Tg3b);
ebe8=hwind*(Tg3a-Tamb)+eg3*s*(Tg3a^4-Tamb^4)+Kg/dg3*(Tg3a-Tg3b);

EBE=[ebe1; ebe2; ebe3; ebe4; ebe5; ebe6; ebe7; ebe8];

J(1,1)=0;
J(1,2)=0;
J(1,3)=0;
J(1,4)=0;
J(1,5)=0;
J(1,6)=0;
J(1,7)=h1pv + (4*Tg1b^3*s)/(1/eg + 1/ep - 1);
J(1,8)=Bref*C*Eref*G*apv*tf*tsys - hins - (4*Tpv^3*s)/(1/eg + 1/ep - 1) - h1pv;
J(2,1)=0;
J(2,2)=0;
J(2,3)=0;
J(2,4)=0;
J(2,5)=0;
J(2,6)=Kg/dg1;
J(2,7)=- h1pv - Kg/dg1 - (4*Tg1b^3*s)/(1/eg + 1/ep - 1);
J(2,8)=h1pv + (4*Tpv^3*s)/(1/eg + 1/ep - 1);
J(3,1)=0;
J(3,2)=0;
J(3,3)=-(4*Tg2a^3*s)/(1/eg + 1/ep - 1);
J(3,4)=0;
J(3,5)=0;
J(3,6)=Kg/dg1 - (Nu*((6046672997316513*Thtfin)/147573952589676412928 +
(6046672997316513*Thtfout)/147573952589676412928 + (7263264103176555*(Thtfin/2 + Thtfout/2 -
273)^2)/37778931862957161709568 - (242113991745861*(Thtfin/2 + Thtfout/2 -
273)^3)/96714065569170333976494 08 + (4625276745052741*(Thtfin/2 + Thtfout/2 -
273)^4)/63382530011411470074835 1602688 - 11814381204047307441/73786976294838206464))/dhtf;
J(3,7)=-Kg/dg1;
J(3,8)=(4*Tpv^3*s)/(1/eg + 1/ep - 1);
J(4,1)=0;
J(4,2)=0;
J(4,3)=(4*Tg2a^3*s*(tf - 1))/(1/eg + 1/ep - 1);
J(4,4)=-(Nu*((6046672997316513*Thtfin)/147573952589676412928 +
(6046672997316513*Thtfout)/147573952589676412928 + (7263264103176555*(Thtfin/2 + Thtfout/2 -
273)^2)/37778931862957161709568 - (242113991745861*(Thtfin/2 + Thtfout/2 -
273)^3)/96714065569170333976494 08 + (4625276745052741*(Thtfin/2 + Thtfout/2 -
273)^4)/63382530011411470074835 1602688 - 11814381204047307441/73786976294838206464))/dhtf;
J(4,5)=1000*(Thtfin - Thtfout)*((5566305024241857*Thtfin)/4611686018427387904 +
(5566305024241857*Thtfout)/4611686018427387904 + (3517631763508563*(Thtfin/2 + Thtfout/2 -
273)^2)/590295810358705651712 - (1128796705133297*(Thtfin/2 + Thtfout/2 -
```

```
273)^3)/3777893186295716170956 8 + (6835285926914227*(Thtfin/2 + Thtfout/2 -
273)^4)/154742504910672534362390528 + 241818944523010822387/288230376151711744000);
J(4,6)=-(Nu*((6046672997316513*Thtfin)/147573952589676412928 +
(6046672997316513*Thtfout)/147573952589676412928 + (7263264103176555*(Thtfin/2 + Thtfout/2 -
273)^2)/3777893186295716170956 8 - (242113991745861*(Thtfin/2 + Thtfout/2 -
273)^3)/9671406556917033397649408 + (4625276745052741*(Thtfin/2 + Thtfout/2 -
273)^4)/6338253001141147007483516026 88 - 11814381204047307441/737869762948382064 64))/dhtf;
J(4,7)=0;
J(4,8)=-(4*Tpv^3*s*(tf - 1))/(1/eg + 1/ep - 1);
J(5,1)=0;
J(5,2)=0;
J(5,3)=Kg/dg2 + (4*Tg2a^3*s*(ahtf - 1))/(1/eg + 1/ep - 1);
J(5,4)=(Nu*((6046672997316513*Thtfin)/147573952589676412928 +
(6046672997316513*Thtfout)/147573952589676412928 + (7263264103176555*(Thtfin/2 + Thtfout/2 -
273)^2)/3777893186295716170956 8 - (242113991745861*(Thtfin/2 + Thtfout/2 -
273)^3)/9671406556917033397649408 + (4625276745052741*(Thtfin/2 + Thtfout/2 -
273)^4)/6338253001141147007483516026 88 - 11814381204047307441/737869762948382064 64))/dhtf - Kg/dg2;
J(5,5)=0;
J(5,6)=0;
J(5,7)=0;
J(5,8)=-(4*Tpv^3*s*(ahtf - 1))/(1/eg + 1/ep - 1);
J(6,1)=0;
J(6,2)=(4*Tg3b^3*eg*s)/(eg - 2) - h;
J(6,3)=h + Kg/dg2 - (4*Tg2a^3*eg*s)/(eg - 2);
J(6,4)=-Kg/dg2;
J(6,5)=0;
J(6,6)=0;
J(6,7)=0;
J(6,8)=0;
J(7,1)=Kg/dg3;
J(7,2)=(4*Tg3b^3*eg*s)/(eg - 2) - Kg/dg3 - h;
J(7,3)=h - (4*Tg2a^3*eg*s)/(eg - 2);
J(7,4)=0;
J(7,5)=0;
J(7,6)=0;
J(7,7)=0;
J(7,8)=0;
J(8,1)=hwind + Kg/dg3 + 4*Tg3a^3*eg3*s;
J(8,2)=-Kg/dg3;
J(8,3)=0;
J(8,4)=0;
J(8,5)=0;
J(8,6)=0;
J(8,7)=0;
J(8,8)=0;

if abs(det(J))<epsilon
   error('Jacobian Matrix is Singular-try new x0');
   abort;
end;

xn=x-inv(J)*EBE;
x=xn;
if abs(EBE)<epsilon
   finalanswer=x
```

```
    iter=100-N
    PVefficiency=Eref*(1-Bref*(finalanswer(8,1)-Tref))
   ThEf=(finalanswer(5,1)*1000*(0.002414*((Thtfout+Thtfin)/2-273)+0.0000059591*((Thtfout+Thtfin)/2-273)^2-
0.000000029879*((Thtfout+Thtfin)/2-273)^3+0.000000000044172*((Thtfout+Thtfin)/2-273)^4+1.498)*(Thtfout-
Thtfin))/(C*G);
    if  (0 < ThEf)
     Thermalefficiency =ThEf
    else Thermalefficiency=0
    end
    toc
    return;
end;

if abs(EBE)>maxval
    iter=100-N;
    disp(['iterations=',num2str(iter)]);
    error ('solution divergence');
    abort;
end;

N=N-1;
x=xn;
end;
error('no convergence after 100 iterations.');
abort;
```