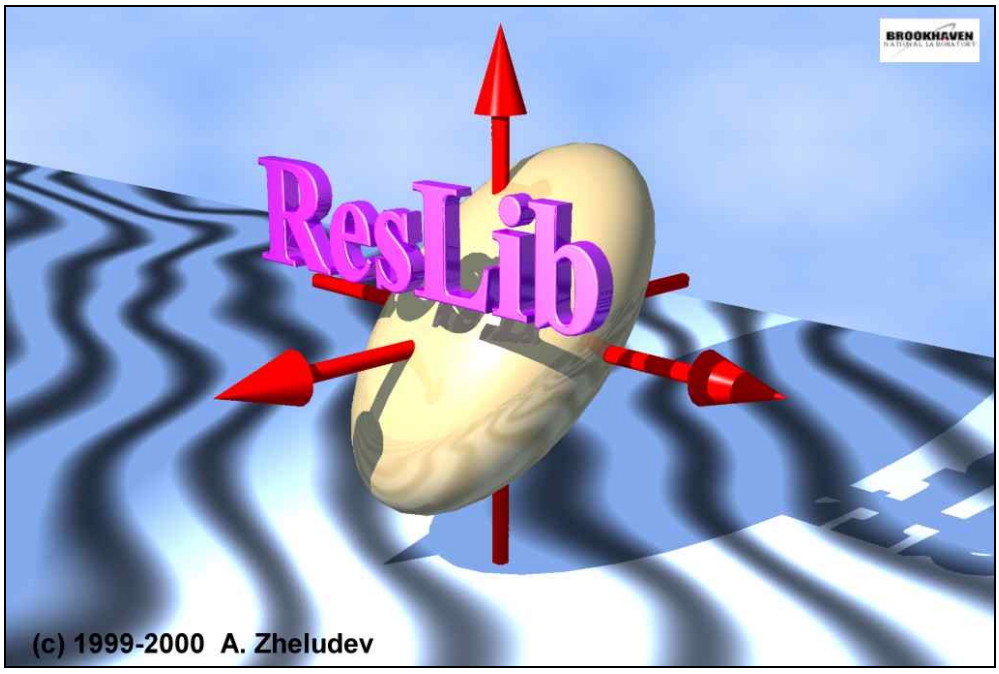
ResLib v.2.1g

3-axis resolution routines for MatLab

Andrey Zheludev
Brookhaven National Laboratory



Introduction

Perhaps the most important aspect of analyzing inelastic neutron scattering data measured using a 3-axis spectrometer is properly taking into account the experimental resolution. Even though one is fairly confident that the Gaussian approximation to the resolution function is accurate, the problem is still quite non-trivial for the following reasons: 1) The parameter space is 4-dimensional (3 dimensions for the wave vector transfer and one for energy transfer); 2) In many cases in the course of a single inelastic scan the resolution function changes significantly from one point to the next; and 3) The width of the experimental resolution is often larger than the size of characteristic features of the dynamic structure factor. In general, to properly analyze experimental scans using a parameterized model cross section S(Q,ω,p), one needs to 1) be able to calculate the resolution function at each data point, given the spectrometer configuration and sample parameters, 2) numerically convolute the theoretical cross section with this resolution function; and 3) fit the convoluted cross section to the data. The ResLib library is designed to accomplish these tasks.

Resolution calculation

ResLib is based on the Cooper-Nathans Gaussian approximation [1-3] for the resolution function, given by:

$$\left.\frac{d\sigma}{d\Omega\, dE'}\right|_{\substack{\vec{k}-\vec{k}'=\vec{q}_0 \\ E-E'=\hbar\omega_0}} \approx R_0(\vec{Q}_0)\iiiint d^4\vec{Q}\, S(\vec{Q})\exp\left[-\frac{1}{2}(Q^i-Q_0^i)(Q^j-Q_0^j)M_{ij}(\vec{Q}_0)\right] \qquad (1)$$

$$\vec{Q}=(q_x,q_y,\hbar\omega,q_z)$$

q_x and q_y are components of the scattering vector in the (horizontal) scattering plane, and q_z is the component of the scattering vector along the vertical axis. Compared to the original Cooper-Nathans formulas, we include the k_f/k_i factor into R_0. R_0 is normalized not to the incident flux on the first (in-pile) collimator, but to the flux at monitor position (after second collimator). The resolution prefactor also includes the effect of finite sample mosaic [2].

As an option, the prefactor R_0 can include the effect of analyzer reflectivity. The effective reflectivity in the Cooper-Nathans (Gaussian) model differs form the real reflectivity of the analyzer crystal. To compensate for this effect, we correct R_0 so that integral reflectivity of the effective Gaussian matches the integral reflectivity of an ideal crystal. The reflectivity curve of an ideal crystal is taken from Ref. [4]. At E < 5.5meV, where no parasitic (hkl) reflections reducing the reflected PG (002) intensity are present [5], this approximation is valid for PG to within a few percent [6]. At higher final neutron energies the calculated correction can not be considered valid, as the Shapiro-Chesser effects become important [5].

[1] M. J. Cooper & R. Nathans, Acta Cryst. 23, 357, (1967).
[2] S. A. Werner & R. Pynn, J. Appl. Phys. 42, 4736, (1971).
[3] N. J. Chesser & J. D. Axe, Acta Cryst. A29, 160, (1972).
[4] G. E. Bacon & R. D. Lowde, Acta Cryst. (1948).
[5] S. M. Shapiro, N. J. Chesser, Nucl.Instr.& Meth. (1972).
[6] T. Riste & K. Otnes, Nucl. Instr.& Meth. (1969); B. Dorner & A. Kollmar, J.Appl. Cryst. (1974).

ResMat.m

This function calculates the Cooper-Nathans resolution matrix M and the prefactor R_0 for a coordinate system defined by the scattering vector **q**: the x axis is chosen along $\mathbf{k}_i$-$\mathbf{k}_f$ = **q**. The y axis is perpendicular to **q**, in the scattering plane. y is clockwise from x, if the positive direction of motor 4 is clockwise, and counter-clockwise from x otherwise.

```
function [R0,M]=ResMat(Q,W,EXP)
```

Input arguments:

- **Q** is the wave vector transfer in $Å^{-1}$, and
- **W** is the energy transfer in meV.
- **EXP** is a structure that contains all the information on the experimental setup:

- **EXP.mono** is a structure that describes the monochromator:
  EXP.mono.tau is the monochromator reciprocal lattice vector in $Å^{-1}$.
  EXP.mono.mosaic is the monochromator mosaic in minutes of arc.
- **EXP.ana** is a structure that describes the analyzer:
  EXP.ana.tau is the analyzer reciprocal lattice vector in $Å^{-1}$.
  EXP.ana.mosaic is the analyzer mosaic in minutes of arc.
  EXP.ana.thickness is the analyzer thickness in cm for ideal-crystal reflectivity corrections.
  EXP.ana.Q is the kinematic reflectivity coefficient for this correction. It is given by $Q = 4\frac{|F|^2}{V_0^2}\frac{(2\pi)^3}{\tau^3}$, where V_0 is the unit cell volume for the analyzer crystal, F is the structure factor of the analyzer reflection, and τ is the analyzer reciprocal lattice vector. For PG(002) Q~0.1287. Set **EXP.ana.Q** to a negative value if you don't want the correction done.
- **EXP.sample** is a structure that describes the sample:
  EXP.sample.mosaic is the sample mosaic in the scattering plane in minutes of arc.
  EXP.sample.vmosaic is the vertical sample mosaic in minutes of arc.
- **EXP.hcol(1:4)** are the horizontal collimations in minutes of arc starting from the in-pile collimator.
- **EXP.vcol(1:4)** are the vertical collimations in minutes of arc starting from the in-pile collimator.
- **EXP.efixed** is the fixed incident or final neutron energy, in meV.
- **EXP.infin** is +1 if incident energy is fixed or -1 in a fixed-final setup.
- **EXP.horifoc** is <=0 for a standard 3-axis setup with a flat analyzer. Set this flag to 1 when using a horizontally-focusing analyzer. In this case **Exp.hcol(3)** should be replaced by the angular size of the analyzer, as seen from the sample position.
- **EXP.dir1=1**, if monochromator scattering direction is opposite to that of sample, and -1 otherwise.
- **EXP.dir2=1**, if analyzer scattering direction is opposite to that of sample, and -1 otherwise.

Output arguments:

- **R0** is the Cooper-Nathans prefactor and **M** is a 4x4 resolution matrix.

Vectorization:

The function can be used to simultaneously calculate resolution matrixes for several scattering vectors, energies, and experimental setups. The input arguments **Q**, **W** and **EXP** can be vectors of the same length N, in which case **R0** will be returned as a vector of length N and **M** will be a 4x4xN array. If some of the input arguments are scalars, they will be automatically replicated into vectors of appropriate length.

Dependencies:

ResMat.m is a stand-alone function that does not rely on any other components in ResLib.

ResMatS.m

This function is very similar to **ResMat.m**, and calculates the Cooper-Nathans resolution matrix M and the prefactor R_0 for a coordinate system defined by the sample axes. The x axis is chosen along the first user-defined orienting vector in the scattering plane. The y axis is in the scattering plane, perpendicular to x, in the direction of the second user-defined orienting vector. The z axis completes a right-handed coordinate system.

```
function [R0,M]=ResMatS(H,K,W,L,EXP)
```

Input arguments:

- **H**, **K** and **L** specify the wave vector transfer in the scattering plane and are given in Reciprocal-lattice units.
- **W** is the energy transfer in meV.
- **EXP** is a structure that contains all the information on the experimental setup. In addition to the fields described above, it contains the following:
  - **EXP.sample.a** is the a lattice constant in Å.
  - **EXP.sample.b** is the b lattice constant in Å.
  - **EXP.sample.c** is the c lattice constant in Å.
  - **EXP.sample.alpha** is the a angle in degrees of arc.
  - **EXP.sample.beta** is the b angle in degrees of arc.
  - **EXP.sample.gamma** is the g angle in degrees of arc.
  - **EXP.orient1(1:3)** are Miller indexes of the first orienting reciprocal-lattice vector in the scattering plane.
  - **EXP.orient2(1:3)** are Miller indexes of the second orienting reciprocal-lattice vector in the scattering plane.

Output arguments and vectorization

Similar to those for **ResMat.m**.

Dependencies:

ResMatS.m requires **ResMat.m**, **StandardSystem.m** , **star.m**, **scalar.m**, **modvec.m** to function properly.

MakeExp.m

This function is an example of creating the **EXP** structure for use with **ResMat.m** and **ResMatS.m**. The user is encouraged to make copies of this m-file and edit them to generate particular experimental setups. The function takes no arguments and returns a **ResMatS**-compatible structure **EXP**.

Crystallography calculations utility functions

ResLib contains several routines for crystallographic calculations "free of charge". These are **star.m**, **scalar.m** and **modvec.m**. They are particularly useful in user-supplied cross section functions used with ConvRes.m ans ConvResSMA.m (see below). All this stuff is explained in the International Tables Of Crystallography, vol. C.

star.m

Given direct-space lattice parameters, this function calculates reciprocal-space lattice parameters (or vice-versa).

```
function [V,Vstar,rlattice]=star(lattice)
```

Input arguments:

- **lattice** is a structure that contains direct-space lattice parameters (note the difference with the structure **EXP.sample** described above!):
  - **lattice.a** is the a lattice constant
  - **lattice.b** is the b lattice constant
  - **lattice.c** is the c lattice constant

- **lattice.alpha** is the a angle in radians
- **lattice.beata** is the b angle in radians
- **lattice.gamma** is the g angle in radians

Output arguments:

V and **Vstar** are the direct and reciprocal unit cell volumes. **rlattice** has the same structure as lattice, but contains reciprocated cell parameters.

Vectorization:

The function can be used to simultaneously reciprocate several sets of lattice parameters. For N calculations each field of **lattice** should be a vector of length N or a scalar (if identical values are assumed for a particular cell constant). **V** and **Vstar** and the fields of **rlattice** will then be vectors of length N.

Dependencies:

star.m is a stand-alone function that does not rely on any other components in ResLib.

scalar.m

Calculate the scalar product of two vectors defined by their fractional cell coordinates. Alternatively, calculates the scalar product of two reciprocal vectors defined by their Miller indexes.

```
function s=scalar(x1,y1,z1,x2,y2,z2,lattice)
```

Input arguments:

- **lattice** is as in **star.m**. It contnains all the lattice parametrs for this calculation. Alternatively, it contains reciprocal-lattice parameters.
- **x1, y1, z1** are fractional cell coordinates of the first vector. Alternatively, they define the Miller indexes of the first reciprocal-lattice vector.
- **x2, y2, z2** are fractional cell coordinates of the second vector. Alternatively, they define the Miller indexes of the second reciprocal-lattice vector.

Output arguments:

The returned value is the scalar product of the two vectors.

Vectorization:

The function can be used to simultaneously calculate several scalar products. For N calculations, **x1…z2** are vectors of length N or scalars. Correspondingly, each field of **lattice** should be a vector of length N or a scalar. The returned value will be a vector of length N as well.

Dependencies:

scalar.m is a stand-alone function that does not rely on any other components in ResLib.

modvec.m

Calculate the length of a vector defined by its fractional cell coordinates. Alternatively, calculates the length of reciprocal vector defined by its Miller indexes.

```
function m=modvec(x,y,z,lattice)
```

Input arguments:

- **`lattice`** is as in **`star.m`**. It contnains all the lattice parametrs for this calculation. Alternatively, it contains reciprocal-lattice parameters.
- **`x, y, z`** are fractional cell coordinates of a vector. Alternatively, they define the Miller indexes of a reciprocal-lattice vector.

Output arguments:

The returned value is the vector's length.

Vectorization:

The function can be used to simultaneously calculate several lengths. For N calculations, **`x,y,z`** are vectors of length N or scalars. Correspondingly, each field of **`lattice`** should be a vector of length N or a scalar. The returned value will be a vector of length N as well.

Dependencies:

`modvec.m` relies on **`scalar.m`**.

Convolution with resolution function: general 4-D convolution

A neat trick suggested by Igor Zaliznyak is to replace the indefinite 4-D integral in Eq. 1 by an integral over a 4D hypercube:

$$\left.\frac{d\sigma}{d\Omega\, dE'}\right|_{\substack{\vec{k}-\vec{k}'=\vec{q}_0 \\ E-E'=\hbar\omega_0}} \approx R_0(\vec{Q}_0)\frac{1}{\sqrt{|M|}}\int_{-\pi/2}^{-\pi/2}d\varphi_1\int_{-\pi/2}^{-\pi/2}d\varphi_2\int_{-\pi/2}^{-\pi/2}d\varphi_3\int_{-\pi/2}^{-\pi/2}d\varphi_4 S(\vec{Q})\frac{\exp\left[-\frac{1}{2}tg^2\varphi_1-\frac{1}{2}tg^2\varphi_2-\frac{1}{2}tg^2\varphi_3-\frac{1}{2}tg^2\varphi_4\right]}{\cos^2\varphi_1\cos^2\varphi_2\cos^2\varphi_3\cos^2\varphi_4} \qquad (2)$$

where the new variables are defined as:

$$\begin{aligned}
q_x - q_{0,x} &= \frac{1}{\sqrt{\widetilde{M}_{11}}}tg\varphi_1 \\
q_y - q_{0,y} &= \frac{1}{\sqrt{\widetilde{M}_{22}}}tg\varphi_2 - \frac{\widetilde{M}_{12}}{\widetilde{M}_{22}\sqrt{\widetilde{M}_{11}}}tg\varphi_1 \\
\omega - \omega_0 &= \frac{1}{\sqrt{M_{33}}}tg\varphi_3 - \frac{M_{23}}{M_{33}\sqrt{\widetilde{M}_{22}}}tg\varphi_2 - \frac{\left(\frac{M_{13}}{M_{33}} - \frac{M_{23}\widetilde{M}_{12}}{M_{22}\widetilde{M}_{22}}\right)}{\sqrt{\widetilde{M}_{11}}}tg\varphi_1 \\
q_z - q_{0,z} &= \frac{1}{\sqrt{M_{44}}}tg\varphi_4 \\
\widetilde{M}_{11} &= M_{11} - \frac{M_{13}^2}{M_{33}} - \frac{\widetilde{M}_{12}^2}{\widetilde{M}_{22}} \\
\widetilde{M}_{12} &= M_{12} - \frac{M_{13}M_{23}}{M_{33}} \\
\widetilde{M}_{22} &= M_{22} - \frac{M_{23}^2}{M_{33}}
\end{aligned} \qquad (3)$$

If this integral is evaluated numerically by asampling of integrand in a set of points uniformly distributed in ϕ-space, the sampling points will be distributed according to a Lorenzian in q,ω-space. The widths of the 4D-Lorenzian will be determined by the resolution ellipsoid. More points will get sampled near the resolution center, and less on the "tails". This allows one to achieve much better results with fewer sampling points. Speeds up computation tremendously!

`ConvRes.m`

Numerically convolutes a user-supplied cross section function S(q,ω) (a separate m-file) with the Cooper-Nathans resolution function in a given set of points in ω-q space. This process requires laborious calculations and one always has to compromise between computation times and accuracy. The sharper the

intrinsic features of S(q,ω), the more difficult it is to get an accurate result. In particular, if S has a singular (single-mode) form, straightforward 4D convolution doesn't work at all. A work-around is to artificially introduce some intrinsic width in the excitations. A much better approach is to use **ConvResSMA.m**, described below. In fact, try to use **ConvResSMA.m** whenever possible, as it is much faster than **ConvRes.m**.

ConvRes.m implements two algorithms of numerical integration. The "fixed-sampling" method samples S(q,ω) on a fixed grid in ϕ-space. Alternatively, the Monte Carlo algorithm can be used, eliminating any "structured" artifacts (a common problem with fixed sampling), but introduces "noise". MC is good for simulations, but the irreproducible "noise" will confuse many least-squares fitting routines. For an example of use see **ConvDemo.m**.

```
function conv=ConvRes(sqw,pref,H,K,L,W,EXP,METHOD,ACCURACY,p)
```

Input arguments:

- **'sqw'** is the name of the user-supplied cross section. It must be defined in a separate m-file and have the form:
  ```
  function s=sqw(H,K,L,W,p,sample,rsample)
  ```
  The arguments **H**, **K** and **L** specify the scattering vector in rec. lattice units, and **W** is the energy transfer (in meV). **p** is an additional parameter that can be anything: a number, an array, a structure, or a cell array. For convenience, lattice constants and reciprocal-lattice constants are passed of to **sqw** through the structures **sample** and **rsample** for internal use. These two arguments have the same structure as the lattice argument for **star.m**. **sqw** may or may not use the lattice constants, but must list as its arguments. See **SqwDemo.m** for an example.
  The function **sqw** must be vectorized, i.e., if **H**, **K**, **L** and **W** are vectors of length N, the output should be a vector of values of the same length. Correspondingly, the fields in **sample** and **rsample** are expected to vectors of length N as well: one set of lattice constants for each (h,k,l,ω) data point. Even when some of the input arguments are vectrors of length N, others may be scalars, and in this case should be treated as equivalents on length-N vectors of constant values.
- **'pref'** is the name of a user-supplied function that acts as a prefactor to **sqw** and is assumed to vary slowly with wave vector and energy. The actual structure factor S(q,ω) will be a product of **sqw** and **pref**. Unlike **sqw**, **pref** will not be convoluted with the resolution function, to save computation time. Typically, **pref** would include the magnetic form factor, absorption corrections or polarization factors. If **pref=[]**, then the structure factor will simply be as calculated by **sqw**. If used, the function **pref** must be defined in a separate m-file and be of the same form as **sqw**. See **PrefDemo.m** for an example.
- **H**, **K**, **L** and **W** specify the wave vector and energy transfers at which the convolution is to be calculated. **H**, **K** and **L** are given in reciprocal lattice units, and **W** in meV.
- The structure **EXP** contains all the information on the sample and lattice parameters. It has the form used with **ResMatS.m**, as described above.
- The string argument **METHOD** specifies which 4D-integration method to use. **ACCURACY** determines the number of sampling points in the integration. The following options are available:
  - **METHOD='fix'**: Sample the cross section on a fixed grid of points uniformly distributed in ϕ-space. If this option is chosen, **2*ACCURACY(1)+1** points are sampled along ϕ_1, ϕ_2 and ϕ_3, and **2*ACCURACY(2)+1** along ϕ_4 (vertical direction).
  - **METHOD='mc'**: 4D Monte Carlo integration. The cross section is sampled in **1000*ACCURACY** randomly chosen points, uniformly distributed in ϕ-space.
- The parameter **p** is passed on, without change, to **sqw** and **pref**. Very useful when you have a parameterized cross section that you want to fit to actual data sets.

Output arguments:

conv is the calculated value of the cross section, folded with the resolution function, at the give **H**, **K**, **L**, **W** position.

Vectorization:

The code is vectorized with respect to **`H`**, **`K`**, **`L`**, **`W`** and **`EXP`**. It can be used to simultaneously calculate convolutions at several ω-q points or even for several experimental setups. Any vectors among **`H`**, **`K`**, **`L`**, **`W W`** and **`EXP`** must be of the same length. If any of these are scalars, they will be automatically replicated into vectors of appropriate length. For input vectors of length N the result is, correspondingly, a length-N row-vector of convolution values.

Dependencies:

To properly function **`ConvRes.m`** requires the following m-files from ResLib: **`ResMatS.m`**, **`ResMat.m`**, **`StandardSystem.m`**, **`star.m`**, **`scalar.m`**, **`modvec.m`**.

Convolution with resolution function: single-mode and Voigt profiles

In many cases the dynamic structure factor can be broken down into several single-mode contributions:

$$S(\vec{Q}) = \sum_{i=1}^{N} s_i(\vec{q})\,\delta(\omega - \omega_i(\vec{q})) \tag{4}$$

Here $s_i(\mathbf{q})$ is the intensity of the i-th mode, and $\omega_i(q)$ is its dispersion relation. It may appear discouraging that **`ConvRes.m`** becomes unusable in this case, as it can not handle infinitely sharp features of the cross section. A work-around would be to artificially introduce finite energy-widths of excitations. A much better approach is to eliminate the delta-functions analytically, reducing a 4-D numerical convolution to a much faster 3-D convolution.

This procedure can be generalized to the case when each mode is characterized by some intrinsic Lorenzian energy-width $\Gamma_i(q)$:

$$S(\vec{Q}) = \frac{1}{\pi}\sum_{i=1}^{N} s_i(\vec{q})\,\frac{\Gamma_i(\vec{q})}{\Gamma_i(\vec{q})^2 + [\omega - \omega_i(\vec{q})]^2} \tag{5}$$

While a convolution of a Gaussian and a Lorentzian (so-called Voigt function) can not be computed analytically, some very efficient numerical approximations are available [8] .

In either case, the ϕ-transformation is performed only in 3 dimensions. Exact expressions used in ResLib are as follows:

$$\left.\frac{d\sigma}{d\Omega\,dE'}\right|_{\substack{\vec{k}-\vec{k}'=\vec{q}_0 \\ E-E'=\hbar\omega_0}} \approx R_0(\vec{Q}_0)\,\frac{1}{\sqrt{\widetilde{M}_{11}\widetilde{M}_{22}-\widetilde{M}_{12}^2}\sqrt{\widetilde{M}_{33}}}\sum_{i=1}^{N}\int_{-\pi/2}^{\pi/2}d\varphi_1\int_{-\pi/2}^{\pi/2}d\varphi_2\int_{-\pi/2}^{\pi/2}d\varphi_3 s_i(\vec{q})V(\widetilde{\omega}_i,\widetilde{\Gamma}_i)\frac{\exp\left[-\frac{1}{2}tg^2\varphi_1-\frac{1}{2}tg^2\varphi_2-\frac{1}{2}tg^2\varphi_3\right]}{\cos^2\varphi_1\cos^2\varphi_2\cos^2\varphi_3}$$

$$\widetilde{\Gamma}_i = \Gamma_i\frac{1}{\sqrt{M_{33}}}$$

$$\widetilde{\omega}_i = \omega - \omega_i(\vec{q}) + \frac{M_{13}q_x + M_{23}q_y}{\sqrt{M_{33}}}$$

$$V(x,y) \equiv \frac{1}{\pi}\int dt\,\exp(-t^2)\frac{y}{y^2+(t-x)^2}$$

$$q_y - q_{0,y} = \frac{\sqrt{\widetilde{M}_{11}}}{\sqrt{\widetilde{M}_{11}\widetilde{M}_{22}-\widetilde{M}_{12}^2}}tg\varphi_2 \tag{6}$$

$$q_x - q_{0,x} = \frac{1}{\sqrt{\widetilde{M}_{11}}}tg\varphi_1 - \frac{\widetilde{M}_{12}}{\sqrt{\widetilde{M}_{11}\widetilde{M}_{22}-\widetilde{M}_{12}^2}\sqrt{\widetilde{M}_{11}}}tg\varphi_2$$

$$q_z - q_{0,z} = \frac{1}{\sqrt{M_{44}}}tg\varphi_3$$

$$\widetilde{M}_{11} = M_{11} - \frac{M_{13}^2}{M_{33}}$$

$$\widetilde{M}_{12} = M_{12} - \frac{M_{13}M_{23}}{M_{33}}$$

$$\widetilde{M}_{22} = M_{22} - \frac{M_{23}^2}{M_{33}}$$

[8] J. Humlicek, J. Q. Spec. Rad. Transfer 27, 437 (1982); F. Schreier,J. Q. Spec. Rad. Transfer 48, 743 (1992).

ConvResSMA.m

ConvResSMA.m is specifically designed to handle user-supplied cross section functions written is the single-mode form. Having one dimension less to deal with, it is much faster than **ConvRes.m** and should be used in its stead whenever possible. For an example of use see **ConvDemo.m**.

```
function conv=ConvResSMA(sqw,pref,H,K,L,W,EXP,METHOD,ACCURACY,p)
```

Input arguments:

- '**sqw**' is the user-supplied cross section. It must be defined in a separate M-file and have the form:
  function [w0,S,HWHM]=sqw(H,K,L,p,sample,rsample)
  The input arguments are as those for the cross section function for **ConvRes.m**. See **SMADemo.m** for an example. For M modes and N data points the output arguments are MxN matrixes. Fore mode j at reciprocal-space point (h_i,k_i,l_i) **s(j,i)** is the mode intensity s_j, **w0(j,i)** is mode energy ω_j (in meV) , and **HWHM** is the Lorenzian energy half-widths at half-height Γ_j. If **HWHM=0**, the mode is assumed to have a d-function energy profile.
  The function **sqw** must be vectorized, i.e., if **H**, **K** and **L** vectors of length N, the outputs should be vectors of values of the same length. Correspondingly, the fields in **sample** and **rsample** are expected to vectors of length N as well: one set of lattice constants for each (h,k,l,ω) data point. Even when some of the input arguments are vectrors of length N, others may be scalars, and in this case should be treated as equivalents on length-N vectors of constant values.
- '**pref**' has a similar meaning to that in **ConvRes.m**. The difference is that it calculates a whole set of prefactors, one for each mode defined by **sqw**. For M modes and N data points, it should return an MxN matrix. It uses the same set of input arguments as **sqw**. See **SMAPrefDemo.m** for an example.
- **H**, **K**, **L**,**W**, **p**, and **EXP** have the same meaning and form as for ConvRes.m.
- The string argument **METHOD** specifies which 3D-integration method to use. **ACCURACY** determines the number of sampling points in the integration. The following options are available:
  - **METHOD='fix'**: Sample the cross section on a fixed grid of points uniformly distributed in ϕ-space. If this option is chosen, **2*ACCURACY(1)+1** points are sampled along ϕ_1 and ϕ_2, and **2*ACCURACY(2)+1** along ϕ_3 (vertical direction).
  - **METHOD='mc'**: 3D Monte Carlo integration. The cross section is sampled in **1000*ACCURACY** randomly chosen points, uniformly distributed in ϕ-space.

Output arguments:

See output arguments for **ConvRes.m**.

Vectorization:

See vectorization of **ConvRes.m.**

Dependencies:

To properly function **ConvResSMA.m** requires the following m-files from ResLib: **ResMatS.m**, **ResMat.m**, **StandardSystem.m**, **star.m**, **scalar.m**, **modvec.m**.

Least-squares fitting of convoluted cross sections

FitConv.m

This function provides Levenberg-Marcquardt least squares fitting of convoluted cross secions to experimental data. The algorith is exactly as in NR 15.5. Analytic derivatives are not supported in the current version. This function calls **ConvRes.m** for convolution calcualtions at each iteration step. An example of use is given in **FitDemo.m**.
The function call has the following form:

```
function [pa,dpa,chisqN,sim,CN,PQ,nit,kvg,details]
=FitConv(H,K,L,W,EXP,Iobs,dIobs,sqw,pref,pa,ia,METHOD,
ACCURACY,nitmax,tol,dtol)
```

Input arguments:

- '**sqw**', '**pref**', **METHOD** and **ACCURACY** are exactly as in **ConvRes.m**
- **H**, **K**, **L**, **W**, **EXP** are vectors that contain the wave vector components, energy transfers, and experimental conditions for each experimental data point. They all must be of the same length or scalars. See **ConvRes.m** for more details.
- **Iobs**, **dIobs** are vectors of the same length as **H…EXP**, and contain the observed scattering intensities and error bars for each data point, respectively.
- **pa** is the initial guess for parameter values. This should be a 1-dimensional array of the form accepted by the **sqw** and **pref** functions.
- **ia** (optional) - a list with zero elements corresponding to fixed parameters and ones for parameters that should be varied. This array should be of the same size as pa. The default value is **ia=ones**.
- **nitmax** (optional)- the maximum number of iterations allowed before the program exits. The default value is **nitmax=20**.
- **tol** (optional)- the convergence criterion: convergence declared when chi-squared decreases by a relative amount less than **tol**. The default value is **tol=0.001**.
- **dtol** (optional)- parameter for relative change in numerical derivatives, default **dtol=1e-5**.

Output arguments:

- **pa**,**dpa** - refined parameters and error bars.
- **chi2N** - chi-squared normalized by degrees of freedom.
- **CN** - normalized correlation matrix - includes varying parameters only (in order).
- **PQ** - probability that chi2N exceeds that observed.
- **nit** - number of iterations to convergence.
- **kvg** - convergence flag:
  kvg=0 did not converge due to too many iterations **nit >= nitmax**
  kvg=1 converged normally
  kvg=2 questionable convergence (**final_lamda> 1e-3**)
- **sim** - (optional) fitted value of convoluted cross section at input points
- **details** - optional output structure with fields:
  - **chisq** (raw chi-squared)
  - **DF** (degrees of freedom)
  - **Ndata** (number of data points)
  - **Npar** (total number of parameters)
  - **Nvar** (number of parameters varied)
  - **C** (raw covariance matrix)
  - **final_lamda** (final value of λ used in the fitting process)
  - **yf** (fitted values of function at input points)

Dependencies:

To properly function **FitConv.m** requires the following m-files from ResLib: **ConvRes.m**, **ResMatS.m**, **ResMat.m**, **StandardSysem.m, star.m**, **modvec.m**, **scalar.m**.

FitConvSMA.m

This function is very similar to **FitConv.m**, but uses SMA cross sections as in **ConvResSMA.m**.
The function call has the following form:

```
function [pa,dpa,chisqN,sim,CN,PQ,nit,kvg,details]
=FitConvSMA(H,K,L,W,EXP,Iobs,dIobs,sqw,pref,pa,ia,METHOD,
ACCURACY,nitmax,tol,dtol)
```

Input arguments:

'**sqw**', '**pref**', **METHOD** and **ACCURACY** are exactly as in **ConvResSMA.m**
Other parameters as in **FitConv.m**.

Dependencies:

To properly function **FitConvSMA.m** requires the following m-files from ResLib: **ConvResSMA.m**, **ResMatS.m**, **ResMat.m**, **StandardSysem.m**, **star.m**, **modvec.m**, **scalar.m**.

Graphic representations of resolution ellipsoids

It is often very helpful to visualize resolution ellipsoids for each point in an inelastic scan, see how they change from one point to the next, and try to understand how they are oriented relative to the dispersion surface in the system under investigation. Also, while the resolution matrix contains all the information on the instrumental resolution at a given point, parameters such as resolution volume, projected energy width, Bragg peak width, etc., are much more useful in planning actual measurements. ResLib provides two routines for visualizing resolution ellipsoids and calculating the most relevant resolution parameters.

ResPlot.m

This function plots projections and sections of resolution ellipsoids for a user-defined scan, and calculates a bunch of relevant resolution characteristics for the center-point of the scan. As an option, it also plots the dispersion relation calculated from a user-supplied SMA cross section function, of the type used by **ConvResSMA.m**. The projection and section planes for resolution ellipsoids are defined by the orienting vectors specified in **EXP**. By changing these one can get projections and sections with any desired reciprocal-space planes. An example of use can be found in **PlotDemo.m**. **ResPlot.m** has the following form:

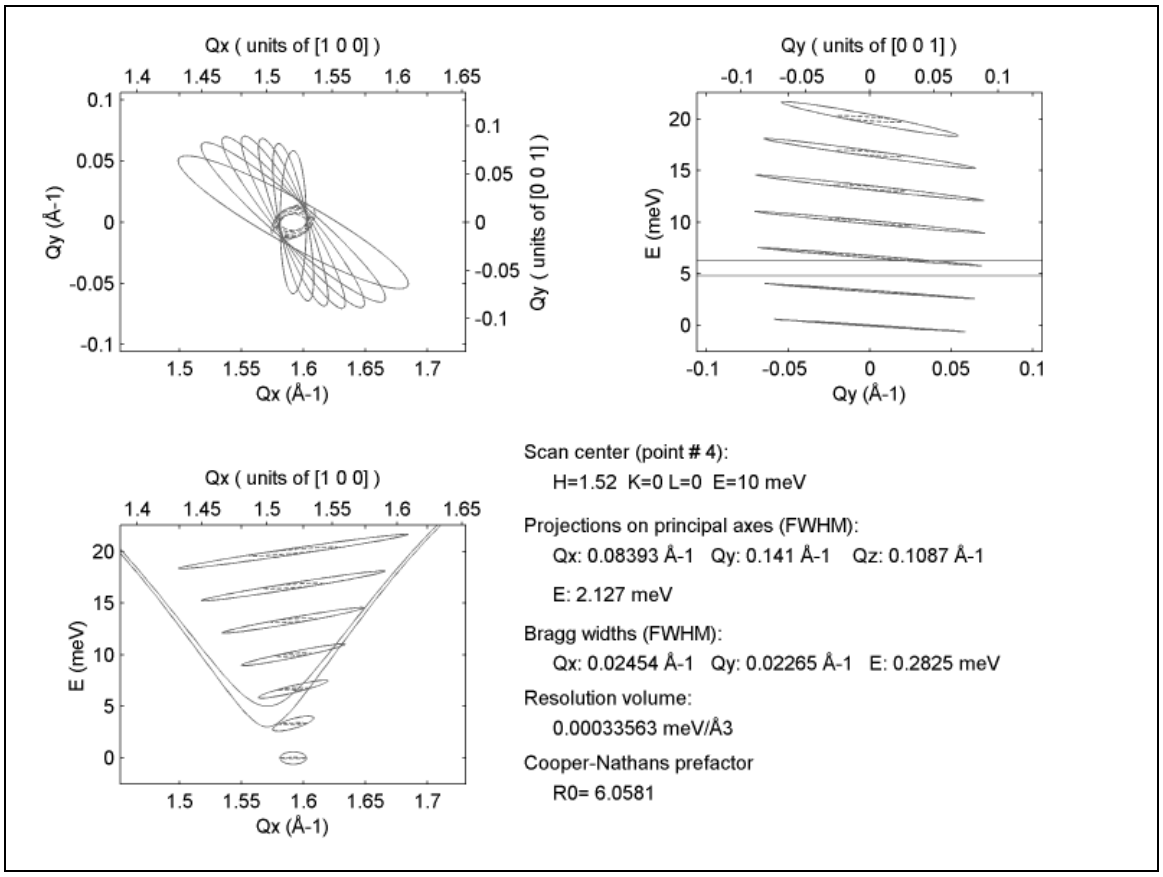
```
function ResPlot(H,K,L,W,EXP,SMA,SMAp)
```

Input arguments:

- **H**, **K**, **L**, **W**, **EXP** are vectors that contain the wave vector components, energy transfers, and experimental conditions for each experimental data point. They all must be of the same length or scalars. See **ConvRes.m** for more details. **ResPlot.m** is designed to work with a single scan, measured in a particular experimental setup. Therefore, calculations for *all* scan points will be performed using experimental conditions for the *center-point* of the vector EXP (center of scan).
- **SMA** (optional) is the name of an SMA cross section function to be used for are vectors of the same length as **H…EXP**, and contain the observed scattering intensities and error bars for each data point, respectively.
- **SMAp** (optional) - parameters to be passed to the cross section function.

Output:

All the information is sent to the current figure. A typical output looks like this:



- Upper left: projection onto the scattering plane (large ellipses) and constant-energy sections (inner ellipses).
- Lower left: Q_x-E projections and sections. Additional lines (here: parabola) show the dispersion of the two branches defined in the supplies SMA cross section function. The Q_y and Q_z values for this dispersion curve are taken for the mid-point in the scan.
- Upper right: Q_y-E projections and sections. Additional lines (here: straight lines)show the dispersion of the two branches defined in the supplies SMA cross section function. The Q_x and Q_z values for this dispersion curve are taken for the mid-point in the scan.

Dependencies:

To properly function **`PlotRes.m`** requires the following m-files from ResLib: **`ResMatS.m`**, **`ResMat.m`**, **`StandardSysem.m`**, **`star.m`**, **`modvec.m`**, **`scalar.m`**.

`ResPlot3D.m`

This function is similar to **`ResPlot.m`** but plots a very neat 3D representation of resolution ellipsoids, projections and dispersion surfaces. See **`Plot3DDemo.m`** for an example of use.

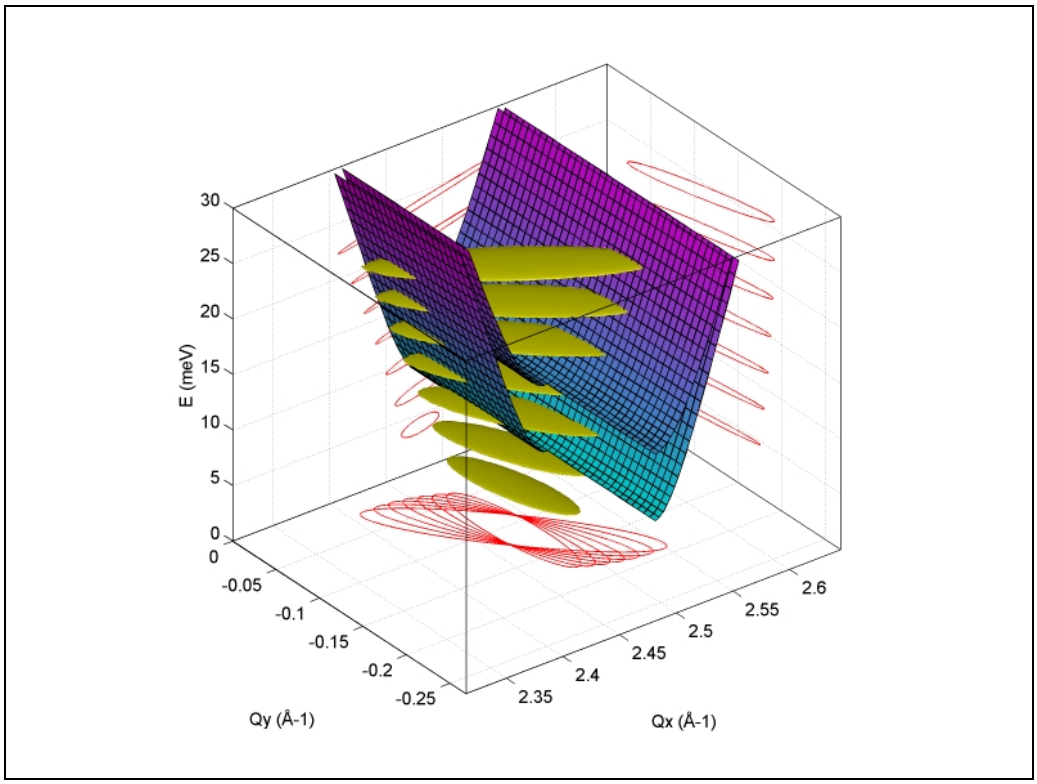
```
function ResPlot3D
(H,K,L,W,EXP,RANGE,EllipsoidStyle,XYStyle,XEStyle,YEStyle,SMA,SMAp,SXg,
SYg)
```

Input arguments:

- **H**, **K**, **L**, **W**, **EXP** are as in **ResPlot.m**.
- **RANGE** defines the range of momentum and energy transfer to use in the plot. RANGE must be of the form [Qxmin, Qxmax, Qymin, Qymax, Emin, Emax], in $Å^{-1}$ and meV, respectively.
- **EllipsoidStyle** (optional) is the color of resolution ellipsoids in the plot. Default is **EllipsoidStyle='red'.**
- **XYStyle** (optional) - line style to use for the projection onto the scattering plane, in the format used with MatLab's **plot** command (something like **'r--'** for a red dashed line). Use the value **'none'** to suppress projections.
- **XEStyle** (optional) - line style to use for the projection onto the Q_x-E plane, in the format used with MatLab's **plot** command. Use the value **'none'** to suppress projections.
- **YEStyle** (optional) - line style to use for the projection onto the Q_y-E plane, in the format used with MatLab's **plot** command. Use the value **'none'** to suppress projections.
- **SMA** and **SMAp** (both optional) are as in **ResPlot.m**.
- **SXg** and **SYg** (both optional) define the 2D grid of Q_x and Q_y values for plotting the dispersion surfaces. They must be of the same form as used in MatLab's **surf** command, and as generated by **meshgrid**. A good example of use is given in **Plot3Ddemo.m**. If omitted, the dispersion surface is plotted of a 40x40 grid within the boundaries specified by the first 4 elements of **RANGE**.

Output:

All the information is sent to the current figure. A typical output looks like this:



Dependencies:

To properly function **ResPlot3D.m** requires the following m-files from ResLib: **ResMatS.m**, **ResMat.m**, **StandardSysem.m, star.m**, **modvec.m**, **scalar.m**.

Downloads, packaging and installation

File download

The most recent version of all ResLib files are available here for download as .zip or .tar.gz archives from http://neutrons.phy.bnl.gov/ResLib.

Packaging:

ResMat.m - calculate Cooper-Nathans resolution matrixes.
ResMatS.m - same thing, in a coordinate system defined by the scattering vector.
ConvRes.m - convolute a user-defined cross-section function with the resolution. ***ConvResSMA.m*** - convolute a user-defined single-mode cross-section with the resolution.
FitConv.m - fit convoluted cross section to experimental data.
FitConvSMA.m - fit convoluted SMA cross section to experimental data.
ResPlot.m - 2-D visualization of resolution ellipsoids and dispersion.
ResPlot3D.m - 3-D visualization of resolution ellipsoids and dispersion.
scalar.m - scalar product of vectors defined by fractional cell coordinates or Miller idexes.
modvec.m - length of vectors defined by fractional cell coordinates or Miller idexes.
star.m - calculate reciprocal-lattice parameters, unit cell volume and reciprocal volume.
StandardSystem.m - internal use.
MakeExp.m - Example of seting up a structure that contains details on experimental conditions for use with ResMat and ConvRes.
SqwDemo.m - example of a user cross section function for use with ConvRes.
SMADemo.m - example of a user single-mode cross section function for use with ConvResSMA.
PrefDemo.m - example of a "slowly varying prefactor" function for use with ConvRes and SqwDemo.
SMAPrefDemo.m - example of a "slow prefactor" function for use with ConvResSMA and SMADemo.
ConvDemo.m - a demo script of the convolution routines.
FitDemo.m - a demo script of the fitting routines.
PlotDemo.m - a demo script of ResPlot
Plot3DDemo.m - a demo script for ResPlot3D
Demo.dat - an actual data set used in FitDemo
ResLib.m - list of functions supplied in ResLib.
Manual.pdf - this file.

Installation

Uncompress the archive to a separate new directory and add it to your MatLab path by using the **`addpath`** command. Run the **`ConvDemo`** script to see that ResLib works on your system. Run the **`PlotDemo`** and **`Plot3DDemo`** scripts to get some nice graphics. Change to the directory where the files are installed (to have **`Demo.dat`** in the current directory) and try **`FitDemo`**. This last one may take a long time though…

Credits

Many ideas and entire code fragments are borrowed from IDL routines written at Johns Hopkins University by Collin L. Broholm and Igor Zaliznyak (now at BNL). Other code fragments originally written by Peter Boni at BNL (now at PSI) for the old-and-reliable BNL FIT3AX Fortran program. The resolution calculation was tested against the very reliable Resolution program written at BNL for Macintosh by Ben Sternlieb. Igor Zaliznyak helped in testing ResLib agains his own IDL routines. Levenberg-Marquardt algorithm based on C Numerical Recipies, as adapted for MatLab by Stephen E. Nagler (ORNL). Voigt function by A. N. Maurellis.

Disclaimer

There obviously is no guarantee that the above routines work right. 3-axis resolution calculations are a tricky business and have on many occasions led to incorrect interpretation of experimental data, outright wrong publications, and destruction of glorious careers of young neutron scatterers. Use these m-files freely... at your own risk.