# User Manual of Debussy Program

Version 2.0, May 4, 2025

## Contents

# 1 Introduction

# 2 Main input file `filename.dwa`

## 2.1 General Info

IMPORTANT: `Debussy` reads a fixed name input file, `‘debussy.inp’`, which must contain the string `filename.dwa`.

1 FILE TYPE: `form=formatted, access=sequential, recordlength=256` (max.).

2 READING METHOD: each record is read into a 256-places character variable `rline`.

2 COMMENTS: marked by a `‘!’` or a > in the first character of `rline`.

4 IDENTIFIER: the first four characters in `rline`: `ident=rline(1:4).`

5 BUFFER: the fifth and following characters in `rline`: `buffer=rline(5:).`

6 MANDATORY/OPTIONAL FLAG: a symbol `!` specifies that a given entry is always mandatory; `*` that it is always optional; `?` that it is conditionally optional (depending on the value of other entries, as specified).

7 FILES AND PATHNAMES: when we refer to a file name, unless otherwise specified, we refer to the complete (pathname-inclusive) name, as and when required by the operating system. *Exception:* only for structural databases, the filename and pathname are given separately.

8 SECTIONS: The file `filename.dwa` is organized in three consecutive main sections, in the order:

I the **Datasets** Section (Sec. 2.2);
II the **Structure models** Section (Sec. 2.4);
III the **Refinement and Output** Section (Sec. 2.5).

A line with IDENTIFIER `****` marks the start of every section.

## 2.2 Datasets

This section of `filename.dwa` contains the directives for reading and using one or more experimental datasets.

For each dataset, we read

| `ident` | `buffer` content | |
|---|---|---|
| `#`$n$␣␣[a] | **TECHNIQUE**<br>**TECHNIQUE** is a 3-characters string indicating the experimental technique (only `XRD` is allowed for now). Neutron and Electron Powder Diffraction techniques are under development. Please contact us for further details. | ! |
| `data` | **DATA_FILENAME**<br>Two cases: either **DATA_FILENAME** is a string indicating the file name where the experimental data are stored or “none”. In the latter case only a simulation can be performed. | ! |

[a] $n$ in `ident` is the progressive dataset number.

*Continues...*

| ident | buffer content | |
|---|---|---|
| form | **FORM**; **NDATA**; **NSKIP_HEAD, NSKIP_FOOT**<br>Four integers: **FORM** always mandatory, **NDATA** and an ordered pair **NSKIP_HEAD, NSKIP_FOOT** depending on cases. The first mandatory integer **FORM** = 1, 2, 3 or 4 indicates the kind of data in the file:<br>1: only intensities (counts) $I_k$, $k = 1 \dots N$.<br>2: pairs angle-intensity (degrees-counts) as $2\theta_k, I_k$, $k = 1 \dots N$.<br>3: pairs intensity-standard deviation (counts-counts) as $I_k, \sigma_k$, $k = 1 \dots N$.<br>4: triples angle - intensity - standard deviation (degrees-counts-counts) as $2\theta_k, I_k, \sigma_k$, $k = 1 \dots N$.<br>If the file format is such that every record ($\equiv$ every row) contains only one data point (*i.e.* one intensity if **FORM**=1, one pair angle-intensity if **FORM**=2, and so on) this will be enough. Otherwise, it is necessary to supply more informations.<br>• If there is more than one data point per record (row), then the total number of data points in the file must be given as **NDATA**, always after **FORM** and before the pair **NSKIP_HEAD, NSKIP_FOOT** if the latter is present.<br>• If the file contains a header and/or a footer, then: the number of lines of the file header must be given as **NSKIP_HEAD** and the number of footer lines is to be given by **NSKIP_FOOT**. Remark: except in the case that neither header nor footer are present, it is mandatory to give both values as an ordered pair. | ! |

*Continues...*

| ident | buffer content | |
|---|---|---|
| rang | **ANGRANGE**, $n_{every}$<br>A real triple **ANGRANGE** $= 2\theta_{min}, 2\theta_{max}, 2\theta_{step}$ ($^o$), optionally followed by a positive integer $n_{every}$. Cases:<br>• In case **FORM**=2, **ANGRANGE** is optional; if present, $2\theta_{min}$ and $2\theta_{max}$ will be used as cutoff limits, skipping data beyond. $n_{every}$ is not used.<br>• In case **FORM**=1 or 3, **ANGRANGE** is mandatory ($n_{every}$ is not used.) and it must hold that:<br>$2\theta_k = 2\theta_{min} + (k-1)2\theta_{step},\ k = 1 \ldots$ **NDATA**;<br>**NDATA**$= 1 + (2\theta_{max} - 2\theta_{min})/2\theta_{step}$.<br>• In case **FORM**= 4, the optional integer value $n_{every}$ will be read and used (if present). It is useful to decimate huge high-resolution data sets, keeping only one point every $n_{every}$, so as to speed up initial runs. | * |
| inst | **ADD_INST** + **IRFP**$_{1,\ldots}$ **IRFP**$_{12}$.<br>**ADD_INST** is an integer flag specifying if an IRF has to be added to the calculated pattern of this dataset and if yes, which type. Value **ADD_INST=0** does not add IRF. **ADD_INST=1** adds an IRF with Caglioti equation parameters. **ADD_INST=2** adds an IRF with pseudo-Voigt parameters. See. Sec. 2.3 for details and explanations.<br>The rest (12 real numbers) are the parameters describing the wanted IRF. The first 6 (**IRFP**$_{1,\ldots}$ **IRFP**$_6$) are for the optics contribution (either form); **IRFP**$_1$ is for the axial divergence; **IRFP**$_{8,\ldots}$ **IRFP**$_{12}$ are sample-related effects. The parameters are described clearly in Sec. 2.3 | ? |

*Continues...*

| ident | buffer content | |
|---|---|---|
| blnk | **BLANK_FILENAME**<br>If present[1], this option indicates that the background is to be evaluated by scaling a measured 'blank'. **BLANK_FILENAME** is the name of the file where the measured 'blank' is stored. Its content and format are: $2\vartheta_k$ $I_k$ (intensity) $\sigma_k$ (sigma, optional), in case of single 'blnk' signal; $2\vartheta_k$ $I_k^1$ (intensity) $I_k^2$ $\ldots I_k^{\texttt{blnc}}$, in case of multiple 'blnk'. $2\vartheta_k$ should be equal to the corresponding values in the dataset (within 5 decimal digits). It is warmly advised that $I_k's$ of the blank be filtered to reduce noise. | * |
| blnc | **FILENAME_NBLNK** + **Porod TERM**. Here, **FILENAME_NBLNK** is an integer specifying the number of 'blank' signals contained in **BLANK_FILENAME** of blnk. **Porod TERM** is an integer flag assuming values 0 or 1, specifying whether a Porod-like term should (=1) or not (=0) added to the fit of the small-angle component of the observed data. If a **Porod TERM** is being specified then also **FILENAME_NBLNK** must be given (setting to 0 is allowed). | ? |
| cheb | **CHEB_NC**<br>If present[1], this option indicates that the background is to be modeled as a linear combination of Chebyshev polynomials. **CHEB_NC** is an integer specifying the maximum number of coefficients. | * |
| youn | **YOUNG_NC**<br>If present[1], this option indicates that the background is to be modeled as a Young polynomial. **YOUNG_NC** is an integer specifying the maximum degree. | * |

*Continues...*

[1]In case of multiple datasets from the same teechnique, if the same background is assumed for all, the background setting (type = blnk, cheb, youn and number of coefficients or filename, as appropriate) must be specified only for the first dataset, otherwise a setting must be specified for each dataset.

| ident | buffer content | |
|---|---|---|
| wave | **WAVELENGTH**<br>• Case of single-wavelength:<br>**WAVELENGTH** is a real number $\lambda$ corresponding to the value of the wavelength in Å;<br>• Case of double-wavelength:<br>**WAVELENGTH** is a real triple $\lambda_1, \lambda_2, w_\lambda$ containing the values of the wavelengths (Å) and the intensity ratio of $I(\lambda_2)/I(\lambda_1)$, respectively. | ! |
| esdw | **WAVELENGTH_ERROR**<br>A real number **WAVELENGTH_ERROR**$= \sigma_\lambda$ representing the absolute maximum error on the measured wavelength (in the case of synchrotron radiation). | * |
| beam | **RADIATION**<br>A single character **RADIATION** which may be X or S indicating the radiation type used to collect experimental diffraction data (among the featured ones), according to the following notation:<br>X: X-Ray laboratory source<br>S: X-Ray synchrotron source (default)<br>Neutron and electron diffraction data modeling is under development. Please contact us for further details. | * |
| mono | **MONO_POSIT**, **COBRA**<br>In case **RADIATION** = X:<br>one integer flag **MONO_POSIT** = 1 or 2 if monochromator is on incident or diffracted beam, respectively; plus one real number **COBRA** (= the cosine of the monochromator Bragg angle) for the polarization correction. | * |

*Continues...*

| ident | buffer content | |
|---|---|---|
| pola | **ECCENTRICITY**, **PHI**, **COBRA** In case **RADIATION** = X, three real numbers: **ECCENTRICITY** ratio $b/a$ of electric field; **PHI** angle $\phi$ (deg) of the closest ellipse axis (def. as $b$) to the scattering plane; **COBRA** (the cosine of the monochromator $2\theta$ Bragg angle) for the polarisation correction. If given disables reading of mono. For $s-$polarised Synchrotron radiation **ECCENTRICITY**≈0.01, **PHI**=0.0, **COBRA**=1.0, for circularly polarised laboratory X-ray source without monochromator **ECCENTRICITY**=1.0, **PHI**=0.0, **COBRA**=1.0. | * |
| geom | **GEOM**, **CORR** One string **GEOM** which can be transmDS: Debye-Scherrer geometry; transmFP: Flat-plate transmission geometry; reflecBB: Bragg-Brentano geometry; thinfilm: thin-film diffraction; plus one real number **CORR**=$\mu t$ for the absorption correction (WARNING! Correction not yet implemented!) | * |

## 2.3 About the IRF parameter (flag `inst`)

IRF parameters are *DATASET* dependent. For every dataset, one must (can) give a new input string like

```
inst  [I1]   [r01] [r02] [r03] [r04] [r05] [r06] [r07]   [r08] [r09] [r10] [r11] [r12]
```

where after the label `inst` an integer `[i1]` and 12 floats `[r01]`...`[r12]` are read.

- `[i1]` is the flag `INST_FLAG` $\rightarrow$ `INST_FLAG_W`, that can be 0,1,2, meaning

  0 : no IRF will be used

  1 : IRF will be used. The angle-dependent part contains a Voigt where the Caglioti equation (see Sec. 2.3.3) 6 parameters give the angular dependence of the Gaussian and Lorentz FWHMs.

  2 : IRF will be used. The angle-dependent part contains a Voigt (see Sec. 2.3.4) 6 parameters give the angular dependence of the Gaussian and Lorentz FWHMs (through those of the corresponding pseudo-Voigt given in input).

- `[r01]`...`[r06]` are the 6 parameters mentioned above.
- `[r07]` is the axial divergence width parameter $\zeta$ (see Sec. 2.3.1).
- `[r08]` is the capillary width parameter $\xi$ (see Sec. 2.3.2).
- `[r09]` is the wobbling width parameter $\omega$.
- `[r10]`...`[12]]` will be defined.

### 2.3.1 Definition of $\zeta$

The angular width $\zeta$ must be calculated from parameters

$s$ : the horizontal slit width [mm] (horizontal means orthogonal to the diffraction plane). Typ. $s$=4 mm

$h$ : the horizontal detector width [mm] (horizontal means orthogonal to the diffraction plane). Typ. $s$=8 mm (MythenII)

$R$ : the sample-detector distance.Typ. $R$=762.5 mm (MythenII, layer 1), $R$=760 mm (MythenIII)

The distribution used is a one-sided exponential (we write $x$ for the general diffraction angle $2\theta$, $x_0$ for the current central $2\theta$):

$$P(x) = \begin{cases} \dfrac{1}{\zeta \cot(x_0)} \exp\left(-\dfrac{|x-x_0|}{\zeta \cot(x_0)}\right) & \text{if} \quad (x-x_0)\cot(x_0) \leqslant 0 \\ 0 & \text{if} \quad (x-x_0)\cot(x_0) > 0 \end{cases} \tag{1}$$

It is normalized to 1 and its first moment is

$$\int_{-\infty}^{+\infty} \mathrm{d}x \; xP(x) = -\zeta \cot(x_0).$$

Comparing it with that of the (ugly) exact function as in [E. Prince and B. H. Toby, *J. Appl. Cryst.* (2005) **38**, 804-807] : Eqs. (1, 4a)] it turns out that we must have (in degrees)

$$\zeta = \frac{180}{\pi}\left(\frac{h^2+s^2}{24\,R^2}\right) \tag{2}$$

### 2.3.2 Definition of $\xi$

The angular width $\xi$ must be calculated from parameters

$d$ : the capillary diameter. Typ. $d$=0.3 mm

$R$ : the sample-detector distance.Typ. $R$=762.5 mm (MythenII, layer 1), $R$=760 mm (MythenIII)

Then simply

$$\xi = \frac{180}{\pi}\left(\frac{d}{2R}\right) \tag{3}$$

### 2.3.3 Caglioti equation parameters

The Voigt function is the convolution of a Gaussian

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) \tag{4}$$

and a Lorentzian or Cauchy function

$$L(x) = \frac{\pi}{w}\frac{1}{1+(x/w)^2} \tag{5}$$

so

$$V(x) = \int_{-\infty}^{+\infty} \mathrm{d}x' \, G(x')L(x-x').$$

Note that the FWHM of the two cuves are

$$\mathrm{FWHM}_G = 2\sigma\sqrt{2\log(2)}; \qquad \mathrm{FWHM}_L = 2w.$$

Modern evaluation of $V(x)$ is performed using the complex error function (Faddaeeva function). In the Caglioti parametrization, the widths $\sigma$ and $w$ are given by

$$\begin{aligned} \sigma &= \frac{1}{2\sqrt{2\log(2)}}\sqrt{U\tan^2(\theta)+V\tan(\theta)+W} \\ w &= X\tan(\theta)+Y\sec(\theta)+Z \end{aligned} \tag{6}$$

where $U, V, W, X, Y, Z$ are the `[r01]` ... `[r06]` input values (in the same order). Care must be taken that the square root argument remains $> 0$!

### 2.3.4 Pseudo-Voigt parameters

This parametrization relies on approximating the $V(x)$ by a pseudo-Voigt function. In this case, a global FWHM $F$ and a mixing parameter $\eta$ ($0 \leqslant \eta \leqslant 1$) are given directly by the parameters:

$$\begin{aligned} F &= h_a + h_b \tan(\theta) + h_c \sec(\theta) \\ \eta &= lor_a + lor_b \tan(\theta) + lor_c \sec(\theta) \end{aligned} \tag{7}$$

where $h_a, h_b, h_c, lor_a, lor_b, lor_c$ are the `[r01]...[r06]` input values (in the same order). We transform now $F, \eta$ into the corresponding $\sigma, w$ in order to evaluate a Voigt profile. We first solve numerically the equation

$$\eta = y\big(1.36603 - 0.47719y + 0.11116y^2\big)$$

where

$$y = \frac{2w}{F} \qquad \leftrightarrow \qquad w = \frac{yF}{2}$$

Then we solve numerically the equation (valid for a Voigt)

$$F = 0.5346(2w) + \sqrt{0.2166(4w^2) + 8\log(2)\sigma^2} = 0.5346yF + \sqrt{0.2166y^2F^2 + 5.545177444479562\sigma^2}$$

for the separate width $\sigma, w$.

## 2.4 Structure models

This section of filename.dwa contains the directives for reading and using one or more structural models from one or more Database(s).

For each structural model, we read

| ident | buffer content | |
|---|---|---|
| %$m$␣␣[b] | **STRUCTURE_NAME**<br>**STRUCTURE_NAME** is a string indicating the file name of the structure model. Such string will be used by the program as a rootname for some output files.<br>[b] $m$ in ident is the progressive structure model number. | ! |
| DB0x | **PATH_DATABASE** is a string indicating the Database pathname where the sampled distances files are stored. DB0x is the Database Code, with x=1,2,3,4. In case of a multiple structures input, the various DB0x have to be listed in increasing order of x. | ! |
| nclu | **NCLU**<br>**NCLU** is an integer specifying the maximum NC size (as number of shells) of the associated DB03. | ! |
| nrod | **NROD**<br>**NROD** are four integers specifying the minimum (always =1) and maximum NC size (as number of shells, i.e. "1 n 1 m") of the associated DB04, along a=b and c crystallographic directions, respectively. | ! |
| shap | **SHAPE**<br>**SHAPE** is a 3-characters string = SPH or PAR or CYL or HEX, specifying the shape of the NC family in the associated DB0x. For DB03, SPH (sphere) is the unique choice; for DB04 the shape can be chosen among PAR, CYL, HEX. | ! |
| prot | **PROTOTYPING** is a string with values 'yes'/'no' indicating whether structure prototyping is being performed. | * |

*Continues...*

[1]Structure prototyping consist in using a previously generate database for either iso-structural compounds (only atomic species changes) or unit cell parameters adjustments ($a$, $b$ and $c$ are different) or both.

| ident | buffer content | |
|---|---|---|
| chem | **ATOMS**<br>**ATOMS** is a string, read if **PROTOTYPING** is 'yes', as `'A=Au,B=Ni,C=K,...'` associating the letter with the corresponding chemical symbol used to assign different atomic species to the database files **PATH_DATABASE**. | ? |
| cell | **CELL**<br>**CELL** mandatory (!) only for DB02; is a 6-entries real vector giving the unit cell axes lengths and angles $a$, $b$, $c$, $\alpha$, $\beta$, $\gamma$; lengths in Å, angles in degrees ($^o$). Else **CELL** is read if **PROTOTYPING** is 'yes' and used to assign new cell parameters $(a, b, c)$ to the database files **PATH_DATABASE**. | ? |
| parx | **PARAMETER_FILE**<br>**PARAMETER_FILE** is a string such that **PARAMETER_FILE**`.par` is the file name where, on each line, an identifier string, lower bounds, initial values and upper bounds of the parameters needed by the $m$-th structure model are given. See example in Sec. 5.2.<br>NOTE: the file extension needs to be `.par`. At the end of each run, refined parameter values will be saved in a newly created file named **STRUCTURE_NAME0**$m$**_Best**`.par`, where $m$ is the progressive number assigned to the structure in **STRUCTURE_NAME**. | ! |

## 2.5 Refinement and Output

| ident | buffer content | |
|---|---|---|
| simu | **SIMUL_FLAG**<br>**SIMUL_FLAG** is an integer flag with possible values:<br>0 : perform a full refinement as specified below. Default.<br>1 : perform only once a calculated-spectrum for each of the input datasets, based on the parameters in the various **PARAMETER_FILE**s.<br>-1 : to be used after a refinement run (**SIMUL_FLAG**=0) to calculate the standard errors of the refined parameters, based on the parameters in the various **PARAMETER_FILE**s. | * |
| calm | **CALC_FLAG**<br>**CALC_FLAG** is an integer flag with possible values:<br>0 : Output only the global calculated spectrum for each dataset. Default.<br>1 : Output the structure type-separated components of the calculated spectra for each dataset. | * |
| seed | **SIMUL_FLAG**<br>and **REFINEMENT_FILE**<br>is the random seed to initialise the generation of random numbers during refinement. If this keyword is omitted, the random seed is generated automatically. Useful for generating reproducible random numbers sequences within the refinements, with newer GNU compilers ($gfortran > v6$), which have a different random number generation than older versions. | * |
| rfil | **REFINEMENT_FILE**<br>**REFINEMENT_FILE** is a string such that **REFINEMENT_FILE**.ref is the file name where the refinement strategy is stored. See example in Sec. 5.3. | ! |

*Continues...*

| ident | buffer content | |
|---|---|---|
| outs | **OUTPUT_FILE**,<br>**OUTPUT_FILE** is a string containing the (path plus) rootname of some of the output files, which is used by the program in case **SIMUL_FLAG** is 1. It can be used to control the name of the files. Outputs are (by file extension):<br>• **OUTPUT_FILE_Best**.dis Extended output of all interesting results and derived statistics, *e.g.* average size, distributions, weight fractions. The filename.dwa will be used as root for this output, in case **SIMUL_FLAG** is 0 .<br>• **OUTPUT_FILE_TECHNIQUE**#$n$.cal Calculated spectra for every dataset (#$n$ is the progressive number in the dataset section), and depending on the value of **CALC_FLAG**: only a cumulative one for each dataset or all separated components. In case **SIMUL_FLAG** is 0, the filename of each dataset will be used as root for the corresponding output. | * |
| rpdf | **RPDF**,<br>**RPDF** is an integer flag with possible values:<br>0 : Nothing [:-)].<br>1 : output the radial pair distribution function of experimental, (total) calculated and background in the file **DATA_FILENAME**.rpdfn. | * |

## 3 Parameter file *.par

### 3.1 Main Info

The .par file contains a list of parameters along with their lower and upper bounds, plus a flag indicating whether the parameter can be refined or not. The parameters are divided into two groups: global and atom-specific. Global parameters are the first 11 (line 1-11). Atom-specific parameters are 7 per each atomic species, each group starts with the `AT00n` line. The meaning of the parameters listed in the `.par` file is given in the following.

### 3.1.1 Global parameters

`STcod` : Flag identifying one of the implemented function to model the dependence of (uniform isotropic) lattice expansion *vs.* cluster size for the related phase. In detail:

- `STcod 0` Four-paramteter smoothed-step dependence: *cf.* Eq. (8). Use for univariate (DB02) only.
- `STcod 1` Constant (independent from size). Use for bivariate and univariate.
- `STcod 2` Inverse-linear. Use for univariate only.
- `STcod 3` Inverse-linear. Use for bivariate only.
- `STcod 4` Inverse-linear. Use for univariate only.
- `STcod 5` Inverse-linear. Use for bivariate only.

`VALn1` : Integer auxiliary value (fixed point $\nu_1$) for the lattice expansion function. It is useful with narrow distributions and with the function corresponding to `STcod 2`. $\nu_1$ should be chosen near the (best guess of) the distribution center. See Sec. 3.1.3 for details.

**Format:** `Xname` $x_L$ $x_i$ $x_U$ $I_{flag}$
where:

- `Xname` is the parameter identifyer string;
- $x_L$ is the lower bound of the parameter value;
- $x_i$ is the initial parameter value;
- $x_U$ is the upper bound of the parameter value;
- $I_{flag}$ is a flag (0 or 1) that indicates if the parameter is in use (output only, in autoamatically generated `.par` files.

The first five parameters are needed to describe a - possibly bivariate - log-normal size distribution. For a simple univariate log-normal size distribution only two parameters are needed, and these are the first two. All five are used for a bivariate distribution. Our choice of parametrization is the distribution average and standard deviation. For bivariate log-normal size distribution, these are referred to the log-axes. See the Appendix for details on the log-normal distributions and their parametrization.

`AV1LN` : Average of the log-normal particle number distribution in size. Used for *monovariate cluster families* and *bivariate cluster families (first growth direction)*.

`SD1LN` : Standard deviation of the log-normal particle number distribution in size. Used for *monovariate cluster families* and *bivariate cluster families (first growth direction)*.

`AV2LN` : Average of the log-normal particle number distribution in size. Used only for *bivariate cluster families (second growth direction)*.

SD2LN : Standard deviation of the log-normal particle number distribution in size. Used only for *bivariate cluster families (second growth direction)*.

PHILN : Log-axes inclination of the log-normal particle number distribution in size. Used only for *bivariate cluster families*.

The second set of four parameters are needed to describe an uniform expansion factor applied to the clusters of a family as a function of size. Different functional dependencies are implemented (selected by label STcod above), and of course the bivariate size dependence is considered. Details are as follows.

STR_i : Parameter $\Omega$, always describing the lattice expansion in the limit of infinite size, except when STcod=2, where it gives the value at the fixed point $n = \nu_1+1$, immediately following the value of VALn1=$\nu_1$. See Sec. 3.1.3. For STcod=4 or 5 parameter $\alpha$, with $1+\alpha$ describing the strain in the limit of infinite size, see Sec. 3.1.3, Eq. (13) and Eq. (14).

STR_1 : Parameter $\xi$, always describing the lattice expansion either at the smallest size ($n = 1$) when STcod=0; or, when STcod=2, at the fixed point $n = \nu_1$ specified by the value of VALn1=$\nu_1$. See Sec. 3.1.3, Eq. (9). For STcod=4 and parameter $\beta$, see Sec. 3.1.3, Eq. (13) and Eq. (14).

STR_C : Parameter $\nu_0$, center of the step (for STcod=0). See Sec. 3.1.3.

STR_W : Parameter $w_s$, width of the step (for STcod=0). See Sec. 3.1.3.

### 3.1.2 Atom-specific parameters

Site occupancy factors and thermal parameters are selectively refined for each atomic species, according to the scheme defined in the ATOOn line.

ATOOn : LAW_OKK LAW_BTH, where LAW_***=1,2 defines the law to be used, 1: size-independent, 2:size-dependent (See Sec. 3.1.4).

OKK_I : Parameter $O_1$, describing the occupancy factor independent of size if LAW_OKK=1, or the largest size if LAW_OKK=2. See Sec. 3.1.4.

OKK_0 : Parameter $O_0$, describing the occupancy factor at the smallest size. See Sec. 3.1.4.

OKK_L : Parameter $O_L$, describing the growth/decay constant of the occupation factor.

BTH_I : Parameter $B_1$, describing the thermal parameter $B$ independent of size if LAW_BTH=1, or at the largest size if LAW_BTH=2. See Sec. 3.1.4.

BTH_0 : Parameter $B_0$, describing the thermal parameter $B$ at the smallest size. See Sec. 3.1.4.

BTH_L : Parameter $B_L$, describing the growth/decay constant of the thermal parameter. See Sec. 3.1.4.

### 3.1.3 Lattice expansion functions

A resume in tabular form of advised correspondences between DB-types and `STcod`:

| `STcod` | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| `DB01` | – | $\surd$ | – | – | – | – |
| `DB02` | $\surd$ | $\surd$ | $\surd$ | – | – | – |
| `DB03` | – | $\surd$ | $\surd$ | – | $\surd$ | – |
| `DB04` | – | $\surd$ | – | $\surd$ | – | $\surd$ |
| `DB05` | – | $\surd$ | – | – | – | – |

A lattice expansion function is a function $X_s(n)$, or $X_s(n_1, n_2)$ in the bivariate case, where $n$ (or the pair $(n_1, n_2)$) is the cluster index and size parameter, the latter meaning that the cluster diameter $D$ is a linear function of $n$ (or of $(n_1, n_2)$). Therefore all quantities that depend on size can be considered as functions of $n$ (or of $(n_1, n_2)$). The functions corresponding to `STcod`=1,4 and 5

are, instead, directly expressed in terms of $D$ ,thus the lattice expansion function is written $X_s(D)$, or $X_s(D_{ab}, D_c)$ in the bivariate case. The lattice expansion function works simply by multiplying all interatomic distances of cluster $n$ by $X_s(D)$ (or, all interatomic distances of cluster $(D_{ab}, D_c)$ by $X_s(D_{ab}, D_c)$). Different functions are provided, connected with the flag `STcod` and the uni- or bi-variateness of the cluster family, as follows:

Case `STcod`=0: a smoothed-step function, of limited use, valid only for univariate families, `DB02`-type.

$$X_s(n) = \Omega - \frac{\Delta}{1 + \exp\left(\dfrac{n - \nu_0}{w_s}\right)} \qquad \text{where} \qquad \Delta = \left[1 + \exp\left(\frac{1 - \nu_0}{w_s}\right)\right](\Omega - \xi) \tag{8}$$

Note that

$$X_s(1) = \xi; \qquad X_s(\infty) = \Omega.$$

Case `STcod`=1: a constant function.

$$X_s(n) = \Omega \tag{9}$$

Case `STcod`=2: an inverse-linear function.

$$X_s(n) = \Omega + (\Omega - \xi)(\nu_1 + 1/2)\left[1 - \frac{\nu_1 + 3/2}{n + 1/2}\right]. \tag{10}$$

Note that

$$X_s(\nu_1) = \xi; \qquad X_s(\nu_1 + 1) = \Omega.$$

Case `STcod`=3: an inverse-linear function - all types.

If univariate:

$$X_s(n) = \xi + \frac{3}{2}\frac{\Omega - \xi}{n + 1/2} \tag{11}$$

If bivariate:

$$\begin{aligned} A_1(n_1) &= \xi + \frac{3}{2}\frac{\Omega - \xi}{n_1 + 1/2}; \\ A_2(n_2) &= \xi + \frac{3}{2}\frac{\nu_0 - \xi}{n_2 + 1/2}; \\ X_s(n_1, n_2) &= \left[A_1^2(n_1) A_2(n_2)\right]^{1/3} \end{aligned} \tag{12}$$

Case `STcod`=4: an inverse-linear function.

$$X_s(D) = \left(1 - \frac{\alpha}{D + \alpha}\right)(1 + \beta) \tag{13}$$

Case `STcod`=5: an inverse-linear function.

$$X_s(D) = \left(1 - \frac{\alpha}{\frac{V}{S} + \alpha}\right)(1 + \beta) \tag{14}$$

Note that

$$V = \pi \frac{D_{ab}^2}{4} D_c; \qquad S = \pi D_{ab} D_c + \frac{\pi D_{ab}^2}{2}$$

NB: functions corresponding to `STcod`=0, 2 and 3 are DEPRECATED. Rather use `STcod`= 4, 5.

### 3.1.4 Site occupancy factor and thermal parameter functions

Depending on the value of the `LAW_***`-flag, two kind of functions are available for site occupancy factor and thermal parameters:

Case `LAW_***`=1: constant function

$$O(D) = O_1 \qquad \text{resp.} \qquad B(D) = B_1 \tag{15}$$

Case `LAW_***`=2: size-dependent function

$$O(D) = O_1 + (O_0 - O_1)\exp(-D/O_L) \qquad \text{resp.} \qquad B(D) = B_1 + (B_0 - B_1)\exp(-D/B_L) \tag{16}$$

Where $D$ represents the cluster diameter for `DB03,DB05` or the diameter of the equivalent volume sphere for `DB04`.

For thermal parameter or Debye-Waller factor we use the crystallographers' parametrization, using the scalar $B$ to describe uniform isotropic thermal motion. This introduces a factor

$$\mathrm{e}^{-B\sin^2(\theta)/\lambda^2} = \mathrm{e}^{-Bq^2/4}$$

where $B = 8\pi^2\langle u^2\rangle$ if $\langle u^2\rangle$ is the mean-square displacement and $q = 2\sin(\theta)/\lambda$ is the reciprocal space variable.

# 4 Refinement file `*.ref`

## 4.1 Main Info

The `.ref` file provides a high level management for the refinement strategy. Whereas in the `.par` file refinement flags control each single parameter, in this file refinement flags can be used to control groups of parameters: size, lattice expansion, sof and D.-W. of each atom for each structure and datasets adding the option of performing multistage refinements.

### 4.1.1 Parameters

The file is organized in a nested-box like manner, with three mandatory elements: the `Structure` and `Dataset` contained in the `Stage` one.

The fist line contains the line: `Number of stages # X` where `X` is the total number of stages to be performed.
Follows the line `stage # N float ALGORITHM` where `N` is the stage number: `X1, 2, ..., X`; `float` is a number indicating the tolerance for convergence for the specified `ALGORITHM`. `ALGORITHM` can be any of `COMPLEX`, `ANNEAL`.
Follows a line with: `% S`, where `S` is the structure number, e.g. `1`. `Structure` elements must be placed in order of increasing `S` and their sequence must be the same as the one provided in the`.par` file. The following line contains the flags for the group of parameters: size, lattice expansion plus one sof and D.-W. flag for each atomic species of the specified structure. The number of flags is therefore 2+2$A$, where $A$ is the number of atomic species of the specified structure, that is the minimum number of flags is four since at least one atom must be present. Then a line with : `# D`, where `D` is the total number of datasets. Then a line with the flags for each dataset.

All flags can be `1` or `0` meaning *refine* or *not refine*. Comment lines can be inserted using the `!` symbol at the beginning of the line.
Examples of various `.ref` files are presented in Section 5.3.

## 5 Examples

### 5.1 filename.dwa examples

```
******* Datasets section ************
#1         XRD
  data    anatase_17keV.dat
  form    4
  cheb    12
  wave    0.729586
  esdw    0.0007
  beam    X
  geom    reflecBB 0.0
******* Structures section ************
%1        anatase
  db04    CLAUDE-DB/Anatase/DISTANCES/SAMPT0120A/anatase
  nrod    1 20 1 20
  shap    PAR
  parx    anatase.par
******* Refinement section ************
  simu    0
  rfil    anatase.ref
  calm    1
  outs    anatase_0
```

Figure 1: Debussy .dwa input file for a powder specimen of anatase: example of a single experimental dataset (anatase_17keV.dat) and a single DB04 database, previously created by CLaUDe. This kind of input enables the program to run in 'refinement mode' (simu 0), using the refinement strategy specified in anatase.ref.

```
******* Datasets section ************
#1        XRD
  data    Ti6_mix_11-3keV.dat
  form    4
  blnk     glass_of_Ti6_mix_11-3keV.dat
  cheb    12
  wave    1.097282
  esdw    0.0002
  beam    S
  geom    transmDS 0.0
#2        XRD
  data    Ti6_mix_17keV.dat
  form    4
  blnk    glass_allTiO2_17keV.dat
  cheb    7
  wave    0.729586
  esdw    0.0007
  beam    S
  geom    transmDS 0.0
******* Structures section ************
%1        anatase
  db04    CLAUDE-DB/Anatase/DISTANCES/SAMPTO120A/anatase
  prot     yes
  nrod    1 20 1 20
  shap    PAR
  cell      3.79936  3.79936  9.49801   90.00  90.00  90.00
  parx    anatase.par
%2        rutile
  db04    CLAUDE-DB/Rutile/DISTANCES/SAMPTO120A/rutile
  nrod    1 20 1 50
  shap    PAR
  parx    rutile.par
******* Refinement section ************
  simu    1
  rfil       Ti6_mix.ref
  calm    1
  outs    Ti6_mix_2wl
```

Figure 2: Debussy .dwa input file for a powder mixture of anatase and rutile: example of input working with two experimental datasets, two experimental blanks and two DB04 databases. This kind of file enables the program to run in ‘simulation mode’ (simu 1).

## 5.2 Parameter file: examples and main Info

The .par file contains a list of parameters along with their lower and upper bounds, plus a flag indicating whether the parameter can be refined or not. The parameters are divided into two groups: global and atom-specific. Global parameters are the first 11 (line 1-11). Atom-specific parameters are 7 per each atomic species, each group starts with the `ATO0n` line.

```
STcod          3
VALn1          0
AV1LN        8.0000000000000000       8.2448747266176179       15.0000000000000000   1
SD1LN        2.0000000000000000       2.8702408279602150        5.0000000000000000   1
AV2LN        8.0000000000000000      10.0068018005545450       15.0000000000000000   1
SD2LN        2.0000000000000000       3.6982245056089482        5.0000000000000000   1
PHILN      -90.0000000000000000      45.4229305919096050       90.0000000000000000   1
STR_i        0.9500000000000000       1.0000000000000000        1.0500000000000000   1
STR_1        0.9500000000000000       1.0000000000000000        1.0500000000000000   1
STR_C        1.0000000000000000       1.0000000000000000        1.0500000000000000   1
STR_W        0.10000000000000000E-01  1.0000000000000000      100.0000000000000000   0
ATO01          1    2
OKK_I       0.10000000000000000E-01   1.0000000000000000        1.0000000000000000   0
OKK_O       0.10000000000000000E-01   1.0000000000000000        1.0000000000000000   0
OKK_L        1.0000000000000000       100.00000000000000        1000.0000000000000   0
BTH_I       0.10000000000000000E-01   1.0000000000000000        10.000000000000000   1
BTH_O       0.10000000000000000E-01   1.0000000000000000        10.000000000000000   1
BTH_L        1.0000000000000000       100.00000000000000        1000.0000000000000   1
ATO02          1    2
OKK_I       0.10000000000000000E-01  1.0000000000000000         1.0000000000000000   0
OKK_O       0.10000000000000000E-01  1.0000000000000000         1.0000000000000000   0
OKK_L        1.0000000000000000      100.00000000000000         1000.0000000000000   0
BTH_I       0.10000000000000000E-01  1.0000000000000000         10.000000000000000   1
BTH_O       0.10000000000000000E-01  1.0000000000000000         10.000000000000000   1
BTH_L        1.0000000000000000      100.00000000000000         1000.0000000000000   1
```

Figure 3: Example of .par file used to supply parameter lower bounds, values and upper bounds. The final column is a flag that indicates which parameters can be refined and which not. In this example, the .par file refers to a DB04 database structure, for which only one flag is set to 0.

```
STcod         1
VALn1         0
AV1LN       3.0000000000000000       5.0000000000000000      10.0000000000000000   1
SD1LN       2.0000000000000000       3.0000000000000000      5.00000000000000000   1
AV2LN       0.0000000000000000       0.0000000000000000      0.00000000000000000   0
SD2LN       0.0000000000000000       0.0000000000000000      0.00000000000000000   0
PHILN       0.0000000000000000       0.0000000000000000      0.0000000000000000    0
STR_i       0.9500000000000000       1.0000000000000000       1.0500000000000000   1
STR_1       0.9500000000000000       1.0000000000000000       1.0500000000000000   1
STR_C       1.0000000000000000       1.0000000000000000       1.0500000000000000   0
STR_W       0.10000000000000000E-01  1.0000000000000000     100.0000000000000000   0
AT001         1    1
OKK_I      0.10000000000000000E-01   1.0000000000000000       1.0000000000000000   0
OKK_0      0.10000000000000000E-01   1.0000000000000000       1.0000000000000000   0
OKK_L       1.0000000000000000       100.00000000000000       1000.0000000000000   0
BTH_I      0.10000000000000000E-01   1.0000000000000000       10.000000000000000   1
BTH_0      0.10000000000000000E-01   1.0000000000000000       10.000000000000000   1
BTH_L       1.0000000000000000       100.00000000000000       1000.0000000000000   1
AT002         1    1
OKK_I      0.10000000000000000E-01  1.0000000000000000        1.0000000000000000   0
OKK_0      0.10000000000000000E-01  1.0000000000000000        1.0000000000000000   0
OKK_L       1.0000000000000000      100.00000000000000        1000.0000000000000   0
BTH_I      0.10000000000000000E-01  1.0000000000000000        10.000000000000000   1
BTH_0      0.10000000000000000E-01  1.0000000000000000        10.000000000000000   1
BTH_L       1.0000000000000000      100.00000000000000        1000.0000000000000   1
```

Figure 4: Example of .par files used to supply parameter lower bounds, values and upper bounds for a DB03 database structure.

## 5.3 Refinement file example

```
 Number of stages #  1
 stage #  1 0.0000001 COMPLEX
 %1
! 1 2 3 4 5 6
  1 0 0 1 0 1
 #1
  1
```

Figure 5: Example of .ref file used to supply refinement codes for bivariate-log-normal size parameters: first code; lattice expansion: second code; for site occupation factors: third and fifth codes; and thermal factor of atoms: fourth and sixth codes; within one single stage for a single structure with two atomic species and a single dataset input. The Simplex optimization algorithm is required.

```
 Number of stages #  1
 stage #  1 0.0000001 ANNEAL
 %1
! 1 2 3 4 5 6
  1 0 1 1 1 1
 #1
  1
```

Figure 6: Example of .ref file used to supply refinement codes (1) as in the previous example plus the global refinement flag for occupancy. The Simulated Annealing optimization algorithm is required.

```
 Number of stages #  1
 stage #  1 0.0000001 COMPLEX
 %1
! 1 2 3 4
  0 0 0 1
 %2
! 1 2 3 4
  0 0 0 1
 #2
  1 1
```

Figure 7: Example of .ref file used to supply refinement codes (1) for correcting the thermal factor of atoms (code 4) within one single stage, for a double structure (both monoatomic) and a double dataset input.

```
  Number of stages #  3
 stage #  1 0.0001 COMPLEX
 %1
! 1 2 3 4 5 6
  0 1 0 0 0 0
 #1
  1
 stage #  2 0.0001 COMPLEX
 %1
! 1 2 3 4 5 6
  1 0 1 1 1 1
 #1
  1
 stage #  3 0.0001 COMPLEX
 %1
! 1 2 3 4 5 6
  1 1 1 1 1 1
 #1
  1
```

Figure 8: Example of .ref file used to supply refinement codes (1) in a multiple refinement stage.

## 6 Output files

Summary of the main output files created by `Debussy`:

1) `OUTPUT_FILE_Best.dis` or `filename_Best.dis`: Extended output of all interesting results and derived statistics, *e.g.* average size, distributions, weight fractions. The `filename.dwa` is used as root for this output, in case **SIMUL_FLAG** is 0 in Refinement/Output Section.

2) At the end of each refinement run a refinement summary file named **REFINEMENT_FILE**`_ref.sum` is created containing information about the last refinement cycle.

3) In case **SIMUL_FLAG** is -1 at the end of the refinement run a file named `refinement.std` is created containing the refined parameters and estimates for their standard deviations as well as the upper diagonal of the full covariance matrix.

For each structure:

4) At the end of each refinement run, refined parameter values will be saved in a newly created file named `STRUCTURE_NAME0`$m$`_Best.par`, where $m$ is the progressive number assigned to the structure in **STRUCTURE_NAME**. The same kind of files are also created after each refinement stage.

5) At the end of each (simulation or refinement) run, a file named `STRUCTURE_NAME0`$m$`_plot1D.mtx` (for DB02 and DB03) or `STRUCTURE_NAME0`$m$`_plot2D.mtx` is created for each structure. It contains information for each single NC of the family (size, mass, volume, etc.). $m$ is the progressive number assigned to the structure in **STRUCTURE_NAME**.

For each dataset:

6) `OUTPUT_FILE_TECHNIQUE`$n$`.cal` or `DATASET_NAME_Best.cal` : Calculated patterns for every dataset ($n$ is the progressive number in the dataset section). The content depends on the value of **CALC_FLAG** in Refinement/Output Section.
`OUTPUT_FILE_TECHNIQUE#`$n$`.rpdf` or `DATASET_NAME_Best.rpdf`: Calculated pair distribution functions for observed, total calculated and background patterns - experimental feature.

# A Log-normal distributions

## A.1 One-dimensional log-normal distributions. Choice of the power.

Lognormal distributions (unnormalized form) we define as

$$F_k(x; x_0, w) = x^k \mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2}{w^2}}$$

The 'standard' lognormal is

$$F(x; x_0, w) \equiv F_{-1}(x; x_0, w) = \frac{1}{x}\ \mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2}{w^2}}$$

Note that - as $x = \mathrm{e}^{\log(x)}$ - for any $k$ we can write

$$F_k(x; x_0, w) = x^k \mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2}{w^2}} = \frac{1}{x}\mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2-2(k+1)w^2\log(x)}{w^2}}$$

and completing the square at exponent

$$\begin{aligned} F_k(x; x_0, w) &= \frac{1}{x}\mathrm{e}^{-\frac{1}{2}\frac{\left(\log(x)-\left[\log(x_0)+(k+1)w^2\right]\right)^2}{w^2}}\ \mathrm{e}^{\frac{1}{2}w^2(k+1)^2+(k+1)\log(x_0)} \qquad (17) \\ &= \left[x_0^{k+1}\mathrm{e}^{\frac{1}{2}w^2(k+1)^2}\right]\ \frac{1}{x}\ \mathrm{e}^{-\frac{1}{2}\frac{\left(\log(x)-\log\left(x_0\mathrm{e}^{(k+1)w^2}\right)\right)^2}{w^2}} \end{aligned}$$

that is

$$F_k(x; x_0, w) = \left[x_0^{k+1}\mathrm{e}^{\frac{1}{2}w^2(k+1)^2}\right]\ F\left(x;\ x_0\mathrm{e}^{(k+1)w^2}, w\right)$$

## A.2 Normalization, moments, average/standard deviation

We can evaluate the first moments of lognormal distributions:

$$m_{0;k} = \int_0^{+\infty} \mathrm{d}x\ F_k(x; x_0, w) = \mathrm{e}^{\frac{1}{2}(k+1)^2w^2} w x_0^{k+1}\sqrt{2\pi} \qquad (18)$$

$$m_{1;k} = \int_0^{+\infty} \mathrm{d}x\ xF_k(x; x_0, w) = \mathrm{e}^{\frac{1}{2}(k+2)^2w^2} w x_0^{k+2}\sqrt{2\pi} \qquad (19)$$

$$m_{2;k} = \int_0^{+\infty} \mathrm{d}x\ x^2F_k(x; x_0, w) = \mathrm{e}^{\frac{1}{2}(k+3)^2w^2} w x_0^{k+3}\sqrt{2\pi} \qquad (20)$$

In particular, for the standard $k = -1$ choice, we have

$$m_0 \equiv m_{0;-1} = w\sqrt{2\pi}$$

so the normalized form is

$$\widehat{F}(x; x_0, w) \equiv \frac{1}{m_{0;-1}}F_{-1}(x; x_0, w) = \frac{1}{xw\sqrt{2\pi}}\ \mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2}{w^2}}$$

Similarly, the normalized general-$k$ form is

$$\widehat{F}_k(x; x_0, w) = \frac{x^k\mathrm{e}^{-\frac{1}{2}(k+1)^2w^2}}{wx_0^{k+1}\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\frac{(\log(x)-\log(x_0))^2}{w^2}}$$

Now we can evaluate the first moments of normalized lognormal distributions:

$$\begin{aligned}
\widehat{m}_{0;k} &= \int_0^{+\infty} \mathrm{d}x\ \widehat{F}_k(x;x_0,w) = 1 && (21)\\
\widehat{m}_{1;k} &= \int_0^{+\infty} \mathrm{d}x\ x\widehat{F}_k(x;x_0,w) = \mathrm{e}^{\frac{2k+3}{2}w^2} x_0 && (22)\\
\widehat{m}_{2;k} &= \int_0^{+\infty} \mathrm{d}x\ x^2\widehat{F}_k(x;x_0,w) = \mathrm{e}^{(2k+4)w^2} x_0^2 && (23)
\end{aligned}$$

and for the standard $k=-1$ choice:

$$\begin{aligned}
\widehat{m}_{0} &= \int_0^{+\infty} \mathrm{d}x\ \widehat{F}(x;x_0,w) = 1 && (24)\\
\widehat{m}_{1} &= \int_0^{+\infty} \mathrm{d}x\ x\widehat{F}(x;x_0,w) = \mathrm{e}^{\frac{1}{2}w^2} x_0 && (25)\\
\widehat{m}_{2} &= \int_0^{+\infty} \mathrm{d}x\ x^2\widehat{F}(x;x_0,w) = \mathrm{e}^{2w^2} x_0^2 && (26)
\end{aligned}$$

So, averages $\langle x\rangle$ and standard deviations $\sigma$ are given by

$$\begin{cases}
\langle x\rangle = \widehat{m}_{1;k} & = \mathrm{e}^{\frac{2k+3}{2}w^2} x_0\\
\sigma = \left[\widehat{m}_{2;k} - \widehat{m}_{1;k}^2\right]^{1/2} & = \langle x\rangle\sqrt{\mathrm{e}^{w^2}-1}
\end{cases}$$

and for the standard case $k=-1$:

$$\begin{cases}
\langle x\rangle = \widehat{m}_{1} & = \mathrm{e}^{\frac{1}{2}w^2} x_0\\
\sigma = \left[\widehat{m}_{2} - \widehat{m}_{1}^2\right]^{1/2} & = \langle x\rangle\sqrt{\mathrm{e}^{w^2}-1}
\end{cases} \qquad (27)$$

Inverting the equations, we obtain (general $k$):

$$\begin{cases}
w = \sqrt{\log\left(1+\dfrac{\sigma^2}{\langle x\rangle^2}\right)}\\
x_0 = \dfrac{\langle x\rangle}{\left(1+\dfrac{\sigma^2}{\langle x\rangle^2}\right)^{k+\frac{3}{2}}}
\end{cases}$$

and for the standard case $k=-1$:

$$\begin{cases}
w = \sqrt{\log\left(1+\dfrac{\sigma^2}{\langle x\rangle^2}\right)}\\
x_0 = \dfrac{\langle x\rangle}{\left(1+\dfrac{\sigma^2}{\langle x\rangle^2}\right)^{\frac{1}{2}}}
\end{cases} \qquad (28)$$

Note:
- For given $\langle x\rangle, \sigma$, $w$ is independent of the $k$ choice, while $x_0$ changes;
- For given $x_0, w$, $\sigma/\langle x\rangle$ is independent of the $k$ choice, while $\langle x\rangle$ changes.

## A.3 Bivariate

We can define a bivariate lognormal - w.r.t. its eigenaxes - as the product of two univariate lognormals. Now we give the two separate average/st.dev. pairs, denoting them

$$(a_1, s_1); \qquad (a_2, s_2)$$

and the conjugate variable pairs (center/width) as

$$(c_1, w_1); \qquad (c_2, w_2)$$

so the LN looks like

$$LN_0(x, y) = \frac{e^{-\frac{1}{2}[\log(x)-\log(c_1)]^2/w_1^2}\, e^{-\frac{1}{2}[\log(y)-\log(c_2)]^2/w_2^2}}{2\pi x y w_1 w_2}$$

Things become a bit more complex when the axes are rotated. First we have to define the log-axes:

$$u = \log(x); \qquad v = \log(y).$$

In the log-axes, we have a normal distribution:

$$LN_0(u, v) = \frac{e^{-\frac{1}{2}(u-\log(c_1))^2/w_1^2}\, e^{-\frac{1}{2}(v-\log(c_2))^2/w_2^2}}{2\pi w_1 w_2 e^{u+v}}$$

Suppose that a distribution is a LN referred to rotated log-axes (unit vectors)

$$u' \to (\cos(\phi), \sin(\phi)); \qquad v' \to (-\sin(\phi), \cos(\phi))$$

corresponding to a counterclockwise rotation. The transformation is

$$\begin{cases} u' &= u\cos(\phi) + v\sin(\phi) \\ v' &= v\cos(\phi) - u\sin(\phi) \end{cases}$$

or in matrix form

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

We have also to transform the center:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} \log(c_1) \\ \log(c_2) \end{pmatrix}$$

Now it should suffice to substitute $u \to u'$, $v \to v'$ in the former expression.

$$LN_\phi(u, v) = \frac{e^{-\frac{1}{2}(u'-z_1)^2/w_1^2}\, e^{-\frac{1}{2}(v'-z_2)^2/w_2^2}}{2\pi w_1 w_2 e^{u'+v'}} \quad \Bigg| \quad u' = u\cos(\phi) + v\sin(\phi),\; v' = v\cos(\phi) - u\sin(\phi)$$

IMPORTANT: The input parameters for `Debussy` [file `.par`] are those hereby defined, namely the averages and stadard deviations referred to the log-axes $(a_1, s_1)$; $(a_2, s_2)$, and the angle $\phi$ (in deg.). Also the center/width pairs $(c_1, w_1)$; $(c_2, w_2)$ - derived from the former *via* Eq. (28), see the routine `avsd2cenlar`, are used to speed up the calculation. These parameters are intuitive but otherwise devoid of meaning when $\phi \neq 0$.

We can write the exponent in matrix form: call

$$T \equiv \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}$$

and

$$A \equiv \begin{pmatrix} \frac{1}{2w_1^2} & 0 \\ 0 & \frac{1}{2w_2^2} \end{pmatrix}; \qquad \boldsymbol{U} \equiv \begin{pmatrix} u - \log(c_1) \\ v - \log(c_2) \end{pmatrix}$$

then

$$LN_\phi(u,v) = \frac{\mathrm{e}^{-\boldsymbol{U}\cdot T^t A T \boldsymbol{U}}}{2\pi w_1 w_2}$$

## A.4 Projected statistical informations

After so doing, we can calculate the projected averages:

$$\begin{aligned} \langle x \rangle &= \int_0^{+\infty} \mathrm{d}x \int_0^{+\infty} \mathrm{d}y \; x \; LN_\phi(u,v) && (29) \\ &= \int_{-\infty}^{+\infty} \mathrm{d}u' \int_{-\infty}^{+\infty} \mathrm{d}v' \; \mathrm{e}^{u'\cos(\phi) - v'\sin(\phi)} \; LN_0(u',v') && (30) \\ &= c_1 \; \mathrm{e}^{\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]/4} && (31) \end{aligned}$$

where the inverse coordinate transformation has been used. Similarly, we obtain

$$\langle y \rangle = c_2 \; \mathrm{e}^{\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]/4}$$

Now we examine the second moments:

$$\begin{aligned} \langle x^2 \rangle &= \int_0^{+\infty} \mathrm{d}x' \int_0^{+\infty} \mathrm{d}y' \; \mathrm{e}^{2(u'\cos(\phi) - v'\sin(\phi))} \; LN_0(x',y') && (32) \\ &= c_1^2 \; \mathrm{e}^{\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)} && (33) \end{aligned}$$

and

$$\begin{aligned} \langle y^2 \rangle &= \int_0^{+\infty} \mathrm{d}x' \int_0^{+\infty} \mathrm{d}y' \; \mathrm{e}^{2(u'\sin(\phi) + v'\cos(\phi))} \; LN_0(x',y') && (34) \\ &= c_2^2 \; \mathrm{e}^{\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)} && (35) \end{aligned}$$

and

$$\begin{aligned} \langle xy \rangle &= \int_0^{+\infty} \mathrm{d}x' \int_0^{+\infty} \mathrm{d}y' \; \mathrm{e}^{v'\cos(\phi) + u'\sin(\phi)} \mathrm{e}^{u'\cos(\phi) - v'\sin(\phi)} \; LN_0(x',y') && (36) \\ &= c_1 c_2 \; \mathrm{e}^{\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\sin(2\phi)\right]/2} && (37) \end{aligned}$$

Now we have all we need to calculate variances and covariance:

$$\begin{aligned} V_{xx} &= \langle x^2 \rangle - \langle x \rangle^2 = c_1^2 \; \mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]} \left[\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]} - 1\right]; && (38) \\ V_{yy} &= \langle y^2 \rangle - \langle y \rangle^2 = c_2^2 \; \mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]} \left[\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]} - 1\right]; && (39) \\ V_{xy} &= \langle xy \rangle - \langle x \rangle \langle y \rangle = c_1 c_2 \; \mathrm{e}^{\frac{1}{2}\left(w_1^2+w_2^2\right)} \left[\mathrm{e}^{\frac{1}{2}\left(w_1^2-w_2^2\right)\sin(2\phi)} - 1\right]; && (40) \end{aligned}$$

and as usual define the standard deviations $(\sigma_1, \sigma_2)$ and the correlation cosine:

$$\begin{aligned}
\sigma_1 \equiv V_{xx}^{1/2} &= c_1\ \mathrm{e}^{\frac{1}{4}\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}\sqrt{\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}-1}\ ; && (41)\\
\sigma_2 \equiv V_{yy}^{1/2} &= c_2\ \mathrm{e}^{\frac{1}{4}\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}\sqrt{\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}-1}\ ; && (42)\\
\cos(\psi) \equiv \frac{V_{xy}}{\sigma_1\sigma_2} &= \frac{\mathrm{e}^{\frac{1}{2}\left(w_1^2-w_2^2\right)\sin(2\phi)}-1}{\sqrt{\left[\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}-1\right]\left[\mathrm{e}^{\frac{1}{2}\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]}-1\right]}}\ ; && (43)
\end{aligned}$$

and summarizing also the $x$-, $y$- averages (symbols $\alpha_1$, $\alpha_2$):

$$\begin{aligned}
\alpha_1 \equiv \langle x\rangle &= c_1\ \mathrm{e}^{\left[\left(w_1^2+w_2^2\right)+\left(w_1^2-w_2^2\right)\cos(2\phi)\right]/4}\ ; && (44)\\
\alpha_2 \equiv \langle y\rangle &= c_2\ \mathrm{e}^{\left[\left(w_1^2+w_2^2\right)-\left(w_1^2-w_2^2\right)\cos(2\phi)\right]/4}\ . && (45)
\end{aligned}$$

These quantities are supplied in output by `Debussy` as 'projected averages/standard deviations'.

In particular, for $\phi = 0$ we retrieve the familiar result (compare with formula at Eq. (27), replacing $x_0$ with $c_1, c_2$ and $w$ with $w_1, w_2$, $\langle x\rangle$ with $a_1, a_2$, $\sigma$ with $s_1, s_2$):

$$\begin{aligned}
\alpha_1 &= c_1\ \mathrm{e}^{w_1^2/2} = a_1\ ; && (46)\\
\alpha_2 &= c_2\ \mathrm{e}^{w_2^2/2} = a_2\ ; && (47)\\
\sigma_1^2 &= \left\langle x^2\right\rangle - \langle x\rangle^2 = c_1^2\ \mathrm{e}^{w_1^2}\left[\mathrm{e}^{w_1^2}-1\right] = s_1^2\ ; && (48)\\
\sigma_2^2 &= \left\langle y^2\right\rangle - \langle y\rangle^2 = c_2^2\ \mathrm{e}^{w_2^2}\left[\mathrm{e}^{w_2^2}-1\right] = s_2^2\ ; && (49)\\
V_{xy} &= \langle xy\rangle - \langle x\rangle\langle y\rangle = 0\ . && (50)
\end{aligned}$$

## A.5 Eigensystem: statistics on the physical axes

We can form the covariance matrix

$$C = \begin{pmatrix} V_{xx} & V_{xy} \\ V_{xy} & V_{yy} \end{pmatrix}$$

whose eigenvalues, eigenvectors are

$$\begin{aligned}
e_1 &= \frac{1}{2}\left(\sigma_1^2+\sigma_2^2-\tau\right)\ ; && (51)\\
\boldsymbol{v}_1 &= \frac{1}{\sqrt{2\tau}}\left(-\sqrt{\tau-\left(\sigma_1^2-\sigma_2^2\right)},\ \frac{2\,\sigma_1\sigma_2\cos(\psi)}{\sqrt{\tau-\left(\sigma_1^2-\sigma_2^2\right)}}\right)\ ; && (52)\\
e_2 &= \frac{1}{2}\left(\sigma_1^2+\sigma_2^2+\tau\right)\ ; && (53)\\
\boldsymbol{v}_2 &= \frac{1}{\sqrt{2\tau}}\left(+\sqrt{\tau+\left(\sigma_1^2-\sigma_2^2\right)},\ \frac{2\,\sigma_1\sigma_2\cos(\psi)}{\sqrt{\tau+\left(\sigma_1^2-\sigma_2^2\right)}}\right)\ . && (54)
\end{aligned}$$

where we set

$$\tau \equiv \sqrt{\sigma_1^4+\sigma_2^4+2\sigma_1^2\sigma_2^2\cos(2\psi)}$$

These eigen-quantities are supplied in output by `Debussy` as 'eigen-averages / eigen-standard deviations'.

## B Error estimation and propagation

TO DO ..

## References