Example 1.Examples of routines

Table 1 Selection of routines of the SAXS program package. They can be applied to image series.

Program name	Short description
saxs_mac	standard operations on images, e.g. multiply by factor and add constant
saxs_add, saxs_sub, saxs_mul, saxs_div	add, subtract, multiply, divide several images
saxs_normn	flatfield correction and normalization of scattering patterns to absolute units in 1/sterad
saxs_waxs	waxs projection and backprojection
saxs_angle	transformation to polar coordinates
saxs_row, saxs_col	row, column projection of images
saxs_ascii	convert images to ascii
saxs_patch	patch images to other images
saxs_stat	calculate image statistics
saxs_bsl, saxs_tiff	conversion of edf files to bsl/tiff files
saxs_new	create new (empty) images
saxs_poisson	generate poisson statistics
saxs_gauss	add gaussian noise
sphere2saxs	creation of scattering patterns from models or tables
binary2saxs	conversion of binary files to edf saxs files
rapid2saxs	conversion of bsl rapid files to edf saxs files

Example 2.The ESRF data file format (EDF) used by the SAXS program package

The following description should be sufficient to read the structure of the ESRF data file (EDF) written by the SAXS program package. To write EDF data files the routines and the documentation available with the package should be used.

EDF files created by the SAXS program package contain one or more sets of data blocks starting with a header consisting of ASCII characters (1-127) followed by binary data. The header consists of keyword-value pairs.

The header starts with a line feed ('\n' = ASCII 0x0a), followed by an opening curly brace ('{' = ASCII 0x7b) and ends after a closing curly braze ('}' = ASCII 0x7d) followed by a line feed ('\n' = ASCII 0x7d).

The header length is variable. For convenience, the header size and the binary block size are written as multiples of 512 bytes. Raw data files should only contain a single image block that may be followed by a variance block.

Keywords are not case sensitive. A keyword is always followed by an equal sign that separates it from its value. Each value is terminated by a semicolon ';' followed by carriage

return ('\r' = ASCII 0x0d) and line feed. Keywords and values are trimmed, i.e. leading and trailing white spaces are removed. Curly braces must not appear inside headers and should be removed from parameter strings.

The first header keyword is EDF_DataBlockID. Its parameter starts with the image number, followed by the word Image and ends with the suffix Psd for image data or Error for variance data.. The three parts are separated by dots. The image numbers of corresponding image blocks and variance blocks are identical.

The second keyword is EDF_BinarySize. It describes the size of the data block in bytes that follows the header block.

The third keyword is EDF_HeaderSize. It describes the size of the header block in bytes.

The next keywords can follow in any order. The contents of the binary block is described by ByteOrder, DataType and Dim_1, Dim_2. The meaning of these keywords is more or less evident, but for a proper explanation the documentation of the SAXS package should be consulted.

There exist several implementations of the ESRF data format. The most common one apart from the implementation in the SAXS package omits the opening line feed and starts immediately with an opening curly brace. It does not contain the keywords EDF_DataBlockID, EDF_BinarySize and EDF_HeaderSize. The corresponding values must be found by scanning the header. The image number is found after the keyword Image, the binary size is found after the keyword Size. The header size must be determined by searching the closing curly brace. The binary size is not necessarily a multiple of 512 bytes. This implementation does not allow the storage of variance data.

Example 3.Example of an ESRF data format (EDF) header

The typical block structure of a SAXS EDF file is shown (\n=ASCII 0x0d, \r=ASCII 0x0a). The variance block can be omitted.

```
\n
{\r\n
EDF_DataBlockID = 1.Image.Psd ; \r\n
EDF_BinarySize = 1048576 ; \r\n
EDF_HeaderSize = 8192 ; \r\n
ByteOrder = LowByteFirst ; \r\n
DataType = FloatValue ; \r\n
Dim_1 = 512 ; \r\n
Dim_2 = 512 ; \r\n
<other keyword value pairs>
}\n<binary image data>
```

```
\n
{\r\n
EDF_DataBlockID = 1.Image.Error ; \\r\n
EDF_BinarySize = 1048576 ; \r\n
EDF_HeaderSize = 8192 ; \r\n
ByteOrder = LowByteFirst ; \r\n
DataType = FloatValue ; \r\n
Dim_1 = 512 ; \r\n
Dim_2 = 512 ; \r\n
<other keyword value pairs>...
}\n<binary variance data>
```

Example 4. Example of a correction script

Below, a typical correction script is shown (UNIX c-shell). The spatial distortion correction is omitted because it is not part of this package. It would be done before normalization. The commands sphere2saxs and saxs_new create test images. In this way the script can be tested without detector data. The result can either be displayed with FIT2D (file type KLORA) or EDFPLOT. The ASCII file (data_0_ccd_1.txt) can be displayed with standard plot programs.

```
#!/bin/csh
# GENERATE INPUT IMAGES FOR TEST
sphere2saxs -omod n -norm -ocon 100 -odim 512 512 \
   -odis 10_m data_0_ccdraw
if ($status) exit
saxs_new -omod n -ocon 100 -odim 512 512 data_0_ccddark
if ($status) exit
saxs_new -omod n -odim 512 512 -odum -1 mask.edf
if ($status) exit
saxs_new -omod n -odim 512 512 -ocon 1 flat_1b1_z1.edf
if ($status) exit

# SUBTRACT DARK IMAGE
saxs_sub -omod n -oave -obin 1 1 +pass data_0_ccdraw \
    data_0_ccddark data_0_ccd.sub
if ($status) exit
# NORMALIZE
```

```
saxs_normn -odum -10 -rsys region -omod n -p +ccd +flat \
                                          -i3nam flat_1b1_z1.edf -i1nam data_0_ccd.sub \
   -onam data_0_ccd.nrm
if ($status) exit
# ADD MASK
saxs_add -rsys region -omod n \
  data_0_ccd.nrm mask.edf data_0_ccd.msk
if ($status) exit
# WAXS PROJECTION (minor effect for SAXS data)
saxs_waxs -omod n data_0_ccd.msk data_0_ccd.wax
# AZIMUTHAL REGROUPING
saxs_angle -rsys normal -omod n \
    data_0_ccd.wax data_0_ccd.ang
if ($status) exit
# AZIMUTHAL AVERAGING
saxs_row -omod n data_0_ccd.ang data_0_ccd.row
if ($status) exit
# OUTPUT OF AVERAGED SCATTERING CURVE
saxs_ascii +waxs +swap -scf "2*pi" +hedl "#q*nm I[Title]" \
  data_0_ccd.row
```