Small-Angle Scattering kit for Interpretation and Analysis (SASkia)

Almut Stribeck

July 2018

1 Introduction

Motivation. SASkia is a computer program under development. Its purpose is the analysis of **small-angle scattering** data. At the present stage of development I focus on the analysis of curves, although several methods are already dealing with multidimensional data. The code and parts of it are free to use in other programs.

SASkia is written in **python**. Python 2.7 is the preferred version, because the shortcomings of Python 3 still make problems. The design goal is to use only standard python packages except for the command interpreter[1].

The native **data formats** of SASkia are human readable[2]. Some more data formats are supported from the beginning[3]. The author offers implementation help for further formats, if a description of the format and sample files are provided.

SASkia is not a commercial program. It is continuously extended. Instead of a user manual some examples are given that hopefully will help the user to learn **how to use the program**. Feedback is appreciated.

Commands, Largo scripts, and python scripts. For the purpose of processing of big data SASkia is controlled by commands. Commands provide *basic* operations on a scattering curve. Many tasks of scientific relevance require to carry out a *sequence* of basic operations. Such sequences are provided by LARGO-scripts. Such scripts are text files which can easily be adapted to special conditions of the data. Tasks which require knowledge of a special experimental setup are served by python scripts. Processing of 2D scattering patterns is also provided by python scripts – at this stage of program development.

LARGO[4]-scripts concatenate a sequence of commands. They define a "super command". The call of a LARGO-script can pass arguments to the contained commands.

[1] The command interpreter is based on the package cmd2

[2] Curves are stored in simple text files with comments and attributes, 2D patterns are stored in JSON text files.

[3] EDF files written at ALBA and ESRF; TIFF files written by MAR detectors; 2D patterns written by the author's PV-WAVE procedures ("IMA"); curves in data files written at ALBA.

[4] Load with ARguments and GO

LARGO-scripts may be nested. A typical LARGO-script computes, e.g., the linear correlation function from an isotropic scattering curve contained in the file "mycurve.dat" (@cor1,mycurve).

More versatile than LARGO-scripts are python scripts, which can also be called from the command line. Python scripts require programming skills. They can access the native data of SASkia. They can call commands. They cannot call LARGO-scripts. They can implant their own data structures in SASkia. A typical python script reads all raw data related to an experimental run together with its environmental data and writes for each frame a curve file which contains the curve and all its environmental attributes (@@mergecAT,Y023_up_000,sy02_up,saxs). mergecAT: merge curves written at "A"lba in a run related to "T"emperature variation. The first argument "Y023_up_000" is the run name. The second argument "sy02_up" is the file prefix for the output files. The third argument "saxs" tells the script to use the data of the SAXS detector.

2 Installation the simple way

2.1 MS-Windows computers

Generate a **user account with a single word** (e.g. "joe", not "joe smith"). This avoids a file path which includes blank characters (" "). **Do not make any directories which include blanks!** Otherwise the command processor will run crazy.

Download "Anaconda". Anaconda is a complete python environment. Install Anaconda using the flavor Python 2.7. Do not use Python 3![5]. If Python 3 is already installed with Anaconda, you may run into problems[6].

> From `www.anaconda.com/download` get the anaconda "Windows Installer" for Python 2.7 version (For a 64-bit Windows in my case: Anaconda2-5.2.0-Windows-x86-64.exe) and install. I used the default options.

Let us assume that you are the user "myself", then your python has been installed in

```
C:\Users\myself\Anaconda2
```

and your uninstaller is

```
C:\Users\myself\Anaconda1\Uninstall-Anaconda2.exe
```

You are asked, if you want to install the editor "Visual Studio Code" by Microsoft, which supports Python coding. Why not?

From the App list start "Anaconda Prompt". A console window opens. Install the basic command interpreter which is needed by saskia. For this environment three more packages are requested, which should be installed before cmd2:

[5] In July 2018 Python 3 is still broken with respect to matplotlib interactive graphics

[6] With my limited knowledge of Windows it was impossible to replace in Anaconda Python 3 by Python 2 and get saskia to work.

```
C:\users\myself>pip install msgpack
C:\users\myself>pip install argparse
C:\users\myself>pip install grin
C:\users\myself>pip install cmd2
```

If later some packages should be missing, then install them, too!

Be sure to have a python directory in your data volume:

```
C:\users\myself>D:
D:>md python
D:>cd python
D:\python:>
```

Place the file saskia_distro1807.zip in this directory.

Then unzip it. This generates the directory tree for saskia. Check, if saskia can be started:

```
D:\python:>unzip saskia_distro1087.zip
D:\python:>cd saskia
D:\python\saskia:>python saskia.py
Retrieving individual configuration from saskia.rc ----------------------
SASkia>
```

This is the saskia prompt which indicates successful start of saskia. You leave saskia by the `quit` command. Try the `help` command.

If you do not end at the prompt, but packages are missing, then install them using pip and try again.

2.2 Linux computers

If Python 2.7 is not already installed, then use your package installer to install it.

3 Installation problems? Information which may help

3.1 Installation for Python 2.7

Even in 2018 Python 2.7.12 is still the standard provided by Linux (Ubuntu and Kubuntu), and after own experiences with both 2.7 and 3.6 I agree that this is a very good decision. Python 2.7 has everything needed, it is slim and fast.

If a Python 2.7 is provided by the operating system, then only a few modules must be installed in order to run SASkia.

For (K)ubuntu Linux there are only very few python modules which are not automatically installed by the default distribution. They are installed by:

```
sudo apt-get install python-tk
sudo python -m pip install pygame
```

If some module should be missing in some other environment, then it may help to look below where the compilation of Python 3.6 and the manual installation of all modules is described.

3.2 The SASkia package

The SASkia package requires the files `sa*.py` to be in the program directory (suggestion: `~/python/saskia`).

User-written scripts can be placed in a subdirectory scripts. Following the suggested example this would be `~/python/saskia/scripts`.

More flexibility can be gained by adapting the file `saskia.rc` to the special environment of the user. Presently SASkia only reads the variable scriptpath from `saskia.rc`. The path may be specified in an operating-system independent notation syntax using dot-symbols instead of slashes (Linux, Mac) or backslashes (MS-Windows)

SASkia can be made callable from anywhere by providing a corresponding script (bash-script for Linux, bat-file for Windows) in a directory which is in the search path. For Linux this script is conveniently placed in `~/bin/`

```
#! /bin/bash
# Script: ~/bin/saskia
# Make it executable by: "chmod 755 saskia"
# call SASkia from everywhere
python ~/python/saskia/saskia.py $*
```

NOTICE: FEATURES FROM ONGOING WORK For the evaluation of scattering patterns I have written several extra modules (`sapiawin.py`, `sapatt.py` ...), which are not called by SASkia directly. They are only called by python scripts which interact with SASkia. Such scripts are called from SASkia following the syntax "@@scriptname".

For example, some of these python scripts read a complete set of curves and convert it into two-dimensional data sets. Such data sets are "patterns" (`sapatt.py`), which are displayed in **p**ython-**i**nter**a**ctive windows (`sapiawin.py`) and stored in complex data files. Such data files can presently conform to one of four different standard formats. If such scripts are called, the corresponding library modules must have been installed. These libraries are:

JSON My standard cross-platform representation of 2D data

json-tricks A more powerful data format, only readable by python

pickle The standard python-dinosaur format. Slow.

xdrlib Data format of IDL and PV-WAVE (to read old 2D files)

By deleting respective parts of `sapatt.py` it is possible to eliminate the need to import some of these modules, if some of the functionality is not needed. JSON is my favorite for permanent storage of complex scattering data, because it is used by many other programs (well introduced) and human-readable (i.e. JSON is no binary file, but a text file).

SASkia has been developed using Python 2.7 under Linux (Kubuntu). From time to time the program is tested under Python 3.6 under Linux (Redhat) and under Anaconda-Python 3.6 running under MS Windows. Because the basic cPython of MS-Windows

does not support the interactive hook since 3.5.2, we must wait for a correction of this lapsus (*http://bugs.python.org/issue31546 PyOS_InputHook is not called*)

3.3 Intallation of Python 3.6

If Python3.6 is not already present, it must be downloaded, compiled, and installed. I give some hints which may then be important.

3.3.1 matplotlib Graphics backend interfaces

All graphics and interaction with graphics are realized by the standard package matplotlib, which needs at least one "graphics backend". The standard graphics backend is TkAgg. **Only if Python must be compiled, the following information is essential**.

In order for the graphics backend to become available, the corresponding system libraries must be supplied. For instance, if the standard TkAgg backend must be made available under RedHat or CentOS, then **before compiling Python** as root do:

```
yum install tk
yum install tk-devel # (BOTH must be provided!)
```

or for the Qt4 backend:

```
yum install PyQt4
yum install PyQt4-devel#
```

or for pycairo:

```
yum install pycairo
yum install pycairo-devel
```

For ubuntu:

```
sudo apt-get install tk
sudo apt-get install tk-dev
```

3.3.2 Installing Python from sources

Download a python source archive, unpack it, enter the generated subdirectory and as normal user do:

```
./configure --enable-optimizations
make
make test # (not necessary)
```

The make command compiles python **and** all those **matplotlib backend interfaces** which have development headers.

As super user do:

```
make install
```

With the tk-packages pre-installed we now have _tkinter, the interface for TkAgg.

3.4 Installing all packages

This section is in particular important, if no python was pre-installed. All of these packages belong to the standard libraries under (K)ubuntu, but some of them may miss in other python distributions. All packages must be installed with administrative privilege (under Linux: as “root” or with “sudo”).

It may be necessary to install the package installer "pip" itself. Under Kubuntu (which runs Python 2.7) this is done by the instruction

```
sudo apt install python-pip
```

Only on a sudoer-Linux enter a root command shell:

```
sudo bash
```

On root-Linux systems and on MS-Windows in the Anaconda console, as long as there is only one Python version installed:

```
pip install cmd2
```

This automatically installs pyparsing, pyperclip, six

```
pip install ipython # otherwise SAScur has no "ipy"
pip install numpy # For curves and some numerics
pip install matplotlib # For the graphics to work
pip install pygame # For acoustic signals
pip install LaTeX # For Label typesetting aka LaTeX
pip install Pillow # Just for fun: The new image library.
```

If Python has been compiled by the user on a CentOS system and scipy is not available:

```
yum install lapack # For the Fast Fourier Transform
yum install lapack-devel
```

and for all systems which do not have scipy, but lapack and its development headers (lapack-devel) are already available

```
pip install scipy # There you go
```

If both Python 2.7 and Python 3 are installed on the same computer, all the pip commands must be prepended by the clause `python2 -m` (if preparing to use Python 2.7) or `python3 -m` (if preparing to use Python 3). If the packages are installed on a sudoer system by individual sudo commands, the sudo command should be called like `sudo -H` to specify that the packages are installed in the systemwide home directory to be accessible for all users of the computer.

4 SASkia, its command language, and cmd2

SASkia is controlled by commands. The command language interpreter (CLI) is based on the excellent basic CLI module "cmd2". I have chosen not to rely on the newest version of cmd2, but to freeze in an older version that still has the feature of command abbreviation. This module is provided as "savcmd2.py". Nevertheless, "cmd2" should be available as well, because then the background modules pyparsing, pyperclip and six are automatically installed, and these modules are required by savcmd2.py. I admit, in doing so I am relying on that the developers of the background modules will not trim their interfaces.

The reason for this decision is not only that I like to be allowed to abbreviate command names, but also the fact that the developers have chosen to revoke a feature within a few months. Revocations must be noticed years in advance, otherwise the developer of the user program has to closely monitor the used library modules, take care of changed interfaces, adapt the user program and inform all users, that an update is necessary. I try to avoid such work with running targets.