

Optimisation in Refinement

Garib N Murshudov

MRC-LMB

Cambridge

Outline

- **Introduction to optimisation techniques**
- **MX refinement**
- **Fast gradient calculation**
- **Information matrix for various likelihood**
- **Fast approximate Hessian and information matrix**
- **Constrained optimisation and stabilisation**

Introduction

Most of the problems in scientific research can be brought to problem of optimisation (maximisation or minimisation).

Formulation of the problem in a general form:

$$L(x) \rightarrow \min$$

$$C_i(x) = 0 \quad i=1, k$$

$$B_j(x) > 0, \quad j=1, m$$

If $m=0$, $k=0$ then the problem is called unconstrained optimisation, otherwise it is constrained optimisation

Introduction

Part 1) Unconstrained optimisation: gradients, Hessian, information matrix

Part 2) Constrained optimisation and stabilisation

Crystallographic refinement

The form of the function in crystallographic refinement has the form:

$$L(p) = L_x(p) + wL_G(p)$$

Where $L_x(p)$ is the -loglikelihood and $L_G(p)$ is the -log of prior probability distribution - restraints.

It is only one the possible formulations. It uses Bayesian statistics. Other formulation is also possible. For example: stat physical energy, constrained optimisation, regularisation.

-loglikelihood

-loglikelihood depends on the assumptions about experimental data, crystal contents and parameters. For example with the assumptions that all observations are independent (e.g. no twinning), there is no anomalous scatterers and no phase information available, for acentric reflections it becomes:

$$L_x(p) = \sum \frac{|F_o|^2 + |F_c|^2}{\Sigma} - \log(I_0(2|F_o||F_c|/\Sigma)) + \log(\Sigma) + \text{const}$$

And for centric reflections:

$$L_x(p) = \sum \frac{|F_o|^2 + |F_c|^2}{2\Sigma} - \log(\cosh(|F_o||F_c|/\Sigma)) + 0.5\log(\Sigma) + \text{const}$$

All parameters (scale, other overall and atomic) are inside $|F_c|$ and Σ

Note that these are loglikelihood of multiples of chi-squared distribution with degree of freedom 2 and 1

-loglikelihood

Sometimes it is easier to start with more general formulation.

$$L_X(p) = -\log\left(\int P(|F_o|; F)P(F; |F_c|, \Sigma) dF\right)$$

$|F_o|$ - observation. Could be amplitudes or intensities

$|F_c|$ - calculated structure factors

Σ - the second central moment. It has reflection multiplicity and other overall crystal and pseudo-crystal parameters

F - "true" structure factor. It is not observable and is integrated out.

But we want to calculate them

By changing the first and/or the second term inside the integral we can get functions for twin refinement, SAD refinement. By playing with the second central moments (variance) and overall parameters in $|F_c|$ we can get functions for pseudo-translation, modulated crystals. For ML twin refinement reftmac uses this form of the likelihood

Geometric (prior) term

Geometric (prior) term depends on the amount of information about parameters (e.g. B values, xyz) we have, we are willing (e.g. NCS restraints) and we are allowed (software dependent) to use. It has the form:

$$L_G(p) = \sum w_{b_i} (b_i - b_m)^2 + \sum w_{a_i} (a_i - a_m)^2 + \text{others}$$

b_i, b_m - ideal and model bond lengths

a_i, a_m - ideal and model angles

others may include: chiral volumes, planarities, B values, rigid bond for B values, NCS, torsion angles etc

Note that some of these restraints (e.g. bonds, angles) may be used as constraints also

**So, simple refinement can be thought of as unconstrained optimisation.
There is (at least) one hidden parameter - weights on X-ray. It could be
adjusted using ideas from constrained optimisation (Part 2)**

Introduction: Optimisation methods

Optimisation methods roughly can be divided into two groups:

- 1) **Stochastic**
- 2) **Detrministic**

Stochastic

- **Monte-Carlo and its variations**
- **Genetic algorithms and its variants**
- **Molecular dynamics**

Deterministic

- **Steepest descent, conjugate gradient and its variants**
- **Newton-Raphson method**
- **Quasi-Newton methods**
- **Tunnelling type algorithms for global optimisation (it can be used with stochastic methods also)**

Steepest descent: algorithm

- 1) Initialise parameters - p_0 , assign $k=0$
- 2) Calculate gradient - g_k
- 3) Find the minimum of the function along this gradient (i.e. minimise $(L(p_k - \alpha^* g_k))$)
- 4) Update the parameters $p_{k+1} = p_k - \alpha^* g_k$
- 5) If shifts are too small or the change of the function is too small then exit
- 6) Assign $k=k+1$, go to step 2

Unless parameters are uncorrelated (i.e. the second mixed derivatives are very small) and are comparable in values this method can be painfully slow. Even approximate minimum is not guaranteed.

Conjugate gradient: algorithm

- 1) Initialise parameters, p_0 , $k=0$
- 2) Calculate gradients g_0
- 3) Assign $s_0 = -g_0$
- 4) Find the minimum $L(p_k + \alpha s_k)$
- 5) Update parameters $p_{k+1} = p_k + \alpha_k s_k$
- 6) If converged exit
- 7) Calculate new gradient - g_{k+1}
- 8) Calculate: $\beta_k = (g_{k+1}, g_{k+1} - g_k) / (g_k, g_k)$
- 9) Find new direction $s_{k+1} = -g_{k+1} + \beta_k s_k$
- 10) Go to step 4

Conjugate gradient: Cont

Some notes:

- 1) It is the most popular technique
- 2) Variation of search direction calculation is possible
- 3) If it is used for multidimensional nonlinear function minimisation then it may be necessary to restart the process after certain number of cycles
- 4) It may not be ideal choice if parameters are wildly different values (e.g. B values and xyz)
- 5) When used for linear equation solution with symmetric and positive matrix then step 4 in the previous algorithms becomes calculation of $\alpha_k = (\mathbf{H}_k \mathbf{p}_k + \mathbf{g}_k, \mathbf{s}_k) / (\mathbf{s}_k, \mathbf{H}_k \mathbf{s}_k)$

(Quasi-)Newton 1: algorithm

- 1) Initialise parameters: $p_0, k=0$
- 2) Calculate gradients - g_k and (approximate) second derivatives H_k . It could be diagonal
- 3) Solve the equation $H_k s = -g_k$
- 4) Find the minimum $L(p_k + \alpha s_k)$ and denote $p_{k+1} = p_k + \alpha s_k$
- 5) Check the convergence
- 6) Update “Hessian” H_{k+1} . Options: 1) second derivative matrix, 2) information matrix, 3) BFGS type where H_{k+1} depends on H_k , gradients and shifts
- 7) Go to step 3

(Quasi-)Newton 2: algorithm

- 1) Initialise parameters: p_0 , $k=0$
- 2) Calculate gradients - g_k and (approximate) inversion of second derivatives B_k . It could be diagonal
- 3) Calculate $s_k = -B_k g_k$
- 4) Find the minimum $L(p_k + \alpha s_k)$ and denote $p_{k+1} = p_k + \alpha s_k$
- 5) If converged exit
- 6) Update “inverse Hessian” B_{k+1} . Options: 1) inversion of second derivative matrix, 2) inversion of information matrix, 3) BFGS type where B_{k+1} depends on B_k , gradients and shifts
- 7) Go to step 3

Note 1: If you are working with sparse matrix then be careful: usually Hessian (also known as interaction matrix) is more sparse than its inverse (also known as covariance matrix). It may result in unreasonable shifts.

Note 2: In general conjugate gradient and this technique with diagonal initial values are similar.

Note 3: Preconditioning may be difficult

Note 4: BFGS belongs to this family of optimisations, LBFGS builds the matrix B implicitly

Tunnelling type minimisation: algorithm

- 1) Define $L_0=L$, $k=0$
- 2) Minimise L_k . Denote minimum p_k
- 3) Modify the function: $L_{k+1}=L_k+f(p-p_k)$. Where f is a bell shaped function (e.g. Gaussian)
- 4) Minimise L_{k+1}
- 5) If no solution then exit
- 6) Increment k by 1 and go to step 2

Note on local minimisers

In good optimisers local minimisers form a module in a larger iterations. For example in tunnelling type optimisers, unconstrained optimisers etc.

Conjugate gradient can also be used as a part of (quasi-) Newton methods. In these methods one needs to solve the linear equation and it can be solved using conjugate gradient (see below)

Inverting large matrices: Stochastic approach

Most of (quasi)Newton methods are brought to the solution of a linear system

$$\mathbf{Ax} = -\mathbf{g}$$

Where \mathbf{x} is shift vector and \mathbf{g} is gradient. \mathbf{A} is a symmetric matrix. In many cases we can make sure that \mathbf{A} is non-negative.

There are several ways of solving this type of equations:

- 1) Invert matrix \mathbf{A} and $\mathbf{x} = -\mathbf{A}^{-1}\mathbf{g}$. Often \mathbf{A} is very large, it is not practical to use this approach. \mathbf{A} may not be positive definite. Then we can use Tikhonov regulariser or similar**
- 2) Eigen value filtering: It is useful for small systems**
- 3) Conjugate gradient to solve linear equation: We do not get inversion directly**
- 4) Matrix inversion using stochastic methods**

Inverting large matrices: Stochastic approach

Gibbs sampler

Let us assume we have A and we want to calculate $B=A^{-1}$. If A is positive definite matrix then it defines a gaussian distribution

$$P(x) = N \exp(-x^T A x) \quad (1)$$

Elements of B – b_{ij} can be written (they are elements of the covariance matrix):

$$b_{ij} = E(x_i x_j)$$

Where x_i, x_j elements of the random vector with probability distribution (1). Using this fact and Gibbs sampler it is possible to calculate approximation to the inverse of the matrix. Gibbs sampler based on the conditional distribution of $x_i | x_{-i}$, where $-i$ is indices not including i . When A is sparse then conditional distribution will depend only on few neighbouring elements.

Newton-Raphson method

Taylor expansion of the objective function $f(\underline{x})$ around a working \underline{x}_k

$$\underline{H}\underline{s} = -\underline{g}$$

\underline{H}

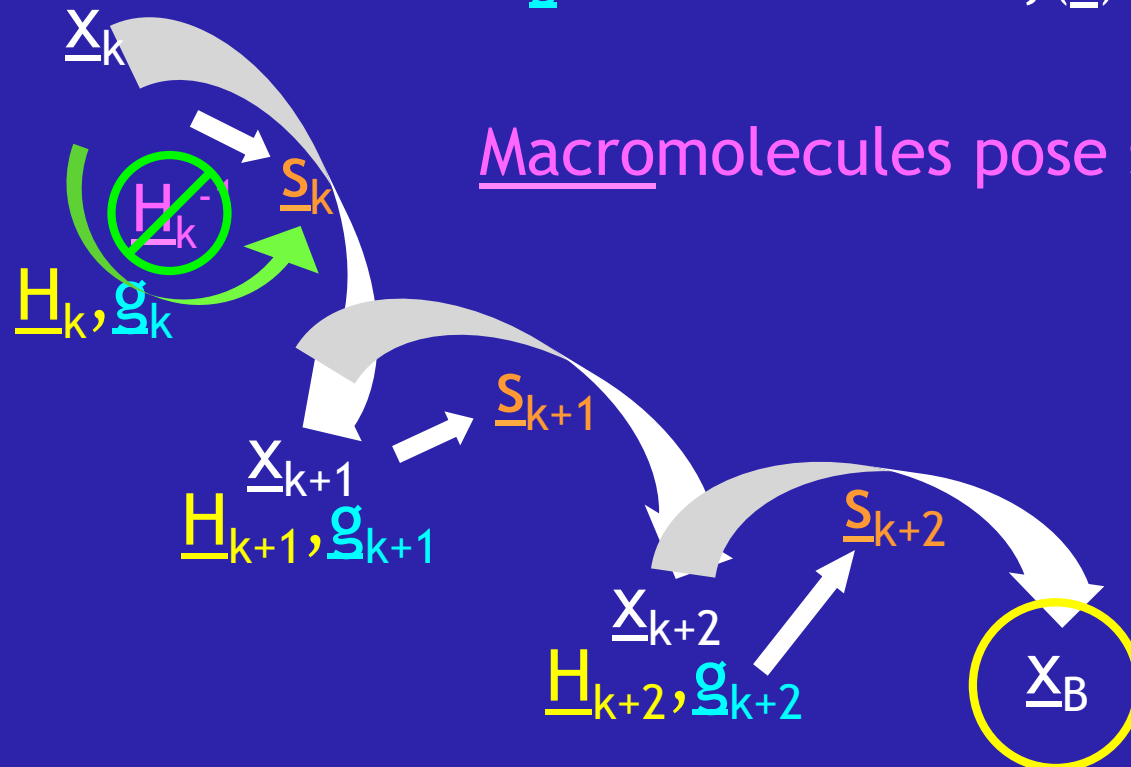
Second-derivative matrix of $f(\underline{x})$

\underline{s}

Shift vector to be applied to \underline{x}_k

\underline{g}

Gradient of $f(\underline{x})$



Macromolecules pose special problems

Macromolecules

The calculation and storage of \underline{H} (\underline{H}^{-1}) is very expensive

\underline{H} in isotropic refinement has $4N \times 4N$ elements
2500 atoms \rightarrow 100 000 000 elements

$$\begin{pmatrix} \frac{\partial^2 f}{\partial p_1 \partial p_1} & \cdot & \cdot & \cdot & \frac{\partial^2 f}{\partial p_1 \partial p_{10000}} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{\partial^2 f}{\partial p_{10000} \partial p_1} & \cdot & \cdot & \cdot & \frac{\partial^2 f}{\partial p_{10000} \partial p_{10000}} \end{pmatrix}$$

\underline{H} in anisotropic refinement has $9N \times 9N$ elements
2500 atoms \rightarrow 506 250 000 elements

Direct calculation

$$\text{time} \propto N_{\text{el}} \times N_{\text{refl}}$$

FFT methods

$$\text{time} \propto c_1 N_{\text{el}} + c_2 N_{\text{refl}} \log N_{\text{refl}}$$

[Agarwal, 1978]

[Murshudov et al., 1997]

[Tronrud, 1999]

[Urzhumtsev & Lunin, 2001]

Fast gradient calculation

The form of the function as sum of the individual components is not needed for gradient.

In case of SAD a slight modification is needed

$$L = \frac{1}{2} \sum L_h, \quad L_h = L_{-h}$$

$$\frac{\partial L}{\partial x_{ij}} = \sum \left(\frac{\partial L}{\partial A_h} - i \frac{\partial L}{\partial B_h} \right) \frac{\partial F_h}{\partial x_{ij}} = \iiint \mathfrak{S} \left(\frac{\partial L}{\partial A_h} + i \frac{\partial L}{\partial B_h} \right) \mathfrak{S} \left(\frac{\partial F_h}{\partial x_{ij}} \right) dx$$

$$\mathfrak{S} \left(\frac{\partial F_h}{\partial x_{ij}} \right) = \frac{\partial \rho_i(x)}{\partial x_{ij}}$$

L_h - component of the likelihood depends only on the reflection with Miller index h

x_{ij} - j -th parameter of i -th atom

ρ_i - i -th atoms

Gradients of likelihood

We need to calculate the gradients of likelihood wrt structure factors.

In a general form:

$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = \int \left(\frac{\partial L_2}{\partial A_c} + i \frac{\partial L_2}{\partial B_c} \right) P(F; |F_o|, |F_c|, \Sigma) dF$$
$$L_2 = -\log(P(F; |F_c|, \Sigma))$$

If the second term is Gaussian then calculations reduce to

If

$$P(F; F_c, \Sigma) = \prod_{\Sigma}^c \exp\left(-\frac{(F - F_c)^2}{\Sigma}\right)$$

then

$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = \frac{F_c - \int F P(F; |F_o|, F_c, \Sigma) dF}{\Sigma}$$

Gradients: real space fit

For model building gradient has particularly simple form.
In this case minimisation tries to flatten the difference map.

$$L = \frac{1}{2} \int (\rho_o(x) - \rho_c(x))^2 dx \propto \frac{1}{2} \sum (F_o - F_c)^2$$
$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = F_c - F_o$$

Gradients: least-squares

In this case difference map with calculated phases is flattened

$$L = \frac{1}{2} \sum (|F_o| - |F_c|)^2$$
$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = (|F_c| - |F_o|) \exp(i\varphi_c)$$

Gradients: Maximum likelihood

In this case weighted difference map is flattened

$$L_x(p) = \sum \frac{|F_o|^2 + |F_c|^2}{\Sigma} - \log(I_0(2|F_o||F_c|/\Sigma)) + \log(\Sigma) + const$$

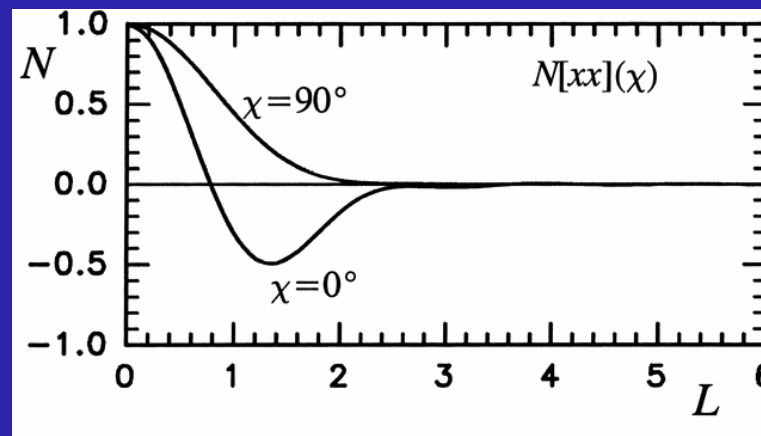
$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = \frac{2}{\Sigma} (|F_c| - m |F_o|) \exp(i\varphi_c)$$

and in more general form

$$\frac{\partial L}{\partial A_c} + i \frac{\partial L}{\partial B_c} = \frac{2}{\Sigma} (F_c - m \langle F \rangle)$$

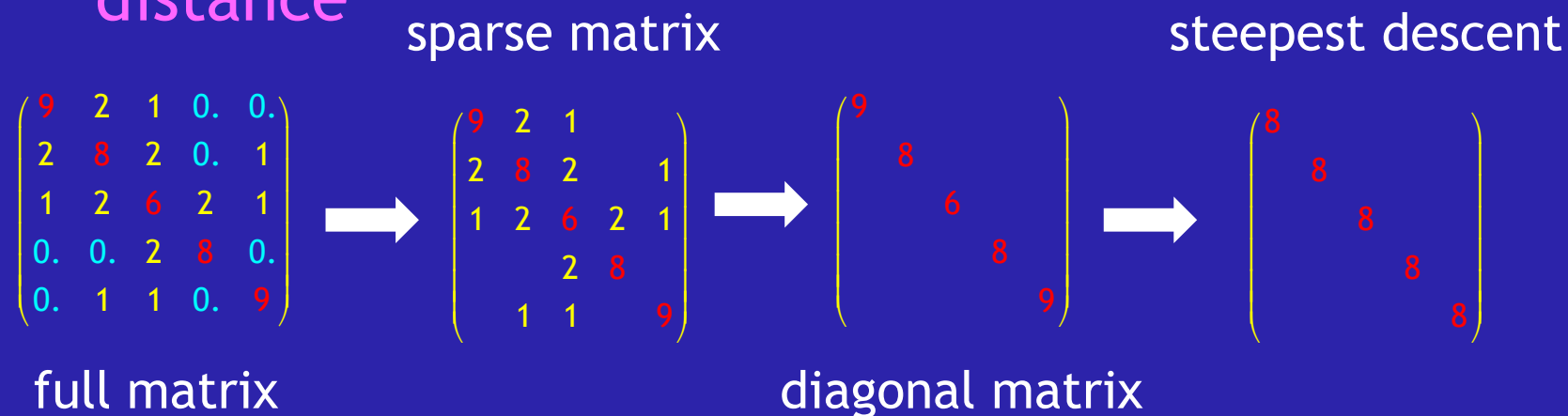
where $\langle F \rangle$ is the expectation value of "true" structure factors with probability distribution of "true" structure factors given observations, phase information, calculated structure factors and other overall parameters
Refmac uses this form for twinned refinement against unmerged data with intensities as observations

Approximations to Hessians



[Templeton, 1999]

The magnitude of matrix elements decreases with the lengthening of the interatomic distance



Sparse matrix in refinement

- 1) The set of the contacting atoms. Denote this set by *list*. These include all pairs of atoms related by restraints: bonds, angles, torsions, vdws, ncs etc
- 2) Size of each parameter (for positional 3, for B values 1, for aniso U values 6, for occupancies 1)
- 3) Design the matrix

H_{ii} are quadratic matrices. They reflect interaction of atom with itself

H_{ij} are rectangular matrices. They reflect interaction between contacting atoms

$$\begin{pmatrix} H_{11} & & & H_{1j} & & \\ & H_{22} & & & & \\ & & & \ddots & & \\ H_{j1} & & & & & \\ & & & & & \\ & & & & & H_{nn} \end{pmatrix}$$

The number of the elements need to be calculated:

$$\sum n_i^2 + \sum_{(i,j) \in list} n_i n_j$$

Preconditioner

We need to solve

$$Hs = -g$$

When parameter values are wildly different (e.g. B values and xyz) then this equation can be numerically ill-conditioned and few parameters may dominate. To avoid this, preconditioning could be used.

The equation can be solved in three steps: 1) Precondition 2) Solve 3) remove conditioning.

The algorithm

$$P^T H P P^{-1} s = -P^T g$$

$$H_1 s_1 = -g_1$$

$$s = P s_1$$

$$H_1 = P^T H P$$

$$g_1 = P^T g$$

Algorithm

- 1) Calculate preconditioner
- 2) Calculate H_1 and g_1
- 3) Solve the linear equation (e.g. using conjugate gradient)
- 4) Remove conditioner

Preconditioner for refinement

Inversion and square root should be understood as pseudo-inversion and that for matrices.

If any singularities due to a single atom then they can be removed here.

The resulting matrix will have unit (block)diagonal terms

$$P = \begin{pmatrix} P_{11} & & & \\ & P_{22} & & \\ & & \ddots & \\ & & & P_{nn} \end{pmatrix}$$
$$P_{ii} = \sqrt{H_{ii}^{-1}}$$

Fisher's method of scoring (scoring method)

The objective function $f(\underline{x})$ is the likelihood $L = -\log P(\underline{o}; \underline{p})$



[Fisher, 1925]

$$\underline{Hs} = -\underline{g} \quad \longrightarrow \quad \underline{Is} = -\underline{g}$$

$$\underline{g} = \frac{\partial L}{\partial \underline{p}}$$

Score vector

$$\underline{H} = \frac{\partial^2 L}{\partial \underline{p} \partial \underline{p}^T}$$

Observed information matrix

$$\underline{I} = \left\langle \frac{\partial^2 L}{\partial \underline{p} \partial \underline{p}^T} \right\rangle$$

Fisher's information matrix

$$\underline{I} = \left\langle \underline{g}(\underline{p}) \underline{g}(\underline{p})^T \right\rangle$$

$$\langle \xi \rangle = \int \dots \int \xi e^{-L} d\omega_1 \dots d\omega_n$$

Positive semidefinite

Example of information

Assume that the distribution is von Mises. I.e the distribution of observation (θ) given model (μ) is

$$P(\theta;\mu) = \exp(X \cos(\theta - \mu)) / I_0(X)$$

Where $I_k(X)$ is k-th order modified Bessel function of the second kind

Then the -loglikelihood function

$$L(\theta;\mu) = -X \cos(\theta - \mu) - \log(K)$$

Gradient:

$$g(\mu) = X \sin(\theta - \mu)$$

Observed information

$$\underline{H} = X \cos(\theta - \mu) - \text{could be negative as well as positive (from } -X \text{ to } X)$$

Expected Information

$$\underline{I} = X I_1(X) / I_0(X)$$

It is never negative. It can be 0 for uniform distribution only, i.e. $X=0$

Example: Cont

If parameters are too far from optimal (e.g. around $p = o + \pi$) then method using the second derivative (observed information) would not go downhill. If expected information is used then shifts are always downhill.

Near to the maximum, observed information is X and expected information is $m X$ where $m = I_1(X) / I_0(X) < 1$. So shifts calculated using expected information is larger than shifts calculated using observed information therefore oscillation around the maximum is possible.

Information matrix and observations

Information matrix does not depend on the actual values of observations. It depends on their conditional distribution only. This fact could be used for planning experiment.

Note that in case of least-squares (i.e. gaussian distribution of observations given model parameters) the normal matrix is expected as well as observed information matrix

Information matrix and change of variables

If variables are changed to new set of variables there is a simple relation between old and new information matrix

If $q = q(p)$

$$I_{p_i p_j} = \sum I_{q_k q_l} \frac{\partial q_k}{\partial p_i} \frac{\partial q_l}{\partial p_j}$$

$I_{p_i p_j}$, and $I_{q_i q_j}$ - elements of information matrix for p and q variables

$\frac{\partial q_k}{\partial p_i}$ - elements of Jacobian matrix

As a result information matrix can be calculated for “convenient” variables and then converted to desired ones. For example in crystallography it can be calculated for calculated structure factors first.

Fisher's information in crystallography

$$L = \frac{1}{2} \sum_{\underline{h}} L_{\underline{h}} \quad \begin{array}{l} L_{\underline{h}} = L_{-\underline{h}} \\ F_{\underline{h}} = F_{-\underline{h}}^* \end{array} \quad (\text{no anomalous scatterers})$$

$$I_{p_i(n), p_j(m)} = \left\langle \frac{\partial^2 L}{\partial p_i \partial p_j} \right\rangle =$$

$$= \sum_{\underline{h}} \delta_{nm} \left\langle \frac{\partial L}{\partial A_{\underline{h}c}} - i \frac{\partial L}{\partial B_{\underline{h}c}} \right\rangle \left(\frac{\partial^2 F_{\underline{h}c}}{\partial p_i \partial p_j} \right) + \quad I_0$$

Likelihood
weighting
factor
 $W_{\underline{h}}$

$$+ \sum_{\underline{h}} \frac{1}{2} \left\langle \frac{\partial^2 L}{\partial A_{\underline{h}c}^2} + \frac{\partial^2 L}{\partial B_{\underline{h}c}^2} \right\rangle \left(\frac{\partial F_{\underline{h}c}}{\partial p_i} \right) \left(\frac{\partial F_{\underline{h}c}}{\partial p_j} \right)^* + \quad I_1$$

$$+ \sum_{\underline{h}} \frac{1}{2} \left\langle \frac{\partial^2 L}{\partial A_{\underline{h}c}^2} - \frac{\partial^2 L}{\partial B_{\underline{h}c}^2} - 2i \frac{\partial^2 L}{\partial A_{\underline{h}c} \partial B_{\underline{h}c}} \right\rangle \left(\frac{\partial F_{\underline{h}c}}{\partial p_i} \right) \left(\frac{\partial F_{\underline{h}c}}{\partial p_j} \right) \quad I_2$$

Approximate information matrix

$$I_{p_i, p_j} \cong \sum_{\underline{h}} w_s \left(\frac{\partial F_{hc}}{\partial p_i} \right) \left(\frac{\partial F_{hc}}{\partial p_j} \right)$$

$$I_{p_i(n)p_j(m)} \cong K_{p_i p_j} q_n q_m \sum_{\underline{h}} w_s H_{p_i p_j} f_n^0 f_m^0 t_n t_m \text{trig}_{p_i p_j} (2\pi \underline{h} \underline{D}_{nm})$$

p_i	p_j	$K_{p_i p_j}$	$H_{p_i p_j}$	$\text{trig}_{p_i p_j}$
x_i	x_j	$2\pi^2$	$h_i h_j$	cos
B	B	1/32	$ s ^4$	cos
U_{ij}	U_{kl}	$2\pi^4 c_{ij} c_{kl}$	$h_i h_j h_k h_l$	cos
q	q	1	1	cos
x_i	B	$\pi/4$	$h_i s ^2$	sin
x_i	U_{kl}	$2\pi^3 c_{kl}$	$h_i h_k h_l$	sin
x_i	q	π	h_i	sin
B	U_{kl}	$c_{kl}/4$	$ s ^2 h_k h_l$	cos
B	q	1/8	$ s ^2$	cos
U_{ij}	q	$\pi^2 c_{ij}$	$h_i h_j$	cos

Information matrix

Let us consider the information matrix:

$$I_{p_i(n)p_j(m)} \cong K_{p_i p_j} q_n q_m \sum_{\underline{h}} W_s H_{p_i p_j} f_n^0 f_m^0 t_n t_m \text{trig}_{p_i p_j} (2\pi \underline{h} \underline{D}_{nm})$$

Only component dependent on observation is W_s

$$W_s = \int \left(\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial B_c^2} \right) \exp(-L) d|F_o|$$

L - - loglikelihood function

$|F_o|$ - observations

A_c, B_c - real and imaginary parts of the calculated structure factors

Different type of refinement uses different likelihood function and therefore different W_s . Let us consider some of them

Information: real-space fit

In this case differences between “observed” and calculated electron density is minimised

$$L = \int (\rho_o(x) - \rho_c(x))^2 dx \propto \sum (F_o - F_c)^2$$
$$\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial A_c^2} = 1$$

There is no dependence on observations. So all components of information matrix can be tabulated only once.

Information: least-squares

In this case the distribution of observation given calculated structure factors is Gaussian:

$$L = \frac{1}{2} \sum (|F_o| - |F_c|)^2$$
$$\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial B_c^2} = 1 + \frac{(|F_c| - |F_o|)}{|F_c|}$$
$$\int \left(\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial B_c^2} \right) P(|F_o|; |F_c|) d|F_o| = 1$$

Again there is no dependence on observations. This formulations can be modified by addition of weights for each observation. Then dependence of information matrix will be on these weights only.

For the most cases modified (or iteratively weighted) least-squares approximates more sophisticated maximum likelihood very well.

If weights depend on the cycle of refinement then calculation can be done at each cycle, otherwise only once.

Information: least-squares

Note that expected information removes potentially dangerous term dependent on $|F_c| - |F_o|$ and makes the matrix positive

Information: Maximum likelihood

The simplest likelihood function has a form (acentric only):

$$L_x(p) = \sum \frac{|F_o|^2 + |F_c|^2}{\Sigma} - \log(I_0(2|F_o||F_c|/\Sigma)) + \log(\Sigma) + const$$

Corresponding term for information matrix:

$$\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial B_c^2} = 2\left(\frac{2}{\Sigma} - \frac{m|F_o|}{\Sigma|F_c|} - 2\frac{m'}{\Sigma^2}|F_o|^2\right)$$

To calculate

$$\int \left(\frac{\partial^2 L}{\partial A_c^2} + \frac{\partial^2 L}{\partial B_c^2}\right) P(|F_o|; |F_c|) d|F_o|$$

we need to calculate integrals

$$\int m|F_o| P(|F_o|; |F_c|) d|F_o| \text{ and } \int m'|F_o|^2 P(|F_o|; |F_c|) d|F_o|$$

Here m is figure of merit and m' is its first derivative

These integrals or corresponding form using first derivatives can be calculated using Gaussian integration

Information: maximum likelihood

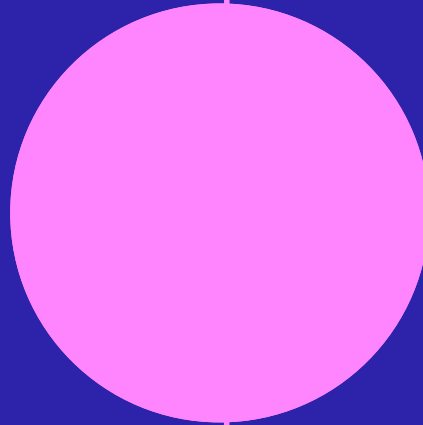
Alternative (simpler?) way of calculations (refmac uses this form):

$$\int \left(\left(\frac{\partial L}{\partial A_c} \right)^2 + \left(\frac{\partial L}{\partial B_c} \right)^2 \right) P(|F_o|; |F_c|) d|F_o| = \frac{4}{\Sigma^2} \int (m|F_o| - |F_c|)^2 P(|F_o|; |F_c|) d|F_o|$$

Again Gaussian integration can be used. Note the the result of this integral is always non-negative. Refmac uses Gaussian integration designed for this case.

Integral approximation of I

$$I_{p_i(n)p_j(m)} \cong K_{p_i p_j} q_n q_m \sum_h W_s H_{p_i p_j} f_n^0 f_m^0 t_n t_m \text{trig}_{p_i p_j} (2\pi h D_{nm})$$



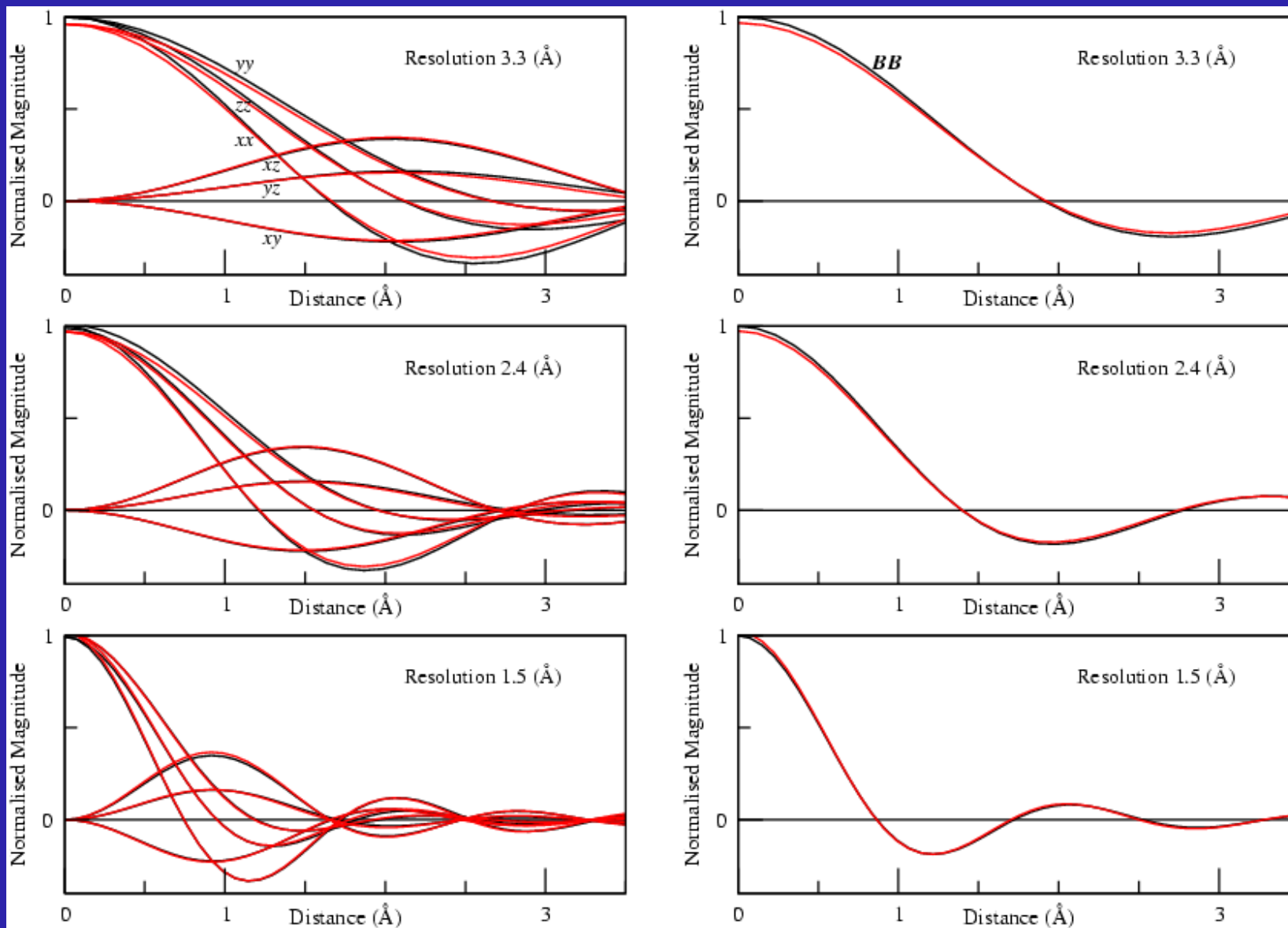
$$I_{p_i(n),p_j(m)} \cong K_{p_i p_j} q_n q_m \int_{\text{res. sphere}} W_s H_{p_i p_j} f_n^0 f_m^0 t_n t_m \text{trig}_{p_i p_j} (2\pi h D_{nm})$$

[Agarwal, 1978]

[Dodson, 1981]

[Templeton, 1999]

Analytical I versus integral I



Block diagonal version

Because of sharp decrease of the components of the information matrix vs the distance between atoms and there are almost no atoms closer than 1.2Å, using only diagonal terms of the information matrix works very well. Their calculation is extremely fast. Much faster than gradient calculations.

Calculation of the sparse information matrix with pretabulation is also extremely fast

Fast evaluation of I

Two-step procedure

1. **Tabulation step** - a limited set of integrals are tabulated in a convenient coordinate system
2. **Rotation step** - the matrix element in the crystal system is calculated from the tabulated values using a rotation matrix

Summary

- Gradients can be calculated very fast using FFT
- Use of Fisher information matrix improves behaviour of optimisation
- The integral approximation allows the calculation of a sparse Fisher's information in 'no time'
- The scoring method using sparse fast-evaluated Fisher's information has been implemented in the program *REFMAC5*
- The method works satisfactorily

Books and other references

1. **Nocedal J, Wright SG, Numerical optimisation**
2. **Press WH, Teukolsky SA, Vetterling WT, Flannery BP, Numerical Recipes: The Art of Scientific Computing**
3. **Geman S, Geman D, (1984) “Stochastic Relaxation Gibbs Distributions, and the Bayesian Restoration of Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence 6(6):721-741**
4. **Harville DA (1999) “Use of the Gibbs sampler for large possibly sparse, positive definite matrices” Linear Algebra and its Applications 289:203-224**
5. **Garcia Cortes LA, Garbillo C (2008) A Monte Carlo algorithm for efficient large matrix inversion, <http://arxiv.org/abs/cs/0412107>**
6. **Kendal MG, Stuart A, Ord JK, Kendall’s advanced theory of statistics, volume 2**

Acknowledgements

- **Roberto Steiner** – implementation of information matrix
- **Andrey Lebedev** – Discussion and for many ideas
- **Wellcome Trust** – money
- **BBSRC** – money

Part 2

Constrained optimisation and stabilisation

Constrained optimisation

Now consider the minimisation problem with equality constraints

$$L(p) \rightarrow \min$$

$$C_i(p) = 0, i = 1, m$$

The simplest approach to this problem is change of variables. I.e. if we can find new variables (q) so that

$$p = p(q)$$

$$C_i(p(q)) \equiv 0$$

Then we can use chain rule to calculate derivatives wrt q and do minimisation. It can be done for simplest case, e.g. rigid body refinement, TLS refinement and in some software torsion angle refinement (CNS and phenix.refine). In general case it is not that easy to find new variables. We need alternative methods.

Lagrange multipliers

Replace the original problem with

$$T(p) = L(p) + \sum \lambda_i C_i(p)$$

Find stationary points of this equation. I.e. solve

$$\frac{\partial T}{\partial p_j} = 0, \quad \frac{\partial T}{\partial \lambda_i} = 0$$

It is an irreplaceable technique in proving something or as a tool to derive equations. For example finding optimal rotation between two given coordinate sets. But it is not very useful for practical constrained optimisation.

Penalty functions

Add constraints as penalty functions and minimise the new function:

$$T(p) = L(p) + \sum \frac{1}{\sigma_{ki}^2} C_i^2(p) \rightarrow \min$$

Choose the sequence of $k \rightarrow \infty \Rightarrow \sigma_{ki} \rightarrow \infty$

I.e. choose the initial values of weights, minimise and update weights

It is very similar to what we use for restrained refinement with one exceptions that weights on geometry are adaptive.

It works in many cases however when weights become very small then the problem becomes numerically ill-conditioned.

Augmented Lagrangian

This technique combines penalty function and Lagrange multiplier techniques. It meant to be very efficient.

With some modification it can be applied to refinement and other constrained optimisation problems.

The new function is:

$$T(p, \lambda, \mu) = L(p) + \sum \lambda_{ki} C_i(p) + \sum \frac{1}{2\sigma_{ki}^2} C_i^2(p) \rightarrow \min$$

and update λ and σ after each minimisation step.

Augmented lagrangian: algorithm

- 1) Choose convergence criteria, initial parameters $p_0, \lambda_{0i}, \sigma_{0i}$
- 2) Find the minimum of $T(p, \lambda, \sigma)$ wrt p with soft criteria: I.e. if $|T| < t_k$ terminate
- 3) If t_k is small exit
- 4) Update Lagrange multipliers $\lambda_{k+1i} = \lambda_{ki} + 1/\sigma_{ki}^2 C_i(p_k)$
- 5) Update σ_{k+1i} . If corresponding constraint is good enough do not change it, otherwise reduce it
- 6) Go to step 2

Augmented Lagrangian: Cont

If this technique is applied as it is then it may be very slow. Instead the modified algorithm could be used for MX refinement.

- 1) Choose constraints**
- 2) Calculate gradients and (approximate) second derivatives for remaining restraints and contribution from likelihood**
- 3) Form the function $L(p) = 1/2 p^T H p + p^T g$**
- 4) Apply augmented Lagrangian to function $L(p)$. Use preconditioned conjugate gradient linear equation solver.**

Note: For linear constraints the algorithm becomes very simple.

Stabilisation

One of the problems in optimisation with (approximate) second derivative is the problem of ill-conditioned matrices. In refinement we use

$$L(s) = \frac{1}{2} s^T H s + s^T g \rightarrow \min$$

Where H is (approximate) Hessian or information matrix, g is gradient and s is shift to be found. If H is ill-conditioned (i.e. some of the eigenvalues are very small) then the problem of optimisation becomes ill-defined (small perturbation in the inputs may cause large variation in the outputs).

Causes of instability

There are two main reason of instability:

- 1) Low resolution. Then fall-off matrix elements vs distance between atoms is not very fast and values of matrix elements are comparable.**
- 2) Atoms are very close to each other. Then corresponding elements of non-diagonal terms are very close to diagonal terms. In other words atoms interact with each other very strongly.**
- 3) Wildly different parameter values and behaviours (B value, xyz)**

Stabilisation

There are several techniques for stabilisation of ill-defined problem (in mathematics it is called regularisation).

- 1) Explicit reparameterisation. If it is possible to change variables and reduce dimension of the problem then it can be done so. Examples: rigid body refinement, TLS, torsion angle refinement**
- 2) Eigen value filtering: Calculate eigenvalues of the matrix and remove small eigenvalues thus reduce dimensionality of the problem. It is implicit reparameterisation**
- 3) Tikhonov's regulariser**

Regularisers

If the problem is ill-defined then the function can be replaced by the new function:

$$L_{\alpha}(s) = L(s) + \frac{1}{2}\alpha s^T s$$

It has effect of increasing diagonal terms of the matrix.

This technique has many names in different fields: In statistics - Ridge regression, in numerical analysis Levenberg–Marquardt, in mathematical physics - Tikhonov's regulariser.

Stabiliser and preconditioner

When applied as is the stabiliser has the effect on all parameters. But it may turn out that the problem is only due to few atoms. In this case it is better to apply stabiliser after preconditioning.

Stabiliser can also be done in two stages:

- 1) During preconditioning, blocks where ill-conditioning happens can be dealt with using eigen value filtering. It would deal with such problems as those in special positions**
- 2) During linear equation solution stabiliser can be added after preconditioning.**

Acknowledgements

- **Roberto Steiner** – implementation of information matrix
- **Andrey Lebedev** – Discussion and for many ideas
- **Wellcome Trust** – money
- **BBSRC** – money