PDFgetX3 Documentation

Release 1.0-r2491-20130712

Pavol Juhás, Timur Davis, Christopher L. Farrow, Simon J. Billinge

July 12, 2013

CONTENTS

CHAPTER

ONE

INTRODUCTION

PDFgetX3 is a simple yet powerful program for converting X-ray powder diffraction data to atomic Pair Distribution Functions (PDFs). PDFgetX3 can be used in a batch mode to convert a series of data files without user intervention. PDFgetX3 can be also run in an interactive mode that allows to control process parameters and plot the PDFs and any intermediate results. Users can interactively tune the PDF processing parameters, visualize their effect on the results and adjust them to their optimum values. The PDFgetX3 software comes with a Python library of PDF processing functions, which can be easily used in custom Python scripts.

1.1 License notice

1.2 Authors

This code was written by members of the Billinge Group at Columbia University including:

Pavol Juhás, Timur Davis, Christopher Farrow, Simon Billinge.

1.3 Reference

If you use this program for a scientific research that leads to publication, we ask that you acknowledge use of the program by citing the following paper in your publication:

> P. Juhás and T. Davis, C. L. Farrow, S. J. L. Billinge PDFgetX3: A rapid and highly automatable program for processing powder diffraction data into total scattering pair distribution functions, *J. Appl. Cryst.* **46**, 560-566 (2013)

CHAPTER

TWO

INSTALLATION

2.1 Software requirements

PDFgetX3 has been written in Python programming language, therefore to use the software, you must have Python 2.6 or Python 2.7 installed. [1] In addition, the following third-party Python libraries are also required:

- distribute - tools for installing Python packages
- NumPy - library for scientific computing with Python
- matplotlib - Python plotting library
- IPython - enhanced interactive Python shell
- PyReadline - required for a full IPython functionality on Windows

Standard Python releases can be obtained from http://www.python.org/download/. The third-party libraries can be found at the Python Package Index or using any Internet search engine.

Another, more convenient option is to obtain science-oriented Python distributions, such as PythonXY or Enthought Canopy. These distributions already include all the necessary libraries, so the required Python software can be all installed in one step.

On Linux operating systems the third-party libraries are usually included in a system software package repository. For example on an Ubuntu Linux computer the software dependencies can be all installed with a single shell command

```
sudo apt-get install \
  python-distribute python-numpy python-matplotlib ipython
```

This may be, of course, just as well accomplished using the GUI driven Synaptic package manager. Other Linux distributions may use different software management tools, but the names of the necessary packages should be very similar to those above.

On Windows operating system, it may be necessary to add the `C:\Python27` directory and the scripts directory `C:\Python27\Scripts` to the system `PATH`. Some Python distributions may already do that as a part of their installation process. The easiest way to check is to start the **Command Prompt**, type there `python` and see if this starts the Python interpreter.

2.2 PDFgetX3 installation

PDFgetX3 is distributed as a Python egg package, which can be obtained from the Columbia Flintbox. Once all the required software is in place, start the command prompt on Windows or a Unix terminal on Linux or Mac, navigate to

[1] Python 3 is not supported yet.

the directory that contains the egg file and execute the following command:

```
easy_install diffpy.pdfgetx-VERSION-pyX.Y.egg
```

Here VERSION needs to be replaced to match the actual filename, and X.Y is the Python version, for example 2.7. It is critical to use the correct egg file for your Python version as the program would not work otherwise. On Linux and Mac operating systems the installation may need to run with root user privileges, for example, by prepending `sudo` to the command line above. If root access is not available, use the easy_install options *`--prefix`* or *`--install-dir`* to install PDFgetX3 to a user-writable directory.

To verify if pdfgetx3 has been correctly installed, type the following command:

```
pdfgetx3 --version
```

This should display the software version, which should be the same as the VERSION string in the egg file name. The installation also includes a **plotdata** command for an easy plotting of text data files. To verify if plotdata works, run the `plotdata --version` command.

2.3 IPython magic command

These instructions are intended for IPython users who would like to integrate PDFgetX3 into their IPython environment. If you don't plan to use IPython, you can safely ignore this section.

When pdfgetx3 is run in an interactive mode, it starts IPython interactive shell and defines an extra `%pdfgetx3` magic command within the IPython session. The IPython magic commands are not valid Python code, but work in a similar fashion as standard shell commands. The `%pdfgetx3` magic can be thus used with the same options and arguments as if run from the shell. This is useful for processing more files, while preserving all plots or variables that were already created within the IPython session.

The `%pdfgetx3` magic command can be defined permanently so it is available in any IPython sessions. To make the `%pdfgetx3` command permanent, first check what is the IPython version. If it is 0.11 or later,

1. find the `profile_default/ipython_config.py` file and open it in a text editor. If that file does not exists, it can be generated by executing

   ```
   ipython profile create
   ```

   in a system shell.

2. navigate to the paragraph that contains the `c.InteractiveShellApp.extensions` and add there the following line:

   ```
   c.InteractiveShellApp.extensions = ['diffpy.pdfgetx.ipy_pdfgetx3']
   ```

   There must be no leading indent, i.e., the text must start at the very first column.

The magic commands can be also defined in an older IPython 0.10, but the procedure is slightly different. The trick is to find the `HOME/.ipython/ipy_user_conf.py` file and add there the following statement

```
import diffpy.pdfgetx.ipy_pdfgetx3
```

Make sure this line has a correct indentation, i.e., it has the same amount of leading whitespace, as do the surrounding lines.

CHAPTER

THREE

QUICK-START GUIDE

This guide assumes that the PDFgetX3 program has been correctly installed and can be executed by typing **pdfgetx3** in a shell window. Please, refer to the *installation* section if this is not working yet.

The pdfgetx3 program is a command-line application, therefore all the input files and run-parameters are supplied either as command-line arguments or through a configuration file. In general, the pdfgetx3 is executed from a command shell as

```
pdfgetx3 [options] input1 input2 ... inputN
```

The `inputN` stands for an input powder diffraction data. The `inputN` file is a simple two-column text file, where the first column corresponds to either the 2Θ diffraction angle, or a momentum transfer, Q, in inverse nanometer or inverse ångström units. The second column contains the corresponding X-ray intensities. The input file may start with a header containing comments or metadata related to the measurement. PDFgetX3 will ignore any text leading to a long two-column section. The example input files in this manual were created with the FIT2D program using its "chi" output format, thus we will also refer to them as "chi-files".

The command-line options are arguments that start with a dash "-" and are used to specify run-parameters or modify the program behavior. The options can be specified in a short form that consists of a dash and a single character, or in a long, more descriptive format starting with a doubled dash `--`. Options may require values. For short options, the value may be joined to the option string, for example *`-w0.142774`*, while for the long options it has to be separated with an equal sign, e.g., *`--wavelength=0.142774`*. Although all the PDF calculation parameters can be passed as command line options, it is often more convenient to set them in a configuration file. When run parameter is present both in a configuration file and as command-line option, the command-line value takes precedence. The command-line options are all described in the *Options and parameters* section of this manual. A brief summary of options can be also displayed by executing

```
pdfgetx3 --help
```

The best way of getting familiar with PDFgetX3 is to process the example diffraction data described in the *Tutorial*. In general, the first step is to create a commented configuration file `pdfgetx3.cfg` using:

```
pdfgetx3 --createconfig=pdfgetx3.cfg
```

The configuration file can have any name, but it is preferable to use either `pdfgetx3.cfg` or `.pdfgetx3.cfg`, for these files are automatically loaded by PDFgetX3. All other configuration files must be passed explicitly to the program using the *`-c, --config`* option.

Open the pdfgetx3.cfg file in a **text** editor. The lines that start with a hash mark `#` are comments and are not used. The lines starting with a right brace `[` denote sections in the configuration file. The active lines are all formatted as "NAME=VALUE". Although PDFgetX3 has many options, in general only a few of them are critical for the PDF calculation:

- `dataformat` – specifies the input data format

- `wavelength` – radiation wavelength in Å required for the twotheta format.
- `composition` – chemical composition of the sample
- `qmaxinst` – upper Q boundary for a meaningful measurement intensities.
- `qmax` – Q-cutoff for the Fourier transformation that yields the PDF.

Save the updated configuration file and run pdfgetx3 on the input data `FILENAME.chi` as

```
pdfgetx3 --verbose=info -t gr FILENAME.chi
```

Here the *`--verbose=info`* option makes pdfgetx3 print more information about its operation. This helps to verify if the configuration file is indeed loaded and if the parameter values are assigned as intended. The PDFgetX3 will not write any output files unless told so. The *`-t gr`* option tells the program to save the final $G(r)$ curve as a `FILENAME.gr` file in the working directory.

The saved .gr file contains a header with all the calculation parameters and the input file name. The .gr file can be therefore also used as a configuration file in order to redo the same calculation

```
pdfgetx3 -c FILENAME.gr --plot=fq,gr
```

Note this command does not include any .chi file and this will as a result process the previously used input `FILENAME.chi`. The *`--plot=fq,gr`* option tells PDFgetX3 to display 2 plots for the reduced structure function $F(Q)$ and the final PDF $G(r)$. The *`--plot`* option also implies an *interactive mode* therefore the program does not exit, but starts an interactive IPython session. To exit the interactive mode, type `exit()` and press Enter.

Please refer to the *tutorial section* for a step-by step processing of the example files and for demonstration of the PDFgetX3 functions.

CHAPTER

FOUR

TUTORIAL

In this tutorial we will convert several X-ray powder diffraction patterns to corresponding PDFs. Open a terminal on a Unix-based system or a Command Prompt on Windows and navigate to the `examples` folder included with the PDFgetX3 distribution. The `examples` folder can be found in the parent "doc" directory relative to this document or another option is to just search your file system for one of the input files mentioned below. The example files are also available at http://www.diffpy.org/doc/pdfgetx3/pdfgetx3-examples.zip.

4.1 Nickel X-ray PDF

4.1.1 predefined configuration file

Change to the `Ni` directory. The file named `ni300mesh_300k_nor_1-5.chi` contains powder X-ray data measured from nickel at the Advanced Photon Source beamline 6ID-D. The file contains two columns for the 2Θ scattering angles and X-ray intensities. The second file `kapton_bgrd_300k_nor_2-3.chi` contains the background measurement, i.e., the intensities from an empty capillary. Finally, the `pdfgetx3.cfg` contains a complete configuration parameters for converting the powder pattern to a PDF. Since all processing parameters are already defined in the configuration file, the first PDF calculation is very simple and involves running the **pdfgetx3** program with the powder data file as an argument:

```
$ pdfgetx3 ni300mesh_300k_nor_1-5.chi
```

For the first run there should be no output on the screen, however a new file, `ni300mesh_300k_nor_1-5.gr` should appear in the work directory. We can use the **plotdata** program, installed with PDFgetX3, to plot the output data:

```
$ plotdata ni300mesh_300k_nor_1-5.gr
```

This will open a graph window and start an IPython interactive session. To exit and close the figure, type `exit()` on the IPython prompt. Let's run the program again, but now with a *`--verbose=info`* option, to show more details about the program actions.

```
$ pdfgetx3 --verbose=info ni300mesh_300k_nor_1-5.chi

INFO:log level set to 20
INFO:checking for environment variable PDFGETX3PATH
INFO:searching for default config file /home/user/.pdfgetx3.cfg
INFO:searching for default config file .pdfgetx3.cfg
INFO:searching for default config file pdfgetx3.cfg
INFO:loaded default config file pdfgetx3.cfg
INFO:parsing config file section [DEFAULT]
INFO:set config.rmin = 0.0
```

```
INFO:set config.rstep = 0.01
INFO:set config.rmax = 30.0
INFO:set config.qmax = 26.0
INFO:set config.outputtypes = ['gr']
INFO:set config.dataformat = 'twotheta'
INFO:set config.qmaxinst = 26.5
INFO:set config.backgroundfile = 'kapton_bgrd_300k_nor_2-3.chi'
INFO:set config.wavelength = 0.14277400000000001
INFO:set config.composition = 'Ni'
INFO:finished parsing config file
INFO:processing command line options
INFO:set config.verbose = 'info'
INFO:finished with command line options
INFO:using 1 input files from the command line.
INFO:configuring PDFGetter mode 'xray'
INFO:calling config_xray
INFO:started PDF processing.
INFO:processing 'ni300mesh_300k_nor_1-5.chi'
INFO:resolved output file '' as 'ni300mesh_300k_nor_1-5.gr'
WARNING:ni300mesh_300k_nor_1-5.gr already exists.
WARNING:Use "--force" to overwrite.
INFO:elapsed time: 0.13
```

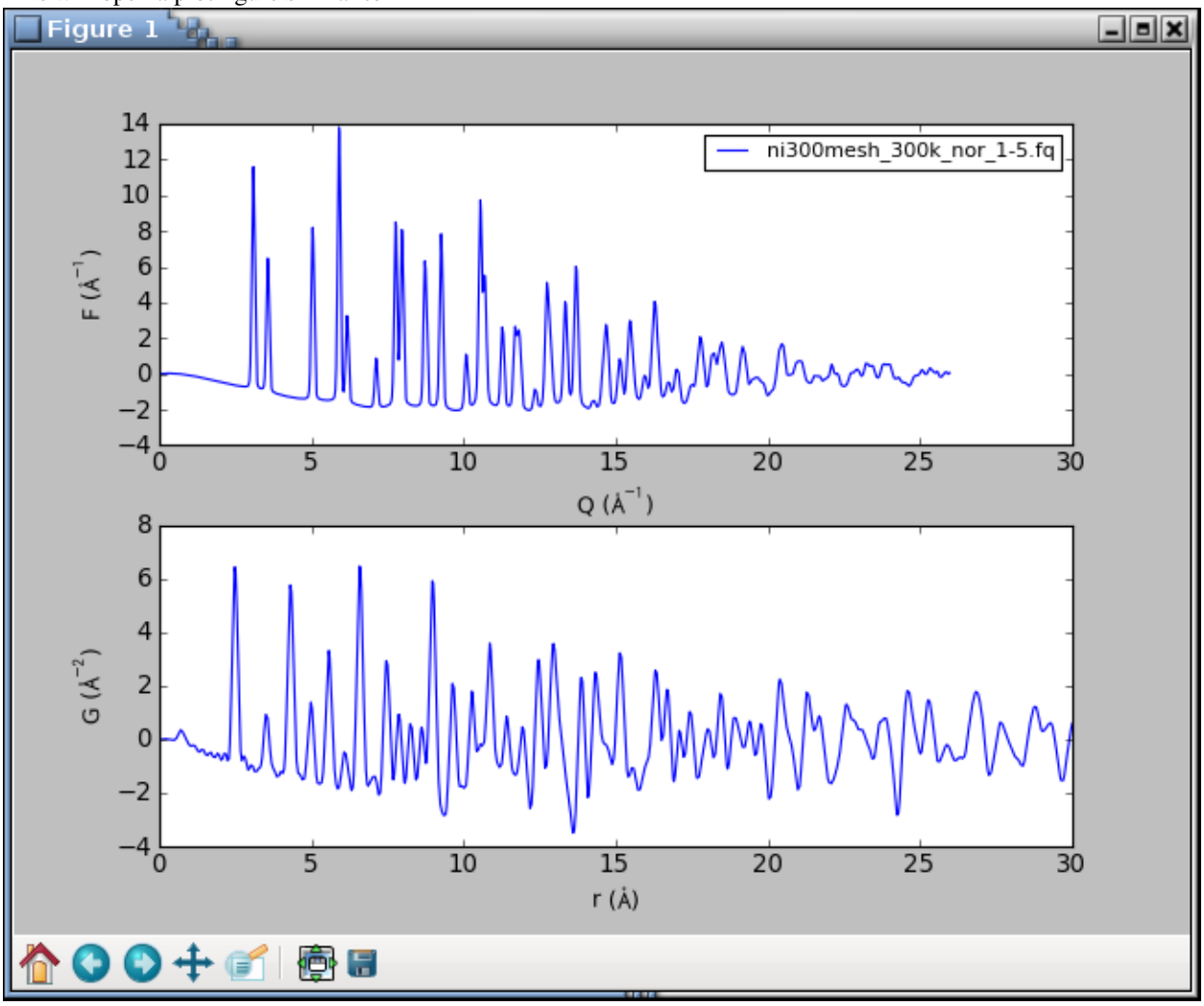
Here we can see what configuration files are searched, which of them get loaded and what are the effective values of the processing parameters. Unless the *--verbose* option is effect, the program shows only messages that have either WARNING or ERROR importance. The warning line above indicates no output has been written, because that file already exists. This safety check can be overruled with the *--force* option, upon which pdfgetx3 would overwrite any existing files.

PDFgetX3 output files start with a header that lists all the processing parameters and can be used as a valid configuration file with the *-c* option. Another option, *--plot=[iq,sq,fq,gr]* turns on plotting of the final PDF or of some other result. A side effect of the *--plot* option is that pdfgetx3 starts in an interactive mode, so the user can manipulate or save the plots. To put it all together, we are now going to redo the original PDF and plot its reduced total scattering function $F(Q)$ and the PDF curve $G(r)$. This time the chi file is not necessary, because the input file is already listed in the gr file that is now used as a custom configuration:

```
$ pdfgetx3 -c ni300mesh_300k_nor_1-5.gr --plot=fq,gr

Welcome to pylab, a matplotlib-based Python environment [backend: GTKAgg].
For more information, type 'help(pylab)'.
WARNING:ni300mesh_300k_nor_1-5.gr already exists.
WARNING:Use "--force" to overwrite.

Variables related to PDF processing:

pdfgetter    -- PDFGetter used for calculation.
config       -- configuration data used by PDFGetter.
                See config.inputfiles for a list of inputs.
iraw         -- matrix of input raw intensities, 2 rows per file
iq sq fq gr  -- intermediate results per each input file stored
                as matrix rows

Functions:

tuneconfig   -- dynamically tune configuration variable
processFiles -- process specified data files
clearSession -- clear all elements from the inputfiles, iraw,
              iq, sq, fq and gr variables.
```

```
plotdata      -- plot all or selected columns from a text data file
loadData      -- load all or selected columns from a text data file

Use "%pdfgetx3" for a fresh run without exiting IPython.
In [1]:
```

This will open a plot figure similar to



Because of the *interactive mode* implied by plotting, the program enters an IPython session. The IPython environment is preloaded with several extra functions and variables related to the PDF processing. For example, the `config` variable stores all the configuration parameters, and its content can be displayed with the `print` statement as

```
In [1]: print config

args = ['-c', 'ni300mesh_300k_nor_1-5.gr', '--plot=fq,gr']
configfile = ni300mesh_300k_nor_1-5.gr
...
qmax = 26.0
...
```

The `processFiles()` function allows to redo the whole calculation and plotting process for additional input files or for new parameter values. To plot the $F(Q)$ and $G(r)$ curves calculated at Q_{max} = 22 Å^{-1}, we can call `processFiles()` and pass it a keyword argument for the new qmax as follows:

```
In [2]: processFiles(qmax=22)

# the qmax parameter was updated to a new value, thus
In [3]: config.qmax
Out[3]: 22
```

There should be now two lines in each plot axis corresponding to the results at Q_{max} equal 26 and 22 Å^{-1}. To exit the program, type `exit()`.

4.1.2 processing from scratch

We have already encountered the command-line *option -c* for specifying a custom configuration file. A special argument "NONE", will make pdfgetx3 ignore any configuration files and start up in a default state. We can use this feature to process the nickel PDF as if we did not have any configuration file:

```
$ pdfgetx3 -c NONE ni300mesh_300k_nor_1-5.chi

WARNING:Nothing to do, use "-t" or "--plot" options.
ERROR:Configuration error: wavelength not specified.
ERROR:See "--help" for more hints.
```

There is an error, for the wavelength is necessary to convert the scattering angle 2Θ to momentum transfer Q. The X-ray wavelength was 0.142774 Å, which can be passed with the *-w, --wavelength* option:

```
$ pdfgetx3 -c NONE ni300mesh_300k_nor_1-5.chi -w 0.142774
...
ERROR:Configuration error: Chemical composition not known.
ERROR:See "--help" for more hints.
```

There is still an error. The PDF calculation needs an average X-ray scattering factor of the material, which is obtained from sample chemical composition. The composition can be specified with the *--composition* option. The example below uses a "\" character to indicate the command continues on the next line. Such syntax works in Unix terminals, but on Windows the command has to be typed all on a single line:

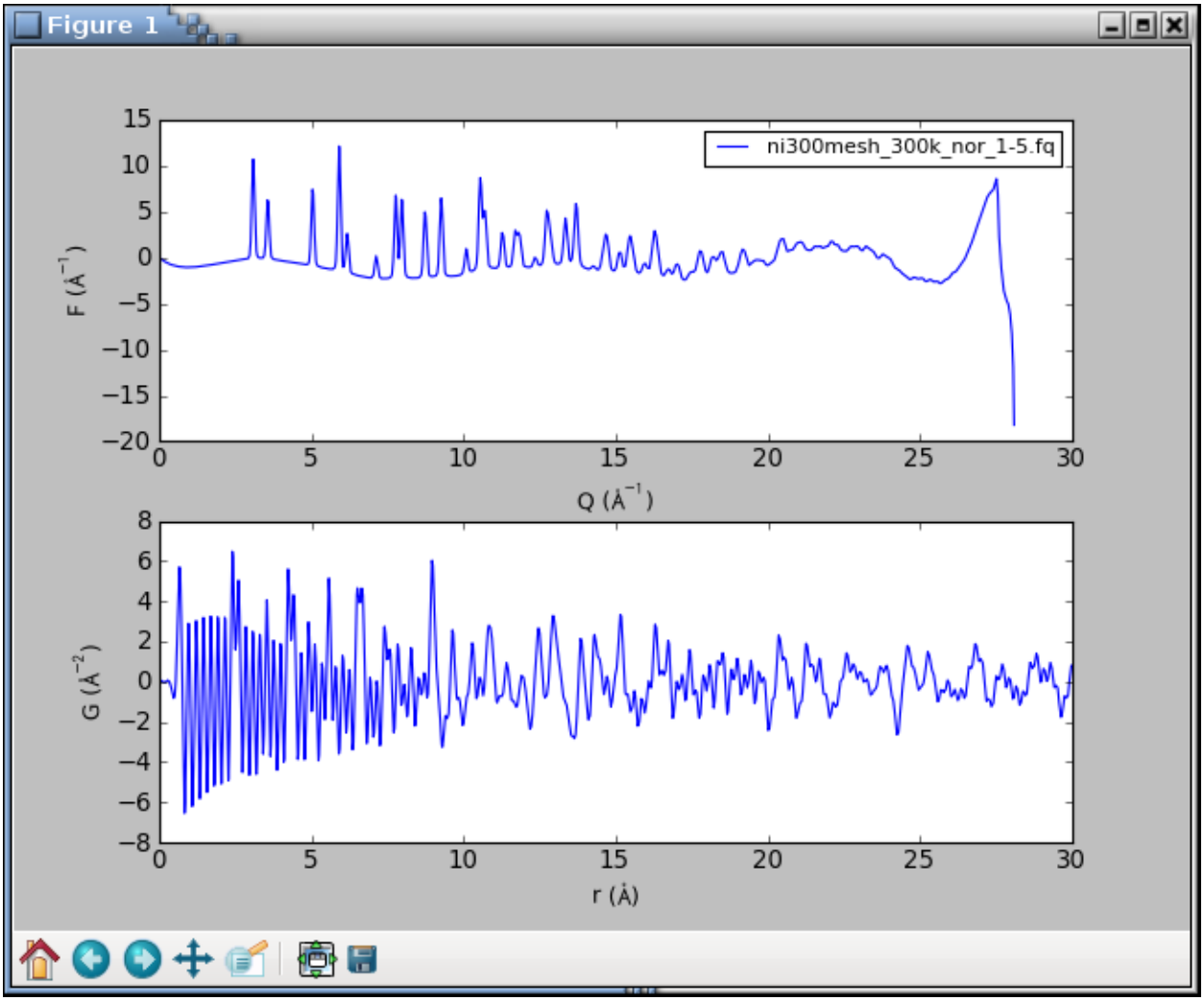
```
$ pdfgetx3 -c NONE ni300mesh_300k_nor_1-5.chi -w 0.142774 \
           --composition=Ni

WARNING:Nothing to do, use "-t" or "--plot" options.
```

There was no error message this time, but the program complains about a lack of action. The pdfgetx3 program does not write any results unless instructed by the *-t, --outputtypes* option. The outputtypes option recognizes the following result types: "iq", "sq", "fq", "gr". One or more of these type strings, separated by a comma, can be included with the *-t* option, which will produce the corresponding output files. An empty string, such as `-t ""`, or `-t NONE` may be used to clear any outputtypes defined in the configuration file, and avoid the unseemly file-exists warnings.

At this point, we will not write any output files, but will use the *--plot* option to display the calculated curves. The *--plot* accepts the same arguments as outputtypes, so to display the $F(Q)$ and $G(r)$ curves we shall run

```
$ pdfgetx3 -c NONE ni300mesh_300k_nor_1-5.chi -w 0.142774 \
           --composition=Ni --plot=fq,gr

WARNING:qmaxinst reset to last nonzero point qmaxinst=28.0865680161
WARNING:qmax reset to the data boundary qmaxinst=28.0865680161
```

which should open the following plot window:



The graphs look terrible. The PDF is very noisy and the $F(Q)$ curve shows a sudden break at about 27 Å^{-1}. What happened? The powder intensities are inaccurate at a very top of the detector angular range. The interactive session is setup with `iraw`, `iq`, `sq`, `fq`, `gr` variables for the original raw data and intermediate results. We are going to plot the "iq" variable that has the input intensities resampled on the Q grid. The `clf()` function clears the figure, the iq variable is a two-row matrix with Q and I rows, and the `axis()` function let's us zoom to a given range:

```
In [1]: clf()
In [2]: plot(iq[0], iq[1])
Out[2]: [<matplotlib.lines.Line2D at 0x3e20f50>]
In [3]: axis([20, 29, 0, 3000])
Out[3]: [20, 29, 0, 3000]
```

The graph shows a sudden drop in the raw intensities at 27 Å^{-1}. The `qmaxinst` variable defines a Q cutoff for a meaningful instrument intensities and, to be on a safe side, we are going to set it to 26.5 Å^{-1}

```
In [4]: processFiles(qmaxinst=26.5)
WARNING:qmax reset to the data boundary qmaxinst=26.5
```

The updated curves looks reasonable without any oscillations and breakpoints. The `tuneconfig()` function provides a GUI-driven way for visualizing the processing parameters and their effect on the results. Type `tuneconfig()` to execute the function, which should open a new window with several sliders. Try to move different sliders and see how do the $F(Q)$ and $G(r)$ curves change. The `rpoly` parameter controls the degree of data-correction polynomial and is an approximate low-r bound of reliable G values. Once the parameters are tuned, they may be set

to exact values. We will also turn on the writing of the $G(r)$ curve and save it to an output file `nicmd.gr`:

```
In [14]: config.qmax = 26
In [15]: config.outputtypes = 'gr'
In [16]: config.output = 'nicmd'
In [17]: processFiles()
```

4.2 Platinum data series

PDFgetX3 has been designed to handle large series of data files. With the fast area-detectors it is easy to measure hundreds of X-ray patterns in a time or temperature series. Normally, these input files need to be entered as command line arguments to the pdfgetx3 program. This is usually no problem with Unix-like shells, which expand filename patterns to a list of matching files. However, such file generation is in general not available on Windows. The input file names tend to include scan number that may run in thousands, and even Unix most shells do not have an easy facility (the z-shell being a notable exception) for matching a range of scan numbers.

4.2.1 matching input files

The **pdfgetx3** program includes a built-in function for finding a set of input files. The command line arguments are normally taken as input file names. However, if the *-f, --find* option is present, the arguments are understood as patterns and the program looks for files that match ALL of them. Another option *-l, --list* makes pdfgetx3 print out the matching files without any other action, which can be used to verify if the patterns match intended files.

We will try out this file search on platinum example files. Open a terminal and navigate to the `Pt` directory. There should be a `series` subdirectory with 6 chi files indexed from 903 to 908. At first, let's stay in the `Pt` directory and run the following command

```
$ pdfgetx3 --list --find

Pt_bulk-00055-pdfgetx2.gr
Pt_bulk-00055-pdfgetx3.gr
Pt_bulk-00055.chi
Pt_bulk-00055.gr
empty_capillary-00032.chi
pdfgetx3.cfg
plotpdfcomparison.py
```

Without any patterns the file search matches all files in the current directory. Now let's try to add name patterns. There are few special patterns, for example `^` matches at the beginning of the filename, `$` at the end and `<N-M>` matches a range of integer values from N to M. The patterns containing `"^$<>"` need to be quoted as these characters have special meaning in the shell. Here are some examples how it works.

Filenames containing "y":

```
$ pdfgetx3 --list --find y
empty_capillary-00032.chi
plotpdfcomparison.py
```

Filenames that containing both "y" and "chi", here we use the abbreviated forms *-l* and *-f*:

```
$ pdfgetx3 -lf y chi
empty_capillary-00032.chi
```

Filenames that start with "e":

```
$ pdfgetx3 --list --find "^e"
empty_capillary-00032.chi
```

Filenames that contain *character* "2":

```
$ pdfgetx3 --list --find 2
Pt_bulk-00055-pdfgetx2.gr
empty_capillary-00032.chi
```

Filenames that contain *number* "2":

```
$ pdfgetx3 -lf "<2>"
Pt_bulk-00055-pdfgetx2.gr
```

4.2.2 data search path

PDFgetX3 can be run with the *-d, --datapath* option, which tells it to search additional directories for input data files. The *-d* option can be used several times to search more directories. The data directories can be also defined with the PDFGETX3PATH environment variable. Here we will use the *-d* option to match files in the series subdirectory. The search stops at the first directory that contains any match, therefore

```
$ pdfgetx3 --datapath=series --list --find Pt chi
Pt_bulk-00055.chi
```

matches just one file in the current working directory, but

```
$ pdfgetx3 --datapath=series --list --find Pt "<906->.chi"
series/Pt_bulk_ramp03-00906.chi
series/Pt_bulk_ramp03-00907.chi
series/Pt_bulk_ramp03-00908.chi
```

finds 3 files, because only the series folder contains file names with "Pt" and a number "906" or higher followed by ".chi".

4.2.3 output file names

By default the output files are saved in the current directory. The output path, can be changed with the *-o, --output* option. The *-o* recognizes several aliases that are replaced with parts of the input file name, for example, "@b" expands to an extension-stripped base name. In similar faction, "@o" is replaced with the output type extension. Thus to generate PDFs for all files in the series directory and save them in the series-gr subfolder do

```
$ pdfgetx3 -d series --find "<900-910>.chi" --output=series-gr/@b.@o
```

The extension ".@o" is automatic when not included anywhere in the output file name. Thus to process the Pt series at Q_{max} = 18 Å^{-1} while saving the results in the same folder, but with "qmax18" in their filename can be done with:

```
$ pdfgetx3 -d series --find "<900-910>.chi" --qmax=18 -o series-gr/@b_qmax18
```

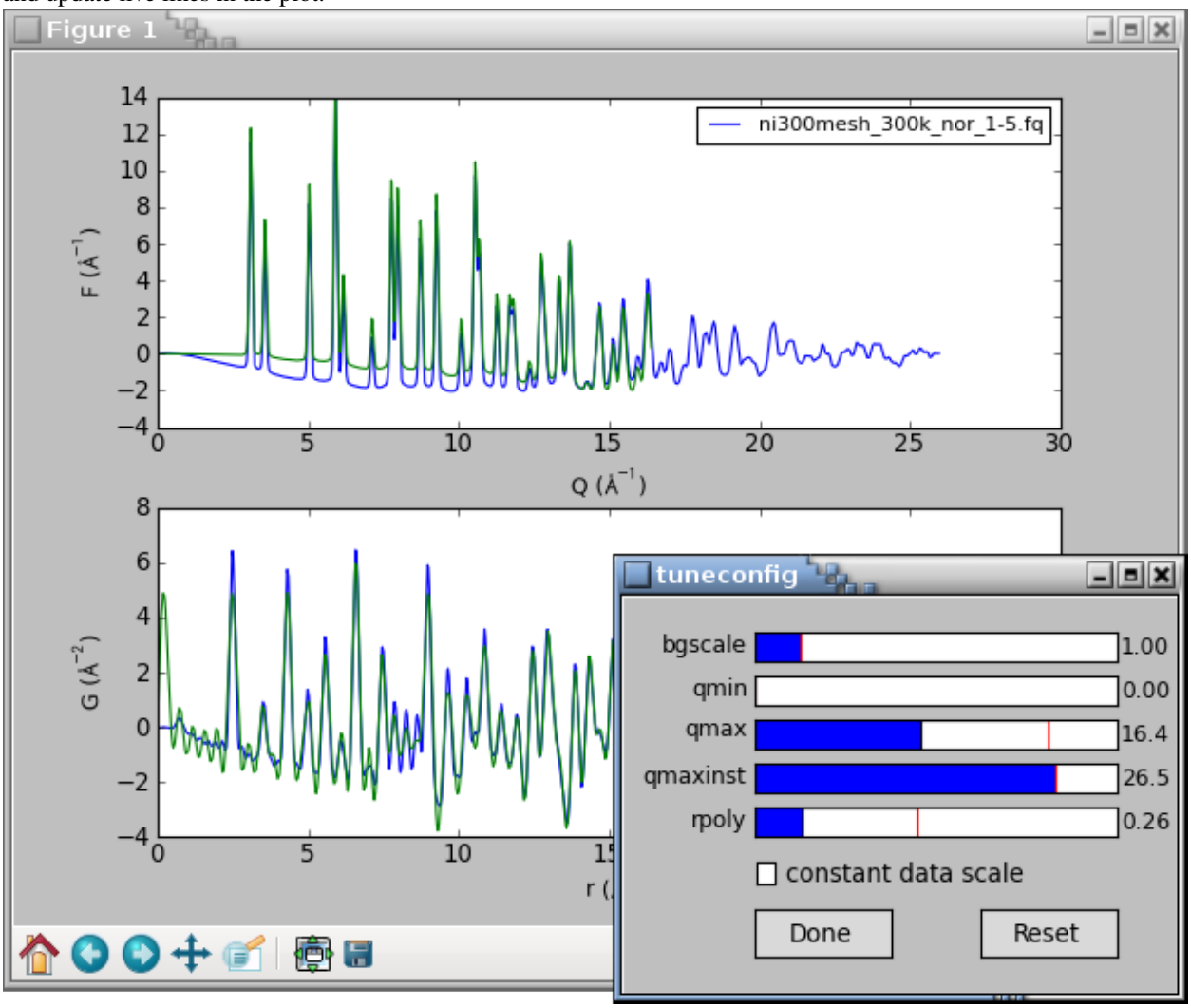
The series-gr directory should now contain 12 gr files, 6 of them processed at the Q_{max} = 27 Å^{-1} from the configuration file and 6 other at Q_{max} = 18 Å^{-1}.

4.3 Interactive tuning of parameters

One of the most powerful features of PDFgetX3 is the ability to tune PDF processing parameters in an interactive mode and immediately visualize their effect on the results. To demonstrate this feature, navigate to the `examples/Ni` directory in the shall and process the nickel PDF while plotting the $F(Q)$ and $G(r)$ curves. Because of plotting the program will open an interactive IPython session. The tuning mode can be then entered by calling the `tuneconfig()` function from the IPython environment

```
$ pdfgetx3 --plot=fq,gr ni300mesh_300k_nor_1-5.chi
...
In [1]: tuneconfig()
```

The `tuneconfig()` function will by default add a second set of live lines for the plotted curves and open a GUI dialog with sliders for the tunable process parameters. Changing any slider would immediately recalculate the PDF and update live lines in the plot.



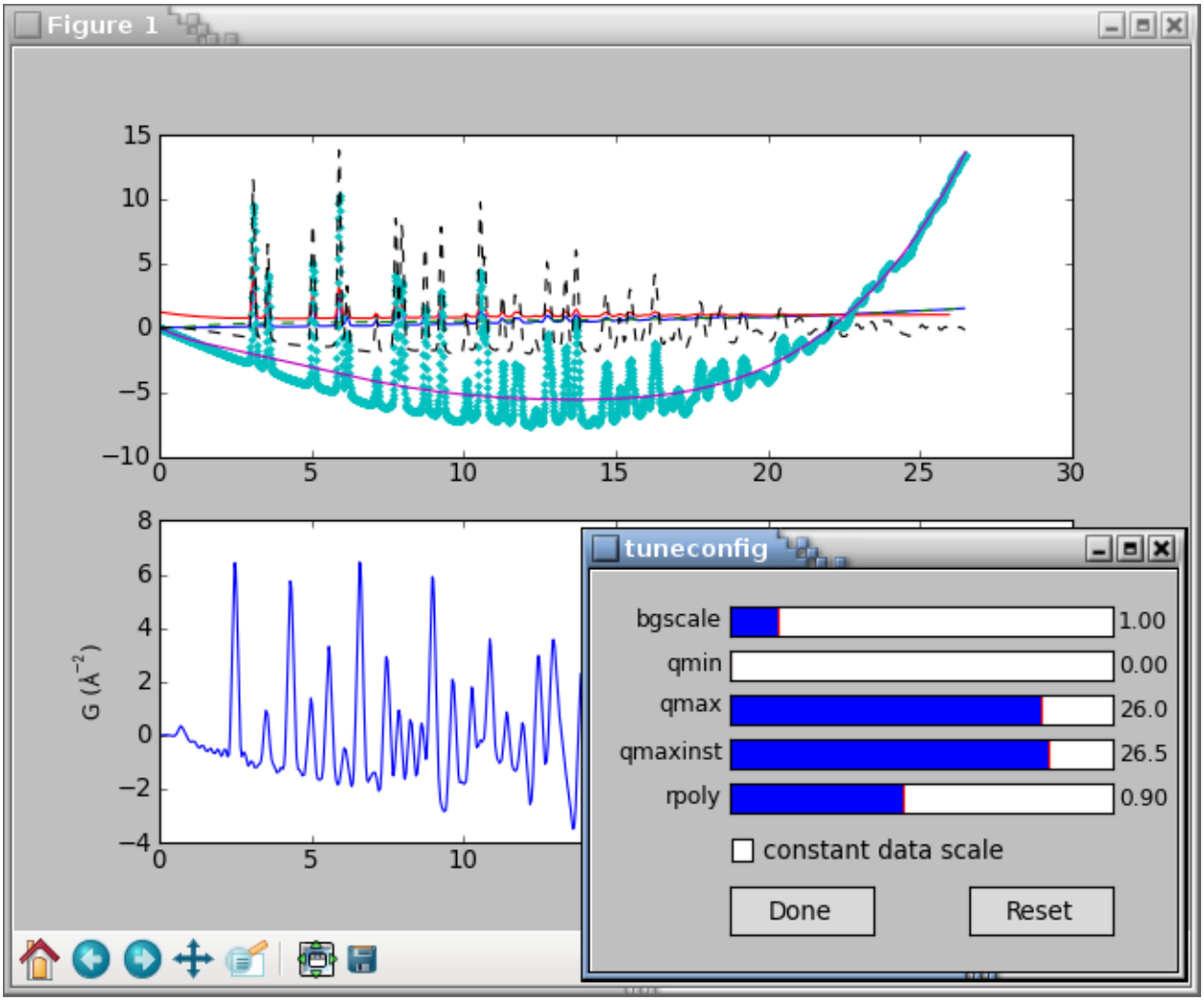
The *constant data scale* check-box rescales the result curves to a constant maximum value. This is useful for assessing if a parameter change produces different curve shape or if it just rescales the results. The tunable parameters are described in the *PDF parameters section*. Only the active parameters are displayed in the tuneconfig GUI, thus there would be no slider for the `bgscale` parameter if PDF has been processed without any background data.

By default the `tuneconfig()` function displays the same curves as specified by the `--plot` option, however it

can be configured to show arbitrary intermediate results or even visualize selected steps in the PDF processing. We shall demonstrate this by showing a live-plot of the polynomial correction together with the final PDF. At first, we shall use the `describe()` method of the `pdfgetter()` object to print out the chain of transformations involved in the PDF processing and obtain a reference to the transformation object *t4* that applies the polynomial correction. The transformation object can be then included in a list of plot identifiers that are passed to the `tuneconfig()` function

```
$ pdfgetx3 --plot=fq,gr ni300mesh_300k_nor_1-5.chi
...
Use "%pdfgetx3" for a fresh run without exiting IPython.

In [1]: clf()
In [2]: pdfgetter.describe()
0   TransformTwoThetaToQA
    convert x data from twotheta to Q in 1/A
1   TransformQGridRegular
    Remove the data outside the (qmin, qmaxinst) range
2   TransformBackground
    subtract background intensity
3   TransformXrayASFnormChris
    scale and normalize intensities by x-ray scattering factors
4   TransformSQnormRPoly
    Normalize S(Q) by fitting a polynomial
5   TransformSQToFQ
    Convert S(Q) to F(Q).
6   TransformFQgrid
    Resample F(Q) to a regular grid suitable for FFT
7   TransformFQToGr
    Convert F(Q) to G(r).
In [3]: t4 = pdfgetter.getTransformation(4)
In [4]: tuneconfig([t4, 'gr'])
```

The *clf()* function used above clears the figure to remove the initial $F(Q)$ line from the first panel. Overall, this should display the following plot:



The tuning can be finished by clicking the *Done* button or closing the tuneconfig GUI window. The parameter values can be thereafter adjusted to a rounded values by setting a corresponding attribute of the `config` object, for example:

```
config.bgscale = 1.5
```

See the `tuneconfig()` documentation for more details.

CHAPTER

FIVE

FILES USED BY PDFGETX3

5.1 Configuration file

Configuration files may define the PDF processing parameters. By default, the **pdfgetx3** program attempts to read ".pdfgetx3.cfg" file from the user HOME directory, then ".pdfgetx3.cfg" and "pdfgetx3.cfg" files from the current working directory. If configuration file has a different name, it needs to be specified with the *-c, --config* option.

The easiest way of creating a configuration file is to generate a template content using the *--createconfig* option as

```
pdfgetx3 --createconfig=test.cfg
```

and then change the generated test.cfg file in your favorite text editor. The configuration file follows a simple "var-name=value" syntax, any lines starting with "#" are ignored as comments.

The configuration file has several sections marked as `[SECTIONNAME]`. The `[DEFAULT]` section is mandatory and it contains the default global settings. Any other sections are optional and they are applied only when selected with the *-s, --section* option on the command line. Thus

```
pdfgetx3 --config=test.cfg --section=nacl
```

would read the parameters from the `[nacl]` section after reading the defaults. Having several sections in the configuration file is useful when there are multiple measurements that share most of the parameters, but differ in a few of them, for example in the chemical composition. The configuration file can then contain sections per each sample that define only the composition, while all other parameters are specified just once in the global DEFAULT section.

5.2 Input files

PDFgetX3 accepts input powder diffraction data in the form of two-column text file, where the first column x is either the scattering angle twotheta in degrees, momentum transfer Q in inverse nanometers or Q in inverse ångströms. The second column y contains the corresponding scattered intensities normalized per unit solid angle. The actual type of the x-values is identified by the `dataformat` parameter. The input files may contain header with comments or metadata, and the actual data are read from the first long section of numerical values.

The input files are usually passed as command-line arguments to the **pdfgetx3** program and must be paths accessible from the current working directory. The environment variable

PDFGETX3PATH
: specifies additional directories that are searched for input and background data files. The `PDFGETX3PATH` is a list of absolute or relative paths separated by **":"** on Linux and Mac or by **";"** on Windows, that are searched for input files, when these cannot be found in the current working directory.

The *-d, --datapath* option may be used to define additional data directories besides those in the PDFGETX3PATH.

When the *--find* option is active, the pdfgetx3 arguments are understood as filename patterns and the input files are searched in the current and datapath directories.

5.3 Output files

PDFgetX3 can produce up to four different output data files:

- .iq, $I(Q)$ – These are the background-corrected intensities sampled on a regular Q-space grid in inverse ångströms.
- .sq, $S(Q)$ – This file contains the total scattering structure function, with the intensities normalized by average scattering factors and corrected by a polynomial fit.
- .fq, $F(Q)$ – This file contains the reduced structure function equal to $Q(S(Q) - 1)$.
- .gr, $G(r)$ – this is the resultant PDF, where the first column is the separation r in ångströms and the second is the function G in $Å^{-2}$.

You can choose which files to output either in the configuration file via the outputtypes parameter, or at the command line with the *-t, --outputtypes* option.

The header of all output files contains the parameter values that were used in the calculation and thus it is by itself a valid configuration file. When passed as an argument to the *--config* option, the PDFgetX3 will exactly reproduce the previous calculation.

CHAPTER

SIX

INTERACTIVE MODE

The interactive mode is activated by using either the *-i, --interact* option or a non-empty *--plot* option. In the interactive mode the program starts an IPython interactive shell and pre-loads several functions and variables related to the PDF calculation. It also defines a `%pdfgetx3` macro, which can be used with a same command-line syntax as the **pdfgetx3** program from a system shell. The interactive session is also initialized with all functions from the matplotlib `pylab` module for convenient plotting. The functions and variables related to PDF processing are:

pdfgetter()
: Instance of the `PDFGetter` class which serves as a low-level function that calculates the PDF. This is a callable object, which takes as an argument a pair of input arrays for (two-theta, intensity). It can be also called with a keyword argument *filename=FILE*, which would read the input arrays from the specified file. When called with no arguments, it calculates PDF from the last input data.

  Returns A pair of output arrays *(r, G)*.

config
: Instance of the `PDFConfig` class, which stores all the program parameters and a list of input files. Use `print config` to show the current configuration values. This is the same object as `pdfgetter.config`. Configuration may be changed by assigning to a `config` object attribute, for example:

  ```
  config.qmax = 21
  ```

  The `config` values can be also changed by calling the `pdfgetter()` or `processFiles()` with a corresponding keyword argument, for example `processFiles(qmax=20, force=True)`.

iraw
iq
sq
fq
gr
: These variables are assigned the input raw intensities and the intermediate results, stored as matrix rows. The matrix rows correspond to twotheta1, intensity1, twotheta2, intensity2, etc. Because matrices are iterated row first, the raw intensities from all input files can be plotted with the matplotlib plot function as `plot(*iraw)`.

  These variables should be considered read-only and are reset with subsequent PDF calculations.

tuneconfig(*plotids=None*, *pdfgetter=None*, *axeslist=None*)
: Show a GUI dialog for interactive tuning of configuration variables.

  Parameters

  - **plotids** – string or iterable that specify what interactive plots should be tuned. By default the same as config.plot. Can be also an integer index or name of a transformation in pdfgetter or a reference to a `Transformation` object.

- **pdfgetter** – optional PDFGetter object to be tuned. This is by default the interactive `pdfgetter()` object.
- **axeslist** – optional list of matplotlib Axes for displaying the interactive plots. When None, use subplot(N, 1, i) to create the parent axes.

processFiles (*filename=None*, ***kwargs*)

Process all input files again with the current configuration values. This is a higher-level function than `pdfgetter()`, as it also saves output files and produces plots as specified by the `config` object.

Parameters

- **filename** – Input datafile or a list of several input files to be processed to PDFs and saved or plotted according to the `config` values.
- **kwargs** – optional keyword arguments that are applied to the `config` object, for example *(force=True, qmax=18)*.

This function also updates the `config.inputfiles` list and the `iraw`, `iq`, `sq`, `fq` and `gr` interactive variables.

clearSession()

Clear all elements from the `config.inputfiles` and also the `iraw`, `iq`, `sq`, `fq` and `gr` variables.

Returns No return value.

loadData (*filename*, *minrows=10*, ***kwargs*)

Load an array of floating point numbers from a text data file.

The data reading starts at the first matrix-like block of at least *minrows* rows and constant number of columns. This seems to work for most of text data files including those generated by PDFgetX2.

Parameters

- **filename** – File to be loaded.
- **minrows** – Minimum number of rows in the first data block. All rows must have the same number of floating point values.
- **usecols** – Indices or names of the columns to be loaded, by default all columns in a data block. Data blocks that do not contain sufficient number of columns are skipped. When *usecols* contains string items, these are looked up in the datafile header and translated to column indices. When *usecols* is a single string, it gets split to names at any comma or whitespace character.
- **unpack** – Return data as a sequence of columns that allows tuple unpacking such as *x, y = loadData(FILENAME, unpack=True)*. Note transposing the loaded array as loadData(FILENAME).T has the same effect.
- **kwargs** – Keyword arguments that are passed to the numpy.loadtxt function.

Returns Returns a numpy array of the loaded data.

plotdata (*filenames*, *style=None*, *x=None*, *y=None*, *hold=None*, ***kwargs*)

Plot one or more text data files.

Parameters

- **filenames** – Filename or a list of text data files to be plotted.
- **style** – Optional format string for the matplotlib plot function.
- **x,y** – Column or columns to be used for the x and y data. This is most often a single integer or an iterable of integer indices. It can be also a comma separated string of column names that are matched against column headers in the text data file. Also accepted is a string integer

or an integer range formatted as `START:STOP` or `START:STOP:STEP`. A special symbol `":"` can be used for a data-row index. Otherwise these arguments must be compatible with the *usecols* argument of the `loadData()` function. When not specified `plotdata()` uses the first column as x and the second column as y data.

- **hold** – Add new lines to the plot when *True*, replace the old lines when *False* or reuse the axes hold state if *None*.
- **kwargs** – Any other keyword arguments passed to the matplotlib plot function.

Returns This function returns a list of matplotlib Line2D objects.

findfiles(*patterns=()*, *path=None*)

Find files in the current directory that match all specified patterns.

Parameters

- **patterns** – String patterns that must all match in returned filenames. Can be a single string with patterns separated by whitespace characters. The patterns are matched either as fixed strings or as *special patterns*.
- **path** – Optional list of directories to be searched instead of the current directory. Can be also a string which is taken as a single directory path.

Returns Return a list of matching filenames. Return all files in the current directory when called without *patterns*.

CHAPTER

SEVEN

OPTIONS AND PARAMETERS

PDFgetX3 is very flexible in allowing users to customize the actions of the program. It has a number of parameters that can be specified either in configuration file or as a command line options. Here is a complete description of the parameters and options used by the program.

Note: The command line options start with a leading "-" and can be only used as command line arguments when starting the **pdfgetx3** program. Within configuration file the parameter names are plain words without any leading dashes. Finally, parameters can be also set in the interactive mode as attributes of the `config` object, but the assignments must be valid Python statements. Here are examples of setting composition of a processed specimen using each of these forms:

1. assigned in the configuration file `pdfgetx3.cfg`:

   ```
   ...
   composition = CaTiO3
   ...
   ```

2. set as a command-line option when starting **pdfgetx3**:

   ```
   pdfgetx3 --composition=CaTiO3
   ```

3. set in the IPython interactive mode:

   ```
   pdfgetx3 -i
   ...
   In [1]: config.composition = "CaTiO3"
   ```

7.1 Program operation

-h, -help
Display a brief usage information with a list of command line options and exit.

-V, -version
Display the program version and exit.

-manual
Open this manual in a Web browser and exit.

-f, -find

Select input files that match all filename patterns. The command line arguments are normally taken as input files. However, with the *--find* option the arguments are understood as filename patterns and the matching files are all used as inputs. The input files are searched in the current and `datapath`

directories. The file search stops at the first directory that contains any matching files. The search patterns are interpreted as fixed strings all of which must be present in the matching file name. The syntax supports several special patterns:

^	match at the beginning of the string, i.e., ^*start* matches only filenames that start with *start*.
$	match the end of string, for example, *.chi$* matches file names that end with *.chi*.
<N>	match number N preceded by any number of leading zeros, e.g., <7> would match in f7.chi, f007.chi, but not in f77.chi
<N-M>	match an integer range from N to M inclusive. The matched number may have one or more leading zeros.
<7->	match number 7 or larger.
<-7>	match number 7 or smaller.
<->	match any integer number.

The *"^$<>"* characters are often special to the Unix or Windows command shells, therefore they need to be enclosed in double quotes (") when used on the command line.

See also `datapath`, `PDFGETX3PATH` and *tutorial examples*.

`-l, -list`
List all input files and exit. This is useful with the *`--find`* option to verify if the input files are matched as intended.

7.2 Configuration file options

`-c` `CONFIG`**`, -config`**`=CONFIG`
Read custom configuration file after loading the default ones. Do not load any configuration file when *NONE*.

`-s` `NAME`**`, -section`**`=NAME`
Load the custom configuration file section `[SectionName]` after loading the `[DEFAULT]` section. This is useful for creating several configuration variants in a single configuration file.

`-createconfig``=FILE`
Write template configuration to a new FILE and exit. Write to the standard output when FILE is "-".

See also the *configuration file* section for further details.

7.3 Input and output options

`inputfile`
This parameter allows to specify one or more input files in the configuration file, one file per line. The `inputfile` is only used if no input files were provided on the **pdfgetx3** command line.

`dataformat`

`-format``=FORMAT`
Format of input files. Available formats are: `twotheta`, `QA`, `Qnm` corresponding to a two-column text data where the first column is either the scattering angle 2Θ in degrees, Q in inverse Ångströms or Q in inverse nanometers.

`backgroundfile`

`-b` `FILE`**`, -background`**`=FILE`
Optional datafile with background intensities from an empty sample holder. It must be in the same dataformat as other input files.

datapath

-d DATAPATH, **-datapath**=DATAPATH
One or more extra directories to be searched for input or background data files. The default path is given by the PDFGETX3PATH environment variable. The *-d* option can be specified several times to add more directories, these are prepended in front of any default value. Within configuration file the datapath directories have to be listed each on a separate line.

A special value "NONE" (or "none") clears any previously defined paths and only the further paths, if any, would be searched for inputs.

output

-o OUTPUT, **-output**=OUTPUT
Output file name, write to the standard output when "-". The *-t, --outputtypes* option controls what results are being saved. Normally the OUTPUT is used as a custom basename for the output files. The OUTPUT may contain @f, @h, @r, @e, @t, @b, @o tokens, which are expanded as follows:

token	example	definition
@h	dir1/dir2	the input file head directory or '.'
@r	dir1/dir2/filename	the input path with extension removed
@e	dat	the input file extension without '.'
@t	filename.dat	the tail component of the input file
@b	filename	the tail component with extension removed
@o	gr	the output extension iq, sq, fq or gr

An empty value works the same as *@b.@o* and saves the data in the current directory with a proper extension for the saved results. When *@o* is not present in OUTPUT, it would be appended as a default filename extension.

outputtypes

-t TYPES, **-outputtypes**=TYPES
Result types to be saved, one or more comma separated values. Supported values are "iq", "sq", "fq", "gr", corresponding to the $I(Q)$, $S(Q)$, $F(Q)$ and $G(r)$ curves; these are also used as output file extensions.

Result files are not written when empty, "none" or "NONE".

force

-force
Overwrite existing output files. By default the output files are not written if they already exist. Possible values in a configuration file are "true", "yes", "1" or "false", "no", "0". Note that in an interactive mode the values assigned to `config.force` must be Python booleans, e.g., `True`, `False`, `1`, `0`. The "yes" and "no" strings are only meaningful in the configuration file.

7.4 PDF parameters

wavelength

-w FLOAT, **-wavelength**=FLOAT
X-ray wavelength in Ångströms. This value is required for the "twotheta" dataformat in order to convert the scattering angles 2Θ to a momentum transfer Q. For other data formats the wavelength is not necessary and may be left undefined.

composition

-composition=STRING
Chemical composition of the sample. Supported formats are "PbTi0.5Zr0.5O3", "Pb 1 Ti 0.5 Zr 0.5 O 3" or "CH3 CH2 OH". Space characters are ignored, unit counts can be omitted, but it is important to use a proper

upper and lower case in atom symbols. Elements can appear several times in the formula, e.g., "CH3 CH3", but parenthesis are not supported.

bgscale

-bgscale=FLOAT
Scaling of the background intensities loaded from the `backgroundfile`, by default 1.

rpoly

-rpoly=FLOAT
r-limit for the maximum frequency in the $F(Q)$ correction polynomial. The PDF is unreliable at shorter *r*, however a very small rpoly would disable polynomial correction and give noisy PDF. Larger values produce closer fits with a higher degree polynomial, but when too large, they might smooth-out a useful signal in the data. The default is 0.9.

qmaxinst

-qmaxinst
The Q cutoff for the meaningful input intensities in inverse Ångströms. Some data files may contain trailing zeros or unreliable intensities at the upper bound of the detector range. The `qmaxinst` defines a threshold for unreliable data. The parameter is also used as an upper boundary for the polynomial fit correction of the $S(Q)$ data.

qmin

-qmin
The lower Q-limit for the Fourier transformation of the $F(Q)$ curve in inverse Ångströms.

qmax

-qmax
The upper Q-limit for the Fourier transformation of the $F(Q)$ curve in inverse Ångströms. This is essentially a limit, where sample signal decays to the level of data noise.

rmin

-rmin=FLOAT
Lower bound of the *r*-grid for the calculated PDF in Ångströms.

rmax

-rmax=FLOAT
Upper bound of the *r*-grid for the calculated PDF in Ångströms.

rstep

-rstep=FLOAT
Spacing of the *r*-grid for the calculated PDF in Ångströms.

7.5 Other parameters

plot

-p TYPES, **-plot**=TYPES
Plot the specified results. A comma separated list with one or more items from "iq", "sq", "fq", "gr". No plot is produced when empty, "none" or "NONE". Setting this option turns on the interactive mode.

interact

-i, -interact
Start an IPython interactive session after processing all files. Useful for tuning the configuration parameters or interactive plotting. This is always on when plot option has been set. See also *Interactive mode* for further details.

verbose

-verbose=VALUE
Level of detail for the program to report about its actions. Possible values are *error*, *warning*, *info*, *debug*, *all* or an integer number from 0 to 5. Messages are completely suppressed when 0, all messages are printed when verbose is 5 ("all") or higher. This option is useful for diagnostics of any unexpected behavior in the program.

CHAPTER

EIGHT

THE PLOTDATA PROGRAM

The PDFgetX3 software includes a simple stand-alone utility **plotdata** for plotting text data files. In most cases this program can be invoked from a command-shell as

```
plotdata file1.dat file2.dat
```

which plots the numerical data from the text files `file1.dat`, `file2.dat` together in a single graph. By default the first column is used as an x variable and the second column is used for the y values. After displaying the plot the program starts an IPython interactive session allowing the user to modify or save plots. The IPython session is initialized with the `filenames` variable containing a list of plotted files. It also pre-loads the `plotdata()` and `findfiles()` functions just as in the **pdfgetx3** interactive session. The `plotdata()` function works in a similar way as the **plotdata** program, just its arguments need to be passed as Python function arguments instead of command-line options. Thus an equivalent call of the `plotdata()` function would be:

```
In [1]: plotdata(['file1.dat', 'file2.dat'])
```

8.1 selecting files

The **plotdata** program includes a file searching feature that is useful for selecting a set of files in large directories. It is also convenient for Windows operating systems, where the command prompt cannot do filename expansion for patterns such as `*.dat`. The file search feature is controlled by the following options:

-f, -find
: Use the command line arguments as filename patterns and plot the files matching all patterns. This works almost the same as the *pdfgetx3 --find* option, however **plotdata** does not recognize `datapath` and searches for files only in the current directory. The filename patterns are described with *pdfgetx3 --find*. If patterns contain any special characters *"^$<>"* they need to be enclosed in double quotes (") on the command line so they don't get interpreted by the command shell.

-l, -list
: List the input files and exit. This is useful in conjunction with the *-f, --find* option to check if data files are selected as intended.

Assuming the current directory contains 20 files named `file1.dat`, `file2.dat`, ..., `file20.dat`, the plotting of files 9 to 13 could be done (with a check listing) as follows

```
$ plotdata -fl "<9-13>.dat"
file9.dat
file10.dat
file11.dat
file12.dat
```

```
file13.dat
$ plotdata -f "<9-13>.dat"
```

Within an interactive IPython session the equivalent plot could be produced by combining the `plotdata()` and `findfiles()` functions as

```
In [1]: plotdata(findfiles("<9-13>.dat"))
```

8.2 selecting x and y data

The **plotdata** program provides several ways of selecting columns for x or y data and for specifying plot markers or line formats. The columns can be specified using their integer index, but one needs to keep in mind the index of the first column is "0" as per Python indexing conventions. Here is a list of options supported by the *plotdata* program (and function):

-x X
: index or name of the x-column to plot. See the *-y* option for more details.

-y Y
: index or name of the y-column or columns to plot. The *Y* column specification can be a comma separated list of indices, column names or Python-like ranges, for example "1,2", "G", "1:4" (START:STOP, same as "1,2,3"), "1:4:2" (START:STOP:STEP, same as "1,3"). The column indexing starts at "0" therefore the first column should be specified as "0". A special column name "." can be used for row index.

  The column names work if the data section in the file is preceded by a headline of unique column names, for example:

  ```
  x     square      cube
  1     1           1
  2     4           8
  3     9           27
  4     16          64
  ```

  For this file the plotdata program will recognize column names "x", "square" and "cube" and an implicit "." for row index.

-s STYLE, **-style**=STYLE
: optional plot format specification. See the matplotlib plot function for a list of available formats.

-h, -help
: display a brief usage info and exit.

-V, -version
: show program version and exit.

8.3 plotdata examples

The `examples/plotdata` directory contains a `sincos.dat` file that has 3-columns of data labeled as "x", "sin" and "cos". Here are several examples of the **plotdata** capabilities when used from command line - the user is encouraged to try them out:

```
plotdata sincos.dat
plotdata -y 1,2 sincos.dat
plotdata -x . -y 0:3 sincos.dat
plotdata -y cos sincos.dat
```

```
plotdata -x sin -y cos -sr-- sincos.dat
plotdata -x 0,1 -y 1,0 sincos.dat
```

An equivalent usage from a general IPython session would be:

```
In [1]: from diffpy.pdfgetx.plotdata import plotdata
In [2]: plotdata('sincos.dat')
In [3]: plotdata('sincos.dat', y=[1,2])
In [4]: plotdata('sincos.dat', x='.', y=':3')
In [5]: plotdata('sincos.dat', y='cos')
In [6]: plotdata('sincos.dat', x='sin', y='cos', style='r--')
In [7]: plotdata('sincos.dat', x=[0,1], y=[1,0])
```

INDEX

Symbols

A

B

C

D