dopey

dopey is a set of python scripts that can load and visualize data from Prodigy. The latest version of **dopey** can be downloaded from https://github.com/matlea/dopey.
This document is updated 2024.12.03 by Mats Leandersson.

The files

The package contains the following files. They are constantly updated and more files are added when needed. There is a slight (*i.e.* high) probability that this document is not updated as often as the scripts are.

dopey_loader.py
dopey_plot.py
dopey_spin.py
dopey_methods.py
dopey_constants.py
dopey_dichroism.py
dopey_export2txt.py
changelog.txt

dopey.pdf
dopey_dictionaries.pdf

dopey.pdf and dopey_dictionaries.pdf contains descriptions of dopey and its methods, and of the data format it uses.

The scripts

All methods takes a **dopey** dict as an input. **dopey** dicts are loaded with the `load()` method or is a output from one of the dopey methods. All methods prints information to screen as default but this can be turned off by passing the argument `shup` (shut up) as False. Example:

```
data = dop.load("data/arpes.xy"

============================
Data info:
  File name:      data/arpes.xy
  Spectrum id:    1
  Lens mode:      WideAngleMode
  Scan mode:      SnapshotFAT
  Ep, Ek:         20.00 eV, 19.26 eV
  Type of data:   arpes
Axes:
  x:              400 points, Kinetic energy (eV), from 18.165626 to 20.354374
  y:              400 points, Angle (deg.), from -14.9625 to 14.9625
Intensity:
  intensity:      (400, 400)
============================
```

The methods prints info, warnings, and errors. Errors can not be turned off.

dopey_load.py

Contains the main method of the whole package: `load()`, plus two other methods that might or might not be useful.

`load()`	Loads data from Prodigy .xy files, sorts them into manageable arrays, scalars, and dicts, and returns everything as a dict. See dopey_dictionaries.pdf for more information about the sctructure of the data. If dopey does not recognize the data in the .xy file it will notify the user. The data will be returned non-sorted. Arguments: `file_name` (str), `shup` (bool) Returns: dict
`info()`	Takes a **dopey** dict as input and prints information about it contents. Arguments: `D` (dict) Returns: None
`dictContents()`	Takes a **dopey** dict as input and lists all it keys with info about their contents. Arguments: `D` (dict) Returns: None

dopey_plot.py

A couple of methods for visualizing the data loaded with load() or produced by other **dopey** methds.

`plot()`	Takes a **dopey** dict as input and generates a pyplot graph based on the passed keyword arguments. The ccepted keyword arguments are different for the various types of data. The accepted keyword arguments for the particular data passed as D are printed to screen (for not `shup = False`). Arguments: `D` (dict), `ax` (pyplot axis), `shup` (bool), and kwargs. Returns: pyplot axis Exception: Fermi map data is plotted in an interactive iPyWidget display by using method `fermiMapInteractive()`.
`fermiMapInteractive()`	See above.
`waterfallPlot()`	Generates waterfall plots from 2d data sets. Arguments: `D` (dict), `ax` (pyplot axis), `shup` (bool), and kwargs. Returns: pyplot axis

dopey_methods()

`subArray()`

Cuts out a subset of the data and returns it as a new **dopey** dict. For instance, an ARPES map reccorded over the -15 to +15 degree range can be reduced to -5 to +5 degrees by
`new_data = subArray(data, axis = 'y', v1 = -5, v2 = 5).`
See more about axes in dopey_dictionaries.pdf.

Arguments: D (dict), axis (str), v1 (float), v2 (float), and shup (bool).
Returns: dict.

`compact()`

Compacts one dimension of a data set. For instance, an iso-energy cut can be extracted from Fermi map data by (in combination with `subArray()`)
`new_data = compact(data, axis = "x")`

Arguments: `D` (dict), `axis` (str), and `shup` (bool).
Returns: dict.

`secondDerivative()`

Takes the 2nd derivative of a 1d or 2d data set.

Arguments: `D` (dict), `axis` (str), and `shup` (bool).
Returns: dict.

`shiftAxis()`

Arguments: `D` (dict), `axis` (str), `shift` (float), and `shup` (bool).
Returns: dict.

`gaussianSmooth()`

Arguments: `D` (dict), `sigma` (float or list), and `shup` (bool).
Returns: dict.

`align()`

A method to help with aligning a sample. Might at some point be moved to the dopey_plot.py file. Plots an iso-energy cut from a Fermi map measurements.

Arguments: `D` (dict).
Returns: None.

dopey_spin.py

`quickSpin()`

This method is the base method for spin data. It does a quick analysis and present it as plots of various kinds based on keyword arguments. It returns a new **dopey** dict that includes asymmetry, partial intenisties, etc. See dopey_dictionaries.pdf for details.

Arguments: `D` (dict), `shup` (bool), and `hide_plot` (bool).
Returns: dict.

Additional arguments: `coil` and `rotator` (integers). These arguments are used by other methods later on in the workflow. They have to be added manually here as Prodigy is not saving them as meta data in the .xy file.

Keyword arguments: These are mainly used for the plots. See help for plots.

`despikeSpin()`	Remove spikes based on keyword arguments and returns and updated **dopey** dict. The method is to calculate a mean value for each point (for all positive magnetisations and all negative magnetisations of the target) and ignore data points that are more than `N` (float) standard deviations from this. Arguments: `D` (dict), `N` (float), and `shup` (float). Returns: dict. Keyword arguments: `Nm` and `Np` (floats). Used in case different N:s are needed for positive and negative magnetisations.
`normalizeSpin()`	Provides various options for normalization of spin data based on keyword arguments. *Note: this method is only ready to accept spin edc and spin mdc data sets. Will be updated.* Arguments: `D` (dict) and `shup` (float). Returns: dict. Keyword arguments: `N` and `Np` (integers). `N` is the point number where the normalization is done and `Np` is the number of points to average over.
`insertSpinData()`	This method is used to combine two separate measurements of the same type. Use *e.g.* when a long measurement has been divided up into several shorter measurements. Works for spin edc, mdc, and map. Arguments: `D1` (dict), `D2` (dict), and `shup` (float). Returns: dict.
`inspectSpin()`	Provides a more detailed visualization of the collected data. Arguments: `D` (dict) and `shup` (float). Returns: None.
`polarization()`	Calculates the polarization based on one, two, or three measurements. Arguments: `D` (list), `sherman` (float), and `shup` (bool). Returns: dict. `D` is a list containing dicts from `quickSpin()`.
`rotatePolarization()`	This method is used to compensate for none-normal emission measurements. It rotates the calculated spin orientation from the lab coordinate system into the sample coordinate system. Arguments: `D` (dict), `polar` (float), and `shup` (bool). Returns: dict. The `D` dict is a dict from the `polarization()` method.
`updateSpinData()`	Use this method after manually manipulating the intensity arrays in dopey dicts. Ask the staff for details.

dopey_export2txt.py

Contains one method for exporting the sorted data from .xy files so that it can be loaded into prefered software.

`export2txt()` Export data from **dopey** dicts to text files. Accepts a selected number of measurement types (but at the moment not all).

Arguments: `D` (dict), `fn` (string), and `shup` (bool).
Returns: None.

`fn` is the name of the file to be saved. Preferably chose something simple, like `some_name.dat`.

dopey_constants.py

Contains no methods but defines various constants and values used by the methods in the other files.

dopey_dichroism.py

Methods for visualizing dichroism data.

`plotDichroism()` Takes two **dopey** dicts as input, one for CP+ and one for CP-, as Fermi maps or ARPES cuts, and displays an interactive plot.

Arguments: `D1` and `D2` (dicts).
Returns: None.

`dichroism()` Takes two **dopey** dicts as input, one for CP+ and one for CP-, as Fermi maps or ARPES cuts (or other 2d cuts extracted from Fermi maps) and returns a new dict containing dichroism data.

Arguments: `D1` and `D2` (dicts).
Returns: Dict.

Keyword arguments: depends on type of data. Use `help(dichroism)` for details.

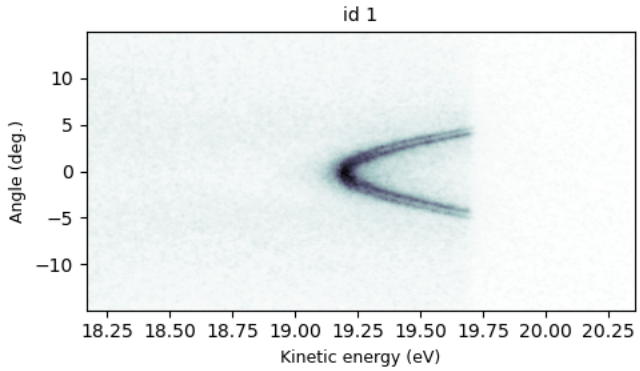
Examples

Providing a number of simple examples, using the various methods on different type of measurements.

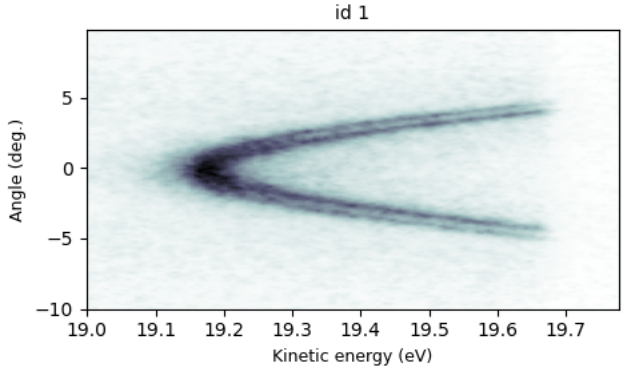
Import the **dopey** package: `import dopey`

ARPES

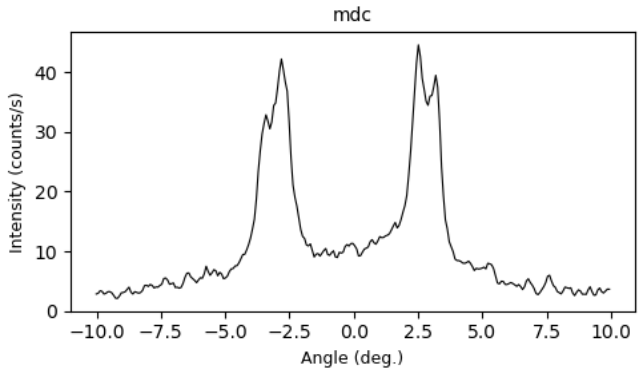
```
data = dopey.load("arpes.xy")
dopey.plot(data)
```




```
data = dopey.subArray(data, axis = "x", v1 = 19.00, v2 = 19.80)
data = dopey.subArray(data, axis = "y", v1 = -10, v2 = 10)
dopey.plot(data)
```

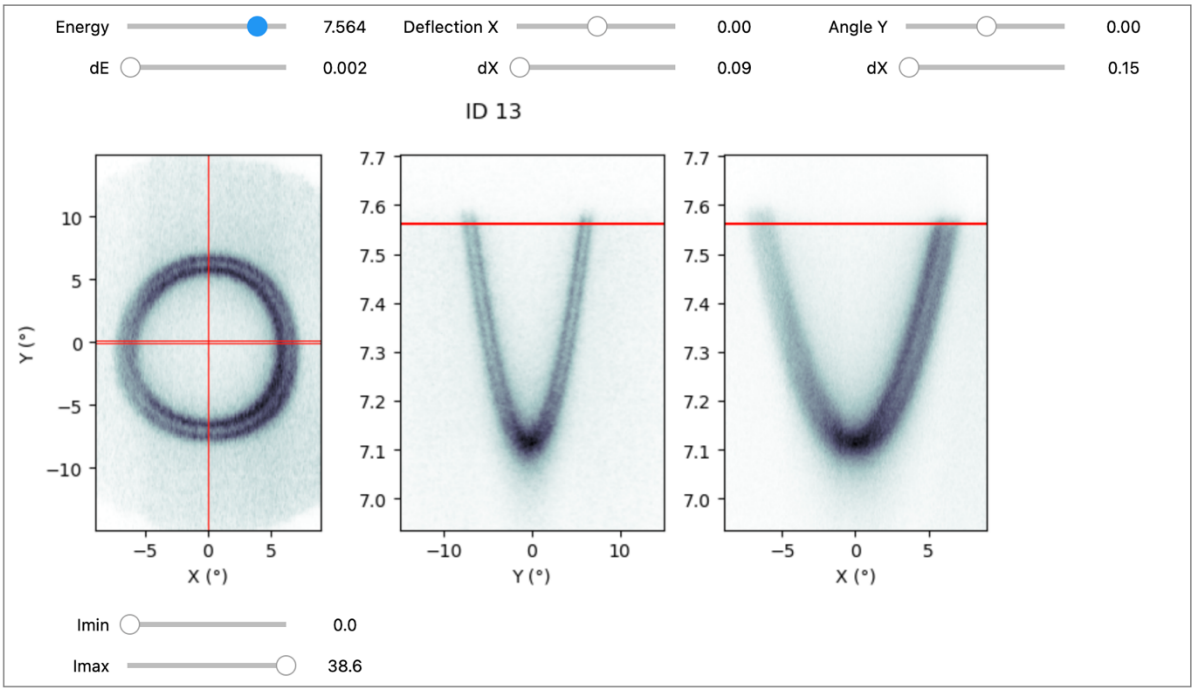



```
data = dopey.subArray(data, axis = "x", v1 = 19.40, v2 = 19.42)
data = dopey.compact(data, axis = "x")
dopey.plot(data, title = “mdc”)
```



Fermi map

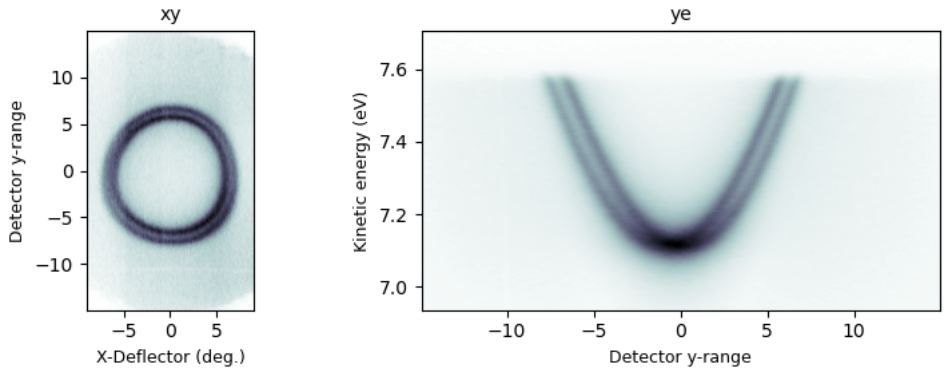
```
data = dopey.load("fermi_map.xy")
dopey.plot(data)
```




```
fig, ax = plt.subplots(ncols = 2, figsize = (8,3))

cut_xy = dopey.subArray(data, axis = "x", v1 = 7.56, v2 = 7.58)
cut_xy = dopey.compact(cut_xy, axis = "x")
ax[0] = dopey.plot(cut_xy, rotate = True, aspect = "equal", title = "xy",
ax = ax[0])

cut_ye = dopey.subArray(data, axis = "z", v1 = -1, v2 = 1)
cut_ye = dopey.compact(cut_ye, axis = "z")
ax[1] = dopey.plot(cut_ye, rotate = True, title = "ye", ax = ax[1])
```

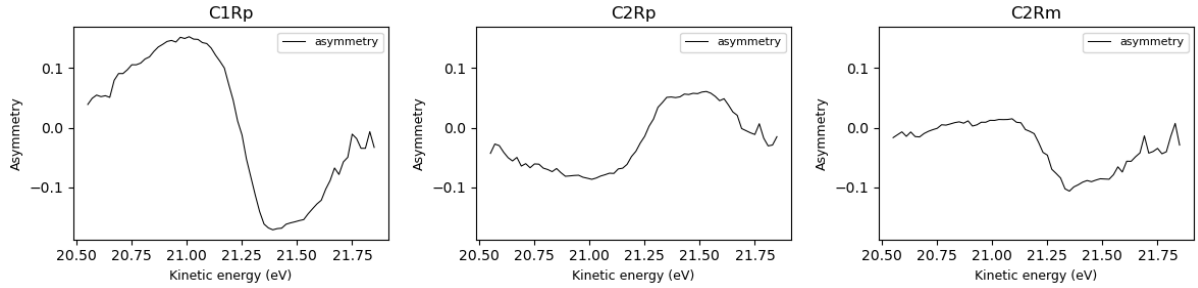


Spin EDC

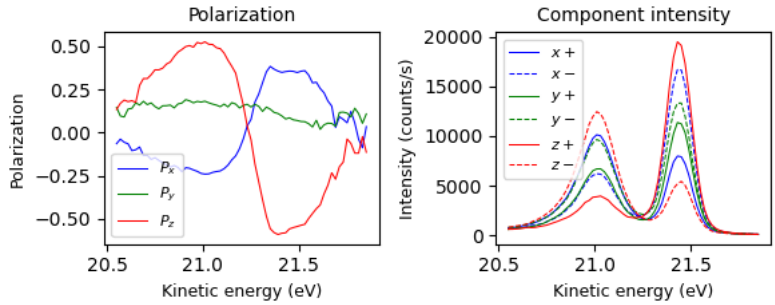
```
edc1 = dopey.load("data/WSe2_2/id6.xy")
edc2 = dopey.load("data/WSe2_2/id7.xy")
edc3 = dopey.load("data/WSe2_2/id8.xy")

edc1 = dopey.quickSpin(edc1, coil = 1, rotator =  1, hide_plot = True)
edc2 = dopey.quickSpin(edc2, coil = 2, rotator =  1, hide_plot = True)
edc3 = dopey.quickSpin(edc3, coil = 2, rotator = -1, hide_plot = True)
```

```
fig, ax = plt.subplots(ncols = 3, figsize = (12,3))
ax[0] = dopey.plot(edc1, ax = ax[0], intensity = "asymmetry")
ax[1] = dopey.plot(edc2, ax = ax[1], intensity = "asymmetry")
ax[2] = dopey.plot(edc3, ax = ax[2], intensity = "asymmetry")
for a, ttl in zip(ax, ["C1Rp", "C2Rp", "C2Rm"]):
    a.set_title(ttl)
    a.set_ylim(ax[0].get_ylim())
```




```
pol = dopey.polarization(D = [edc1, edc2, edc3])
dopey.plot(pol)
```




```
pol = dopey.rotatePolarization(pol, 32)
dopey.plot(pol)
```