Software manual - perfectpattern_spheres and perfectpattern_ellipsoids

Brian R. Pauw
(Dated: January 18, 2011)

CONTENTS

GENERAL INTRODUCTION

This document attempts to explain the usage of some software packages written in Matlab revolving around the analysis of small-angle scattering patterns. The functions mostly take a few required input arguments, and optional arguments are supplied in so-called “parameter-value” pairs. This means that two arguments are supplied to the function, one “parameter”, usually a string identifying the second argument, the “value”. This looks like this in Matlab (where `>>` is the matlab prompt):

```
>>  [output_variable_1,output_variable_2,...,output_variable_n]=
          function_name(required_input_1,required_input_2,'parameter_1',1234,'parameter_2',4567);
```

These parameter-value pairs can be placed in any order (with one exception in “perfectpattern” programs), as long as the correct value follows the parameter. Limited input checks are done, but it is mostly up to the user to try to not make mistakes.

The code should be well-commented and readable, and no mistakes should remain. If, however, you do see room for improvement, please let me know so we can work on it!

PERFECTPATTERN_SPHERES AND PERFECTPATTERN_ELLIPSOIDS

purpose

These small, straightforward bits of Matlab software generate scattering patterns in the q-range requested for polydisperse, dilute spheres or ellipsoids. While nothing new per se, this implementation is guaranteed to produce correct scattering patterns irrespective of the width of the distribution. Allow me to quickly explain.

The normal way of calculating these patterns in fitting functions and the likes, is to choose an upper size limit (perhaps related to the width and mean of the distribution), and divide the size range between zero and this upper limit into perhaps 100 different sizes. Then the scattering pattern of each of these contributions is calculated, multiplied with their probability (obtained from the probability (or size) distribution function), multiplied with the square of the particle volume for that size, and then summed. In all, then, this is a numeric integration over the volume-square weighted size distribution.

The problem lies in the determination of the upper limit and the number of divisions required. In the past, I have tried using the cumulative distribution function to select ”smart” divisions, or adjusting the width and mean to compensate for the volume-square weighting, but this often resulted in the appearance of oscillatory behaviour in the scattering patterns. An alternative solution was therefore required.

These functions are not necessarily fast enough for fitting purposes, but they can be used for checking the applicability of your fitting procedures. You should get out of your fitting functions what you put into these simulated patterns.

These functions work by random generation of a number of spheres or ellipsoids. A scattering pattern is calculated from an initial number of spheres or ellipsoids. Then, the scattering pattern is calculated of the original block with the addition of a new set of shapes. This is repeated until the effect of adding a new block on the scattering pattern no longer exceeds a certain threshold.

Included are the programs for generating scattering patterns using polydisperse distributions of spheres or ellipsoids. All distributions supported by the “statistics library”’s RANDOM function are available. For ellipsoids, an additional distribution can be used for the aspect ratio.

usage

```
[q,I,varargout]=perfectpattern_spheres(q,varargin)
```

The input consists of a vector of q, if this only contains two values, it is assumed that these indicate the minimum and maximum q, and this range will be divided into a number of subdivisions (which can be controlled through the “nq” parameter.

The output parameters are q (now a vector of length ¿2) and the simulated intensity. The optional output arguments are (in sequence): A vector containing the simulated radii (useful for making histograms) and a “parameters” structured array containing all the settings specified using the parameter-value pairs provided. This makes it easier to store the settings needed for recreating the intensity or post-execution evaluation of the parameters.

Optional parameter-value pairs

examples

Simple form producing 100 points of the scattering pattern of spheres with a Gaussian distribution, a distribution mean of 10, and a distribution width of 20 (note that negative sizes are discarded):

```
[qsph,Isph]=perfectpattern_spheres([0.01 0.5],'dparam',[10 20]);
```

which produces a scattering pattern as shown on the left-hand side of Figure 1. The histogram on the right-hand side of that figure was generated by histogramming (`hist(Rsph,50)`)the sphere radii obtained by adding a third output argument:

```
[qsph,Isph,Rsph]=perfectpattern_spheres([0.01 0.5],'dparam',[10 20]);
```

After extraction of the parameters structured array, using:

TABLE I. Parameter-value pairs and explanation.

Parameter	Value type	Description
dtype	string	identifies the probability distribution function, default is 'norm' for gaussian distributions. Can be anything supported by Matlab's "random" function. 'lognormal' is implemented separately
dparam	vector	a vector containing the distribution parameters
nperblock	integer	number of spheres to be rendered at once before evaluation of the effect on the scattering pattern. Default 100
weighting	'string'	should be either of 'relative' or 'poisson'. In 'relative', the 'convcrit' parameter indicates the maximum error per point, irrespective of the intensity. If 'poisson', the maximum error is never larger than the square root of the intensity. 1/'convcrit' is used as an intensity multiplier, so that a 'convcrit' of 1e-5 will result in an I(0) of 1e5 "counts", and the error is never larger than sqrt(I). Default is 'poisson'
convcrit	float	convergence criterium or 1/intensity multiplier, default is 1e-5, providing quite smooth solutions.
timeout	float	if the process takes more than n seconds to complete, halt execution
nq	integer	if input argument *q* is a two-element vector, it is divided into *nq* equidistant sections
vwdist	string (logical)	if set to 'true', it is assumed the volume squared weighting has been taken into account in the distribution parameters and the intensity is therefore not volume-square weighted separately
param	structured array	struct from previous experiments containing the parameters. Only parameter-value pairs following this one will be applied.
d2type	string	(perfectpattern_ellipsoids) the distribution type for radius 2 (ellipsoid R,R,R2), or "strict" for a single value d2param
d2param	vector	(perfectpattern_ellipsoids) the distribution parameters for distribution 2/d2type
d2isaspect	binary	(perfectpattern_ellipsoids) if 1 (default), this means that the distribution described by d2type is an aspect ratio distribution. if 0, R2D is a radius distribution like RD

```
[qsph,Isph,Rsph,param]=perfectpattern_spheres([0.01 0.5],'dparam',[10 20]);
```

We can save ourselves some time adapting the previously used simulation, by not having to enter the previous parameterset, but instead putting the parameters to change *after* the "param" parameter:

```
[qsph,Isph,Rsph,param]=perfectpattern_spheres([0.01 0.5],'param',param,'dtype','lognormal',...
'nperblock',200,'convcrit',1e-6);
```

Notice that we have not specified the distribution parameters again, and obtain an updated "param" structured array. The resulting pattern and histogram is shown in Figure 2.

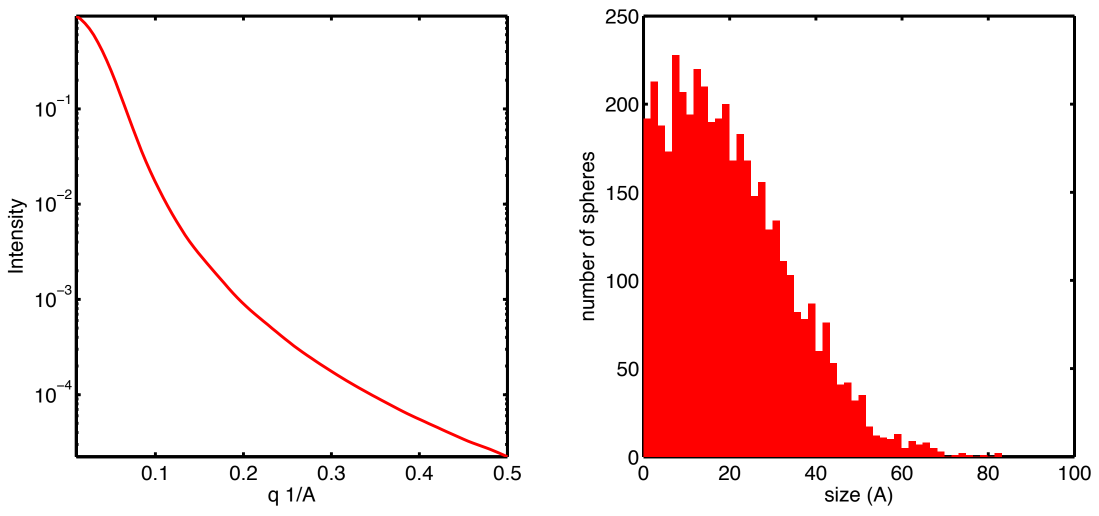

FIG. 1. Scattering pattern (left) of isolated polydisperse spheres (histogrammed on the right), generated using the perfectpattern_spheres program.

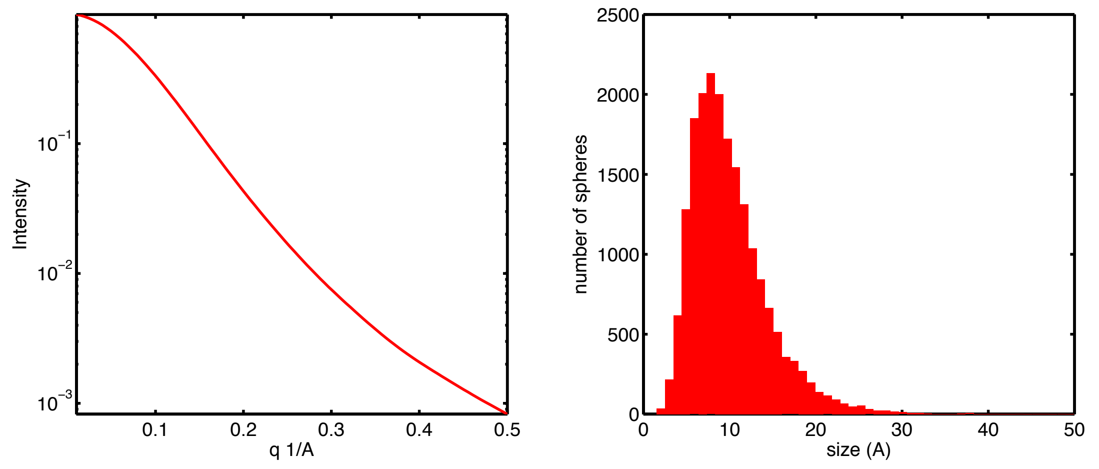

FIG. 2. Scattering pattern (left) of isolated polydisperse spheres (histogrammed on the right), generated using the perfectpattern_spheres program.