Statement of purpose:

This code is from the early stages of my student research on optimizing multi-junction non-current matching solar panels. These use multiple active layers—materials that convert solar radiation into electric current—in a single panel to increase efficiency of solar radiation to electric current. Each active layer converts photons of an energy higher than it's band gap, which is the energy needed to push an electron from that material. The powerout put from a layer can be approximated by multiplying the short circuit current by the open circuit voltage. This voltage is a product of the band gap so the higher the band gap the higher the voltage.

The program estimates power conversion using measured data of the AM 1.5 solar spectrum. This gives data for the density of photons per change in wavelength per meter squared at the earth surface. The program generates combinations of band gaps in descending value (E.G. [3,2,1] but not [2,3,1]) so that that high energy photons ejects electrons from higher band gap cells while lower energy photons are captured by lower cells. This model provides a quick estimate to show where maximum efficiencies might be located so that more complex calculations might be done in a targeted manner. The program uses trapezoidal numerical integration to find the number of photons that a band can capture and then using this to calculate current. Ideal materials are assumed giving 90% conversion of photons to electric current and a voltage 90% of that corresponding to the band gap are assumed.

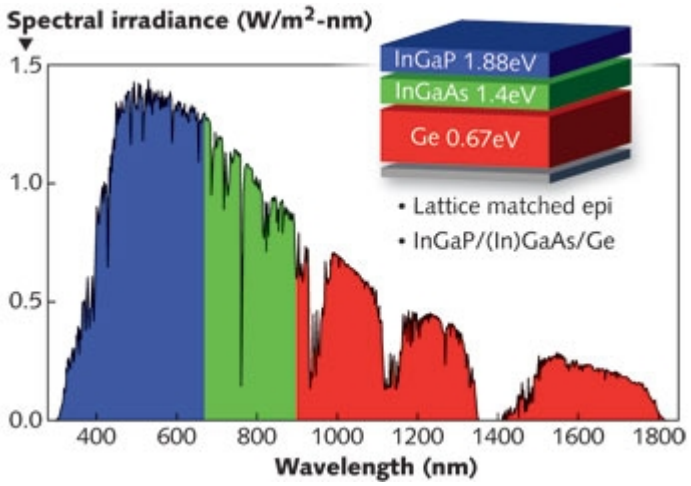

Figure 1. Showing an example of how the solar spectrum is gathered by the layers. The blue area is photons absorbed by the top layer, the green by the middle layer, and the red by the bottom layer.

Instructions for running:

One may open fullnpanel_approx.py and edit the parameters for the generation of the band gap combinations in the “user input” section. It is set to three panels currently. These parameters are listed as “bands_ranges_input” and “brand_voltage_spacing” the first of which changes the number of band gaps and allowed values, the second changes the spacing of the generation of band gaps (I.E. if they are .01 eV apart or some other value).

If this is not desired one may simply run this to generated a CSV file labeled as “outputxpanel.csv” with the band gaps listed in in order from top to bottom in the panel and the efficiency listed in the last column. Here “x” is the number of active layers used or the number of band gaps in the calculations.

The number of most efficient values with their location in the CSV file will be printed. The user may change this value in the user input section. It is listed as “top_values_input”.

If there are two panels or three panels the fullnpanel_approx.py file will automatically call a second program make an interpolated contour plot (cubic interpolation) or a three dimensional scatter plot with coloring of the points to indicate efficiency.

Files:

Fullnpanel_approx.py will build the list of values and call other files as needed.

Contour_plot.py is a file which will interpolate the data for a two layer panel and plot the lines of efficiency in a contour surface plot.

Threedscatter.py will build a three dimensional plot of points in a three dimensional space where each axis represents the band gap of a layer and the color of the plotted points represents the efficiency of that combination.

Results, discussion, and future work:

This code was meant to give some insight into what range of band gaps should be the area of concentration for further modeling with more computationally intensive methods. This model does not incorporate a number of real world phenomena that can reduce efficiency such as losses to reflection, the conversion of photons into thermal energy, and the effects of increasing temperature on the operation of the panel. There was a breakdown in accuracy for panels with more than four layers where optical effects become of significant consideration. Further work would include these effects in initial calculations rather than accounting for them after estimating the areas of highest efficiency. To increase computational efficiency it may be necessary to refine algorithms for calculation or possibly use methods for parallel computing to better control processor usage.

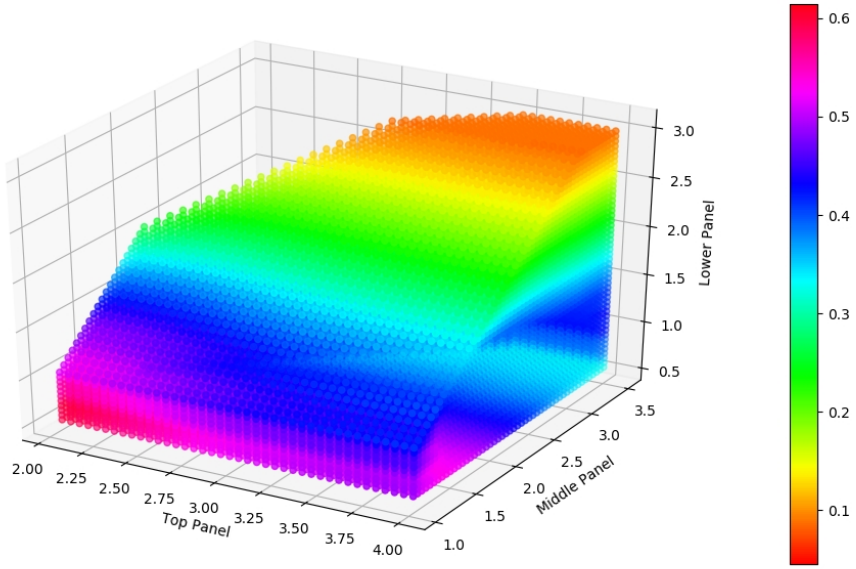

Figure 2. This is an example of the scatter plot output for a three layer cell. That is a cell with three independent band gaps.