

Lisa software – tutorial

Warning: this is a draft version, may contain mistakes. If you spot a mistake, please contact me.

Disclaimer:

The author does not make any warranty, expressed or implied, or assume any legal liability or responsibility for the accuracy, completeness or usefulness of any information contained in this work. Use the information and instructions contained in this work at your own risk.

Requirements

To run Lisa, one needs:

1. Python, version 2.7 (usually preinstalled on Linux distributions and OS, can be installed on Windows).
2. Numerical python – numpy. To install it on Ubuntu Linux, type:

```
sudo apt-get install python-numpy
```

3. Matplotlib library to plot the results. To install it on Ubuntu Linux, type:

```
sudo apt-get install python-matplotlib
```

4. Scipy library. To install on it Ubuntu Linux, type:

```
sudo apt-get install python-scipy
```

Example run – single simulation

1 Adjust the parameters in the parameters.py file

In particular, set the output flags to "True" for a single calculation.

Comment

If `jv_flag` is set to `true`, Lisa calculates the full JV characteristic and saves it to the file. This can take much longer than standard simulations. When this flag is set to `false`, Lisa uses algorithms to find maximum of the power density curve and Voc, and therefore needs much less voltage points to calculate the solar cell performance. The calculations are much faster (the calculation time depends linearly on the number of voltage points).

2 Run the solver

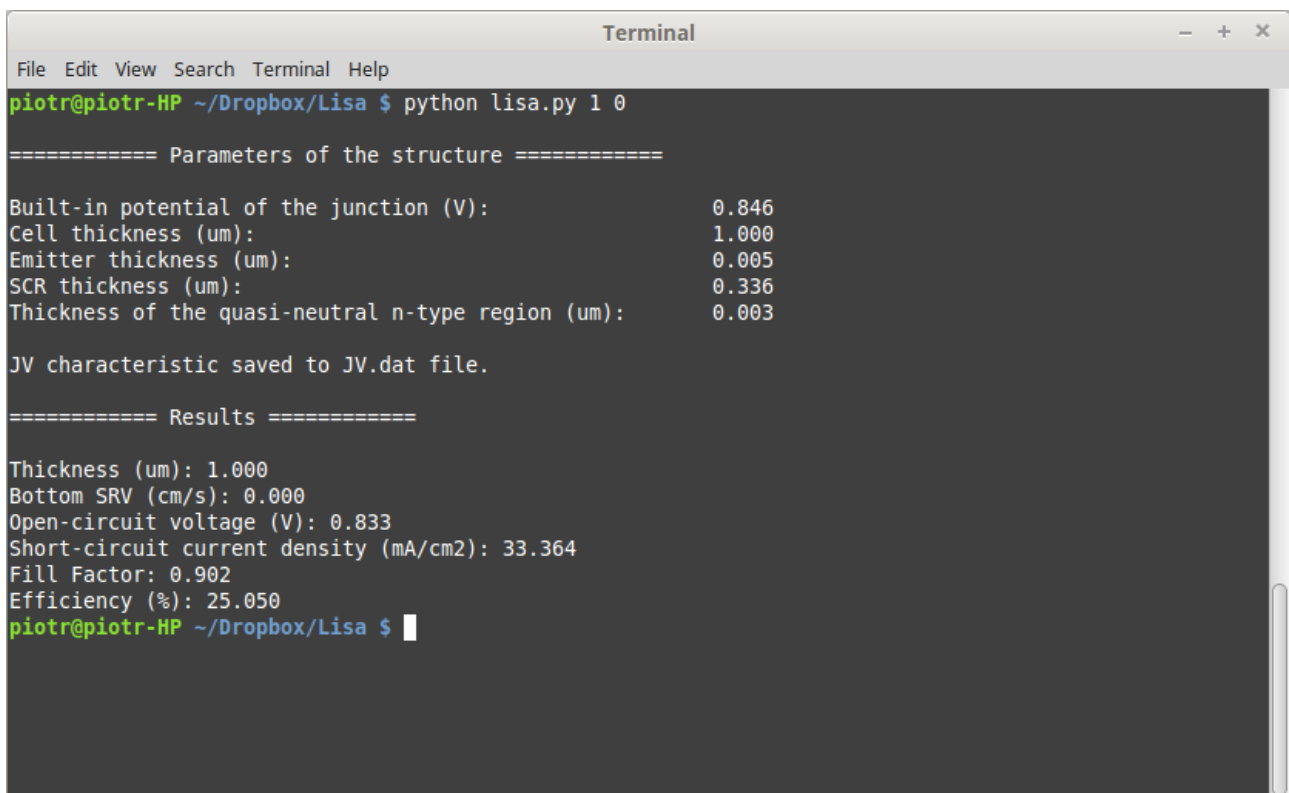
In the console type:

```
python lisa.py [thickness in um] [SRV in cm/s]
```

For example:

```
python lisa.py 1 0
```

3 The results are printed in the console



The screenshot shows a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "piotr@piotr-HP ~/Dropbox/Lisa \$". The command "python lisa.py 1 0" has been executed. The output is as follows:

```
===== Parameters of the structure =====  
  
Built-in potential of the junction (V):      0.846  
Cell thickness (um):                        1.000  
Emitter thickness (um):                     0.005  
SCR thickness (um):                         0.336  
Thickness of the quasi-neutral n-type region (um): 0.003  
  
JV characteristic saved to JV.dat file.  
  
===== Results =====  
  
Thickness (um): 1.000  
Bottom SRV (cm/s): 0.000  
Open-circuit voltage (V): 0.833  
Short-circuit current density (mA/cm2): 33.364  
Fill Factor: 0.902  
Efficiency (%): 25.050  
piotr@piotr-HP ~/Dropbox/Lisa $
```

4 Plot JV curve

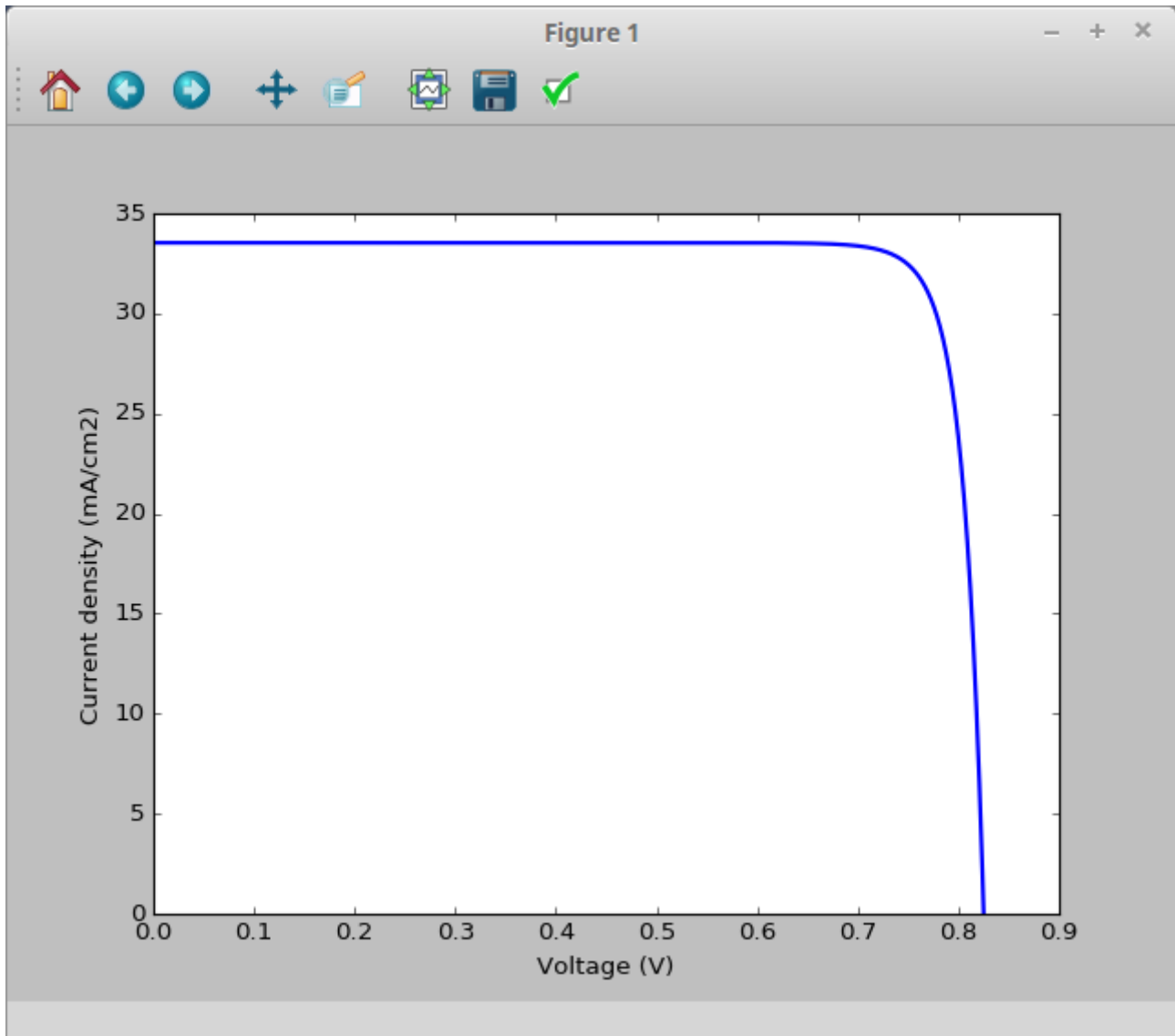
In the console type

```
python plots.py jv [filename]
```

The file should be in the **Results/** directory. For example:

```
python plots.py jv JV.dat
```

You should get plot similar to this one:



The axis of the plot should be automatically adjusted to the value of V_{oc} .

Example run – batch simulation

For example, we want to calculate efficiency limits in a given thickness range.

1 Adjust the parameters in the parameters.py file

In particular, set all output flags to "False" for a single calculation.

2 Prepare loop

Edit loop.py file. E.g., set thickness range. Then run it:

```
python loop.py
```

File loop.sh will appear, execute it:

```
sh loop.sh
```

The calculations will start, and the results will be printed on a screen. If you want to save the results to file, instead of the command above, you type:

```
sh loop.sh >> [filename]
```

The output will be saved to the given file. If later you want to plot the results with Lisa, the best is to save the file in the ./Results folder.

3 Plot the results

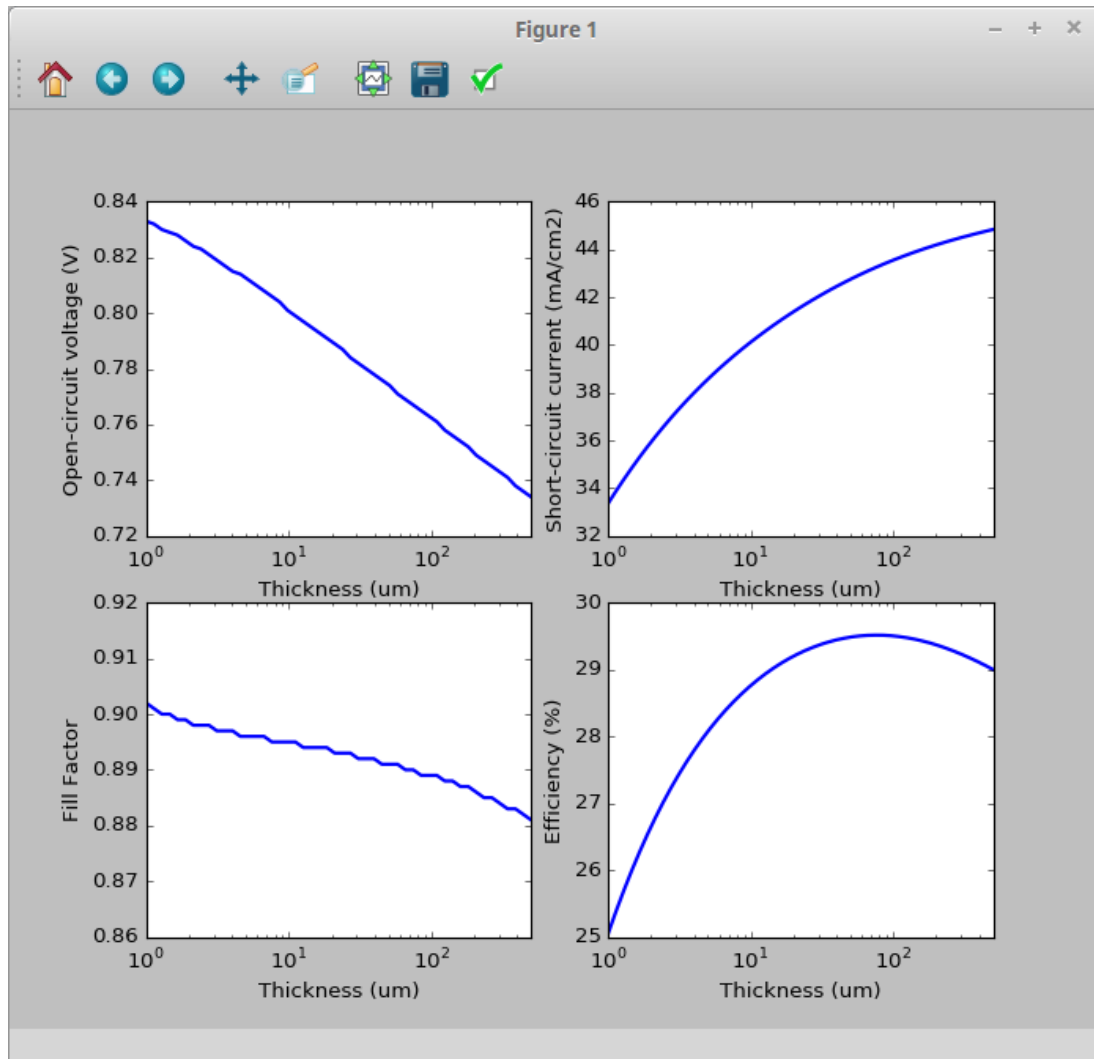
Now you can plot characteristics of the cells. Type:

```
python plots.py panel [filename]
```

For example:

```
python plots.py panel loop.dat
```

You should get plot similar to this one:



4 Plot the results – comparison

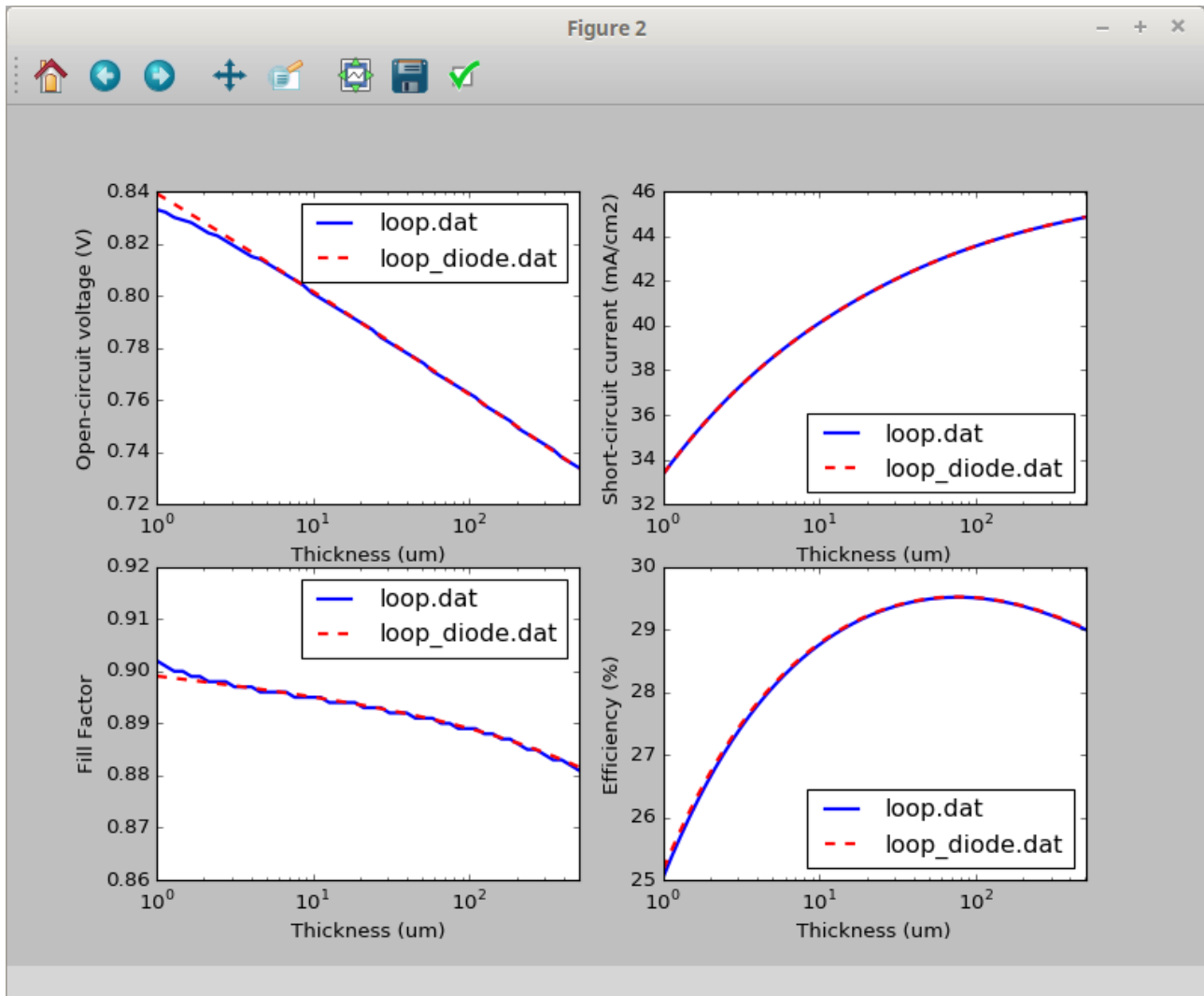
It is often useful to compare our results with a different data set. For example, we want to compare the results obtained using the Hovel model with the results obtained using the diode equation. To do so, type:

```
python plots.py panel_comparison [filename 1] [filename 2]
```

For example:

```
python plots.py panel_comparison loop.dat loop_diode.dat
```

You should get a plot similar to this one:



Example run – batch simulation (2) – contour plot

Contour plots can be very helpful when we want to plot a certain quantity, say efficiency, as a function of two parameters.

1 Prepare loop

This time, in the `loop.py` file, we are going to make a loop for two variables.

2 Plot the results

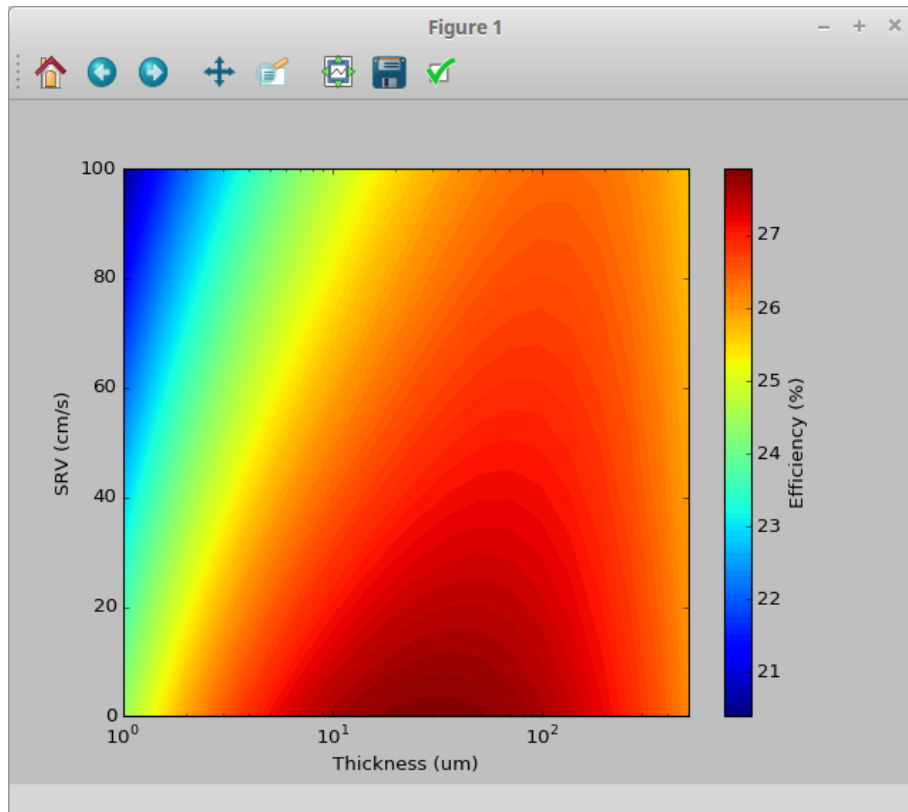
To plot the contour plot, type:

```
python plots.py contour [filename]
```

For example:

```
python plots.py contour loop_SRV.dat
```

You should get a plot similar to this one:



Lisa is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.