Fitting SANS data – an introduction and some practical advice

Dr. Richard Heenan
ISIS Facility,
Rutherford Appleton Laboratory

richard.heenan@stfc.ac.uk

INTRODUCTION

- **SANS results always tell you something !**
- **Understanding and fitting the results often takes far longer than preparing the samples or collecting the data !**
- **There is a lot of help available – but it can be hard to find !**
- **Don't waste good data …**

Outline:

1. **Fitting methods**
2. **Review "least squares" and "numerical integration"**
3. **Some fitting programs.**
4. **IFT – inverse Fourier transform.**
5. **One example FISH**

1. Fitting methods

- Simple but effective detective work: D = 2π/Qpeak, "straight line fits" on special axes – Guinier, Porod, Kratky, log-log, etc. – before computers the best you could do. Also vary concentration, temperature etc.
- Model fits by : trial and error,
  grid search,
  least squares,
  Levenberg-Marquardt.
- IFT – Fourier transform I(Q) to real space.

TRIAL & ERROR ?

e.g. program SANS equations in an Excel spread sheet and use trial and error to "fit" the data.
I do not advise this, except as a good way to get a feel for a simple problem, or to generate test data for a fitting routine.
Proper fits need proper weighting schemes and carefully debugged software (lot of possibilities for overflow & underflow etc).
Many models need a good numerical integration algorithm.

GRID SEARCH ?

An automated grid search can try all possible combinations of parameters – better as part of a more sophisticated algorithm – see EMBL MIXTURE program.

2(a) LEAST SQUARES FITS

Suppose we have data y_i where i = 1 to N at points Q_i
Calculated data $CALC_i$ = function(Q_i, a_1, a_2, ...a_M)
has M parameters a_j where j = 1 to M,

"Best fit" has lowest value of a "merit function" e.g.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - CALC_i}{\sigma_i} \right)^2$$

Minimise chi squared – but why ?
Statistical theories say:
If errors (uncertainites, standard deviations) σ_i are *independent* and have a *normal (Gaussian) distribution* then the minimum χ^2 is "the most likely" solution, and χ^2/(N-M) ~ 1.

(Normal distribution has ±2σ 68% of the time, ±3σ 95% of the time.)

LEAST SQUARES EQUATION

In matrix form, with N data, M parameters a_j

Parameter shifts, Δa, column vector of M rows is:

$$\Delta a = (D^T WD)^{-1} (D^T WE)$$

D^TWD the "least squares matrix",

Derivative matrix D (or J for Jacobian) of each calculated data point with respect to each adjusting parameter has N rows x M columns,

weights W_{ii} = 1/ σ_i^2 are diagonal N x N,

differences E = (obs - calc) is a column of N rows.

(Not that hard to write your own program, or to use a "package" from Numerical Recipes or similar.)

"Numerical Recipes in C++: The Art of Scientific Computing" William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery; *or* similar for C, Fortran 90, Fortran 77 etc.

LEAST SQUARES FITS to SANS & SAXS data

- SANS data has Poisson ($\sigma = \sqrt{\text{counts}}$) *not* Normal statistics, and neighbouring points may be correlated due to Q resolution and systematic errors. (SAXS data is often presented with too many, rather noisy data points.)
- Least squares "error estimates" for our model parameters, are thus often inaccurate as simple trial and error around best fit will show.
- Also "outliers" – points well away from the rest of the data seem to count "too much" and may need to be removed.
- M x M matrix $C = (D^TWD)^{-1}$ is variance-covariance matrix, Diagonal elements are parameter variance, $C_{jj} = \sigma_j^2$ - *assuming* normal errors etc.
  Off-diagonal C_{ij} give *correlation coefficients*, - identify poor parametrisation of model. Eigenvectors of C can show best & worst determined parameter combinations.
- Fits are almost always "non-linear" in the parameters so have to be iterated.

CONSTRAINTS

Many SANS problems are under-determined, "constraints" based on known or expected information are needed. Good model fitting programs allow you to add or remove constraints.

e.g. forcing Vshell/Vcore to an expected value, needs Rcore = f.Router where either Rcore or Router may be polydisperse.

e.g. particle with surfactant stabilser, Router = Rcore + Tshell where Tshell is fixed and again either Rcore or Router may be polydisperse.

e.g. requiring a micelle radius to be less than the expected stretched length of a surfactant molecule forces ellipsoidal or cylindrical structures over spheres.

e.g. conditions on a particle size distribution (such as "Maximum Entropy") - may be regarded as modifying the "merit function".

Some times when close to a "good fit" constraints can be relaxed.

Even with constraints a standard least squares fit can be badly behaved, so :

Steepest Descent & the Marquardt-Levenberg method

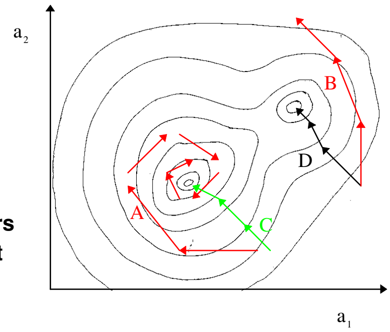
Following "steepest descent" of χ^2 surface may reach minimum.

Marquardt (using an idea of Levenberg) noted connection between least squares and steepest descent.

Multiply diagonal elements of least squares matrix (D^TWD) by ($1+\lambda$)

where λ is small for least squares or λ large for steepest descent gives a route between the two extremes.

Schematic Least Square & Steepest Descent, on a 2 parameter χ^2surface



Least squares follows A (well behaved, else might work using partial shifts) or B (blows up), steepest descent follows C or D (local minimum), Marquardt steers between B & D or A& C but might fall into the local minimum.

Marquardt recipe

Guaranteed to find a better fit, but not necessarily the best!

(i) start with a modest $\lambda \sim 1$,

(ii) compute D (and save it) and χ^2

(iii) calculate parameter shifts with diagonal elements of least squares matrix, (D^TWD), multiplied by ($1+\lambda$)

(iv) compute new parameters and their χ^2

(v) if fit has converged, or too many iterations, stop !

(vi) if fit improves, keep new parameters, divide λ by 10 and return to (ii)

(vii) if fit worsens, multiply λ by 10, return to (iii) (no new computation of D needed, so is efficient)

NOTE - to obtain the proper error estimates σ_j on parameters a_j set $\lambda=0$ for a final calculation.

Multiple data sets

Simultaneous fits to multiple data can greatly improve a model.

•Different sample-detector distances.
•Different "contrasts" - or neutron plus X-ray data.

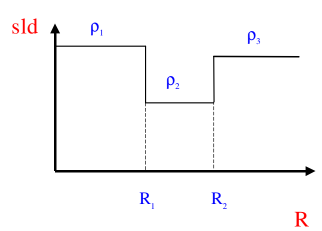
BUT – the samples must be identical in size, shape, concentration etc, OR the models adjusted e.g. for possible changes on deuteration, uncertainties in concentration or neutron sld's.

Semi-automatic fitting of many data sets where structure changes e.g. kinetic or stop-flow measurements is an area to be developed. (Needs to record the changing parameters, fits and make plots …)

Convenient programming ?

Beware – in program I(Q), especially "scale factor" may look different to the equation we put in the paper, but will (of course) be equivalent.

Simple example, core plus shells spheres:



$$I(Q) = N\left\{(\rho_1 - \rho_2)V_1 F(Q,R_1) + (\rho_2 - \rho_3)V_2 F(Q,R_2) + ...\right\}^2$$

$$I(Q) = (NV_1)\left[\frac{1}{V_1}\left\{(\rho_1 - \rho_2)V_1 F(Q,R_1) + (\rho_2 - \rho_3)V_2 F(Q,R_2) + ...\right\}^2\right]$$

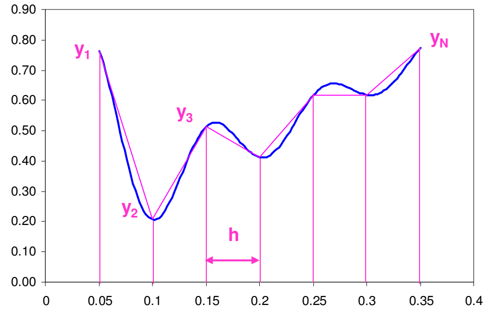
$$F(Q,r) = \frac{3(\sin(Qr) - Qr\cos(Qr))}{(Qr)^3}$$

Core volume fraction, may be known, certainly should not change rapidly, so is less correlated to other parameters.

2(b) Numerical Integration - 1

SANS intensities (and their derivatives) frequently require numerical integration, e.g. to sum over particle size N(r), or to integrate P(Q) for rods (discs) or ellipsoids.
"Trapezium rule" adds up (almost) N equally spaced points *h* apart.

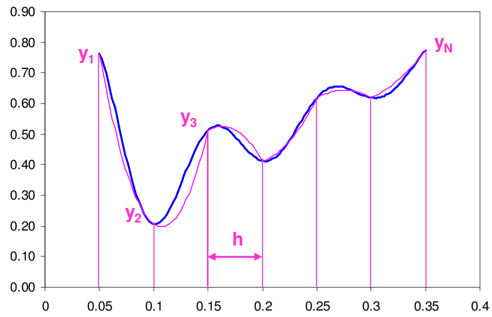
$$\int_{x_1}^{x_N} y(x)dx \approx h\left[\tfrac{1}{2}y_1 + y_2 + y_3 + \ldots + y_{N-1} + \tfrac{1}{2}y_N\right]$$



Numerical Integration - 2

"Simpson's rule" is better, has different "weights", equivalent to fitting a quadratic equation through adjacent groups of three points:

$$\int_{x_1}^{x_N} y(x)dx \approx \frac{h}{3}\left[y_1 + 4y_2 + 2y_3 + 4y_4 \;\ldots\; + 2y_{N-2} + 4y_{N-1} + y_N\right]$$



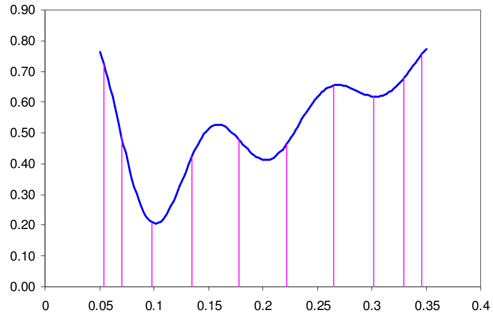
$$F(Q) = 2Q + \frac{\sin^2(QL)}{QL}$$

Where L= 30, unlikely to actually need this integral for SANS, but illustrates the point !

Numerical Integration - 3

In "Quadrature methods" the points (abscissae) are not equally spaced, and weights for each y(x) specially calculated. E.g. a "10 point Gauss-Legendre quadrature" integrates a function as if it were fitted by a 10th order polynomial. It would do:

$$\int_{-1}^{+1} y(x)dx \approx 0.06667\big(y(-0.9739) + y(+0.9739)\big) + 0.14945\big(y(-0.8651) + y(+0.8651)\big) +$$
$$+ 0.21909\big(y(-0.6794) + y(+0.6794)\big) + 0.26923\big(y(-0.4334) + y(+0.4334)\big) +$$
$$+ 0.29552\big(y(-0.1489) + y(+0.1489)\big)$$



Though the integration is from -1 to +1, the abscissae can be rescaled to suit a given range, as in the plot above, note end points are *not* at ends of range.

Numerical Integration - 4

IF function y(x) is *well* approximated by a polynomial then a quadrature integral is more accurate than using Simpson's rule with many more points, and the program is much more efficient.

Some functions (e.g. long thin rods in shear) have strong oscillations for which Gaussian quadrature does not work well and Simpson's Rule is still best.

Always test carefully an unfamiliar calculation!

Numerical Integration - 5

Iterative or Adaptive Integrations

"Trapezium rule" is easily iterated by squeezing in extra points and/or dividing the integration range into sections. Can also be done for "Simpson's rule" with a little more work. But increasing the number of points in a Gaussian quadrature results in a completely new set of abscissae and weights.

The Gauss-Kronrod method is an "adaptive" integration scheme which expands the Gauss-Legendre polynomials in an optimal way, re-using results of the previous iteration. It uses a sequence of N points such as N=10, 21, 43, 87.

By sub-dividing the integration range more time is spent to improve the parts with worst integration errors.

Iterative integrations usually need to be given an absolute or relative error for the integration result so they know when to stop !

See "Numerical Recipes" Chapter 4 for much more information about errors and singularities and other types of quadrature.

3. Some fitting programs for SANS

all are linked from www.small-angle.ac.uk

FISH (R.K.Heenan)
http://www.isis.stfc.ac.uk/instruments/loq/data-analysis/loq-data-analysis2520.html
~ 70 models which may be combined in many ways. Includes multiple data sets and Q resolution smearing. Wide range of constraints. Can rapidly try several different models without leaving the program.
Old version was "Difficult" to learn ? New interface by Jonathan Rawle, is easy!
Detailed manual with list of models and equations used.

SASVIEW (DANSE collaboration, now involving ISIS)
http://sourceforge.net/projects/sasview/ eventually to "take over from FISH". It has a lot of models (many from older NIST, IGOR interface). Can use Python to combine existing models or to write your own. Similar to Mantidplot, using python to drive C++.

SASFit (Joachim Kohlbrecher) and Scatter (Stephan Forster) have some further very clever models.

For a quick estimate of I(Q) see NIST "SANS simulator" at
http://www.ncnr.nist.gov/resources/simulator.html

EMBL BioSAXS group (D.Svergun)

http://www.embl-hamburg.de/biosaxs/software.html
well known, suite of programs ATSAS 2.1 (CRYSON, DAMMIN, SASHA etc) for scattering from usually, but not entirely, monodisperse bio-molecules.

Less well known, program MIXTURE deals with mixtures of polydisperse spheres, cylinders, ellipsoids or dumb-bells, with hard or sticky sphere S(Q) using a number of different optimisation methods.

ccpSAS new USA/UK project involving ISIS, NIST, UCL, etc to provide force field directed fits for biomolecules (and later polymers & surfactants). Parts of this wil use SASSIE code from http://sourceforge.net/projects/sassie/

OTHERS – Individual researchers codes which they may let others use. May be very specialised, be careful to use them appropriately, and may lack documentation.
Some examples:
Prof. J.S.Pedersen (Aarhus) specific codes for polymer decorated particles.

Prof. T.Cosgrove (Bristol) polymer density profiles at interfaces.

HMI (Berlin) polarised neutron group have codes for magnetic systems.

Where to start ?

For a genuine "unknown" try Guinier & log-log plots for clues.

Powers of Q: Q^{-1} – long thin rods,
Q^{-2} can be discs, gels, intermediate parts of polymer coil etc,
Q^{-3} surface or volume fractals, or just a transition to …

Q^{-4} Porod, smooth interface,

$Q^{-4.2}$ to Q^{-6} diffuse but not rough interface (rare).

Beware interparticle S(Q) can be strong for swollen and/or charged systems.

Model fits to cylinder/disc, polydisperse spheres, Gaussian coil, can help.

Core/shell sphere, cylinders, ellipsoids may be the next step.

LET THE FLAT BACKGROUND (incoherent & inelastic) ADJUST

Check the absolute SANS intensities!

HOW MUCH INFORMATION DO I NEED ?

Simple size and shape or detailed analysis ….

4. IFT – Inverse Fourier Transform

Not really "fitting". Due to finite Q range of SANS data, first fit spline or other "regularisation" functions to I(Q) that can then be Fourier transformed to give a real space distribtion, weighted be neutron scattering length densities.

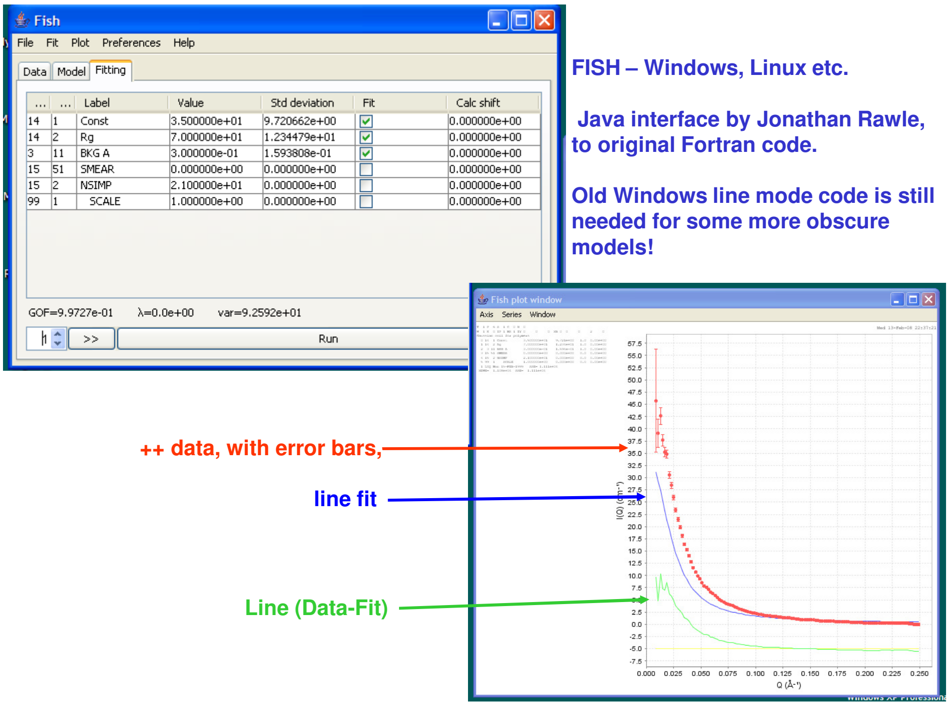
Can be useful for "unknowns". O.Glatter promotes IFT as a "model free" approach. For even modestly complex cases interpretation of g(r) by inspection is difficult and methods used to separate S(Q) or to assume particle types etc. amount to "modelling" ?

Some IFT programs

SANSVIEW – has IFT as well as normal fitting.

GNOM – see EMBL, BioSAXS group.

GIFT – direct from Prof. Otto Glatter, otto.glatter@kfunigraz.ac.at (pay fee to Univ. Graz)



FISH – Windows, Linux etc.

Java interface by Jonathan Rawle, to original Fortran code.

Old Windows line mode code is still needed for some more obscure models!

++ data, with error bars,

line fit

Line (Data-Fit)

FISH - LSINP.DAT Least Squares Input, model file

The model file is a special feature of FISH. Different models may be rapidly tried and new combinations of models made without doing any programming.

e.g. Gaussian coil model 14 LTYP=1, flat background model 3 LTYP=11, Q smearing model 15, LTYP=51

```
MODEL LTYP                   value               esd         pshift     shift

 T   1 P   6 S   1 C   0 N   0
 W   1 K   0 IP  1 MS  1 IY  1 Q   3 R   5 XB  0    1    0    2    0    0    0    5    0
  Gaussian coil for polymers
  1  14   1  Const          35.00000        0.000E+00    1.0   0.00E+00
  2  14   2  Rg             7.000000E+01    0.000E+00    1.0   0.00E+00
  3   3  11  BKG A          0.3             0.0          1.0   0.0
 15  15  51  SMEAR          0.000000E+00    0.000E+00    0.0   0.00E+00
 16  15   2  NSIMP          2.100000E+01    0.000E+00    0.0   0.00E+00
  6  99   1      SCALE      1.000000E+00    0.000E+00    0.0   0.00E+00
   1   2  CAL            CALC  2  BKG  0  POL  4   SSE=  0.000E+00
```

4=1.0 would include approx Q resolution smearing

These two columns of numbers tell FISH what to calculate, the text label is irrelevant, as are parameter numbers in first column. Type in 2=70 to change value of param 2

FISH

K2=1 Marquardt fit, K2= 0 Least squares

```
T   1 P 15 5   1 C   1 N   5
W   1 K   1 IP  0 MS 1 IY 1    -6    -6     0     0  121   112     6
SCHULTZ POLYDISP TWO SHELL * HARD SPHERE
  1 10   1 RHO1-RHO2      2.500000E+00     0.000E+00    0.0   0.00E+00
  2 10   2 R1             6.492493E+01     0.000E+00   -2.0   0.00E+00
  3 10   1 RHO2-RHO3     -6.701326E-01     1.022E-02    1.0  -7.42E-04
  4 10   2 R2=R1+DR       1.173167E+02     0.000E+00   -1.0   0.00E+00
  5  2   2        DR      5.239183E+01     4.426E-01    1.0  -3.10E-02
  6  6  11 SCHZ.SCALE     3.793647E-07     4.259E-09    1.0   2.45E-10
  7  6  12      RBAR      6.492493E+01     2.023E-01    1.0   2.59E-02
  8  6  13        R-SHIFT 0.000000E+00     0.000E+00    0.0   0.00E+00
  9  6  14 SIG/(RB-R0)    2.544773E-01     2.536E-03    1.0  -6.71E-05
 10 22   1 HS S(Q) VOL    0.000000E+00     0.000E+00    0.0   0.00E+00
 11 22   2 SPH RADIUS     3.500000E+01     0.000E+00    0.0   0.00E+00
 12  3  11 BKG A          1.477111E-02     9.646E-04    1.0  -1.56E-05
 15 15  51 SMEAR          0.000000E+00     0.000E+00    0.0   0.00E+00
 16 15   2 NSIMP          2.100000E+01     0.000E+00    0.0   0.00E+00
 17 99   1 SCALE          1.000000E+00     0.000E+00    0.0   0.00E+00
   1   1 ad0335        CALC  2 BKG  0 POL  3   SSE= 3.527E+01
   1     4     2     5     0
   1.00000    1.00000    0.00000    0.00000
1.000E-02 3.794E-07 2.545E-01 4.000E+00 2.000E+03
```

Polydisperse

Constraint

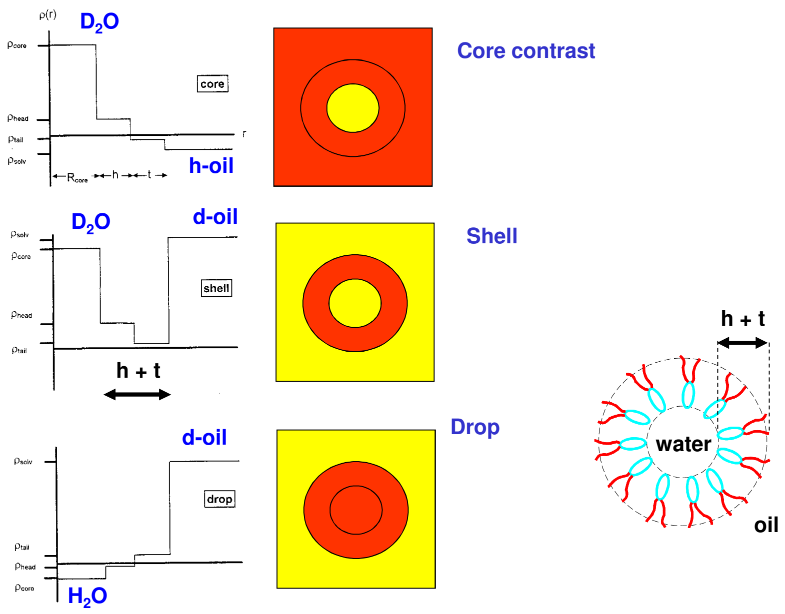
Constraint, type=1

N1 = Marquardt λ

Check N4 = R search & store step & N5 = Rmax (< 512 x N4)

[R_{bar} is at param 7, constraints routine should reset param 2 using R_{bar} at end of each iteration - used to leave at final value of integration]

Simultaneous 3 Contrasts



FISH – polydisp core, 2 shell, 3 contrasts - part 1

```
T  1 P 44 S  3 C 15 N  5
W  1 K  1 IP 0 MS 1 IY 1    -6    -6 XB 0     0  363     9     5
 General microemulsion  3  DATA SETS     core, shell, drop
 1 88   1 SET 1 core    0.000000E+00    0.000E+00   0.0  0.00E+00
 2 10   1 A             6.000000E+00    0.000E+00  -1.0  0.00E+00
 3 10   2 R1            8.397082E+01    0.000E+00  -2.0  0.00E+00
 4 10   1 B             4.000000E-01    0.000E+00  -1.0  0.00E+00
 5 10   2 R1+R3         8.697082E+01    0.000E+00  -1.0  0.00E+00
 6 10   1 G (zero)      0.000000E+00    0.000E+00  -1.0  0.00E+00
 7 10   2 R1+R3+R4      9.597082E+01    0.000E+00  -1.0  0.00E+00
 8  3 11 BKG SET 1      1.462204E-01    4.116E-04   1.0  0.00E+00
 9 88   2 SET 2 shell   0.000000E+00    0.000E+00   0.0  0.00E+00
10 10   1 A             6.000000E+00    0.000E+00  -1.0  0.00E+00
11 10   2 R1            8.397082E+01    0.000E+00  -2.0  0.00E+00
12 10   1 F             4.000000E-01    0.000E+00  -1.0  0.00E+00
13 10   2 R1+R3         8.697082E+01    0.000E+00  -1.0  0.00E+00
14 10   1 E            -6.700000E+00    0.000E+00  -1.0  0.00E+00
15 10   2 R1+R3+R4      9.597082E+01    0.000E+00  -1.0  0.00E+00
16  3 11 BKG SET 2      1.121031E-03    1.733E-04   1.0  0.00E+00
17 88   3 SET 3 drop    0.000000E+00    0.000E+00   0.0  0.00E+00
18 10   1 C             0.000000E+00    0.000E+00  -1.0  0.00E+00
19 10   2 R1            8.397082E+01    0.000E+00  -2.0  0.00E+00
20 10   1 D             4.000000E-01    0.000E+00  -1.0  0.00E+00
21 10   2 R1+R3         8.697082E+01    0.000E+00  -1.0  0.00E+00
22 10   1 E            -6.700000E+00    0.000E+00  -1.0  0.00E+00
23 10   2 R1+R3+R4      9.597082E+01    0.000E+00  -1.0  0.00E+00
24  3 11 BKG SET 3      6.108395E-02    1.974E-04   1.0  0.00E+00
25 88   0 ALL SETS      0.000000E+00    0.000E+00   0.0  0.00E+00
26  2   1 x CORE/HEAD   0.000000E+00    0.000E+00   0.0  0.00E+00
27  2   1 y SOLV/TAIL   0.000000E+00    0.000E+00   0.0  0.00E+00
contd
```

88 n "switch" data sets

FISH – polydisp core, 2 shell, 3 contrasts - part 2

```
24  3 11 BKG SET 3      6.108395E-02    1.974E-04   1.0  0.00E+00
25 88   0 ALL SETS      0.000000E+00    0.000E+00   0.0  0.00E+00
26  2   1 x CORE/HEAD   0.000000E+00    0.000E+00   0.0  0.00E+00
27  2   1 y SOLV/TAIL   0.000000E+00    0.000E+00   0.0  0.00E+00
28  2   1 R3 HEAD       3.000000E+00    0.000E+00   0.0  0.00E+00
29  2   1 R4 TAIL       9.000000E+00    0.000E+00   0.0  0.00E+00
30  2   1 core (H)      0.000000E+00    0.000E+00   0.0  0.00E+00
31  2   1 core (D)      6.000000E+00    0.000E+00   0.0  0.00E+00
32  2   1 head          0.000000E+00    0.000E+00   0.0  0.00E+00
33  2   1 tail         -4.000000E-01    0.000E+00   0.0  0.00E+00
34  2   1 solv (H)     -4.000000E-01    0.000E+00   0.0  0.00E+00
35  2   1 solv (D)      6.300000E+00    0.000E+00   0.0  0.00E+00
36  6 11 SCH SCALE      3.088799E-07    3.484E-11   1.0  0.00E+00
37  6 12       RBAR     1.562683E+01    6.218E-04   1.0  0.00E+00
38  6 13       R-SHIFT  0.000000E+00    0.000E+00   0.0  0.00E+00
39  6 14 SIG/(RB-R0)    2.000000E-01    0.000E+00   0.0  0.00E+00
40 22   1 HS S(Q) VOL   0.000000E+00    0.000E+00   0.0  0.00E+00
41 22   2 SPH RADIUS    6.000000E+01    0.000E+00   0.0  0.00E+00
42 15   1 SMEAR         0.000000E+00    0.000E+00   0.0  0.00E+00
43 15   2 NSIMP         2.100000E+01    0.000E+00   0.0  0.00E+00
44 99   1 SCALE         1.000000E+00    0.000E+00   0.0  0.00E+00
  1  1 c              CALC 4 BKG 0 POL 7  SSE= 1.020E+03
  2  2 s              CALC 5 BKG 0 POL 7  SSE= 9.081E+02
  3  3 d              CALC 6 BKG 0 POL 7  SSE= 1.520E+03
contd
```

10^{-4} ϕ if sld in 10^{10} cm^{-1}

here 0.31% v of core particle

No S(Q) if vol=0.0

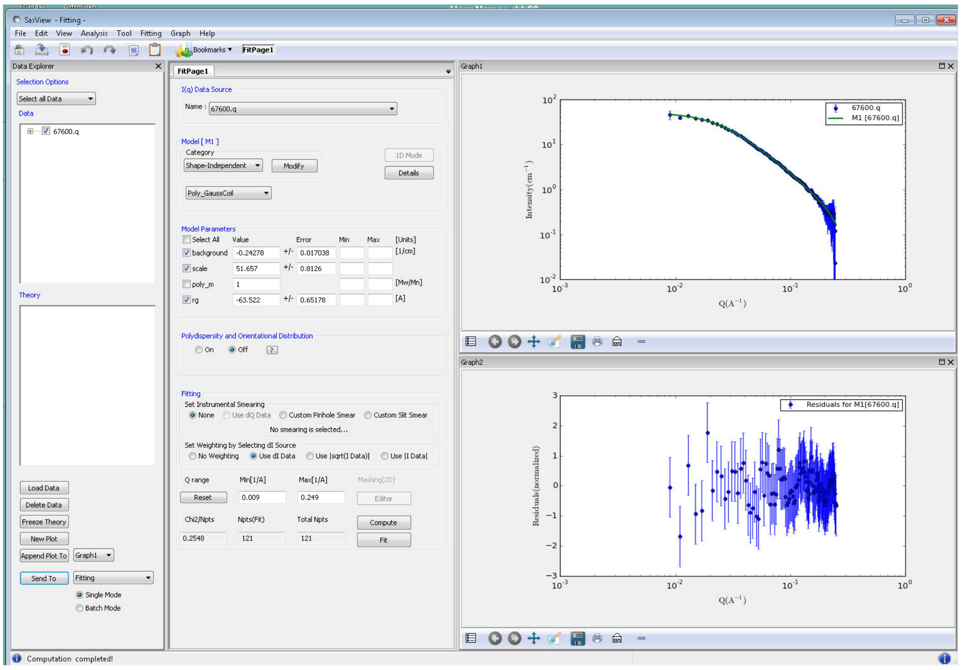
FISH – polydisp core, 2 shell, 3 contrasts - part 3

```
 3  3 d            CALC  6 BKG  0 POL  7   SSE= 1.520E+03
  12    2  26   31   32
    0.00000   0.00000   0.00000   0.00000
   12  10  26   31   32
    0.00000   0.00000   0.00000   0.00000
   12  18  26   30   32
    0.00000   0.00000   0.00000   0.00000
   12  14  27   33   35
    0.00000   0.00000   0.00000   0.00000
   12  22  27   33   35
    0.00000   0.00000   0.00000   0.00000
   12   6  27   33   34
    0.00000   0.00000   0.00000   0.00000
   13   4  32   33   26   31   27   34
    0.00000   0.00000   0.00000   0.00000
   13  20  32   33   26   30   27   35
    0.00000   0.00000   0.00000   0.00000
   13  12  32   33   26   31   27   35
    0.00000   0.00000   0.00000   0.00000
   11   5   3   28   29   29
    1.00000   1.00000   0.00000   0.00000   0.00000
   11   7   3   28   29   29
    1.00000   1.00000   1.00000   0.00000   0.00000
   11  21  19   28   29   29
    1.00000   1.00000   0.00000   0.00000   0.00000
   11  23  19   28   29   29
    1.00000   1.00000   1.00000   0.00000   0.00000
   11  13  11   28   29   29
    1.00000   1.00000   0.00000   0.00000   0.00000
   11  15  11   28   29   29
    1.00000   1.00000   1.00000   0.00000   0.00000
  1.000E+00 3.089E-07 2.000E-01 2.000E+00 1.000E+03
```

Lots of constraints !

N4 & N5 Rstep & Rmax

SASVIEW has a lot of different features and capabilities



Conclusions

- Model fitting is usually the best method.
- Most least squares fitting programs have similar characteristics –

  List of parameters with errors, on/off flag, constraints or links etc.

  Some plots.

  Complex model will require a complex interface !
- A variety of software exists. This is good - can “benchmark” programs.
- Most instrument scientists are glad to help you get started with fitting (especially if they get their name on a publication !)

APPENDIX – for mathematicians LEAST SQUARES FITS

A knowledge of the maths of "least squares fitting" is very useful to understand its limitations and practical difficulties !
Suppose we have data y_i where i = 1 to N at points x_i e.g. SANS or reflection data at N values of Q.
The calculated data $CALC_i$ = function(x_i, a_1, a_2, …a_M) has parameters, which we want to optimise, a_j where j = 1 to M,
The "best fit" is the one with the lowest value of a "merit function"

$$\chi^2 = \sum_{i=1}^{N}\left(\frac{y_i - CALC_i}{\sigma_i}\right)^2$$

We minimise chi squared because statistical theories say: If errors (uncertainites, standard deviations) σ_i are *independent* and have a *normal (Gaussian) distribution* then the minimum χ^2 is "the most likely" solution, and $\chi^2/(N-M) \sim 1$.
(Normal distribution has ±2σ 68% of the time, ±3σ 95% of the time.)

BUT "Poisson distribution" for neutron counts has broad tail for small counts. So our first limitation is that the Merit function is not quite correct! "Outliers" problem, set W_i to zero ? Nor are the data always entirely independent.

NOTE "Maximum entropy" adds another part to the merit function to say that the fitted function or model must compare well to our "prior knowledge" – can be useful for say a particle size distribution.

To solve Least Squares: At minimum of χ^2 , derivative with respect to parameters a_j = zero, weights $W_i = 1/\sigma_i^2$

$$\frac{\partial(\chi^2)}{\partial a_j} = 0 \text{ gives } M \text{ equations } \sum_i W_i (y_i - CALC_i)\frac{\partial CALC_i}{\partial a_j} = 0$$

Simplest case is "linear" in the unknown parameters a_k (this does NOT mean the terms in the model have to be simple linear functions).

$$CALC_i = \sum_{k=1}^{M} a_k Fun_k(x_i) = \sum_{k=1}^{M} a_k D_{ik}$$

"basis functions" are actually derivatives D_{ij} of model

$$D_{ij} = \frac{\partial CALC_i}{\partial a_j} = Fun_j(x_i)$$

Linear case (e.g. straight line or polynomial fit) solves *exactly* for the best parameters a_j, but we will write equations for more general "non-linear" case:

Present a_j^{now} give $CALC_i^{now}$ & differences $E_i = (y_i - CALC_i^{now})$

need shifts $a_j = a_j^{now} + \Delta a_j$ which gives best (or smaller) χ^2.

$$CALC_i = CALC_i^{now} + \sum_k \Delta a_k D_{ik}$$

give a set of M simultaneous equations

$$\sum_i W_i \left(E_i - \sum_k \Delta a_k D_{ik} \right) D_{ij} = 0$$

In matrix form:

$$\Delta a(D^T W D) - D^T W E = 0$$

Rearranging gives:

$$\Delta a = (D^T W D)^{-1} (D^T W E)$$

To find parameter shifts, Δa, column vector of M rows, we need
$D^T WD$ the "least squares matrix", where
derivative matrix D (can be J for Jacobian) has N rows x M columns,
weights W_{ii} are a diagonal N x N,
differences E = (obs - calc) is a column of N rows.
Superscripts "T" mean transpose (i.e. $D_{ij}^T = D_{ji}$)
"-1" means the matrix inverse – found by computer subroutine.

e.g two parameter and 3 data points, solve for shifts *$a = \Delta a_1$* & *$b = \Delta a_2$* :

$$a(W_1 D_{1a}^2 + W_2 D_{2a}^2 + W_3 D_{3a}^2) + b(W_1 D_{1a} D_{1b} + W_2 D_{2a} D_{2b} + W_3 D_{3a} D_{3b}) - (W_1 D_{1a} E_1 + W_2 D_{2a} E_2 + W_3 D_{3a} E_3) = 0$$
$$a(W_1 D_{1a} D_{1b} + W_2 D_{2a} D_{2b} + W_3 D_{3a} D_{3b}) + b(W_1 D_{1b}^2 + W_2 D_{2b}^2 + W_3 D_{3b}^2) - (W_1 D_{1b} E_1 + W_2 D_{2b} E_2 + W_3 D_{3b} E_3) = 0$$

ERRORS & CORRELATION PARAMETERS

M x M matrix $C = (D^TWD)^{-1}$ is variance-covariance matrix,

Its diagonal elements are variance, $C_{jj} = \sigma_j^2$ = square of the *statistical* standard deviation for each parameter a_j - *assuming* the conditions above on σ_i !

Systematic errors, e.g. imperfect data treatment, or Q resolution must be considered separately ! Thus parameter "errors" for SANS fits are often too small (test effect on fit of changing parameters values).

Off-diagonal C_{ij} give *correlation coefficients*, - identify poor parametrisation of model.

"NON-LINEAR" LEAST SQUARES CASE

Solution for Δa needs to be iterated. It is a wonder that it works at all !

Practicalities of Least Squares

(a) Need routines for CALC, derivatives D for given parameters & Q

(b) a routine to invert symmetric matrix (i.e. to solve a set of equations).

D_{ij} may be calculated numerically (less reliably) by shifing parameter up or down:

$$D_{ij} \approx \frac{1}{\delta}\left(CALC(x_i, a_1, a_2, ..(a_j + \delta), ...a_n) - CALC(x_i, a_1, a_2, ..a_j, ...a_n)\right)$$

When χ^2 ceases to improve the fit has "converged".

Not all fits are "well behaved" !

Adjust scale parameters first, then release others, include a flat background for SANS, but check its value is reasonable !

Oscillatory Δa_j may be damped down by applying fraction of shifts.

Worse behaviour ? - trial & error, try a steepest descent fit (guaranteed to improve), or apply constraints using prior knowledge.

Constraints

Help locate physically meaningful fits from amongst whole families of possible numerical solutions. Reduce number of unknown parameters.
e.g. fix shell thickness to give known shell to core molar volume ratio;
e.g. keep core radius below or at fully extended surfactant tail lengths;
e.g. try fix absolute scale factor to known volume fraction.

Least squares program has to adjust the derivatives to allow for constraints:
If we remove parameter $a_j =$ *function*(a_k) we know $\frac{\partial CALC_i}{\partial a_j}$
But least squares needs

$$\frac{\partial CALC_i}{\partial a_k} = \frac{\partial CALC_i}{\partial a_k} + \frac{\partial CALC_i}{\partial a_j}\frac{\partial a_j}{\partial a_k} = \frac{\partial CALC_i}{\partial a_k} + \frac{\partial CALC_i}{\partial a_j}\frac{\partial f(a_k)}{\partial a_k}$$

Which has to be programmed from known *function*(a_k)