ERIS — A Program for MEM analysis from X-Ray and Neutron Diffraction Data

Koichi MOMMA[1]
National Museum of Nature and Science,
4-1-1 Amakubo, Tsukuba, Ibaraki 305-0005, Japan

July 25, 2024

[1]E-mail: vesta.dev@gmail.com

Contents

LICENSE AGREEMENT

ERIS LICENSE

Version 0.99

ERIS is distributed free of charge but currently copyrighted with its source code not open to the public because we wish to control its development and future by ourselves.

Permission to use this software is hereby granted under the following conditions:

1. Whenever original results acquired with ERIS are published in journals, proceedings, and review articles, the program name "ERIS" should explicitly be stated. A paper about ERIS, describing its original features, is currently under preparation. Once it is published, it should be cited.

2. ERIS is provided "as is" without any expressed or implied warranty. We do not provide any technical support for free.

3. You should not redistribute any copy of the distributed files unless you have a written permission from us.

If you find a problem or a bug on use of ERIS, please let us know and help us improve it.

Chapter 1

INTRODUCTION

The maximum entropy method (MEM) [1,2] is very useful to modify structural models adopted in Rietveld analysis and determine distributions of electron and nuclear (coherent-scattering length, b_c) densities from X-ray and neutron diffraction data, respectively [3]. The termination effect is less serious in MEM analysis than in Fourier/D synthesis because MEM is capable of estimating structure factors of high-Q reflections that have not been measured experimentally. In addition, after observed structure factors of overlapped reflections have been estimated on the basis of results of Rietveld analysis, subsequent MEM analysis more or less improves structure factors, which makes MEM very effective in the extraction of the maximum amount of structural information from powder diffraction data.

Three-dimensional (3D) visualization of electron/nuclear densities resulting from diffraction data achieves a better understanding of the highly disordered structure, chemical bonding (X-ray diffraction), or anharmonic thermal motion. For example, diffusion paths of mobile chemical species in ionic conductors have been visualized in three dimensions from X-ray and neutron powder diffraction data by a sophisticated structure-refinement technique called MEM-based pattern fitting (MPF) [3–6]. MPF may be able to reveal even anharmonic thermal vibrations, which are very difficult to analyze from powder diffraction data because of many parameters to be refined by a nonlinear least-squares method.

ERIS is a successor of MEM analysis programs Dysnomia [6–8], PRIMA, and ALBA [3,9]. PRIMA was a fortran program for MEM analysis of electron and nuclear density from X-ray and neutron diffraction data. ALBA was a fortran program for MEM analysis of the Patterson function from X-ray and neutron diffraction data to estimate intensities of unobserved and/or individual reflections of overlapped peaks in powder diffraction. PRIMA was first rewritten from scratch in the C++ language and released as Dysnomia in 2011.

Dysnomia provides the following features which were not supported in PRIMA:

1. linear combination of the generalized F and G constraints,
2. weightings based on lattice-plane spacings.
3. new implementation of optimization algorithms (a variant of the Cambridge algorithm [10] and the limited-memory BFGS algorithm [11]) that converge to solutions close to the true maximum-entropy conditions,
4. automatic selection of fast Fourier transform (FFT) and discrete Fourier transform (DFT),
5. parallel processing using an API (application programming interface), OpenMP, for multi-platform shared-memory parallel programming,

ERIS integrates the features of Dysnomia and ALBA, with a couple of new and unique features.

1. a new type of weighting function, proportional to exponential of d-spacing,
2. a feature to automatically determine the best weighting parameters,
3. implemented the maximum entropy Patterson method (MEP),
4. automatically impose constraints of extinction conditions in MEP analysis,
5. generate an input file of Superflip after convergence of MEP analysis to solve crystal structure by the charge flipping method.

ERIS is the name of the Greek goddess of strife and discord. She is mother of Dysnomia, who is also the goddess of discord. Since the lawlessness state stands for higher-entropy state compared to ordered state, it is suitable for the name of the MEM analysis program, in which the information entropy, S, is maximized under some constraints. The program name ERIS also stands for Entropy-based Real-space Inference System.

Chapter 2

THEORETICAL BACKGROUND

2.1 The Fundamental Principle of MEM in Crystallography

The general principle of MEM analysis is to find the maximum of the information entropy, S, under several constraints by iterative procedures. A few variations of the MEM formalism have hitherto been adopted in the literature [1,12,13]. Here, we will follow the formalism of Collins [1] based on Jaynes's expression of S [14].

In MEM analysis from X-ray and neutron diffraction data, electron and nuclear densities in the unit cell are represented by those in voxels (parallelepipeds) whose numbers along a, b, and c axes are N_a, N_b, and N_c, respectively; densities within a voxel is regarded as even. Let N ($= N_a N_b N_c$) be the total number of voxels in the unit cell, ρ_k the normalized density at position r_k in the 3D gridded space, τ_k the normalized density derived from prior information for r_k, and ρ_k^* the density at r_k. Then, S is computed by

$$S = -\sum_{k=1}^{N} \rho_k \ln\left(\frac{\rho_k}{\tau_k}\right) \tag{2.1}$$

with

$$\rho_k = \frac{\rho_k^*}{\sum_{k=1}^{N} \rho_k^*}. \tag{2.2}$$

S is maximized under constraints C by a method of undetermined Lagrange multipliers,

$$Q = S - \lambda C - \mu\left(\sum_{k=1}^{N} \rho_k - 1\right), \tag{2.3}$$

where Q is maximized through iterative computations with respect to ρ and two Lagrange multipliers, λ and μ.

At the optimum solution of MEM, partial derivatives of Eq. (2.3) with respect to any variable should be 0. Therefore, the entropy and constraint should fulfill the following condition:

$$\frac{\partial S}{\partial \rho_k} = \lambda \frac{\partial C}{\partial \rho_k} + \mu. \tag{2.4}$$

Alternatively, they should fulfill the equivalent set of equations in reciprocal space:

$$\frac{\partial S}{\partial F_i} = \lambda \frac{\partial C}{\partial F_i}. \tag{2.5}$$

2.2 Optimization Algorithms

The following MEM algorithms are implemented in ERIS:

1. 0th-order single pixel approximation (ZSPA) algorithm [15] .
2. Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [11].

Electron or nuclear densities determined with these algorithms are similar to each other in most cases. However, solutions obtained with the ZSPA algorithm do not satisfy true maximum-entropy (MaxEnt) conditions [16], whereas the L-BFGS algorithm always gives solutions close to the true MaxEnt ones. Another algorithm, the Cambridge algorithm also gives solutions close to the true MaxEnt conditions [16]. Although a variant of the Cambridge algorithm was implemented in Dysnomia, it was removed in ERIS because the L-BFGS algorithm is much faster than the Cambridge algorithm while the two algorithm give the same solution.

2.2.1 ZSPA algorithm

In the ZSPA algorithm, ρ_k in the $(i+1)$-th iteration is approximated by

$$\rho_k^{(i+1)} = \frac{\rho_k^{(i)}}{Z} \exp\left(-\lambda \cdot \frac{\partial C}{\partial \rho_k^{(i)}}\right) \tag{2.6}$$

with

$$Z = \sum_{k=1}^{N_\mathrm{v}} \exp\left(-\lambda \cdot \frac{\partial C}{\partial \rho_k^{(i)}}\right). \tag{2.7}$$

Convergence criterion of the ZSPA algorithm is $C = 0$. Since this algorithm is quite simple and easy to implement, it is widely utilized in almost all programs for MEM analysis of electron/nuclear density from diffraction data. However, note that this algorithm does not actually maximize the information entropy S.

2.2.2 L-BFGS algorithm

The L-BFGS method [11] is a variant of quasi-Newton algorithms, requiring significantly less memory than other quasi-Newton methods. Instead of storing a dense $n \times n$ approximation (n: the number of variables in the problem) to the inverse Hessian matrix, L-BFGS stores only a few vectors that represent the approximation implicitly. Let $d_k = H_k g_k$ be the search direction at k-th iteration, where H_k is the inverse Hessian matrix of the function, $Q(\rho)$, to be maximized, $g_k \equiv \nabla Q(\rho)$ is the gradient of Q, $s_k = \rho_{k+1} - \rho_k$, $y_k = g_{k+1} - g_k$, and $r_k = 1/y_k^T s_k$. Then, d_k is computed as follows:

$q = g_k$
For $(i = k-1;\ i >= k-m;\ i--)\{$
$\quad \alpha_i = r_i s_i^T q$
$\quad q = q - \alpha_i y_i$
$\}$
$H_k^0 = \dfrac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}$
$d = H_k^0 q$

For $(i = k-m;\ i <= k-1;\ i++)\{$
$\quad \beta_i = r_i y_i^T d$
$\quad d = d + s_i(\alpha_i - \beta_i)$
$\}$

The initial approximate of the inverse Hessian matrix, H_k^0, is represented as a diagonal matrix. The Lagrange multipliers, λ and μ in Eq. (2.3), cannot be optimized simultaneously with ρ during the L-BFGS iterations. Then, in the L-BFGS method, λ and μ are gradually changed, and L-BFGS optimizations are repeated until the two constraints, *i.e.*, Eqs. (2.2) and (3.15), are fulfilled to achieve convergence. The convergence of the L-BFGS algorithm is judged by

$$\sqrt{\sum_{k=1}^{N_\mathrm{V}} \rho_k \left(\frac{\partial Q}{\partial \rho_k}\right)^2} < \varepsilon \left\langle \frac{\partial S}{\partial \rho_k} \right\rangle, \tag{2.8}$$

where ε is a small threshold value, which is set at 1×10^{-3} by default.

2.2.3 Cambridge algorithm

A variant of the Cambridge algorithm [10] was implemented in Dysnomia but removed from ERIS in favor of the L-BFGS algorithm. Both algorithms converges to the optimal MEM solution but L-BFGS algorithm is much faster. A brief summary of the Cambridge algorithm is given below just for the record.

The Cambridge algorithm adopts local quadratic approximation to Eq. (2.3). For simplicity, Dysnomia utilized only the diagonal elements of the Laplacian, $\nabla^2 Q$, and approximate $Q(\rho+\Delta\rho)$ by

$$Q(\rho + \Delta\rho) = Q(\rho) + \Delta\rho \left(\frac{\partial Q}{\partial \rho}\right) + \frac{1}{2}\Delta\rho^2 \left(\frac{\partial^2 Q}{\partial \rho^2}\right). \tag{2.9}$$

Then, Q can be optimized by

$$\Delta\rho_k = -\left(\frac{\partial Q}{\partial \rho_k}\right)\left(\frac{\partial^2 Q}{\partial \rho_k^2} + \alpha \cdot \frac{\partial^2 S}{\partial \rho_k^2}\right)^{-1}, \tag{2.10}$$

where α (≥ 0) is the damping factor to ensure that $\rho + \Delta\rho$ remains in the trust region of the quadratic approximation [10]. Convergence is judged by comparing the gradient of Eq. (2.3) with a small threshold value, ε:

$$\frac{1}{N_\mathrm{v}} \sum_{k=1}^{N_\mathrm{V}} \left(\frac{\partial Q}{\partial \rho_k}\right)^2 < \varepsilon. \tag{2.11}$$

2.3 Fourier Transform Algorithms

In MEM analysis, $F(\boldsymbol{h}_j)$'s are calculated by Fourier transform, which is very time-consuming, from electron or nuclear densities, with a result that the CPU time for MEM analysis generally depends on both N and N_F. FFT makes it possible to process huge amounts of data with very large values of N and N_F at high speed. Therefore, ERIS benefits MEM analyses of crystalline materials with large unit cells and low symmetry. ERIS is also very suitable for analyzing neutron diffraction data where nuclear densities tend to be confined in narrower spaces.

In ERIS, FFT is selected if

$$N_{\mathrm{a}} N_F n_{\mathrm{c}} > \frac{100N}{\ln N}, \tag{2.12}$$

where N_{a} is the total number of voxels in the asymmetric unit, and n_{c} is a factor to be selected depending on the presence or absence of an inversion center at the origin (centrosymmetric: 1, noncentrosymmetric: 2).

Parallel computation

For small-scale problems, a DFT routine making full use of multi-threaded parallel processing with the OpenMP technology is automatically selected to avoid overhead caused by FFT. Thus, in both FFT and DFT, the processing ability of ERIS increases in a nearly linear fashion with increasing number of CPU cores on the use of recent multi-core CPUs.

Chapter 3

CONSTRAINTS

With ERIS, S in Eq. (2.1) is maximized by imposing a linear combination of the generalized F and G constraints.

3.1 *F* Constraint

Let N_F be the total number of reflections, $F_{\rm o}(\boldsymbol{h}_j)$ the observed structure factor, $F(\boldsymbol{h}_j)$ the calculated structure factor, and $\sigma(\boldsymbol{h}_j)$ the standard uncertainty of $|F_{\rm o}(\boldsymbol{h}_j)|$. The F constraint used in MEM analysis from diffraction data is formulated as

$$C'_F = \sum_{j=1}^{N_F} |\Delta F_j|^2 - N_F, \tag{3.1}$$

where ΔF_j is the normalized residual for the structure factor:

$$\Delta F_j = \frac{F_{\rm o}(\boldsymbol{h}_j) - F(\boldsymbol{h}_j)}{\sigma(\boldsymbol{h}_j)}. \tag{3.2}$$

In the actual implementation of the F constraint in ERIS, C'_F is normalized with respect to N_F:

$$C_F = \frac{C'_F}{N_F} \tag{3.3}$$

$$= \frac{1}{N_F} \sum_{j=1}^{N_F} |\Delta F_j|^2 - 1. \tag{3.4}$$

3.2 *G* Constraint

Information about overlapped reflections in powder diffraction data can be introduced into MEM analysis through the G constraint [17]:

$$C'_G = \sum_{j=1}^{N_G} \left| \frac{G_{{\rm o}j} - G_j}{\sigma_j} \right|^2 - N_G$$

$$= \sum_{j=1}^{N_G} |\Delta G_j|^2 - N_G \tag{3.5}$$

with

$$G_j = \left[\frac{\sum_{i=1}^{L_j} m_i \left| F(\boldsymbol{h}_i) \right|^2}{\sum_{i=1}^{L_j} m_i} \right]^{1/2} . \tag{3.6}$$

In Eqs. (3.5) and (3.6), N_G is the total number of groups comprising overlapped reflections, $G_{\mathrm{o}j}$ is the sum of the integrated intensities of overlapped reflections in group j, L_j is the number of overlapped reflections in group j, and m_i is the multiplicity of reflection i. Combination of C'_F and C'_G affords an integrated constraint, C_2:

$$C_2 = \frac{C'_F + C'_G}{N_F + N_G}. \tag{3.7}$$

3.3 Generalized Constraint

The classical F and G constraints are based on χ^2 statistics, whose use assumes that experimental errors in $|F_{\mathrm{o}}(\boldsymbol{h}_j)|$'s are random with a Gaussian (normal) distribution. Although the use of χ^2 constraint is justified by the Gaussian distribution of errors, it is not a sufficient condition to ensure the Gaussian distribution of residuals between observed and estimated values. In the case of MEM analysis from diffraction data, some low-Q reflections tend to have very large $|\Delta F_j|$'s, which leads to a noisy distribution of electron or nuclear densities estimated by MEM. Such a characteristic in MEM analysis from diffraction data was theoretically explained by Jauch [18]. To reduce such an undesirable tendency of MEM, Palatinus and van Smaalen [19] proposed to use higher order moments of ΔF_j as a constraint.

A probability distribution of a random variable x is characterized by the values of its central moments M_n. For the normalized Gaussian distribution, the central moments are defined as

$$M_n(\mathrm{Gauss}) = \int_{-\infty}^{\infty} x^n (2\pi)^{-1/2} \exp\left(-x^2/2\right) \mathrm{d}x. \tag{3.8}$$

The moments of odd n are all zero while those of even n are

$$M_{2n}(\mathrm{Gauss}) = \prod_{i=1}^{n} (2i-1). \tag{3.9}$$

In the case of N samples of the variable x, the central moments M_n can be computed by

$$M_n = \frac{1}{N} \sum_{i=1}^{N} x_i^n. \tag{3.10}$$

The generalized F constraint of order n is formulated as

$$C'_{Fn} = \frac{1}{M_n(\mathrm{Gauss})} \sum_{j=1}^{N_F} |\Delta F_j|^n - N_F, \tag{3.11}$$

$$C_{Fn} = \frac{C'_{Fn}}{N_F}. \tag{3.12}$$

The classical F constraint corresponds to the generalized constraint of order 2, C_{F2}. The classical G constraint is also generalized as

$$C'_{Gn} = \frac{1}{M_n(\mathrm{Gauss})} \sum_{j=1}^{N_F} |\Delta G_j|^n - N_G. \tag{3.13}$$

The generalized F and G constraints of order n, are combined to give a C_n constraint:

$$C_n = \frac{C'_{Fn} + C'_{Gn}}{N_F + N_G}. \tag{3.14}$$

3.4 Linear Combination of Generalized Constraints

ERIS adopts a linear combination of the generalized F and G constraints with relative weights, λ_n ($n = 2, 4, 6, ...$):

$$C = \sum_n \lambda_n C_n \tag{3.15}$$

with

$$C_n = \frac{1}{(N_F + N_G) M_n(\text{Gauss})} \left[\sum_{j=1}^{N_F} w_j \left(|\Delta F_j|\right)^n + \sum_{j=1}^{G_F} w_j \left(|\Delta G_j|\right)^n \right] - C_{\text{w}_n}, \tag{3.16}$$

where w_j is the weighting factor (see 4), and C_{w_n} is the criterion for convergence. When w_j is unity, the ideal constraint is $C_{\text{w}_n} = 1$ for any even n. Such a rigorous constraint seems to be hardly satisfied in actual problems. Accordingly, ERIS always determines C_{w_n} automatically so as to satisfy Eq. (3.4) or (3.7) on use of the G constraint.

Information about central moments of the normalized residuals, ΔF_j and ΔG_j, is output to file *.out, for example,

```
C2  =  9.9992285E-01   ln(C2 ) =  -0.000077
C4  =  4.0802412E+00   ln(C4 ) =   1.406156
C6  =  2.8819103E+01   ln(C6 ) =   3.361038
C8  =  2.0165493E+02   ln(C8 ) =   5.306558
C10 =  1.2533774E+03   ln(C10) =   7.133597
C12 =  6.8066503E+03   ln(C12) =   8.825655
C14 =  3.2373854E+04   ln(C14) =  10.385106
C16 =  1.3593284E+05   ln(C16) =  11.819916
```

The first column, `C2`–`C16`, lists the central moments of order n for $|\Delta F_j|^n$ and $|\Delta G_j|^n$ normalized by those for Gaussian distribution. The second column, `ln(C2)`–`ln(C16)`, gives their natural logarithms.

If the distributions of ΔF_j and ΔG_j are Gaussian, values in the first column are all 1 while those in the second column are all 0. Normalized central moments larger than 1 imply that information in observed data is underestimated and not fully reconstructed by MEM. On the contrary, normalized central moments smaller than 1 imply the overestimation of data, which result in overfit to observed data.

In MEM analyses from X-ray diffraction data with traditional second-order constraints, the normalized central moments of order $n > 2$ are typically larger than 1, as shown in the above example. In such a case, increase λ_n of order 4 or higher to put restraints on them and bind them to the ideal value. The central moment of a higher order may be smaller than 1 depending on the type and quality of data, particularly when $N_F + N_G$ is relatively small. In such a case, set λ_n such that $\lambda_2 = 1$ and $\lambda_n = 0$ for $n > 2$, which corresponds to the traditional χ^2 constraint.

On the use of the ZSPA algorithm, substitution of Eq. (3.15) for C in Eq. (2.6) will give rise to a problem that only the C_n constraint with the highest order practically takes effect at the beginning of MEM iterations. This is because the highest order C_n constraint can be orders of

magnitude larger than the C_2 constraint at early cycles of MEM iterations. Because the ZSPA algorithm do not optimize the information entropy, and it is sensitive to calculation route, the final results will also be strongly constrained by the highest order C_n. To overcome such a problem specific to the ZSPA algorithm, λ_n in Eq. (3.15) in the i-th iteration, $\lambda_n^{(i)}$, is adjusted as follows. If ΔF_j's and ΔG_j's are kept close to Gaussian distribution during MEM iterations, the order of C_n in the (i)-th iteration is roughly equal to $(C_2)^{n/2}$. Then $\lambda_n^{(i)}$ is replaced by

$$\lambda_n^{(i)} = \frac{\lambda_n}{(C_2)^{n/2-1}}. \tag{3.17}$$

At any rate, be sure to check whether or not 3D distribution of electron/nuclear densities is physically and chemically reasonable with VESTA [20].

Chapter 4

WEIGHTING

4.1 Why weighting of observation is necessary

To obtain reasonable distribution of densities, it is reported that observed diffraction data need to be weighted with a proper function. Without weighting, residuals of observed and calculated structure factors, $(F_\mathrm{o}F_\mathrm{c})$, of low-$Q$ (large d-spacing) region tend to be very large.

The necessity of weighting in MEM for crystallography comes from intrinsic nature of crystals. Although the information theory assumes that the probabilities of individual events to occur are independent of each other, electron/nuclear densities or their Patterson functions in crystals are not random but smooth, having strong correlation with vicinal positions. Therefore, the left-hand values of Eq. (2.4) is also smooth in real space. In reciprocal space, such nature of crystals causes systematic decrease of structure factor amplitudes with decreasing d-spacing, and amplitudes of the left-hand values of Eq. (2.5) also decrease with decreasing d-spacing. On the other hand, the right-hand values of Eqs. (2.4) and (2.5) are functions of ΔF, which should not depend on d-spacing but reflect random errors of observations. MEM formalism forces these two different kinds of terms to be equivalent, and this is why a proper weighting is necessary to suppress systematic dependency of ΔF to d-spacing.

4.2 Weighting based on exponential of d-spacing

4.3 Weighting based on n-th power of d-spacing

Weighting in the F constraint on the basis of the lattice-plane spacing was first proposed by de Vries *et al* [21]. Its effectiveness was later confirmed by some other researchers including Hofmann *et al.* [22].

Let $\boldsymbol{s}_j$ be the reciprocal-lattice vector $(= h\boldsymbol{a}^* + k\boldsymbol{b}^* + l\boldsymbol{c}^*)$ for reflection j, x the real number for weighting, and d_j the lattice-plane spacing $(= 1/|\boldsymbol{s}_j|)$. Then, the weighting factors, w_j, in Eq. (3.16) is given by

$$\begin{aligned} w_j &= \frac{1}{|\boldsymbol{s}_j|^x}\left[\sum_{i=1}^{N_F}\frac{1}{|\boldsymbol{s}_i|^x}\right] \\ &= d_j^x\left[\sum_{i=1}^{N_F} d_i^x\right]. \end{aligned} \tag{4.1}$$

If $x = 0$, no weight is imposed in the same way as with PRIMA [9].

This weighting scheme is suitable for X-ray diffraction where parts of low-Q reflections often have very large ΔF_j values. Values of around 2 are usually recommended in X-ray powder diffraction; x is increased as the quality of intensity data is improved. In the case of single-crystal X-ray diffraction, de Vries *et al.* [21] empirically found $x = 4$ to be optimum. On the other hand, this approach is not suited for the analysis of neutron diffraction data, where b_c values of constituent elements remain constant regardless Q, particularly those measured at low temperature. Therefore, x is usually set at 0 in neutron diffraction.

Ad-hoc weighting described above will be effective, particularly, in MEM analysis from X-ray diffraction data, where $\sigma_j(\boldsymbol{h}_j)$'s tend to be estimated at unreasonably large values for low-Q reflections.

4.4 Weighting based on multiplicity of reflections

Chapter 5

ADJUSTMENT OF STANDARD UNCERTAINTIES

5.1 Powder Diffraction Data (*.fos)

In powder diffraction, $\sigma(\boldsymbol{h}_j)$'s are estimated on the basis of the law of propagation of errors in combination with counting statistics:

$$\sigma(\boldsymbol{h}_j) = \frac{|F_{\rm o}(\boldsymbol{h}_j)|}{2} \left\{ \left[\frac{1}{EI_{\rm o}(\boldsymbol{h}_j)} \right]^2 + \left[\frac{\sigma(s)}{s} \right]^2 \right\}^{\frac{1}{2}}, \tag{5.1}$$

where E is the factor to adjust $\sigma(\boldsymbol{h}_j)$, $I_{\rm o}(\boldsymbol{h}_j)$ is the observed integrated intensity, s is the scale factor, and $\sigma(s)$ is the standard uncertainty of s [6]. Both s and $\sigma(s)$ results from Rietveld analysis or whole-pattern fitting with RIETAN-FP. E is selected in such a way that electron- or nuclear-density distribution which is physically and chemically reasonable results from MEM analysis. Convenient bash scripts, MPF_multi.command, for automatic MPF analyses enable us to change E as specified in *.prf (see 6.2.2).

In the case of angle-dispersive-type powder diffraction where the step width, $\Delta 2\theta$, is usually constant, E is approximately equal to $1/\Delta 2\theta$ (unit: rad^{-1}), which is output as `E(SCIO)` in *.lst by RIETAN-FP[1] [23]. On the other hand, E depends on the step width, Δt, of the time-of-flight (TOF) in TOF neutron powder diffraction; a utility called Alchemy [24, 25] can be used to convert files output by GSAS and Fullprof into *.fos or *.mem (see 5.2).

5.2 Single-Crystal Data (*.mem)

The estimation of $\sigma(\boldsymbol{h}_j)$ is difficult even in single-crystal X-ray or neutron diffraction; $\sigma(\boldsymbol{h}_j)$'s are often underestimated or overestimated. With ERIS, $\sigma(\boldsymbol{h}_j)$'s can be changed with the adjustment factor, E, according to

$$\sigma'(\boldsymbol{h}_j) = \frac{\sigma(\boldsymbol{h}_j)}{\sqrt{E}} \tag{5.2}$$

without any conversion of $\sigma(\boldsymbol{h}_j)$'s in *.mem.

[1] http://fujioizumi.verse.jp/download/download_Eng.html

Chapter 6

INSTALLATION AND EXECUTION

6.1 Installation

ERIS runs on Windows, OS X, and Linux. URLs of their files are as follows:

Windows
http://jp-minerals.org/eris/archives/ERIS.zip

OS X, 64 bit application
http://jp-minerals.org/eris/archives/ERIS

Linux, 64 bit x86_64
http://jp-minerals.org/eris/archives/ERIS-x86_64.tar.bz2

Linux, 64 bit Arm64
http://jp-minerals.org/eris/archives/ERIS-arm64.tar.bz2

Once the above archive files are downloaded and extracted, the executable file "ERIS" (or ERIS.exe) can be directly executed without any additional installation process. To install ERIS to interoperate with RIETAN-FP, place the extracted files in a specific directory. On Windows, extract the zip files and move the resulting files to C:\Program Files\RIETAN_VENUS\. On macOS, mount ERIS.dmg by double-clicking its icon and move all the files to folder /Applications/RIETAN_VENUS/. Both versions can be run in combination with a shell script, MPF_multi.command, written in bash for MPF. MPF_multi.command and its manual are included in distribution files of the RIETAN-FP–VENUS package.

6.2 Execution

6.2.1 Interactive mode

Run ERIS by double-clicking its icon or entering the name of its executable binary file in a command line. Then, the program asks questions that must be answered by the user. In what follows, these questions are represented by boxed texts with gray background.

```
# The name of the MEM data set file, *.fos or *.mem.
```

Type the absolute path of an input file, for example, C:\users\user_name\sample\sample.fos (Windows).

```
# The type of the MEM data set file.
# 0. (a) X-ray diffraction data or (b) neutron diffraction data of a compound
#    containing no element with a negative bc value.
# 1. Neutron diffraction data of a compound containing at least one element
#    with a negative bc value.
```

This option is set for the backward compatibility of input files because the file format of *.mem is different depending on dataset types.

```
# From which densities will you start?
# 0. Uniform densities.
# 1. Restart from densities recorded in the 3D densities file (*.pgrid).
```

If you choose the second option, initial densities are input from *.pgrid. This option is useful when you restart MEM analysis after precedent iterations. Note that this option is different from the choice of prior density τ_k in Eq. (2.1) in strict sense, although on the use of ZSPA algorithm, use of non-uniform prior is approximated by restarting calculation from τ_k. For rigorous analyses using non-uniform prior, use L-BFGS or Cambridge algorithms and specify the prior data on the later question. MEM analysis can only be restarted from densities recorded in *.pgrid output by ERIS.

```
# Optimization algorithm.
# 0. 0th order single-pixel approximation (ZSPA).
# 1. The Limited-memory BFGS algorithm (L-BFGS).
# 2. The Cambridge algorithm (Obsolete).
```

Read 2.3 for details in the above two algorithms.

```
# Which initial Lagrangian multiplier will you use (LM_type)?
# 0. The value input by user.
# 1. The value calculated by ERIS.
# 2. The value written in the MEM data set file.
```

This question appears only on use of the ZSPA algorithm, and option 2 appears only when the format of the MEM dataset file is *.mem.

```
# The initial Lagrangian multiplier, lambda
```

This question is asked only when 0 is input in the previous question.

```
# A parameter x to impose weighting factors on the basis of
# lattice-plane spacing.
```

Parameter x is included in Eq. (4.1). Input $x = 0$ in classical MEM analysis where no weight is imposed in the F constraint. If $x > 0$, it is regarded as x in Eq. (4.1). As described in 4, typically $x = 0$ in neutron diffraction, $x = 1$–2 in X-ray powder diffraction, and $x = 4$ is

optimum in single-crystal X-ray diffraction.

```
# The coefficient, t, to adjust the Lagrangian multiplier.
```

This question appears only on use of the ZSPA algorithm. Input a value between 0.05 and 0.1 for fast computation. If λ is too large, MEM iterations diverge and never reach the solution. If λ is too small, the convergence of the MEM equation is ensured whereas the computation time may increase considerably, with no convergence achieved on specification of a relatively small maximum number of cycles. The coefficient t allows us to find the best λ value automatically through multiplying λ by $1 + t$ in every cycle of MEM iterations until λ becomes too large to converge the MEM equation, which is different from the manner of changing λ in PRIMA [9].

```
# The coefficient, E, to adjust estimated standard uncertainty of
# structure factors.
```

For details in E, see 5.1 and 5.2.

```
# Will you save a feedback data file?
# 0. Yes (output only individual F data).
# 1. Yes (output all the F data including those for grouped reflections
#    and estimated for unobserved reflections).
# 2. No.
```

In MPF, option 1 is usually selected to include $F(\boldsymbol{h}_j)$'s of high-Q reflections whose profiles are partly lacking.

```
# Will you save a e(GAUSS) distribution data file?
# 0. Yes (raw data file, *_eps.raw).
# 1. No.
```

When option 0 is selected, a text file storing histogram of

$$\frac{|F_{\mathrm{o}}(\boldsymbol{h}_j)| - |F(\boldsymbol{h}_j)|}{\sigma(\boldsymbol{h}_j)}$$

will be output. See also section 8.1.

```
# Fractions of lambda for generalized constraints.
# (8 parameters; l_2, l_4, l_6, l_8, l_10, l_12, l_14, l_16)
```

Eight parameters, λ_n ($n = 2 - 16$), for the fractions of generalized constraints with orders of n (see 3.4).

```
# Which prior\index{prior} densities will you use?
# 0. Uniform densities.
# 1. Densities recorded in the 3D densities file (*_prior.pgrid).
```

This option appears only on the use of the L-BFGS algorithm. When option 1 is selected, τ_k in Eq. (2.1) will be read in from 3D densities file. To use non-uniform prior densities in the ZSPA

algorithm, restart the calculation using a 3D data file (*.pgrid) storing prior densities.

```
# Will you save a preferences file?
# 0. No.
# 1. Yes (*.prf is output).
```

If *.prf is output, all the above parameters are stored in it and it can be reused as a template file for subsequent MEM analyses.

After answering all the questions, the conditions of MEM analysis are displayed in the screen. Before starting MEM iterations, ERIS asks the maximum number of iterations.

```
# Maximum number of cycles.
```

The iterations will continue until the convergence is achieved (CONSTR ≤ 1). When the convergence is reached, ERIS terminates after outputting CPU times.

6.2.2 Running ERIS using condition files

If the entire conditions of MEM calculations in `filename.prf` have already been input by modifying a template file with a text editor, ERIS can be executed in several different ways.

Let the name of *.prf be sample.prf. Then, ERIS can be run from the command line by typing

```
ERIS sample.prf
```

to read in sample.prf. When no environment variable PATH is set to a directory under which the program is placed, type the full path of the program, *e.g.*,

```
"C:\Program Files\RIETAN_VENUS\ERIS.exe" sample.prf
```

if ERIS.exe is stored in folder "C:\Program Files\RIETAN_VENUS\" on Windows. Of course, sample.prf has to be placed in the current folder in the above case. A simple batch file (Windows) or a shell script (OS X or Linux) where such a command is recorded would be more convenient.

The most practical and convenient way of carrying out MPF by ERIS is the use of MPF_multi.command because E in Eq. (5.1) can be automatically changed during a series of MPF analyses. For details in this bash script, read the manuals of MPF_multi.command, MPF_multi_Win.pdf (Windows) or MPF_multi_Mac.pdf (OS X), included in a distribution file, documents.zip.[1]

[1] http://fujioizumi.verse.jp/download/download_Eng.html

Chapter 7

INPUT FILES

7.1 Condition Files

- *.eris: An input file storing all the conditions of MEM calculations.
- *.prf: An input file of PRIMA storing conditions of MEM calculations. This file keeps backward compatibility with PRIMA while some new parameters are added for new features in Dysnomia and ERIS.
- *.alb: An input file of ALBA storing conditions of MEM calculations.

7.2 Intensity Files

- *.fcf, *.cif: Standard CIF file format.
- *.fos: Powder diffraction data generated by RIETAN-FP [6].
- *.mem: (a) Single-crystal data or (b) powder data where standard uncertainties of observed structure factors are given.
- *.baymem: .
- *.ffo: Powder diffraction data generated by RIETAN-FP [6].
- *.hkl: .

7.3 File format of *.eris

7.3.1 title

- Value: string of characters up to 80 characters long
- Default: none
- Description: Title of the calculation.

7.3.2 algorithm

- Value: 0/1
- Default: 1

- Description: Switch algorithm of MEM calculation.

  0. ZSPA algorithm.
  1. L-BFGS algorithm.

7.3.3 beam_source

- Value: 0/1
- Default: 0
- Description: From which experiment was the diffraction data obtained? This keyword only affects on the Wilson plot performed to calculate scale factor in the MEP analysis.

  0. X-ray diffraction data.
  1. Neutron diffraction data.

7.3.4 cell

- Value: a b c α β γ
- Default: none
- Description: Gives unit cell parameters for input data *.hkl, *.ffo, in which unit cell parameters are not recorded.

7.3.5 data

- Value: filename
- Default: compulsory keyword – no default
- Description: Gives the name of diffraction data file. It can be specified as a relative path from the place of *.eris file.

7.3.6 delta_d

- Value: positive real number
- Default: 0
- Description: Maximum difference in d/Angstrom in grouped reflections. Reflections with the d-spacing closer than delta_d will be considered as overlapped and treated as grouped reflections.

7.3.7 elements

- Value: positive integer, followed by a pair of element name and real number in each line
- Default: 0
- Description: Number of chemical species in the unit cell. It is followed by lines of equal numbers, each line describing a pair of element names and amounts in the unit cell. These data are used for the Wilson plot performed to calculate scale factor in the MEP analysis.

7.3.8 epsilon

- Value: positive real number
- Default: 0
- Description: A small value ε, used as a criterion for convergence of optimization. The convergence of the L-BFGS algorithm is judged by

$$\sqrt{\sum_{k=1}^{N_{\mathrm{V}}} \rho_k \left(\frac{\partial Q}{\partial \rho_k} \right)^2} < \varepsilon \left\langle \frac{\partial S}{\partial \rho_k} \right\rangle , \tag{7.1}$$

7.3.9 lambda

- Value: positive real number
- Default: automatically determined
- Description: The initial Lagrangian multiplier λ used in the ZSPA algorithm. This keyword has no effect on the use of the L-BFGS algorithm.

7.3.10 lambda_coef

- Value: positive real number
- Default: 0.05
- Description: A coefficient t, to adjust Lagrangian multiplier λ. This keyword is used only in the ZSPA algorithm and has no effect on the use of the L-BFGS algorithm.

7.3.11 max_cycles

- Value: positive integer
- Default: 10000
- Description: Maximum number of MEM cycles.

7.3.12 resolution

- Value: positive real number
- Default: 0.1
- Description: Resolution of real space voxels in angstrom.

7.3.13 restart

- Value: 0/1
- Default: 0
- Description: From which densities will you start the calculation?

  0. Start from flat (or prior) density.
  1. Start from the results of previous run stored in *.pgrid file.

7.3.14 scale_sigma

- Value: real value
- Default: 1
- Description: Scale input $\sigma(F)$ as $\sigma(F)' = \text{scale\_sigma} \times \sigma(F)$.

7.3.15 space_group

- Value: [space group number] [setting number]
- Default: 1 1
- Description: Gives the space group number and setting number.

7.3.16 scio

- Value: real value
- Default: 1
- Description: Scale input $\sigma(F)$ as $\sigma(F)' = \sigma(F)/\sqrt{\text{scio}}$.

7.3.17 use_prior

- Value: 0/1
- Default: 0
- Description: Which prior densities will you use?

  0. Flat density.
  1. Prior density recorded in *_prior.pgrid file.

7.3.18 weight_cn

- Value: eight real numbers
- Default: 1 0 0 0 0 0 0 0
- Description: Fractions of lambda for generalized constraints.

7.3.19 weight_m

- Value: real number
- Default: 0
- Description: Scale input $\sigma(F)$ as $\sigma(F)' = c \times \sigma(F)/\sqrt{m}$, where m is multiplicity of the reflection F and c is a constant value automatically determined so as not to change total scale of $\langle\sigma(F)\rangle$. See section 4.4 for more detail.

7.3.20 weight_d

- Value: [auto/exp/power] [real number]
- Default: auto
- Description: Weighting of observed reflections based on lattice-plane spacing d.

7.4 File format of *.prf

The content of *.prf storing various flags and specifications for MEM analysis is described in 6.2.1, and execution of ERIS and the subsequent input of *.prf in 6.2.2.

7.5 File format of *.alb

7.6 File format of *.mem

A line to give values (`POP1`, `POP2`, `IPTYP`, `JPH`, `JPK`, and `JPL`) related to preferred orientation is currently a dummy one. Never analyze powder diffraction data where preferred orientation was observed.

Grid numbers along a, b, and c axes should be selected in such a way that symmetry elements such as mirrors, rotation axes, and inversion centers coincide with intersections of grid lines. In addition, the grid numbers must be appropriately set by considering lattice parameters, a, b, and c. On the use of FFT, the best performance is achieved when division number along each axis is represented as

$$2^p \times 3^q \times 4^r \times 5^s \times 7^t \times 11^u \times 13^v,$$

where p, q, r, s, t, u, and v are integers equal to or larger than 0. Any other integers are also allowed; the calculation time is proportional to $N \log N$ even for prime numbers of N.

Beware that the sum, T^-, of negative b_c's in the unit cell should be set at 0.0 even if the sample contains no element with a negative b_c value, *e.g.*, H, Li, Ti, or Mn.

In a solid solution where two or more elements occupy the same site, a special procedure is required for calculating the sum, T^+, of positive b_c's and/or the sum, T^-, of negative b_c's in the unit cell [26]. Suppose a virtual chemical species with an average coherent-scattering length, $\bar{b}_c$, calculated from the occupancies of the constituent elements. Then, add $\bar{b}_c$ multiplied by the number of the virtual chemical species in the unit cell to T^+ if $\bar{b}_c > 0$ or T^- if $\bar{b}_c < 0$.

7.7 File format of *.fos

With Alchemy [24], text files with the following two formats can be obtained from output files, *.lst, of GSAS:

***.fos**

Regardless of `ID(neg.)`, both positive and negative values of total scattering amplitudes are always input. If `ID(neg.)` = 0, the second amplitude must be 0.0 (dummy).

***.mem**

If `ID(neg.)` = 0, only a positive total scattering amplitude is input, followed by λ in the same line. If `ID(neg.)` = 1, both positive and negative values of total scattering amplitudes are input, followed by λ as well.

Chapter 8

OUTPUT FILES

8.1 Output Files

***.out**

The standard output file of ERIS.

***.pgrid**

Three-dimensional voxel data storing electron or nuclear densities in a binary format. The content of this file can be visualized in three dimensions using VESTA [20] together with a crystal-structure model if necessary.

***.fba**

A list of structure factors calculated by Fourier transform from electron/nuclear densities resulting from MEM analysis. This file is used by RIETAN-FP [27] for whole-pattern fitting in MPF [3–5].

***.raw**

A text file storing histogram of the difference between the absolute $F_{\mathrm{o}}(\boldsymbol{h}_j)$ and $F(\boldsymbol{h}_j)$ values divided by the standard uncertainty, $\sigma(\boldsymbol{h}_j)$, of $|F_{\mathrm{o}}(\boldsymbol{h}_j)|$ to give a sample of the random variable with normalized Gaussian distribution, $\epsilon(\mathrm{Gauss})$ [19]:

$$\epsilon(\mathrm{Gauss}) = \frac{|F_{\mathrm{o}}(\boldsymbol{h}_j)| - |F(\boldsymbol{h}_j)|}{\sigma(\boldsymbol{h}_j)} \qquad (j = 1\text{–}N_F).$$

This file is useful for evaluating distribution of errors resulting from MEM analysis.

8.2 File format of *.pgrid

Volumetric data are composed of regular grids in 3D space. A volume element, "voxel," at each grid point represents a value on the grid point. Voxels are analogous to pixels, which represent 2D image data.

8.2.1 Periodic grid vs. general grid

In general, two types of formats are used to record volumetric data in files: general and periodic grids. **Figure 8.1** schematically illustrates the general concepts of the two kinds of the formats.

The general grid is a uniform one spanned inside a bounding box for molecules and a unit cell for crystals. For crystal structures, part of data in the general grid are redundant owing to the periodicity of the data. For example, a numerical value at (1, 1, 1) are equal to that at the origin, *i.e*, (0, 0, 0). Grids where these redundant points have been omitted are called periodic ones.

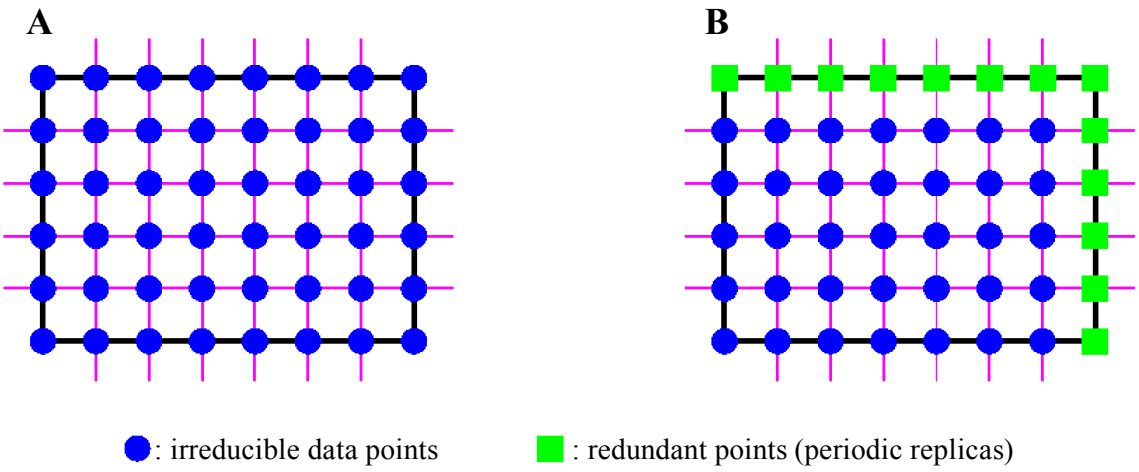

Figure 8.1: Two types of the grids for volumetric data on a plane. (A) general grid and (B) periodic grid.

8.2.2 Pseudocode to write *.pgrid

This is a binary file storing a series of physical values in each voxel followed by some header data. Part of the C++ source code to write the header part is as follows.

```
char title[80]; // 80 characters of title that ends with '\0'.
int version[4] = {3, 0, 0, 0}; //The current version number is 3 0 0 0
int gType;   // 0: general grid (*.ggrid); 1: periodic grid (*.pgrid).
int fType;   // 0: raw data; 1: paired record of index and value of voxel.
int nVal;    // 1 or 2; Number of data values for each voxel.
int dim;     // Only dimension=3 is currently supported.
int nVox[3]; // Number of voxels along a, b, and c axis.

// Total number of voxels to be recorded in the file.
// nAsym = nVox[0]*nVox[1]*nVox[2] in the case of fType = 1.
int nAsym;

float cell[6]; // Unit cell parameters: a, b, c, alpha, beta, gamma

// Allocate voxel data
float *rho = new float[nVox[0] * nVox[1] * nVox[2] * nVal];

// Set values to the above parameters before writing them.

// Write header
fwrite(version, sizeof(int),   4, fptr);
fwrite(title,   sizeof(title), 1, fptr);
fwrite(&gType,  sizeof(int),   1, fptr);
```

```
fwrite(&fType,  sizeof(int),   1, fptr);
fwrite(&nVal,   sizeof(int),   1, fptr);
fwrite(&dim,    sizeof(int),   1, fptr);
fwrite(nVox,    sizeof(int),   3, fptr);
fwrite(&nAsym,  sizeof(int),   1, fptr);
fwrite(cell,    sizeof(float), 6, fptr);
```

For the raw data type (when fType = 0), binary data of each voxel is output in the following order.

```
if(fType = 0){
    for(int k=0; k<nVox[2]; k++){
        for(int j=0; j<nVox[1]; j++){
            for(int i=0; i<nVox[0]; i++){
                m = k*nVox[0]*nVox[1] + j*nVox[0] + i;
                for(int n=0; n<nVal; n++){
                    fwrite(&rho[nVal*m + n], sizeof(float), 1, fptr);
                }
            }
        }
    }
}
```

Note that values of `nVox` are not the same in the case of `fType = 0` (*.ggrid) and `fType = 1` (*.pgrid). When saving periodic data in the former format, *i.e.*, as the general grid format, each value of `nVox` is 1 voxel larger than that of the latter format. The difference is due to the redundant data points at $x = 1$, $y = 1$, and $z = 1$, which are recorded in *.ggrid but not in *.pgrid. For non-periodic data, always use *.ggrid instead of *.pgrid.

When the data is periodic and space group symmetry is higher than -1, the file size can be reduced by only recording data of symmetrically independent voxels. In this case, set the parameter `fType = 1` and write a list of symmetry operations before voxel data.

```
// Additional parameters used when fType = 1.
int nPos; // Number of symmetry operations to be written.

// nCen: A flag if the recorded symmetry operations need to be expanded by
// inversion operation.
// 0 for non-centrosymmetric space groups or when all symmetry operations
// are recorded.
// 1 for centrosymmetric and only half of symmetry operations are recorded.
// VESTA and ERIS always record full set of symmetry operations and set
// nCen=0 to allow arbitrary origin shift.
int nCen;

// nSub: Number of lattice points.
// nSub>1 in the case of complex lattices such as I, F, A, B, C etc.
// VESTA and ERIS always record full set of symmetry operations and set
// nSub=1 to allow arbitrary origin shift.
int nSub;

// A list of lattice points in integer.
// For the P-lattice, or when all symmetry operations are recorded,
// subPos[1][3] = {0, 0, 0};
```

```
// subPos[2][3] = {{0, 0, 0},{nVox[0]/2, nVox[1]/2, 0}};         C-lattice
// subPos[2][3] = {{0, 0, 0},{nVox[0]/2, nVox[1]/2, nVox[2]/2}}; I-lattice
// etc.
int subPos[NSUB][3];

// A 4x4 matrix class storing symmetry operation in column-major format.
// The first 3x3 part represent rotation.
// The 4th column represent translation.
// An operator (i,j) in the following code is defined to return a value of
// i-th raw and j-th column in double.
Matrix *symOP;

// A list of indices of voxel positions in asymmetric unit.
int *ijk;

// Set values to the above parameters before writing them.

if(fType = 1){
    fwrite(&nPos,   sizeof(int), 1, fptr);
    fwrite(&nCen,   sizeof(int), 1, fptr);
    fwrite(&nSub,   sizeof(int), 1, fptr);

    // Write symmetry operations
    // A position X is transformed to X' by X' = symOP[i] * X.
    for(i=0; i<nPos; i++){
        for(j=0; j<3; j++){
            for(k=0; k<3; k++){
                int a = (int)(symOP[i](k,j));
                fwrite(&a, sizeof(int), 1, fptr);
            }
        }
        for(j=0; j<3; j++){
            int a = (int)(symOP[i](j,3) * nVox[j]);
            fwrite(&a, sizeof(int), 1, fptr);
        }
    }
    for(i=0; i<nSub; i++){
        fwrite(subPos[i], sizeof(int), 3, fptr);
    }

    // Write voxel data in asymmetric unit.
    // ijk is 3 indices of position of i_th voxel.
    for(i=0; i<nAsym; i++){
        m = ijk[i*3+2]*nVox[0]*nVox[1] + ijk[i*3+1]*nVox[0] + ijk[i*3];
        fwrite(&m,      sizeof(int),   1, fptr);
        for(int n=0; n<nVal; n++){
            fwrite(&rho[nVal*m + n], sizeof(float), 1, fptr);
        }
    }
}
```

Bibliography

[1] D. M. Collins, *Nature (London, U. K.)*, **298**, 49 (1982).

[2] M. Sakata and M. Sato, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **46**, 263 (1990).

[3] F. Izumi and R. A. Dilanian, "Recent Research Developments in Physics," Vol. 3, Part II, Transworld Research Network, Trivandrum (2002), pp. 699–726.

[4] F. Izumi, S. Kumazawa, T. Ikeda, W.-Z. Hu, A. Yamamoto, and K. Oikawa, *Mater. Sci. Forum*, **378–381**, 59 (2001).

[5] F. Izumi, *Solid State Ionics*, **172**, 1 (2004).

[6] F. Izumi and K. Momma, *IOP Conf. Ser.: Mater. Sci. Eng.*, **18**, 022001 (2011).

[7] K. Momma and F. Izumi, *Z. Kristallogr.*, Proc. 1, 195 (2011).

[8] K. Momma, T. Ikeda, A. A. Belik, and F. Izumi, *Powder Diffr.*, **28**, 184 (2013).

[9] R. A. Dilanian and F. Izumi, "Super-fast Program, PRIMA, for the Maximum-Entropy Method," National Institute for Materials Science, Tsukuba (2010).

[10] S. F. Gull and J. Skilling, "Quantified Maximum Entropy, MemSys5 Users' Manual," Maximum Entropy Data Consultants Ltd., Suffolk (1999).

[11] J. Nocedal, *Math. Comp.*, **35**, 773 (1980).

[12] S. F. Gull and G. J. Daniel, *Nature (London, U. K.)*, **272**, 686 (1978).

[13] G. Bricogne, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **44**, 517 (1988).

[14] E. T. Jaynes, *Phys. Rev.*, **106**, 620 (1957).

[15] S. Kumazawa, M. Takata, and M. Sakata, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **51**, 47 (1995).

[16] S. van Smaalen, L. Palatinus, and M. Scheider, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **59**, 459 (2003).

[17] S. Kumazawa, Y. Kubota, M. Takata, M. Sakata, and Y. Ishibashi, *J. Appl. Crystallogr.*, **26**, 453 (1993).

[18] W. Jauch, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **50**, 650 (1994).

[19] L. Palatinus and S. van Smaalen, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **58**, 559 (2002).

[20] K. Momma and F. Izumi, *J. Appl. Crystallogr.*, **44**, 1272 (2011).

[21] R. Y. de Vries, W. J. Briels, and D. Feil, *Acta Crystallogr., Sect. A: Found. Crystallogr.*, **50**, 383 (1994).

[22] A. Hofmann, J. Netzel, and S. van Smaalen, *Acta Crystallogr., Sect. B: Struct. Sci.*, **63**, 285 (2007).

[23] F. Izumi, "Multi-Purpose Pattern-Fitting System RIETAN-FP," National Institute for Materials Science, Tsukuba (2012).

[24] F. Izumi and Y. Kawamura, *Bunseki Kagaku*, **55**, 391 (2006).

[25] Y. Kawamura, K. Momma, and F. Izumi, *Hamon*, **23**, 72 (2013).

[26] R. Ali, M. Yashima, and F. Izumi, *Chem. Mater.*, **19**, 3260 (2007).

[27] F. Izumi and K. Momma, *Solid State Phenom.*, **130**, 15 (2007).

Index