FitWidget

March 9, 2017

1 FitWidget

FitWidget is a widget to fit curves (1D data) with interactive configuration options, to set constraints, adjust initial estimate parameters...

1.1 Creating a FitWidget

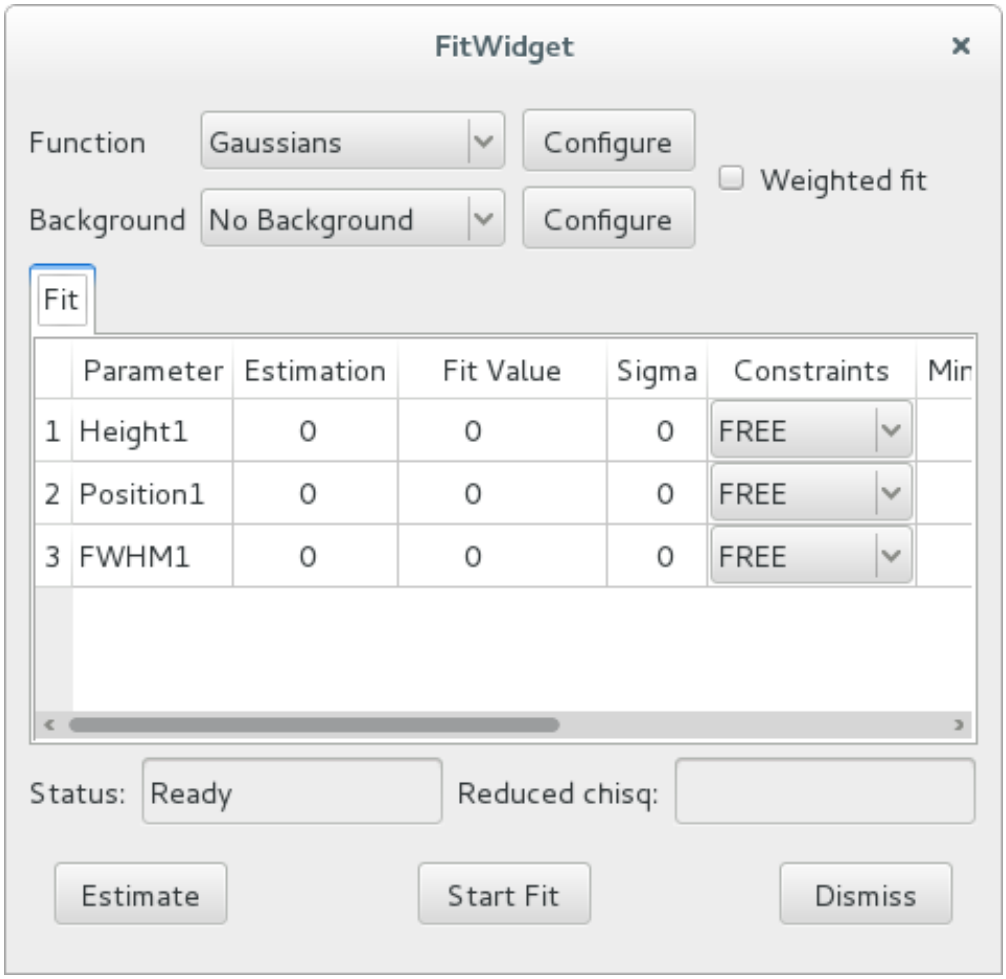
First load the data.

```
In [ ]: # opening qt widgets in a Jupyter notebook

        %gui qt

        # in a regular terminal, run the following 2 lines:

        # from silx.gui import qt
        # app = qt.QApplication([])
```

```
In [ ]: #%pylab inline

        import silx.io

        specfile = silx.io.open("data/31oct98.dat")
        xdata = specfile["/22.1/measurement/TZ3"]
        ydata = specfile["/22.1/measurement/If4"]

        from silx.gui.plot import Plot1D
        plot=Plot1D()
        plot.addCurve(x=xdata, y=ydata)
        plot.show()
```

Then create a FitWidget.

```
In [ ]: from silx.gui.fit import FitWidget

        fw = FitWidget()
        fw.setData(x=xdata, y=ydata)
        fw.show()
```



The selection of fit theories and background theories can be done through the interface. Additional configuration parameters can be set in a dialog, by clicking the configure button, to alter the behavior of the estimation function (peak search parameters) or to set global constraints.

When the configuration is done, click the Estimate button. Now you may change individual constraints or adjust initial estimated parameters.

You can also add peaks by selecting *Add* in the dropdown list in the *Constraints* column of any parameter, or reduce the number of peaks by selecting *Ignore*.

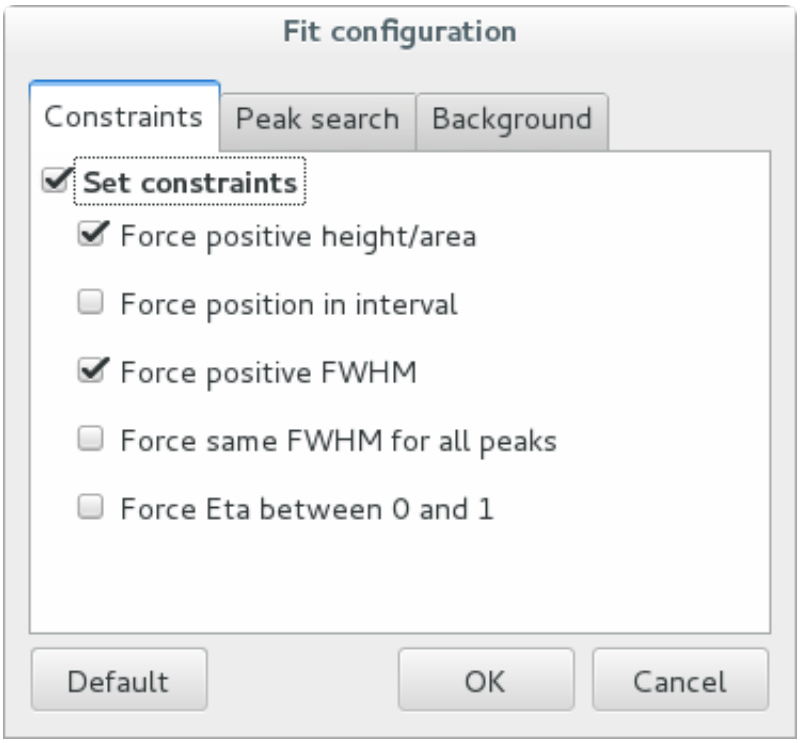
When you are happy with the estimated parameters and the constraints, you can click the "Fit" button. At the end of the fit process, you can again adjust the constraints and estimated parameters, and fit again. Only click "Estimate" if you want to reset the estimation and all constraints (this will overwrite all adjustements you have done).

1.2 Open the FitWidget through a PlotWindow

Rather than instantiating your own FitWidget and loading the data into it, you can just select a curve and click the fit icon inside a PlotWindow or a Plot1D widget.

A Plot1D always has the fit icon available, but for a PlotWindow you must specify an option `fit=True` when instantiating the widget.

```
In [ ]: from silx.gui.plot import PlotWindow
```



FitConfig

```
pw = PlotWindow(fit=True, control=True)
pw.addCurve(x=xdata, y=ydata)
pw.show()
```

A FitWidget opened through a PlotWindow is connected to the plot and will display the fit results in the PlotWindow, which is great for comparing the fit against the original data.

1.3 Exercice

Write a cubic polynomial function $y = ax^3 + bx^2 + cx + d$ and its corresponding estimation function (use $a = 1, b = 1, c = 1, d = 1$ for initial estimated parameters).

Generate synthetic data.

Create a FitWidget and add a cubic polynomial function to the dropdown list. Test it on the synthetic data.

1.3.1 Polynomial function

Tips: - Read the documentation for the module `silx.math.fit.fittheory` to use the correct signature for the polynomial and for the estimation functions. - Read the documentation for the module `silx.math.fit.leastsq` to use the correct format for constraints. Disable all constraints (set them to FREE)

Links: - http://pythonhosted.org/silx/modules/math/fit/fittheory.html#silx.math.fit.fittheory.FitTheory.fun
- http://pythonhosted.org/silx/modules/math/fit/fittheory.html#silx.math.fit.fittheory.FitTheory.estimate
- http://pythonhosted.org/silx/modules/math/fit/leastsq.html#silx.math.fit.leastsq

```
In [ ]: # fill-in the blanks
        def cubic_poly(x, ...):
            """y = a*x^3 + b*x^2 + c*x +d

            :param x: numpy array of abscissa data
            :return: numpy array of y values
            """
            return ...

        def estimate_cubic_params(...):
            initial_params = ...
            constraints = ...
            return initial_params, constraints
```

1.3.2 Synthetic data

Tip: use the `cubic_poly` function

```
In [ ]: import numpy
        x = numpy.linspace(0, 100, 250)
        a, b, c, d = 0.02, -2.51, 76.76, 329.14
        y = ...
```

1.3.3 FitWidget with custom function

Tips:

- you need to define a customized FitManager to initialize a FitWidget with custom functions

Doc:

- http://www.silx.org/doc/silx/dev/modules/math/fit/fitmanager.html#silx.math.fit.fitmanager.FitMana

```
In [ ]: %gui qt
        #from silx.gui import qt
        from silx.gui.fit import FitWidget
        from silx.math.fit import FitManager

        # Uncomment this line if not in a jupyter notebook
        # a = qt.QApplication([])

        ...

        fitwidget = FitWidget(...)
        fitwidget.setData(x=x, y=y)

        fitwidget.show()

In [ ]:
```