Python-based XPS fitting by *LMFIT* with Excel VBA assistant

Hideki NAKAJIMA, 1 Feb. 2018, rev. 9 Aug. 2019

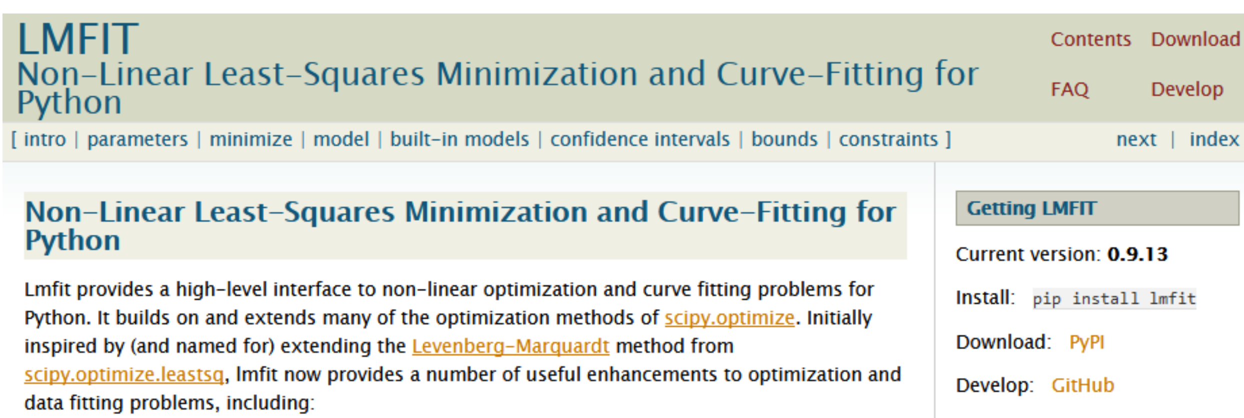
Abstract: Python is open source programming language but still robust with powerful scientific library available online. For spectroscopy, the curve fitting quantitates the elements and their chemical states based on the knowledge of atomic and molecular structures and instrumentations, which constraint the parameters of peaks and backgrounds in the spectrum. The LMFIT package is only available for detailed setup on constraints to meet spectroscopic demands from users in Python platform. Excel XPS analysis package “EX3ms” performs SR-based XPS curve fitting analysis, and it can generate the command lines for LMFIT to inspect the fitting results consistent. This technical note briefly summarizes how to run the LMFIT with Excel VBA assistant.

Think Python 2e to learn Python programming for beginners
http://greenteapress.com/wp/think-python-2e/

Anaconda distribution for Python GUI
https://www.anaconda.com/download/

python™ ANACONDA.

Non-Linear Least-Squares Minimization and Curve-Fitting (LMFIT) package by M. Newville.
https://lmfit.github.io/lmfit-py/
http://lmfit.github.io/lmfit-py/installation.html
https://groups.google.com/forum/#!forum/lmfit-py



LMFIT
Non-Linear Least-Squares Minimization and Curve-Fitting for Python

Contents Download FAQ Develop

[intro | parameters | minimize | model | built-in models | confidence intervals | bounds | constraints] next | index

Non-Linear Least-Squares Minimization and Curve-Fitting for Python

Lmfit provides a high-level interface to non-linear optimization and curve fitting problems for Python. It builds on and extends many of the optimization methods of scipy.optimize. Initially inspired by (and named for) extending the Levenberg-Marquardt method from scipy.optimize.leastsq, lmfit now provides a number of useful enhancements to optimization and data fitting problems, including:

Getting LMFIT

Current version: **0.9.13**

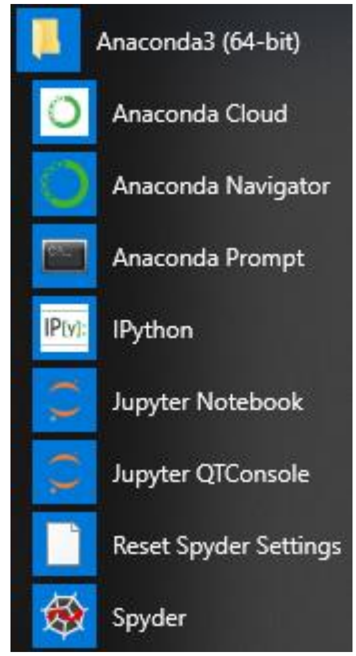
Install: `pip install lmfit`

Download: PyPI

Develop: GitHub

How to install lmfit Python packages in Anaconda.

https://www.quora.com/How-do-I-install-Python-packages-in-Anaconda



- Install new package in Anaconda (not in conda-forge)
  - Quit Python
  - Anaconda Prompt
  - > g:
  - > cd ProgramData\Anaconda3\Scripts
  - > pip install lmfit

- For MacOS
  - - Open terminal
  - - cd anaconda¥bin
  - - pip install lmfit

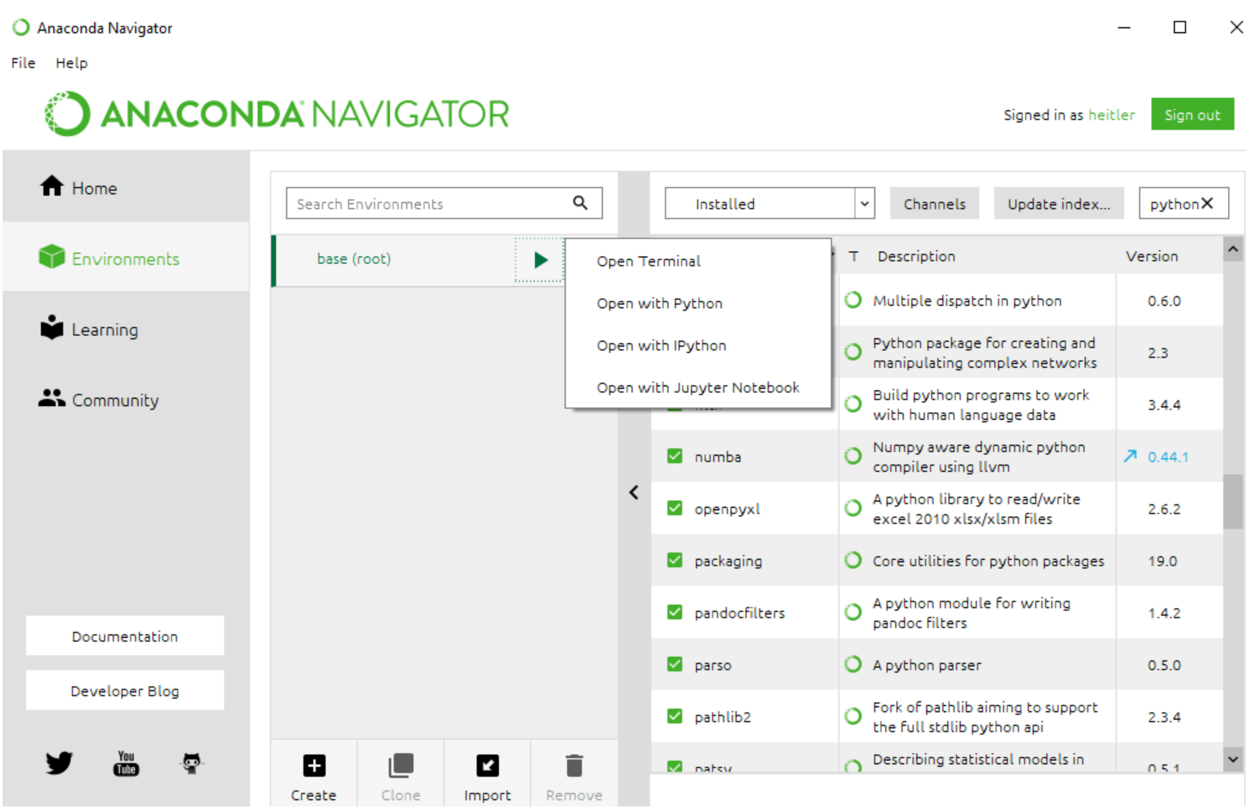
- Install new package in Anaconda in conda-forge
  - - Anaconda Navigator
  - - Environment - root - all - select packages

How to update the Anaconda Python environment.

https://www.anaconda.com/keeping-anaconda-date/

Open Anaconda prompt

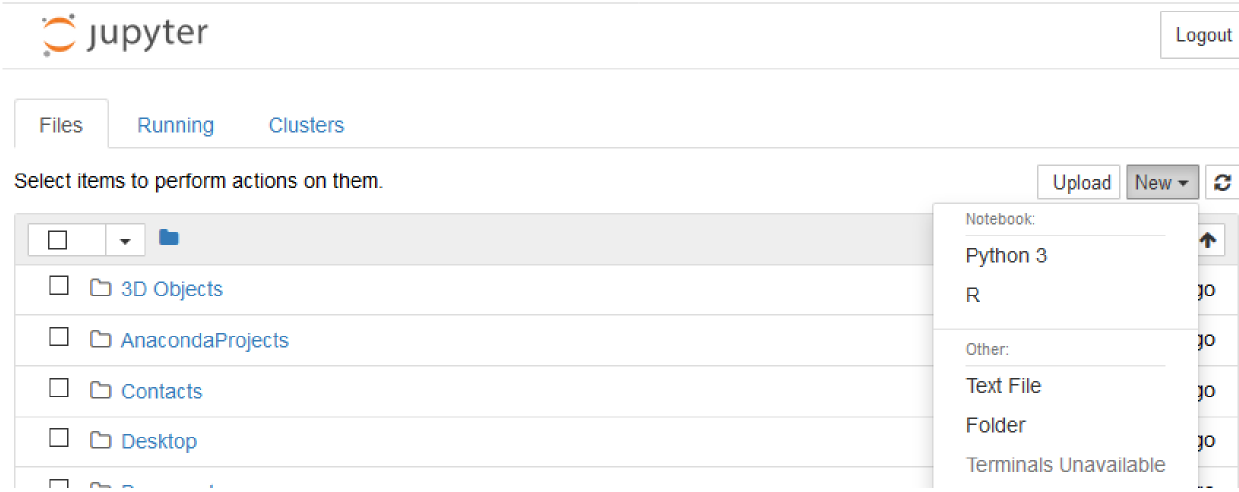
> conda update --all

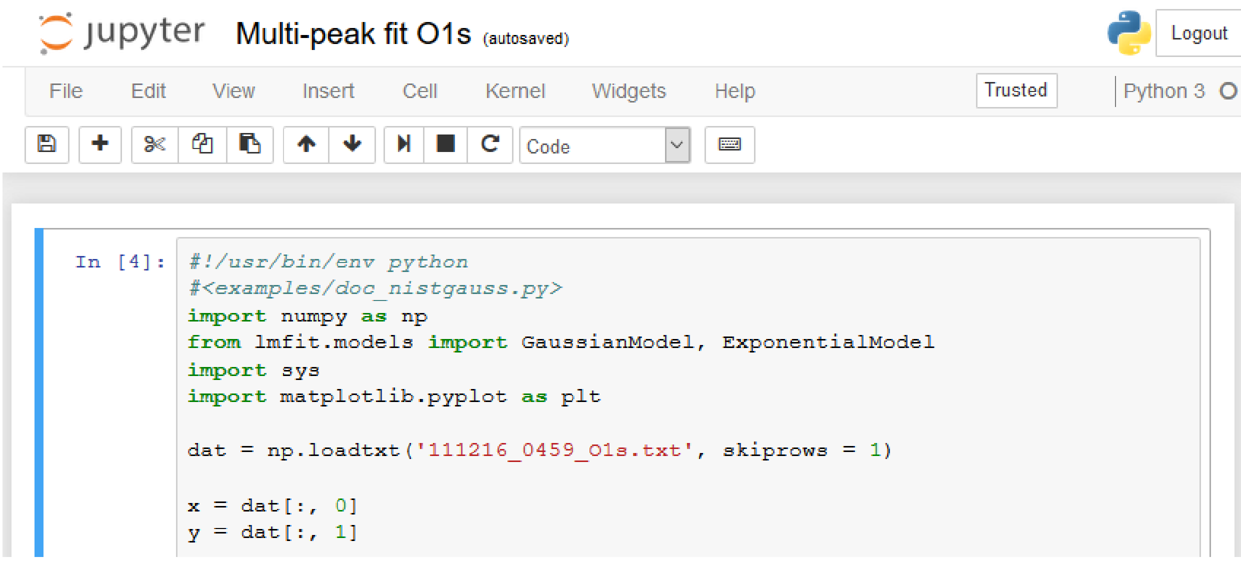


How to run the Python environment based on your browser.

1. Open Jupyter Notebook.
    - Tutorial: https://plot.ly/python/ipython-notebook-tutorial/
2. Export spectral data in the binding energy calibrated from Excel into text.
    - Run the code with “exp2” in the A1 cell of Graph sheet.
    - Run the code in Exp sheet to export data in text file.
3. Fit the spectrum by Excel VBA.
    - GitHub: https://github.com/heitler/xps-excel-macro
4. Export the python code in Fit sheet.
    - “lmfit” in D1 cell and run the code.
5. Upload a text data file under Jupyter home directory such as “C:\Users*username*\”.
6. Create new notebook and paste a whole template script as described below.
7. Replace the green parts with the code generated in the Excel VBA Pyt sheet. However, the script generation is limited as below.
    - Gaussian peak and polynomial baseline only
    - Use standard deviation (sigma) for width; FWHM = sigma * 2.3548
    - See more https://lmfit.github.io/lmfit-py/builtin_models.html
        i. Peak-like models: Lorentzian, Voigt, PseudoVoigt, etc.
        ii. Step-like models: Arctan, Erf
        iii. Baseline models: Exponential, PowerLaw, etc.
8. Run the python script by shift + enter.

	A	B	C	D	E	F	G	H
1								
2	import numpy as np							
3	from lmfit.models import GaussianModel, ExponentialModel, PolynomialModel							
4	import matplotlib.pyplot as plt							
5	import xpspy as xpy							
6								
7	dat = np.loadtxt('110904_1625_1.txt', skiprows = 1)							
8	x0 = dat[:, 0]							
9	y0 = dat[:, 1]							
10								
11	xmin = 77							
12	xmax = 97							
13	[x, y] = xpy.fit_range(x0, y0, xmin, xmax)							
14								
15	x_bg = xpy.shirley_calculate(x, y, 0.0001, 10)							
16	y = y - x_bg							
17								
18	gauss1 = GaussianModel(prefix='g1_')							
19	pars = gauss1.make_params()							
20	pars['g1_center'].set(83.9)							
21	pars['g1_sigma'].set(1, min=0.2, max=4)							
22	pars['g1_amplitude'].set(2695.99, min=0)							
23								
24	gauss2 = GaussianModel(prefix='g2_')							
25	pars.update(gauss2.make_params())							
26	pars['g2_center'].set(87.5)							
27	pars['g2_sigma'].set(0.92, min=0.2, max=4)							
28	pars['g2_amplitude'].set(2021.99, min=0)							





Example Python code for XPS Au4f, C1s, O1s, and NEXAFS Fe L3 edge.
https://gist.github.com/heitler/5a90520ff36ec9656fa1d6c89d5fbfab

- xpspy.py package includes the Shirley and Tougaard background subtraction at a specific energy range.

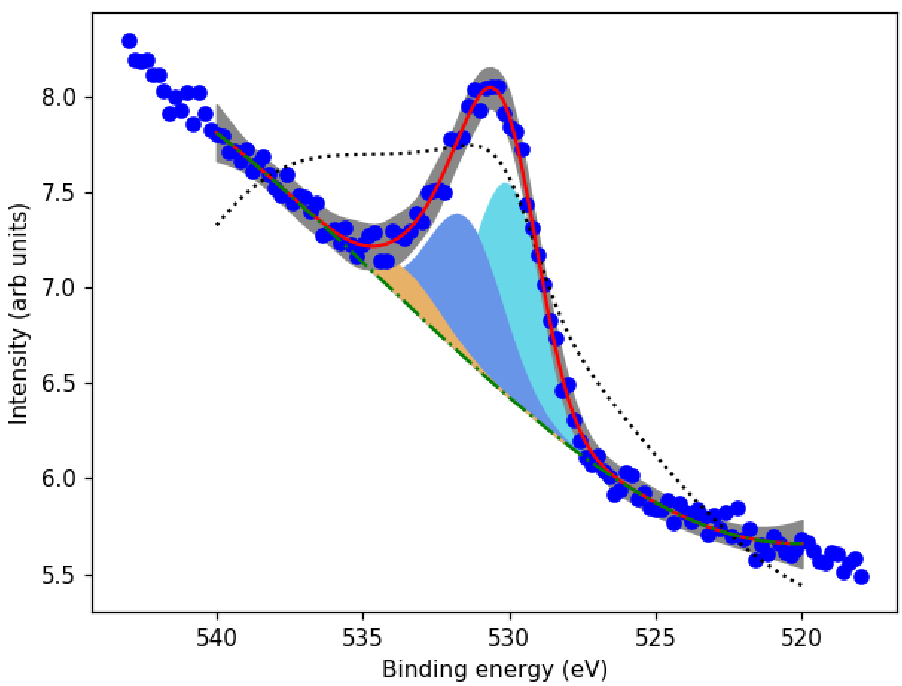
Example Python code for Shirley background.
https://github.com/kaneod/physics/blob/master/python/specs.py

- To import Shirley BG, save shirleybg.py in the same directory.

Template #1 for O1s:

```
import numpy as np
from lmfit.models import GaussianModel, ExponentialModel, PolynomialModel
import matplotlib.pyplot as plt
import xpspy as xpy

dat = np.loadtxt('111216_0459_O1s.txt', skiprows = 1)

x0 = dat[:, 0]
y0 = dat[:, 1]

xmin = 520
xmax = 540

[x, y] = xpy.fit_range(x0, y0, xmin, xmax)

#bg_mod = ExponentialModel(prefix='bg_')
#pars = bg_mod.guess(y, x=x)
bg_mod = PolynomialModel(3, prefix='bg_')
pars = bg_mod.guess(y, x=x)

gauss1  = GaussianModel(prefix='g1_')
pars.update( gauss1.make_params())
pars['g1_center'].set(530, vary = False)
pars['g1_sigma'].set(1.08, min=0.2, max=2)
pars['g1_amplitude'].set(1.03, min=0)

gauss2  = GaussianModel(prefix='g2_')
pars.update( gauss2.make_params())
pars['g2_center'].set(531.5, vary = False)
pars['g2_sigma'].set(1.27, min=0.2, max=2)
pars['g2_amplitude'].set(0.9, min=0)

gauss3  = GaussianModel(prefix='g3_')
pars.update( gauss3.make_params())
pars['g3_center'].set(533.5, vary = False)
pars['g3_sigma'].set(1.27, min=0.2, max=2)
pars['g3_amplitude'].set(0.25, min=0)

mod = bg_mod + gauss1 + gauss2 + gauss3

init = mod.eval(pars, x=x)
out = mod.fit(y, pars, x=x)
comps = out.eval_components(x=x)
# print(out.fit_report())

%matplotlib notebook
# %matplotlib inline
```

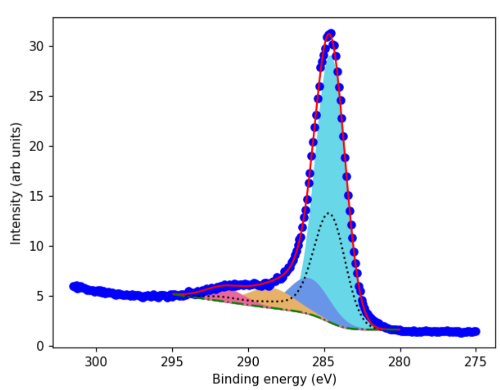
```
plt.figure(1)

plt.xlabel('Binding energy (eV)')
plt.ylabel('Intensity (arb units)')

plt.plot(x0, y0, 'bo')
# plt.plot(x, y+x_bg, 'bo')
plt.plot(x, init, 'k:')
plt.plot(x, out.best_fit, 'r-')
plt.fill_between(x, comps['g1_']+comps['bg_'], comps['bg_'], color="#68d7e8")
plt.fill_between(x, comps['g2_']+comps['bg_'], comps['bg_'], color="#6895e8")
plt.fill_between(x, comps['g3_']+comps['bg_'], comps['bg_'], color="#e8b168")
#plt.plot(x, comps['g1_'], 'm--')
#plt.plot(x, comps['g2_'], 'm--')
#plt.plot(x, comps['g3_'], 'm--')
#plt.plot(x, comps['bg_'], 'g-.')
plt.plot(x, comps['bg_'], 'g-.')

dely = out.eval_uncertainty(sigma=5)
plt.fill_between(x, out.best_fit-dely, out.best_fit+dely, color='#888888')

plt.gca().invert_xaxis()
plt.show()
```



Template #2 for C1s:

```
import numpy as np
from lmfit.models import GaussianModel, ExponentialModel, PolynomialModel
import matplotlib.pyplot as plt
import xpspy as xpy

dat = np.loadtxt('120324_1144_C1s.txt', skiprows = 1)

x0 = dat[:, 0]
y0 = dat[:, 1]

xmin = 280
xmax = 295

[x, y] = xpy.fit_range(x0, y0, xmin, xmax)

#bg_mod = ExponentialModel(prefix='exp_')
#pars = bg_mod.guess(y, x=x)
#bg_mod = PolynomialModel(3, prefix='poly_')
#pars = bg_mod.guess(y, x=x)

#x_bg = xpy.shirley_calculate(x, y, 1e-5, 10)
#y = y - x_bg
x_bg = xpy.tougaard_calculate(x, y, 2866, 1643, 1, 1)
y = y - x_bg

gauss1  = GaussianModel(prefix='g1_')
pars = gauss1.make_params()
#pars.update( gauss1.make_params())
pars['g1_center'].set(284.59, vary = False)
pars['g1_sigma'].set(1.01, min=0.2, max=1.5)
pars['g1_amplitude'].set(26.36, min=0)

gauss2  = GaussianModel(prefix='g2_')
pars.update( gauss2.make_params())
pars['g2_center'].set(286, vary = False)
pars['g2_sigma'].set(1.27, min=0.2, max=1.5)
pars['g2_amplitude'].set(2.63, min=0)

gauss3  = GaussianModel(prefix='g3_')
pars.update( gauss3.make_params())
pars['g3_center'].set(288.5, vary = False)
pars['g3_sigma'].set(1.27, min=0.2, max=1.5)
pars['g3_amplitude'].set(1.54, min=0)

gauss4  = GaussianModel(prefix='g4_')
pars.update( gauss4.make_params())
pars['g4_center'].set(291.59, vary = False)
```

```
pars['g4_sigma'].set(1.27, min=0.2, max=1.5)
pars['g4_amplitude'].set(1.35, min=0)

#mod = gauss1 + gauss2 + gauss3 + gauss4 + x_mod
mod = gauss1 + gauss2 + gauss3 + gauss4

init = mod.eval(pars, x=x)
out = mod.fit(y, pars, x=x)
comps = out.eval_components(x=x)
# print(out.fit_report(min_correl=0.5))

%matplotlib notebook
# %matplotlib inline
plt.figure(1)

plt.xlabel('Binding energy (eV)')
plt.ylabel('Intensity (arb units)')

plt.plot(x0, y0, 'bo')
# plt.plot(x, y+x_bg, 'bo')
plt.plot(x, init+x_bg, 'k:')
plt.plot(x, out.best_fit+x_bg, 'r-')
plt.fill_between(x, comps['g1_']+x_bg, x_bg, color="#68d7e8")
plt.fill_between(x, comps['g2_']+x_bg, x_bg, color="#6895e8")
plt.fill_between(x, comps['g3_']+x_bg, x_bg, color="#e8b168")
plt.fill_between(x, comps['g4_']+x_bg, x_bg, color="#e86897")
plt.plot(x, x_bg, 'g-.')

# dely = out.eval_uncertainty(sigma=5)
# plt.fill_between(x, out.best_fit-dely+x_bg, out.best_fit+dely+x_bg, color='#888888')

plt.gca().invert_xaxis()
plt.show()
```

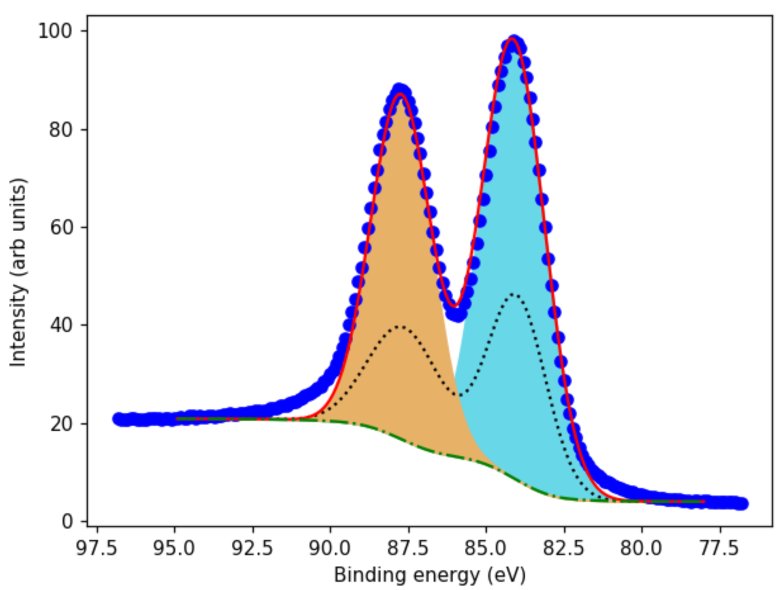



```
Tougaard B: 1383.6866647338247 , C: 1643 , C': 1 , D: 1
```

Template #3 for Au4f:

```
import numpy as np
from lmfit.models import GaussianModel, ExponentialModel, PolynomialModel
import matplotlib.pyplot as plt
import xpspy as xpy

dat = np.loadtxt('110904_1625_Au4f.txt', skiprows = 1)

x0 = dat[:, 0]
y0 = dat[:, 1]

xmin = 78
xmax = 95

[x, y] = xpy.fit_range(x0, y0, xmin, xmax)

#bg_mod = ExponentialModel(prefix='bg_')
#pars = bg_mod.guess(y, x=x)
#bg_mod = PolynomialModel(3, prefix='bg_')
#pars = bg_mod.guess(y, x=x)

x_bg = xpy.shirley_calculate(x, y, 1e-5, 10)
y = y - x_bg
# x_bg = xpy.tougaard_calculate(x, y, 2866, 1643, 1, 1)
# y = y - x_bg

gauss1  = GaussianModel(prefix='g1_')
pars = gauss1.make_params()
#pars.update( gauss1.make_params())
pars['g1_center'].set(84)
pars['g1_sigma'].set(0.94, min=0.2, max=4)
pars['g1_amplitude'].set(88.45, min=0)

gauss2  = GaussianModel(prefix='g2_')
pars.update( gauss2.make_params())
pars['g2_center'].set(87.6)
pars['g2_sigma'].set(1.15, min=0.2, max=4)
pars['g2_amplitude'].set(66.34, min=0)

pars.add('g2_amplitude', expr = 'g1_amplitude * 3 / 4')
pars.add('g2_center', expr = 'g1_center + 3.6 ')

# mod = gauss1 + gauss2 + bg_mod
mod = gauss1 + gauss2

init = mod.eval(pars, x=x)
out = mod.fit(y, pars, x=x)
comps = out.eval_components(x=x)
```

```
# print(out.fit_report(min_correl=0.5))

%matplotlib notebook
# %matplotlib inline
plt.figure(1)

plt.xlabel('Binding energy (eV)')
plt.ylabel('Intensity (arb units)')

plt.plot(x0, y0, 'bo')
# plt.plot(x, y+x_bg, 'bo')
plt.plot(x, init + x_bg, 'k:')
plt.plot(x, out.best_fit + x_bg, 'r-')
plt.fill_between(x, comps['g1_'] + x_bg,  x_bg, color="#68d7e8")
plt.fill_between(x, comps['g2_'] + x_bg,  x_bg, color="#e8b168")
#plt.plot(x, comps['g1_'], 'm--')
#plt.plot(x, comps['g2_'], 'm--')
#plt.plot(x, comps['x_'], 'g-.')
plt.plot(x, x_bg, 'g-.')

# dely = out.eval_uncertainty(sigma=5)
# plt.fill_between(x, out.best_fit-dely + x_bg, out.best_fit+dely + x_bg, color='#888888')

plt.gca().invert_xaxis()
plt.show()
```

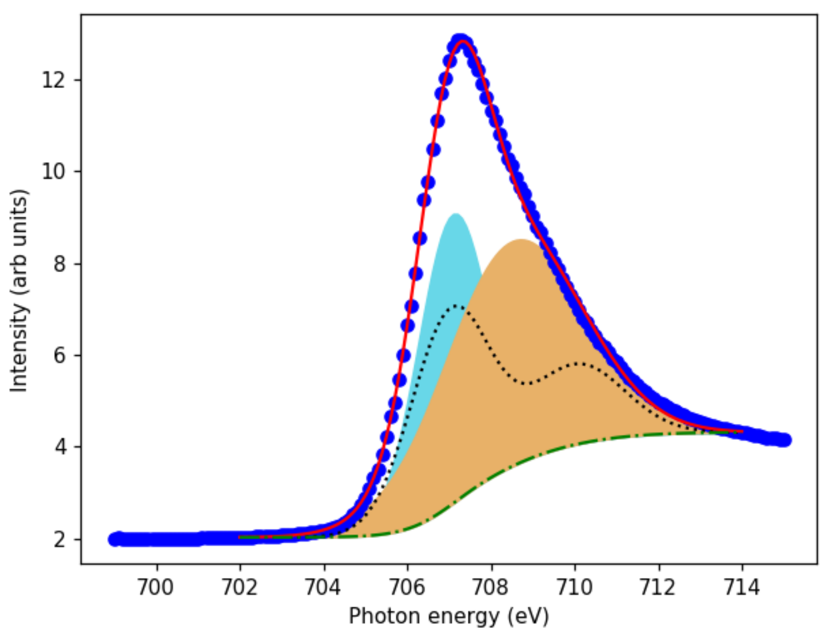


Shirley BG: tol (ini = 1e-05): 0.0 , iteration (max = 10): 6

Template #4 for Fe L3 edge:

```
import numpy as np
from lmfit.models import GaussianModel, ExponentialModel, PolynomialModel
import sys
import matplotlib.pyplot as plt
import xpspy as xpy

dat = np.loadtxt('130310_2254_FeL3.txt', skiprows = 1)

x0 = dat[:, 0]
y0 = dat[:, 1]

xmin = 702
xmax = 714

[x, y] = xpy.fit_range(x0, y0, xmin, xmax)

#bg_mod = ExponentialModel(prefix='bg_')
#pars = bg_mod.guess(y, x=x)
#bg_mod = PolynomialModel(3, prefix='bg_')
#pars = bg_mod.guess(y, x=x)

x_bg = xpy.shirley_calculate(x, y, 1e-5, 10)
y = y - x_bg
# x_bg = xpy.tougaard_calculate(x, y, 2866, 1643, 1, 1)
# y = y - x_bg

gauss1  = GaussianModel(prefix='g1_')
pars = gauss1.make_params()
#pars.update( gauss1.make_params())
pars['g1_center'].set(707)
pars['g1_sigma'].set(0.94, min=0.2, max=4)
pars['g1_amplitude'].set(10, min=0)

gauss2  = GaussianModel(prefix='g2_')
pars.update( gauss2.make_params())
pars['g2_center'].set(710)
pars['g2_sigma'].set(1.15, min=0.2, max=4)
pars['g2_amplitude'].set(5, min=0)

# mod = gauss1 + gauss2 + bg_mod
mod = gauss1 + gauss2

init = mod.eval(pars, x=x)
out = mod.fit(y, pars, x=x)
comps = out.eval_components(x=x)
# print(out.fit_report(min_correl=0.5))
```

```
%matplotlib notebook
# %matplotlib inline
plt.figure(1)

plt.xlabel('Photon energy (eV)')
plt.ylabel('Intensity (arb units)')

plt.plot(x0, y0, 'bo')
# plt.plot(x, y+x_bg, 'bo')
plt.plot(x, init + x_bg, 'k:')
plt.plot(x, out.best_fit + x_bg, 'r-')
plt.fill_between(x, comps['g1_'] + x_bg,  x_bg, color="#68d7e8")
plt.fill_between(x, comps['g2_'] + x_bg,  x_bg, color="#e8b168")
#plt.plot(x, comps['g1_'], 'm--')
#plt.plot(x, comps['g2_'], 'm--')
#plt.plot(x, comps['x_'], 'g-.')
plt.plot(x, x_bg, 'g-.')

# dely = out.eval_uncertainty(sigma=5)
# plt.fill_between(x, out.best_fit-dely + x_bg, out.best_fit+dely + x_bg, color='#888888')

plt.show()
```



Shirley BG: tol (ini = 1e-05): 0.0 , iteration (max = 10): 6