Multiple Junctions Maximize Efficiency under many Marine Environments

1 Motivation

Despite covering over 70% of the Earths surface, very little of our oceans have been explored[6]. Techniques used to survey the oceans generally fall into three groups: surface instruments, manned submersibles, and unmanned submersibles. Surface instruments are devices located above or slightly below the surface of the ocean, carrying sensors capable of monitoring the conditions of the water. While some instruments, such as sonar, can map the sea floor, surface instruments are generally limited in depth and resolution. Meanwhile, manned submersibles, generally submarines, can survey deeper marine environments. They are used to study underwater terrains and ecosystems, but the need for life support limits range, increases costs, and cannot be used in hazardous environments. In these cases, unmanned submersibles, such as autonomous underwater vehicles (AUVs) and remotely operated underwater vehicles (ROUVs), are deployed.

AUVs and ROUVs have already found use in scientific, industrial, and defense applications[7], such as for geological, archaeological, and environmental study or for the repair and maintenance of underwater structures. In some uses, the vehicles are submerged for a limited amount of time and can be serviced after surfacing. However, for applications over longer timescales—especially in the implementation of an autonomous 'Internet of Underwater Things'—the lack of safe, persistent power sources becomes a limiting factor. Tethered power require a nearby surface vessel while batteries are limited in capacity and lifespan.

Solar power is a promising alternative to these options. On land and in space, solar cells are proven to be reliable and effective power sources, experiencing widespread integration throughout industries. However, there are several factors hindering their adoption underwater. Underwater solar cells face the harsher marine environment and must avoid damage from chemical and mechanical stresses. Plants, algae, and microorganisms can accumulate on surfaces and block light, requiring additional biocidal treatments.

In this paper, we address another issue: the different solar spectrums available at different underwater depths. The maximal efficiency of a cell depends on the solar spectrum, as determined by Shockley-Queisser (SQ) limit. On land, silicons bandgap of 1.12eV is near the theoretical best, but underwater and in space, solar spectra, optimal bandgap, and optimal material differ. A major difference from extraterrestrial applications is that AUVs and ROUVs may explore multiple depths with differing solar spectra: as a result, the optimal bandgap can range anywhere from 1.7 to 2.4eV[5]. Additionally, the AUV/ROUV may surface to communicate or maximize solar coverage, and in that case, the optimal bandgap becomes that of land: 1.34eV. We propose a dual-junction solar cell based on the InGaN and CIGS material systems to optimize efficiency across all these scenarios.

2 Approach

Our approach can be divided into three parts. First, we model the solar spectrum at different depths and compute power intensities. Second, we find the optimal bandgap based on the two-junction SQ limit. Finally, we select two materials and designed a potential solar cell structure.

2.1 Modeling Solar Spectra

Pure water is slightly blue. This is because the vibrational modes of waters O-H bonds slightly extend from infrared into the visible spectrum [4], preferentially absorbing redder wavelengths and transmitting a bluer spectrum. However, many other factors influence the absorption spectra in real-life waters, such as the concentration of suspended particles or microorganisms and whether it is saltwater or freshwater [5]. Typically, these waters have similar absorptivity from 600 to 800 nanometers while differing significantly at lower wavelengths. For this project, we decided to use the middle-of-the-road absorption spectra tabulated by Smith et al., based on waters in the Sargasso Sea [1].

The metric of interest is the diffuse attenuation coefficient K_w, which combines the effects of absorption and scattering. We used to following equation to determine the amount of light transmitted to a particular depth, using the standard air mass 1.5 (AM1.5) spectrum as the solar spectrum at the surface.

$$P(\lambda, d) = P(\lambda, 0) \cdot e^{-d \cdot K_w(\lambda)}$$

Figure 1 plots the total irradiance by depth. AM1.5 is normalized to 1000 $\frac{\text{W}}{\text{m}^2}$, but within a few meters this quickly diminishes to half that value as longer wavelengths are rapidly absorbed. At 10 meters, the total power reduces to 250 $\frac{\text{W}}{\text{m}^2}$, and at 50 meters, the total power is only 70 $\frac{\text{W}}{\text{m}^2}$. Figure 2 plots the irradiance by wavelength at the surface, at 10 meters, and at 50 meters. As expected, most light transmitted is between 300 and 600 nanometers.

2.2 Shockley-Queisser Limit

The famous Shockley-Queisser limit combines several forms of losses to determine a fundamental efficiency limit for a solar cell with a limited number of junctions

[8]. The largest loss results from photons with differing energies from the bandgap. Photons with higher energies than the bandgap will excite electrons deep into the conduction band, and as the electron falls back to the edge of the conduction band, the extra energy is lost to heat. Photons with lower energies than the bandgap will not be absorbed at all, resulting in a complete loss. Typically, this loss is reduced by adding extra junctions to match bandgaps to photon energies.

The second largest loss results from radiative recombination. An electron and a hole can meet and recombine, emitting light with the bandgap energy. The rate at which this occurs depends on the applied voltage and the blackbody formula. When the applied voltage is zero, the radiative recombination rate is determined by the following equation, derived from Planck's law:

$$RR(0) = \frac{2\pi}{c^2 h^3} \int_{E_g}^{\infty} \frac{E^2}{\exp\left(\frac{E}{kT}\right) - 1} dE$$

As the applied voltage changes, the radiative recombination rate scales exponentially: $RR(V) = RR(0)e^{\frac{Ve}{kT}}$. This results from the hole and electron quasi-fermi levels moving past each other and increasing the likelihood of an electron and hole pair meeting each other (higher pn).

We implemented these two effects in Python (available in Appendix B) and derived the efficiency limit by bandgap, shown in Figure 3, which is consistent with the expected results. Next, we calculated the efficiency limits for a two-junction cell with bandgaps between 0.5eV and 3.5eV. Figure 4 shows this map for the AM1.5 spectrum, the 10m spectrum, and the 50m spectra from Figure 2. Additionally, we plotted the average of all three efficiencies, since we are looking to optimize a cell that operates under multiple spectra. The spot with the highest efficiency is annotated for each of these maps. As expected, the optimal set of bandgaps occurs with a low-energy bandgap absorbing lower-energy photons and a higher-energy bandgap absorbing higher energy photons. Note that the maximal efficiency increases with depth because the photon energy range is reduced to lower wavelengths. The optimal pair of bandgaps for the combined map is 1.35eV and 2.36eV at an average efficiency of 56.09%.

2.3 Material Choice

After considering several potential material systems, we decided to choose InGaN and CIGS as the materials for the higher and lower bandgap junctions respectively. Firstly, these materials can be tuned to the optimal bandgaps derived earlier: $In_xGa_{1-x}N$ ranges from 0.7eV to 3.4eV based on the value of x, and CIGS ($CuIn_{1-y}Ga_ySe_2$) can be tuned from 1.01eV to 1.63eV based on the value of y. Specifically, the Crosslight material library uses the following formula for calculating the bandgap:

$$In_xGa_{1-x}N\text{: } E_g = 0.7x + 3.4(1-x) - 1.42x(1-x)$$
$$CuIn_{1-y}Ga_ySe_2\text{: } E_g = 1.01 + 0.626y - 0.167y(1.0-y)$$

Solving these values for the wanted bandgaps, we find the optimal element proportions x= 0.28 and y= 0.61.

Another factor to our choice of InGaN and CIGS is their relative lack of toxicity. While there are other semiconductors with bandgaps tunable to 1.35eV and 2.36eV, such as GaInAs and ZnHgSe, we were interested in limiting the amount of heavy metals in our design. Indium, gallium, copper, selenium, and nitrogen are considered relatively nontoxic compared to these metals, which is important since these cells would be used in marine environments where stresses could release them into the water.

3 Device Design

As an initial starting point, we implemented in Crosslight the dual-junction InGaN and CIGS device proposed by Farhadi et al. [3], modifying the elemental proportions to those derived above. However, the upper InGaN junctions efficiency was several orders of magnitude lower than expected. We initially thought that this was due to the CdS window, since CdS has a bandgap of 2.42eV [2], causing it to filter out most of the high energy light before it reaches the InGaN junction. Yet even after removing the window, the performance of the InGaN junction was still extremely poor. After trying many alterations, including adjusting layer widths, doping levels, mesh densities, and simulation settings, we found that adding a small intrinsic region between the p-doped and n-doped layers was able to substantially increase the cell efficiency. We eventually optimized the device to the following structure:

	Contact	
0.3μm	$In_{0.28}Ga_{0.72}N$	p=8×10^{19}cm^{-3}
0.06μm	$In_{0.28}Ga_{0.72}N$	
3.0μm	$In_{0.28}Ga_{0.72}N$	n=6×10^{18}cm^{-3}
	Contact	
0.1μm	Vacuum	
	Contact	
0.1μm	$CuIn_{0.39}Ga_{0.61}Se_2$	p=1×10^{19}cm^{-3}
3.2μm	$CuIn_{0.39}Ga_{0.61}Se_2$	n=1×10^{14}cm^{-3}
	Contact	

The final Crosslight simulation code is available in Appendix C.

	CIGS V_{oc} (V)	I_{sc} ($\frac{A}{m^2}$)	P_{max} ($\frac{W}{m^2}$)	FF	InGaN V_{oc} (V)	I_{sc} ($\frac{A}{m^2}$)	P_{max} ($\frac{W}{m^2}$)	FF	Total P_{max} ($\frac{W}{m^2}$)
AM1.5	0.7	85.1	46.0	76.2%	1.6	24.6	34.4	86.7%	80.5
10m	0.6	7.6	3.4	75.1%	1.6	19.1	26.4	87.0%	29.8
50m	0.5	0.5	0.2	69.5%	1.6	7.6	10.2	86.6%	10.4

Table 1. Open circuit voltage, short circuit current, maximum power, and fill factor of CIGS and InGaN junctions at AM1.5, 10m, and 50m solar spectra.

4 Results

We used the `save_as_excel_csv` option in Crosslight to export IV curves and plotted them together with Python; code is available in Appendix D. Figures 5 and 6 show the current and voltage curves for the InGaN and CIGS junctions respectively, and Table 1 tabulates the open circuit voltage, the short circuit current, the maximum power output, and the fill factor for each of the junctions. As expected, open circuit current and power output falls as depth increases, but while the small bandgap CIGS junction drops to a fraction of its output power at just 10 meters, the large bandgap InGaN junction retains more than half of its maximum power at 10 meters and nearly a third at 50 meters.

5 Remarks

Our results prove the effectiveness of the proposed dual-junction design. At the surface, the small bandgap CIGS junction provides more than half of the modules output power, increasing it from 34.4 $\frac{\text{W}}{\text{m}^2}$ to 80.5 $\frac{\text{W}}{\text{m}^2}$. A potential AUV or ROUV would benefit from the doubled power, reducing charging times and enabling higher-power workloads. Meanwhile, at 10m and 50m, the CIGS junction falls to a tenth and a hundredth of its power at the surface, but the power from the InGaN junction allows the module to retain a large fraction of its total power. This allows a potential AUV or ROUV to stay at these depths for longer periods without resurfacing, potentially enabling continous operation for low power workloads.

However, there are many improvements we would implement given more time. Firstly, we would like to further optimize the device structure: the power out is still a small fraction of the total solar power (1000 $\frac{\text{W}}{\text{m}^2}$ at the surface, 250 $\frac{\text{W}}{\text{m}^2}$ at 10m, and 70 $\frac{\text{W}}{\text{m}^2}$ at 50m), even accounting for the SQ losses. Secondly, our structure is obviously not manufacturable in real life due to the vacuum separating the two junctions. Using these results as a basis, we would design a more realistic device structure incorporating the two junctions. Lastly, we would address other features of a solar cell used in marine environments, such as optimizing an antifouling coating and reducing reflective losses.

This project was also not without difficulties. Our biggest obstacle was getting Crosslight to recognize the contacts within the device, since the `layer` file is unable to specify contacts that arent at the top or bottom of the device structure. After much trial and error, we found that the `add_boundary` command in the `geo` file was the one that defined contacts. Even after that realization, we still werent sure that the extra contacts were actually groundedwe suspected that they werent due to the poor performance of the InGaN layer. We ended up isolating the InGaN junction and found that it performed poorly on its own, meaning that the additional contacts werent the problem. It was only after adding the intrinsic layer did we finally get a working device.

6 Conclusion

We demonstrated the effectiveness of a dual junction InGaN and CIGS solar cell in marine applications. We first calculated the solar spectra at various depths and computed the optimal bandgap pair based on the Shockley-Queisser limit. Next, we selected two materials with bandgap ranges within those optimal values and designed and optimized a device structure around them. Finally, we simulated the devices in Crosslight and showed that the resulting device characteristics are advantageous for AUVs and ROUVs.

7 Contributions

Carla proposed the initial marine theme, suggesting some of the unique needs of underwater solar cells that we could address. Additionally, she performed the optimization of the InGaN junction, including introducing the intrinsic layer that substantially improved its efficiency. Larry suggested the dual junction idea, performed the calculations of the optimal bandgaps, and implemented the initial device structure.

8 References

[1] Karen S. Baker and Raymond C. Smith. Optical properties of the clearest natural waters (200800 nm).

Applied Optics, Vol. 20, Issue 2, pp. 177-184, 20:177–184, 1 1981.

[2] Zifei Chen, Arun Ashokan, Salvy P. Russo, and Paul Mulvaney. Temperature dependence of the cds bandgap in the extreme confinement regime. *Nano Letters*, 23:9287–9294, 10 2023.

[3] Bita Farhadi and Mosayeb Naseri. An optimized efficient dual junction ingan/cigs solar cell: A numerical simulation. *Superlattices and Microstructures*, 96:104–110, 8 2016.

[4] Fivos Perakis, Luigi De Marco, Andrey Shalit, Fujie Tang, Zachary R. Kann, Thomas D. Khne, Renato Torre, Mischa Bonn, and Yuki Nagata. Vibrational spectroscopy and dynamics of water. *Chemical Reviews*, 116:7590–7607, 7 2016.

[5] Jason A. Rhr, Jason Lipton, Jaemin Kong, Stephen A. Maclean, and Andr D. Taylor. Efficiency limits of underwater solar cells. *Joule*, 4:840–849, 4 2020.

[6] Jason A. Rhr, B. Edward Sartor, Jason Lipton, and Andr D. Taylor. A dive into underwater solar cells. *Nature Photonics 2023 17:9*, 17:747–754, 8 2023.

[7] Avilash Sahoo, Santosha K. Dwivedy, and P. S. Robi. Advancements in the field of autonomous underwater vehicle. *Ocean Engineering*, 181:145–160, 6 2019.

[8] William Shockley and Hans J. Queisser. Detailed balance limit of efficiency of pn junction solar cells. *Journal of Applied Physics*, 32:510–519, 3 1961.

Appendix A Figures

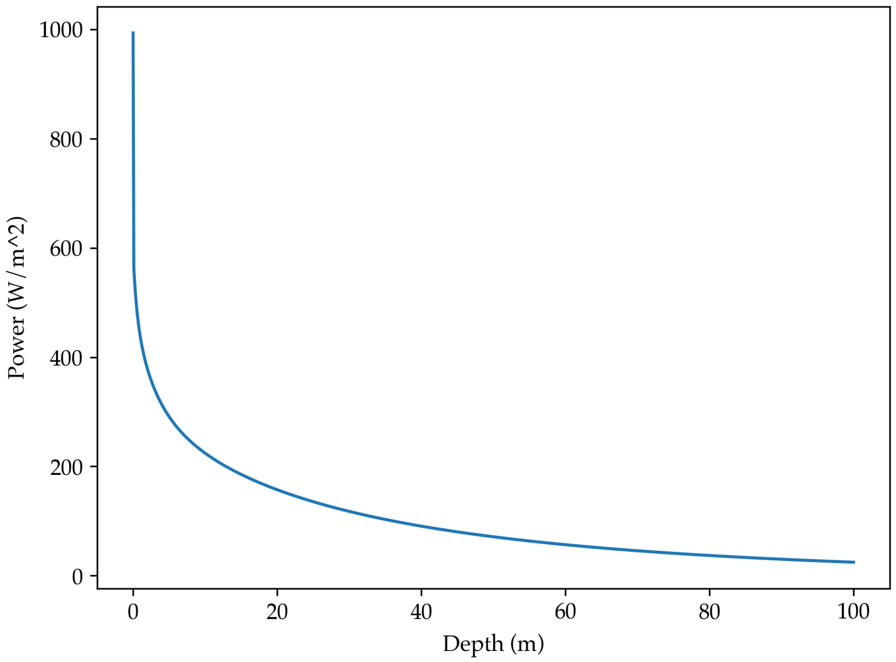

Figure 1. Attenuated power by depth. The surface spectrum is AM1.5, which is normalized to 1000 $\frac{W}{m^2}$. Within a few meters, the longer wavelengths are rapidly absorbed, causing an immediate drop in power. The rest of the graph follows a normal exponential decay.

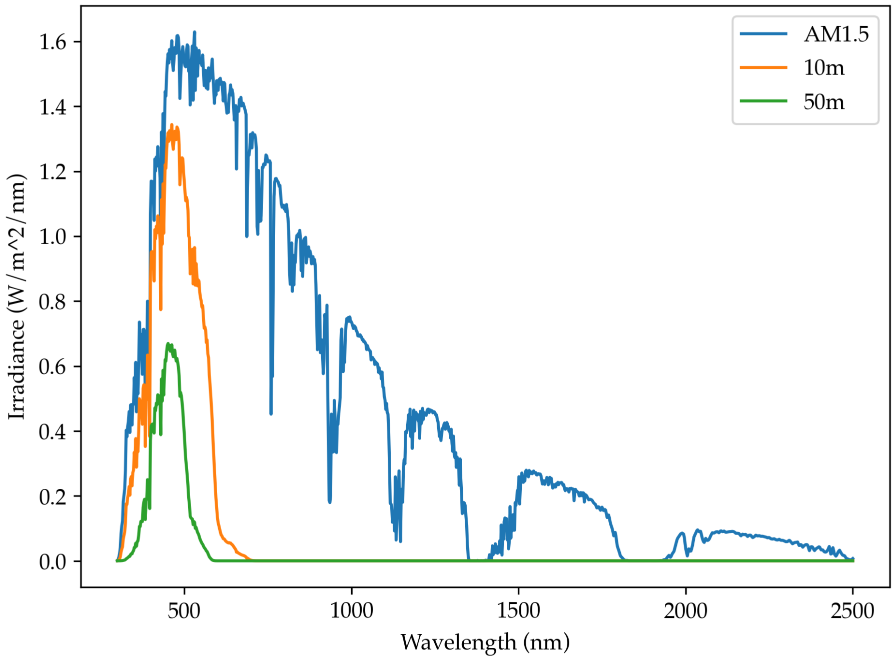

Figure 2. Irradiance by wavelength at the surface, 10 meters, and 50 meters. Wavelengths beyond 700 nanometers are rapidly absorbed, transmitting only lower wavelength light.

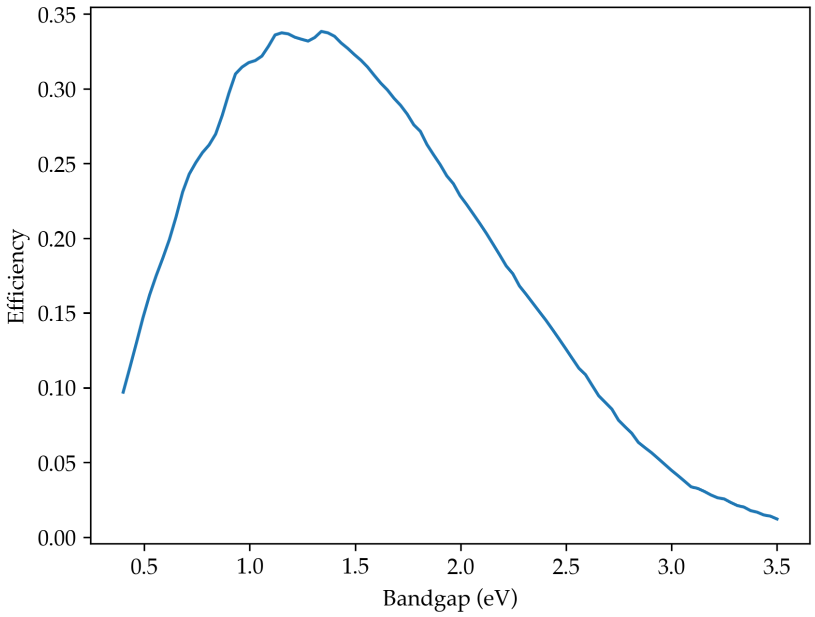

Figure 3. Shockley-Queisser limit for a single bandgap device. The peak efficiencies and the overall shape of the graph is in agreement with literature.

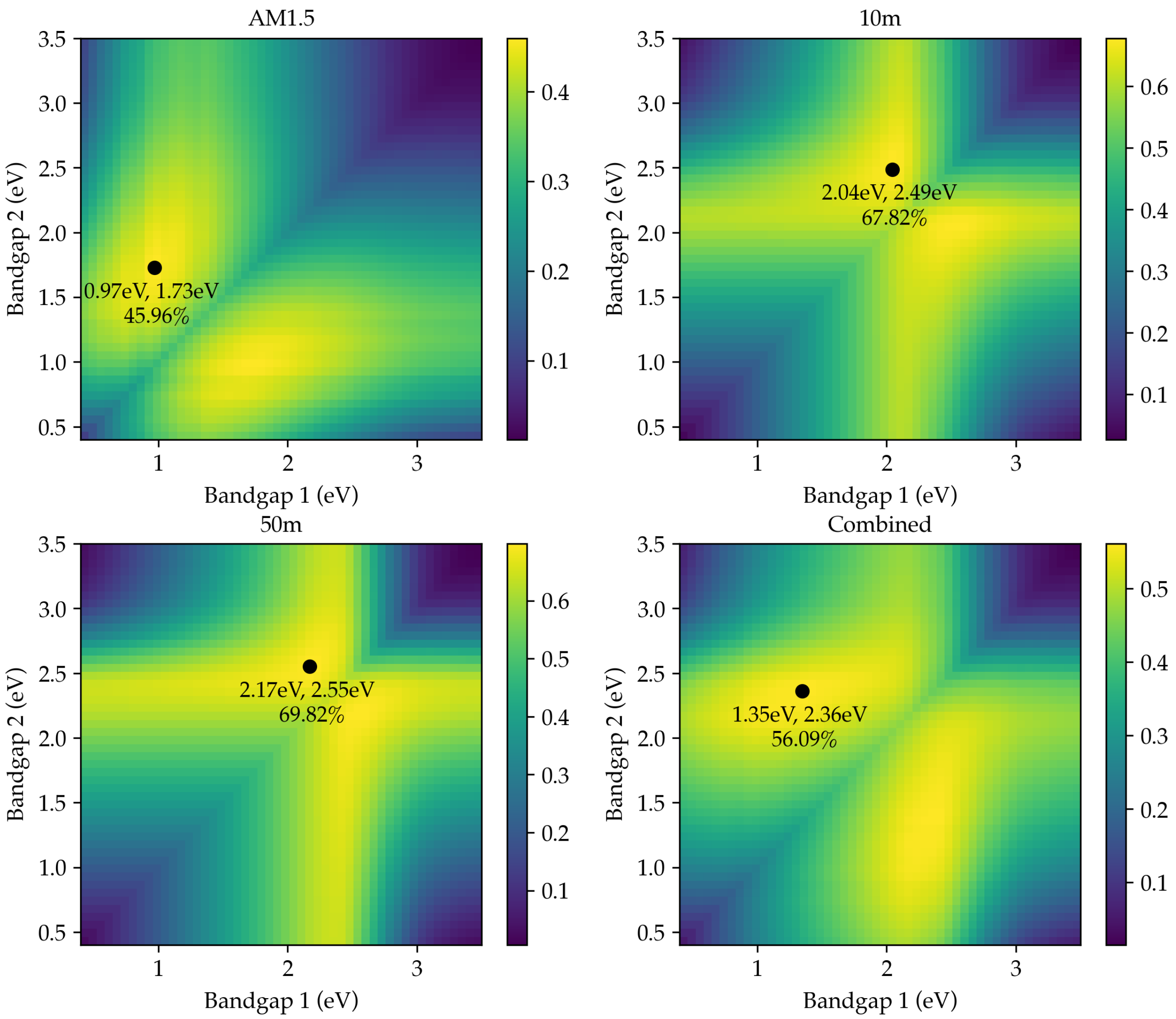

Figure 4. Shockley-Queisser limit for a dual bandgap device. Bandgaps from 0.5 to 3.5 eVs for each bandgap are simulated and plotted for three spectra: AM1.5, 10 meters, and 50 meters. The combined graph shows the arithmetic mean of these three graphs and the points of peak efficiency are annotated.

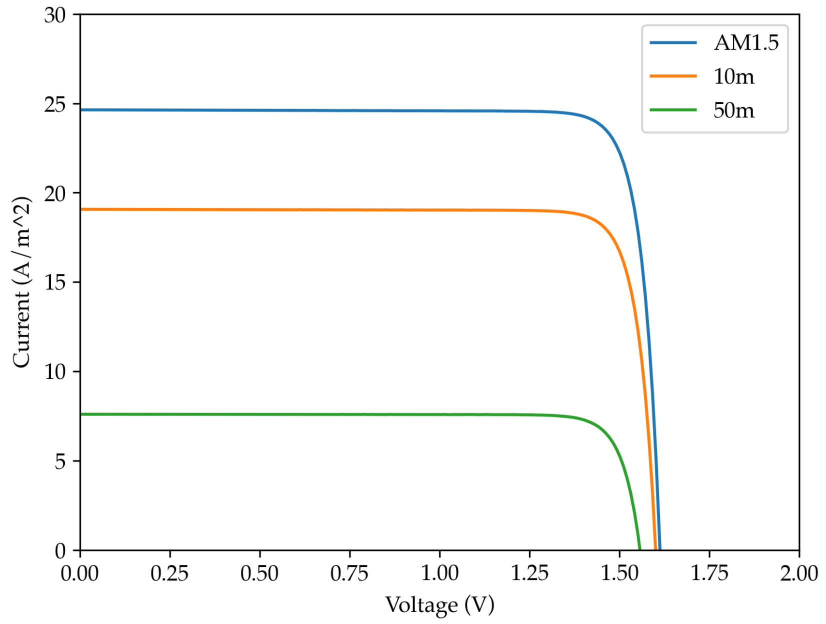

Figure 5. Current and voltage curve of the InGaN junction under the AM1.5, 10 meter, and 50 meter spectra. The larger bandgap of the InGaN junction allows it to receive more of the high energy light that reaches deeper depths.

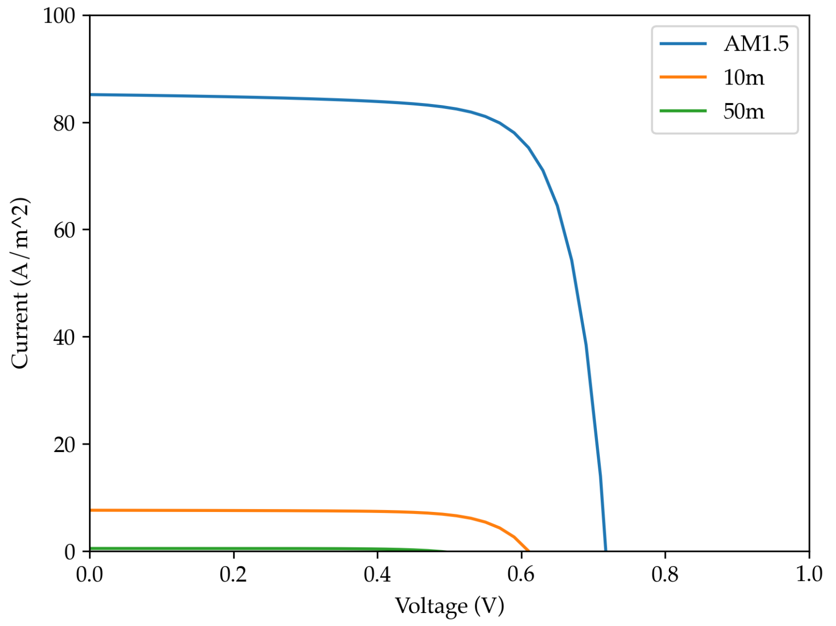

Figure 6. Current and voltage curve of the CIGS junction under the AM1.5, 10 meter, and 50 meter spectra. The smaller bandgap of the CIGS junction allows it to absorb more light at the surface, at the cost of losing power output rapidly at deeper depths.

Appendix B Python Preprocessing Code

```
# %%
import pvlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
plt.style.use('latex.mplstyle')

# %%
# https://opg.optica.org/ao/fulltext.cfm?uri=ao-20-2-177
# nm
wavelengths = [200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320,
    330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480,
     490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630,
    640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790,
     800]

# m^-1
diffuse_attenuation_coeffs = [3.14, 2.05, 1.36, 0.968, 0.754, 0.588, 0.481, 0.394,
    0.306, 0.23, 0.154, 0.116, 0.0944, 0.0765, 0.0637, 0.053, 0.0439, 0.0353,
    0.0267, 0.0233, 0.0209, 0.0196, 0.0184, 0.0172, 0.017, 0.0168, 0.0176, 0.0175,
    0.0194, 0.0212, 0.0271, 0.037, 0.0489, 0.0519, 0.0568, 0.0648, 0.0717, 0.0807,
    0.109, 0.158, 0.245, 0.29, 0.31, 0.32, 0.33, 0.35, 0.4, 0.43, 0.45, 0.5, 0.65,
    0.834, 1.17, 1.8, 2.38, 2.47, 2.55, 2.51, 2.36, 2.16, 2.07]

# insert interpolated values

interp = interp1d(wavelengths, diffuse_attenuation_coeffs, kind='linear',
    fill_value=np.inf, bounds_error=False)
wavelengths = np.linspace(300, 2500, 1001)
diffuse_attenuation_coeffs = interp(wavelengths)

# (W/m^2)/nm
am15 = np.array(pvlib.spectrum.get_am15g(wavelengths))


def calculate_attenuated_am15(distance):
    if distance <= 0:
        return am15
    return am15 * np.exp(-np.array(diffuse_attenuation_coeffs) * distance)

# %%
# plot at 0m, 10m, 50m

plt.plot(wavelengths, am15, label='AM1.5')
plt.plot(wavelengths, calculate_attenuated_am15(10), label='10m')
plt.plot(wavelengths, calculate_attenuated_am15(50), label='50m')
plt.legend()
plt.xlabel('Wavelength␣(nm)')
plt.ylabel('Irradiance␣(W/m^2/nm)')
plt.tight_layout()

plt.savefig('attenuated_am15.png', dpi=300)
```

```
# %%
depths = np.linspace(0, 100, 1001)
am15s = np.array([calculate_attenuated_am15(d) for d in depths])
powers = np.trapz(am15s, wavelengths, axis=1)

plt.figure()
plt.plot(depths, powers)
plt.xlabel('Depth (m)')
plt.ylabel('Power (W/m^2)')
plt.tight_layout()
plt.savefig('attenuated_am15_power.png', dpi=300)

# %%
nm_eV_coeff = 1239.8
J_to_eV = 6.242e18

eVs = nm_eV_coeff / np.array(wavelengths)


def calc_eff(spectrum, bandgaps):
    # the relevant bandgap is the largest one that's smaller than the photon energy
    bandgaps.append(0)
    bandgaps.sort()
    relevant_bandgap_i = np.zeros_like(eVs, dtype=int)
    for i, bandgap in enumerate(bandgaps):
        relevant_bandgap_i[eVs > bandgap] = i
    relevant_bandgaps = np.array(bandgaps)[relevant_bandgap_i]

    # convert to photons / (s * eV * m^2)
    photons = spectrum * J_to_eV / eVs * wavelengths / eVs

    # Radiative recombination calculation
    # 2 * pi / (speed of light ^2 * planck constant ^3) in 1/(m^2 * s * eV^3)
    coeff = 9.883e26
    # kT at 300K in eV
    kT = 0.02585

    input_power = -np.trapz(photons * eVs / J_to_eV, eVs)

    output_power = 0

    for i, bandgap in enumerate(bandgaps):
        if i == 0:
            continue
        energies = np.linspace(bandgap, 10, 1000)
        radiative_recombination_nobias = coeff * np.trapz(energies ** 2 / (np.exp(
            energies / kT) - 1), energies)

        this_photons = photons[relevant_bandgap_i == i]
        this_eVs = eVs[relevant_bandgap_i == i]
        total_photons = -np.trapz(this_photons, this_eVs)

        applied_voltages = np.linspace(0, 5, 1000)
        # in electron charge
        currents = total_photons - radiative_recombination_nobias * (np.exp(
            applied_voltages / kT) - 1)
        power_eV = applied_voltages * currents
        max_power = np.max(power_eV) / J_to_eV
        output_power += max_power

    return output_power / input_power



# %%
# Calculate the Shockley-Queisser limit

test_bandgaps = np.linspace(0.4, 3.5, 100)
efficiencies = [calc_eff(am15, [bandgap]) for bandgap in test_bandgaps]
plt.plot(test_bandgaps, efficiencies)
plt.xlabel('Bandgap (eV)')
plt.ylabel('Efficiency')
plt.savefig('sq_limit.png', dpi=300)

# %%
def graph_efficiencies(ax, spectrums):
    bandgaps1 = np.linspace(0.4, 3.5, 50)
    bandgaps2 = np.linspace(0.4, 3.5, 50)

    efficiencies = np.zeros((len(bandgaps1), len(bandgaps2)))
    for i, bandgap1 in enumerate(bandgaps1):
        for j, bandgap2 in enumerate(bandgaps2):
            for spectrum in spectrums:
                efficiencies[i, j] += calc_eff(spectrum, [bandgap1, bandgap2])

    efficiencies /= len(spectrums)
    ax.imshow(efficiencies, extent=(bandgaps1[0], bandgaps1[-1], bandgaps2[0],
        bandgaps2[-1]), origin='lower')
    ax.set_xlabel('Bandgap 1 (eV)')
    ax.set_ylabel('Bandgap 2 (eV)')


    # add colorbar
    cbar = plt.colorbar(ax.imshow(efficiencies, extent=(bandgaps1[0], bandgaps1
        [-1], bandgaps2[0], bandgaps2[-1]), origin='lower'))

    # add point at the maximum efficiency
    max_efficiency = np.max(efficiencies)
    max_efficiency_i = np.unravel_index(np.argmax(efficiencies, axis=None),
        efficiencies.shape)
    ax.scatter(bandgaps1[max_efficiency_i[0]], bandgaps2[max_efficiency_i[1]],
        color='black')

    ax.text(bandgaps1[max_efficiency_i[0]], bandgaps2[max_efficiency_i[1]], f'\n\n\
        n{bandgaps1[max_efficiency_i[0]]:.2f}eV, {bandgaps2[max_efficiency_i[1]]:.2
        f}eV \n {max_efficiency:.2%}', color='black', fontsize=10.5, ha='center',
        va='center')


# %%
plt.figure()
```

```
f, axes = plt.subplots(2, 2, figsize=(8.3,7))

graph_efficiencies(axes[0,0], [am15])
axes[0,0].set_title('AM1.5')
axes[0,0].set_aspect('equal')

graph_efficiencies(axes[0,1], [calculate_attenuated_am15(10)])
axes[0,1].set_title('10m')
axes[0,1].set_aspect('equal')

graph_efficiencies(axes[1,0], [calculate_attenuated_am15(50)])
axes[1,0].set_title('50m')
axes[1,0].set_aspect('equal')

graph_efficiencies(axes[1,1], [am15, calculate_attenuated_am15(10),
    calculate_attenuated_am15(50)])
axes[1,1].set_title('Combined')
axes[1,1].set_aspect('equal')


plt.tight_layout()


# save the plot
plt.savefig('efficiencies.png', dpi=300)

# %%
# save spectra for crosslight

def save_spectrum(spectrum, filename):
    data = pd.DataFrame({'Wavelength (um)': wavelengths / 1000, 'Irradiance (W/m^2/
        um)': spectrum * 1000})
    data.to_csv(filename, index=False, sep='\t', header=False)

save_spectrum(am15, 'solar.am15')
save_spectrum(calculate_attenuated_am15(10), 'solar.am15_10m')
save_spectrum(calculate_attenuated_am15(50), 'solar.am15_50m')
```

Appendix C Crosslight Code

solar.layer

```
$file:solar.layer
set_polarization screening = 0.9
begin_layer
column column_num=1 w=5. mesh_num=2 r=1.
$
bottom_contact column_num=1 from=0. to=5. &&
contact_num=1 contact_type=ohmic
$
layer_mater macro_name=cigs column_num=1 var_symbol1=x var1=0.61
layer d=3.2 n=100 n_doping1=1.e20 r=0.8
$
layer_mater macro_name=cigs column_num=1 var_symbol1=x var1=0.61
layer d=0.1 n=100 p_doping1=1.e25 r=1.2
$
layer_mater macro_name=vacuum column_num=1
layer d=0.1 n=10 r=1.
$
layer_mater macro_name=ingan column_num=1 var_symbol1=x var1=0.28
layer d=3.0 n=35 n_doping1=6.e24 r=0.9
$
layer_mater macro_name=ingan column_num=1 var_symbol1=x var1=0.28
layer d=0.06 n=20 r=-1.1
$
layer_mater macro_name=ingan column_num=1 var_symbol1=x var1=0.28
layer d=0.3 n=60 p_doping1=8.e25 r=1.1
$

top_contact column_num=1 from=0. to=5. &&
contact_num=2 contact_type=ohmic
end_layer
```

solar.mater

This additional code must be inserted into the generated material file for the extra contacts:

```
contact num=3 type=ohmic
contact num=4 type=ohmic
```

solar.geo

This additional code must be inserted into the generated geometry file for the extra contacts:

```
add_boundary polygon_name=p0002 edge_points=[ b0003 a0003 ] &&
limits=[  0.000000000000E+000  0.500000000000E+001 ] &&
boundary_num=3

add_boundary polygon_name=p0004 edge_points=[  a0004 b0004 ] &&
limits=[  0.000000000000E+000  0.500000000000E+001 ] &&
boundary_num=4
```

solar.sol

```
$file:solar.sol
$***********
begin

$ Load InGaN material parameters
use_macrofile macro1=ingan.mac

load_mesh mesh_inf=solar.msh
include file=solar.mater
include file=solar.doping

output sol_outf=solar.out
more_output light_reflection = yes elec_mobility = yes  &&
hole_mobility = yes space_charge=yes

newton_par damping_step=5. var_tol=1.e-2 res_tol=1.e-2 &&
max_iter=200 opt_iter=30 stop_iter=50 print_flag=3

$
$ Solve for equilibrium condition.
$

equilibrium

newton_par damping_step=1 var_tol=1.e-1 res_tol=1.e-1 &&
max_iter=200 opt_iter=15 stop_iter=50 print_flag=3 &&
change_variable=no

$
$ Turn up the light
$

scan var=light value_to=1. print_step=1. &&
init_step=0.01 min_step=1.e-5 max_step=0.1

$
$ Apply a forward bias to offset the photo current
$

scan var=voltage_2 value_to=2.5 print_step=5. &&
init_step=0.0001 min_step=1.e-5 max_step=0.01

scan var=voltage_1 value_to=-3 &&
init_step=0.01 min_step=1.e-5 max_step=0.02

$
$ Light spectrum
$

$ Change spectrum_file to
light_power spectrum_file = solar.am15_50m light_dir = top  &&
profile = [0,5,0.001,0.001]  std_wave = no &&
fresnel_reflections = no

front_index real_index = 1.0
back_index real_index = 1.7
```

```
end
```

solar.plt

```
$file:solar.plt
$ **************
begin_pstprc
plot_data plot_device=pdf

$ I-V curve for cigs
get_data  main_input=solar.sol sol_inf=solar.out &&
scan_data=(3 4)

plot_scan scan_var=voltage_1 variable=current_3 &&
scale_vertical=1 scale_horizontal=-1 &&
yrange=[0 0.001] save_as_excel_csv=cigs_iv.am15_50m &&
csv_col_label1=voltage csv_col_label2=current

$ I-V curve for ingan
get_data  main_input=solar.sol sol_inf=solar.out &&
scan_data=(3 4) xy_data=[  4  4]

plot_scan scan_var=voltage_2 variable=current_2 &&
scale_vertical=1 scale_horizontal=1 &&
yrange=[0 0.001] save_as_excel_csv=ingan_iv.am15_50m  &&
csv_col_label1=voltage csv_col_label2=current

end_pstprc
```

Appendix D Python Postprocessing Code

```
# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
plt.style.use('latex.mplstyle')

# %%
# A/(m * 5um) -> A/m^2
current_factor = 200000

cigs_iv_am15 = pd.read_csv('cigs_iv.am15.csv', header=None, skiprows=[0], names=['V
    ', 'I'])
cigs_iv_am15_10m = pd.read_csv('cigs_iv.am15_10m.csv', header=None, skiprows=[0],
    names=['V', 'I'])
cigs_iv_am15_50m = pd.read_csv('cigs_iv.am15_50m.csv', header=None, skiprows=[0],
    names=['V', 'I'])

ingan_iv_am15 = pd.read_csv('ingan_iv.am15.csv', header=None, skiprows=[0], names=[
    'V', 'I'])
ingan_iv_am15_10m = pd.read_csv('ingan_iv.am15_10m.csv', header=None, skiprows=[0],
     names=['V', 'I'])
ingan_iv_am15_50m = pd.read_csv('ingan_iv.am15_50m.csv', header=None, skiprows=[0],
     names=['V', 'I'])

# fix current values at start
ingan_iv_am15["I"][:3] = ingan_iv_am15["I"][3]
ingan_iv_am15_10m["I"][:3] = ingan_iv_am15_10m["I"][3]
ingan_iv_am15_50m["I"][:3] = ingan_iv_am15_50m["I"][3]

# %%

plt.plot(cigs_iv_am15['V'], cigs_iv_am15['I'] * current_factor, label='AM1.5')
plt.plot(cigs_iv_am15_10m['V'], cigs_iv_am15_10m['I'] * current_factor, label='10m'
    )
plt.plot(cigs_iv_am15_50m['V'], cigs_iv_am15_50m['I'] * current_factor, label='50m'
    )

plt.xlabel('Voltage␣(V)')
plt.ylabel('Current␣(A/m^2)')

plt.xlim(0, 1.0)
plt.ylim(0, 100)

plt.legend()

plt.savefig('cigs_iv.png', dpi=300)


# %%
plt.plot(ingan_iv_am15['V'], ingan_iv_am15['I'] * current_factor, label='AM1.5')
plt.plot(ingan_iv_am15_10m['V'], ingan_iv_am15_10m['I'] * current_factor, label='10
    m')
plt.plot(ingan_iv_am15_50m['V'], ingan_iv_am15_50m['I'] * current_factor, label='50
    m')

plt.xlabel('Voltage␣(V)')
plt.ylabel('Current␣(A/m^2)')
plt.xlim(0, 2.0)
plt.ylim(0, 30)

plt.legend()

plt.savefig('ingan_iv.png', dpi=300)

# %%
def calculate_specs(df):
    V = df['V']
    I = df['I'] * current_factor

    first_quadrant = (V > 0) & (I > 0)
    V = V[first_quadrant]
    I = I[first_quadrant]
    V_oc = V.iloc[-1]
    I_sc = I.iloc[0]

    P = V * I

    P_max = P.max()
    fill_factor = P_max / (V_oc * I_sc)
    efficiency = P_max / 1000

    return V_oc, I_sc, P_max, fill_factor, efficiency

# create table

rows = [
    ('AM1.5', cigs_iv_am15, ingan_iv_am15),
    ('10m', cigs_iv_am15_10m, ingan_iv_am15_10m),
    ('50m', cigs_iv_am15_50m, ingan_iv_am15_50m),
]

table = []
index = []
column_names = pd.DataFrame([
    ["CIGS", r"$V_{oc}$␣(V)"],
    ["CIGS", r"$I_{sc}$␣$(\frac{A}{m^2})$"],
    ["CIGS", r"$P_{max}$␣$(\frac{W}{m^2})$"],
    ["CIGS", "FF"],
    ["InGaN", r"$V_{oc}$␣(V)"],
    ["InGaN", r"$I_{sc}$␣$(\frac{A}{m^2})$"],
    ["InGaN", r"$P_{max}$␣$(\frac{W}{m^2})$"],
    ["InGaN", "FF"],
    ["Total", r"$P_{max}$␣$(\frac{W}{m^2})$"],
], columns=["ID", ""])


for name, cigs_iv, ingan_iv in rows:
    cigs_V_oc, cigs_I_sc, cigs_P_max, cigs_fill_factor, cigs_efficiency =
        calculate_specs(cigs_iv)
    ingan_V_oc, ingan_I_sc, ingan_P_max, ingan_fill_factor, ingan_efficiency =
        calculate_specs(ingan_iv)

    total_power = cigs_P_max + ingan_P_max

    index.append(name)
    table.append([
        cigs_V_oc, cigs_I_sc, cigs_P_max, cigs_fill_factor,
        ingan_V_oc, ingan_I_sc, ingan_P_max, ingan_fill_factor,
        total_power
    ])

columns = pd.MultiIndex.from_frame(column_names)

table = pd.DataFrame(table, index=index, columns=columns)
table.to_latex('table.tex', escape=False, column_format='lccccccccc',
    formatters=[
        "{:.1f}".format,
        "{:.1f}".format,
        "{:.1f}".format,
        lambda x: "{:.1%}".format(x).replace("%", "\\%"),
        "{:.1f}".format,
        "{:.1f}".format,
        "{:.1f}".format,
        lambda x: "{:.1%}".format(x).replace("%", "\\%"),
        "{:.1f}".format,
    ]
)
```