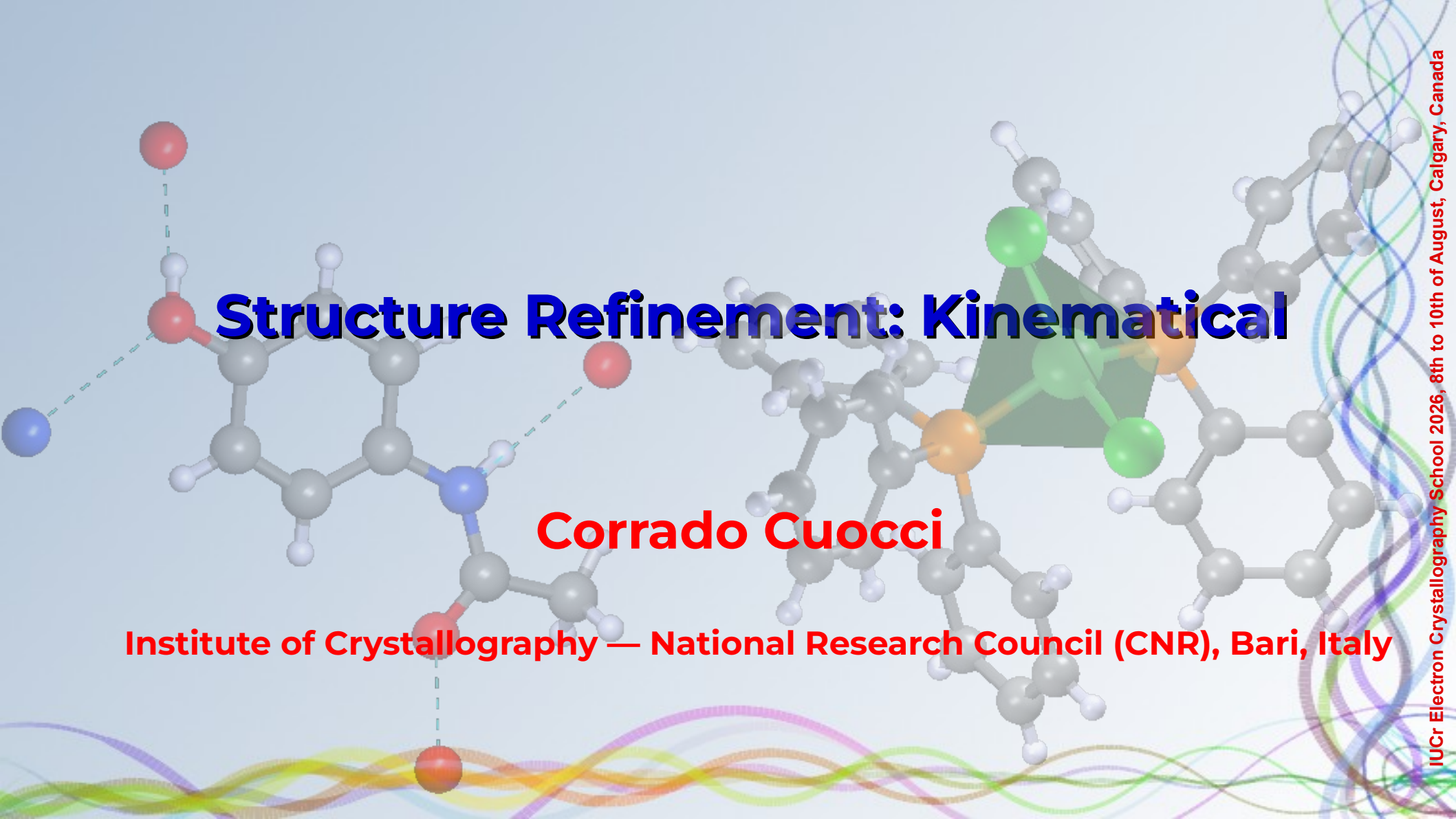


The background features a complex molecular structure with grey carbon atoms, white hydrogen atoms, red oxygen atoms, and blue nitrogen atoms. Some atoms are connected by dashed lines, indicating hydrogen bonds. To the right, there is a green polyhedral structure with orange and green vertices. At the bottom right, a series of colorful, wavy lines in blue, green, yellow, and purple represent a diffraction pattern or electron density map.

Structure Refinement: Kinematical

Corrado Cuocci

Institute of Crystallography — National Research Council (CNR), Bari, Italy

Reasons for Performing Refinement

- To improve phasing
- To try to verify that the structure is 'correct'
- To obtain the 'best' values for the parameters in the model

Trial model

- ◆ Incorrect atom type assignments for some coordinates
- ◆ Atomic coordinates not very accurate
- ◆ Missing structural details: groups of lighter atoms not yet located
- ◆ Disorder not yet identified or modeled
- ◆ Hydrogen positions not yet determined



The model that
best represents
the data

Principles of Least Squares

Set on N observations: $y_1, y_2, \dots, y_n$

weight assigned to the observations: $w_1, w_2, \dots, w_N$

$$S = \sum_{i=1}^n w_i [y_{i,obs} - y_{i,calc}]^2$$

Unknown p parameters: $x_1, x_2, \dots, x_p$

$$y_{1,calc} = M_1(x_1, x_2, \dots, x_p)$$

$$y_{2,calc} = M_2(x_1, x_2, \dots, x_p)$$

.....

$$y_{N,calc} = M_N(x_1, x_2, \dots, x_p)$$

Principles of Least Squares

$$S = \sum_{i=1}^n w_i [y_{i,obs} - y_{i,calc}]^2$$

$$y_{i,calc} = M_i(\mathbf{x}) \quad i = 1, 2, \dots, n$$

$$S = \sum_{i=1}^n w_i [y_{i,obs} - M_i(\mathbf{x})]^2$$

In the matrix form:

$$S = [\mathbf{y} - \mathbf{M}(\mathbf{x})]^T \mathbf{W} [\mathbf{y} - \mathbf{M}(\mathbf{x})]$$

Principles of Least Squares

$$S = [\mathbf{y} - \mathbf{M}(\mathbf{x})]^T \mathbf{W} [\mathbf{y} - \mathbf{M}(\mathbf{x})]$$

W is a diagonal square matrix representing individual weights

$$\mathbf{W} = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{pmatrix}$$

Principles of Least Squares

$$S = \sum_{i=1}^n w_i [y_{i,obs} - M_i(\mathbf{x})]^2$$

Condition for S to be a minimum:

$$\frac{\partial S}{\partial x_j} = -2 \sum_{i=1}^n w_i [y_{i,obs} - M_i(\mathbf{x})] \frac{\partial M_i(\mathbf{x})}{\partial x_j} = 0 \quad j = 1, \dots, p$$

Normal equations

The model functions, $M_i(\mathbf{x})$, are, in general, nonlinear

Linear Least Squares

$$y_{1,calc} = M_1(\mathbf{x}) = b_1 + a_{11}x_1 + a_{12}x_2 + \dots + a_{1p}x_p$$

$$y_{2,calc} = M_2(\mathbf{x}) = b_2 + a_{21}x_1 + a_{22}x_2 + \dots + a_{2p}x_p$$

.....

$$y_{n,calc} = M_n(\mathbf{x}) = b_n + a_{n1}x_1 + a_{n2}x_2 + \dots + a_{np}x_p$$

$$y_{i,calc} = M_i(\mathbf{x}) = b_i + \sum_{j=1}^p a_{ij}x_j \quad i = 1, \dots, n$$

In matrix form: $\mathbf{M}(\mathbf{x}) = \mathbf{b} + \mathbf{A}\mathbf{x}$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N1} & \cdots & a_{np} \end{pmatrix}$$

Linear Least Squares

$$S = [\mathbf{y} - \mathbf{M}(\mathbf{x})]^T \mathbf{W} [\mathbf{y} - \mathbf{M}(\mathbf{x})]$$

$$\begin{array}{c} \uparrow \qquad \qquad \uparrow \\ \mathbf{M}(\mathbf{x}) = \mathbf{b} + \mathbf{A}\mathbf{x} \end{array}$$

$$S = [(\mathbf{y} - \mathbf{b}) - \mathbf{A}\mathbf{x}]^T \mathbf{W} [(\mathbf{y} - \mathbf{b}) - \mathbf{A}\mathbf{x}]$$

Linear Least Squares

$$\frac{\partial S}{\partial x_j} = -2 \sum_{i=1}^n w_i [y_{i,obs} - M_i(\mathbf{x})] \frac{\partial M_i(\mathbf{x})}{\partial x_j} = 0$$

$\frac{\partial M_i(\mathbf{x})}{\partial x_j} = a_{ij}$



$$M_i(\mathbf{x}) = b_i + \sum_{j=1}^p a_{ij} x_j$$

$$\sum_{i=1}^n w_i \left[y_{i,obs} - b_i - \sum_{k=1}^p a_{ik} x_k \right] a_{ij} = 0$$

Linear Least Squares

$$\sum_{i=1}^n w_i \left[y_{i,obs} - b_i - \sum_{k=1}^p a_{ik} x_k \right] a_{ij} = 0 \quad j = 1, \dots, p$$

$$\sum_{i=1}^n a_{ij} w_i \sum_{k=1}^p a_{ik} x_k = \sum_{i=1}^n a_{ij} w_i (y_{i,obs} - b_i) \quad j = 1, \dots, p$$

Normal equations of the weighted linear least-squares problem

$$\sum_{k=1}^p \left(\sum_{i=1}^n a_{ij} w_i a_{ik} \right) x_k = \sum_{i=1}^n a_{ij} w_i (y_{i,obs} - b_i)$$

Linear Least Squares

$$\sum_{k=1}^p \left(\sum_{i=1}^n a_{ij} w_i a_{ik} \right) x_k = \sum_{i=1}^n a_{ij} w_i (y_{i,obs} - b_i) \quad j = 1, \dots, p$$

$\downarrow$ $\downarrow$

$$(\mathbf{A}^T \mathbf{W} \mathbf{A})_{jk} \quad [\mathbf{A}^T \mathbf{W} (\mathbf{y} - \mathbf{b})]_j$$

**Normal equations
(matrix notation)**

$$\mathbf{A}^T \mathbf{W} \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{W} (\mathbf{y} - \mathbf{b})$$

Linear Least Squares

$$\mathbf{A}^T \mathbf{W} \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{W} (\mathbf{y} - \mathbf{b})$$

Solution is:

$$\mathbf{x} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} (\mathbf{y} - \mathbf{b})$$

The explicit computation of the inverse
 $(\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1}$ is avoided in practice

Linear Least Squares

$$\underbrace{A^T W A}_{\mathbf{N}} \mathbf{x} = \underbrace{A^T W (\mathbf{y} - \mathbf{b})}_{\mathbf{q}}$$

$$\mathbf{N} \mathbf{x} = \mathbf{q}$$

N is a symmetric
positive-definite matrix

Solved using:

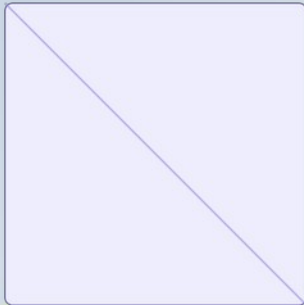
- **Cholesky decomposition**
- **QR decomposition**

Cholesky factorisation

$$\mathbf{N} = \mathbf{L}\mathbf{L}^T$$

$$\mathbf{N} = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{m1} & l_{m2} & l_{m3} & \cdots & l_{pp} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} & \cdots & l_{p1} \\ 0 & l_{22} & l_{32} & \cdots & l_{p2} \\ 0 & 0 & l_{33} & \cdots & l_{p3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & l_{pp} \end{pmatrix}$$

$\mathbf{N}$
symmetric, pos. def.

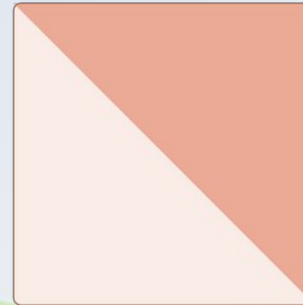


$\mathbf{L}$
lower triangular



=

$\mathbf{L}^T$
upper triangular



.

Cholesky factorisation

$$\mathbf{N} = \mathbf{L}\mathbf{L}^T$$

The system $\mathbf{N}\mathbf{x} = \mathbf{q}$ is solved in two triangular steps:

$$\mathbf{L}\mathbf{L}^T\mathbf{x} = \mathbf{q} \xrightarrow{\text{forward substitution}} \mathbf{L}\mathbf{z} = \mathbf{q} \xrightarrow{\text{forward substitution}} \mathbf{z}$$

$\downarrow$
 $\mathbf{z}$

$$\mathbf{L}^T\mathbf{x} = \mathbf{z} \xrightarrow{\text{back substitution}} \mathbf{x}$$

Limitations: it requires forming $\mathbf{N} = \mathbf{A}^T\mathbf{W}\mathbf{A}$ explicitly, which squares the condition number. This can cause loss of accuracy.

QR factorization

Instead of forming the normal equations ($\mathbf{N}\mathbf{x}=\mathbf{q}$) explicitly, this approach works directly on the weighted data matrix

$$\mathbf{A}^T \mathbf{W} \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{W} (\mathbf{y} - \mathbf{b})$$

$$\tilde{\mathbf{A}} = \mathbf{W}^{\frac{1}{2}} \mathbf{A}$$

$$\tilde{\mathbf{r}} = \mathbf{W}^{\frac{1}{2}} (\mathbf{y} - \mathbf{b})$$



weighted least-squares
problem is equivalent to
an unweighted one on $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \mathbf{x} = \tilde{\mathbf{A}} \tilde{\mathbf{r}}$$

$$\tilde{\mathbf{A}} = \mathbf{Q} \mathbf{R}$$

QR factorization

$$\tilde{\mathbf{A}} = \mathbf{Q}\mathbf{R}$$

$$\tilde{\mathbf{A}} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1p} \\ q_{21} & q_{22} & \cdots & q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{Np} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1p} \\ 0 & r_{22} & r_{23} & \cdots & r_{2p} \\ 0 & 0 & r_{33} & \cdots & r_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{pp} \end{pmatrix}$$

$\tilde{\mathbf{A}}$

design
matrix

$N \times p$

=

$\mathbf{Q}$

orthonormal
columns

$N \times p$

.

$\mathbf{R}$

upper
triangular

$p \times p$

QR Factorisation for Least-Squares Problems

$$\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \mathbf{x} = \tilde{\mathbf{A}} \tilde{\mathbf{r}}$$



$$(\mathbf{QR})^T (\mathbf{QR}) \mathbf{x} = (\mathbf{QR})^T \tilde{\mathbf{r}}$$

$$\tilde{\mathbf{A}} = \mathbf{QR}$$

$$(\mathbf{QR})^T = \mathbf{R}^T \mathbf{Q}^T$$



$$\mathbf{R}^T \mathbf{Q}^T \mathbf{Q} \mathbf{R} \mathbf{x} = \mathbf{R}^T \mathbf{Q}^T \tilde{\mathbf{r}}$$

$$\mathbf{I}_m$$



$$\mathbf{R}^T \mathbf{R} \mathbf{x} = \mathbf{R}^T \mathbf{Q}^T \tilde{\mathbf{r}}$$



$$(\mathbf{R}^T)^{-1} \mathbf{R}^T \mathbf{R} \mathbf{x} = (\mathbf{R}^T)^{-1} \mathbf{R}^T \mathbf{Q}^T \tilde{\mathbf{r}}$$

$$\mathbf{I}_m$$

$$\mathbf{I}_m$$

$$\mathbf{R} \mathbf{x} = \mathbf{Q}^T \tilde{\mathbf{r}} \quad \text{back substitution} \quad \Rightarrow \quad \mathbf{x}$$

- QR avoids the explicit formation of the normal matrix $\mathbf{N}$
- Numerically more stable when parameters are highly correlated or the matrix is ill-conditioned

Nonlinear Least Squares

Expanding the model function $\mathbf{M}(\mathbf{x})$ around the starting point $\mathbf{x}_0$ in Taylor's series and retaining only the linear terms we obtain the equation:

$$M_i(\mathbf{x}) = M_i(\mathbf{x}_0) + \sum_{j=1}^p J_{ij}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)_j$$

$$\mathbf{M}(\mathbf{x}) = \mathbf{M}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\Delta\mathbf{x}$$

Matrix form

$$\Delta\mathbf{x} = (\mathbf{x} - \mathbf{x}_0)$$

Vector of parameter shifts

$$\mathbf{J}(\mathbf{x}_0) \in \mathbb{R}^{N \times p} \quad [\mathbf{J}(\mathbf{x}_0)]_{ij} = \left. \frac{\partial M_i}{\partial x_j} \right|_{\mathbf{x}_0}$$

Jacobian matrix

The Gauss Newton Algorithm

Gauss Newton algorithm for each k iteration

(1) Compute the search direction $\Delta \mathbf{x}$ as the solution of the linear system

$$\mathbf{J}^T \mathbf{W} \mathbf{J} \Delta \mathbf{x} = \mathbf{J}^T \mathbf{W} [\mathbf{y} - \mathbf{M}(\mathbf{x})]$$

(2) Set $\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta \mathbf{x}$

(3) If not converged go to **(1)**, else stop.

Gauss-Newton-Type Methods

- Gauss-Newton method with a **line search**

Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \Delta \mathbf{x}$

where α called step length, is such that the algorithm is in descendant condition:

$$S(\mathbf{x}_k + \alpha \Delta \mathbf{x}) < S(\mathbf{x}_k)$$

is chosen by a line-search procedure

- **Levenberg-Marquardt** algorithm modify the normal equations in

$$(\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \mathbf{I}) \Delta \mathbf{x} = \mathbf{J}^T \mathbf{W} [\mathbf{y} - \mathbf{M}(\mathbf{x})]$$

where $\mathbf{I}$ is the identity matrix, λ is a damping factor

Standard deviations

Variance-covariance matrix: $\mathbf{V}_x = (\mathbf{J}^T \mathbf{W} \mathbf{J})^{-1}$

$$\sigma(x_j) = \sqrt{\frac{(V_x)_{jj} S}{n - p}}$$

n is the number of observations

p is the number of unknown parameters

$(V_x)_{jj}$ is the corresponding diagonal element of the variance-covariance matrix

Inversion via Cholesky $\mathbf{N}^{-1} = (\mathbf{L}^T \mathbf{L})^{-1} = \mathbf{L}^{-1} (\mathbf{L}^T)^{-1} = \mathbf{L}^{-1} (\mathbf{L}^{-1})^T$

Inversion via QR $\mathbf{N}^{-1} = (\mathbf{R}^T \mathbf{R})^{-1} = \mathbf{R}^{-1} (\mathbf{R}^T)^{-1} = \mathbf{R}^{-1} (\mathbf{R}^{-1})^T$

Newton-like Methods

Quadratic approximation $q(\Delta\mathbf{x})$ to S at the current point $\mathbf{x}_k$

$$S(\mathbf{x}_k + \Delta\mathbf{x}) \approx S(\mathbf{x}_k) + \nabla S_k^T \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \mathbf{H}_k \Delta\mathbf{x} = q(\Delta\mathbf{x})$$

∇S_k is the gradient at $\mathbf{x}_k$, a column vector of dimension p

H_k is the Hessian matrix at $\mathbf{x}_k$, $p \times p$ symmetric $H_{ij} = \frac{\partial^2 S}{\partial x_i \partial x_j}$

$$\frac{\partial q}{\partial(\Delta\mathbf{x})} = \nabla S_k + \mathbf{H}_k \Delta\mathbf{x} = \mathbf{0} \quad \Rightarrow \quad \mathbf{H}_k \Delta\mathbf{x} = -\nabla S_k$$

Advantages: very fast convergence near the solution

Quasi-Newton Methods

$$\mathbf{H}_k \Delta \mathbf{x} = -\nabla S_k$$

$$\mathbf{H}_k = 2\mathbf{J}_k^T \mathbf{W} \mathbf{J}_k - 2\mathbf{B}_k \quad \mathbf{B}_k \approx 0 \quad \Rightarrow \quad \text{Gauss-Newton approximation}$$

$$(\mathbf{B}_k)_{ij} = \sum_i^N w_h (y_i - M_i(x_k)) \frac{\partial^2 M_i(x_k)}{\partial x_i \partial x_j}$$

Since computing $\mathbf{H}_k$ exactly is prohibitively expensive, quasi-Newton methods iteratively approximate it using only gradient information. **BFGS** and **L-BFGS** directly update an approximation of $\mathbf{H}_k$, avoiding the computation of $\mathbf{J}_k$ and requiring only gradient evaluations.

Programs using **L-BFGS**: **Phenix**, **CNS**, **Rosetta**, GROMACS/AMBER/CHARMM, SciPy

Conjugate-gradients methods

At each iteration

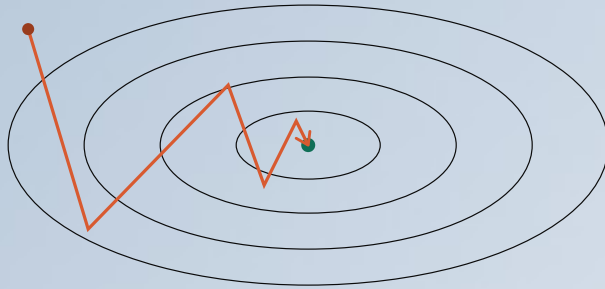
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

Search direction

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$$

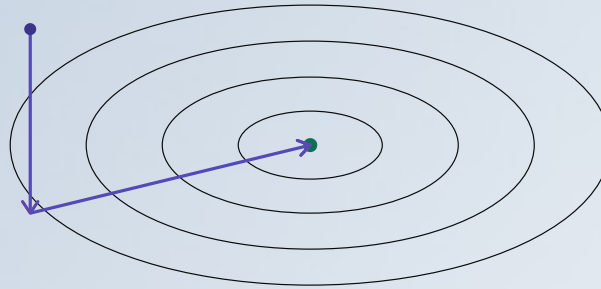
Step length by line search

Steepest descent



Each step undoes part of the last — many zig-zags

Conjugate gradient



Two A-conjugate steps reach the exact minimum

Several formulas exist. The most common are **Fletcher-Reeves** and **Polak-Ribière**. It is chosen to make $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ conjugate.

$$\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0 \quad (i \neq j)$$

Advantages: no Hessian is required, low memory consumption, particularly suitable for very large optimization problems, each iteration is computationally inexpensive.

Disadvantages: rarely competitive with Gauss-Newton / Levenberg-Marquardt.

The Conjugate Gradient Method for Solving Linear Systems

$$\left(\mathbf{J}^\top \mathbf{W} \mathbf{J}\right) \Delta \mathbf{x} = -\mathbf{J}^\top \mathbf{W} \mathbf{r} \quad \text{Gauss-Newton} \quad \Rightarrow \quad \mathbf{A} \mathbf{x} = \mathbf{b}$$

Solving $\mathbf{A} \mathbf{x} = \mathbf{b}$

Is equivalent to minimizing the quadratic function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

This approach combines improved convergence properties with the low memory requirements and computational efficiency of iterative solvers, making it particularly attractive for large-scale optimization problems.

The algorithm

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0, \quad \mathbf{p}_0 = \mathbf{r}_0$$

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A} \mathbf{p}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$$

$$\beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

Maximum Likelihood

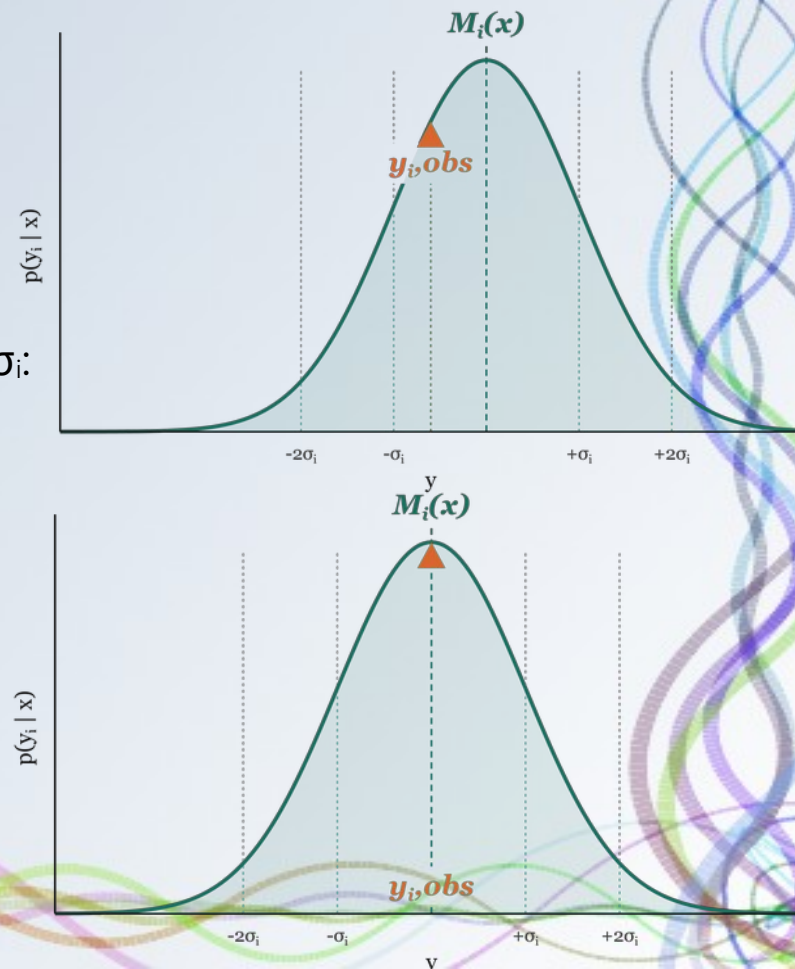
For each observation y_i , we can write the probability of observing it given the model $\mathbf{x}$ as a conditional distribution

$$p(y_i \mid \mathbf{x})$$

If the errors are Gaussian with standard deviation σ_i :

$$p(y_i \mid \mathbf{x}) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp \left(-\frac{(y_i - M_i(\mathbf{x}))^2}{2\sigma_i^2} \right)$$

Common distributions: Gaussian, Poisson, Rice, Wilson, Cauchy (Lorentzian), Laplace.



Maximum Likelihood

Assuming the observations are mutually independent, the joint probability of all observations is the product:

$$L(\mathbf{x}) = \prod_i p(y_i \mid \mathbf{x}) \quad \text{the likelihood function}$$

The ML estimate consists in finding the $\mathbf{x}$ that maximises this sum.

$$\ln L(\mathbf{x}) = \sum_i \ln p(y_i \mid \mathbf{x})$$

The quantity that is actually minimised:

$$-\ln L(\mathbf{x}) = -\sum_i \ln p(y_i \mid \mathbf{x})$$

**ML refinement in programs
such as REFMAC5 and Phenix.refine**

Maximum Likelihood

$$-\ln L(\mathbf{x}) = -\sum_i \ln p(y_i \mid \mathbf{x})$$

$$p(y_i \mid \mathbf{x}) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(y_i - M_i(\mathbf{x}))^2}{2\sigma_i^2}\right)$$

Gaussian error distribution



$$S = \sum_{i=1}^n \frac{(y_{i,obs} - y_{i,calc})^2}{\sigma_i^2}$$

ML with Gaussian errors coincides with least squares.

Functions to minimize

$$S = \sum_{i=1}^n w_i [y_{i,obs} - y_{i,calc}]^2$$



$$S = \sum_{\mathbf{h}} w_{\mathbf{h}} (F_{\mathbf{h}}^o{}^2 - F_{\mathbf{h}}^c{}^2)^2$$

This is currently the most widely used method
and the only one implemented in SHELX

$$S' = \sum_{\mathbf{h}} w'_{\mathbf{h}} (F_{\mathbf{h}}^o - F_{\mathbf{h}}^c)^2$$

$$I_{hkl} = C |F_{hkl}|^2$$

Weights

$$S = \sum_{i=1}^n w_i [y_{i,obs} - y_{i,calc}]^2 \quad w_i = \frac{1}{\sigma^2(y_{i,obs})}$$



$$S = \sum_{\mathbf{h}} w_{\mathbf{h}} (F_{\mathbf{h}}^{o2} - F_{\mathbf{h}}^{c2})^2$$

$$S' = \sum_{\mathbf{h}} w'_{\mathbf{h}} (F_{\mathbf{h}}^o - F_{\mathbf{h}}^c)^2$$

$$w_{\mathbf{h}} = \frac{1}{\sigma^2(F_{\mathbf{h}}^{o2})} \leftarrow \sigma(I_h)$$

$$w'_{\mathbf{h}} = \frac{1}{\sigma^2(F_{\mathbf{h}}^o)} \leftarrow \sigma^2(F_{\mathbf{h}}^o) = \frac{\sigma^2(F_{\mathbf{h}}^{o2})}{4F_{\mathbf{h}}^{o2}}$$

Model and Parameters

$$F_{\mathbf{h}}^c = K \sum_{j=1}^n O_j t_j(s) f_j(s) \exp [2\pi i(hx_j + ky_j + lz_j)]$$

s is $\sin \theta_{\mathbf{h}} / \lambda$

n is the total number of atoms in the unit cell

f_j is the atomic scattering factor  determined by the atom type

x_j, y_j, z_j are the fractional coordinates of the j th atom in the unit cell

t_j is the temperature factor (atomic displacement parameters)

O_j is the occupation factor

K is a scale factor

Flack parameter

Each atom is described by 9–10 parameters.
About 8–10 reflections per parameter

Isotropic Displacement Parameter

$$F_{\mathbf{h}}^c = K \sum_{j=1}^n O_j t_j(s) f_j(s) \exp [2\pi i(hx_j + ky_j + lz_j)]$$

$$t_j(s) = \exp\left(-B_j \frac{\sin^2 \theta}{\lambda^2}\right) = \exp(-B_j s^2)$$

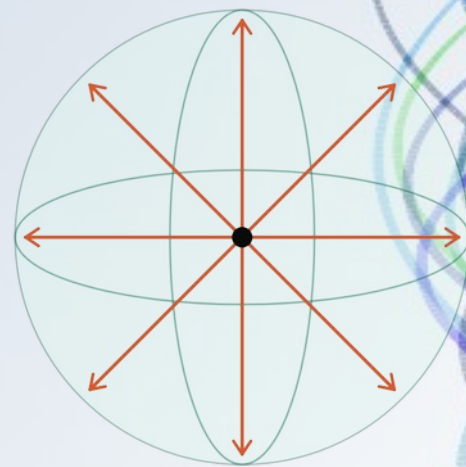
B_j is the displacement parameter of the j th atom

$$B_j = 8\pi \langle \bar{u}^2 \rangle_j$$

$\langle \bar{u}^2 \rangle_j$ is the root mean square deviation in Å

$$U_j = B_j / 8\pi$$

One independent parameter per atom



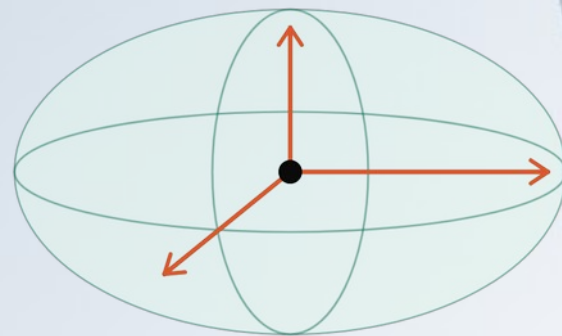
Anisotropic Displacement Parameters

$$F_{\mathbf{h}}^c = K \sum_{j=1}^n O_j \boxed{t_j(s)} f_j(s) \exp [2\pi i(hx_j + ky_j + lz_j)]$$

$$t_j = \exp \left[- \left(B_{11}^j h^2 a^{*2} + B_{22}^j k^2 b^{*2} + B_{33}^j l^2 c^{*2} \right. \right. \\ \left. \left. + 2B_{12}^j h k a^* b^* + 2B_{13}^j h l a^* c^* + 2B_{23}^j k l b^* c^* \right) \right]$$

$$t_j = \exp \left[- 2\pi^2 \left(U_{11}^j h^2 a^{*2} + U_{22}^j k^2 b^{*2} + U_{33}^j l^2 c^{*2} \right. \right. \\ \left. \left. + 2U_{12}^j h k a^* b^* + 2U_{13}^j h l a^* c^* + 2U_{23}^j k l b^* c^* \right) \right]$$

$$t_j = \exp \left[- \mathbf{h}^{*\top} \mathbf{B}^j \mathbf{h}^* \right], \quad \mathbf{h}^* = \begin{pmatrix} h a^* \\ k b^* \\ l c^* \end{pmatrix}$$



$$\mathbf{B}^j = \begin{pmatrix} B_{11}^j & B_{12}^j & B_{13}^j \\ B_{21}^j & B_{22}^j & B_{23}^j \\ B_{31}^j & B_{32}^j & B_{33}^j \end{pmatrix}$$

Six independent parameters per atom

Residual Factors

- The weighted R-factor based on F^2 (wR , wR_2)

$$wR_f = \left[\frac{\sum_{\mathbf{h}} w_{\mathbf{h}} (F_{\mathbf{h}}^o{}^2 - F_{\mathbf{h}}^c{}^2)^2}{\sum_{\mathbf{h}} w_{\mathbf{h}} F_{\mathbf{h}}^o{}^2} \right]^{1/2}$$

- The unweighted R-factor based on F (R , R_1)

$$R_f = \frac{\sum_{\mathbf{h}} |F_{\mathbf{h}}^o - F_{\mathbf{h}}^c|}{\sum_{\mathbf{h}} F_{\mathbf{h}}^o}$$

Residual Factors

- Goodness of fit (GooF, GoF, S).

$$GoF = \left[\frac{\sum_{\mathbf{h}} w_{\mathbf{h}} (F_{\mathbf{h}}^o - F_{\mathbf{h}}^c)^2}{n - p} \right]^{1/2}$$

Constraints

Constraints are mathematical relationships between parameters

Symmetry constraints are mandatory and automatically imposed by the program

- *Special position*

e.g., atom on special position $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ should not be refined,

atom on special position (x, x, x) in space group $P23$ should have equal shift on x, y, z

Constraints

Constraints imposed by the user to reduce the number of parameters

- Riding model (move H atoms synchronously with the C atoms)
- Occupation factor
e.g., A,B atoms in same site: $\text{occA} + \text{occB} = 1$
- Rigid groups
- Constraints on ADPs (TLS groups)

Restraints

$$S = \sum_{\mathbf{h}} w_{\mathbf{h}} (|F_{\mathbf{h}}^o|^2 - |F_{\mathbf{h}}^c|^2)^2 + \sum_j w_j (T_j^o - T_j^c)^2$$

T_j^c is the value of some target function computed from the structural parameters

T_j^o is the corresponding value taken from some other source (databases, similar structures, computational chemistry, ...)

Restraints are generally applied under one (or both) of two situations:

- The starting model is very poor
- Normal refinement has produced an unacceptable structure

Refinement Strategies

■ Adapt strategy to model quality

Very **good model**



full anisotropic refinement straight away

Unreliable model
or poor/scarce data



a more **cautious approach** is required

- ◆ Initialize the scale factor and average isotropic ADP from the Wilson plot.
- ◆ Refine atomic positions first.
- ◆ Perform a few isotropic refinement cycles before introducing anisotropic ADPs.
- ◆ For structures with heavy atoms, refine heavy atoms anisotropically **before** extending to all atoms.
- ◆ Don't refine each stage to completion

Refinement Strategies

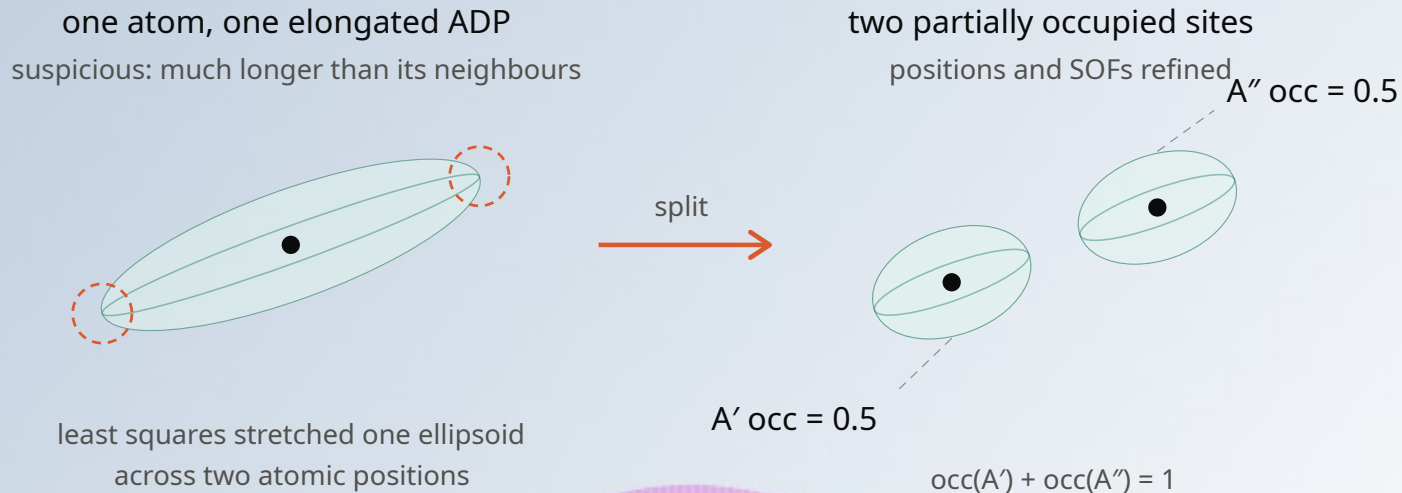
■ ADPs flag misassigned atom types

- ◆ Inspect ADPs carefully, they are powerful indicators of model errors.
- ◆ Unusually large ADPs may indicate an incorrectly assigned atom type.
e.g. if a C atom (six electrons) is erroneously assigned as an N atom (seven electrons), least squares will try to dissipate the extra electron by increasing the ADP
- ◆ Extremely large ADPs often reveal spurious atoms that should be removed.
- ◆ Least-squares refinement can eliminate false atoms, but it cannot create missing ones.

Refinement Strategies

■ ADPs

Highly elongated (cigar-shaped) ADPs usually indicate atom disorder or misidentified positions. Split the atom into two partially occupied sites with refined positions.



Refinement Strategies

■ Finding missing atoms

- ◆ Use chemical knowledge (e.g. complete a phenyl ring geometrically)
- ◆ Or locate them in a Fourier synthesis

■ Hydrogen atoms

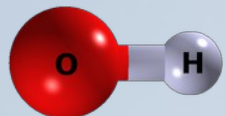
- ◆ Locate by difference fourier maps at the end of isotropic ADP refinement
- ◆ Place H geometrically, refine the bonded (heavier) atoms anisotropically

Refinement Strategies

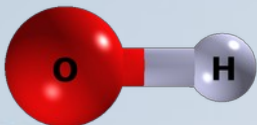
■ Hydrogen atoms refinement (3 strategies)

- ◆ Refine the H atoms with riding constraints
- ◆ Refine them with restraints
- ◆ Recompute their positions geometrically after each cycle of refinement

X-H (X-ray) < X-H (neutrons)



0,80–0,85 Å (X-ray)



~0,96–0,98 Å (neutrons, electrons)

References

- ♦ Watkin, D. J. (2008). Structure refinement: some background theory and practical strategies. *J. Appl. Cryst.* 41, 491–522.
<https://doi.org/10.1107/S0021889808007279>
- ♦ Clegg, W., Blake, A. J., Gould, R. O. & Main, P. (2001). *Crystal Structure Analysis: Principles and Practice*. Oxford University Press, Oxford.
- ♦ Giacovazzo, C., Monaco, H. L., Artoli, G., Viterbo, D., Ferraris, G., Gilli, G., Zanotti, G. & Catti, M. (2002). *Fundamentals of Crystallography*, 2nd edn. IUCr/Oxford University Press, Oxford.
- ♦ Müller, P., Herbst-Irmer, R., Spek, A. L., Schneider, T. R. & Sawaya, M. R. (2006). *Crystal Structure Refinement: A Crystallographer's Guide to SHELXL*. Oxford University Press, Oxford.
<https://doi.org/10.1093/acprof:oso/9780198570769.001.0001>

References

- ♦ Massa, W. (2004). Structure Refinement (Cap. 9). In Crystal Structure Determination, 2nd edn., Springer, Berlin/Heidelberg.
https://doi.org/10.1007/978-3-662-06431-3_9
- ♦ Hendrickson, W. A. (1985). Stereochemically Restrained Refinement of Macromolecular Structures. Methods in Enzymology, 115, 252–270.
[https://doi.org/10.1016/0076-6879\(85\)15021-4](https://doi.org/10.1016/0076-6879(85)15021-4)
- ♦ Tronrud, D. E. (2004). Introduction to macromolecular refinement. Acta Cryst. D60, 2156–2168. <https://doi.org/10.1107/S090744490402356X>
- ♦ Prince, E. & Boggs, P. T. (2006). Least squares (Cap. 8.1, pp. 678–688). In International Tables for Crystallography, Vol. C.
<https://doi.org/10.1107/97809553602060000609>

References

- ♦ Shmueli, U. (2007). Refinement of Structural Parameters (Cap. 10). In Theories and Techniques of Crystal Structure Determination, IUCr/Oxford University Press. <https://doi.org/10.1093/oso/9780199213504.001.0001>
- ♦ van der Maelen Uría, J. F. (1999). Current Methods and Optimization Algorithms for the Refinement of X-Ray Crystal Structures. Crystallography Reviews, 7(2), 125–180. <https://doi.org/10.1080/08893119908039929>