



VIETNAMESE-GERMAN UNIVERSITY

(CSE-ECE)

**FINAL REPORT**

**of**

**………………Group 3………………..**

**A Hybrid PSO and GSA-Based Maximum Power Point Tracking Algorithm for PV Systems**

BY

Huỳnh Nguyễn Chí Hiếu –Nguyễn Đặng Phúc – Phạm Thanh Quang

(17523 – 10221038 - 10221073)

(Supervisor BUI MINH DUONG from VGU)

### Abstract

This study proposes a hybridization of particle swarm optimization (PSO) and gravitational search algorithm (GSA) for maximum power point tracking (MPPT) in photovoltaic (PV) systems. The purpose of this research is to integrate the exploitation ability of PSO with the exploration ability of GSA to synthesize the strengths of both algorithms. The main objective is to reduce oscillation once the maximum power point (MPP) is located. The study employs MATLAB-SIMULINK simulations to evaluate the effectiveness of the proposed methodology, specifically under step changes in irradiance of the PV array.

In theory, the hybrid algorithm should exhibit a greater capacity for excaping local maxima and achieve faster convergence compared to conventional PSO and GSA methods, but the actual simulation results contradict with the expectaion.

### Preface

Renewable energy sources have gained significant importance in recent years due to the growing concerns about environmental sustainability and the need to reduce dependence on fossil fuels. Among these sources, solar power systems have emerged as a promising solution for generating clean and sustainable electricity. However, the efficiency of solar photovoltaic (PV) systems is greatly influenced by the varying environmental conditions, which pose challenges in maximizing their power output.

To address these challenges, researchers and engineers have been developing and refining maximum power point tracking (MPPT) algorithms. These algorithms aim to optimize the operation of PV systems by continuously tracking and maintaining the maximum power point (MPP) under changing conditions. The effectiveness of MPPT techniques plays a crucial role in enhancing the overall performance and efficiency of solar power systems.

This document provides a comprehensive overview of the hybrid PSO-GSA MPPT algorithm, along with its design procedures and simulation results. It serves as a valuable resource for researchers, engineers, and professionals involved in the field of solar power systems, offering insights into the development of advanced MPPT techniques and their practical implementation.

### Table of Contents

**I.** [**Abstract**](#_heading=h.gjdgxs)

**II.** [**Preface**](#_heading=h.366zi3ufvp9t)

**III.** [**Table of contents**](#_heading=h.e49lx6xdo8x7)

**IV.** [**Table of figures**](#_heading=h.7ivgxm5pvexz)

**V.** [**Body of the final Report**](#_heading=h.uokhlp55chvk)

1. [Introduction](#_heading=h.oxwot2ww82e4)
2. [Simulation model for the PV Module](#_heading=h.7cwd6cnd6a1n)
3. [General Overview of PSO](#_heading=h.u0dtwaghzngt)
4. [General Overview of GSA](#_heading=h.r62vz2hk6ly)
5. [General Overview of PSOGSA](#_heading=h.60v5f8c8nbpv)
6. [Implementation](#_heading=h.fn3owlnlpldo)
7. [Simulation Results](#_heading=h.6b9fdrrj4zc4)
8. [Conclusion](#_heading=h.171roxym1m5u)
9. [Appendix](#_heading=h.h6rma3ejtx4a)

[PSO.m](#_heading=h.ej56lhvs74ta)

[GSA.m](#_heading=h.y2dh2fyut9xk)

[PSOGSA.m](#_heading=h.y841v14i0o2d)

**VI.** [**References**](#_heading=h.yfj4kfvg6xz)

### Table of Figures

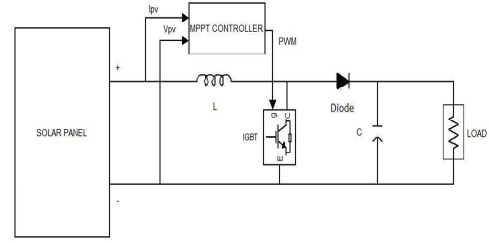
| Figure 1: | [Model layout for the PV system connected boost converter with MPPT controller](#_heading=h.7bsneifrn7gn) |
| --- | --- |
| Figure 2: | [Simulation model for the PV system](#_heading=h.1faamvm57dh2) |
| Figure 3: | [Tracking voltage, current, duty cycle, and power of PSO based MPPT method.](#_heading=h.ymm9irfv7l7t) |
| Figure 4: | [Tracking voltage, current, duty cycle, and power of GSA based MPPT method.](#_heading=h.nkdg591vbncq) |
| Figure 5: | [Tracking voltage, current, duty cycle, and power of PSOGSA based MPPT method.](#_heading=h.ifa35t451fk4) |

### Body of the Final Report

### 1. Introduction

Due to the occurrence of protection failures during natural disasters, nuclear power plants have experienced shutdowns. As a result, there has been a significant shift towards renewable energy sources, with solar power systems garnering considerable attention due to their ability to generate electricity in a pollution-free and radiation-free manner [1]. However, solar energy systems face challenges due to the varying environmental conditions, leading to nonlinear variations in the maximum power point (MPP) in the P-V characteristic curve [2]. Therefore, maximum power point tracking (MPPT) methods are employed to optimize the output power of photovoltaic (PV) arrays by continuously tracking the MPP under different operating conditions. This study proposes the adoption of a hybrid algorithm combining particle swarm optimization (PSO) and gravitational search algorithm (GSA) for MPPT. The design procedures for the hybrid algorithm will be presented, and simulations will be conducted to demonstrate the effectiveness and validity of the proposed MPPT algorithm.

### 2. Simulation model for the PV system

**

#### Figure 1. Model layout for the PV system connected boost converter with MPPT controller.

This model includes a Solar Panel, an MPPT controller system with the inputs are Ipv and Vpv and the outputs are PWM which connect to IGBT/Diode, a single inductor with a capacitor. The MPPT controller is used in solar panel systems to optimize power generation. It continuously monitors and adjusts the operating parameters to ensure the panels are operating at their maximum power point. This optimization maximizes efficiency, increases energy harvest, and promotes sustainable use of solar power.

### 3. General Overview of PSO

The particle swarm optimization (PSO) is basically developed through the simulation of the social behavior of bird flocking and fish schooling. The PSO algorithm maintains a swarm of individuals (called particles), where each particle represents a candidate solution. Particles follow the success of neighboring particles and their own achieved successes. Thus, the position of a particle is therefore influenced by the best particle in a neighborhood (Pbesti, ) and the best solution found by the particle itself (Gbest ).

Particle position (i.e.Duty cycle), is updated by :

1. [1]

The velocity of each particle simulate the moving behavior:

1. [1]

where   
w is the inertia weight  
c1, is cognitive coefficient and c2 is the social coefficient  
r1, r2 ∈ U(0,1) are random numbers  
Pbesti is the personal best position of particle i   
Gbest is the best position of the particles

### 4. General Overview of GSA

The gravitational search algorithm is based on the law of gravity and the notion of mass interactions. The GSA algorithm uses the Newtonian physics theory and its searcher agents are the collection of masses. Newton’s second law says that when a force F is applied to a mass, its acceleration only depends on the force and its mass M.

(3) [1]

From this,the acceleration presents the moving direction of each searcher.   
The velocity and the position could be updated by:

(4) [1]

(5) [1]

where

*randi* is a random variable in the interval [0, 1].

is the current acceleration of the i-th search agent.

is the velocity of the i-th search agent at k-th iteration.

is the position of the i-th search agent at k-th iteration.

### 5. General Overview of PSOGSA

The basic idea of PSOGSA is to combine the ability of social thinking in PSO with the local search capability of GSA. In order to combine these algorithms, (4) is proposed as follow:

(6) [1]

where is the acceleration of agent i at iteration k is given by:

(7) .[1]

where the mass *Fi(t)* is calculated as follows:

(8) .[2]

where the force between i-th particle and j-th particle calculated by

(9) [2]

where

is the gravitational constant

represents the gravitational mass acting on particle j

denotes the gravitational mass acting on particle i.

ɛ = 2.2204\*10^-16

The mass *Mi(t)* is calculated as follows:

(10) [1] with (12) [1]

where

*qi (t )* is the strength of mass *i* at time *t* and, *w(t )* and *b(t )* are defined as follows:

(13) *b(t) =* min*(fi)* [1] *and w(t) =* max*(fi)*[1]

where

*fi* is the fitness function.

w(t) is the worst fitness value at iteration t.

b(t) is the best fitness value at iteration t.

### 6. Implementation

The structure of our algorithm implementation contains 4 main parts: declaration, delay,  
searching, and collective optimization.

a) Declaration:

In this section, we try to declare all the necessary parameters for algorithm such as:  
initialize the number of particles,   
randomize the initial position for each particle,   
and initialize the array of power for the corresponding position.  
assign initial velocities equal to 0,   
initialize the best local and global position, GSA and Hybrid PSOGSA required the local worst position.  
initialize the array of maximum and minimum power for the corresponding position.

b) Delay:

In this part, we use the common loop for delaying the algorithm receive the next input from the Solar panel:  
if(counter >=1 && counter < 4000)  
D=dcurrent;  
counter= counter+1;  
return;  
end

The number of counters can be varied based on the sample time and performance of the algorithm.

c) Searching:

The main point of this part is to observe the performance of each particle during their searching process.

In each iteration, one particle will be observed until their positions are convergent.

if(u>=1 && u<= num\_agency)

D=dc(u);

dcurrent=D;

counter=1;

return;

d) Collective Optimize:

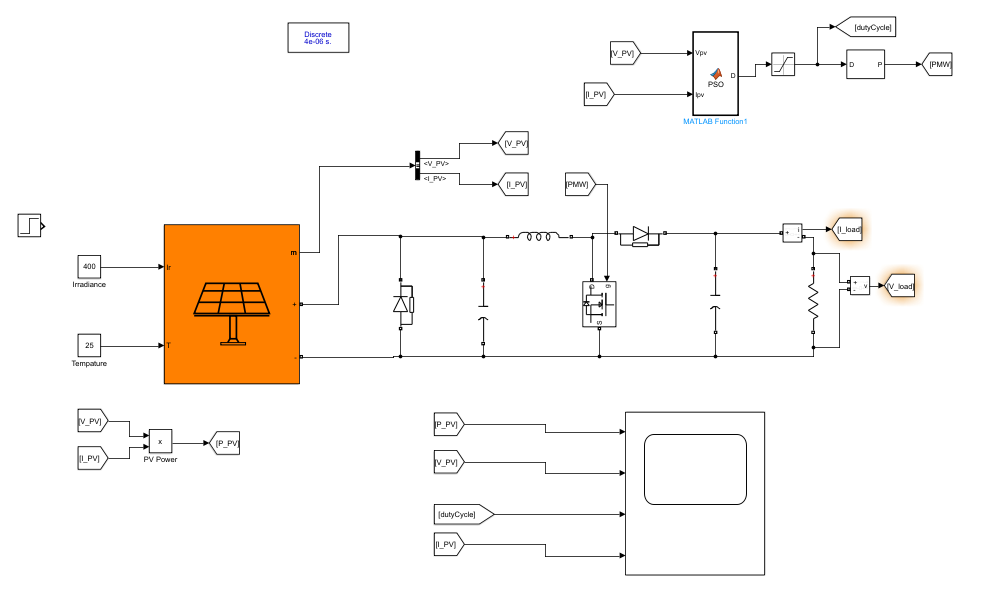
After searching, we will gather the knowledge of the swarm to optimize them by evaluating their performance through achieved power:

[~,i]=max(p);  
gbest=pbest(i);  
D=gbest;

Then update the whole swarm to make sure they converge into the optimal position.

for i=1:num\_agency  
 v(i)=updatevelocity();  
dc(i)=updateduty();  
end

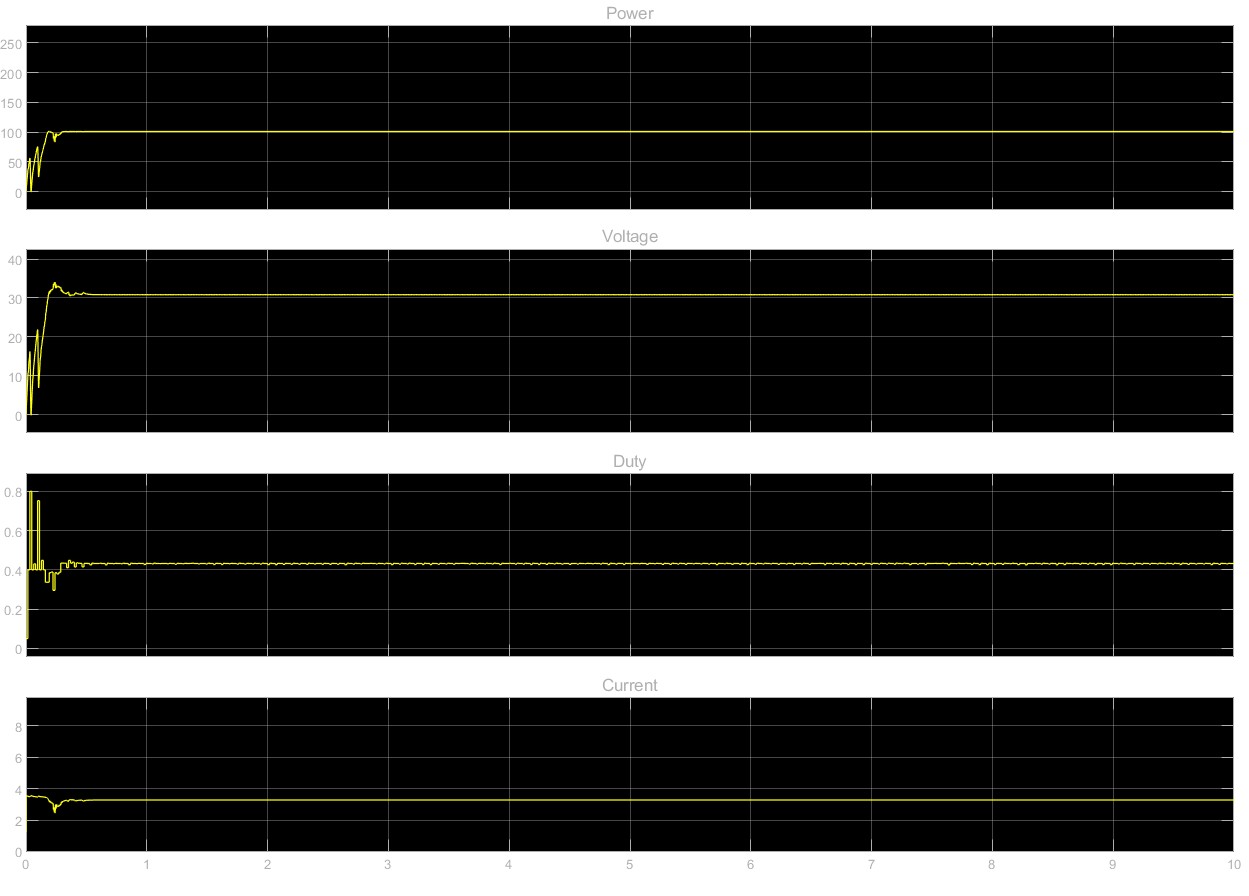
### 7. Simulation Results

****

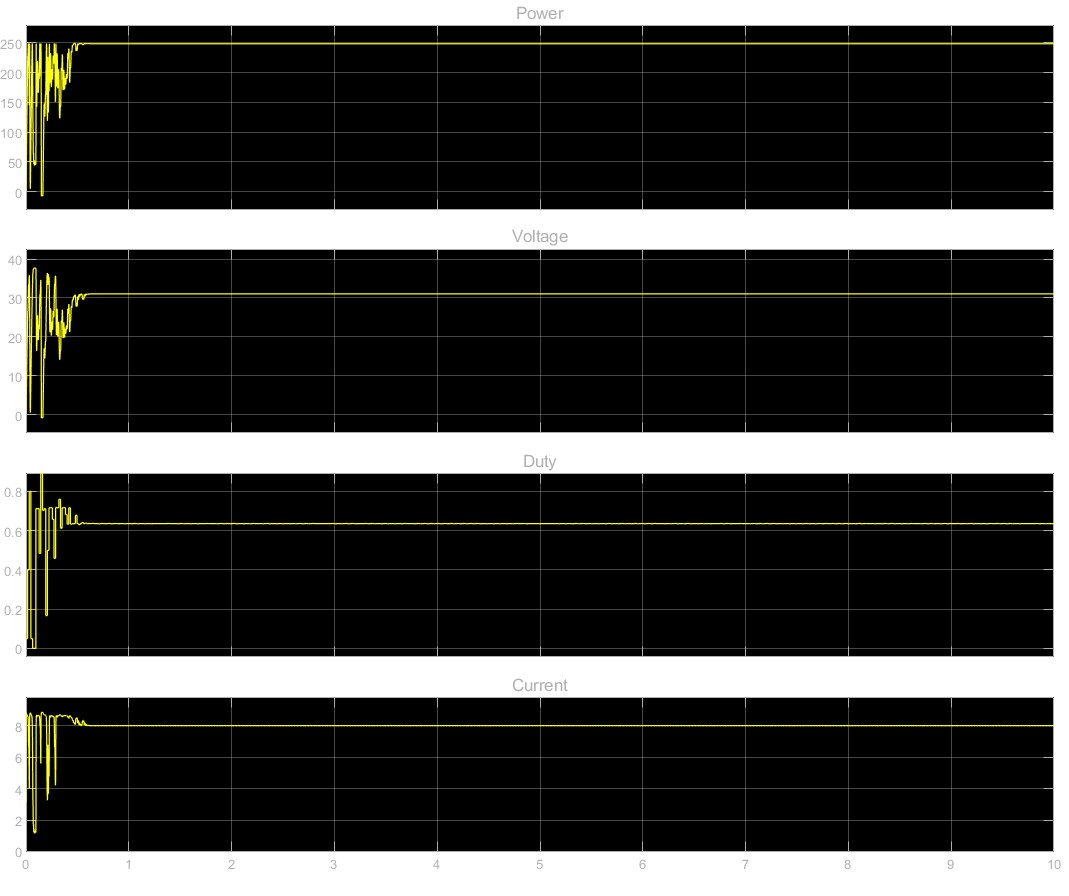
#### Figure 2. Simulation model for the PV system

***Particle Swarm Optimization (PSO)***

Irradiance = 400



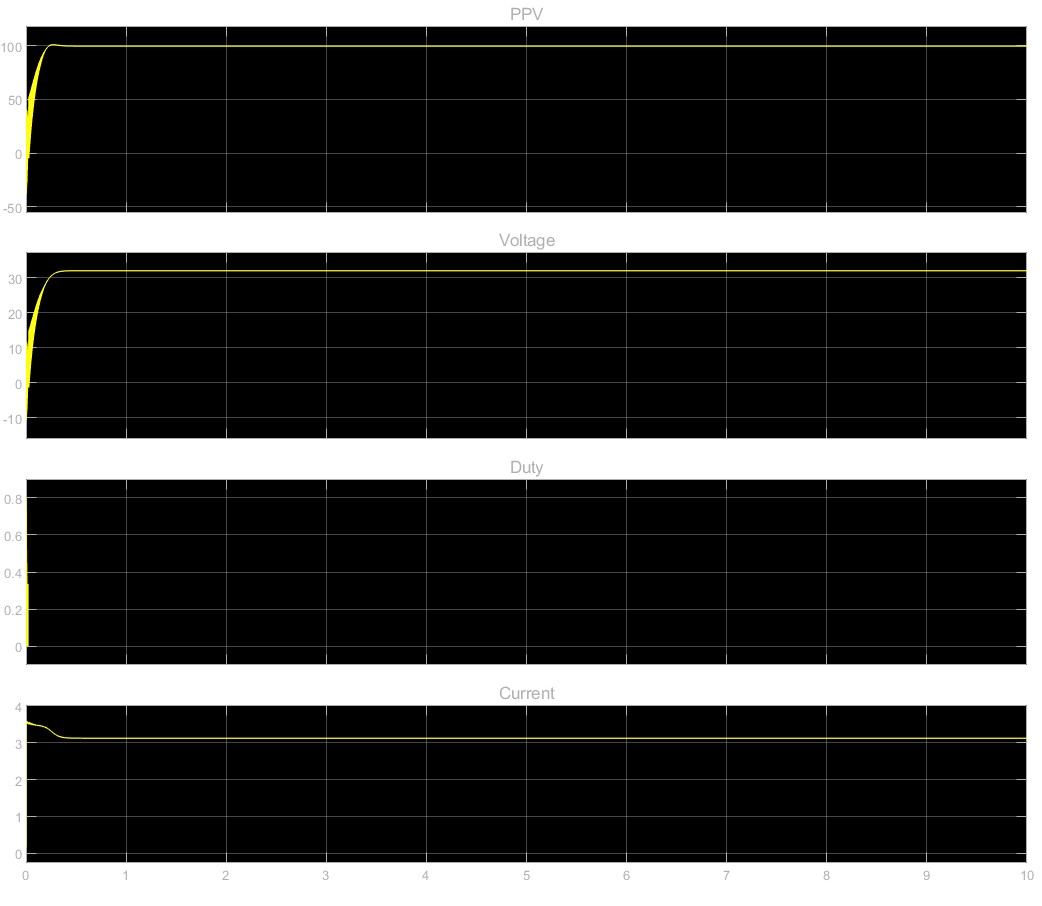
Irradiance = 1000



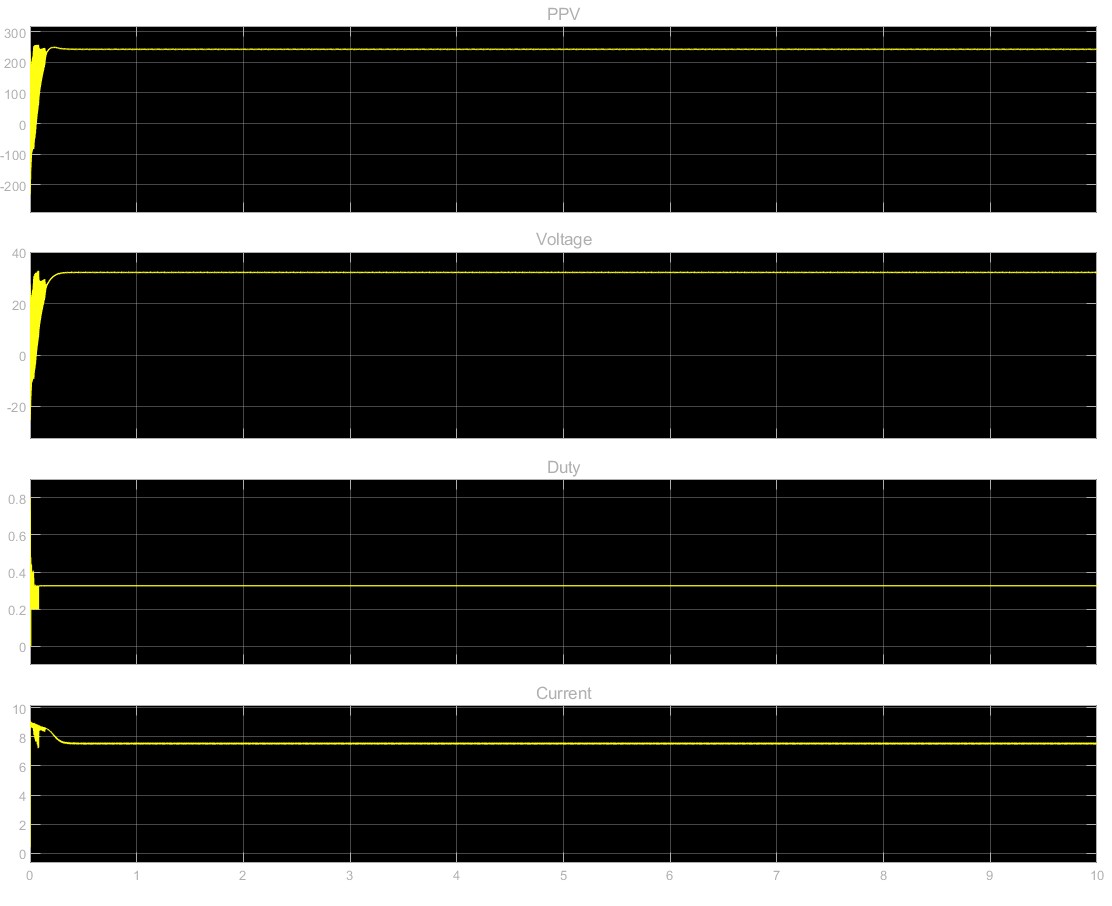
#### Figure 3. Tracking voltage, current, duty cycle, and power of PSO based MPPT method.

***Gravitational Search Algorithm(GSA)***

Irrandiance = 400



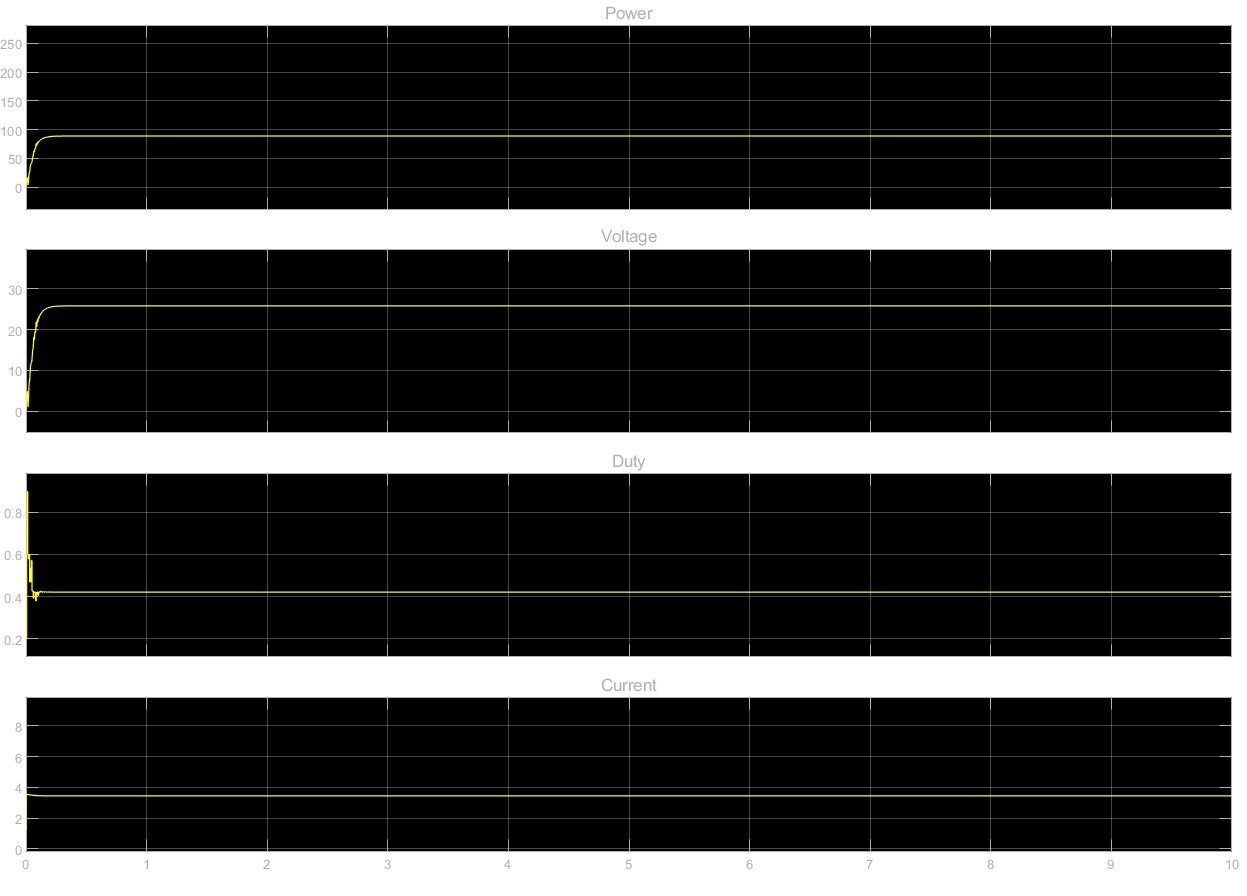
Irradiance = 1000



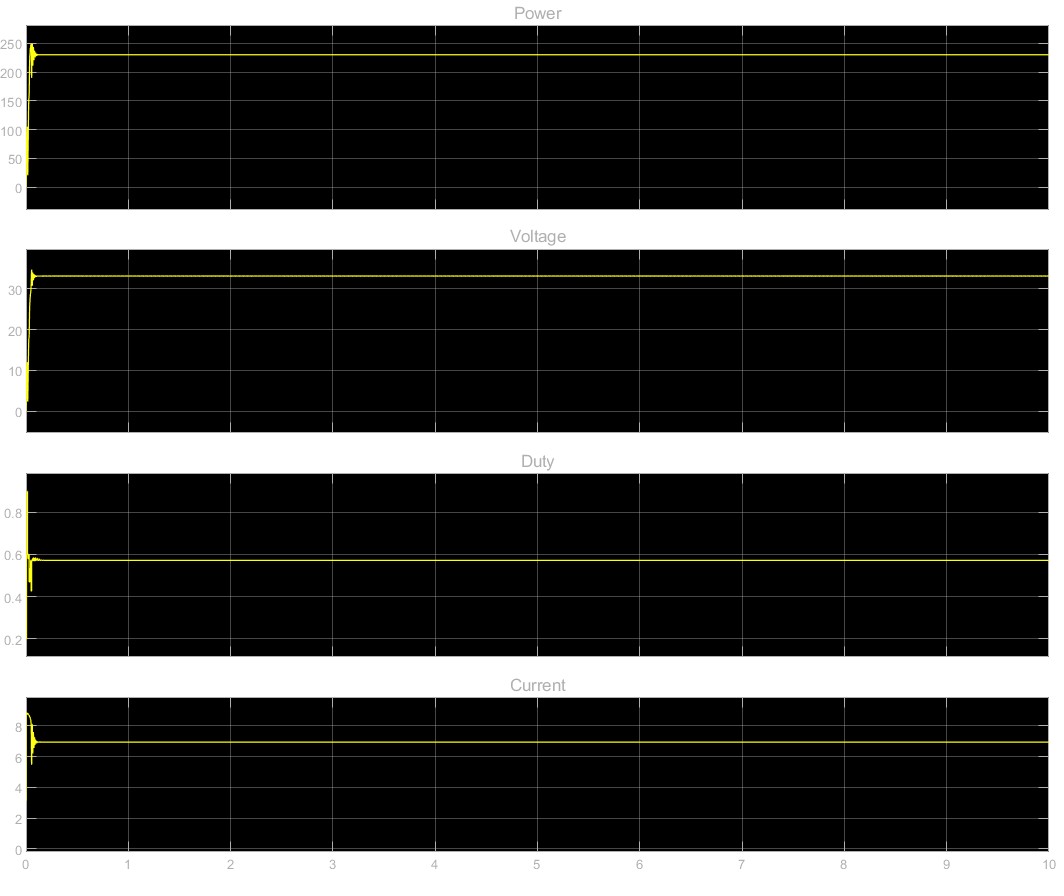
#### Figure 4. Tracking voltage, current, duty cycle, and power of GSA based MPPT method.

PSO-GSA hybrid

400 Irradiance



1000 Irradiance



#### Figure 5. Tracking voltage, current, duty cycle, and power of PSOGSA based MPPT method.

### 8. Conclusion

In theory, the PSOGSA method offers remarkable accuracy and speed compared to conventional PSO and GSA. However, based on actual research, the results show that the PSO, GSA have more efficiency compared to the PSOGSA hybrid and this causes conflict with the paper. The reason for this may come from circuit configuration parameters that have not yet been adapted. Also, there are some limitations in our hybrid structure when it comes to exploiting solar energy resources.

### 

### 9. Appendix

#### PSO.m

function D = PSO(Vpv,Ipv)

%%%CHECKEDDDDDD%%%%%%

persistent u;%u-th particle

persistent dcurrent;%store current duty cycle

persistent pbest;%store local best dc for power

persistent p;% power for each particle

persistent dc;%store duty cycle ~ position

persistent v;%velocity

persistent counter;%delay iteration

persistent gbest;%store global best dc for power

persistent convergence;%check dc convergence status

persistent P\_prev;%store previous power

%initialization

num\_agency = 3;

if(isempty(counter))

counter = 0;

dcurrent=0.5;

gbest=0.9;

p = zeros(num\_agency,1);

v=zeros(num\_agency,1);

pbest = zeros(num\_agency,1);

u=0;

dc=zeros(num\_agency,1);

dc(1)=0.05;

dc(2)=0.4;

dc(3)=0.8;

P\_prev = Ipv\*Vpv;

convergence = 0;

end

if(convergence == 1)

if(abs(Vpv\*Ipv-P\_prev)/(P\_prev + 0.0001) >= 0.3)

counter = 0;

dcurrent=0.5;

gbest=0.9;

p = zeros(num\_agency,1);

v=zeros(num\_agency,1);

pbest = zeros(num\_agency,1);

u=0;

dc=zeros(num\_agency,1);

convergence = 0;

dc(1)=0.05;

dc(2)=0.65;

dc(3)=0.8;

end

end

%%3800

if(counter >=1 && counter < 4000)

D=dcurrent;

counter= counter+1;

return;

end

if(u>=1 && u<= num\_agency)

if((Vpv\*Ipv)>p(u))

p(u) = Vpv\*Ipv;

pbest(u)=dcurrent;

end

end

u=u+1;

if(u==num\_agency+2)

u=1;

end

if(u>=1 && u<= num\_agency)

D=dc(u);

dcurrent=D;

counter=1;

return;

elseif(u==num\_agency+1)

[~,i]=max(p);

gbest=pbest(i);

D=gbest;

dcurrent=D;

counter=1;

for i=1:num\_agency

v(i)=updatevelocity(v(i),pbest(i),dc(i),gbest);

dc(i)=updateduty(dc(i),v(i));

end

P\_prev = Ipv\*Vpv;

convergence = checkconvergence(dc(1),dc(2),dc(3));

return;

else

D=0.5

end

end

function vfinal=updatevelocity(velocity,pobest,d,gwbest)

w=0.1;

c1=1.2;

c2=1.2;

vfinal = (w\*velocity)+(c1\*rand(1)\*(pobest-d))+(c2\*rand(1)\*(gwbest-d));

end

function status = checkconvergence(d1,d2,d3)

status = 0;

rate = 0.05;

if(abs(d1-d2) < rate)

if(abs(d1-d3) < rate)

if(abs(d2-d3) < rate)

status = 1;

end

end

end

end

function dfinal=updateduty(d,velocity)

dup=d+velocity;

if(dup>1)

dfinal=1;

elseif(dup<0.1)

dfinal=0,1;

else

dfinal=dup;

end

end

………………………………………………………………………………………………

#### GSA.m

function D = GSA(Vpv,Ipv)

%%CHECKED%%

persistent u;%u-th particle

persistent dcurrent;%store current duty cycle

persistent pbest;%store local best dc for power

persistent force; %store force

persistent acceleration; %store acceleration

persistent mass; % mass

persistent q; % strength of mass

persistent p; % power for each particle

persistent p\_current; % power current for each particle

persistent p\_min; % power min for each particle

persistent worse; %store best worse of each particle

persistent dc; %store duty cycle ~ position

persistent v; %velocity

persistent counter; %delay iteration

persistent iteration;

persistent gbest;%store global best dc for power

%initialization

num\_agency = 3;

max\_iter = 500;

if(isempty(counter))

counter = 0;

dcurrent = 0.5;

gbest = 0.5;

pbest = zeros(num\_agency,1);

worse = zeros(num\_agency,1);

v = zeros (num\_agency,1);

force = zeros(num\_agency,1);

mass = zeros(num\_agency,1);

q = zeros(num\_agency,1);

p = zeros(num\_agency,1);

p\_current = zeros(num\_agency,1);

p\_min=zeros(num\_agency,1);

p\_min(1)= Vpv\*Ipv;

p\_min(2)= Vpv\*Ipv;

p\_min(3)= Vpv\*Ipv;

acceleration=zeros(num\_agency,1);

u = 0;

dc = zeros (num\_agency,1);

iteration = 1;

%initialize position for each particle

dc(1) = 0.2;

dc(2) = 0.5;

dc(3) = 0.8;

end

if(counter >=1 && counter < 100)

D=dcurrent;

counter= counter+1;

return;

end

if(u>=1 && u<=num\_agency)

p\_current(u) = Vpv\*Ipv;

if((Vpv\*Ipv)>=p(u))

p(u) = Vpv\*Ipv;

pbest(u)=dcurrent;

end

if(Vpv\*Ipv < p\_min(u))

p\_min(u) = Vpv\*Ipv;

worse(u) = dcurrent;

end

end

u=u+1;

if(u== num\_agency + 2)

u=1;

end

if(u >= 1 && u <= num\_agency)

%Avoid over shooting

if(iteration < max\_iter)

D=dc(u);

dcurrent=D;

counter=1;

return;

else

D = dcurrent;

return

end

elseif(u==num\_agency+1)

iteration = iteration +1;

[~,i]=max(p);

gbest=pbest(i);

D=gbest;

dcurrent=D;

counter=1;

%Calculate strength of mass

for i = 1:num\_agency

q(i) = (p\_current(i) - worse(i))/(pbest(i)-worse(i));

end

%Calculate sum of strength of mass

sum\_strength\_of\_mass = q(1) + q(2) + q(3);

%Calculate mass

for i = 1:num\_agency

mass(i) = q(i)/sum\_strength\_of\_mass;

end

%Calculate force

alpha = 20;

G0 = 1;

G = G0 \* exp(-alpha\*iteration/max\_iter);

%G = 6.67430 \* 10^-13; %gravitational constant

e = 2.2204\*10^-16;

force(1) = rand()\*G\*(mass(3)\*mass(1)\*(dc(3)-dc(1))/(Euclidian\_distance(dc(3),dc(1))+e) + mass(2)\*mass(1)\*(dc(3)-dc(1))/(Euclidian\_distance(dc(3),dc(1))+e));

force(2) = rand()\*G\*(mass(3)\*mass(2)\*(dc(3)-dc(2))/(Euclidian\_distance(dc(3),dc(2))+e) + mass(1)\*mass(2)\*(dc(1)-dc(2))/(Euclidian\_distance(dc(1),dc(2))+e));

force(3) = rand()\*G\*(mass(2)\*mass(3)\*(dc(2)-dc(3))/(Euclidian\_distance(dc(2),dc(3))+e) + mass(1)\*mass(3)\*(dc(1)-dc(3))/(Euclidian\_distance(dc(1),dc(3))+e));

%Avoid over shooting

if(iteration > max\_iter)

disp("should done")

D=dcurrent;

return;

end

%Calculate acceleration

for i = 1:num\_agency

acceleration(i) = force(i)/mass(i);

disp(force(i))

end

for i=1:num\_agency

v(i)=updatevelocity(v(i),acceleration(i));

dc(i)=updateduty(dc(i),v(i));

end

return;

else

D=0.5

end

end

function d = Euclidian\_distance(d1,d2)

d = sqrt(d1^2+d2^2);

end

function vfinal=updatevelocity(velocity,acceleration)

vfinal = rand()\*velocity + acceleration;

end

function dfinal=updateduty(d,velocity)

dup=d+velocity;

if(dup>1)

dfinal=1;

elseif(dup<0.1)

dfinal=0,1;

else

dfinal=dup;

end

end

………………………………………………………………………………………………

#### PSOGSA.m

function D = PSOGSA2(Vpv,Ipv)

%%CHECKED TESTING%%

persistent u;%u-th particle

persistent dcurrent;%store current duty cycle

persistent pbest;%store local best dc for power

persistent force; %store force

persistent acceleration; %store acceleration

persistent mass; % mass

persistent q; % strength of mass

persistent p; % power for each particle

persistent p\_current; % power current for each particle

persistent p\_min; % power min for each particle

persistent worse; %store best worse of each particle

persistent dc; %store duty cycle ~ position

persistent v; %velocity

persistent counter; %delay iteration

persistent gbest;%store global best dc for power

persistent convergence;%check dc convergence status

persistent P\_prev;%store previous power

%initialization

num\_agency = 3;

if(isempty(counter))

counter = 0;

dcurrent = 0.5;

gbest = 0.5;

pbest = zeros(num\_agency,1);

worse = zeros(num\_agency,1);

v = zeros (num\_agency,1);

force = zeros(num\_agency,1);

mass = zeros(num\_agency,1);

q = zeros(num\_agency,1);

p = zeros(num\_agency,1);

p\_current = zeros(num\_agency,1);

p\_min=zeros(num\_agency,1);

p\_min(1)= Vpv\*Ipv;

p\_min(2)= Vpv\*Ipv;

p\_min(3)= Vpv\*Ipv;

acceleration=zeros(num\_agency,1);

u = 0;

dc = zeros (num\_agency,1);

P\_prev = Ipv\*Vpv;

convergence = 0;

%initialize position for each particle

dc(1) = 0.2;

dc(2) = 0.6;

dc(3) = 0.9;

end

if(convergence == 1)

if(abs(Vpv\*Ipv-P\_prev) >= 0.3\*P\_prev)

disp("help")

counter = 0;

dcurrent = 0.5;

gbest = 0.5;

pbest = zeros(num\_agency,1);

worse = zeros(num\_agency,1);

v = zeros (num\_agency,1);

force = zeros(num\_agency,1);

mass = zeros(num\_agency,1);

q = zeros(num\_agency,1);

p = zeros(num\_agency,1);

p\_current = zeros(num\_agency,1);

p\_min=zeros(num\_agency,1);

p\_min(1)= Vpv\*Ipv;

p\_min(2)= Vpv\*Ipv;

p\_min(3)= Vpv\*Ipv;

acceleration=zeros(num\_agency,1);

u = 0;

dc = zeros (num\_agency,1);

P\_prev = Ipv\*Vpv;

convergence = 0;

%initialize position for each particle

dc(1) = 0.2;

dc(2) = 0.6;

dc(3) = 0.9;

end

end

%Delay for more visualization

if(counter >=1 && counter < 500)

D=dcurrent;

counter= counter+1;

return;

end

if(u>=1 && u<=num\_agency)

p\_current(u) = Vpv\*Ipv;

if((Vpv\*Ipv)>=p(u))

p(u) = Vpv\*Ipv;

pbest(u)=dcurrent;

end

if(Vpv\*Ipv < p\_min(u))

p\_min(u) = Vpv\*Ipv;

worse(u) = dcurrent;

end

end

u=u+1;

if(u== num\_agency + 2)

u=1;

end

if(u >= 1 && u <= num\_agency)

D=dc(u);

dcurrent=D;

counter=1;

return;

elseif(u==num\_agency+1)

[~,i]=max(p);

gbest=pbest(i);

D=gbest;

dcurrent=D;

counter=1;

%Calculate strength of mass

disp('chia 1');

for i = 1:num\_agency

q(i) = (p\_current(i) - 0.99\*worse(i))/(pbest(i)-worse(i)+ 0.0001);

end

%Calculate sum of strength of mass

sum\_strength\_of\_mass = q(1) + q(2) + q(3);

%Calculate mass

disp('chia 2');

for i = 1:num\_agency

mass(i) = q(i)\*5/sum\_strength\_of\_mass;

end

%Calculate force

G = 6.67430 \* 10^-13; %gravitational constant

e = 2.2204\*10^-16;

disp('chia 3');

force(1) = rand()\*G\*(mass(3)\*mass(1)\*(dc(3)-dc(1))/(Euclidian\_distance(dc(3),dc(1))+e) + mass(2)\*mass(1)\*(dc(3)-dc(1))/(Euclidian\_distance(dc(3),dc(1))+e));

force(2) = rand()\*G\*(mass(3)\*mass(2)\*(dc(3)-dc(2))/(Euclidian\_distance(dc(3),dc(2))+e) + mass(1)\*mass(2)\*(dc(1)-dc(2))/(Euclidian\_distance(dc(1),dc(2))+e));

force(3) = rand()\*G\*(mass(2)\*mass(3)\*(dc(2)-dc(3))/(Euclidian\_distance(dc(2),dc(3))+e) + mass(1)\*mass(3)\*(dc(1)-dc(3))/(Euclidian\_distance(dc(1),dc(3))+e));

%Calculate acceleration

disp('chia 4');

for i = 1:num\_agency

acceleration(i) = force(i)/mass(i);

end

disp('chia 5');

for i=1:num\_agency

v(i)=updatevelocity(v(i),acceleration(i),dc(i),gbest);

dc(i)=updateduty(dc(i),v(i));

end

%disp(P\_prev)

%disp(convergence)

P\_prev = Ipv\*Vpv;

convergence = checkconvergence(dc(1),dc(2),dc(3));

return;

else

D=0.5;

end

end

function status = checkconvergence(d1,d2,d3)

status = 0;

rate = 0.002;

if(abs(d1-d2) < rate)

if(abs(d1-d3) < rate)

if(abs(d2-d3) < rate)

status = 1;

end

end

end

end

function vfinal=updatevelocity(velocity,acceleration,d,gwbest)

w=0.1;

c1=1.2;

c2=1.5;

vfinal = w\*velocity + (rand(1)\*c1\*(acceleration))+(c2\*rand(1)\*(gwbest-d));

end

function d = Euclidian\_distance(d1,d2)

d = sqrt(d1^2+d2^2);

end

function dfinal=updateduty(d,velocity)

dup=d+velocity;

if(dup>1)

dfinal=1;

elseif(dup<0.1)

dfinal=0.1;

else

dfinal=dup;

end

end

### References

[1]B. Goldvin Sugirtha Dhas, S.N. Deepa (2013). ‘A Hybrid PSO and GSA -Based Maximum Power Point Tracking Algorithm for PV Systems’, in Department of Electrical and Electronics Engineering, Anna University, Regional Centre, Coimbatore, India.

[2]Jia Yi Leong, Lenin Gopal, Choo W.R. Chiong, Filbert H. Juwono, Thomas Anung Basuki (2023)  
Hybrid gravitational search particle swarm optimization algorithm for GMPPT under partial shading conditions, Green Technologies and Sustainability,  
Volume 1, Issue 3,100034,ISSN 2949-7361, <https://doi.org/10.1016/j.grets.2023.100034>.

(<https://www.sciencedirect.com/science/article/pii/S2949736123000271>)  
(accessed 3 February 2024)