RMIT University

School of Engineering

EEET2248 – Electrical Engineering Analysis

Lab Experiment #3

Solar Panel Sizing

Lecturer: Dr. Katrina Neville

Student Name: Nathan Paul Williams

Student Number: s3707244

Group: Tuesday 10:30-12:30

Submission Due Date: 11th May 2018

Problem Statement

Task 1: Plot solar exposure data provided by Bureau of Meteorology (BoM) against time

Task 2: Produce a yearly solar exposure model for an average year

Task 3: Find a suitable solar array size for a house requiring 6kWh of power in a specified location

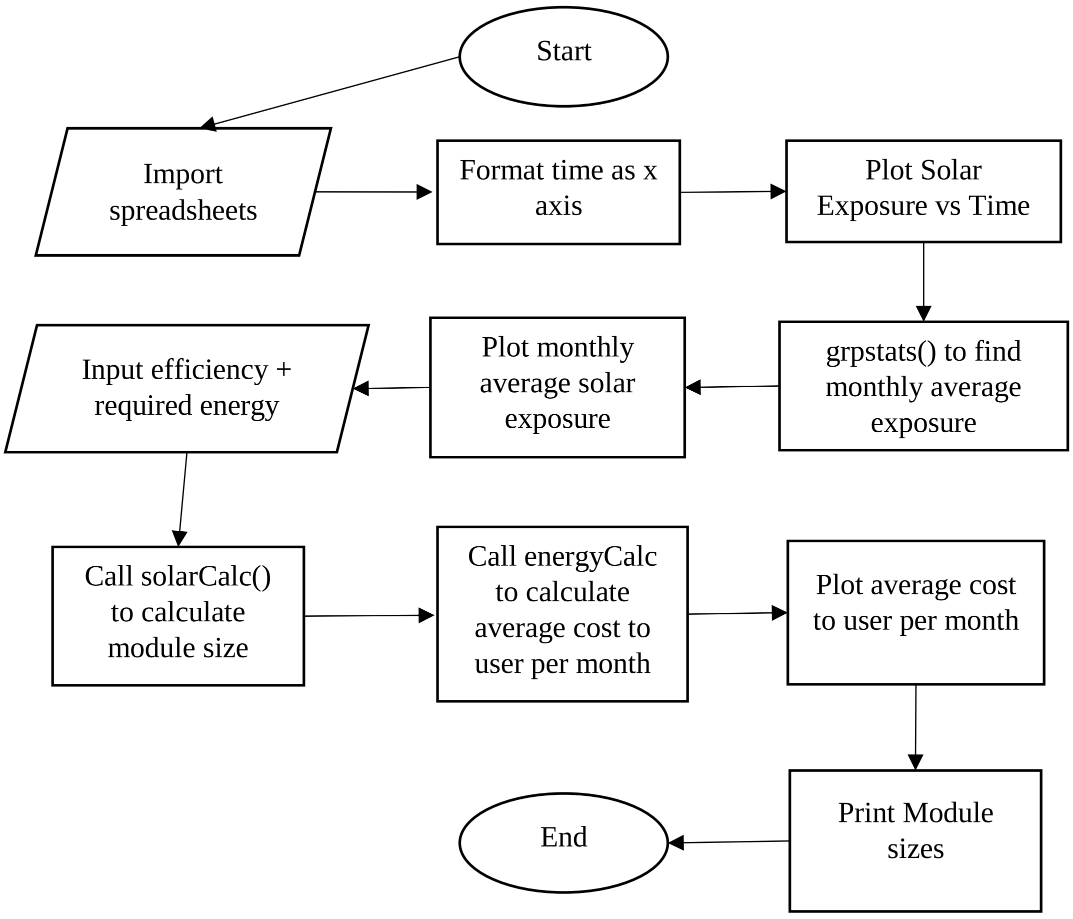
Input

- Solar exposure spreadsheets
- Energy required for example house
- Minimum solar panel module size
- Efficiency of panels
- Energy costs
- Feed-in tariffs

Output

- Solar exposure vs time plot(s)
- Average monthly solar exposure plot(s)
- Solar panel module size(s)
- Average User Cost vs Month plot(s)

Design



Start
Import spreadsheets
Format time as x axis
Plot Solar Exposure vs Time
grpstats() to find monthly average exposure
Plot monthly average solar exposure
Input efficiency + required energy
Call solarCalc() to calculate module size
Call energyCalc to calculate average cost to user per month
Plot average cost to user per month
Print Module sizes
End

Figure 1: Program Structure Flowchart

My project design revolved around ensuring the user had to interpret the output as little as possible to understand the results as well as providing accurate and useful data output. For task 1 I mapped each solar exposure value to a specific date using datenum(). Then plotted date vs exposure as a scatter plot using datetick(). When plotted, I noticed that that the pattern was similar to a sin/cos graph (reason explained in discussion) so I overlayed a cos graph as a rough line of best fit to demonstrate this to the user.

For task 2 I used grpstats() to find an average solar exposure for each month based on the data. I then plotted this as a bar graph for each city. For task 3 I first set values for the efficiency factor and the mean solar exposure, the values used are discussed more below. Then to calculate the required module size I had a user-defined function called solarCalc(). SolarCalc() takes in an exposure, energy requirement and an efficiency factor and returns the required module size to the nearest 0.25kW.

There is also another function called energyCalc(). EnergyCalc() takes in a module size, efficiency factor, solar exposure and a required energy generation value. I pass in the average solar exposure per month and it then returns the average cost per month based on feed-in tariffs and energy costs. For each city the average cost for each month is plotted as a bar graph with negative costs indicating overall money being gained due to feed-in tariffs and positive costs representing cost of drawing from grid. The module sizes returned by solarCalc() are then printed to the CLI in a neat string using fprintf() before the program terminates.

<u>Output</u>

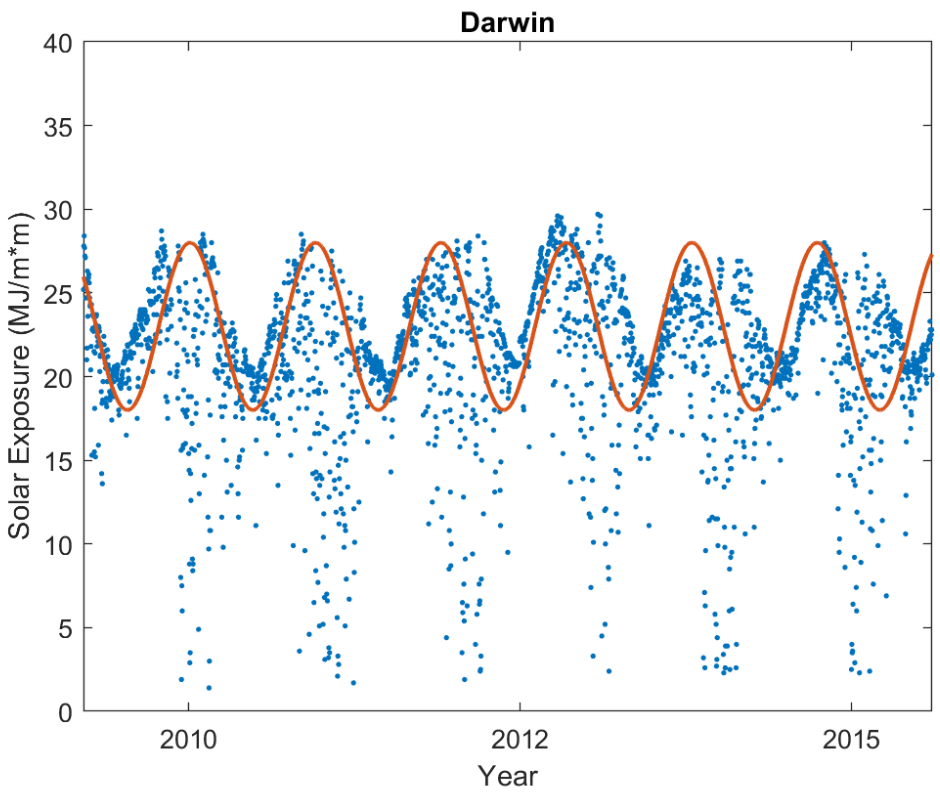


Figure 2: Darwin Solar Exposure

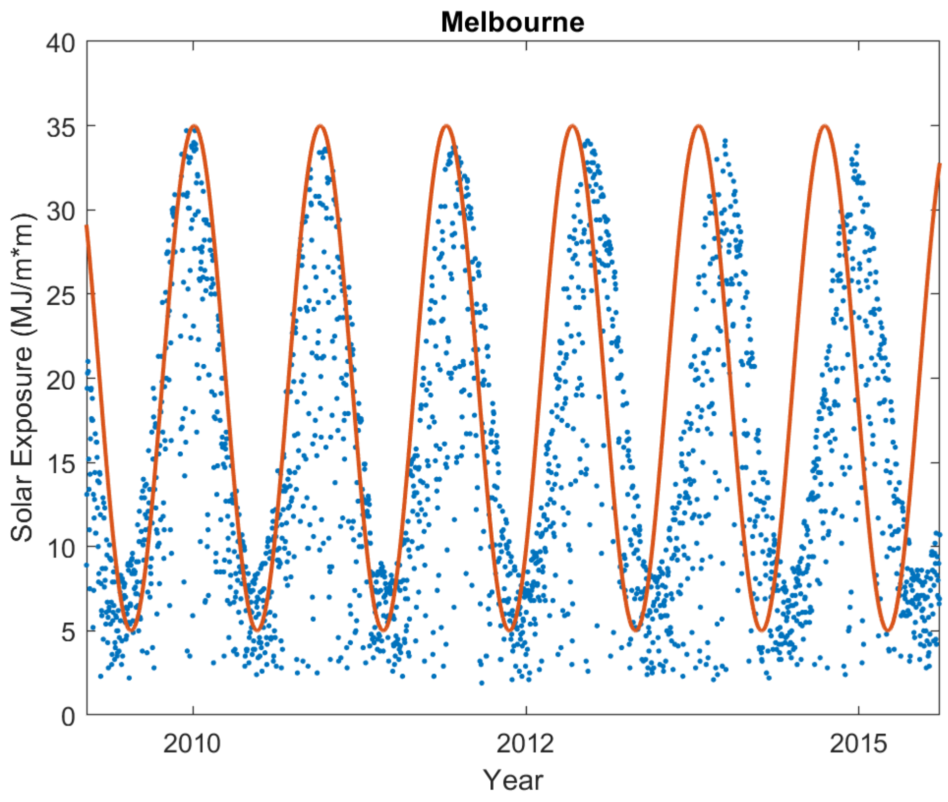


Figure 3: Melbourne Solar Exposure

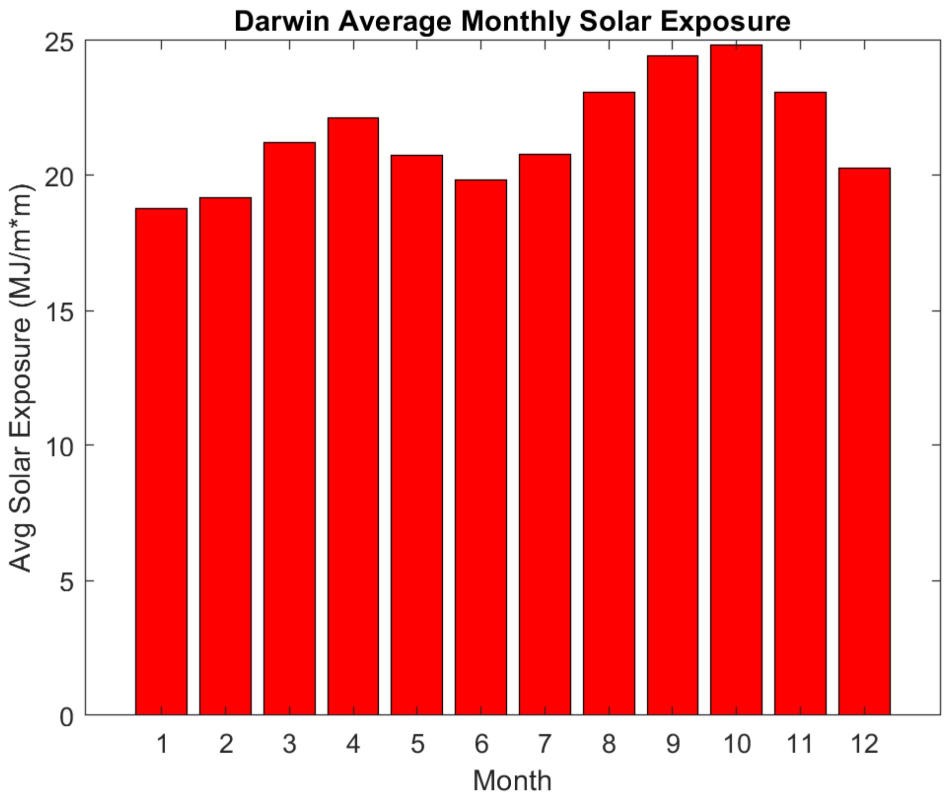


Figure 4: Darwin Average Monthly Solar Exposure

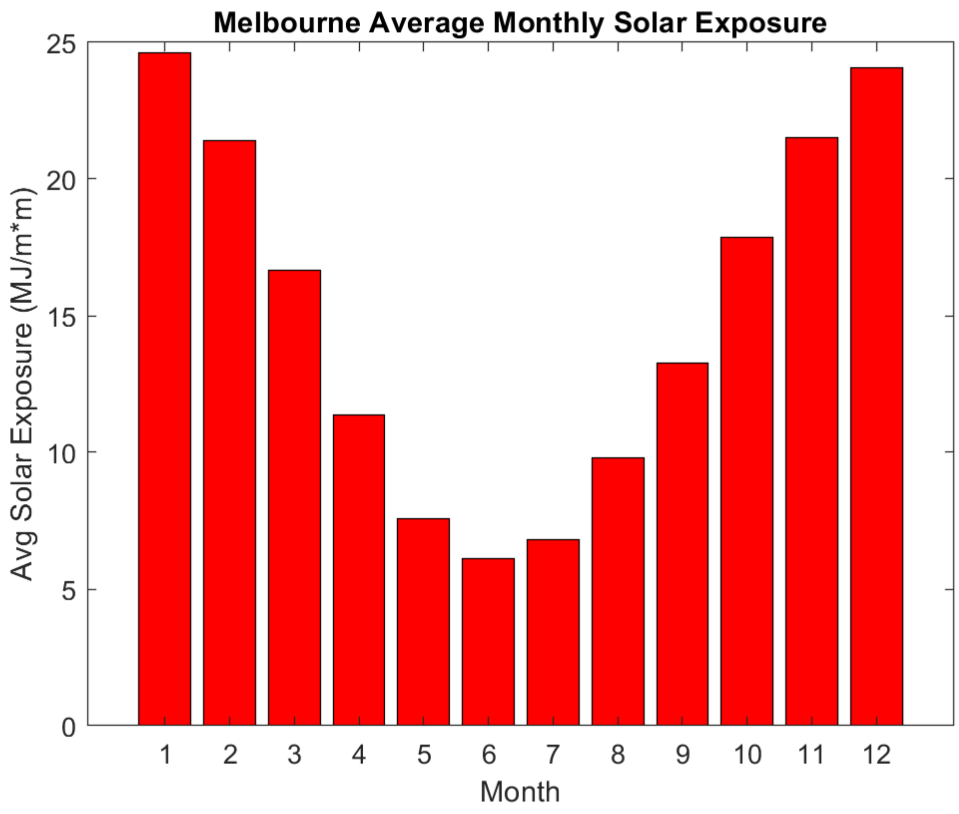


Figure 5: Melbourne Average Monthly Solar Exposure

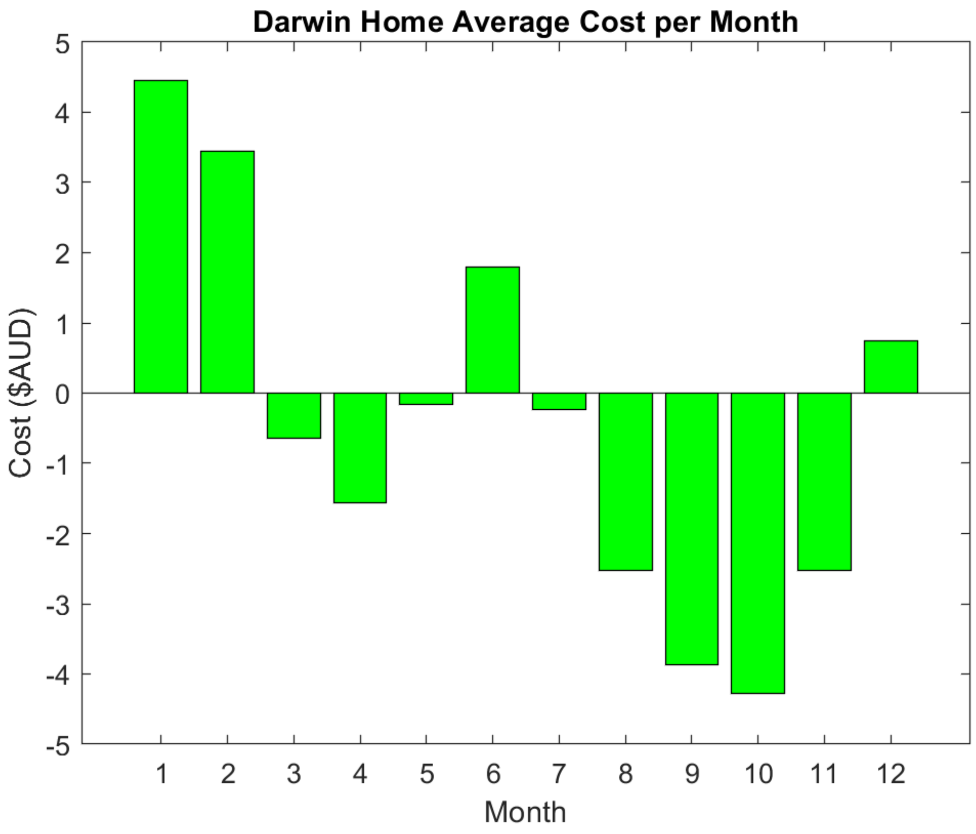


Figure 6: Darwin Average Cost to User per day each Month

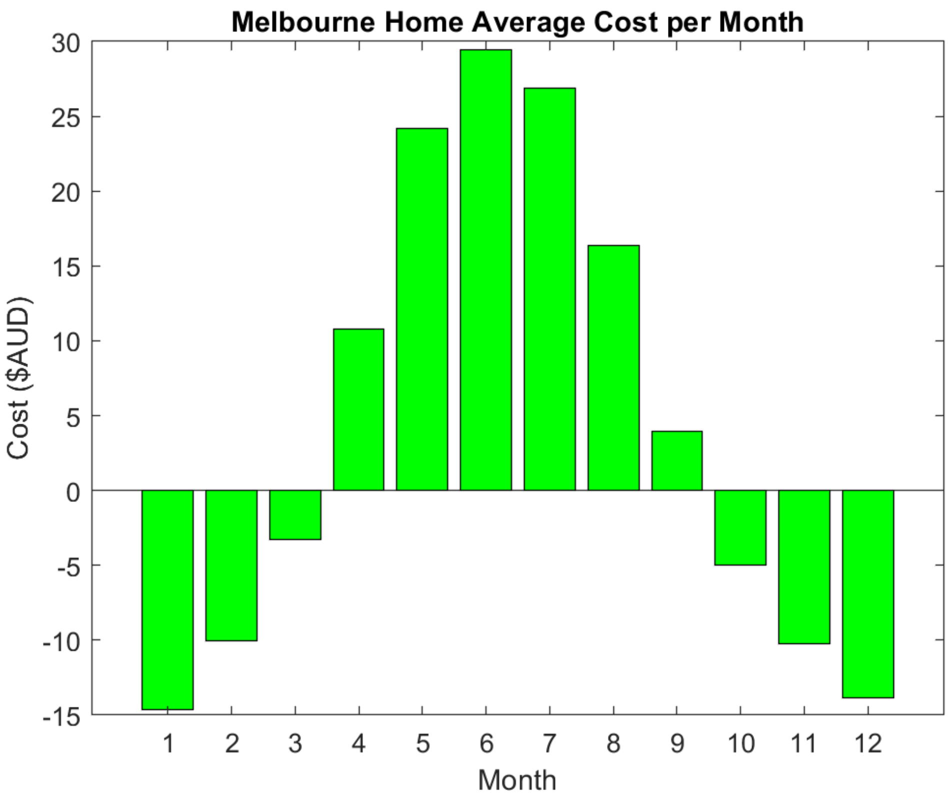


Figure 7: Melbourne Average Cost to User per day each Month

```
The Darwin home would need a module size of 1.75 kW

The Melbourne home would need a module size of 2.5 kW
```

Figure 8: CLI Output

Testing

```
 Dar Module size = 6kWh / (0.6[efficiency] * (21.5343[Darwin average
exposure]/3.6))

= 1.67175 kW

= 1.75 kW [round up to nearest 0.25 kW]

Mel Module size = 6kWh / (0.6[efficiency] * (15.0954[Melbourne average
exposure]/3.6))

= 2.38483 kW

= 2.5 kW [round up to nearest 0.25 kW]
```

Figure 9: Module Size Hand working

Hand working for the module size calculation is shown above in fig 9. The hand working agreed exactly with the CLI output seen in figure 8 however to verify accuracy of the program this should be tested with much more input data. However; for the task it appears the results are correct.

```
 Darwin Energy Produced = 1.75 [modsize] * (21.2132 [march exposure] /3.6) *
0.6 [efficiency] =  6.1871

Energy Net = 6.1871 – 6 [energy used] = 0.1871

Cost = 0.1871 [energy net] * 30.42 [avg days in a month] * -0.113 [feed-in
tariff]

Cost = -0.643 AUD

---------------------------------------------------------------------------

Melbourne Energy Produced = 2.5 [modsize] * (7.5892 [may exposure] /3.6) *
0.6 [efficiency] =  3.1621

Energy Net = 3.1621 – 6 [energy used] = -2.8378

Cost = -2.8378 [energy net] * 30.42 [avg days in a month] * -0.28 [energy
cost]

Cost = 24.17 AUD
```

Figure 10: Cost Calculation Hand Working

The hand working for the cost calculation is shown in fig 10. This is to verify the accuracy of our energyCalc() function as well as the plots shown in fig 6 and fig 7. One cost value for a random month in each city was calculated to check accuracy. For the Darwin example the output cost for March was -0.6434 $AUD. This extra significant figure is most likely due to the rounding during hand calculation. For the Melbourne example the output cost for May was 24.1717 $AUD. Again, the extra significant figures are very likely due to rounding during hand calculations. These results provide evidence our cost calculation is accurate but extra testing would be needed to confirm.

<u>Discussion/Reflections</u>

For task 1 the plots used date as the x axis to assign a day to each data point. One problem with this is that in the spreadsheets there were some NaN values for days where data was missing so these points did not show up on the plots. However; there were relatively very few missing data points so it is unlikely this would effect the shape of the graph too much and the plots still serve the purpose of demonstrating an apparent pattern. The pattern shown is that of a cos/sin graph. It is suspected this is due to the rotation of the earth around the sun affecting the solar exposure throughout the year.

For task 2 plotting the average solar exposure per month it is suspected that the summer months would have higher levels of solar exposure than winter months. This is clearly evident in the Melbourne bar graph with solar exposure in December over 4x greater than in June. However, this pattern is much less evident in the Darwin plot with the exposure appearing to deviate much less from the mean and not in an obvious seasonal cycle. It is likely this is due to Darwin's close proximity to the equator and hence less change in orientation to the sun throughout the year.

For task 3 there are several assumptions made that can affect the results. The first is the efficiency factor. The lab files suggested that efficiency can vary from 50%-70% so for this task we used the median value of 60%. However; the current record for solar cell efficiency in cells that are "feasible for consumer panels" is 26.6% [1]. Therefore, this efficiency value may need to be adjusted in future if the program is to be used for real world applications. Another possible issue is that the solar exposure that is passed to solarCalc() is simply a mean value of all solar exposure which may not be the most suitable value for calculating mod size due to greater standard deviation in locations like Melbourne. In energyCalc() the values for feed-in tariffs [2] and energy costs [3] are based on data in Victoria and while the values are similar nationwide they are not identical and could cause some inaccuracy for the Darwin calculation in particular. Also in energyCalc() the cost per month is calculated by multiply by the average number of days in a month (30.42) and not the number of days in that month in particular. This could cause some slight inaccuracy.

Conclusion

A solution has been implemented that allows the user to easily interpret solar exposure data for Darwin and Melbourne. It also allows the user to calculate required module size of a solar panel array in a specific location, provided there is valid solar exposure data for the location, and provides a brief cost analysis based on yearly solar exposure cycles and energy rates. The program's output agrees with the hand worked testing suggesting an appropriate level of accuracy has been achieved. Due to the lack of user input (excluding the solar exposure data) the program does not need any extra measures implemented to handle error cases at this stage. In conclusion, the program meets the demands of the problem statement(s) and provides a solution with some extra features implemented for easier data interpretation however; the program could require further testing (preferably via a separate test program) to verify the accuracy and expose any possible errors within the solution.

References

[1] "Science Alert". (2017, 05/05/2018). *Scientists Have Broken The Efficiency Record For Mass-Produced Solar Panels*. Available: https://www.sciencealert.com/researchers-have-broken-the-record-for-solar-panel-efficiency-again

[2] "Energy Matters". (2017, 05/05/18). *INFORMATION ON AUSTRALIAN SOLAR FEED-IN TARIFFS*. Available: https://www.energymatters.com.au/rebates-incentives/feedintariff/#victoria

[3] "Sustainability Victoria". (2017, 05/05/18). *Calculate appliance running costs*. Available: http://www.sustainability.vic.gov.au/You-and-Your-Home/Save-energy/Appliances/Calculate-appliance-running-costs

Appendix

Lab3.m

```
%Lab 3
%Solar Radiation
clc;
clear;

%%
%script

%import spreadsheets
darwin = xlsread("solar_data_darwin.xlsx");
```

```
melbourne = xlsread("solar_data_melbourne.xlsx");

%format darwin data
darwinDate = datenum(darwin(:,1),darwin(:,2),darwin(:,3));
darwinSolarExp = darwin(:,4);
xlimit1a = round(size(darwinDate)*(3/4));
xlimit1b = size(darwinDate);
limits1 = [darwinDate(xlimit1a(1)) darwinDate(xlimit1b(1))];

%format melb data
melbDate = datenum(melbourne(:,1),melbourne(:,2),melbourne(:,3));
melbSolarExp = melbourne(:,4);
xlimit2a = round(size(melbDate)*(3/4));
xlimit2b = size(melbDate);
limits2 = [melbDate(xlimit2a(1)) melbDate(xlimit2b(1))];


%plot darwin
figure(1)
plot(darwinDate, darwinSolarExp, '.')
hold on
plot(darwinDate, 5*cos(darwinDate/55 + 10) + 23,'LineWidth',1.5)
datetick('x','yyyy')
ylabel('Solar Exposure (MJ/m*m)')
xlabel('Year')
xlim(limits1)
title('Darwin')
ylim([0 40])



%plot melb
figure(2)
plot(melbDate, melbSolarExp, '.')
hold on
plot(melbDate, 15*cos(melbDate/55 + 10) + 20,'LineWidth',1.5)
datetick('x', 'yyyy')
ylabel('Solar Exposure (MJ/m*m)')
xlabel('Year')
xlim(limits2)
ylim([0 40])
title('Melbourne')

%monthly averages
DarStatArray = grpstats(darwin,darwin(:,2),'mean');
MelStatArray = grpstats(melbourne,melbourne(:,2),'mean');

%plot darwin monthly avg
figure(3)
bar(DarStatArray(:,2),DarStatArray(:,4),'r')
xlabel('Month')
ylabel('Avg Solar Exposure (MJ/m*m)')
title('Darwin Average Monthly Solar Exposure')

%plot melb monthly avg
figure(4)
bar(MelStatArray(:,2),MelStatArray(:,4),'r')
xlabel('Month')
ylabel('Avg Solar Exposure (MJ/m*m)')
title('Melbourne Average Monthly Solar Exposure')

%set efficiency
eff = 0.6; %median of range
energyReq = 6; %kwh required for the house

%mean exp for year
```

```
meanDarSolarExp = mean(DarStatArray(:,4));
meanMelSolarExp = mean(MelStatArray(:,4));

%calc mod sizes
MelModS = solarCalc(meanMelSolarExp,eff,energyReq);
DarModS = solarCalc(meanDarSolarExp,eff,energyReq);

%calc cost
DarCost = energyCalc(DarModS,eff,DarStatArray(:,4),energyReq);
MelCost = energyCalc(MelModS,eff,MelStatArray(:,4),energyReq);

%plot costs each month
figure(5) %dar cost
bar(1:1:12,DarCost,'g')
xlabel('Month')
ylabel('Cost ($AUD)')
title('Darwin Home Average Cost per Month')

figure(6) %mel cost
bar(1:1:12,MelCost,'g')
xlabel('Month')
ylabel('Cost ($AUD)')
title('Melbourne Home Average Cost per Month')

fprintf('The Darwin home would need a module size of %g kW\n',DarModS);
fprintf('The Melbourne home would need a module size of %g kW\n',MelModS);
```

energyCalc.m

```
function cost = energyCalc(modSize,eff,solarExp,req)
    nrgArr = modSize*(solarExp/3.6)*eff;

    nrgArr = nrgArr - req;
    cost = zeros(1,12);
    avgMonth = 30.42; %avg days in a month
    gridCost = 0.28; %cost of power from grid
    gridDebit = 0.113;%amount paid for selling power to the grid

    for i = 1:size(nrgArr)
        x = nrgArr(i);
        if x < 0 %underproducing/buying from griid, +cost
            cost(i) = cost(i) - (gridCost * x * avgMonth); %adds cost of grid draw
        elseif x > 0 %overproducing/selling to grid, -cost
            cost(i) = cost(i) - (gridDebit * x * avgMonth); %substracts cost of selling
        end
    end

end
```

solarCalc.m

```
function modSize = solarCalc(exposure,eff,energy)
   modSize = energy/(eff*(exposure/3.6));
   modSize = ceil(modSize * 4) / 4;
end
```