Analysis of TCS3400 Data Quality Issue and Correction

Bobby Schulz - 9/27/2023

Executive Summary

A long-standing firmware issue was discovered with the internal light sensor (to be used to approximate PAR) on the v2 GEMS Sensing data loggers. This issue significantly impacts the usefulness of this data but can be corrected to a good degree using a post processing algorithm. This issue is present on all loggers from summer 2021 to fall 2023. Issue will be corrected going forward. Issue does not affect v3 loggers. This issue does not affect situations where PAR is below ~300 μmol·m−2·s−1.

Introduction

Background

The TCS3400 is used on all GEMS Sensing v2 data loggers for collecting ambient light intensity from within the logger box. The goal was to use this sensor, along with a formula to correct for spectral qualities of the box lid, to measure Photosynthetically Active Radiation (PAR) in a low cost manner. This metric is of particular use for turf grass and irrigation experiments.

Problem Discussion

In September 2023 data analysis was performed on the TCS3400 Ambient Light Sensor (ALS) data from the v2 systems in order to determine the conversion formula to translate red, green, and blue light intensity into an approximation. During this analysis a long-standing firmware issue was discovered which resulted in improper reporting of the TCS3400 data.

Scope of Impact

This issue affects all v2 data loggers from summer 2021 to September 2023. v3 loggers are not affected.

The issue in question only affects measurements of reasonable value. Based on the conversion factor, which is discussed later, between ALS data and PAR - this issue is only relevant for situations where PAR exceeds ~300 μmol·m−2·s−1. Below this no error is introduced and so no correction is needed.

Source of Issue

The issue was caused by an incorrect mapping of gain values in the TCS3400 driver firmware. It was necessary to implement Automatic Gain Control (AGC) for the TCS3400 so that regardless of the light intensity, an accurate measurement could be made.

During each logging interval the AGC is updated and a new sample is taken. The TCS3400 reports output values in counts, which range from 0 to 65536. When the sample is reported by the data logger it is important that these counts be normalized based on the gain used to take the sample.

This normalization is done by simply dividing the value from the ALS by the maximum possible value (65536) to get a value between 0 and 1, then take this value and divide it by the gain used in performing the sample, as determined the the AGC algorithm. The result of this is a value between 0 and 1 where 1 is the maximum light amplitude the sensor is able to report.

The normalization equation is as follows

$$Channel_{Norm} = (ALS_{Out}/ALS_{Max})/(ALS_{Gain})$$

Where Channel_Norm is the normalized output desired, ALS_Out is the output from the TCS3400, ALS_Max is the max value of a given channel (2^16 = 65536) and ALS_Gain is the commanded gain of the ALS at the time of measurement (1, 4, 16 or 64)

However, when the equation was implemented in the firmware it was implemented incorrectly so that the ALS_Gain is multiplied instead of divided. This results in the reported value being inversely scaled in an inconsistent way.

The implications of this can be seen in Figure 1.

We see the normalized PAR value and the TCS3400 blue channel output side by side over the course of a single day. We see the ALS output increasing consistently with the PAR until a distinct point, annotated with a dotted black line. At this point the light intensity exceeds what can be measured at the higher gain value. The AGC detects this and reduces the gain accordingly. When this happens, the gain reduction reduces the output value by a factor of 4 and *decreases* the correction multiplier by a factor of 2, resulting in an overall gain reduction of 8. In this case two gain steps are made (since the rising sun increases the light intensity more rapidly than we are sensing), so the apparent reduction in amplitude is two factors of 8, or 64.

This means that instead of observing the anticipated increase of about 1.5 times to keep pace with PAR, we see a decrease by a factor of nearly 64.

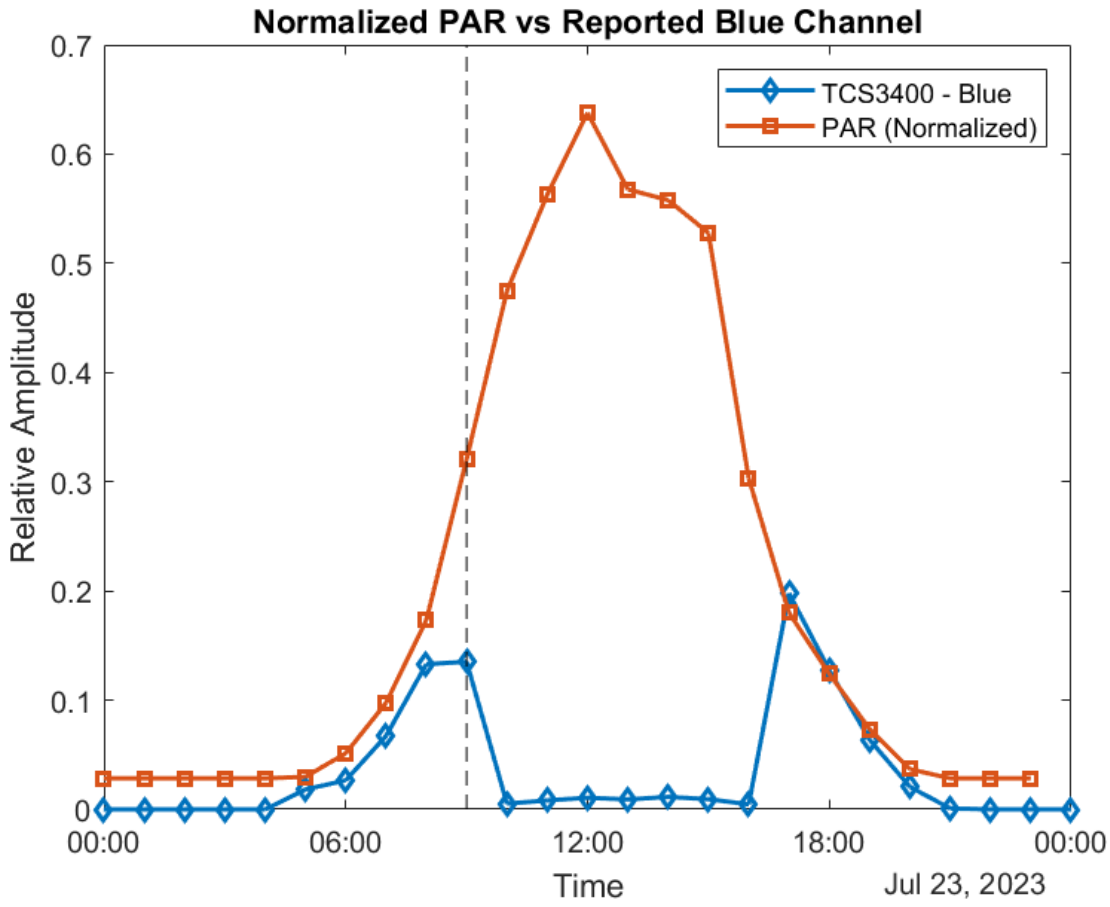


Figure 1

While the impact on the data is significant, it also is visually evident where this error is occurring and it has a particular signature to it which differs from the usual system variance. Our goal is to generate an algorithm which compensates for this erroneous scaling to attempt to at least partially recover the original data.

Methods

In order to attempt to recover from the errors in the ALS data, an algorithm was developed to correct for the gain error by detecting the uncharacteristic changes in output.

All modeling and testing was done on a set of data taken over the course of 3 months in a side by side setting with a v2 node recording ALS sensor data using the built in TCS3400 and a colocated v3 node which used an Apogee SQ-500-SS full spectrum quantum sensor to measure PAR.

Algorithm Development

The core principle of the algorithm is to examine the amount of change between any two adjacent data points and, if the change is sufficiently large, adjust the correction factor.

A correction factor is assigned which is used to scale the input data using a log base 2 system in order to emulate the scaling effect which was inadvertently imposed on the data. This correction factor is updated when a significant increase of decrease of the data is observed and otherwise retains the previous value. First the instances of high rate of change must be sifted out. This is done by taking the data, shifting it one point, then dividing the original data by the shifted result. This result defines the ‘rising’ edges of the system and taking the inverse value of this array gives us an array which defines the ‘falling’ edges of the system. Once these arrays defending the rising and falling events are created, they must be cleaned. First all ‘low points’ are set to zero. Low points are defined as any point where both the original data and shifted data is below a threshold. This ignores errors in very low light situations, in which the data should not be affected anyway, but due to the increased effect of noise reasonable changes could appear significant and trigger the gain adjustment if not filtered out. In addition, all values less than 1 of the rising and falling arrays are set to zero - this indicates the opposite characteristic and will be handled by the complementary array. Next the exponential value is calculated for each by taking the log base 2 of the rising and falling arrays - this converts the values into a frame of reference which is easier to work with. To these exponential arrays, we set all values below a threshold of change to 0. This threshold is the value below which we do not want to gain to be adjusted. We consider this value to be the crossover point between ‘normal’ changes and ‘erroneous’ changes. An exponential value of 3 (2^3 = 8) was chosen as a starting point as the smallest change due to a gain variance we would expect is a factor of 8. It was then empirically found that a value of 2.75 (2^2.75 = 6.73) produced the best R^2 value when compared to PAR and so was chosen.

From this point the following logic was implemented. For each point in the data, if the rising exponent is greater than 0, adjust the gain down by subtracting the rising exponent from the previous gain value. If the falling exponent is greater than zero, then instead adjust the gain value to the falling exponent. If the input value is 0, then set the gain to 0. A zero data value is only observed at true night and we use night as a baseline when we know the state of the system and therefore can be sure about the gain correction to use. If none of these conditions is met, then the gain is kept the same.

Once the gain correction has been calculated, an iterative process of reducing erroneous gain is performed. The algorithm finds all cases where the gain correction applied to the input data results in a value greater than 1. This means an incorrect gain (too high) was applied. For each such point, we reduce the gain exponent by 1 until the output value is less than 1. This solution

is not ideal, but is used to deal with the rare[1] incorrect gain detection. This is more a data cleaning step than an algorithmic step.

Lastly, the gain correction is applied to the entire input.

See code in Appendix A

Algorithm Tuning

In order to fully tune the algorithm, it was necessary to evaluate which combination of bands would best represent PAR. In previous work a combination of the red and blue bands has been used to best represent the PAR bandwidth. This is perfectly reasonable since as we can see in Figure 2 and Figure 3, the photosynthetically active range should overlap well with the red and blue filter bands of the device.

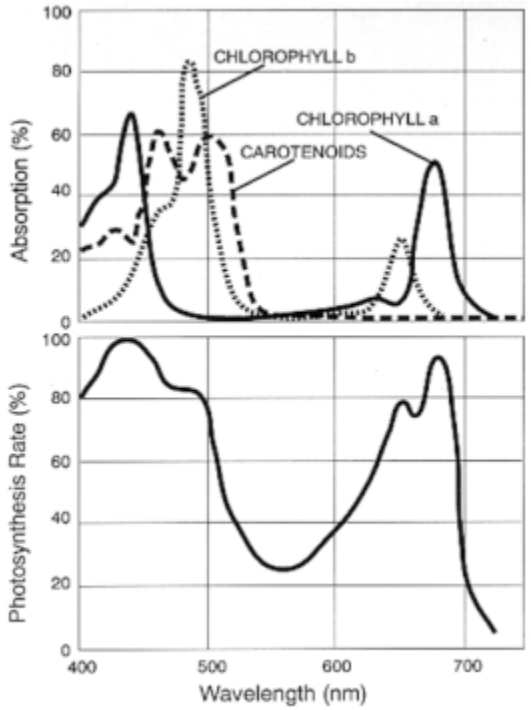


Figure 2
PAR action spectrum - taken from Wikipedia

[1] In a test of 7046 data points, we observed 42 such excessive occurrences

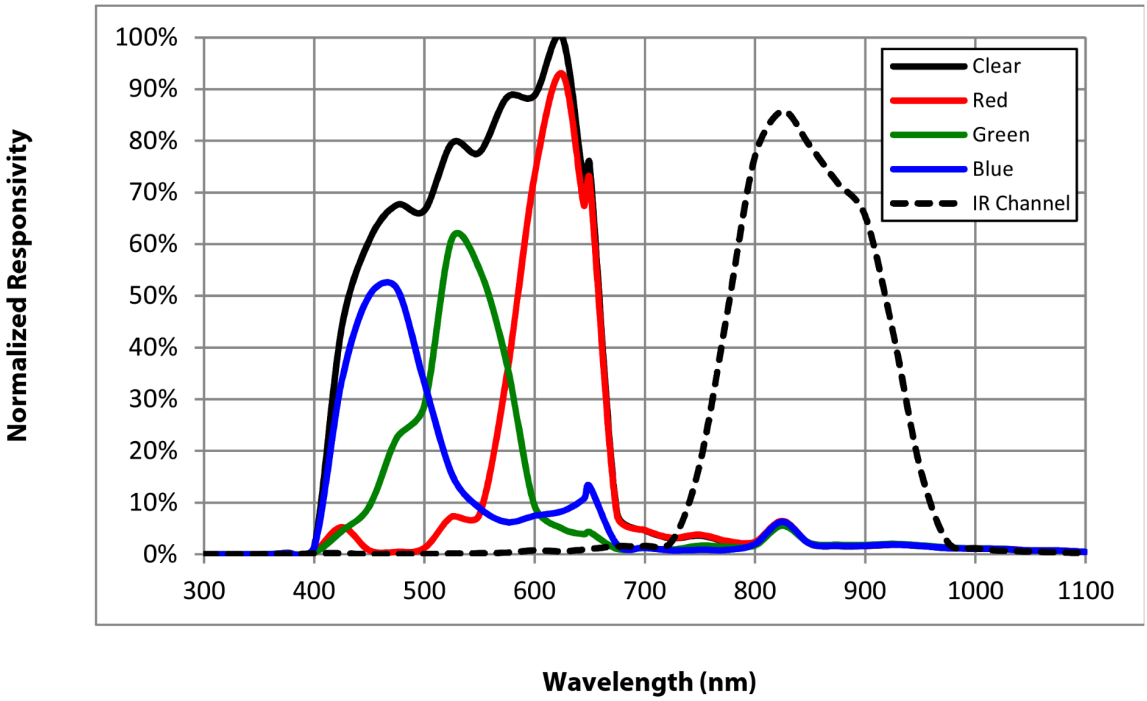


Figure 3
TCS3400 Spectral Response

However, when we perform evaluation on the difference in band output, we find that they are strongly correlated and we see little difference in the band to band response. This is reasonable since the input light source is broad spectrum (sun light) and the filtering effects of the polycarbonate box should be relatively uniform across the frequency range in question.

In our testing it was found that the blue channel alone had the highest correlation with PAR - as a result, this is what was used for algorithm tuning.

PAR Conversion

Once the algorithm had been tuned and the correlation of the output data and PAR was optimized, the scalar multiple needed to be found to convert the corrected blue channel output into PAR (μmol·m−2·s−1).

First the ADC offset was removed from the PAR data - see Figure 4. This offset is an effect of the ADC as this flux is constant, and at night we expect it to be 0.

We then find the scalar value which minimizes the RMSE between PAR and this modeled PAR.

The result of this ends up being 2190.

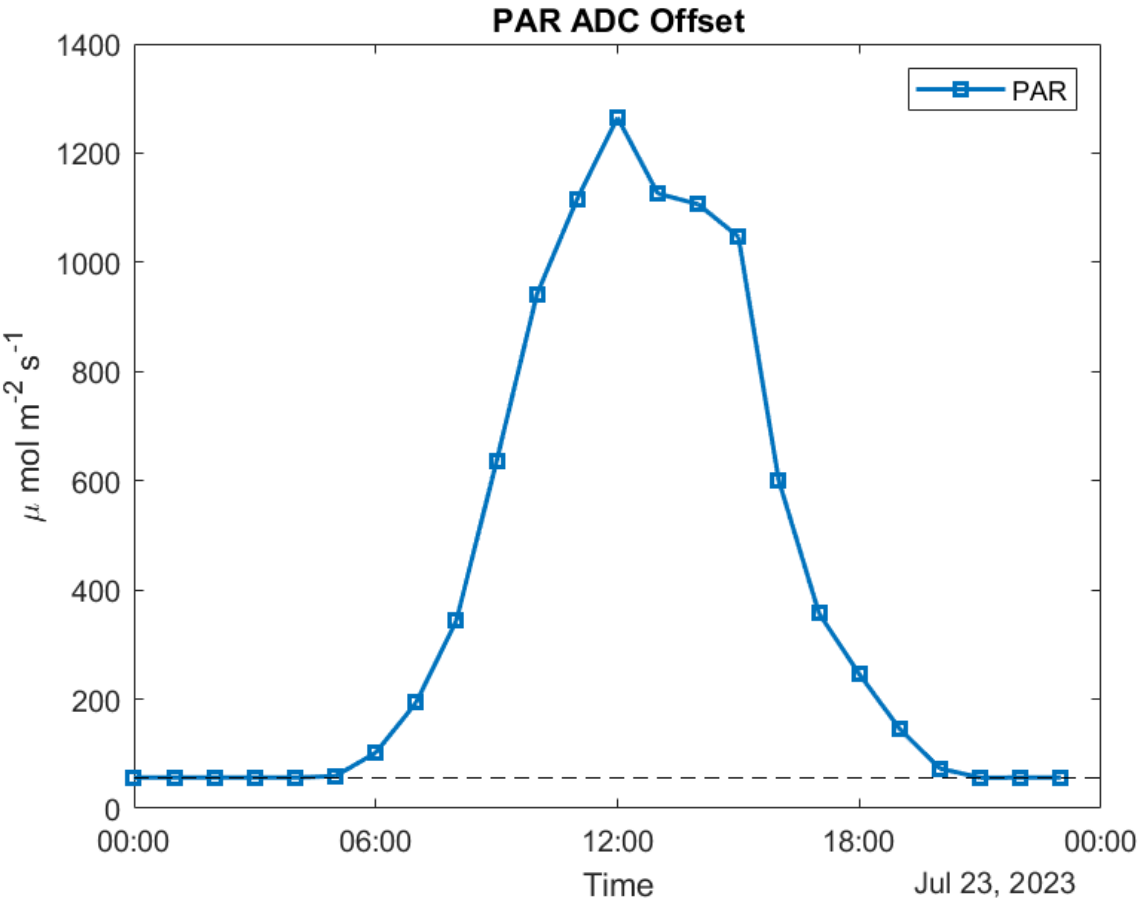


Figure 4

Results

Algorithm Evaluation

Once the algorithm was tuned, the same example day was examined as a representative sample. The results of this can be seen in Figure 5. It is clear that the response is significantly improved over the raw data.

Once the algorithm was tuned, it was tested across hourly mean data for 3 months worth of data - 7046 hourly mean data points. The correlation (R^2) between a given set of bands and PAR is as follows for the various combinations.

	Red	Green	Blue
Red	0.584	0.593	0.648
Green	0.593	0.586	0.650
Blue	0.648	0.650	**0.681**

Combination of Red, Green, Blue resulted in an R^2 of 0.635

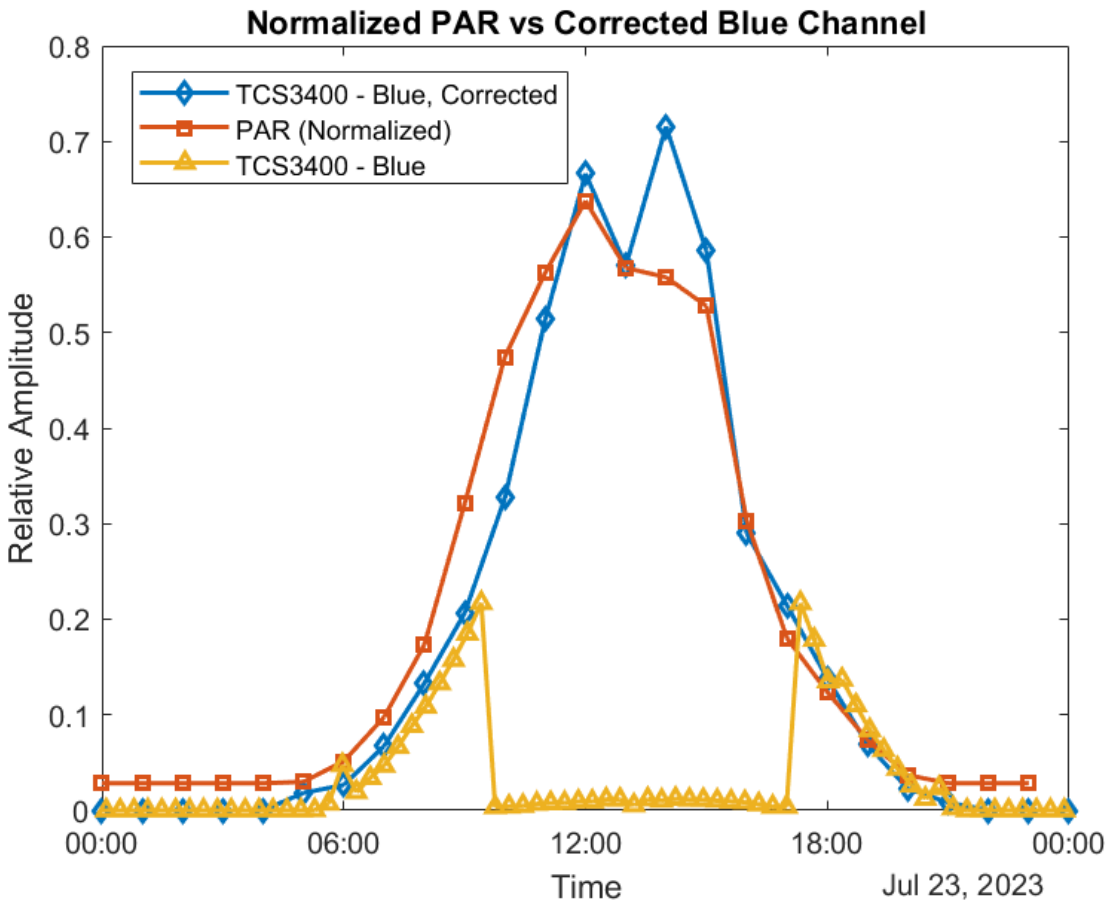


Figure 5

As we see from the table of various combinations of bands - the blue band alone performs the best at modeling PAR given the data available.

PAR Conversion

Once the data was converted into the correct units (μmol·m−2·s−1) it was compared to the ground truth PAR value. The results are as follows across the same set of test data previously used.

Metric	**Value**
R^2	0.681
RMSE	277.56
Mean Absolute Error	155.50
Mean Error	85.51

We see the results of this conversion for the selected day in Figure 6, we can also see the regression results of the fit in Figure 7

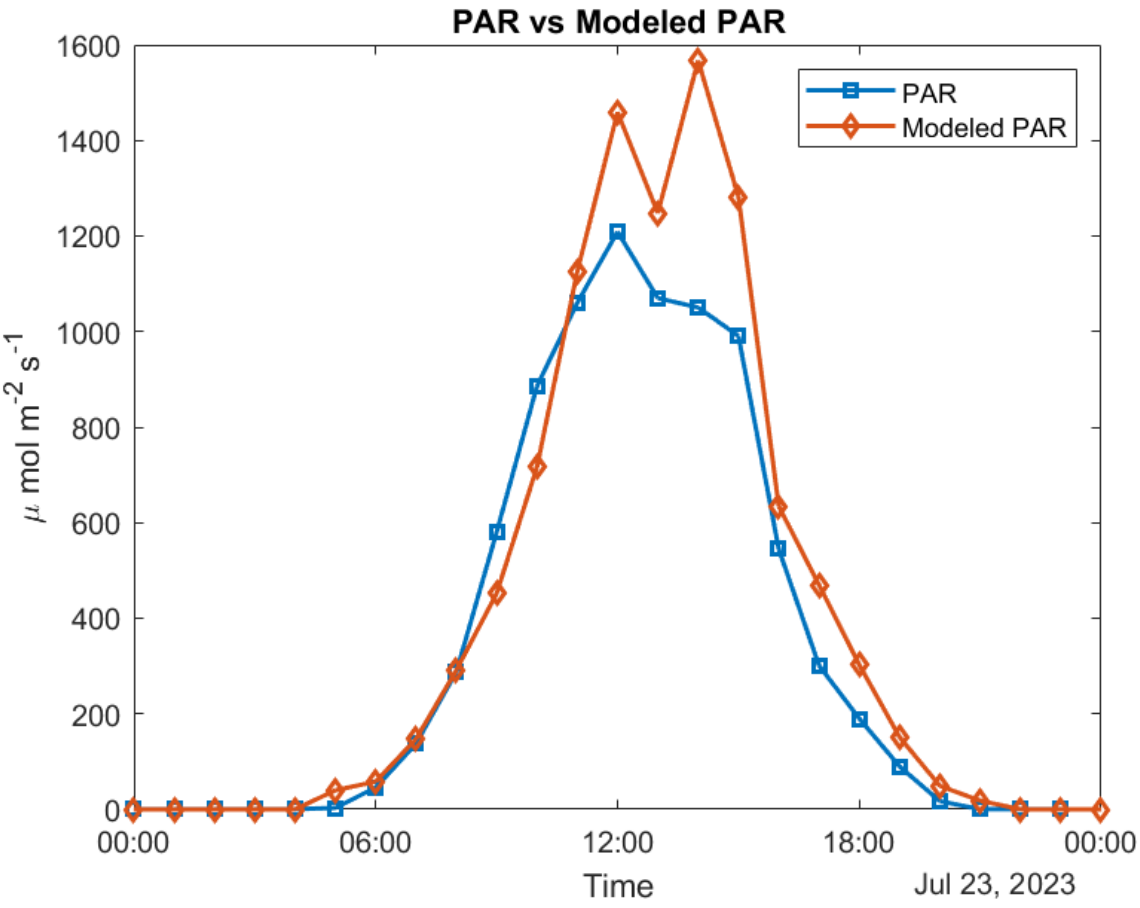


Figure 6

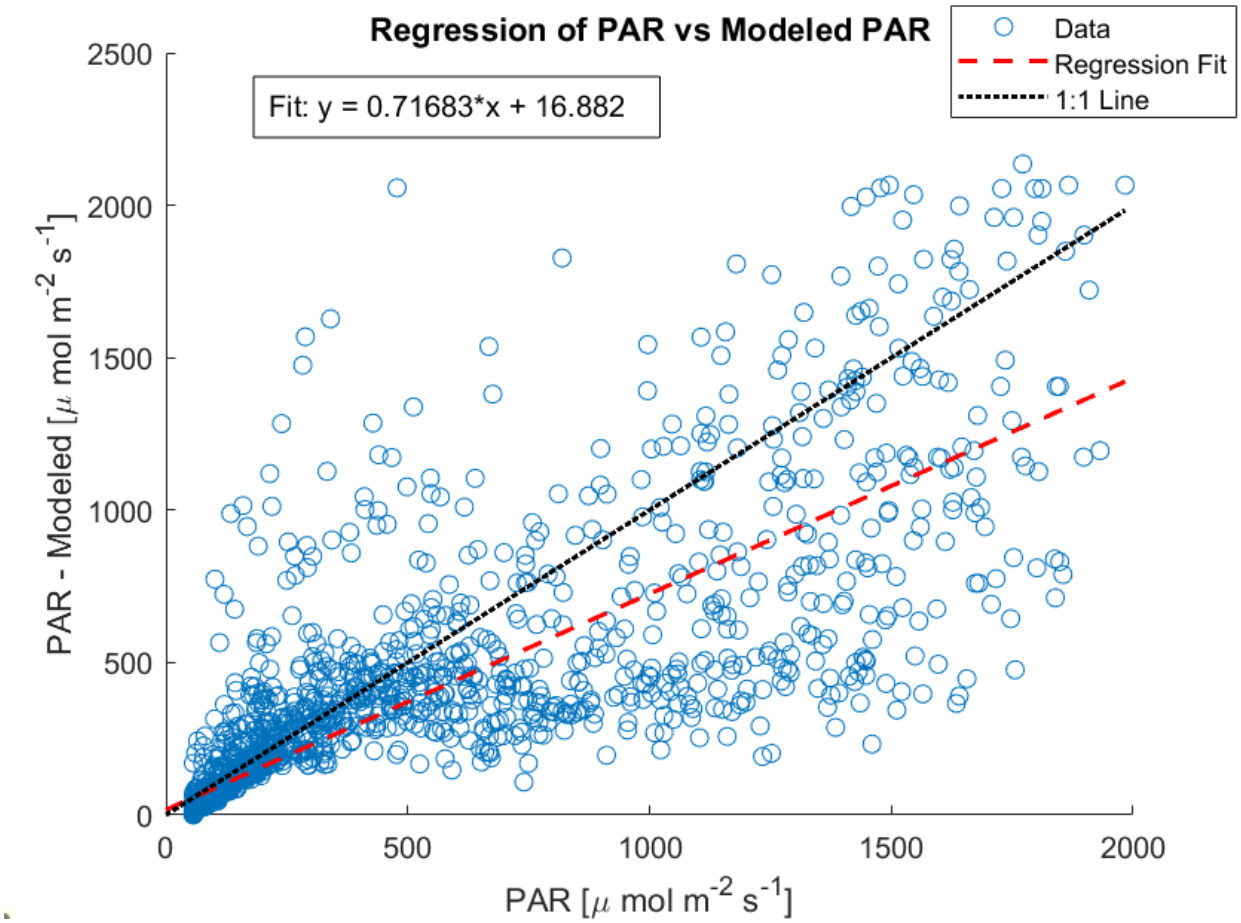


Figure 7

Discussion

While this is not the quality of data we were hoping for - this would seem to be a reasonable correlation given the reconstruction and attenuation involved in the system. We feel this data will still be a useful tool in understanding the system in which the logger is deployed.

While the correlation is not ideal, the overall system error is quite good - with the mean error being less than 5% of the total range of the signal.

These results should give a reasonable approximation of PAR - especially over large time scales.

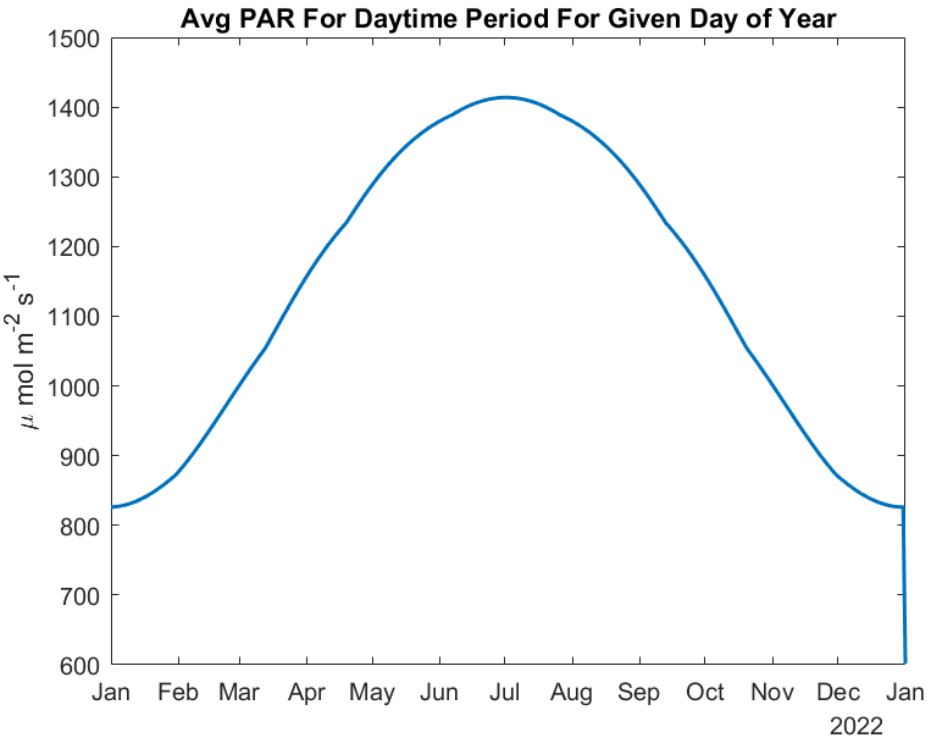
Appendix A

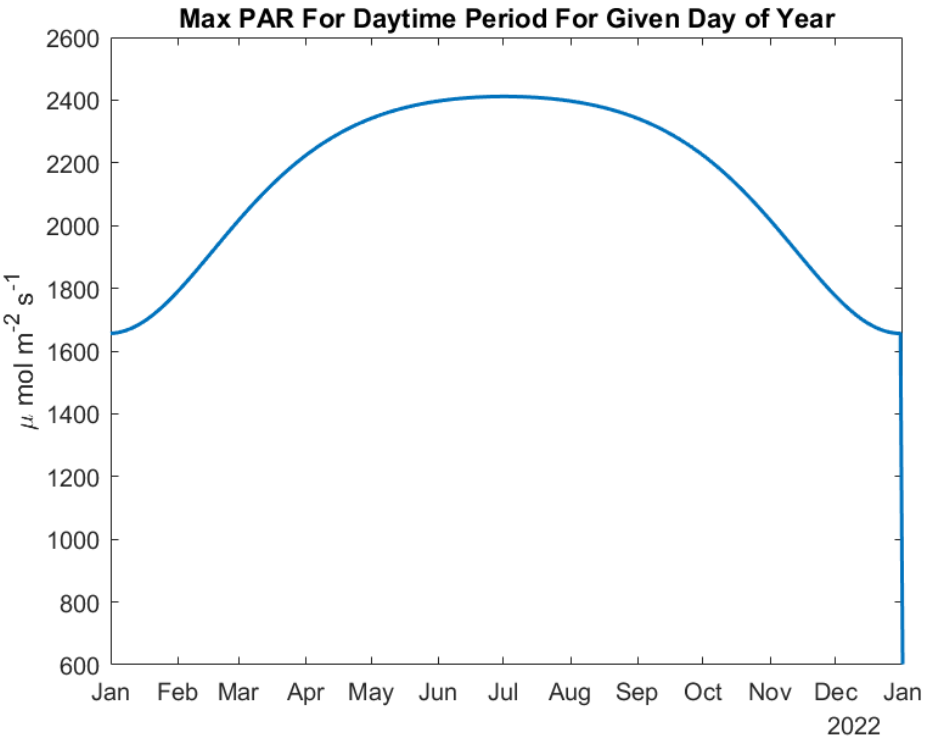
```
Unset
InputValues = TestTT.Blue; %Input data
InputShift = circshift(InputValues, 1); %Shift all values by 1
InputShift(1) = InputValues(1); %Copy the first value back so division = 1
Rising = (InputValues)./InputShift; %Generate arrays which describe the rising and
falling events of the signal
Falling = InputShift./(InputValues);

LowPoints = (InputShift < 0.01) & (InputValues < 0.01); %Find places where both test
points are tiny
Rising(LowPoints) = 1; %Set these tiny points to no change
Rising(Rising < 1) = 1; %Set all negative changes to steady state. Any actual gain
changes will be handled by the complementary test
Falling(LowPoints) = 1;
Falling(Falling < 1) = 1;

GainCorrectExp = ones(height(TestTT), 1); %Generate blank array
RisingExp = (log2(Rising)); %Take log base 2 of scale of change, round this to get
integer - desire nearest integer for best fit
FallingExp = (log2(Falling));
RisingExp(RisingExp < 2.75) = 0; %Only look at gain changes greater or equal to ~8x
FallingExp(FallingExp < 2.75) = 0;
PrevGain = 0;
NewGain = 0;

for i = 1:height(InputValues) %Iterate over whole array
  if RisingExp(i) > 0 %If there is a significant rising event, reduce the gain
correction
    NewGain = PrevGain - RisingExp(i);
  elseif FallingExp(i) > 0 %If there is a significant falling event, increase the gain
correction
    NewGain = FallingExp(i);
  elseif InputValues(i) == 0 %Return to min gain at dark periods
    NewGain = 0;
  else
    NewGain = PrevGain; %If no significant events, continue with previous gain
  end
  if NewGain < 0 %Clamp to 0
    NewGain = 0;
  end
  PrevGain = NewGain; %Update gain values
  GainCorrectExp(i) = (NewGain);
```

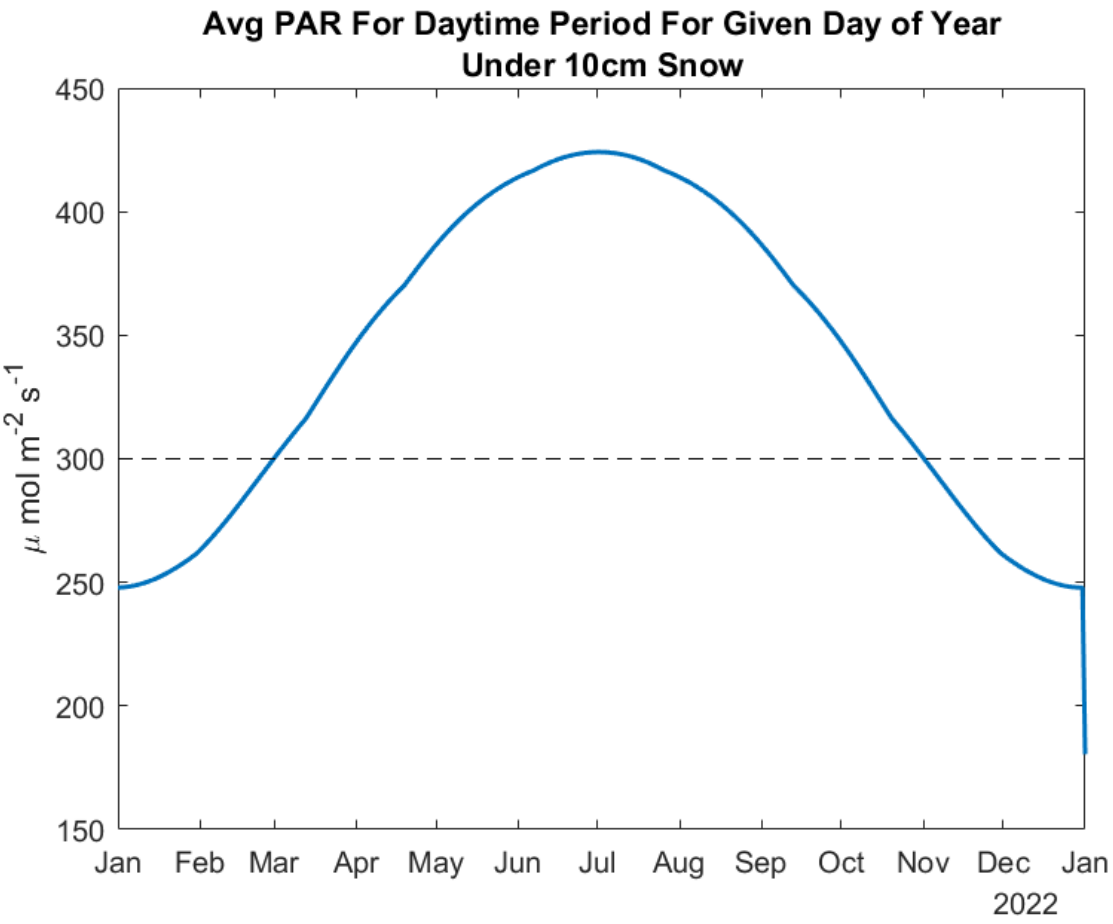
```
end

CorrectedOutput = (InputValues).*(2.^(GainCorrectExp));
%Reduce any excessive gain instances
sum(CorrectedOutput > 1)
while sum(CorrectedOutput > 1) > 0 %Repeat reducing gain until no corrected values
exceed 1
  GainCorrectExp(CorrectedOutput > 1) = GainCorrectExp(CorrectedOutput > 1) - 1;
%Decrease gain for all instances where corrected output exceeds 1
  CorrectedOutput = (InputValues).*(2.^(GainCorrectExp)); %Update test values
%   CorrectedOutput = (InputValues).*(8*GainCorrectExp); %Update test values
end
OutputValues = (InputValues).*(2.^(GainCorrectExp)); %Calculate final output
```

Appendix B

Solar values for various regions under various conditions






Avg PAR For Daytime Period For Given Day of Year
Under 10cm Snow
450
400
350
300
250
200
150
μ mol m-2 s-1
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2022