Spring 4-20-2019

Discrete-Position Solar Tracking for Photovoltaic System

Shengnan Hong
Rose-Hulman Institute of Technology, hongs4@rose-hulman.edu

Zheng Fu
Rose-Hulman Institute of Technology, fuz@rose-hulman.edu

Richard E. Stamper
Rose-Hulman Institute of Technology, stamper1@rose-hulman.edu

2018 Rose-Hulman Scholar Institutional Repository
Department of Mechanical Engineering, Rose-Hulman Institute of Technology
April 20, 2018

DISCRETE-POSITION SOLAR TRACKING FOR PHOTOVOLTAIC SYSTEM

Shengnan (Steven) Hong
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
United States
Hongs4@rose-hulman.edu

Zheng Fu
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
United States
fuz@rose-hulman.edu

Richard E. Stamper
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
United States
Stamper1@rose-hulman.edu

Research Program:
Rose Summer Undergraduate Research Program

Abstract

The purpose of this research is to design a new tracking system for solar panels using the idea of discrete-position tracking. Compared with the traditional fixed solar panel, discrete-position trackers have a higher gain of harvesting solar radiation with smaller misalignment angles. Also, since we are trying to design the a passive tracker with solely mechanical structure to do the kinetics, a discrete-position tracker can decrease the cost of the maintenance to a huge extent in contrast to both one-axis and two-axis continuous tracking systems. The majority of the cost of maintaining a continuous tracker is the motor or hydraulic ram. These devices not only are relatively expensive but also do not adapt robust to different weather condition. Therefore, the major part of this summer research is to develop the algorithm of calculating the gain of discrete-position tracker for solar panels comparing with fixed tracker and continuous tracker.

The report will introduce the background information of current solar trackers and explain the initiative of new design. The majority of work done in this summer was developing a mathematical model that quantifies the energy output with several specific inputs. The report uses a lot of space defining the variables of the model, including the input and output of the system. These variables are strictly and scientifically defined so that they meet the standards of corresponding science institutions, such International Astronomical Union. The final model gives the approximate numbers of energy harvested for fixed and discrete-position trackers compared with the two-axis continuous trackers, noted as the optimum trackers in the code.

Because of the time limitation, the team did not start the optimization for the discrete-position angles. The current model with the given position at latitude of 40 degrees north and longitude of 0 degrees can generate approximately 20 percent more electricity than fixed solar panel with the slope of 10 degrees and orientation of 20 degrees west. However, this configuration of discrete-position tracker only makes up 62.89% electricity compared with continuous trackers based on the calculation. The number could be higher after the optimization for the system, and the optimization will be the next direction of the research.

Introduction

People nowadays pay more attention to renewable energy especially solar energy. For most solar farms, people use fixed position panels because they are relatively cheap. But, fixed panels produce 10-40% less energy because they don't track the sun as it moves across the sky [1]. There are two other kinds of tracking system that track the sun as it moves known as single axis and dual axis tracking systems. The installers of photovoltaic systems generally don't add tracking because of the extra costs and maintenance.

We want to explore a method to make tracking relatively cheap, easy to maintain, and have a high working efficiency to provide people a better choice to use the solar energy. For this project, we want to design a solar panel tracking system that we can set two or three discrete positions for a solar panel to move in one day to increase their efficiency.

Tracking systems are well developed and typically fall within one of two types: two-axis and single-axis. A single-axis tracker only has one degree of freedom that acts as an axis of rotation. It normally faces the true north and rotates around the horizontal axis, but it is possible to align it to face a different direction with advanced tracking algorithms. A dual-axis tracker has two degrees of freedom that acts as axes of rotation. Two axis are normal to each other and move independently. It makes the panel always normal to the sunlight to have the maximum working efficiency. Both of those trackers are continuous trackers, and they have complex controllers and drive mechanisms that contribute to the unattractive capital costs and maintenance costs

Objective of the Research

The long-term objective of this research is to identify innovative ways to significantly reduce the cost of solar tracking systems and increase the efficiency of tracking system. In particular, we are exploring the use of two and three discrete position tracking systems in Figure 1 and 2, since they should enable less complex drive mechanisms and controllers.

Two discrete positions tracking system is a system where solar panel will change position once during the daylight time, from one calculated discrete position to another. The system will change the position back to original position during the night. Three discrete positions tracking system has same concept, but it has two positions changing during the day. Three discrete position trackers systems in Figure 2 are supposed to higher efficiency than two discrete position system in Figure 1.

More specifically, the near-term objective of the work is to develop a model of the performance of discrete position solar tracking systems and compare that performance of discrete position solar tracking systems with the hope that discrete tracking can achieve a significant portion of the benefit of traditional tracking in Figure 4 and 5 at a reduced cost.

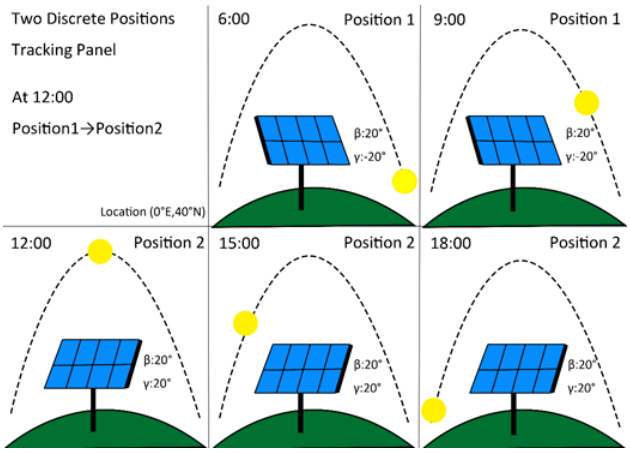

Figure 1. Two Discrete-Positions Tracking Panel Model

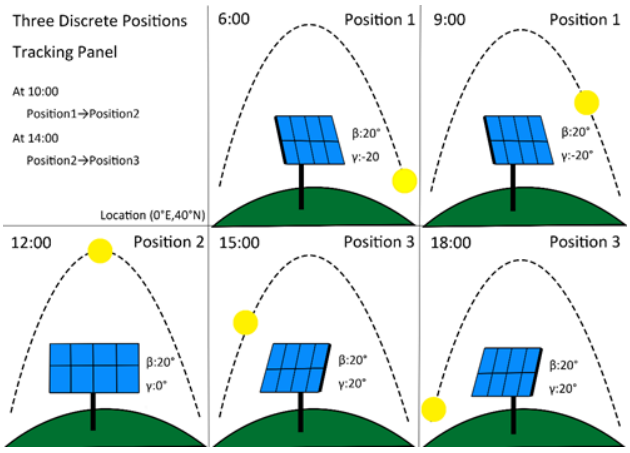

Figure 2. Three Discrete-Positions Tracking Panel Model

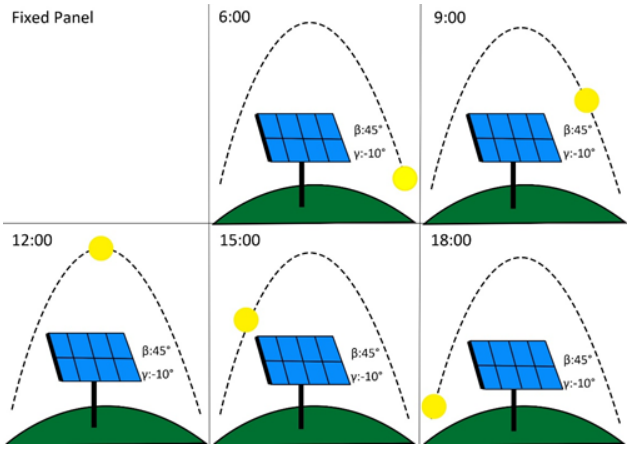

Figure 3. Fixed-Panel Model

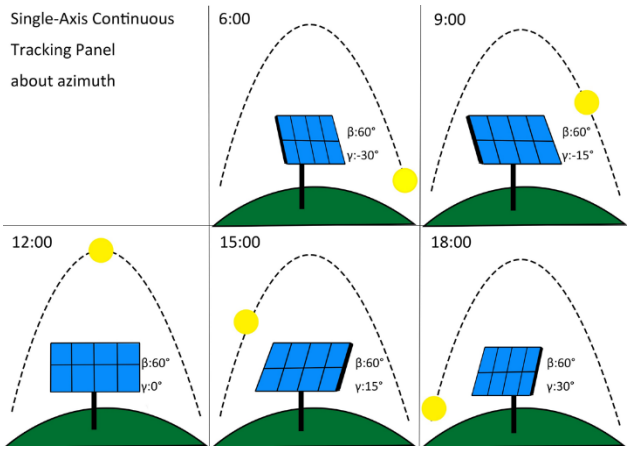

Figure 4. Single-axis Continuous Tracking Panel Model

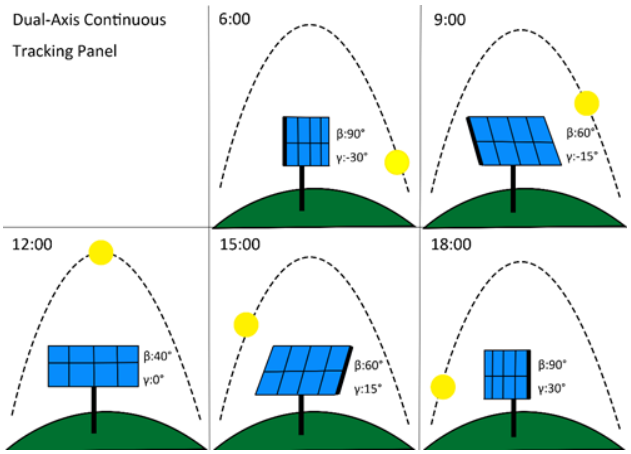

Figure 5. Dual-axis Continuous Tracking Panel Model

Modeling the Performance of Discrete Position Solar Tracking

In order to compare the harvesting efficiency among different types of tracking systems, we built a mathematical model to calculate the amount of the energy produced by solar panels according to the duration of sunshine time. To start with the modeling, we define the variables involving in the process of

calculating the time. We used different Greek letters to represent every variable to make expression simpler.

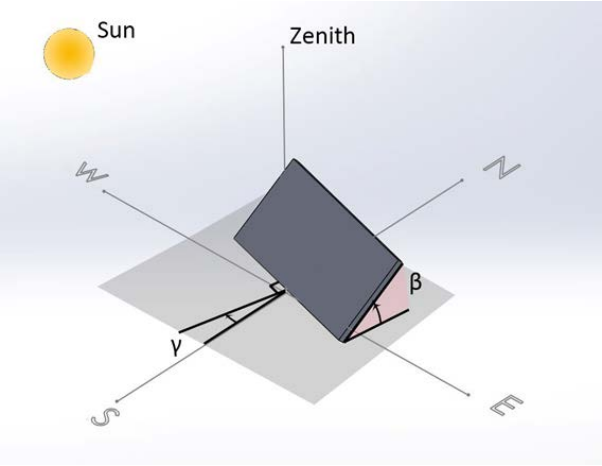

Figure 6. Solar Panel with Horizontal Plane Model

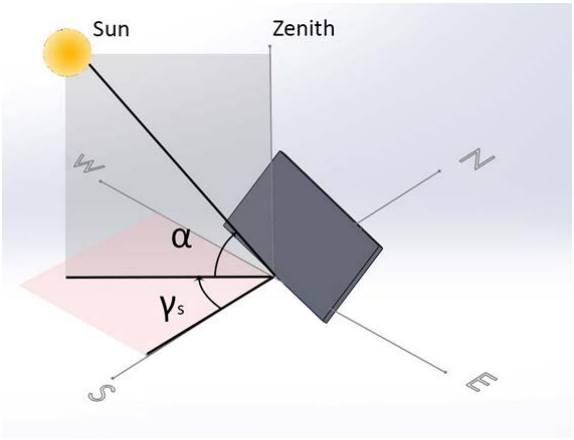

Figure 7. Solar Panel with Sun Position Model

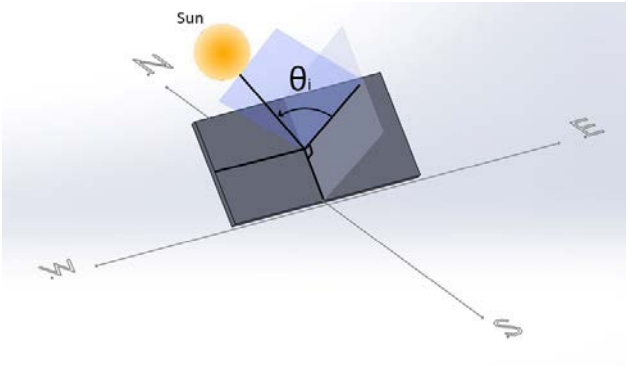

Figure 8. Solar Panel with Sun Position Model

Collector Slope, β

It is the angle between the plane surface, under consideration, and the horizontal. It is taken to be positive for surface sloping towards south and negative for surfaces sloping towards north (Fig. 6) [2].

Surface Azimuth Angle, γ

It is the angle in the horizontal plane, between the line due south and the projection of the normal to the surface (inclined plane) on the horizontal plane (Fig. 6).

Solar Altitude Angle, α

It is the angle between sun ray and its projection in a horizontal plane (Fig. 7).

Solar Azimuth Angle, γ_s

It is the angle in a horizontal plane, between the line due south and the projection of beam radiation on the horizontal plane (Fig. 7)

Angle of Incidence, θ_i

It is the angle between beam radiation on a surface and the normal to that surface (Fig. 8).

Modeling the Motion of the Sun

Declination Angle,

$$\delta = 23.45 \times \sin\left(\frac{360}{365} \times (284 + n)\right) \quad (1)$$

Equation of Time,

$$B = (n-1) \times \frac{360}{365} \quad (2)$$

$$E = 229.2 \times (0.000075 + 0.001868 \times \cos(B) - 0.032077 \times \sin(B) - 0.014615 \times \cos(2B) - 0.04089 \times \sin(2B)) \quad (3)$$

Solar Time

$$ST = t + \frac{E}{60} \quad (4)$$

Hour Angle,

$$\omega = (ST - 12) \times 15° \quad (5)$$

Solar Azimuth Angle,

$$\gamma_s = \sin^{-1}(\cos(\delta) \times \cos(\varphi) \times \cos(\omega) + \sin(\delta) \times \sin(\varphi)) \quad (6)$$

Solar Altitude Angle,

$$\alpha = \sin^{-1}(\cos(\delta) \times \cos(\varphi) \times \cos(\omega) + \sin(\delta) \times \sin(\varphi)) \quad (7)$$

Modeling the Performance of a Discrete Tracking System

Hour Angle on Horizontal Surface

$$\omega_s = \cos^{-1}(-\tan(\varphi) \times \tan(\delta)) \quad (8)$$

Hour Angle on Tilted Surface

$$\omega_{st} = \cos^{-1}(-\tan(\varphi - \beta) \times \tan(\delta)) \ [North] \quad (9)$$

$$\omega_{st} = \cos^{-1}(-\tan(\varphi + \beta) \times \tan(\delta)) \ [South] \quad (10)$$

Angle of Incidence on Horizontal Surface

$$\theta_z = \cos^{-1}(\cos(\varphi) \times \cos(\delta) \times \cos(\omega) + \sin(\delta) \times \sin(\varphi)) \quad (11)$$

Angle of Incidence on Inclined Surface

$$\begin{aligned}\theta_i = (&\cos(\varphi) \times \cos(\beta) + \sin(\varphi) \times \sin(\beta) \\ &\times \cos(\gamma)) \times \cos(\delta) \times \cos(\omega) \\ &+ \cos(\delta) \times \sin(\omega) \times \sin(\beta) \\ &\times \sin(\gamma) + \sin(\delta) \\ &\times (\sin(\varphi) \times \cos(\beta) - \cos(\varphi) \\ &\times \sin(\beta) \times \cos(\gamma))\end{aligned} \quad (12)$$

Equations cited from *Solar Energy, Fundamentals, Design, Modeling, and Applications* [2].

Comparing the Performance of Discrete Tracking to Other Systems

Based on the MATLAB codes, we were able to generate the average percentage of output compared with perfect trackers. By taking different inputs, we received different output results. Then, we used the idea of gradient to fix other variables, and debugged one variable to find the optimized output. According to our simulations, we found that if a solar panel tracker is positioned at a longitude of 0 degrees, and latitude of 40 degrees north, discrete-position trackers can harvest up to 90.67% solar radiation compared with perfect trackers. The results are shown in the tables below. Each table was tabulated with one different variable. The tracker was turned into next position at 10:00 am and 2:00 pm. The tabulated results can be found in the Appendix I for the reference. The MATLAB codes can be found in the Appendix II to IV.

Conclusion

We produced a model of the performance of discrete tracking systems. Compared to a fixed solar panel, discrete-position trackers have a higher harvesting efficiency than fixed solar panels. At a latitude of 40 degrees north and longitude of 0 degrees, the discrete tracker with two given positions can generate approximately 20 percent more electricity than a fixed solar panel with a slope of 10 degrees and orientation of 20 degrees west. However, it only makes up 62.89% electricity compared to continuous trackers based on the calculation. The number could be higher after the optimization for the system. This will be the next direction of the research.

Future Work

For future work, we will start the optimization process for the orientation angle and slope angle of the discrete. This will allow us to determine whether we are going to use two or three-discrete position tracker. Then, the optimization process will mainly use the Lagrange Multiplier with the help of Matlab and Simulink.

Before the process, we need to make several assumptions. Primarily, we assume there is no energy loss during the propagation process. Secondly, we assume the location of solar panel is 40 degrees north latitude and 0 degrees east longitude in order to avoid the impact of time zone during the calculation. The sources we are using for input takes the time zone into account. By setting the longitude as 0 degrees, we can avoid the increasing differences of hour angles. Apart from that, one problem we are facing is that there is also an hour offset due to Daylight Saving Time.

After finishing the optimization process for the previous mentioned location, we can start another location that is close to Terre Haute, allowing us to compare our theoretical numbers with experimental data. With the calculated angles, we can build a simple prototype on campus to test the gain. Finally, we can start to design mechanisms for discrete-position tracker, including actuation and controlling system for the tracker.

References

[1] Landau, Charles. “Optimum Tilt of Solar Panels.” *Optimum Tilt of Solar Panels.* www.solarpaneltilt.com/

[2] G. N. Tiwari. “Chapter 1”. *Solar Energy: Fundamentals, Design, Modeling and Applications.* Alpha Science International, 2013, pp. 16–23.

Appendix I – Calculated Performance for Discrete Tracking System at Different Angles

Table 1. Optimized Performance of Discrete Tracking System for $Gamma_1$ and $Gamma_3$

Gamma	-30	-40	-50	-60	-70	-73	-74	-75	-80
Percentage	65.25	67.21	68.70	69.68	70.12	70.15	70.15	70.14	70.01

Note: $beta_1 = -20$, $gamma_2 = 0$, $beta_2 = 20$, $beta_3 = 20$

Table 2. Optimized Performance of Discrete Tracking System for $Beta_2$

Beta_2	10	20	30	35	37	40	50	60	70
Percentage	68.23	70.14	71.18	71.35	71.35	71.30	70.44	68.61	65.81

Note: $gamma_1 = -75$, $beta_1 = 20$, $gamma_2 = 0$, $gamma_3 = 75$, $beta_3 = 20$

Table 3. Optimized Performance of Discrete Tracking System for $Beta_1$ and $Beta_3$

Beta	20	30	40	50	60	63	65	70	80
Percentage	71.35	77.79	82.79	86.20	87.91	88.09	88.12	87.88	86.11

Note: $gamma_1 = -75$, $gamma_2 = 0$, $beta_2 = 35$, $gamma_3 = 75$

Table 4. Optimized Performance of Discrete Tracking System for $Gamma_2$

Gamma_2	0	1	2	3	5	10	15	20	25
Percentage	88.12	90.67	90.66	90.66	90.63	90.50	90.29	89.99	89.62

Note: $gamma_1 = -75$, $beta_1 = 65$, $beta_2 = 35$, $gamma_3 = 75$, $beta_3 = 65$

Appendix II – MATLAB Code on Performance Calculation

```
%*************************************************************************
%  Discrete_Tracking.m
%
%     PROGRAM DESCRIPTION
%     This program compares percentage between three discrete position tracker
%     and perfect tracker.
%
%   Input  : All input is in the Data section [beta, phi, gamma]
%   Output : Prints results to graphs
%
%                                     Written by Steven Hong and Zheng Fu
%                                                 08/23/2017
%*************************************************************************

clc; clear variables; close all;

%% Input

phi   = 40;       % [degree] latitude
longitude = 0;    % [degree] longitude
increment = 0.01; % time step to calculate hour angle

change_time_1 = 10; % [hour] change time
change_time_2 = 14; % [hour] change time

% position #1
gamma_1 = -30;   % [degree south] orientation angle (due south) / west is positive
direction
                % gamma has to be bigger than 0, the sunshine time is assuming towards
west
beta_1  = 30;   % [degree] Slope angle

% position #2
gamma_2 = 10;    % [degree south] orientation angle (due south) / west is positive
direction
beta_2  = 10;   % [degree] Slope aengle

% position #3
gamma_3 = 30;    % [degree south]
beta_3  = 30;   % [degree]

%% Calculation of Energy Output

[day,sunrise,sunset,start_1,final_1,delta_1] =
Calculation_of_Sunshine_fcn(phi,beta_1,gamma_1);
[~,~,~,start_2,final_2,delta_2] = Calculation_of_Sunshine_fcn(phi,beta_2,gamma_2);
[~,~,~,start_3,final_3,delta_3] = Calculation_of_Sunshine_fcn(phi,beta_3,gamma_3);

% Calculate the sun's equation of time(E)
B = (day-1)*360/365;    % [Equation 1.8]
E = 229.2*(0.000075+0.001868*cosd(B)-0.032077*sind(B)-0.014615*cosd(2*B)-
0.04089*sind(2*B));    % [minute] equation of time

% preallocate space for vector
day_output_1 = zeros(1,365);
day_output_2 = zeros(1,365);
day_output_3 = zeros(1,365);
day_output_perfect = zeros(1,365);
```

```
for k=1:length(day)
    sum_output = 0;             % Initialize the loop index

    time1 = sunrise(k):increment:sunset(k);  % Create time vector to calculate different
hour angle
    time2_1 = start_1(k):increment:change_time_1;     % time step is 0.03 hour
    time2_2 = change_time_1:increment:change_time_2;
    time2_3 = change_time_2:increment:final_2(k);

    [st_1, output_1]  = method_of_equation_fcn(phi,beta_1,gamma_1,delta_1,time2_1,k,E);
    [st_2, output_2]  = method_of_equation_fcn(phi,beta_2,gamma_2,delta_2,time2_2,k,E);
    [st_3, output_3]  = method_of_equation_fcn(phi,beta_3,gamma_3,delta_3,time2_3,k,E);
    day_output_1(k) = sum(output_1);% [unit energy] write daily energy output to a vector
    day_output_2(k) = sum(output_2);% [unit energy] write daily energy output to a vector
    day_output_3(k) = sum(output_3);% [unit energy] write daily energy output to a vector

    st = [st_1 st_2 st_3];
    day_output = day_output_1 + day_output_2 + day_output_3;
    day_output_perfect(k) = length(time1);  % [unit energy] add up energy for every hour
end

% compare the percentage of optimum for every day
compare_optimum = day_output ./ day_output_perfect;

% calculate average percentage of optimum for the whole year
sum_year = sum(day_output);      % [unit energy] add up total amount of energy in one day
sum_year_perfect = sum(day_output_perfect);   % [unit energy] add up total amount of
energy generated by perfect tracking
average_percentage = sum_year / sum_year_perfect;

fprintf('The average percentage is %6.4f\n', average_percentage);

%% Debugging on Max & Min Day
% find the date for maximum and minimum average percentage
[max_percentage, max_date] = max(compare_optimum);
[min_percentage, min_date] = min(compare_optimum);

% find the percentage of optimum of different moment on maximum average percentage day
time2_max_1 = start_1(max_date):increment:change_time_1;
time2_max_2 = change_time_1:increment:change_time_2;
time2_max_3 = change_time_2:increment:final_2(max_date);
[st_max_1, output_max_1] = method_of_equation_fcn(phi,beta_1,gamma_1,delta_1,time2_max_1,
max_date,E);
[st_max_2, output_max_2] = method_of_equation_fcn(phi,beta_2,gamma_2,delta_2,time2_max_2,
max_date,E);
[st_max_3, output_max_3] = method_of_equation_fcn(phi,beta_3,gamma_3,delta_3,time2_max_3,
max_date,E);
st_max = [st_max_1 st_max_2 st_max_3];
output_max = [output_max_1 output_max_2 output_max_3];
misaligned_angle_max = acosd(output_max);

% find the percentage of optimum of different moment on minimum average percentage day
time2_min_1 = start_1(min_date):increment:change_time_1;
time2_min_2 = change_time_1:increment:change_time_2;
time2_min_3 = change_time_2:increment:final_2(min_date);
[st_min_1, output_min_1] = method_of_equation_fcn(phi,beta_1,gamma_1,delta_1,time2_min_1,
min_date,E);
[st_min_2, output_min_2] = method_of_equation_fcn(phi,beta_2,gamma_2,delta_2,time2_min_2,
min_date,E);
[st_min_3, output_min_3] = method_of_equation_fcn(phi,beta_3,gamma_3,delta_3,time2_min_3,
min_date,E);
```

```
st_min = [st_min_1 st_min_2 st_min_3];
output_min = [output_min_1 output_min_2 output_min_3];
misaligned_angle_min = acosd(output_min);   % [degree] calculate the misaligned angle for
every time increment

%% Graph

% Create a graph to display the unit energy generated by fixed panel and perfect tracker
figure;
plot(day,day_output,'-o',day,day_output_perfect,'-');
xlabel('nth day of the year'); ylabel('Unit Energy Generated');
title(['Generated Unit Energy Generated at ',num2str(phi),' degree North,
',num2str(beta_1),' degree slope, ', num2str(gamma_1),' degree orientation angle']);
legend('fixed panel', 'perfect tracker');

% Create a graph to display the percentage of fixed panel comparing with perfect tracking
figure;
plot(day,compare_optimum);
xlabel('nth day of the year'); ylabel('The Percentage of Fixed Panel');
title(['The Percentage at ',num2str(phi),' degree North, ',num2str(longitude),' degree
East']);

% Create a graph to display the misaligned angle for every moment on different days
figure;
plot(st_max, misaligned_angle_max,'r',st_min,misaligned_angle_min,'b');
xlabel('Solar Time (hour)'); ylabel('Misaligned Angle (degree)');
ylim([0,90]);
title([ 'Misaligned Angle During a Day at ',num2str(phi),' degree North,
',num2str(longitude),' degree East']);
legend([num2str(max_date),'th day'],[num2str(min_date),'th day']);

% Create a graph to display the percentage of optimum for every moment on different days
figure;
plot(st_max, output_max,'r',st_min, output_min,'b');
xlabel('Solar Time (hour)'); ylabel('Percentage of Optimum');
ylim([0,1]);
title(['Percentage of Optimum During a Day at ',num2str(phi),' degree
north,',num2str(longitude),' degree East']);
legend([num2str(max_date),'th day'],[num2str(min_date),'th day']);
text(15,0.2,['Position Cofiguation:/n Position #1:',num2str(gamma_1),'degree south',
num2str(beta_1),'degree slope']);
```

Appendix III – MATLAB Function Code on Sunshine Time

```
%*************************************************************************
% Calculation_of_Sunshine_fcn.m
%
%       FUNCTION DESCRIPTION
%       This function calculates the sunshine time for inclined solar
%       panel, and the start and end time for sunshine.
%
%   What comes in : latitude angle(phi),slope angle(beta),orientation
%       angle(gamma)
%   What goes out : a vector containing the day of the year, a vector
%       containing the sunshine time for inclined solar panel, the sunrise and
%       sunset time, start and end time of sunshine, a vector containing the
%       delta angle.
%   Side effects  : Plot debugging graph.
%                                                 Written by Steven Hong
%                                                           08/23/2017
%*************************************************************************

function [day,sunrise,sunset,start,final,delta] =
Calculation_of_Sunshine_fcn(phi,beta,gamma)

day = 1:365; % Generate a vector for a total amount of day in one year
delta = 23.45*sind(360*(284+day)/365); % [degree] Calculate declination angle [Equation
1.9]
duration = 2*acosd(-tand(phi)*tand(delta))/15;  % [hour] Calculate the number of daylight
hours for tracking system [Equation 1.16]

% Calculate the number of sunshine hours for fixed panel in hours
eq_a = cosd(delta)*sind(beta)*sind(gamma);
% calculate the coefficient for trig identity equation
eq_b = (cosd(phi)*cosd(beta)+sind(phi)*sind(beta)*cosd(gamma))*cosd(delta);
% calculate the coefficient fof trig identity equation
eq_c = (sind(phi)*cosd(beta)-cosd(phi)*sind(beta)*cosd(gamma))*sind(delta);
% calculate the coefficient for trig identity equation

% eq_2 is for debugging purpose
% eq_2 is trying to find the sunshine time when gamma=0, compare the new sunshine time
with old equation
eq_a_2 = 0;                         % calculate the coefficient for trig identity equation
eq_b_2 = (cosd(phi)*cosd(beta)+sind(phi)*sind(beta))*cosd(delta); % calculate the
coefficient fof trig identity equation
eq_c_2 = (sind(phi)*cosd(beta)-cosd(phi)*sind(beta))*sind(delta); % calculate the
coefficient for trig identity equation

support_angle_2 = zeros(1,365); % preallocate space for vector
sunshine_2 = 2*(asind(-eq_c_2./eq_b_2) - support_angle_2)/15;   % [hour] Calculate the
number of sunshine hours for fixed panel
sunshine_half_2 = 4*atand((eq_a_2-sqrt(eq_a_2.^2+eq_b_2.^2-eq_c_2.^2))./(eq_c_2-
eq_b_2))/15;
sunshine_0 = 2*acosd(-tand(phi-beta)*tand(delta))/15;  % [hour] Calculate the number of
sunshine hours for fixed panel
% sunshine_0 is for debugging purpose when gamma=0

% Filter irration numbers for tangent function
if eq_a(1) == 0
    support_angle = zeros(1,365) - 90;
else
    support_angle = atand(eq_b./eq_a);
end
```

```
% calculate the number of sunshine hours for fixed panel in hours
sunshine = 2*(asind(-eq_c./(sqrt(eq_a.^2+eq_b.^2))) - support_angle)/15;
sunshine_half_evening = 4*atand((-eq_a-sqrt(eq_a.^2+eq_b.^2-eq_c.^2))./(eq_c-eq_b))/15;
% positive value for evening
sunshine_half_morning = 4*atand((-eq_a+sqrt(eq_a.^2+eq_b.^2-eq_c.^2))./(eq_c-eq_b))/15;
% negative value for morning

% Calculate the start and end moment for fixed panel and perfect tracker
sunrise = 12 - duration/2;
sunset  = 12 + duration/2;

if gamma>0
    sunshine_half = abs(sunshine_half_morning);  % If gamma is positive, the orientation
is devating towards west, which leads to less sunshine in morning
    start   = 12 - sunshine_half/2;              % If gamma is positive, the orientation
is devating towards west, which means it has shade in the morning
    final   = 12 + duration/2;
else
    sunshine_half = abs(sunshine_half_evening);  % If gamma is negative, the orientation
is devating towards east, which leads to less sunshine in afternoon
    start   = 12 - duration/2;
    final   = 12 + sunshine_half/2;         % If gamma is negative, the orientation is
devating towards east, which means it has shade in the afternoon
end

for m=1:365
    if start(m) < sunrise(m)
        start(m) = sunrise(m);
    end
    if final(m) > sunset(m)
        final(m) = sunset(m);
    end
end

% [hour] calculate the actual sunshine time for fixed panel due to orientation angle in
hours
sunshine_complete = (sunshine_half + duration)/2;

% Calibrate actual sunshine time for fixed panel
for m = 1:365
    if sunshine_complete(m) > duration(m)
       sunshine_complete(m) = duration(m);
    end
end

%% Graph for Duration Calculation

% Debugging plot to check the sunshine algorithm
figure;
plot(day,sunshine_0,'-o',day,sunshine_2,'-d',day,sunshine_half_2,'-x');
% Check whether all the lines coincide, which means get the same result
xlabel('nth day in a year');
ylabel('The Duration of the Sunshine (hour)');
title('[Method 1] Debugging Graph - The Duration of Sunshine when gamma=0');
legend('Equation','Trig Method','Half Angle');
text(10,10.7,'Check whether all the lines coincide, which means get the same result');

figure;
plot(day,sunshine,'-o',day,sunshine_half,'-x');
% Check whether two lines coincide, which means get the same result
```

```
xlabel('nth day in a year');
ylabel('The Duration of the Sunshine (hour)');
title('[Method 1]Inspect Graph - Tangent Half-Angle Substitution');
legend('Trig Method','Half-angle');
text(10,14,'Check whether two lines coincide, which means get the same result');

figure;
plot(day,sunshine,'-o',day,duration,'-d',day,sunshine_complete,'-x');
xlabel('nth day in a year');
ylabel('Duration of the Sunshine');
title('[Method 1] Inspect Graph - The Duration of Sunshine for Fixed Panel and Perfect
Tracker');
legend('sunshine', 'duration','actual sunshine');

figure;
plot(day, sunrise,'-o', day, start, '-x');
xlabel('nth day in a year');
ylabel('Time Moment');
title('[Method 1] Visualization Graph - Sunrise and Start of Sunshine Time');
legend('Sunrise','Start of Sunshine');

figure;
plot(day, sunset,'.r', day, final,'.b');
xlabel('nth day in a year');
ylabel('Time Moment');
title('[Method 1] Visualization Graph - Sunset and End of Sunshine Time');
legend('Sunset', 'End of Sunshine');
end
```

Appendix IV – MATLAB Function Code on Optimum Performance

```
%*************************************************************************
% method_of_equation_fcn.m
%
%       FUNCTION DESCRIPTION
%       This function calculates percentage of the optimum for each hour.
%
%   What comes in : a matrix containing student's ID number, 9 homework
%                   scores in percentage, 4 exam scores in percentage.
%   What goes out : a single vector containing the total course grade for
%                   each student.
%   Side effects  : NONE
%                                                 Written by Steven Hong
%                                                            08/21/2017
%*************************************************************************

function [st, output] = method_of_equation_fcn(phi,beta,gamma,delta,time2,date,E)

% preallocate space for vector
st     = zeros(1, length(time2));
output = zeros(1, length(time2));

% calculate the percentage of optimum for every time increment
for p=1:length(time2)
    st(p) = time2(p) + E(date)/60;    % [hour] Solar time [Equation 1.7]
    omega = (st(p) - 12) * 15;        % [degree] Hour angle [Equation 1.10]
    output(p) = (cosd(phi)*
cosd(beta)+sind(phi)*sind(beta)*cosd(gamma))*cosd(delta(date))*cosd(omega)+cosd(delta(dat
e))*sind(omega)*sind(beta)*sind(gamma)+sind(delta(date))*(sind(phi)*cosd(beta)-
cosd(phi)*sind(beta)*cosd(gamma));
% calculate the percentage of optimum for each hour [Equation 1.11]
end
end
```