PyFitIt Manual

17 May 2019

The major part of the code is hidden from user in the list of Python library files, located in pyfitit folder. The control panel is represented by several Jupyter Notebooks which help to setup calculations and visualize results. Places in the Notebooks which must be adopted by user are highlighted by **#Modify** tag. The same places in the Manual are highlighted by color.

0. Fitting smooth

This is a preparation step which helps to adjust parameters of convolution and shift between theoretical spectra and experimental ones

0.1 Import experimental spectrum

```
expSpectrum = readSpectrum('exp_ground.txt', energyColumn=0,
intensityColumn=1,
skiprows = 1)
```

parameters **energyColumn**, **intensityColumn** are used to specify which columns in file with experimental spectrum should be used for plotting, **skiprows** helps to avoid headers.

```
project = createPartialProject(expSpectrum = expSpectrum, fdmnesShift = 7113)
```

specify here roughly the shift between calculated and experimental spectrum

0.2 Import theoretical spectrum

```
xanes = parseFdmnesFolder('fdmnes_fdm_5')
```

specify the folder of FDMNES calculation, which should be copied to the folder with Jupyter Notebook file. Inside FDMNES folder the standard files of calculation should be present: fdmfile, input file and output file (without renaming them after calculation).

1. Create a project

Place in the working folder XYZ file with the structure and file with experimental spectrum (arbitrary header, two columns: energy and XANES).

1.1 Name of the XYZ structure file:

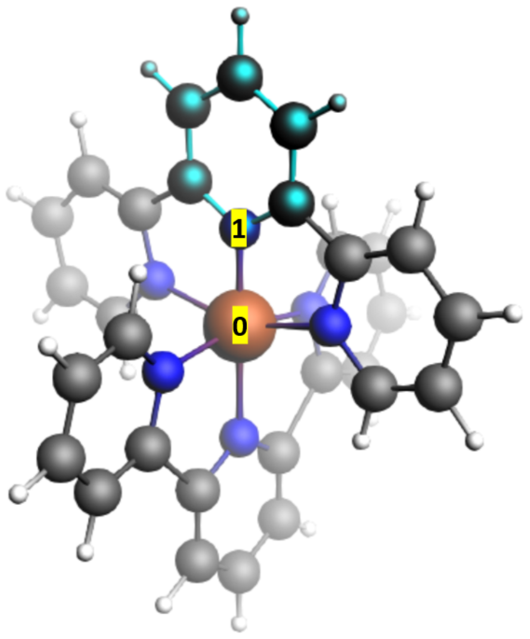
```
m = Molecule(join(projectFolder,'Fe_terpy.xyz'))
```

1.2 Split molecule into parts.

Each part is described by the string which contains atomic numbers from xyz file. By default (when **setParts** is not used), all atoms in the molecule belong to part 0.

```
m.setParts('0', '1-9', '10-19', '20-29', '30-38', '39-48', '49-58')
```

where atom 0 was attached to part 0, atoms 1-9 to part 1, atoms 10-19 to part 2 and so on (numeration in Python starts always from 0).



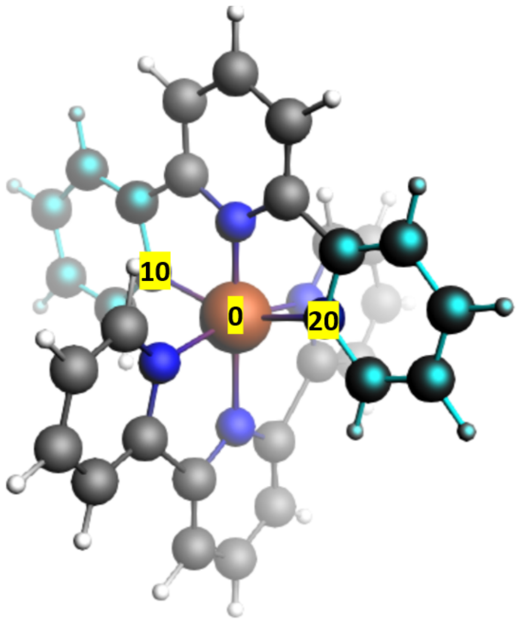


Figure 1. Atom '0' is central orange Fe atom, atoms '1-9' in part 1 are highlighted in left panel, atoms '10-19' in part 2 and '20-29' in part 3 are highlighted in the right panel.

User can select any atoms to group, e.g.:

```
m.setParts('0', '1, 2, 5', '3, 4, 6-11', '12-29', '30-38', '39-48', '49-58')
```

Then coordinates of e.g. twelfth atom (python index 11), which belongs to the third part (python index 2) can be accessed by two equivalent commands **`m.atom[11]`** or **`m.part[2].atom[1]`**. User can further modify positions of the atoms:

```
m.atom[0] = [1,1,1]
```

1.3 Define deformations.

Each deformation is described by its name, parts involved, axis and type.
Example 1:

```
deformation = 'centralRing1_Shift'
axis = normalize(m.atom[1] - m.atom[0])
m.part[1].shift(axis*params[deformation])
```

This example describes deformation named 'centralRing1_Shift', which corresponds to the translation of part 1 along bond between atoms 1 and 0 (see Figure 1). Array **`params[deformation]`** specifies the sliders with the name of deformations which will be used further for fitting.

Example 2:

```
                     deformation = 'centralRing1_Rotate'
                     axis = normalize(m.atom[1] - m.atom[0])
                     m.part[1].rotate(axis, m.atom[0],
params[deformation])
```

This example describes deformation named 'centralRing1_Rotate', which corresponds to the rotation of part 1 around the axis parallel to the bond between atoms 1 and 0, which passes through atom 0.

Example 3:

```
                     deformation = 'sideRings1_Elong'
                     axis1 = normalize(m.atom[10] - m.atom[0])
                     axis2 = normalize(m.atom[20] - m.atom[0])
                     m.part[2].shift(axis1*params[deformation])
                     m.part[3].shift(axis2*params[deformation])
```

This example describes deformation named **`'sideRings1_Elong'`**, which corresponds to the simultaneous translation of **`part 2`** along bond between atoms 10 and 0 and **`part 3`** along bond between atoms 20 and 0 (see Figure 1). Shift is a vector and specified in the example above via axis. It can also be specified by a vector as [1,0,1].

The blocks provided in examples are repeated for each required deformation. User can rotate and shift either part or the whole molecule:

```
m.rotate(axis, center, angle)
m.part[0].rotate(axis, center, angle)
m.shift(shift)
m.part[0].shift(shift)
```

The **`axis`** command specifies the axis for rotation. It can be written as a vector in squared brackets, e.g. **`axis = [0,1,0]`** or using coordinates of atoms, e.g. **`axis = m.atom[0]`** moreover, given two axis, it is possible to define their differences and even their vector product as **`axis = cross(v1, v2)`**. The angle of rotation is specified in radians (see below). To facilitate the implementation of geometric transformations and shorten the code we introduced functions that work with the positions of atoms as vectors:

```
v = m.part[0][0]
v = v / norm(v)
axis = cross(v,vc)
m1 = copy(m)
```

cross - vector product
dot - scalar product
norm - Euclidean vector norm
copy - creates a copy of this molecule

1.4 Name of the file with experiment.

```
def projectConstructor(expFile='exp_ground.txt'):
```

place file with experiment in the same folder with Jupyter Notebook

1.5 Load experimental spectrum

```
project.spectrum = readSpectrum(filePath, energyColumn=0, intensityColumn=1,
skiprows = 1)
```

Specify energy and XANES column numbers (numbering starts from zero) and number of header lines to skip.

1.6 Number of spectrum points for fitting

```
project.maxSpectrumPoints = 100
```

To save computational time the convolution for selected part of spectrum (see p.1.7) is calculated only for a given number of points. The density of this points is larger in energy intervals where experimental spectrum varies more.

1.7 Energy interval for fitting.

```
a = 7113; b = 7213
```

1.8 Ranges of deformations.

```
project.geometryParamRanges = {
'centralRing1_Shift': [-0.3, 0.5],
'sideRings1_Shift': [-0.3, 0.5],
'sideRings1_Elong': [-0.3, 0.5],
'centralRing2_Shift': [-0.3, 0.5],
'sideRings2_Shift': [-0.3, 0.5],
'sideRings2_Elong': [-0.3, 0.5]
}
```

Specify the ranges of deformations, described previously in Section 1.3. For angles, the ranges are specified in radians.

1.9 Parameters of FDMNES calculation.

```
project.FDMNES_calc = {
'Energy range': '-15 0.02 8 0.1 18 0.5 30 2 54 3 117',
'Green': False,
'radius': 5,
}
```

Specify the energy grid for calculation of theoretical spectrum, whether multiple scattering approximation in muffin tin potential is used or not (`'Green': False` or `'Green': True`) and radius of cluster for calculations. Other parameters of FDMNES calculations can also be tuned. See the whole list of accessible commands the fdmnes.py file inside script folder pyfitit.

1.10. Default parameters for convolution.

```
project.FDMNES_smooth = {
'Gamma_hole': 2,
'Ecent': 25,
'Elarg': 50,
'Gamma_max': 15,
'Efermi': 5,
'shift': 7113,
}
```

See FDMNES manual, section Energy Convolution for explanation of these parameters.

1.11 Check deformations

Execute commands:

```
project = projectConstructor()
project.constructMoleculesForEdgePoints()
```

and check XYZ files created in the project directory, which correspond to the molecules with applied deformations (p.1.3 and 1.9). Molecules are saved in files mol_param_value.xyz.

1.12 Save project

```
saveAsProject('FeterpyProject.py')
```

Comments:

The function **`moleculeConstructor`** must have two arguments: **`project`** and **`params`**. Parameter **`project`** is the project in which the molecule will be used. This variable can be useful in cases where the process of creating a molecule depends on the settings of the entire project. Parameter **`params`** defines the geometric parameters that are used to create the molecule. These parameters are configured by the user and correspond to the sliders in the fitting spectrum function **`fitBySliders`**.

2. Calculate XANES for a set of geometries

2.2 Generate input files for XANES training set

```
folder = 'sample'
generateInputFiles(project.geometryParamRanges,
project.moleculeConstructor, sampleCount=100,
method='IHS', spectralProgram='fdmnes',
spectrCalcParams = project.FDMNES_calc,
folder=folder)
```

Alternatives to IHS

```
method = 'grid', sampleCount = {'centralRing1_Shift': 5, 'sideRings1_Shift':
5, 'sideRings1_Elong': 5, 'centralRing2_Shift': 5, 'sideRings2_Shift': 5,
'sideRings2_Elong': 5}

method = 'random', sampleCount = 100

method = 'line', sampleCount = 100, Optional: lineEdges = {'start':{...},
'end':{...}}
```

for lineEdges

The function **`generateInputFiles`** in user define folder creates subfolders for each sampling point in the structural parameters space. The number of points for each calculation is specified by the variable **`sampleCount`**, and user can select one of the three methods to distribute points in the space of structural parameters: **`method = 'IHS'`**, **`'grid'`** or **`'random`**'. Parameters for calculations by FDMNES are taken from project parameter **`FDMNES_calc`**, specified (see Section 1.10)

2.3 Generate input files for supplementary XANES training set (compare different machine learning methods)

```
folderCompare = 'sample_compareMethods'
generateInputFiles(project.geometryParamRanges, project.moleculeConstructor,
sampleCount=200,
method='line', spectralProgram='fdmnes', spectrCalcParams =
project.FDMNES_calc,
folder=folderCompare,
lineEdges={'start':{'centralRing1_Shift': 0, 'sideRings1_Shift': 0,
'sideRings1_Elong': 0,
'centralRing2_Shift': 0, 'sideRings2_Shift': 0, 'sideRings2_Elong': 0},
'end':{'centralRing1_Shift': -0.3, 'sideRings1_Shift': 0.5,
'sideRings1_Elong': 0.5,
'centralRing2_Shift': 0.5, 'sideRings2_Shift': -0.3, 'sideRings2_Elong': -
0.3}})
```

2.4 Save this file as python script and execute remotely on cluster.

```
saveAsScript('sample.py')
```

Calculations can be also executed remotely. To proceed, user should generate a script file from this notebook by the execution of the command **`saveAsScript`** or by saving it manually from the main menu through the command File→Download as → Python (.py)

Install PyFitIt on you cluster using instructions on the website hpc.nano.sfedu.ru/pyfitit

Copy project file, generated python script, file with experiment to a folder at remote cluster.

Before executing script specify following parameters:

- In function **`calcSpectra`** change **`runType='local'`** to **`runType='run-cluster'`** if you use cluster Blokhin. For example, if you want to run in parallel calculations for 10 structures and parallelize each calculation on 6 cores you should specify:
  `calcSpectra(spectralProgram='fdmnes', runType='run-cluster', nProcs=6, memory=5000, calcSampleInParallel=10, folder=folder)`
- in function **`calcSpectra`** change **`runType='local'`** to **`runType='user defined'`** and add to this function parameter **`runCmd='command'`** where **`command`** runs fdmnes on your cluster. For example, if your cluster uses slurm and you want to run in parallel calculations for 10 structures and parallelize each calculation on 4 cores you should specify:
  `calcSpectra(spectralProgram='fdmnes', runType='user defined', runCmd = 'srun -n 4 fdmnes' calcSampleInParallel=10, folder=folder)`

Command **`sample.py`** will work for a long time. Therefore, it should be executed in background or by using manager Tmux. To run in background, you may use command:

```
python sample.py  >  log.txt  2>&1  &
```

(instead of name sample you should specify your name of script)

To run program in Tmux manager user should start this manager by command "*tmux*" or attach to the existing tmux session "*tmux a*" and execute **`python sample.py`** (for example in Blokhin cluster you should use **`/opt/anaconda/bin/python sample.py`**)

2.5 Attention. Start xanes calculation on local computer (can be too long)

```
calcSpectra(spectralProgram='fdmnes', runType='local', calcSampleInParallel=4, folder=folder)
calcSpectra(spectralProgram='fdmnes', runType='local', calcSampleInParallel=4, folder=folderCompare)
```

Function **`calcSpectra`** starts sequential or parallel calculation of spectra for generated input files.

2.6 Collect results into two files: params.txt and xanes.txt

```
collectResults(folder=folder, outputFolder=folder+'_result')
collectResults(folder=folderCompare, outputFolder=folderCompare+'_result')
```

The function **`collectResults`** creates two files from the separate calculations. These are composed by a first file characterized by all theoretical spectra and a second file with the corresponding structural parameters.

3. Inverse approach

3.1 Import project file and experiment

```
project = loadProject('FeterpyProject.py', expFile = 'exp_excited.txt')
```

3.2 Import training set

```
sample = readSample('IHS_729')
sampleCompare = readSample('line_200')
```

Specify the folders inside project folder where files params.txt and spectra.txt are located (these files were created in Section 2.6)

3.3 Compare different machine learning methods

This is optional step. If supplementary training set was also calculated (see Section 2.5) then quality of available machine learning methods is compared on the basis of supplementary points, which represent a line in the space of structural parameters.

```
compareDifferentInverseMethods(sampleTrain = sample, sampleTest =
sampleCompare,
energyPoint=7143, geometryParam='centralRing1_Shift',
project=project, folderToSaveResult = 'results/inverseMethodsCompare')
```

this function trains ML algorithm on the basis of training sample (**`sample`**) and predicts spectrum for given set of points (**`sampleCompare`**). Choose one energy point (**`energyPoint`**) of spectrum. Chose any **`geometryParam`** (specified in Section 1.3) which will be shown on the x-axis to specify the steps along the line. Results will be written to the folder specified in variable **`folderToSaveResult`**.

Two graphs appear which show exact values, calculated by FDMNES along the line and predicted by different ML methods. Second graph shows absolute values of errors for different methods relative to the exact calculation.

3.4 Construct the inverse estimator

```
inverseEstimator = constructInverseEstimator('Extra Trees', project,
project.FDMNES_smooth,
CVcount=4, folderToSaveCVresult='results/inverseEstimator_CVresult')
```

`CVcount` – the parameter to specify parts of training set used for cross validation. 4 means that ¼ of whole training set will be used for cross-validation and ¾ - for training. The cross-validation will be repeated 4 times – for all different 4 parts of training set and results will be averaged.

3.5 Train the estimator

```
inverseEstimator.fit(sample)
```

3.6 Start automatic fit

```
inverseMethod.findGlobalL2NormMinimum(1, inverseEstimator, folderToSaveResult
= 'results/inverseMethodResults')
```

the first parameter specifies number of minimums to search. The minimums are obtained in random order and then sorted. Note that for RBF method the search procedure can be long.

4. Direct approach

4.1 Importing experiment data

```
project = loadProject('FeterpyProject_combined.py', expFile =
'exp_excited.txt')
```

we use combined training set here, where we grouped 6 different parameters of Feterpy molecule into 3 to avoid ambiguity in prediction.

4.2 Importing XANES calculated for support points

```
sample = readSample('IHS_combined')
```

4.3 Construct the direct estimator

```
directEstimator = constructDirectEstimator('RBF', project,
project.FDMNES_smooth, CVcount=4)
```

Specify given ML method: `'Ridge'`, `'Ridge Quadric'`, `'Extra Trees'`, `'RBF'`, `'LightGBM'`, `'NN'`

4.4 Train the estimator

```
directEstimator.fit(sample)
```

4.5 Predict the geometry for experimetal data

```
directEstimator.predict(project.spectrum,        folderToSaveResult          =
'results/directMethodResults', smooth=False)
```

4.6 Predict Radial Distribution

```
m = project.moleculeConstructor({'centralRings_Shift': 0.25,
'sideRings_Elong': 0.17, 'sideRings_Shift': 0})
```

specify the parameters of the molecule to display along with predicted structural parameters.

```
directEstimator.predictRDF(project.spectrum, folderToSaveResult =
'results/directMethodResults',
atoms = [1, 30, 49, 20], smooth=False, check=False,
extraMolecules={'best fit':m})
```

`atoms` are the numbers of atoms in xyz file which positions will be predicted
`smooth=False`, **`check=False`** are set because we work with experiment which should not be additionally convoluted.
`extraMolecules` is used to display RDF for a specified by user above (for visualization only)

`predictRDF` predicts sorted distances to specified atoms (i.e. predicted four values are for atoms 1,30,49,20, but not exactly in the same order as initially specified if distances changed relative to their initial values) and plots distribution function as sum of gaussian with sigma equal to root mean square error calculated by cross-validation.

4.7 Predict Angular Distribution

```
directEstimator.predictAngleDF(project.spectrum, folderToSaveResult =
'results/directMethodResults',
atoms = [1, 30, 49, 20], smooth=False, angleCount=50, sigma=10, check=False,
extraMolecules={'best fit':m})
```

this method predicts directly angular distribution function in each point. I.e. for each structure in training set the angular distribution function is calculated using parameters - **`angleCount`** (number of points to approximate distribution function) and **`sigma`** (parameter of the Gaussian to model the distribution function width). Then algorithm is trained to predict each point of distribution function independently using all points of XANES spectrum.

For smaller sigma user needs to set larger value of **`angleCount`** (calculation time is proportional to **`angleCount`**)

5. Fitting XANES by sliders

5.1 Importing experiment data

```
project = loadProject('FeterpyProject.py', expFile = 'exp_excited.txt')
specify the name of project file and file with experiment
```

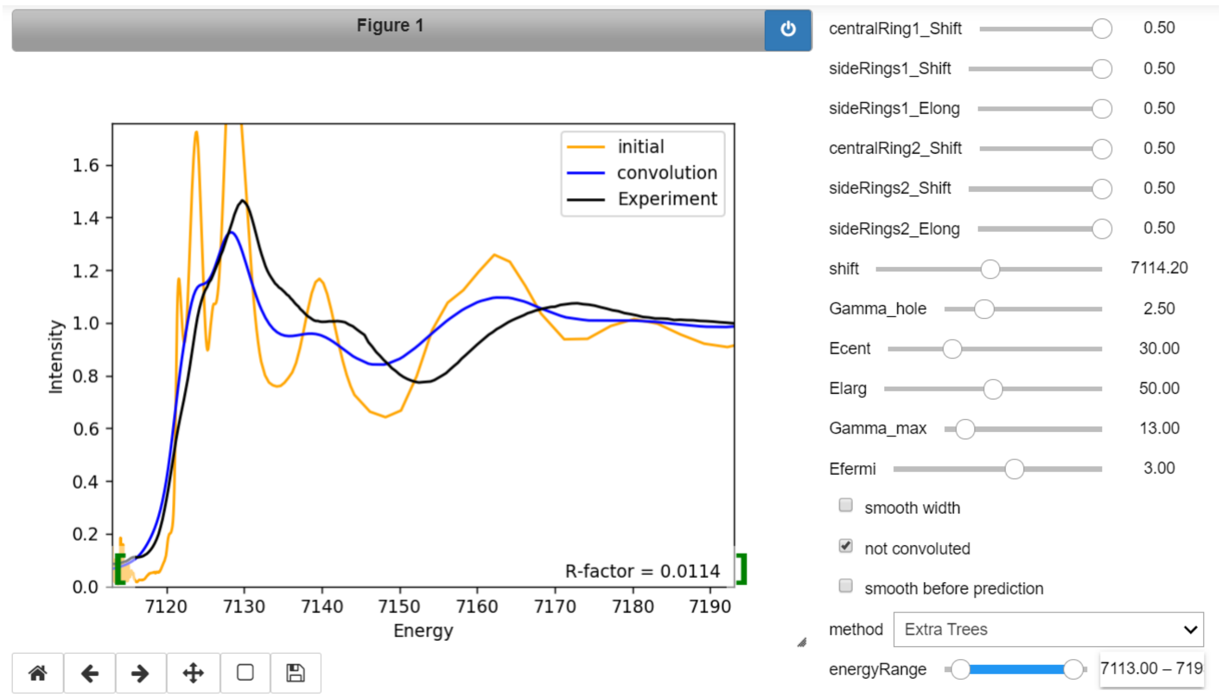
5.2 Importing XANES calculated for training set

```
sample = readSample('IHS_729')
```

specify folder with calculated training set – files params.txt (contains all parameters of deformations) and spectra.txt (contains spectra for each deformation)

5.3 Fitting XANES by sliders

```
fitResult = fitBySliders(sample, project)
```



centralRing1_Shift 0.50
sideRings1_Shift 0.50
sideRings1_Elong 0.50
centralRing2_Shift 0.50
sideRings2_Shift 0.50
sideRings2_Elong 0.50
shift 7114.20
Gamma_hole 2.50
Ecent 30.00
Elarg 50.00
Gamma_max 13.00
Efermi 3.00
☐ smooth width
☑ not convoluted
☐ smooth before prediction
method Extra Trees
energyRange 7113.00 – 719

For slow methods – RBF or lightGBM set option “**`smooth before prediction`**”. This will imply approximation of spectra only on the number of points specified in parameter **`project.maxSpectrumPoints = 100`** (see section 1.6). Otherwise all points calculated by FDMNES will be used.

To see changes of spectra drag slider to the specified position and drop it or specify a required number on the right hand side from slider.

5.4 Save results to files