extended_example_no1

November 18, 2018

1 A more advanced YaSoFo example

(Based on YaSoFo v1.2.6)

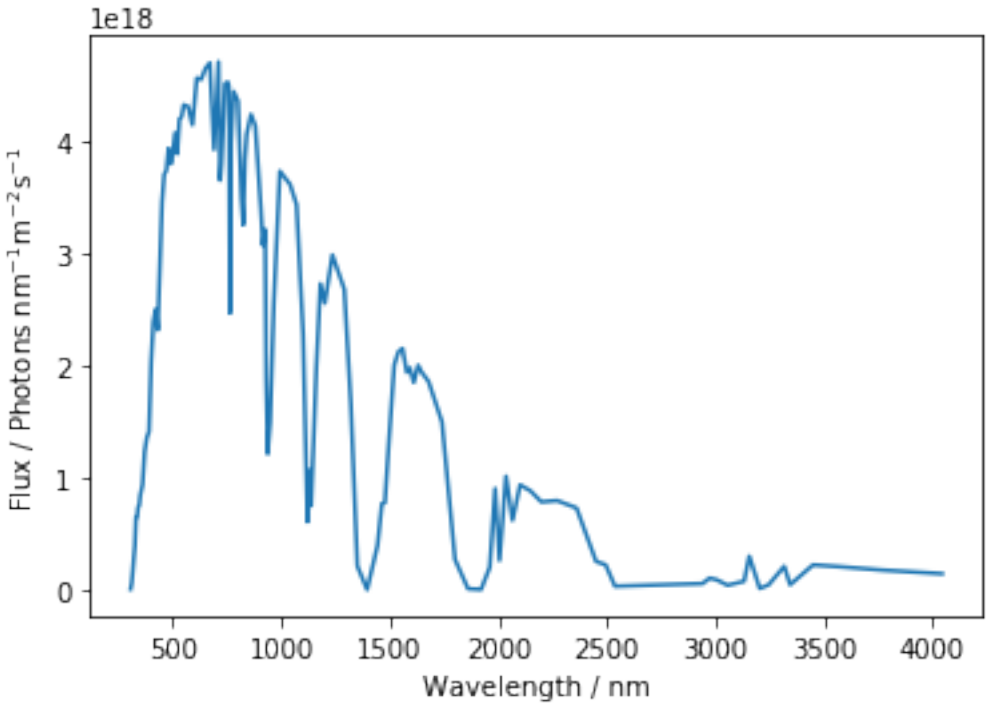
In this example, you will learn how to do a series of calculations with YaSoFo, varying one parameter. The system will be a double junction cell under AM1.5G illumination, with a very good catalyst and intended for water splitting. The parameter, that we change, is the ohmic resistivity. At the end, we will plot the results. Prerequisites: You have installed SciPy (Python 3.x version) on your computer and obtained YaSoFo from https://bitbucket.org/YaSoFo/yasofo . You have started a Python 3 shell, such as ipython, in the directory where the files yasofo.py and am15g.dat are located.

```
In [2]: # remove the following line for plots in a separate window
        %matplotlib inline
        import yasofo as yo
```

```
Spectrum successfully imported. Length: 120 points.
Integrated power: 1000.000 W/m^2. Expected: 1000 W/m^2.
Standard for efficiency calculation assumes 1000 W/m^2!
Length: 120 points. Recommended: 200-500.
```

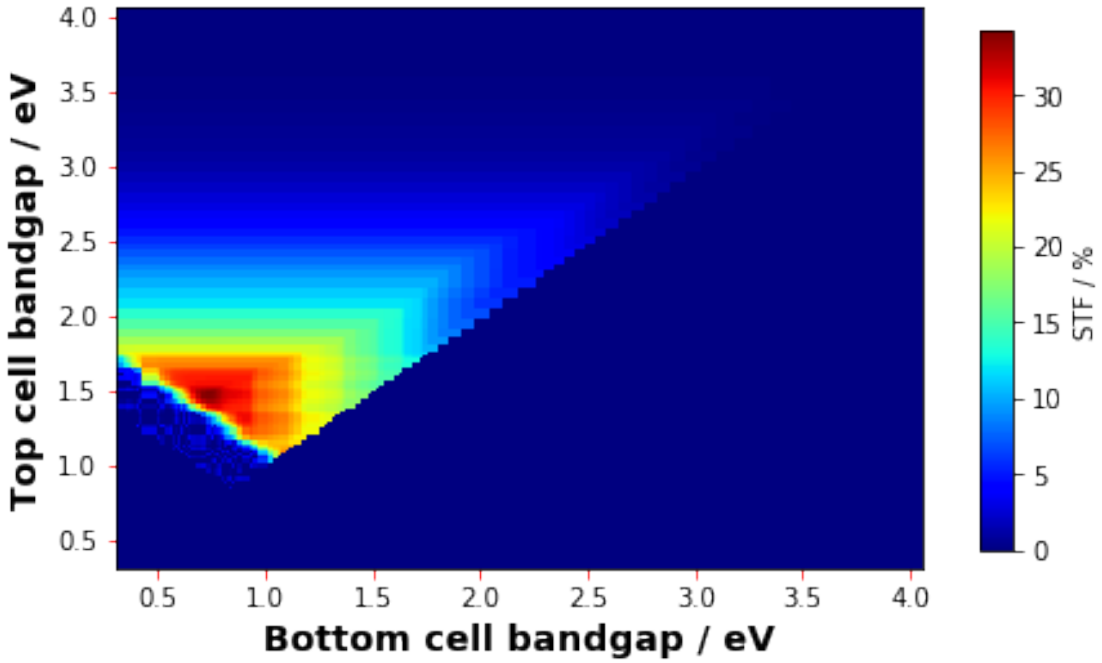
We can check if things are working by plotting the AM1.5G spectrum:

```
In [3]: yo.am15g.plot()
```



Okay, that works. So let us plot the possible bandgap combinations for a double junction cell with the resulting efficiencies for solar water splitting. By not specifying any parameters, the standard settings are used, i.e. AM1.5G, IrO2 as a catalyst, no ohmic drop, and the detailed-balance optimum for the photovoltage (as opposed to a constant loss per junction).

```
In [4]: double_junction = yo.best_gaps_double()
```



These are the standard plotting settings. If you want to control the settings for your plot, have a look at the matplotlib documentation. The computed data is stored in the variable 'double junction', we just created. Let's say we want to study the impact of an ohmic drop from 0 to 25 Ohm in steps of 5 Ohm. We first create an array with the corresponding values:

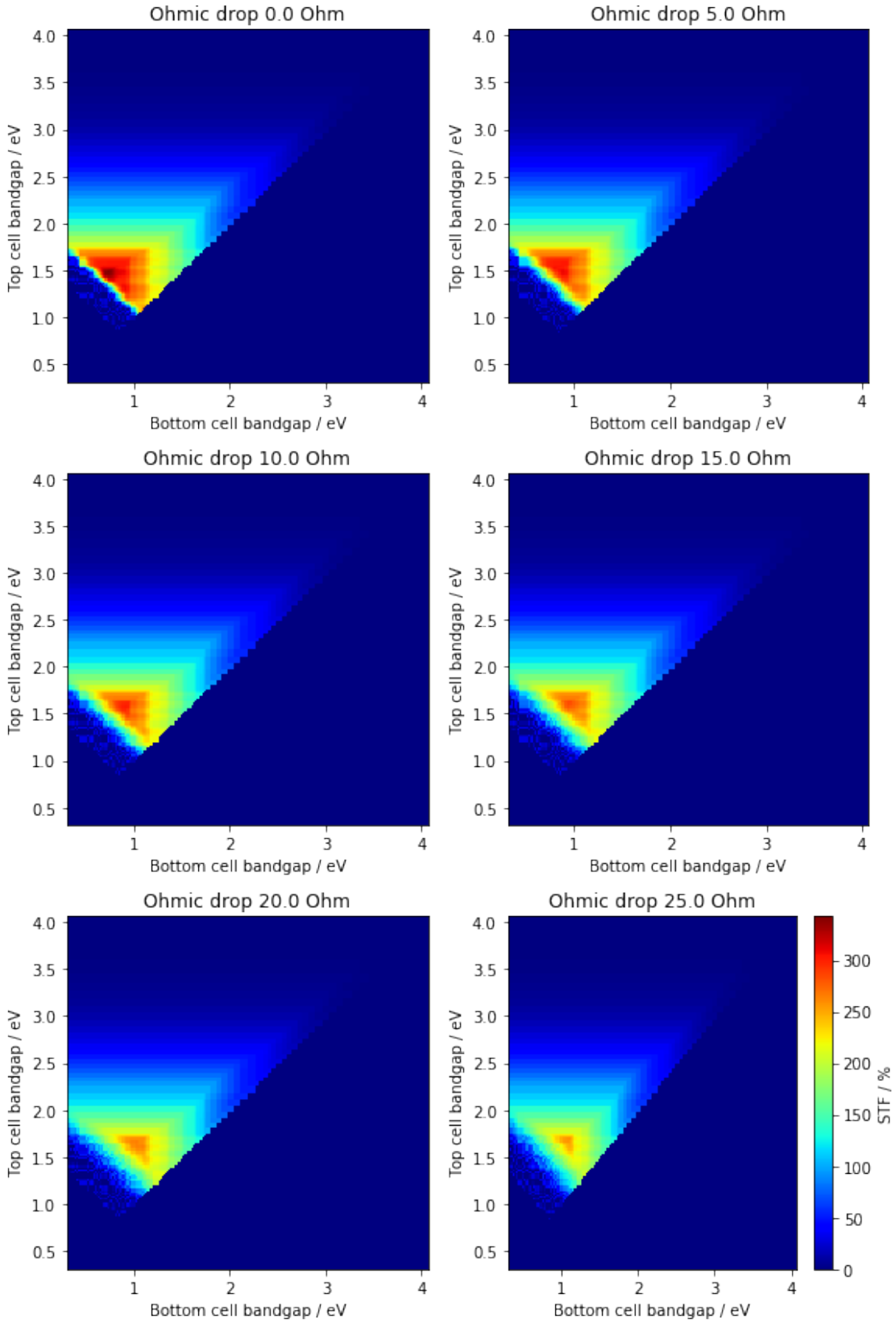
```
In [6]: import numpy as np
        ohmic_drop = np.arange(0, 25.1, 5)
        print(ohmic_drop)

[  0.   5.  10.  15.  20.  25.]
```

Above you see the values for the ohmic drop that will be tested. Now here the actual calculation, together with plotting:

```
In [7]: import math
        import matplotlib.pyplot as plt
        nrows = int(math.ceil(len(ohmic_drop) / 2.))
        efficiencies = []
        for drop in ohmic_drop:
            efficiencies.append(yo.best_gaps_double(cat_para=[yo.IrO2[0],
                yo.IrO2[1], drop], plotting=False))
        # now we have the data, lets plot it nicely
        fig, axes = plt.subplots(nrows, ncols=2, figsize=(8, 12))
        for ax, efficiency, drop in zip(axes.flat, efficiencies, ohmic_drop):
        # we're setting the colour scale to be the same for all,
```

```
# i.e. first dataset
    p = ax.pcolor(efficiency[0],efficiency[0],efficiency[1]*1000,
        cmap=plt.cm.jet, vmin=abs(efficiencies[0][1]*1000).min(),
        vmax=abs(efficiencies[0][1]*1000).max())
    ax.set_xlabel("Bottom cell bandgap / eV")
    ax.set_ylabel("Top cell bandgap / eV")
    ax.set_title("Ohmic drop %.1f Ohm" %drop)
plt.tight_layout()
cb = plt.colorbar(p)
cb.set_label("STF / %")
plt.show(p)
```


Ohmic drop 0.0 Ohm
Ohmic drop 5.0 Ohm
Ohmic drop 10.0 Ohm
Ohmic drop 15.0 Ohm
Ohmic drop 20.0 Ohm
Ohmic drop 25.0 Ohm
Top cell bandgap / eV
Bottom cell bandgap / eV
STF / %
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
4
300
250
200
150
100
50
0

So we see that the max. efficiency is constantly decreasing with solution resistance. Not so surprising. . . Such a looping can be done with any kind of parameter in the model.