- # Real-Space Modeling of SANS Data

FEB 2012 SRK

MAR 2014 SRK

This help file provides instructions for how to use the beta operations available to calculate the scattering from arbitrary shaped objects, as defined in real space. There is a sction that describes the general theory and operation of the macros, then many sections that are examples of the macros in use. These sections are also verification of the correct operation of the various methods to calculate the intensity. Be warned that the operations presented here are in the beta stage. Every attempt has been made to make sure that the calculations are fundamentally correct. The user interface and features available are crude and will be improved with user feedback. Be warned that some of the calculations can be very slow. I have tried to give fair warning to abort lengthy calculations before they start. The actual time predictions will be off, as they are machine dependent.

For detailed instructions, see the sections:

(other examples as I add them to the Macros Menu)

- # Calculating I(q) for Arbitrary Shapes

Updated 6 JAN 2011 SRK

Calculation of the scattered intensity for arbitrary shapes can be done by one of two common methods. Both of these methods have been implemented, and each has particular advantages.

[1] Debye's method of spheres
[2] The "cube" method

In each method, a volume is defined and subdivided up into N x N x N cubes of equal edge length T. A two scattering length density material is defined by either a filled or empty voxel. In general, within the limits of computing power and matrix size, N should be as large as possible and T should be as small as possible.

The most time consuming step to calculate I(q) from either method is the construction of the object. Only simple algorithms have been implemented so far in order to demonstrate the validity of the method.

[1] The sphere method

In this method, the structure is defined in the volume as filled spheres at the center of cubes in a matrix of empty cubes. The primary spheres have the same volume as the voxel, meaning that adjacent spheres overlap. The scattering is calculated as a double sum of the filled voxels only, with calculation of all of the relative distances. The time-consuming step is the double summation.

There are three versions:

DebyeSpheresX: this version uses the full double sum, and can make use of the SLD information, if available. It's XOPed and multithreaded, and that's about as good as it gets for a simplistic calculation like this. It's correct and it's the slowest calculation around. Only use this if you really don't believe any of the other calculations.
- (Do Debye Spheres) button
- Output is ival_full vs qval_full

CalcIQRfromMat_bin: this version used discretized bins for the distances. Implicitly in this is the assumption that all primary spheres are the same size, and the same SLD. These assumptions make the calculations much faster. The double loops in the calculations have been XOPed for speed.
- (Do Binned Debye) button.
- Output is ival_XOP vs. qval_XOP.

CalcIQRfromMat_bin_SLD: this version used discretized bins for the distances. This version, however, allows for the spheres to have different SLDs. These assumptions make the calculations faster than the full double sum, but slower than the binned version where all SLDs are the same. The double loops in the calculations have been XOPed for speed.
- (Do Binned SLD) button.
- Output is ival_SLD vs. qval_SLD.

Pros: You pick the number of q-points and the exact q-range you want
The calculation time is directly proportional to the number of q-points
Can be used as a fitting function if carefully defined.

Cons: For a given N, the total computation time actually depends most strongly on the number of spheres that make up the defined structure.
The orientational average is done by the calculation over the entire volume - no anisotropic information can be derived.

References:

The Simple Debye Spheres calculation is the classic formual of Debye. The distance binned versions make use of the works of O. Glatter and of E. Pantos. Specifically, the binned method where there are different SLDs uses the reference:

E. Pantos, H. F. van Garderen, P. A. J. Hilbers, T. P. M. Beelen, R. A. van Santen, *J. Molec. Struct.* **383** (1996) 303-308.

E. Pantos and J. Bordas, *Pure & Applied Chem.* **66** (1994) 77-82.

The FFT method is based on the following references:

I. S. Barnes, T. Zemb, J. Appl. Cryst. 21 (1988) 373-379. See also K. Rohr.

[2] The cube method

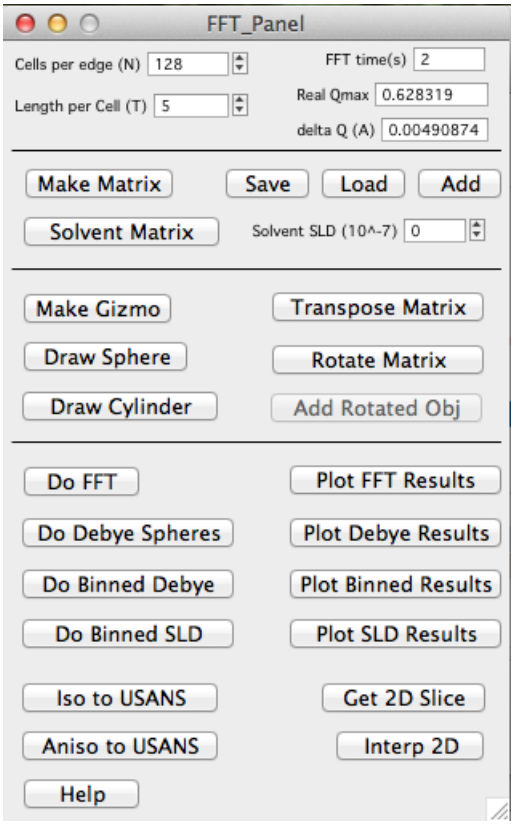
In this method, the structure is defined in the volume as filled cubes in a matrix of empty cubes. The scattering in q-space is obtained directly as a 3D FFT of the volume. The magnitude of the real portion of the result is I(q). The time-consuming step is the binning of the 3D FFT result into 1-D.

Pros: For a given N, the total computation time does not depend on the defined structure.
It is possible to compute directional I(q) since the FFT result is 3D and is averaged to get to 1D data.
The method is intuitive, makes use of optimized FFT libraries.

Cons: The q-range, resolution, and number of q-points is completely defined by the choice of N and T.
N sets the number of points
T sets the Q range
Q: ranges from 0 to 2Pi/T
Delta Q = Pi/NT
(Q > Pi/T is NOT to be trusted since it's the size of an individual cube)

To Use:
The necessary procedures are automatically included. The generic steps are as follows:

1) Macros -> "Load Real Space Modeling" will load all of the necessary procedures and initialize the main panel.



2) Set the parameters of the defining matrix
N is the number of cells in the overall cube (N x N x N) (128x128x128 is the default)
T (in Angstroms) is the physical size of each of the cells (voxels)
Qmax and deltaQ are calculated values that will result from the FFT method
FFT time is only an estimate of the time required, varying as the cube of N.

3) [Make Matrix]
mat is the 3D (byte) matrix of empty/filled (0/1/2/etc.) voxels that define the structure. Note that "empty" = "solvent" and does not need to be 0. Other SLDs can take on different values than 1 as well. The default for the matrix isto be initially filled with a "solvent" of SLD = 0 x 10^-7 (1/A^2). If you want something else, you'll need to enter the value and click [Solvent Matrix]. This will obliterate everything

you've drawn, so be sure to do this step first. Since the multiplier is 10^-7, a value of 63 would be "D2O" as a solvent.

4) [Make Gizmo]
This brings up a 3D plot of the matrix. The box defines the volume. The incident neutron beam is in the -x direction, with the detector in the YZ plane with Z up and Y to the right. With the Gizmo plot on top, from the Gizmo menu, choose "Get Info" to adjust the plot, or other options available. Most commonly you'll want to adjust what voxels are visible. To do this, double-click on "voxelgram0". Here, you can set the size of the points that represent each voxel and up to 5 different discrete values that will be shown. The alpha value controls the transparency. Typical values are less than 0.3

5) Draw something

[Draw Sphere]
Use the wave "mat", and give the real radius and xyz position. Set the SLD value as a 1- or 2-digit integer (it wil be multiplied by 10^-7). Note that the values can be negative too. Setting the value equal to the solvent is equivalent to "erasing" the object, or part of the object.

[Draw Cylinder]
Same idea, just for a cylinder oriented along one of the primary axes.

[Rotate Matrix]
Rotates the matrix by the specified amount around the z-axis. If you have something very large, it may be clipped due to the rotation. The rotated object may be somewhat fuzzy as well due to the mis-registration of the object off of the grid.

[Transpose Matrix]
Transforms the matrix XYZ to any of the other permutations. Good if you've drawn an object and want to see how the 2D scattering would look with a different incident neutron view of your object. See Transposing the Matrix

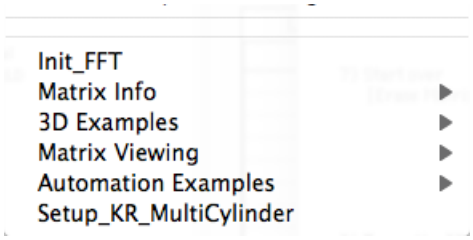
6) Calculate
[Do FFT] or [Do DebyeSpheres] or [Do Binned Debye] or [Do Binned SLD]

7) Plot
[Plot FFT Results] or [Plot Debye Results] or [Plot Binned Results] or [Plot SLD Results]

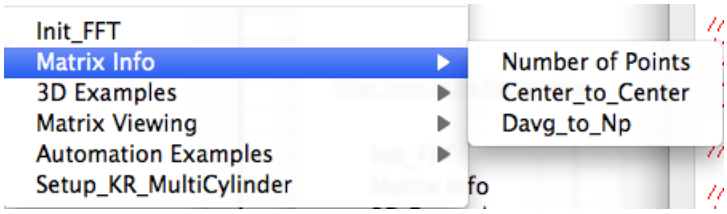
8) Or plot the results in 2D with [Get 2D slice]. This will show a 2D slice of the data in the detector plane in both log scale and linear scale. There is also a 1D representation where the slice only is averaged, not the full 3D FFT (so there is no orientational average of the structure). Remember that the direction of the beam is in the (-)x-direction. The detector is in the YZ plane (Z vertical, Y to the right). [Interp 2D] will interpolate the 2D FFT to match the q-range of a 2D data set that you have already loaded. Currently this can only be used for visualization purposes.

9) Back at the top of the panel there are [Save] and [Load] and [Add] buttons. These allow you to save the matrix that you've drawn - since sometimes it can be very time-consuming to draw it, and then re-load the matrix back in for display and re-calculation. Load will wipe out what's there, Add will add the two matrices together.

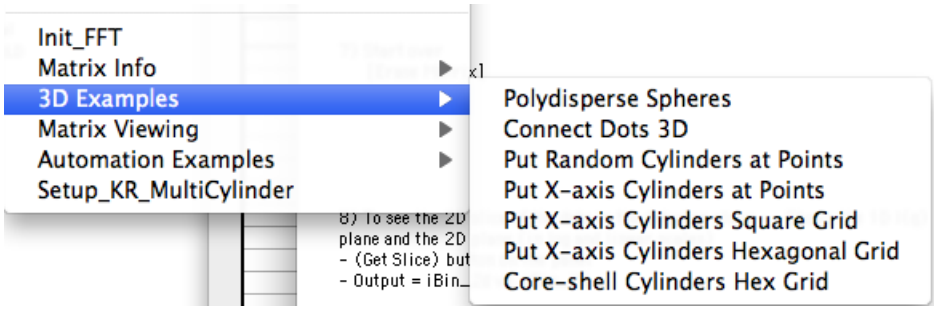
Other items on the Macros menu:



- The initialization is done automatically when these procedures are loaded, but you may re-initialize if you wish to reset to default values.



The "Matrix Info":
- Number of Points: The number of spheres or voxels that are occupied with something other than solvent. Also reports the fractional occupation (= volume fraction) and the overall size of the bounding box. Also reported is a list of the different SLD values that are present in the matrix (quite useful to use in the Gizmo voxel display list).
- Center-to-center: A simple calculation given an input number of points (or spheres) in the current box, the calculation returns the average center to center separation of the points if they were equally distributed thoroughout the volume.
- Davg_to_Np: This is the reverse calculation as C-to-C. The separation is input, and the required number of particles is returned.



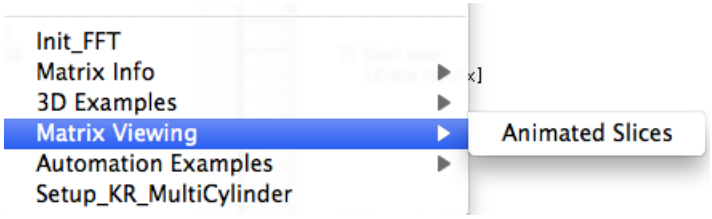
- Polydisperse Spheres: Calculates the scattering from concentrated spheres. The number, radius, polydispersity care all input. The fill/calculation is repeated nPass times and the results are averaged. See FFT of Concentrated Hard Spheres

- Connect Dots 3D: A number of points (nodes) are placed either randomly or using a Sobol sequence. Then the points are connected with a single voxel thickness, up to the connectivity number. See Connected Rods

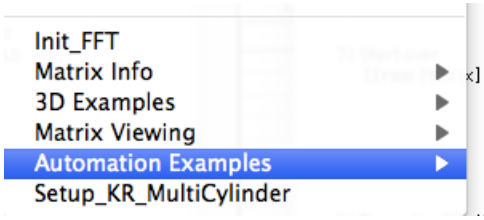
- Put Random Cylinders at Points: A number of points (nodes) are placed either randomly or using a Sobol sequence. Then each point becomes the center of a cylinder of the specified length (of voxels). The cylinders are 1 voxel in diameter and are randomly oriented. Periodic boundaries are allowed.

- Put X-axis Cylinders at Points:

- Put X-axis Cylinders Square Grid:
- Put X-axis Cylinders Hexagonal Grid:
- Core-shell Cylinders Hex Grid: These four routines fill the volume with cylinders where the cylinders are oriented parallel, along the x-axis. The number of cylinders, their length, radius, etc. can be specified. The The macros use different grids as noted. See Filling Lattice Shapes



- Animated slices: shows a loop of 2D images lsicing through the volume.



- Nothing here yet, but this section will have some examples of how you can write procedures to step through and save lengthy calculations that can run for many hours.

- **USANS from Real-space Structures**

Nov 2013

These instructions show the steps necessary to generate a real space structure and convert it to a 1D slit-smeared USANS representation. For structures that have an analytical function, that represenetation can be used directly with UCALC and is the better choice. But for non-regular structures, or for anisotropic structures, real space modeling is a viable technique. USANS is unique in the very sharp resolution in the x-direction (horizontal) and extremely poor resolution in the y-direction. (Note that in the "Gizmo" plot of the real space structure, the detector plane is y-z, and the beam is in the x-direction. Sorry. Different "standard conventions".

So the question is - for anisotropic structures, what does the 1D USANS look like?

For SANS, no resolution smearing is done for any of the real space calculations. (It's just not worth it, and until someone convinces me that it's necessary, it won't be done.)

Setup:
Setup for these calculations is no different than usual, other than being sure to set N and T large

enough to make the box large enough for USANS. The goal is to get the deltaQ small enough to approximate the dQ of the USANS instrument.

N=256 and T=500 is about the smallest you can choose: 256 x 500 = 128,000 Å = 12.8 microns per box side. Configurations with dQ closer to 1e-5 (1/A) are better.

N	T (A)	Box(um)	Qmax(1/A)	dQ(FFT) (1/A)
256	500	12.8	0.0063	2.4e-5
256	1000	25.6	0.0031	1.2e-5
400	500	20	0.0063	1.6e-5

If the box is too small, you'll be warned and not allowed to proceed. Once you've got the real space structure set to what you want (remember that you can use multiple SLDs): you can calculate the USANS intensity using the buttons on the FFT_Panel:

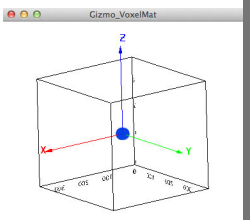
Iso to USANS
or
Aniso to USANS

Isotropic - output is FFT_iUSANS_i vs FFT_iUSANS_q. In 2D, the isotropic output is avg2D and logAvg2D arrays. Use this when you want to orientationally average the real space structure and see what USANS would measure.
Anisotropic - output is FFT_aUSANS_i vs FFT_aUSANS_q. In 2D, the anisotropic output is the detPlane and logP arrays, as generated by the "Get 2D Slice" button. Use this when you have an oriented sample and want to see what USANS would measure. Note again that the beam direction is (-x) and you're seeing the scattering in the YZ plane as defined by the axes on the Gizmo plot. USANS then has good resolution in the "Y" direction and no resolution in the "Z" direction. "USANS_Half" is a data folder containing a representation of the +x "half" of a 2D detector with deltaQ equal to the USANS resolution in that direction. Then the columns are simply summed to generate the USANS slit-smeared data, which is essentially what is happening in a USANS experiment.

These macros will go through all of the steps of calculating the FFT of the structure, and then calculating the appropriate USANS smeared intensity. The 2D representations of the calculation will be automatically displayed along with "iBin", which is the pinhole SANS result for comparison.

Examples:

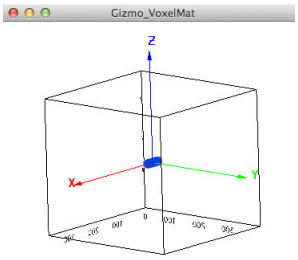
Sphere - to show that the calculation works, and yes, it does:
Using N=400 and T=500, and drawing a sphere of R=10,000 A

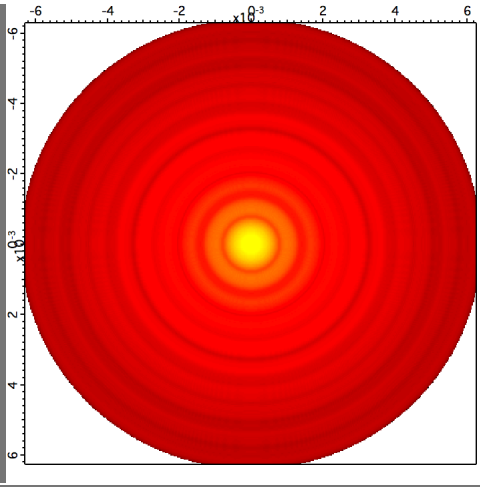


Both the anisotropic and isotropic USANS calculations were done, and are identical to the analytical USANS smearing of the sphere model. iBin is the unsmeared sphere calculation. The isotropic and anisotropic detector planes are identical, so only one is shown.

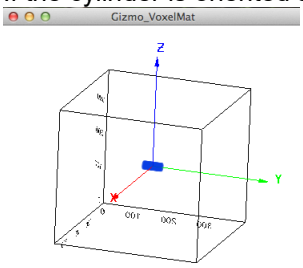
Cylinder - a single cylinder to show that the anisotropic representation is different than the isotropic calculation and that the isotropic calculation matches the analytic calculation.
(the isotropic version is important so that say, non-analytic structures in "solution" can be converted to slit-smeared USANS data)

Using N=400, T=500 and a cylinder of R=5,000 A and L=25,000 A oriented along the x-direction (the beam is in the x-direction, so this is "end-on" into the beam). And the results are:


Gizmo_VoxelMat
Z
X
Y

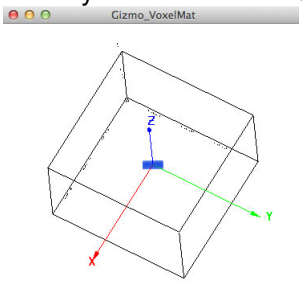

-6
-4
-2
0
2
4
6
x10-3

If the cylinder is oriented along the y-direction (crosswise to the beam):



Now the isotropically averaged USANS is unchanged, but the anisotropic data has fringes on a longer length scale as a result of the longer length of the cylinder now pointing in the horizontal direction where the USANS resolution is best.

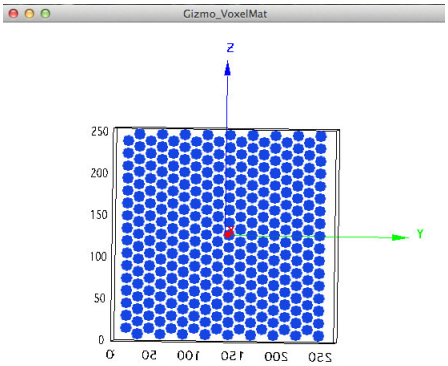
if the cylinder is oriented 30 degrees to the beam:



Hexagonal rods + rotations - to show what the smeared representation looks like - to show the effect of the rotation angle on the peak locations in I(q). Somewhat unexpected results, at least until you stop and think for a few minutes. Can I compare this to a paracrystalline model? Since it's cylinders, does the HCP spacing for spheres apply?

Using N=256, T=1000, make a hexagonal array of cylinders of R=5000, L=50,000, and center-to-center separation of d=15,000 Å

Aligned end-on into the beam:

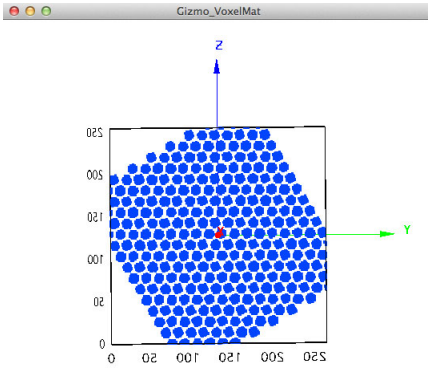


[redacted]

[redacted]

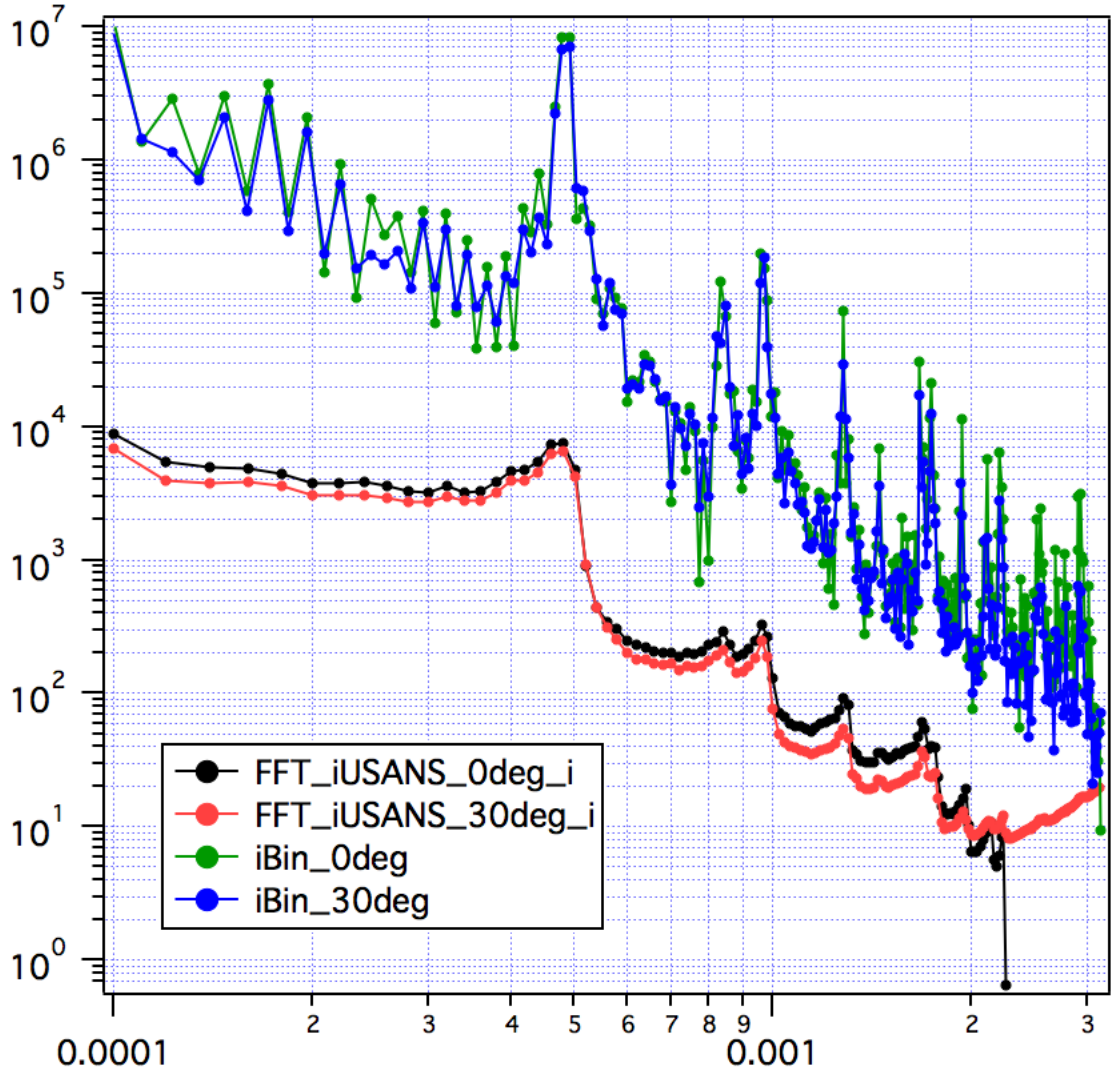
Rotated 30 degrees WRT detector plane:

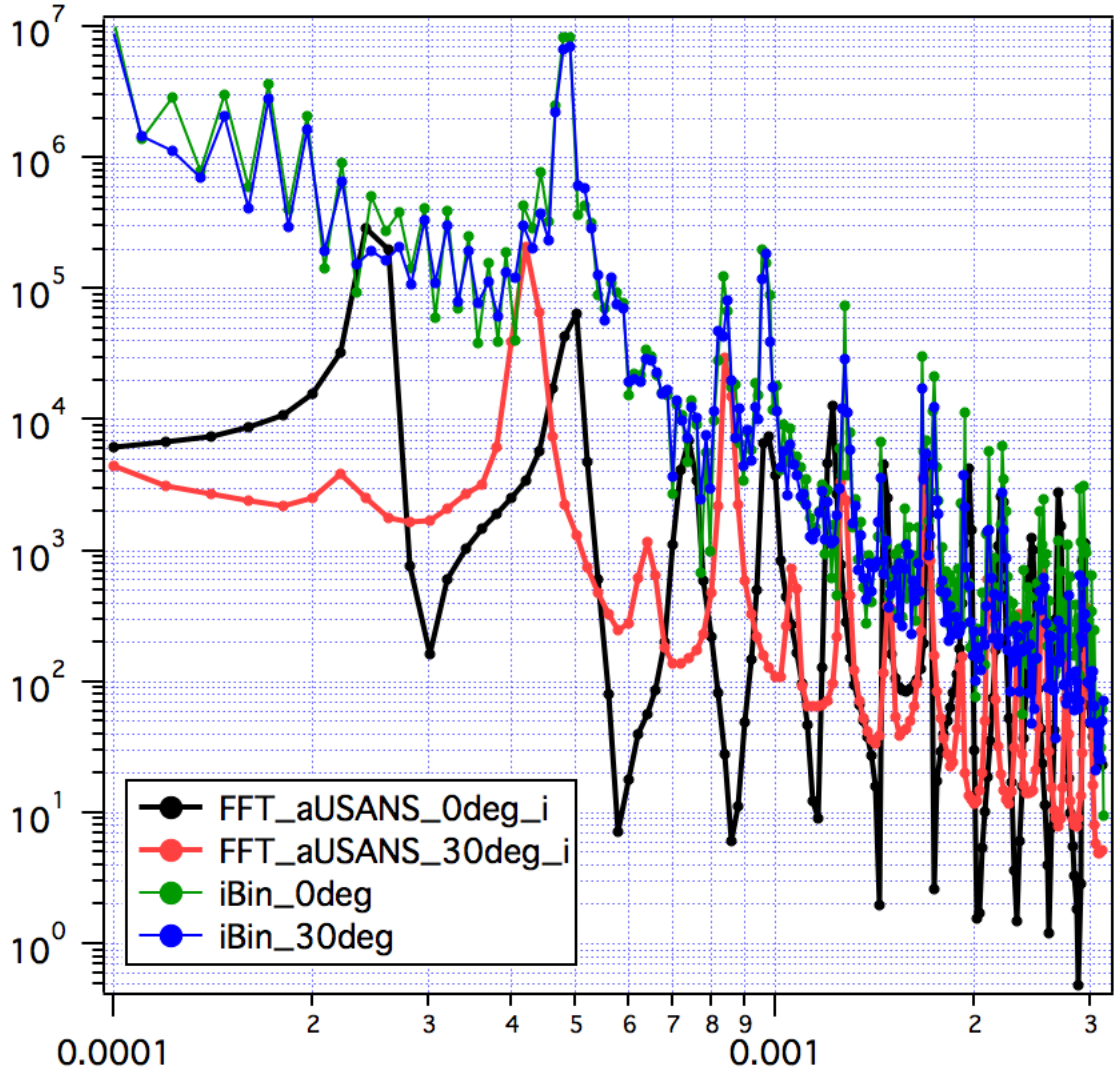
(= 30 degree rotation around the x-axis)



Now again, the isotropic data is unchanged, but the anisotrpoic data shows peaks that are not in the expected locations - but rather odd locations since the detector "sees" vertical "strips" of the scattering.

Plotting the 1D results for the anisotropic and isotropic cases highlight how the hexagonal peaks "move" when the grid is rotated. The full, isotropic (unsmeared) result is also plotted, showing that the rotation has no effect on the result. For the isotropic USANS result, the result is that there is no difference for the different rotations. The 30 degree rotation is slightly lower in intensity since some of the cylinders have been clipped, lowering the occupancy.


FFT_iUSANS_0deg_i
FFT_iUSANS_30deg_i
iBin_0deg
iBin_30deg
0.0001
0.001



Peak positions - as read directly from the graph. I haven't indexed the peaks, but it is clear that for the oriented sample that there are extra peaks that show up at different locations than the isotropic scattering, and most importantly, a peak (or two) appear at lower q values than the "primary" Qo position. This could lead to some very confusing interpretation if the first peak is, say, interpreted as a shift to lower Q of the primary peak of the hexagonal structure.

Point	unsm_peak	USANS_iso_peak	USANS_aniso_0deg_peak	USANS_aniso_30deg_peak
0				0.00022
1			0.00025	0.00042
2	0.00048	0.00048	0.0005	0.00064
3	0.000847	0.00084	0.00074	0.00084
4	0.000969	0.00096	0.00096	0.00106
5	0.00128	0.00128	0.00122	0.00126
6	0.00145	0.00144	0.00146	0.00148
7	0.00167	0.00168	0.0017	0.0017
8	0.00174	0.00176	0.00172	0.00172
9	0.00193	0.00194	0.00196	0.00192

- **Testing of the FFT of Core-Shell Spheres**

Testing routines are in FFT_vs_N_Tests.ipf
(as of Jan 2011)

Calculate both the FFT and the model calculation of core-shell spheres in 1D.

Cases:

Scale = Occupied volume fraction
R = 50 A
t = 30 A (so the outer radius is 50+30=80 A
bkg = 0

core SLD	0	1	1	2	3	3	3
shell SLD	1	2	3	1	1	3	-1
solvent SLD	0	0	0	0	0	0	0
fileLabel	1	2	3	4	5	6	7

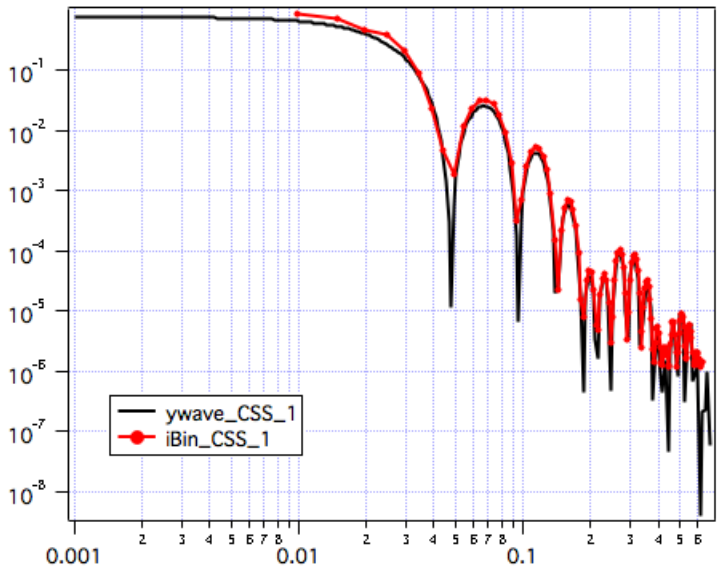
FFT_N = 128
FFT_T = 5 A

phi = 0.00814 (and is the same for all except #1)
phi = 0.00615 (for case #1)

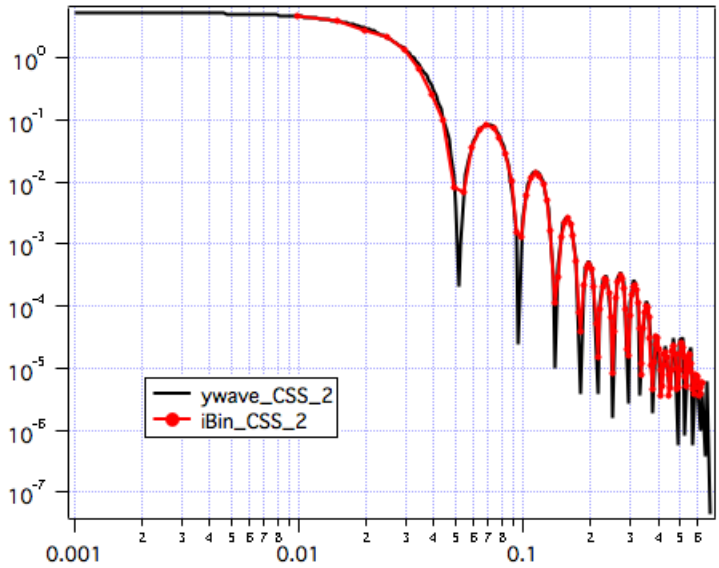
files are labeled ywave_CSS_x (for the model)
files are labeled iBin_CSS_x (for the FFT)

The "fast" Debye method cannot be used for these cases since it is under the assumption of a single SLD. The longer, full calculation can be done. The Debye calculation is scaled to a volume fraction of one, so multiplying by the volume fraction gives the correct result. The full debye calculation is shown for case #7 only. Note that for the 128^3 FFT, the time required = 1 second, while the full Debye calculation requires 140 seconds (using all 4 processors).

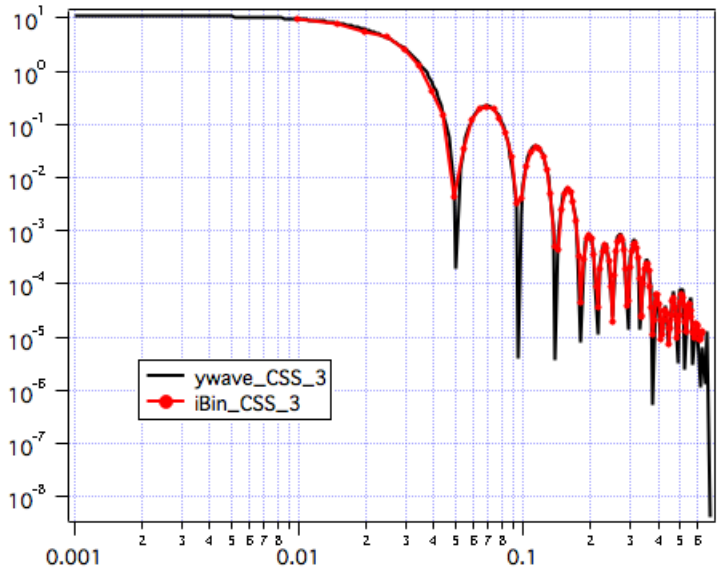
For case #1: The FFT is 33% higher due to the FFT only counting the shell as part of the volume fraction, while the model calculation counts based on the total radius. This leads to a different number density, but is easily corrected for in practice.



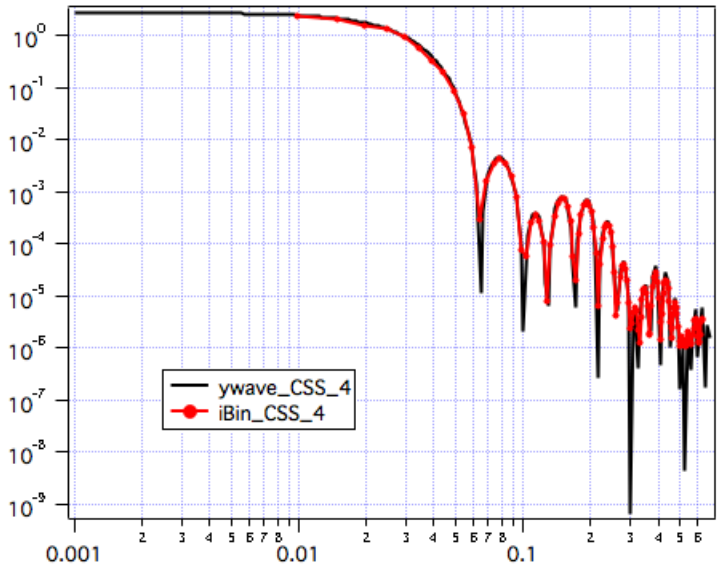
Case #2:


ywave_CSS_2
iBin_CSS_2
0.001
0.01
0.1

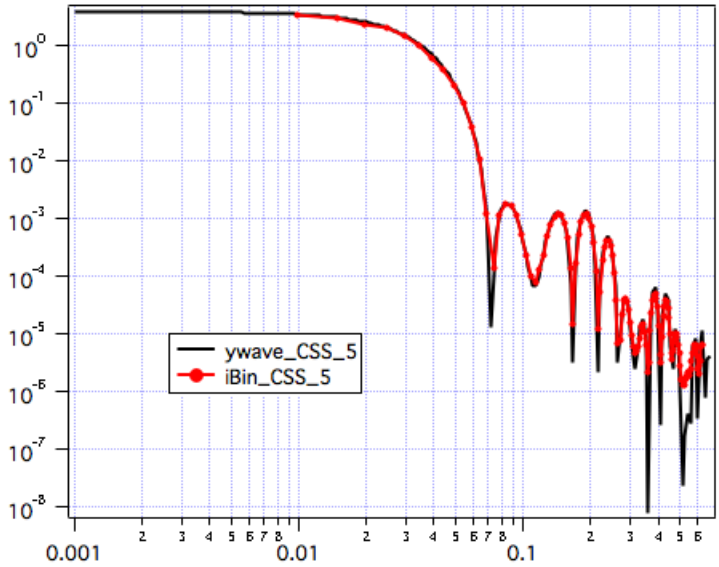
Case #3:


ywave_CSS_3
iBin_CSS_3
0.001
0.01
0.1

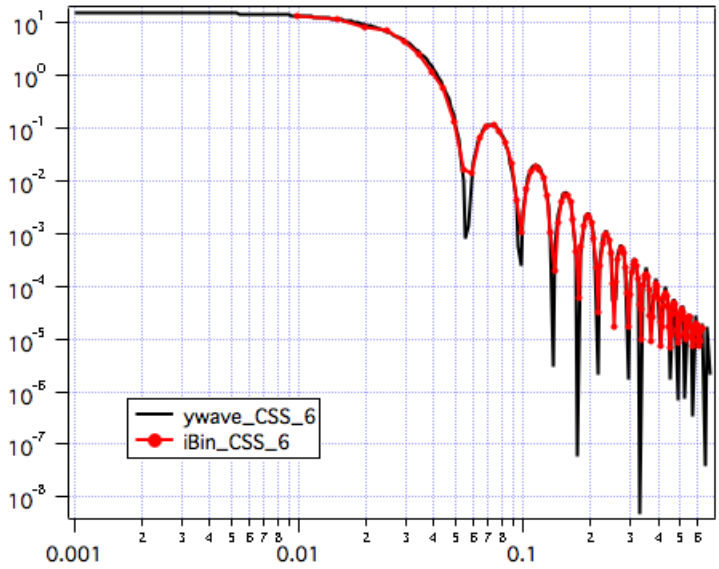
Case #4:


ywave_CSS_4
iBin_CSS_4
0.001
0.01
0.1

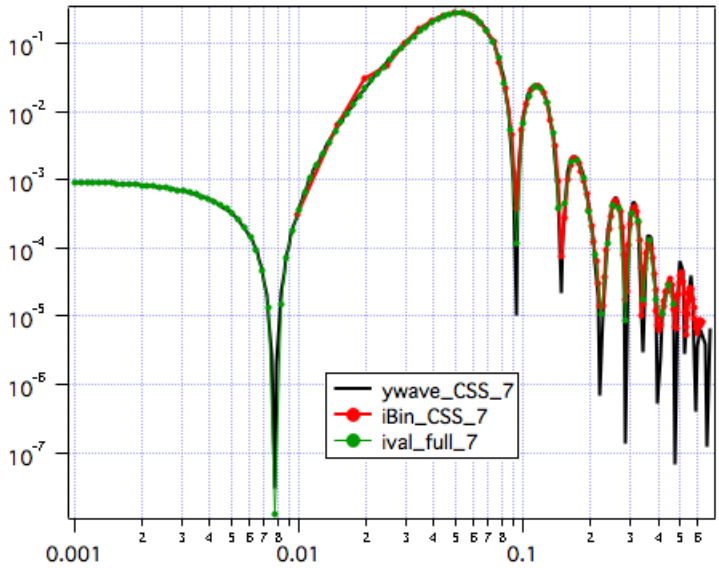
Case #5:



Case #6:



Case #7:



- **Testing of FFT With Different Solvent SLD**

  Testing routines are in FFT_vs_N_Tests.ipf
  (as of Jan 2011)

  Calculate both the FFT and the model calculation of core-shell spheres in 1D. Change the solvent SLD to be sure that scaling, both relative and absolute, is correct. Core and shell are kept at the same SLD for the first set of tests, so it is as a sphere.

  For all FFT calculations:
  FFT_T = 5 A

  Cases:
  Scale = Occupied volume fraction
  R = 50 A
  t = 30 A (so the outer radius is 50+30=80 A
  bkg = 0

core SLD	1	2	2
shell SLD	1	2	2
solvent SLD	0	1	1
fileLabel	1	2	3
FFT_N	128	128	64

  And a second set where the SLD multiplier is 10^-7, meaning that 2 digits are used for the voxels, rather than just one. Note that 3 digits can't be used since the matrix is byte. Here the core and shell SLDs are different, enabling more complex patterns to be calculated.

  root:FFT_delRho = 1e-7 //multiplier for SLD

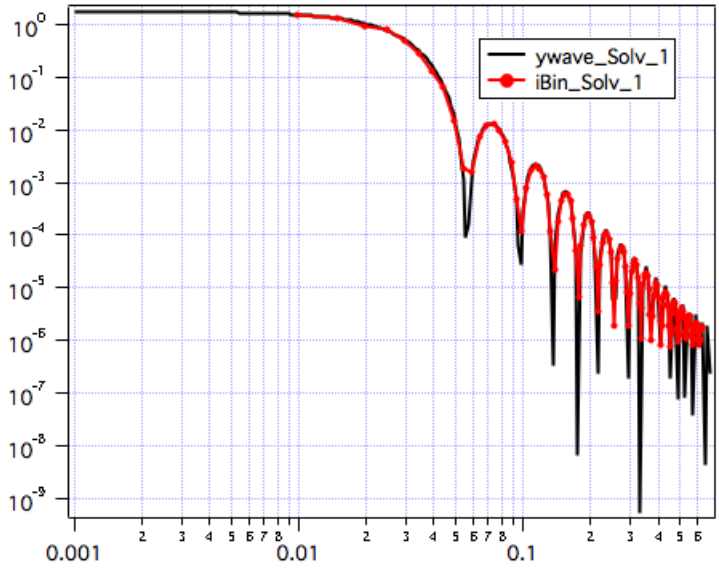
core SLD	32
shell SLD	-10
solvent SLD	0
fileLabel	4
FFT_N	128

files are labeled ywave_Solv_x (for the model)
files are labeled iBin_Solv_x (for the FFT)

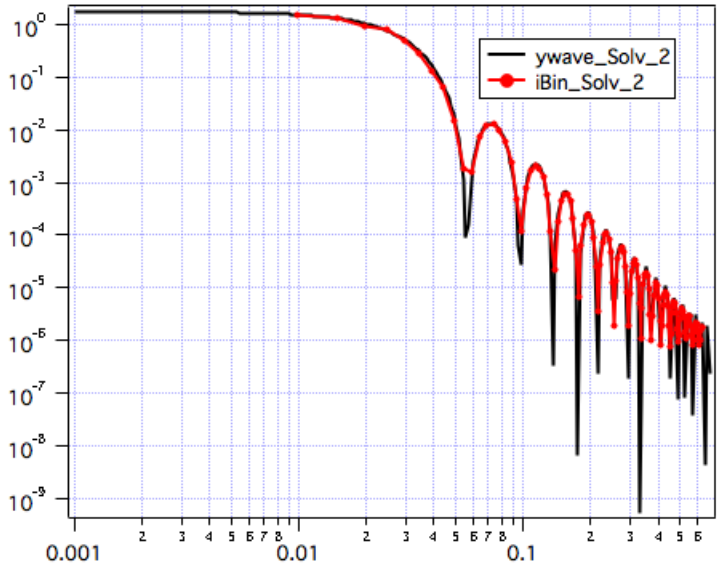
The "fast" Debye method cannot be used for these cases since it is under the assumption of a single SLD. The longer, full calculation can be done. The Debye calculation is scaled to a volume fraction of one, so multiplying by the volume fraction gives the correct result. The full debye calculation is shown for case #7 only. Note that for the 128^3 FFT, the time required = 1 second, while the full Debye calculation requires 140 seconds (using all 4 processors).

Cases 1,2,3 all have the same absolute intensity, as they should:

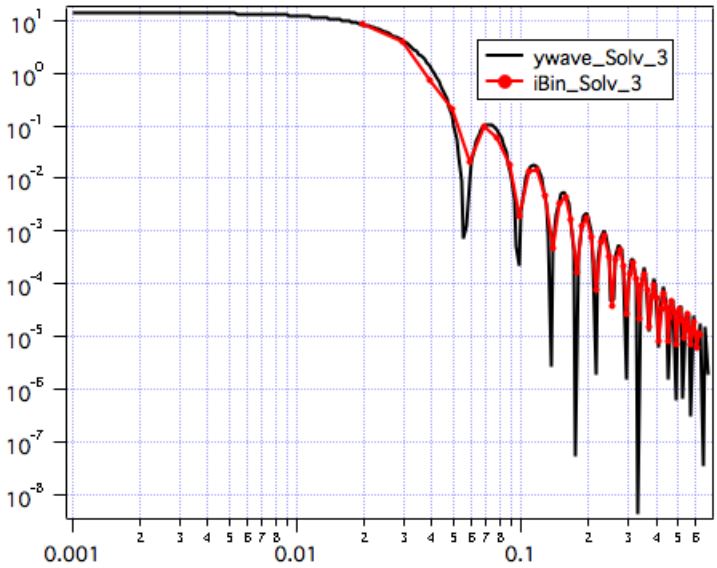
Case #1:



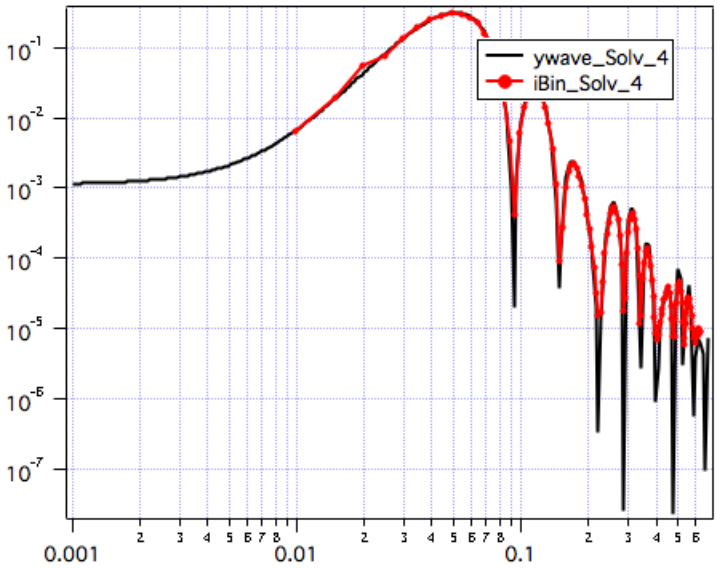
Case #2:



Case #3:



And case #4 correctly calculates the scattering for the odd contrast case where there is a peak in the core-shell scattering:



- **Verification of FFT Scaling**

  24 JAN 2011

  ****
  I(q) [cm^-1] = (FFT result) * phi *(1/Vp)* (1e-6)^2*(1e8)*Tedge^6
  ****
  phi = (number of occupied voxels) / Nside^3
  1/Vp = 1/(total particle volume) = 1/(number occupied * Tedge^3)
  The number of occupied voxels is everything that is not solvent.
  The (1e-6)^2 factor converts the single digit of voxel SLD to correct units.
  The 1e8 factor converts the units of (1/A) to (1/cm)
  Tedge^6 accounts for the volume fer voxel, squared.

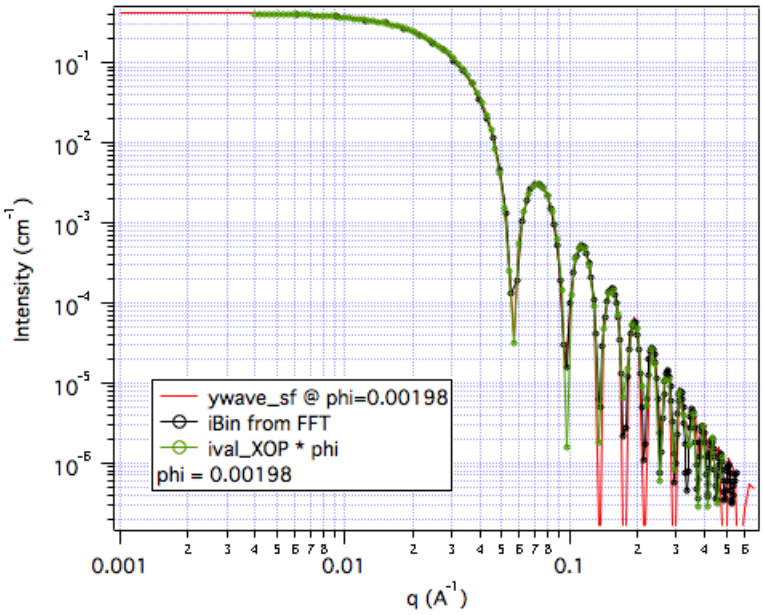
  So it reduces down to:
  I(q) [cm^-1] = (FFT result) * (1e-6)^2*(1e8)*(Tedge/Nside)^3

Nside is the number of cells per edge
Tedge is the size per voxel edge

Using this scaling of the FFT result, the values are scaled to the volume fraction of particles present - the Debye method scales the data to a volume fraction of one. The case below is for:

Point	parameters_sf	coef_sf
0	scale	1
1	Radius (A)	80
2	SLD sphere (A-2)	1e-06
3	SLD solvent (A-2)	2e-06
4	bkgd (cm-1)	0

The FFT parameters are Nside = 180, Tedge = 5.7, and an 80 A radius sphere (np = 11557). Even with these non-standard FFT parameters, the scaling is correct.



- **FFT of Concentrated Hard Spheres**

JAN 2011

1) FFT of randomly placed spheres are repeated multiple times and averaged to identify a reasonable number of passes needed.

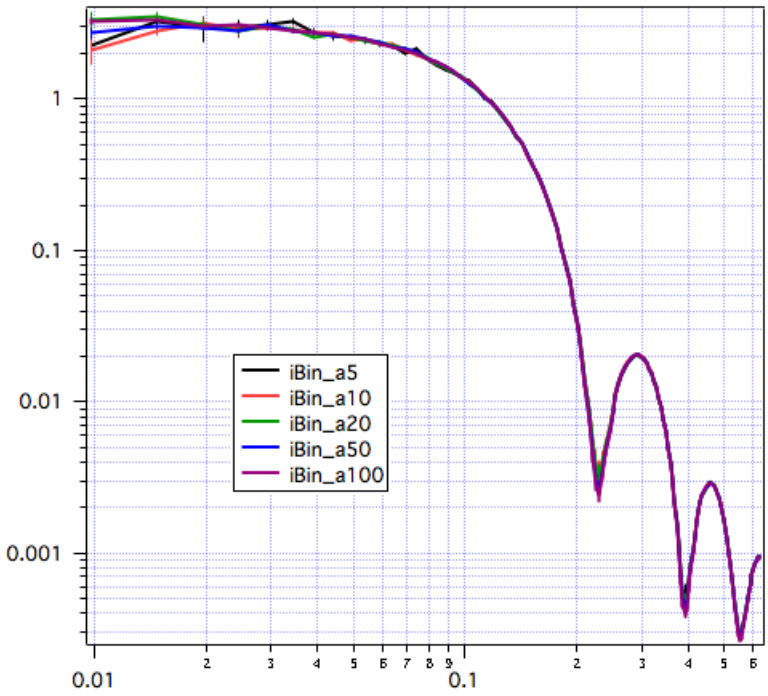
2) Spheres are placed in a volume based on the simple critera of no overlap between spheres.

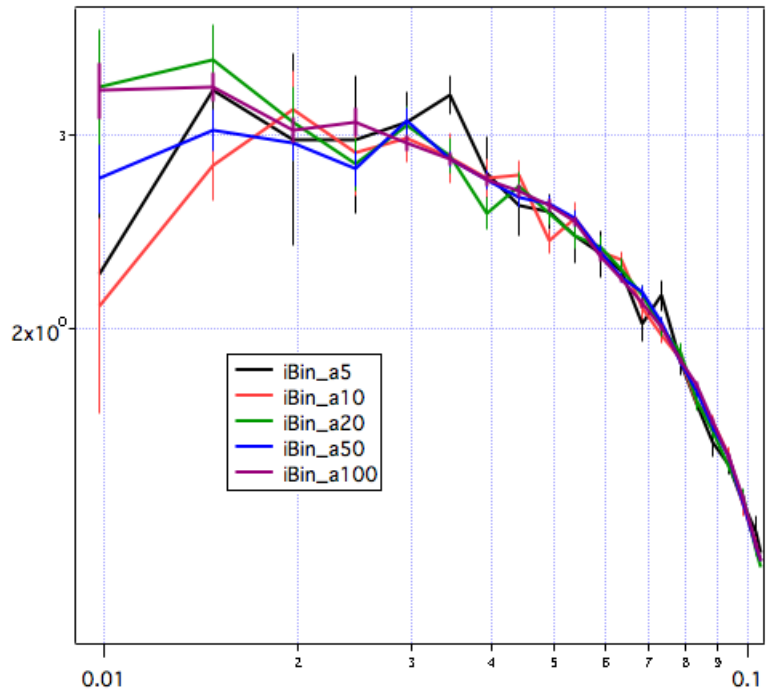
N=128
T=5 A
R sphere = 20 A
no polydispersity
then re-run with 25% polydispersity

The matrix was filled randomly and the FFT performed. The erase/fill/FFT was repeated 20 times and the results were averaged.

20 passes was chosen as a compromise of quality and speed. The results below are for a volume fraction of 0.1% (only 10 spheres) and the number of passes as shown in the legend (5, 10, 20, 50, or 100). The second graph is a zoom in of the low q region. It is expected that the higher concentrations would have a similar smoothing as a function of the number of passes. Error bars are calculated in the usual way, averaging over the number of passes.

For each of the cases, below is the averaged data and an image of the voxelgram, giving an idea of the occupancy at each of the different volume fractions. Each is a representative snapshot of a possible structure at that concentration.

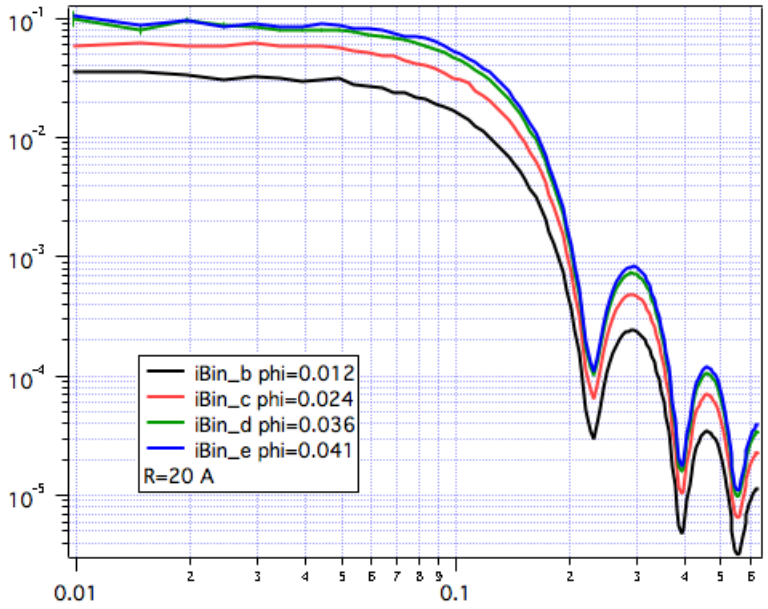


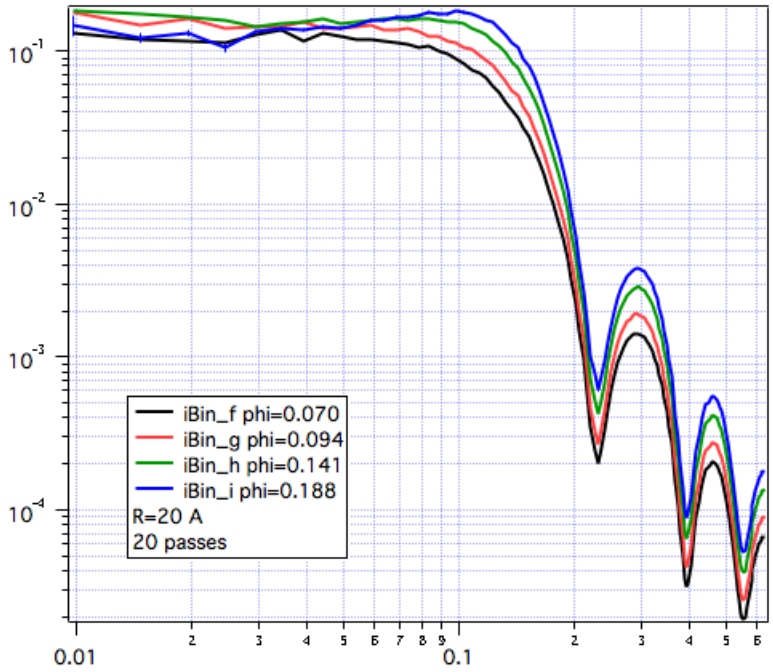


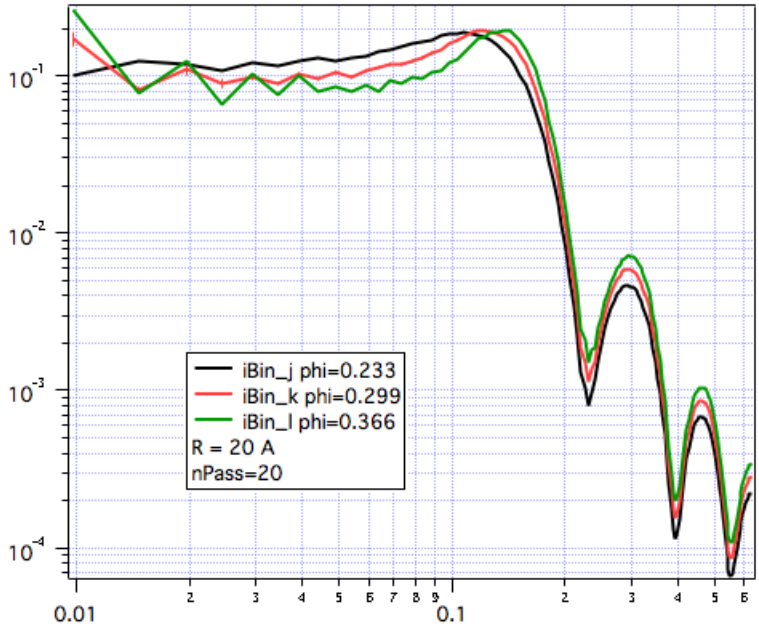
Results for the concentrated spheres

The model calculations are for hard spheres at the volume fraction as calculated by the sphere occupancy. Error bars are only shown on one set per graph. They become qhite small after even just a few averaging passes.

n_spheres	phi_spheres	file_name
100	0.012	iBin_b
200	0.024	iBin_c
300	0.036	iBin_d
350	0.041	iBin_e
600	0.070	iBin_f
800	0.094	iBin_g
1200	0.141	iBin_h
1600	0.188	iBin_i
2000	0.233	iBin_j
2600	0.299	iBin_k
3200	0.366	iBin_l

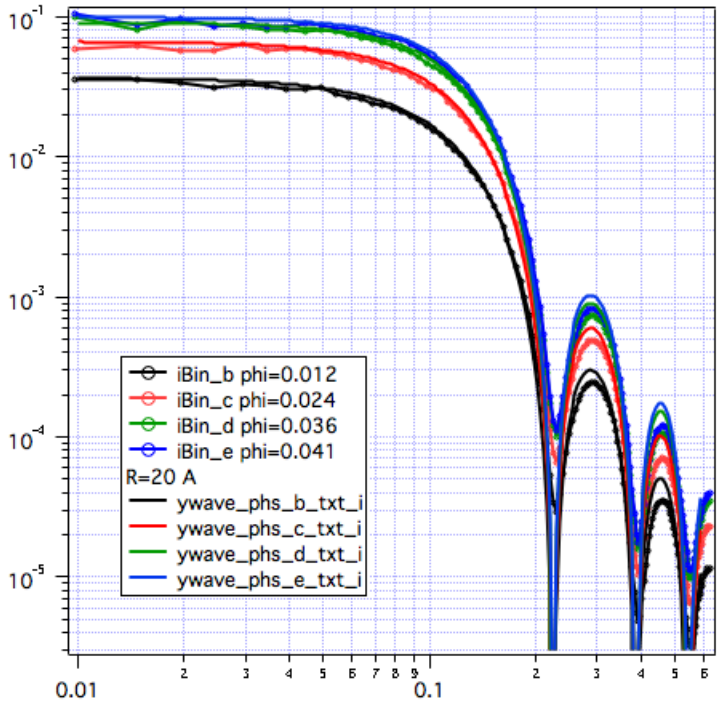




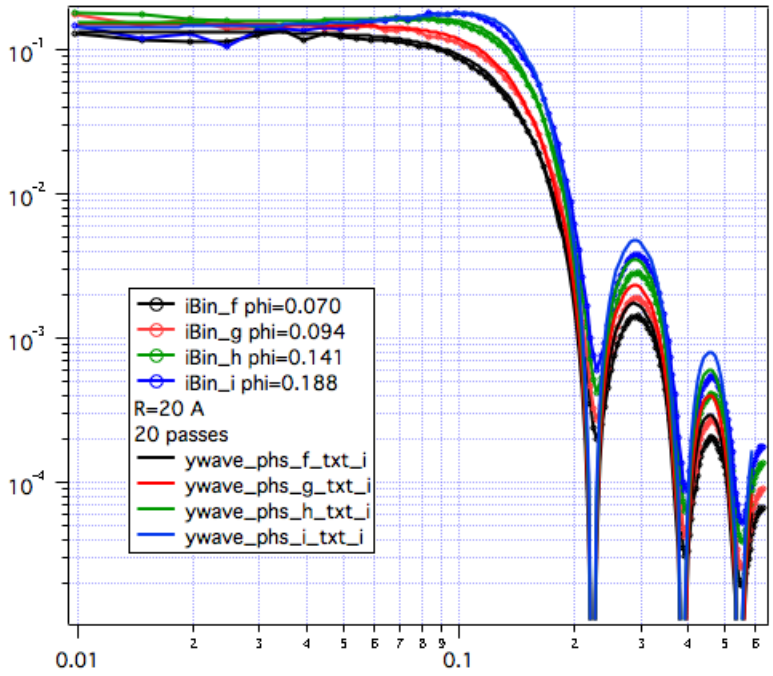


Calculating versus the Polydisperse Hard Sphere model, using the volume fraction of spheres as simulated. The polydispersity is set to pd = 0.001

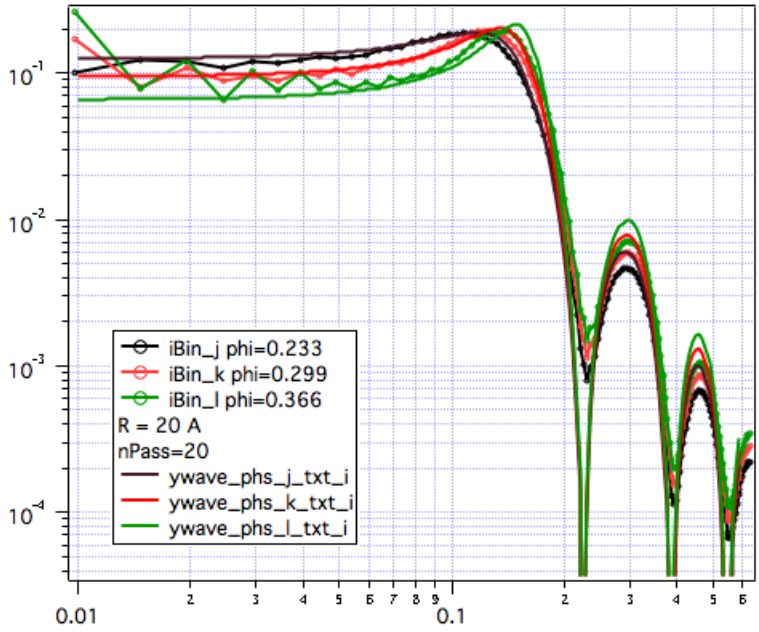
Phi = 0.012
Phi = 0.024
Phi = 0.036
Phi = 0.041



Phi = 0.070
Phi = 0.094
Phi = 0.141
Phi = 0.188

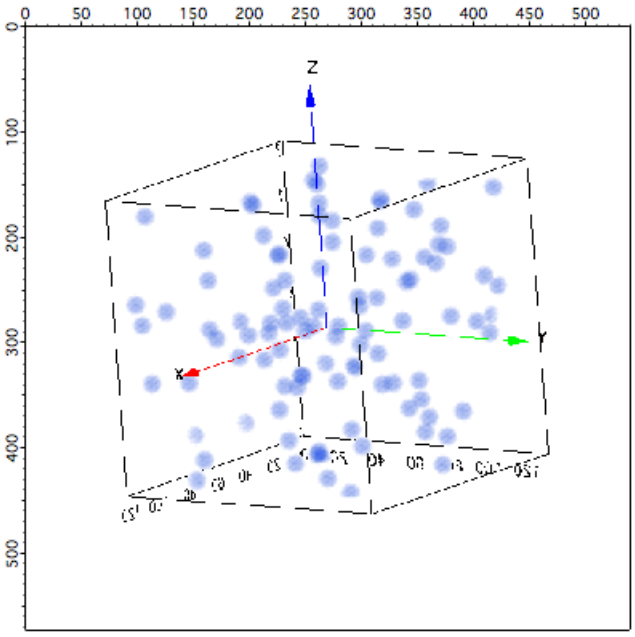


Phi = 0.233
Phi = 0.299
Phi = 0.366

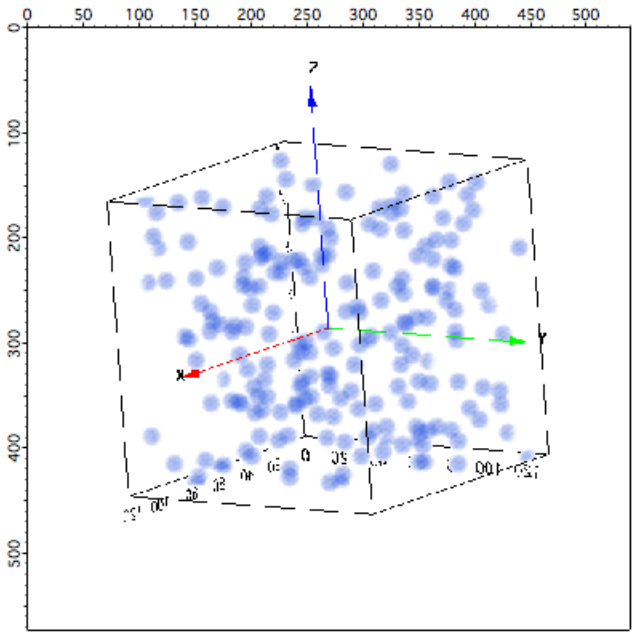


And the representative snapshots of the filled matrix are:

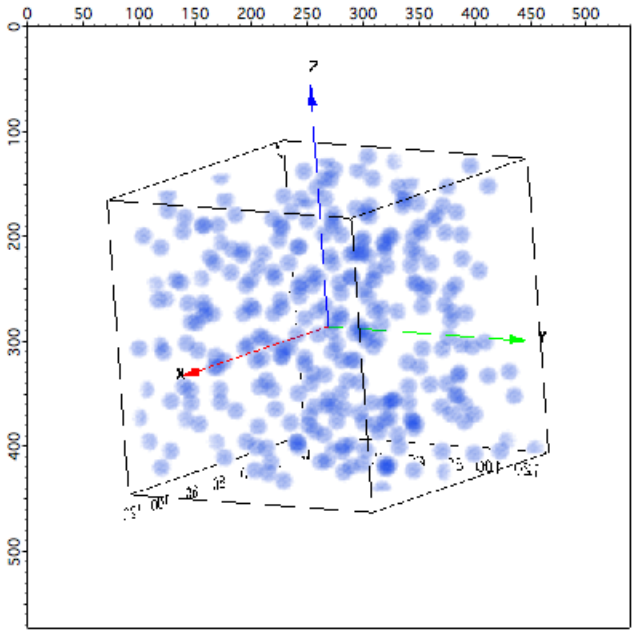
Phi = 0.012



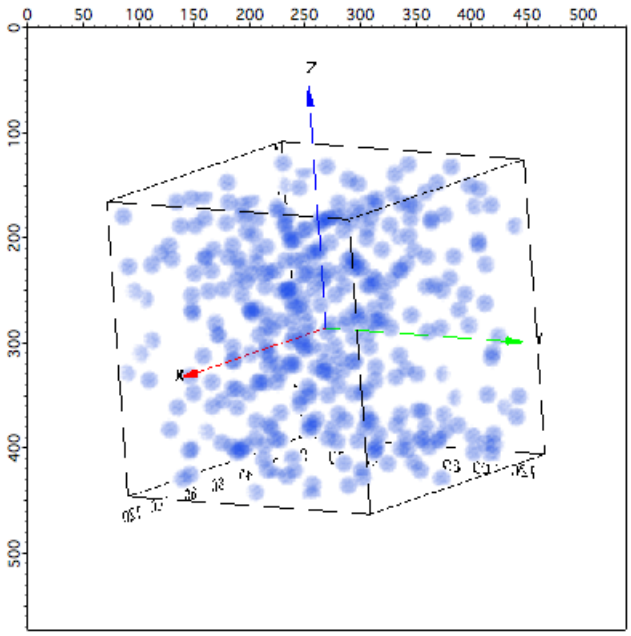
Phi = 0.024



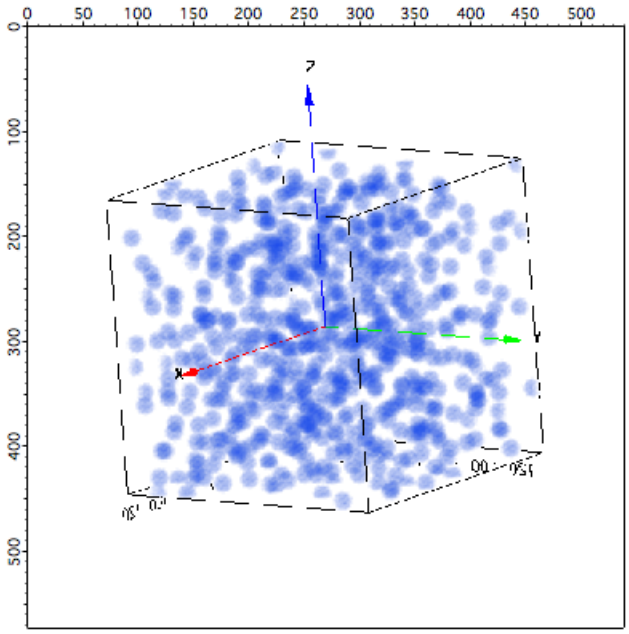
Phi = 0.036



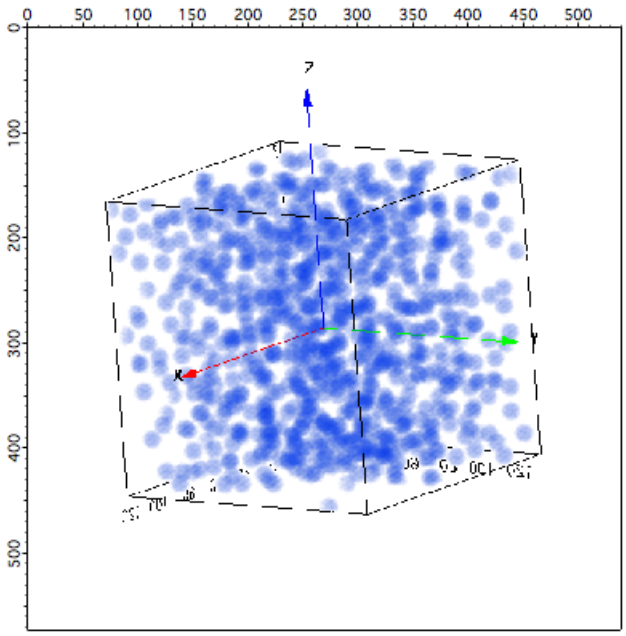
Phi = 0.041



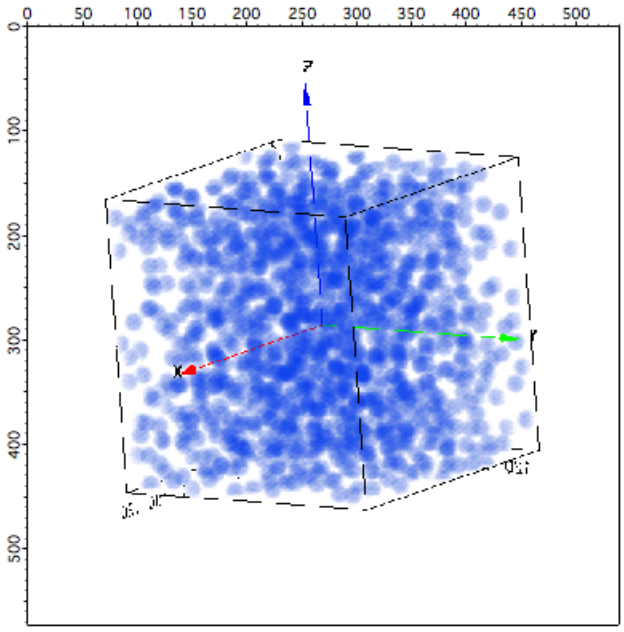
Phi = 0.070



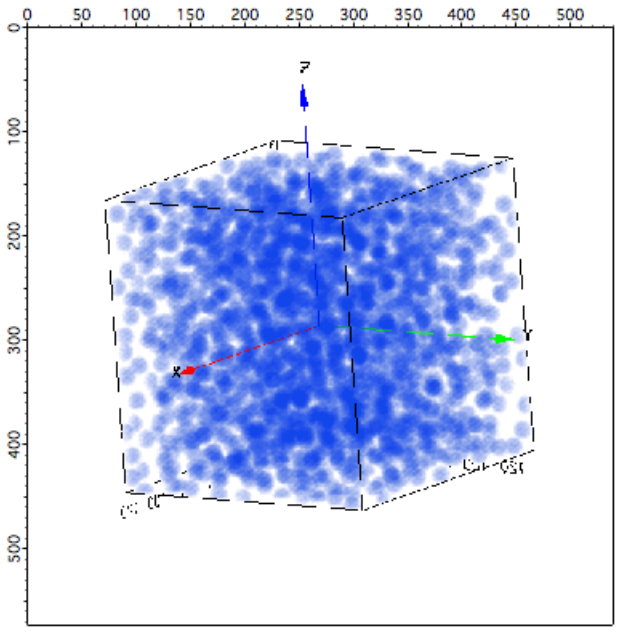
Phi = 0.094



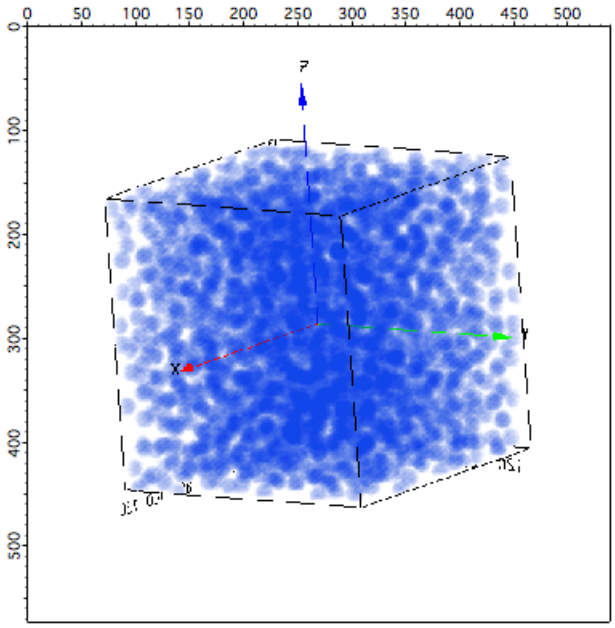
Phi = 0.141



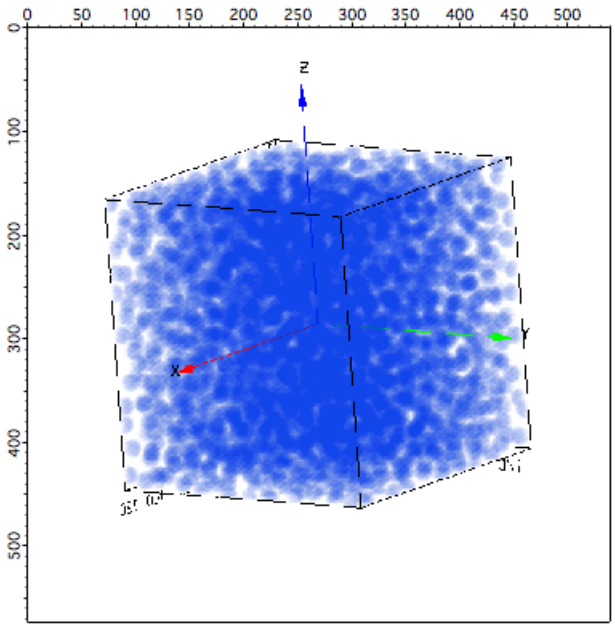
Phi = 0.188



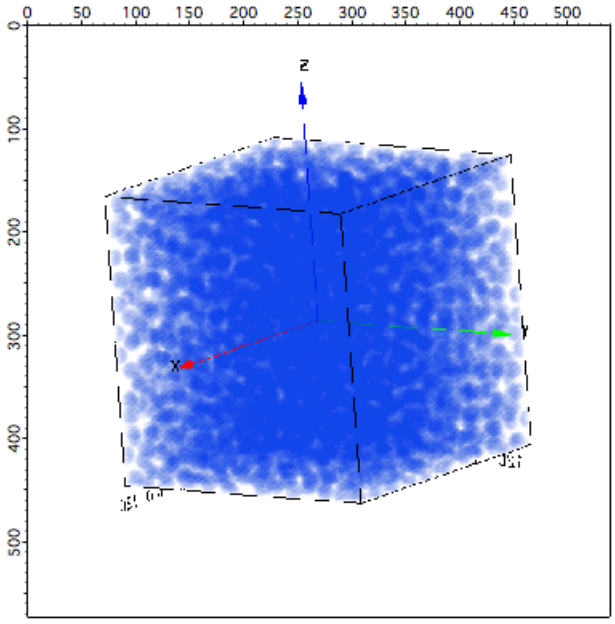
Phi = 0.233



Phi = 0.299



Phi = 0.366



Polydispersity of 25% in the sphere radius:

This may not translate terribly well, since the radius is 20 and the edge is 5, so there are not very many voxels per sphere, but I expect that there enough variation in the sphere radius to show the proper effects in the final FFT result, especially when averaging over multiple configurations.

One discrepancy that showed up in one of the first attempts is that the polydisperse volume fraction is lower than the monodisperse volume fraction for the same number of spheres. This was almost certainly a deficiency in the algoroithm that places the spheres. When the box is starting to get full - the larger spheres in the distribtution are more likely to "fail" on placement. Then the algorithm picks a

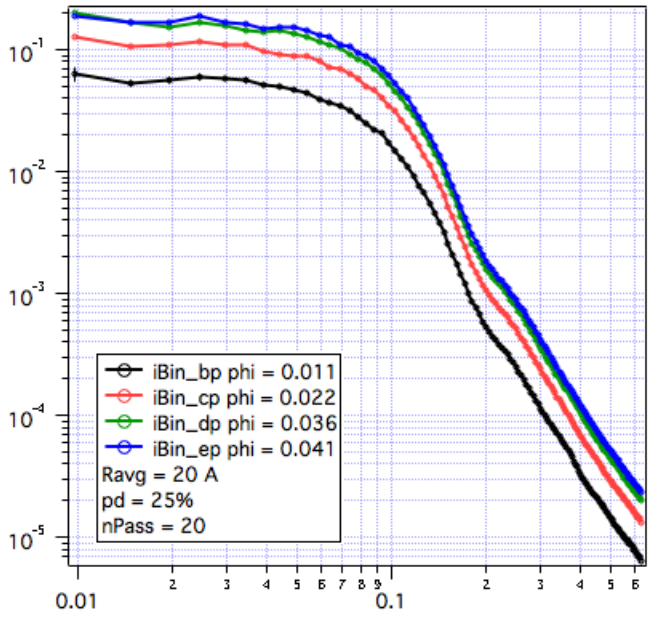
new trial location, and a new sphere radius.

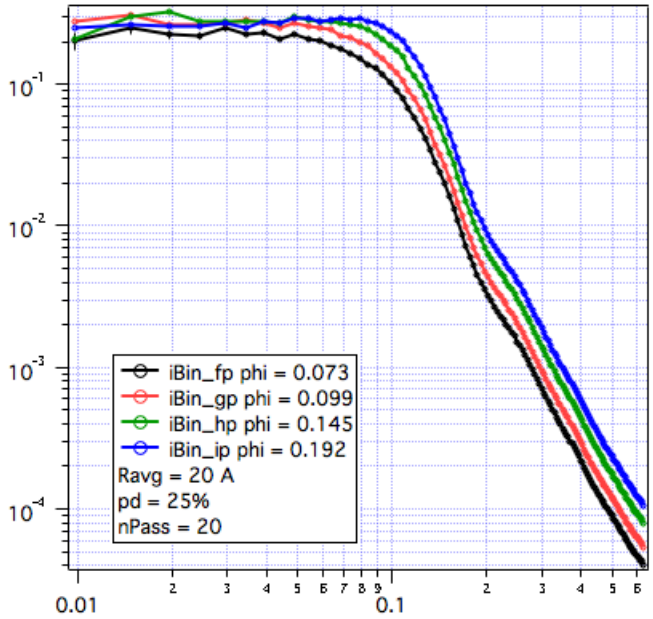
Model calculations here are for the volume fraction as calculated from the occupancy, using the exact polydisperse hard sphere model calculation of Griffiths. Note that the calculation of Griffiths uses Schulz polydispersity, which should not be too much different than a Gaussian polydispersity of 25%.

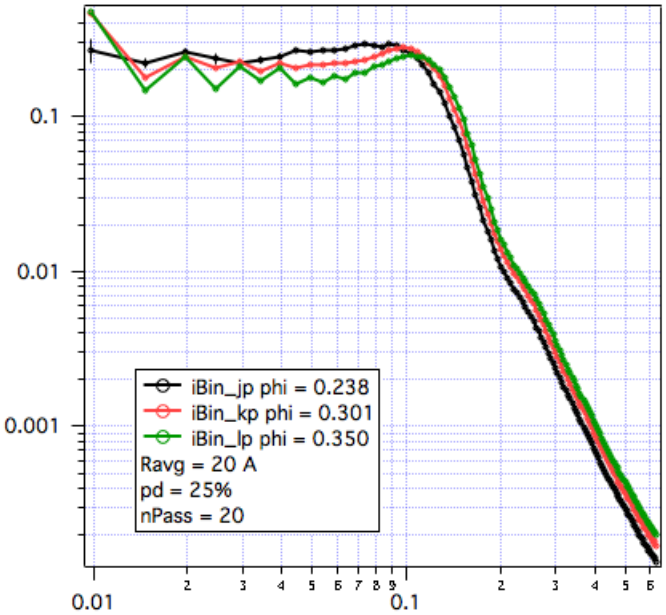
The updated algorithm that is "Better" at placing the spheres now continues to try to fit a sphere in by choosing a new point, keeping the same radius, rather than discarding it. Even so, at high volume fractions (> 0.2), a larger radius trial will frequently fail more than 1000x. At this point, the radius is skipped (and the sphere too). The actual number of spheres placed is corrected in the wave note.

The Gaussian polydispersity has a correction factor of 1+3p^2 on the average volume. for p=0.25, 1+3p^2 = 1.187. So the number of spheres that need to be placed is somewhat smaller for the polydisperse case, as calculated below:

n_spheres	phi_spheres	file_name	n_poly_spheres	phi_polydisp	n_poly_actual
100	0.012	iBin_b	84	0.011	84
200	0.024	iBin_c	168	0.022	168
300	0.036	iBin_d	253	0.036	253
350	0.041	iBin_e	295	0.041	295
600	0.070	iBin_f	505	0.073	505
800	0.094	iBin_g	674	0.099	674
1200	0.141	iBin_h	1011	0.145	1011
1600	0.188	iBin_i	1348	0.192	1348
2000	0.233	iBin_j	1685	0.238	1683
2600	0.299	iBin_k	2190	0.301	2180
3200	0.366	iBin_l	2696	0.35	2618

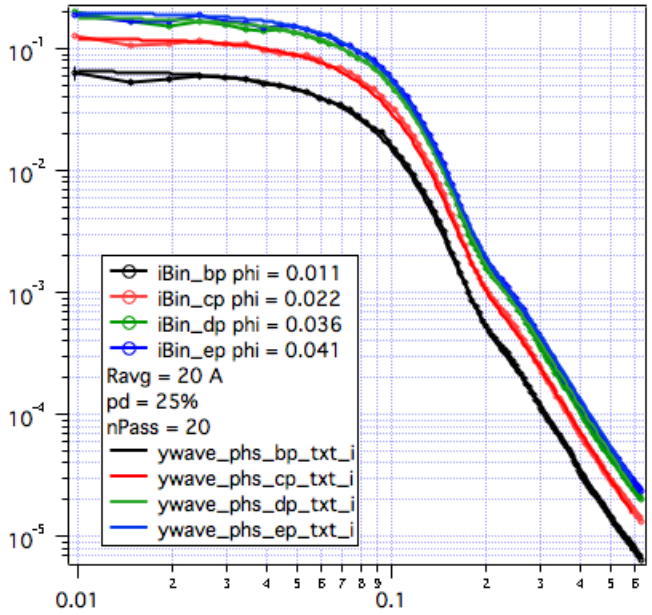




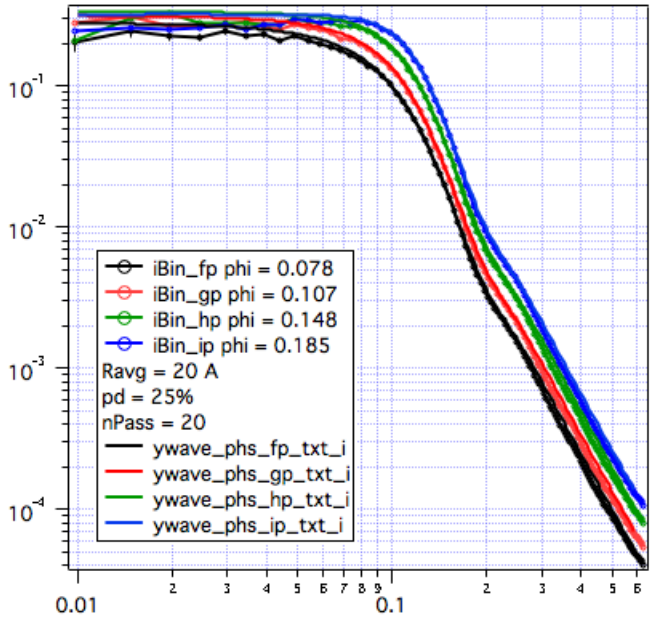


Calculating versus the Polydisperse Hard Sphere model, using the volume fraction of spheres as simulated. The polydispersity is set to pd = 0.25

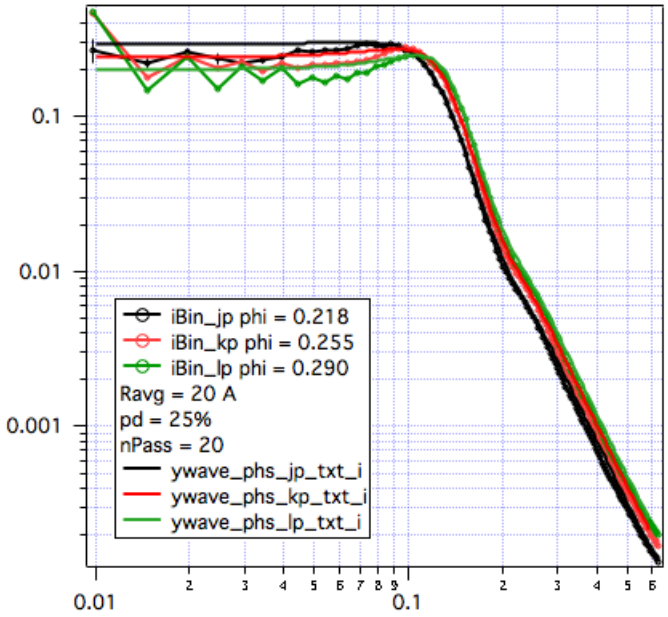
Phi = 0.015
Phi = 0.029
Phi = 0.042
Phi = 0.051



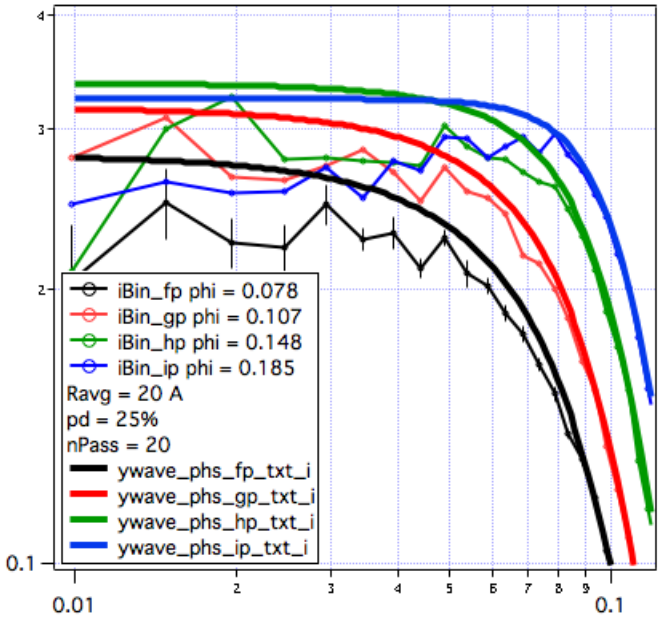
Phi = 0.078
Phi = 0.107
Phi = 0.148
Phi = 0.185

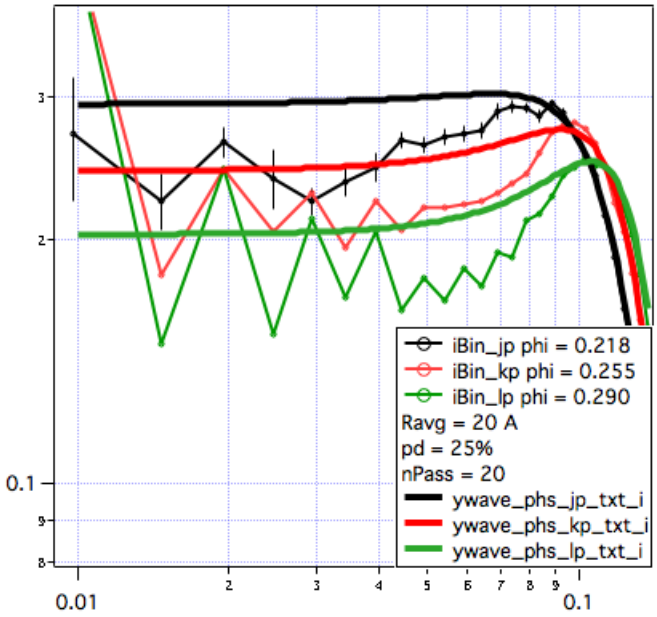


Phi = 0.218
Phi = 0.255
Phi = 0.29



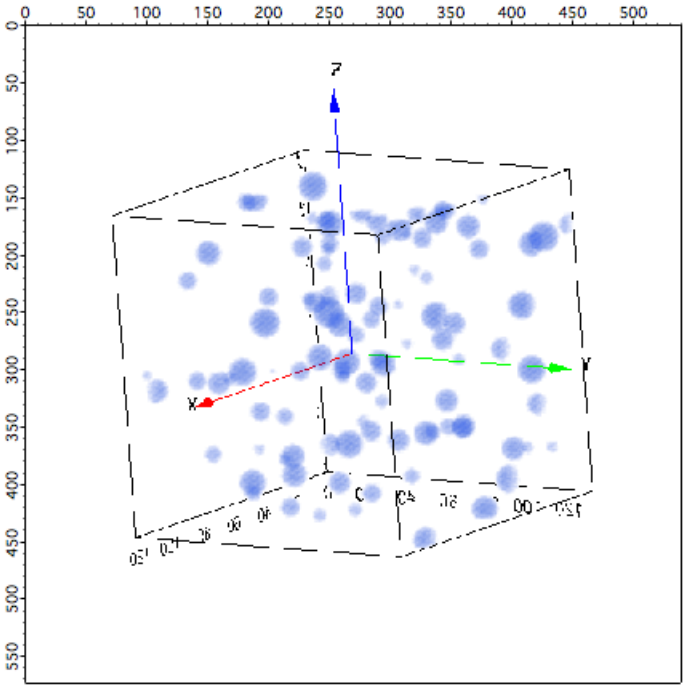
The lowest concentration set appears to be very good at both high and low q-values. Zooming in on the higher concentrations at the low q region shows where the FFT is missing something, and unable to reproduce the exact calculation. This may be due to the increasing skew towards smaller radius particles as the fill becomes more crowded. But the FFT is quite good overall.



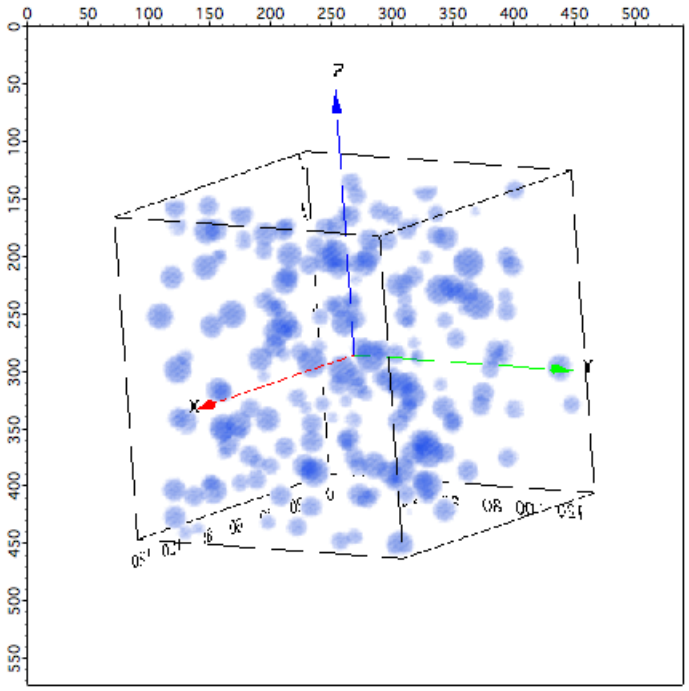


And representative snapshots of the filled matrix are:

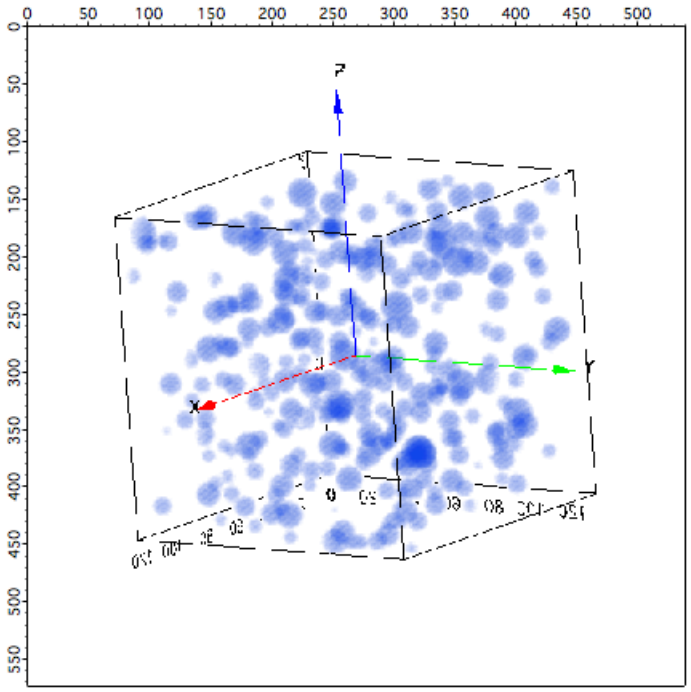
Phi = 0.011



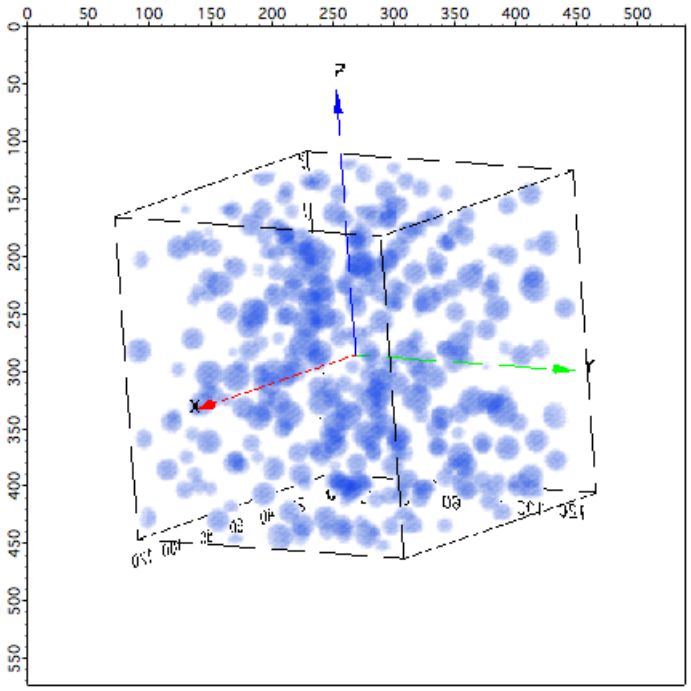
Phi = 0.022



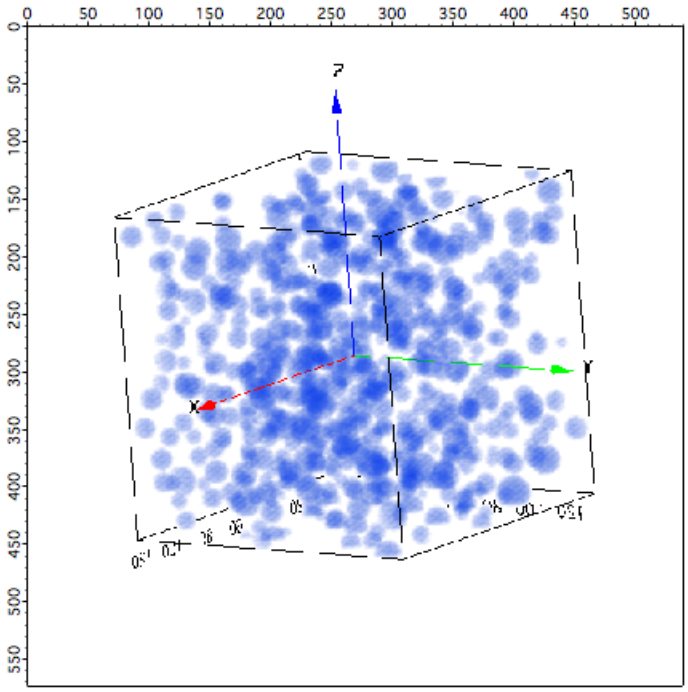
Phi = 0.036



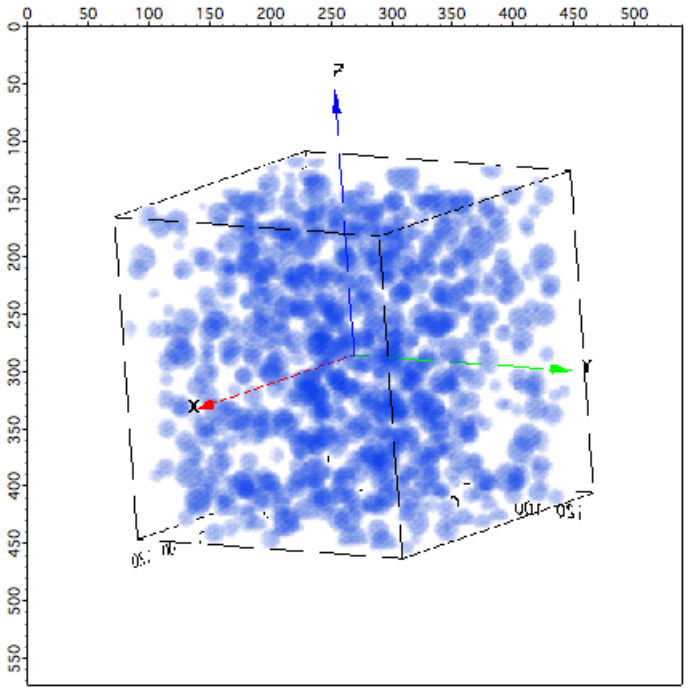
Phi = 0.041



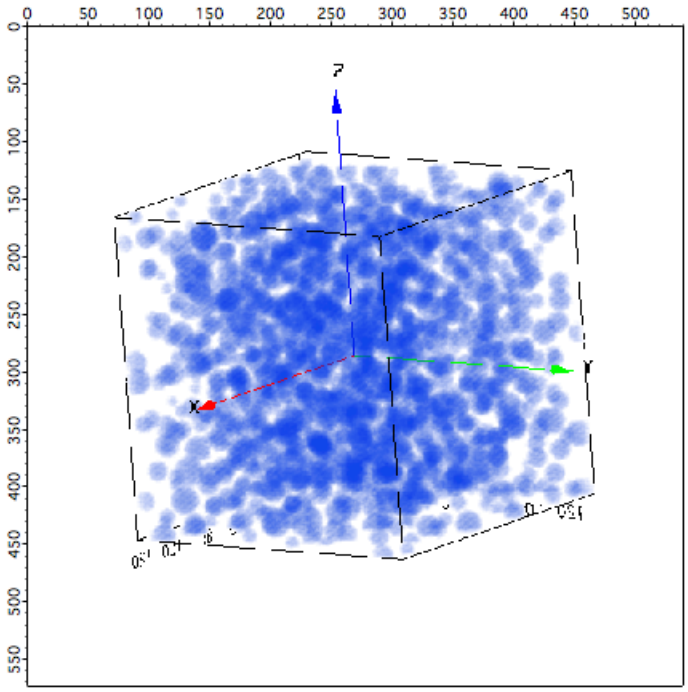
Phi = 0.073



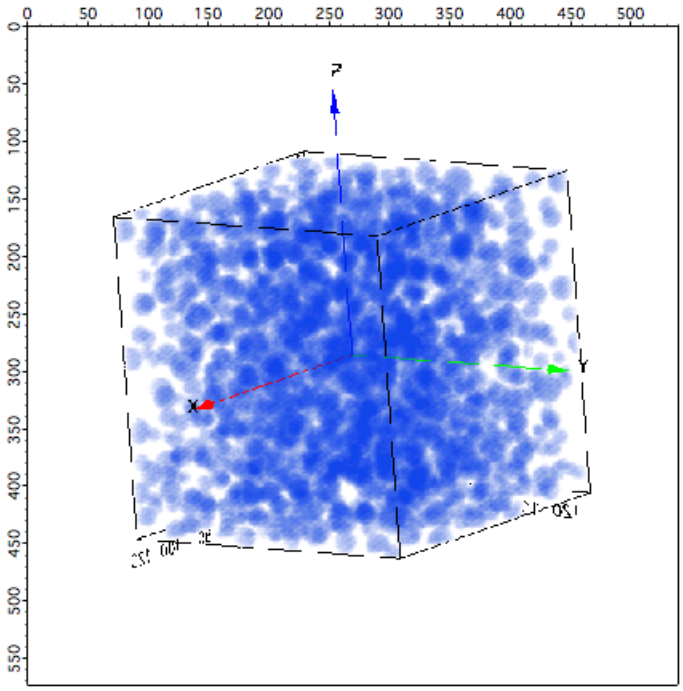
Phi = 0.099



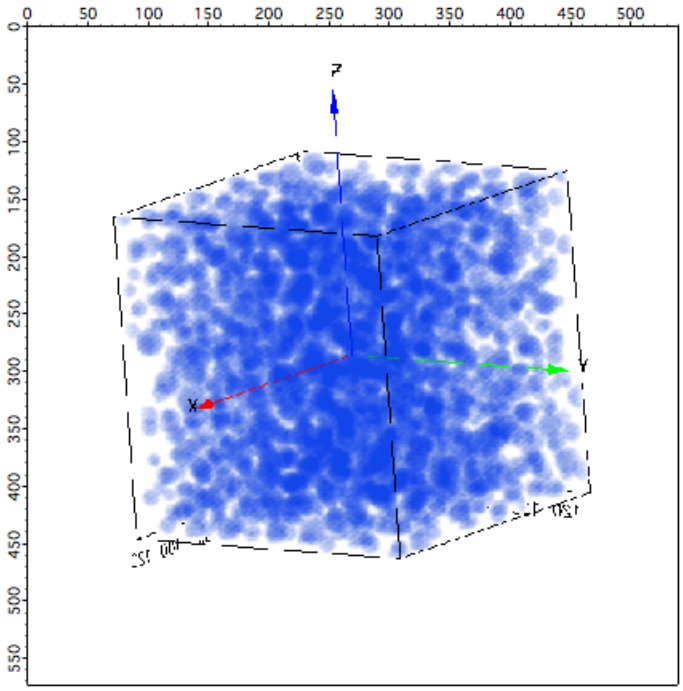
Phi = 0.145



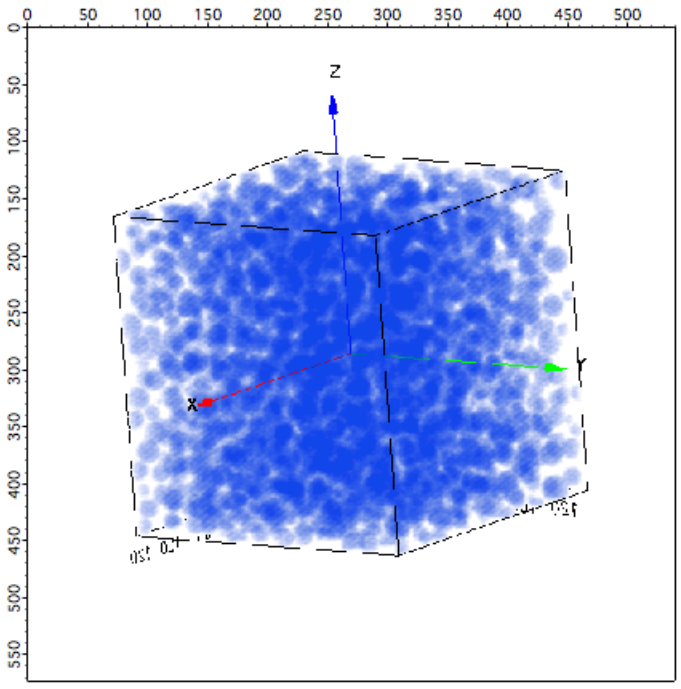
Phi = 0.192



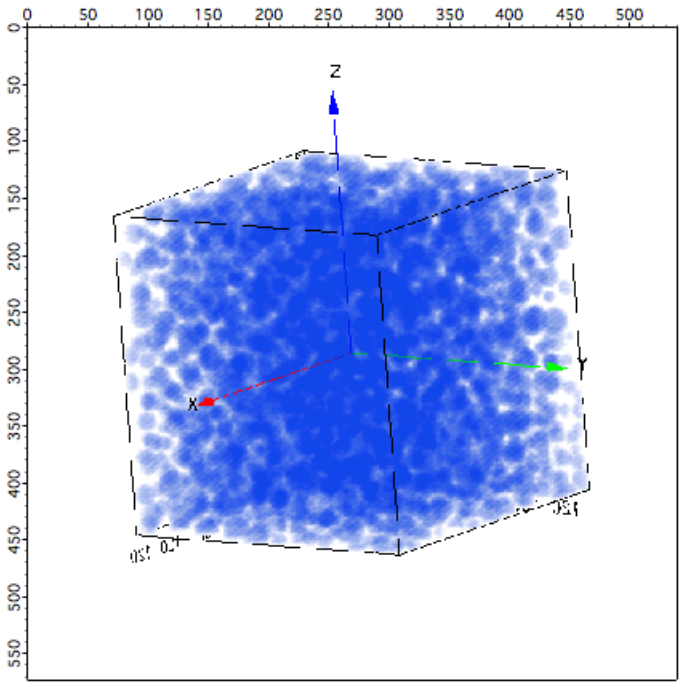
Phi = 0.238



Phi = 0.301



Phi = 0.350



- **Connected Rods**

14 JAN 2011

For this calculation, random points, or nodes, are selected in the box. Then based on Voronoi Tesselation the nodes are connected. No periodic boundary conditions are used. Two different methods of generating the nodes in the initial box are used.

1) Igor's random number generator, enoise()
2) NR's implementation of a Sobol sequence, as an XOP. Note that this is a "quasi-random" number generator that more efficiently fills N-dimensional space with points.

Other conditions:

1) The cylinder connecting the nodes can be the same diameter as the edge dimension, good for representing a larger network structure
2) the cylinder radius can be larger than the edge dimension, giving a more accurate picture of the local cylindrical structure
3) The cylinder can be polydisperse in radius, making it more realistic. Polydispersity in the length is already taken into account in the random nature of the node generation.

Things to think about:
1) Proper polydispersity of the radius. Look to the results from the polydisperse spheres for how it needs to be done.
2) How can I predict the volume fraction based on N, T, Diam, and nNodes? This is very important to be able to compare results to experimental data and to be able to join two q-ranges.
3) How can I put periodic conditions back in? Is it really necessary?
4) What are the artifacts coming from the overall dimensions of the box? Is there a "big cube" scattering from this, and how can I get rid of this?

In these calculations, boxes of different scale and different numbers of nodes have been calculated, so that the different q-ranges can be merged to give a broader q-range. Polydispersity of the radius has not been implemented yet.

Conditions:
Real Cylinder radius = 20 A
N=256
T= 40
(nodes)
(40,100,200,300,400,500)

N=256
T=5
(nodes)
(40,100,200,300,400,500) = (very high volume fractions)
(6,7,9,10,20,30) = (more reasonable volume fractions)

These sets of calculations have been repeated twice, once with each random number generator. This gives a total of 24 data files.

Files are:

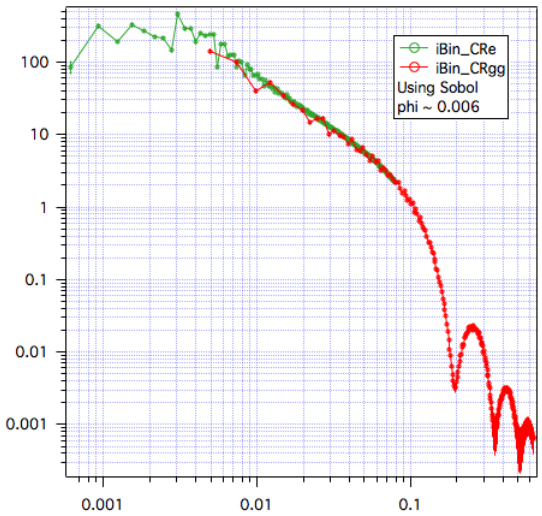
iBin_CRa	intensity
qBin_CRa	q values
eBin_CRa	error
p_connRod_a	2D wave of image of Gizmo

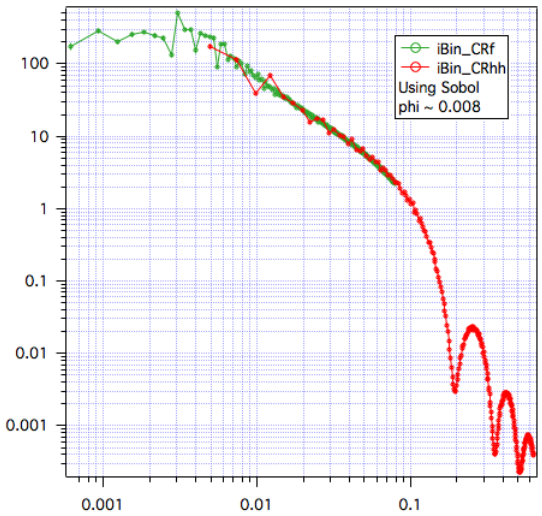
There is a wave note on the intensity with the conditions used, most notably the volume fraction occupied. This must be matched between the two q-ranges. I didn't see any quick calculation that would predict the number of nodes in each range needed for an equivalent volume fraction, but I might be able to find an average relation. The random nature of the fills should come out in the wash by averaging enough replicate fills.

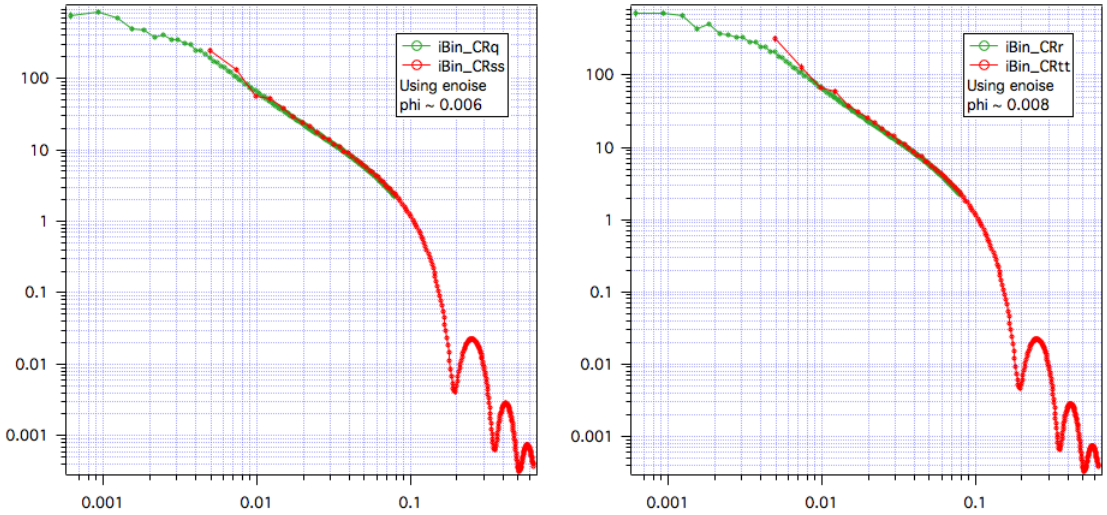
Point	phi_rods	suffix	T_edge	N_nodes	random_type
0	0.00112303	a	40	40	Sobol
1	0.00241653	b	40	100	Sobol
2	0.00410999	c	40	200	Sobol
3	0.00556325	d	40	300	Sobol
4	0.00682805	e	40	400	Sobol
5	0.00803283	f	40	500	Sobol
6	0.0633498	g	5	40	Sobol
7	0.129674	h	5	100	Sobol
8	0.208443	i	5	200	Sobol
9	0.267729	j	5	300	Sobol
10	0.317015	k	5	400	Sobol
11	0.358516	l	5	500	Sobol
12	0.00108832	m	40	40	enoise
13	0.00237708	n	40	100	enoise
14	0.00400404	o	40	200	enoise
15	0.00532799	p	40	300	enoise
16	0.00654989	q	40	400	enoise
17	0.00758886	r	40	500	enoise
18	0.0613427	s	5	40	enoise
19	0.125816	t	5	100	enoise
20	0.198077	u	5	200	enoise
21	0.256339	v	5	300	enoise
22	0.301276	w	5	400	enoise
23	0.341836	x	5	500	enoise
24	0.00616066	gg	5	6	Sobol
25	0.00840739	hh	5	7	Sobol
26	0.0118726	ii	5	9	Sobol
27	0.0138887	jj	5	10	Sobol
28	0.032203	kk	5	20	Sobol
29	0.0491666	ll	5	30	Sobol
30	0.00633413	ss	5	6	enoise
31	0.00819941	tt	5	7	enoise
32	0.0113905	uu	5	9	enoise
33	0.0132645	vv	5	10	enoise
34	0.0313048	ww	5	20	enoise
35	0.0456206	xx	5	30	enoise

Pairs of files to plot together:
Sobol
0.0068 (e) and 0.0062 (gg)
0.0080 (f) and 0.0082 (hh)

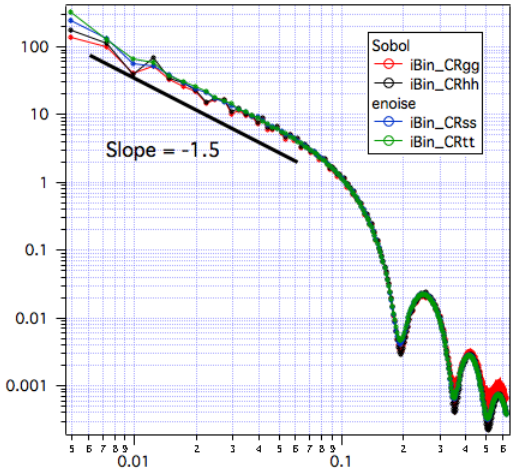
enoise
0.0065 (q) and 0.0063 (ss)
0.0076 (r) and 0.0082 (tt)

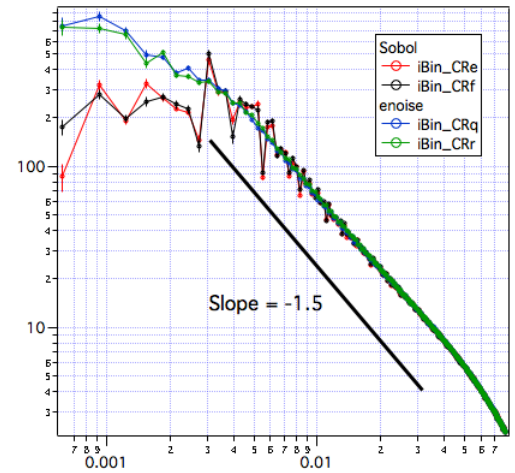




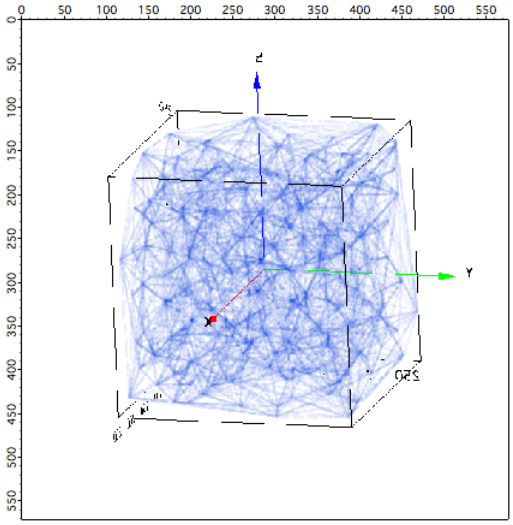


and plotting the high Q and low Q together show the Sobol-generated data to be a little bit noisier. Plus, the enoise-generated structure seems to be larger at low Q. And I really don't have any idea why this would be the case.

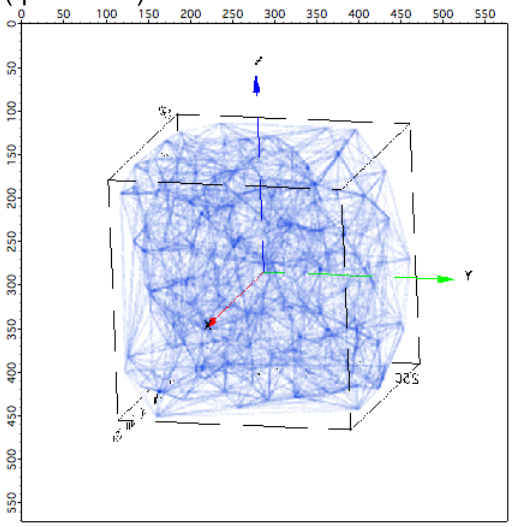




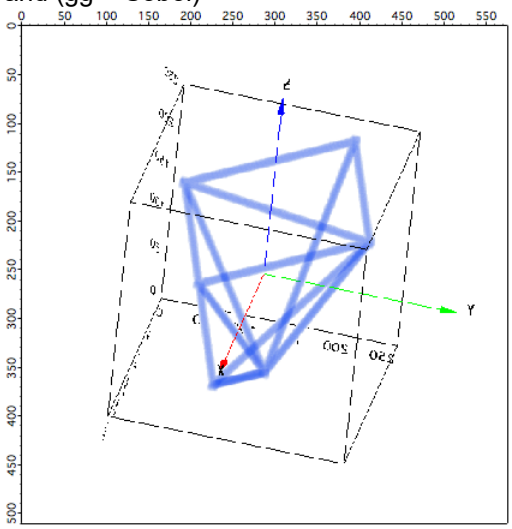
And the respective plots of the cylinder fills:
(e = Sobol)



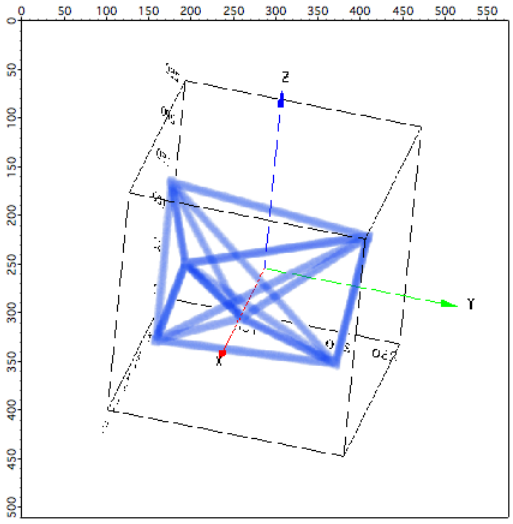
(q = enoise)



and (gg = Sobol)

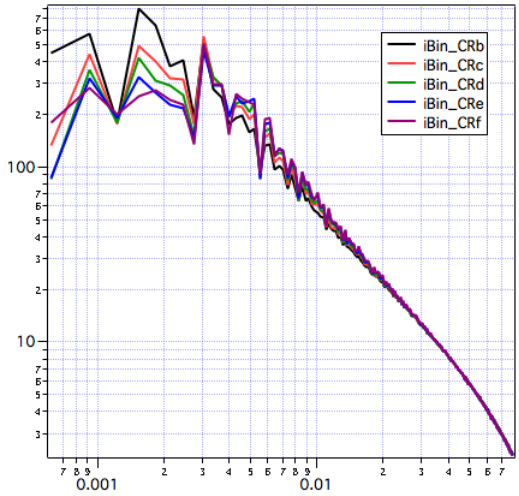


(ss = enoise)

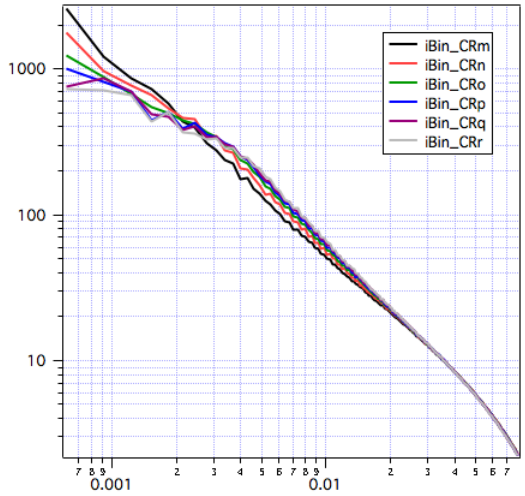


These are the sobol plotted together, and enoise plotted together. What do these graphs say?

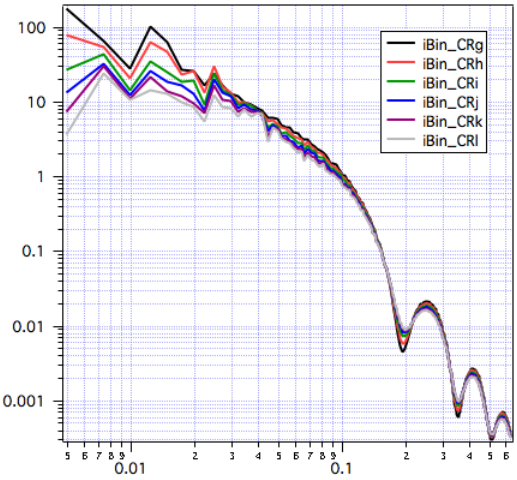
not much here...



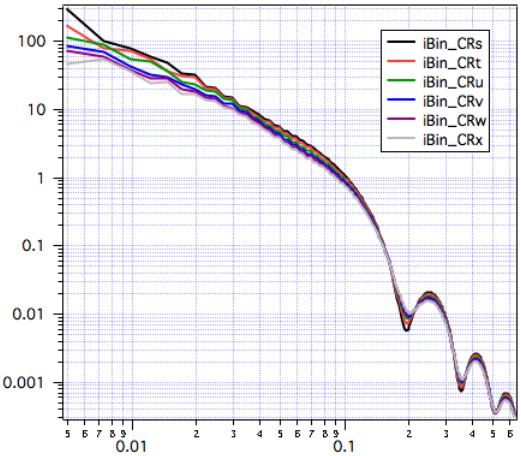
An interesting change in the inflection in the figure below as the volume fraction increases. There must be something here. A change in the average distance between nodes reflected here (shorter distance as the concentration, so the inflection moves to larger Q)



Nothing here, really.



Do I see the shift in inflection in the figure below? Or is it the wrong length scale?



- **Filling Lattice Shapes**

2 MAR 2011

Procedures have been added that allow filling of shapes on a rectangular or hexagonal lattice. What is currently available are functions for placing parallel cylinders in the x-direction of the matrix, which is equivalent to the incident beam direction in the FFT.

How they work:
At the central YZ plane of the matrix, N points are placed in the plane in the specified fill pattern. Then cylinders of a specified length, radius, and center-to-center spacing are drawn at each of the N points. In these procedures, cylinders are drawn in the x-direction. If the list of N fill points are kept intact, then core-shell objects are a trivial extension to draw.

Three fills are provided:
Random (either enoise or Sobol)
Rectangular
Hexagonal

Still to do:
- verify the spot appearance/disappearance as the x-cylinders are rotated off of the beam axis (around Z)
- verify the transformed matrix FFT (parallel to Y and Z)
- add additional planes of points so that fills of spheres can be done to generate SC, HCP, BCC
- introduce some amount of angular disorder in the rods that are placed, and polydispersity of the cylinder radius.
- verify the peaks of the lattice. Be sure that the 2D lattice of cylinders really gives the sqrt 3/5/7/... spacing as the full HCP lattice gives. Also, be sure that the underlying cylinder form factor is not affecting the peaks (and it likely is)

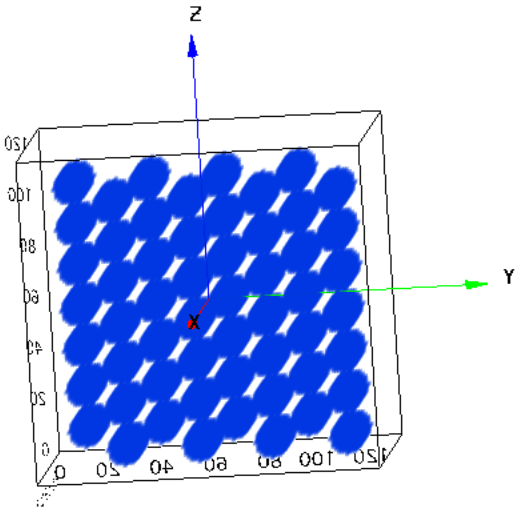
So a simple verification of the hexagonal calculation is shown below. The ctc distance is 80 A, so the hexagonal peaks are expected to be at:
2*pi/80 = 0.0785 1/Å
q*sqrt(3) = 0.136
q*sqrt(5) = 0.175
q*sqrt(7) = 0.207

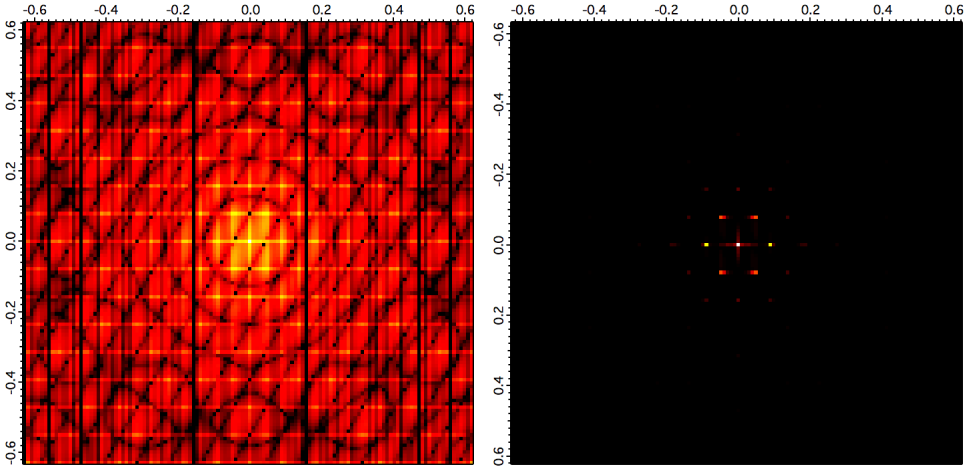
And the FFT or Debye calculation gives:
q1 = 0.0908 1/Å
q2 = 0.157
q3 = 0.238
q4 = 0.271

-- this corresponds to a spacing of 69 Å, and if so, the sqrt(5) peak is missing (should be at 0.203 1/Å). So some tweaking need to be done.

The core-shell cylinders give a different

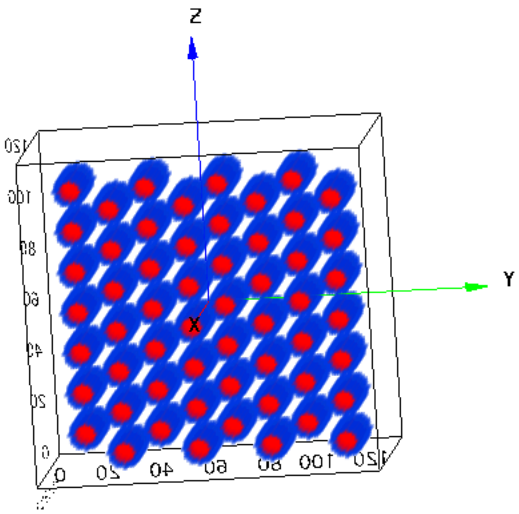
PutXAxisCylindersHexagonal("mat",30,300,80)
(fill = 1)
Number of points = 379680
Fraction occupied = 0.181046
Overall Cube Edge [A] = 640

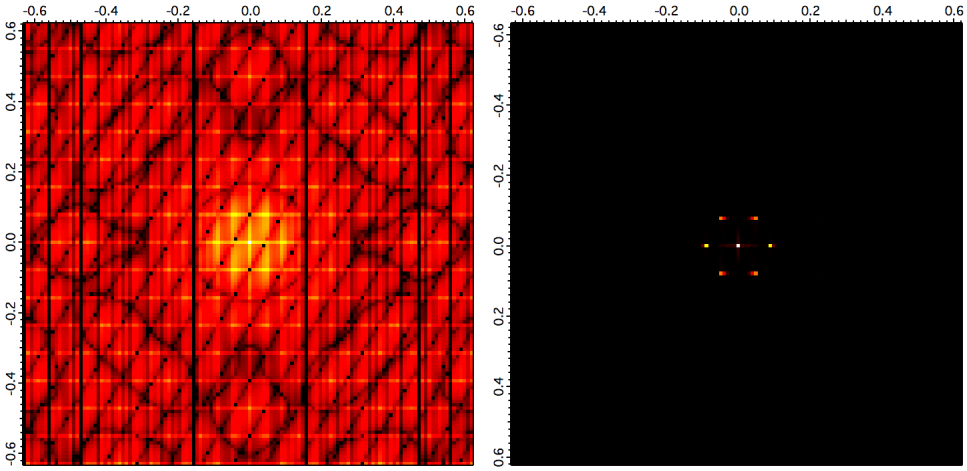


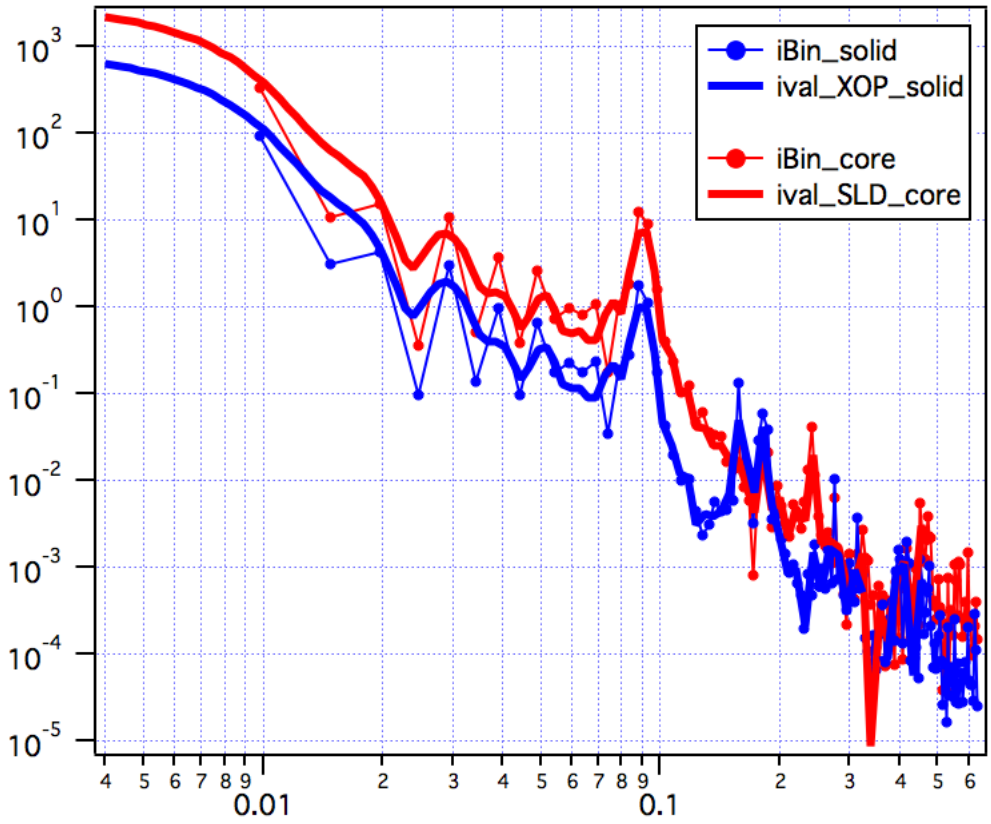


Core-shell cylinders:

30 A radius cylinders, fill = 1
then 20 A radius cylinders, fill = 3
(matrix = 0)


Z
Y
X


-0.6 -0.4 -0.2 0.0 0.2 0.4 0.6
-0.6 -0.4 -0.2 0.0 0.2 0.4 0.6



- **Slicing the 3D FFT to 2D**

This set of macros is to verify the operation of the 2D FFT and slicing by calculating both the FFT and the model calculation of a cylinder in 2D. If everything is working, then the 2D image that is calculated, and the 2D slice of the FFT result should match up.

Then in the 2nd part, the 2D slice of the FFT is interpoated to match the experimental QxQy data points.

1) Comparing the slices to model calculatons

In this, the matrix was set to:

N = 128
T = 5
Solvent SLD = 0

For the 2D cylinder model:
"fake2DData"
128x128 detector, -0.64,0.64 limits on the q-range

Scale = 1
R = 40 A
L = 400 A

cyl SLD 1
solvent SLD 0

bkg = 0

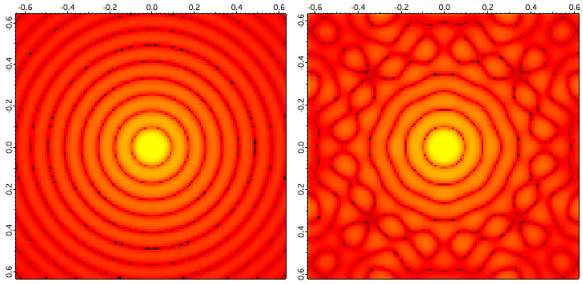
(theta)
(phi) changed as needed

files are labeled ywave_Cyl2D_a (for the model)
files are labeled iBin_Cyl2D_a (for the FFT)
files are labeled Cyl2D_mat_a (for the model calculation)
files are labeled logP_Cyl2D_a (for the detector slice of the FFT)

And the results....

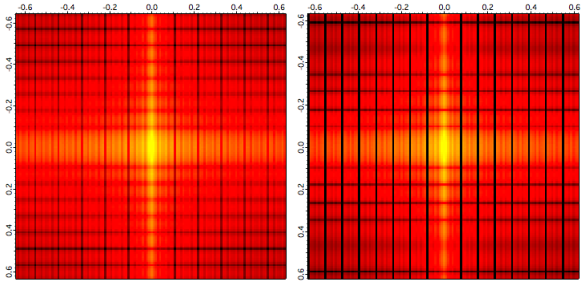
Case 1:
Cylinder is lined up in the X direction as defined in the voxelgram, which is "end-on" or the Z direction as defined in experimental instrument coordinates. The result is symmetric, as expected. See case 4 for an explanation of the "pretty" FFT pattern.

-Model- -FFT-



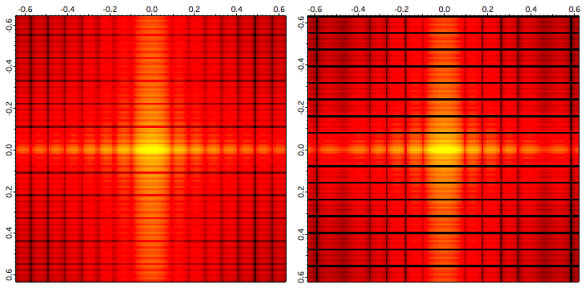
Case 2:
Cylinder is lined up in the Y direction as defined in the voxelgram, which is horizontal or the X direction as defined in experimental instrument coordinates.

-Model- -FFT-



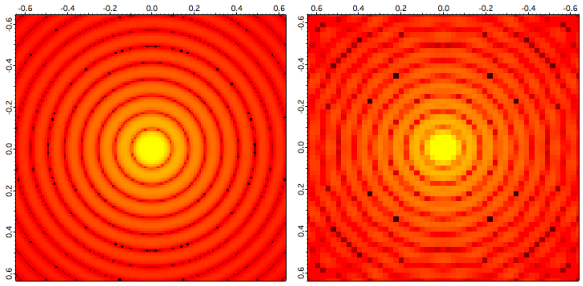
Case 3:
Cylinder is lined up in the Z direction as defined in the voxelgram, which is horizontal or the Y direction as defined in experimental instrument coordinates. Finer spacing of fringes in the vertical directon, the direction of the larger dimension.

-Model- -FFT-



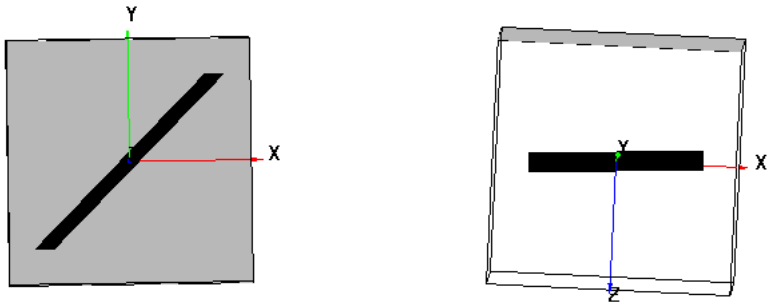
Case 4:
The same as case 1, but using a smaller "T"(T = 2 A) to get a smoother calculation. The "lumpiness" of the case 1 calculation is due to the gridding of the spheres looking end-on. Making things on a smaller grid smooths things out by having more spheres. Somewhat counterintuitive since they are still on a grid and there are more of them, but it is what it is. For the display, the FFT image has the q-range rescaled to -0.64,0.64, zooming in one the same q-range as the model calculation. Makes for a more pixelated image, but is identical.

-Model- -FFT-



Case 5:
Direction is 45 degrees, with the cylinder in the detector plane, (XY as defined by detector, YZ as defined in the voxelgram)
Using ChangeAngle(90,45) defines the direction
theta = 1.5708
phi = 0.785

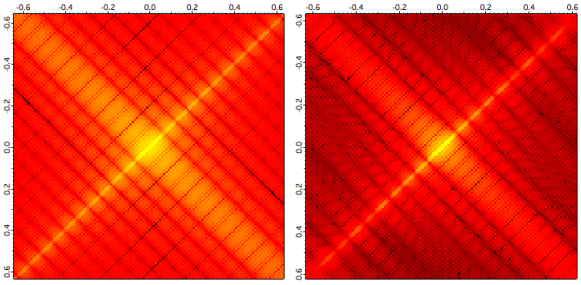
-Front- -Top-



The voxelgram is generated by:
drawing a cylinder in the X-direction

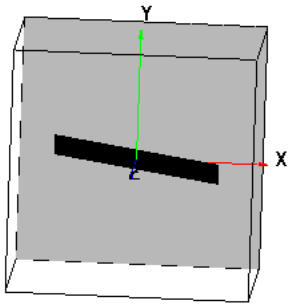
rotating 45 degrees around Z
then transforming XYZ->ZYX

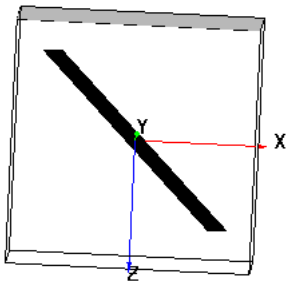
-Model- -FFT-



Case 6:
Direction is 45 degrees, with the cylinder in the horizontal plane, (ZX as defined by detector, XY as defined in the voxelgram)
Using ChangeAngle(45,0) defines the direction
theta = 0.785
phi = 0

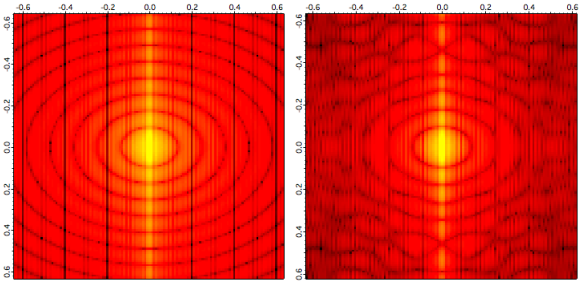
-Front- -Top-





The voxelgram is generated by:
drawing a cylinder in the X-direction
rotating 45 degrees around Z

-Model- -FFT-



The FFT pattern could be improved in quality by using a finer grid, as in the end-on case.

-Jan 10, 2011 SRK

2) Interpolating the FFT Slice to Match Experimental Data

Here, we can simply take the FFT slice and use its data scaling for the Qx and Qy values, since these naturally fall out from the FFT scaling. Then it is a simple (one-liner) to interpolate the slice to match the QxQy scaling of the linear data matrix "_lin". The images can be compared, or the surfaces (Gizmo), or the interpolated surface could be used as a 2D fit function with proper wrapping.

So for a case of a cylinder:
R = 100
L = 200
SLD Cyl = 1e-6
SLD solv = 6e-6

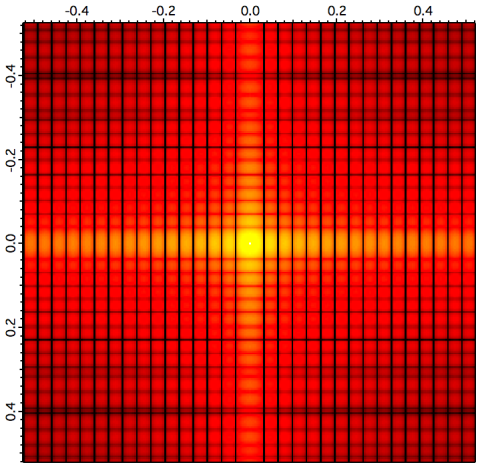
theta = 1.57
phi = 0
= the cylinder is horizontal, pointing along detector X (=FFT Y)

Two experimental 2D data sets are used, SILIC009 (high Q, offset) and SILIC010 (low Q, on-center)

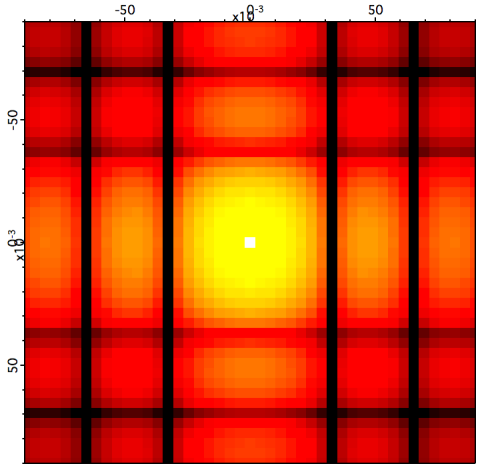
Here, T=6 and N=256 were chosen. These give Qmax = 0.52, and deltaQ = 0.0020 (1/A) These are reasonable values, giving a fine enough grid to interpolate. Data displayed is in log-scaling of intensity to show the details

So for low q, on-center (SILIC010_DAT):

The FFT Detector slice

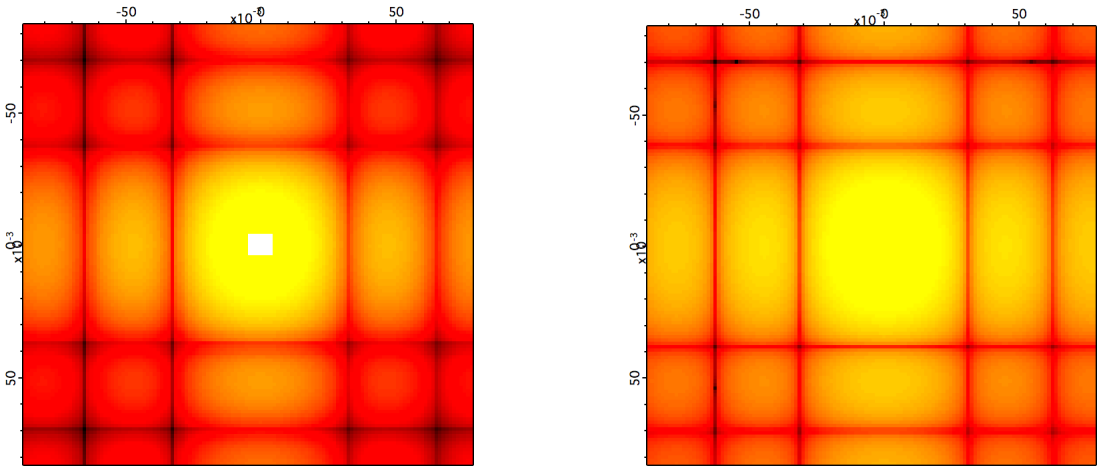


Axes scaled to SILIC010 values



Interpolated Slice
SILIC010_DAT

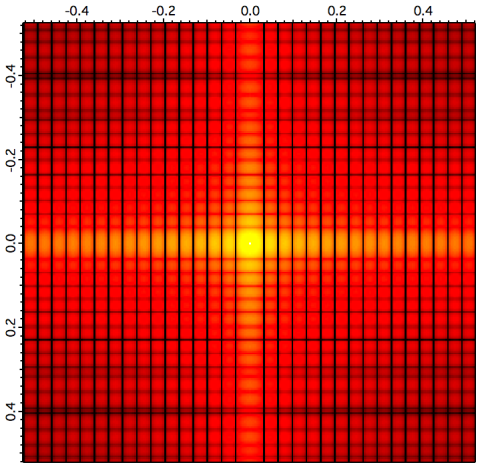
Cyl_2D model for



The pixelation of the useful range of the FFT slice depends on the choice of N and T. Selection guides will need to be developed for that as needed. For these parameters, there is enough in the FFT result to give a good interpolation, that is in excellent qualitative, and good quantitative agreement with the model calculation. Note that the vertical scaling of the FFT slice is WAY off (~ x10^5 too high). That will need to be fixed.

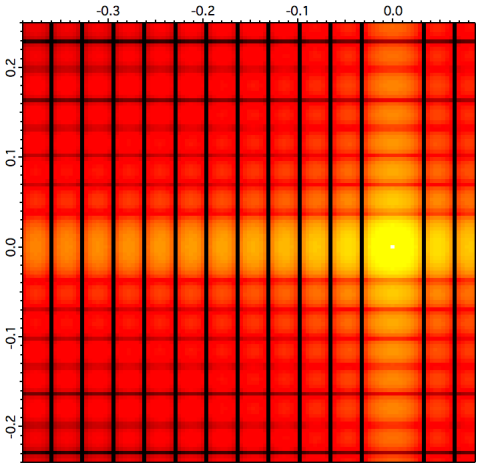
Then for high q, off-center (SILIC009_DAT):
We start with the same FFT result. It's the same structure, and we have enough q-range from the FFT. So there is no need to recalculate the FFT at all.

The FFT Detector slice

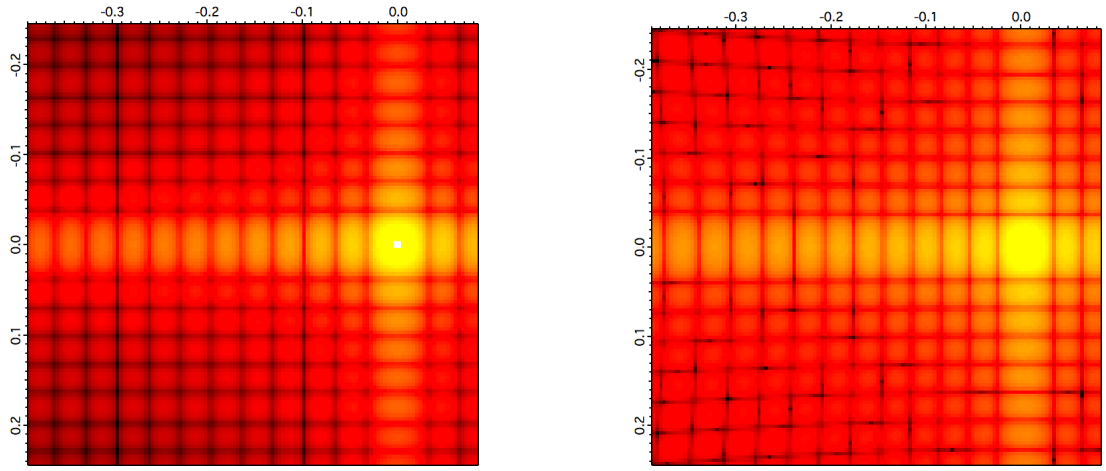


Interpolated Slice
SILIC010_DAT

Axes scaled to SILIC010 values



Cyl_2D model for



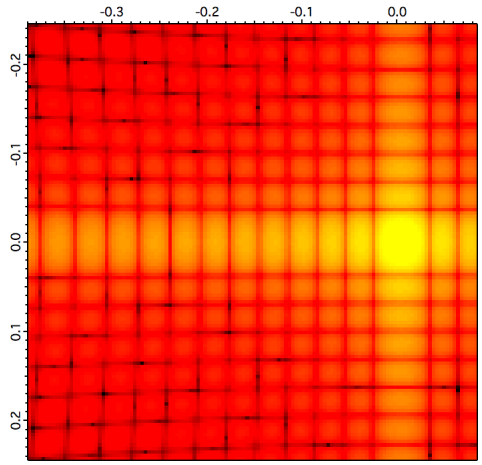
More complex examples could be generated, but these two test cases are sufficient to show what can be done with the 2D slice from the FFT.

Some comments:

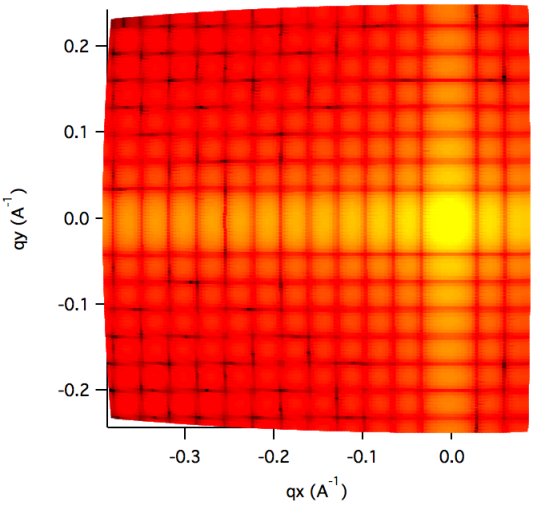
- A "reasonable" match of the delta Q of the data and FFT is best for the interpolation, if possible.
- Increase T to get a smaller delta Q (but you lose the fine detail of the structure)
- Increase N to get more points in the final I(Q) to interpolate (but the calculation slows as N^3)

There is distortion visible as curved bands (the P(Q) minimum) in the FFT for the highest q-values. This is an effect of forcing QxQy onto an evenly-spaced grid, which is only "true enough" at the lowest q-values where the small-angle approximation is good, and Qz cana safely be ignored. Plotting the intensity as a function of Qx and Qy shows how the QxQy surface measured is not a flat, evenly gridded plane. For the comparison of models to data on a surface, this is of no consequence since the surface display is just for viewing, and the fact that both data and model suffer from the same (minor) distortion by displaying them on the same grid. But beware of other calculations and comparisons. Be sure QxQyQz is not forgotten.

FFT slice



Qy vs. Qx plot



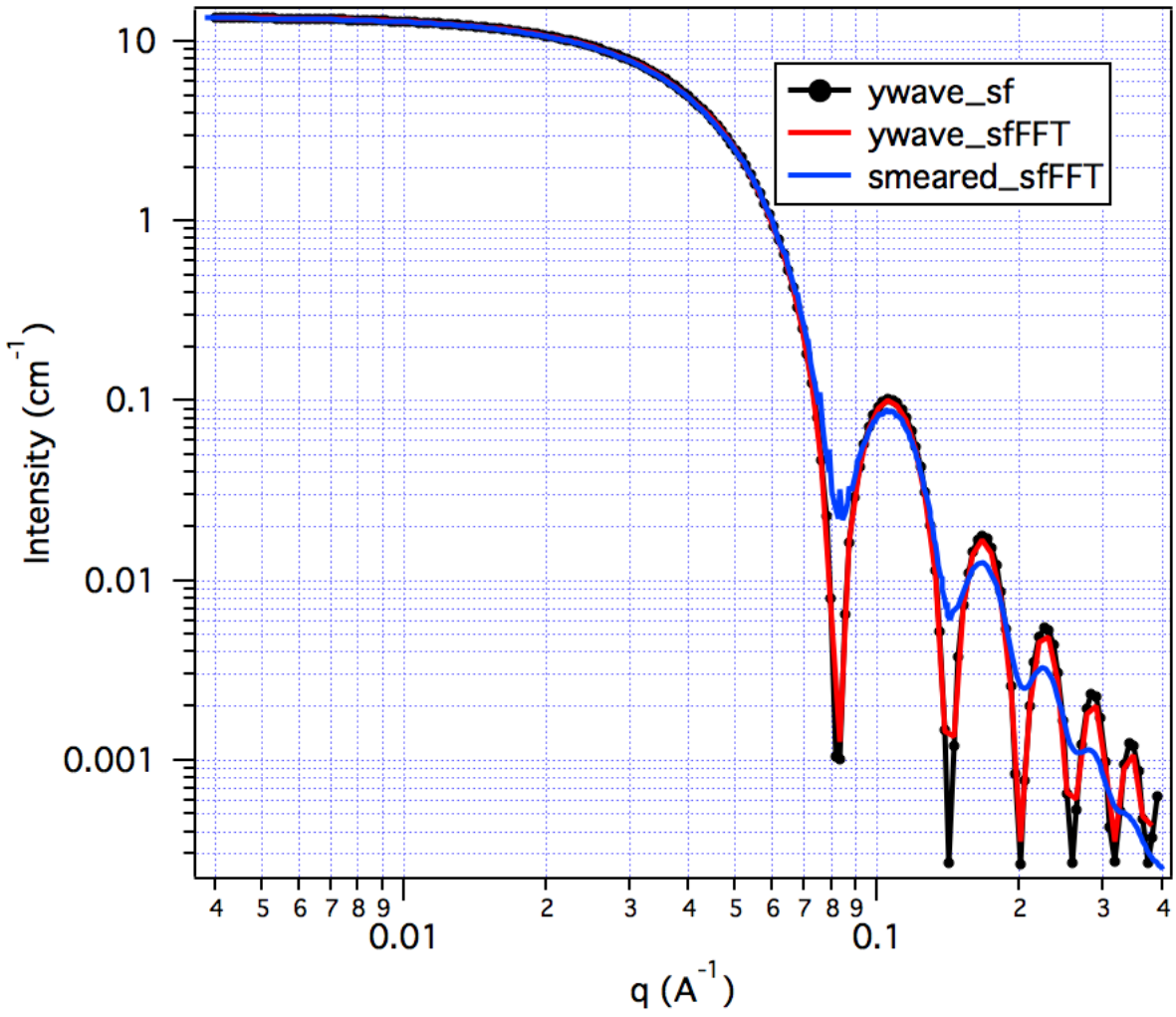
- **Debye Spheres as a Fit Function**

Defining the sphere structure as a filled matrix with a spherical structure gives a fit function that behaves as a normal fit function. Ignore the FFT tag, the calculation is Debye Spheres with a binned distance.

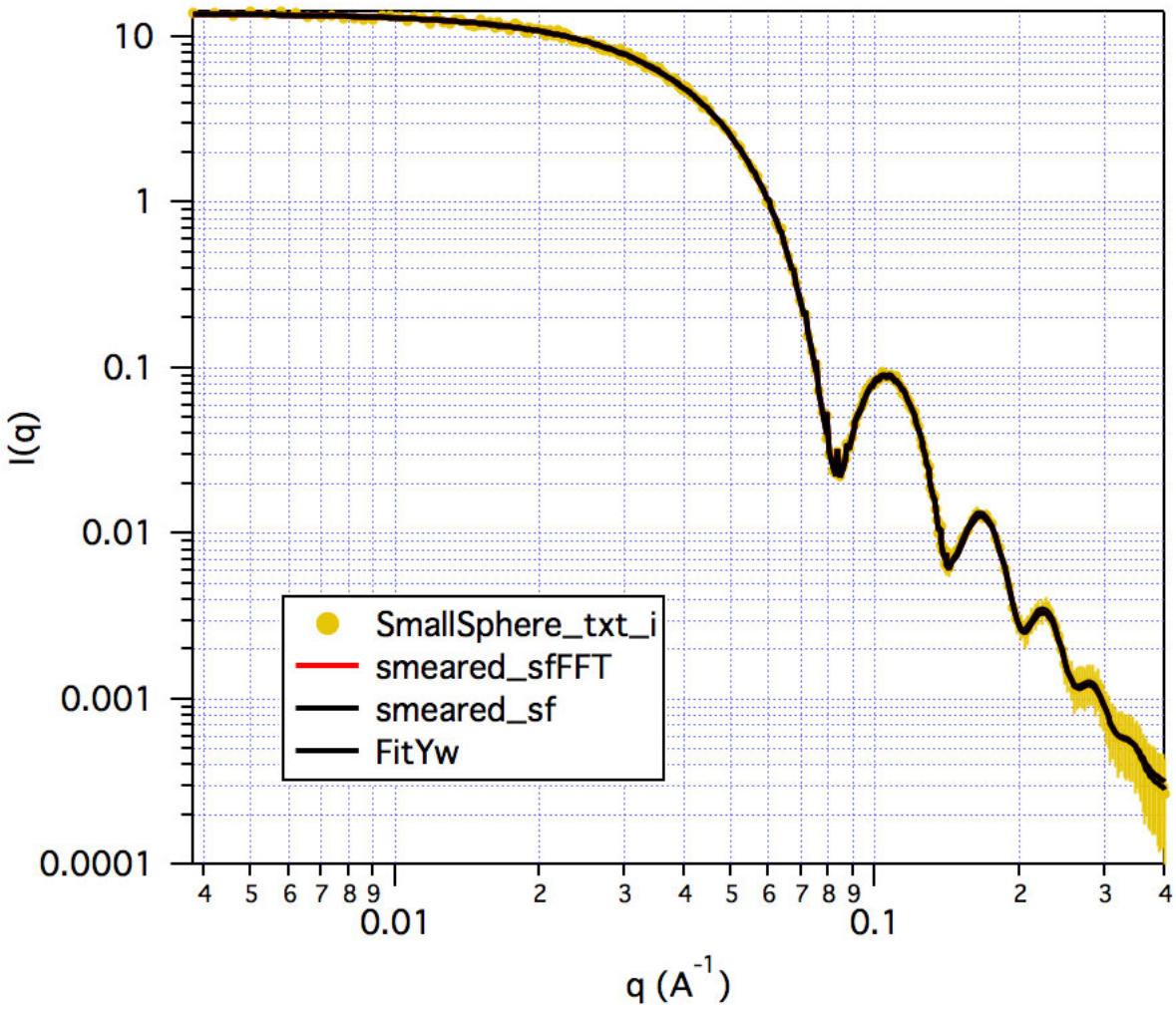
Depending the number of spheres in the calculation, it can be rather slow as expected. The calculations here use the binned distance method since there is only one SLD (different than solvent).

The smeared calculation works too, but is beastly slow, much more so than the expected 20x slower for 20 gauss points. Timing is yet to be verified, then speedup can be targeted if it's ever important.
-- Updated 7 MAR 2011 --
The resolution smearing has been rewritten to be done in an AAO fashion. A vector of Nq*nord abcissa are passed to the function AAO, and then summed back into the Nq points after the smearing is completed.

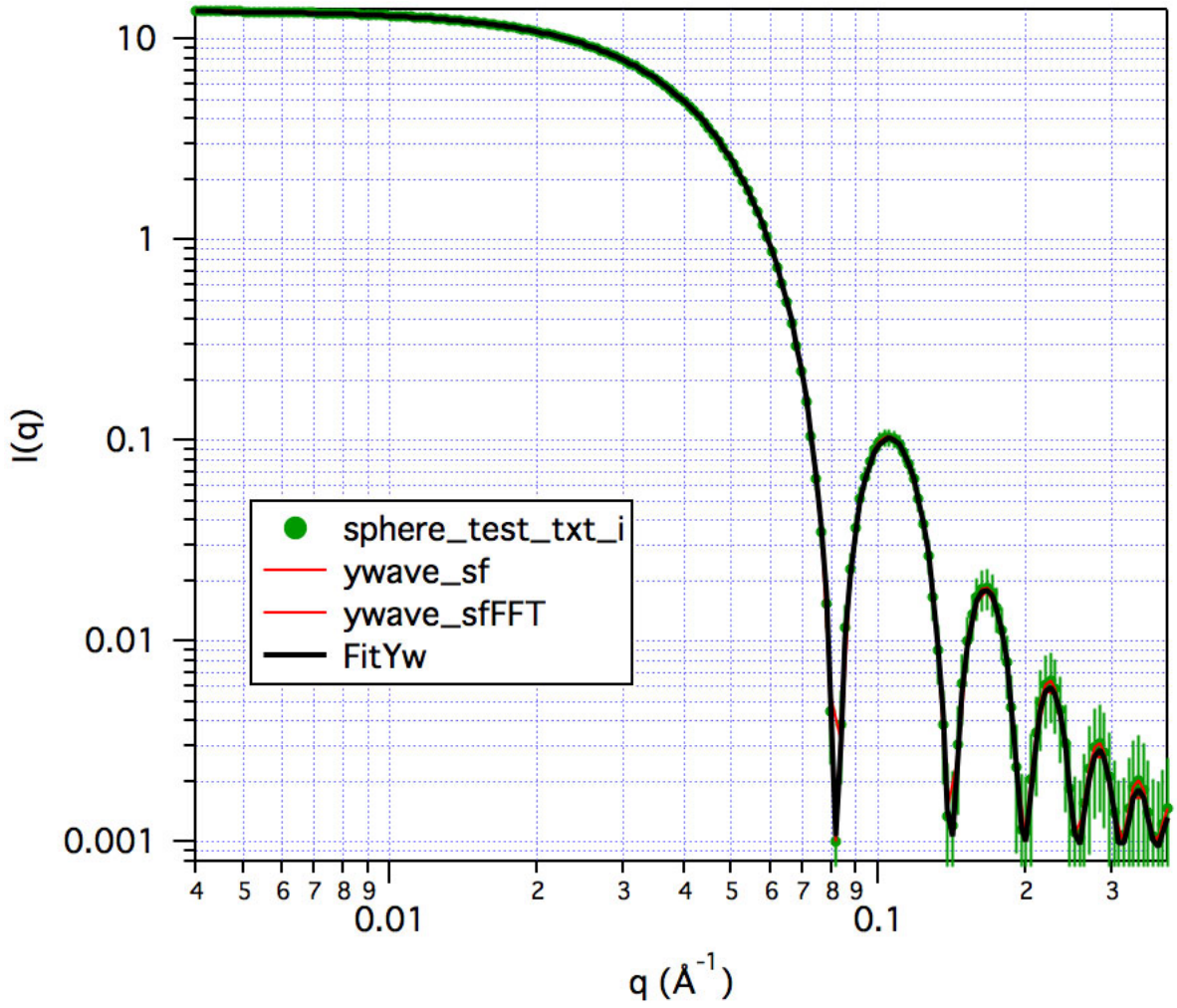
The graph below is the analytic sphere form factor, the Debye Spheres calculation (sfFFT), and the smeared calculation.



Below is a comparison of the fitted SmearedSphere form factor, and the calculated SmearedDebyeSphere fit function. Both are identical. A fit was also done with a smeared sphere FFT function, and the results are identical. (see note above. The AAO smeared fit took 144 s).



Below is model data (with no resolution smearing) and the fits using the analytic sphere form and the Debye Sphere fit function.



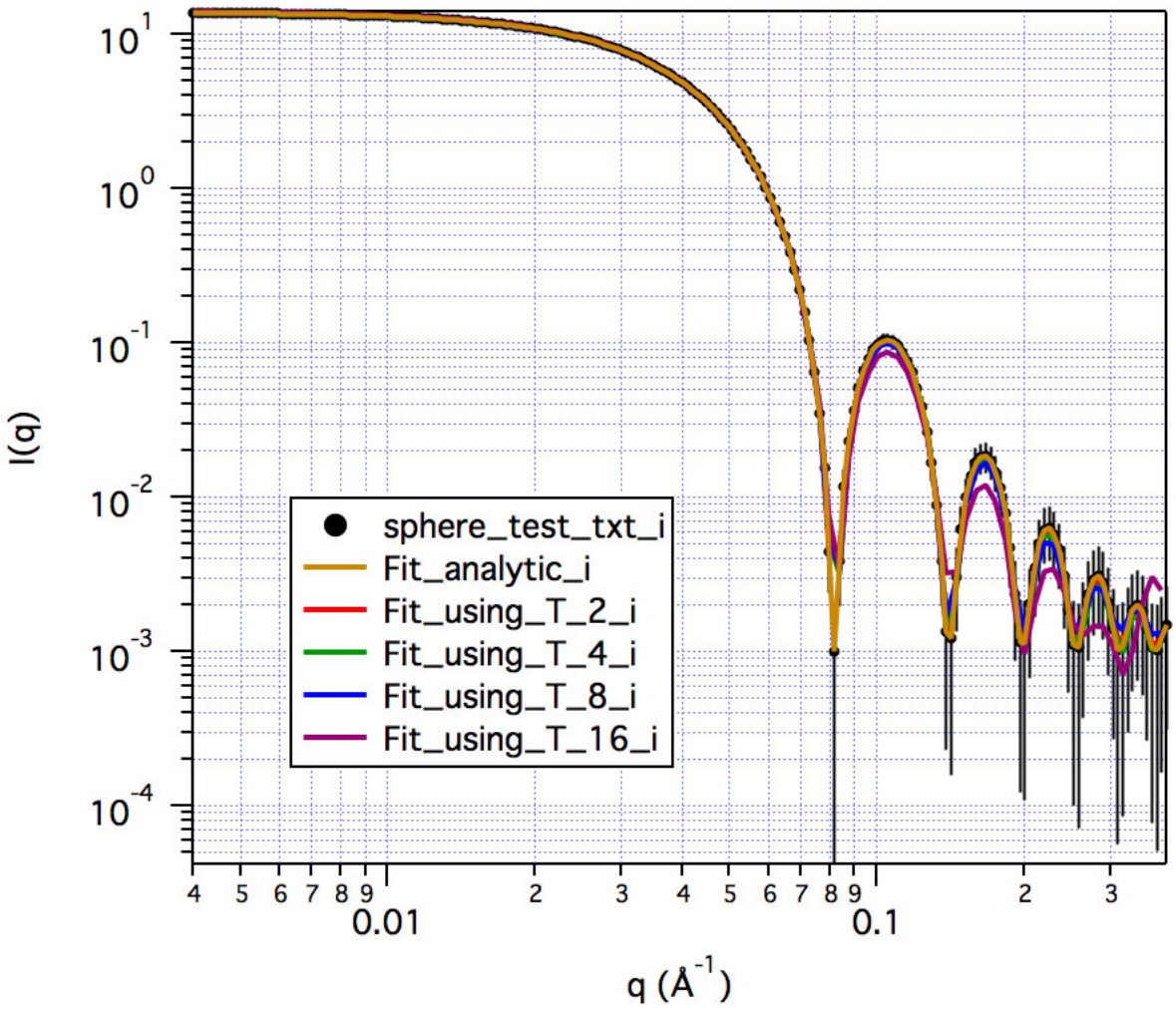
The results of the fit coefficients are identical, within error. To do the DebyeSphere fit, the epsilon value for the sphere radius was set to 6 A. Using epsilon that is too small gives no dependence on that discretized parameter. Not surprising, but now I need to define how large epsilon needs to be to get a proper gradient in that dimension.

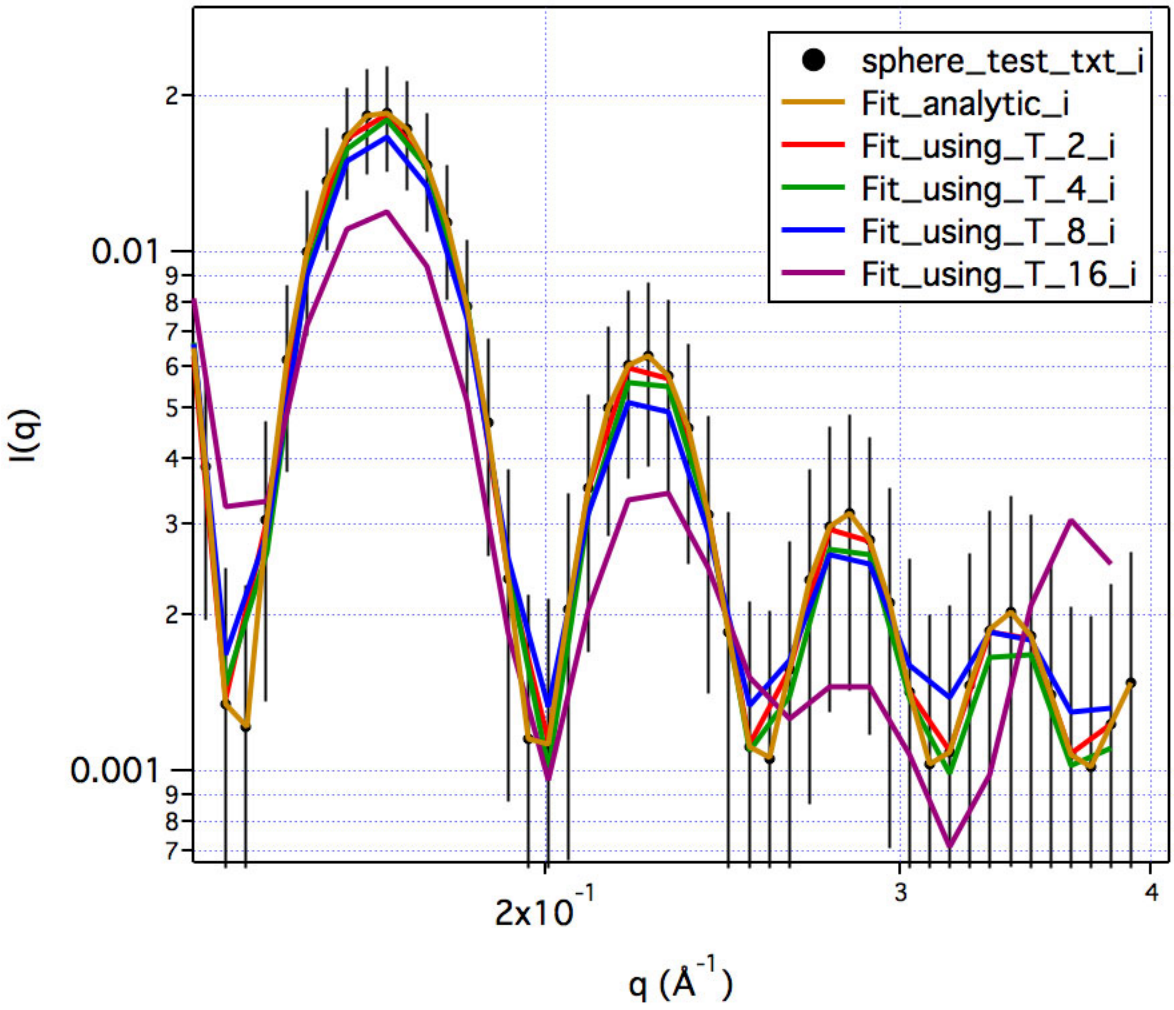
T	time(s)	Chi^2	R (A)		
-	0.5	0.022	55.00	analytic model	
2	1643	0.068	54.97		
4	171	6.2	54.81		
8	21	6.6	54.81		
16	101	103	54.15		

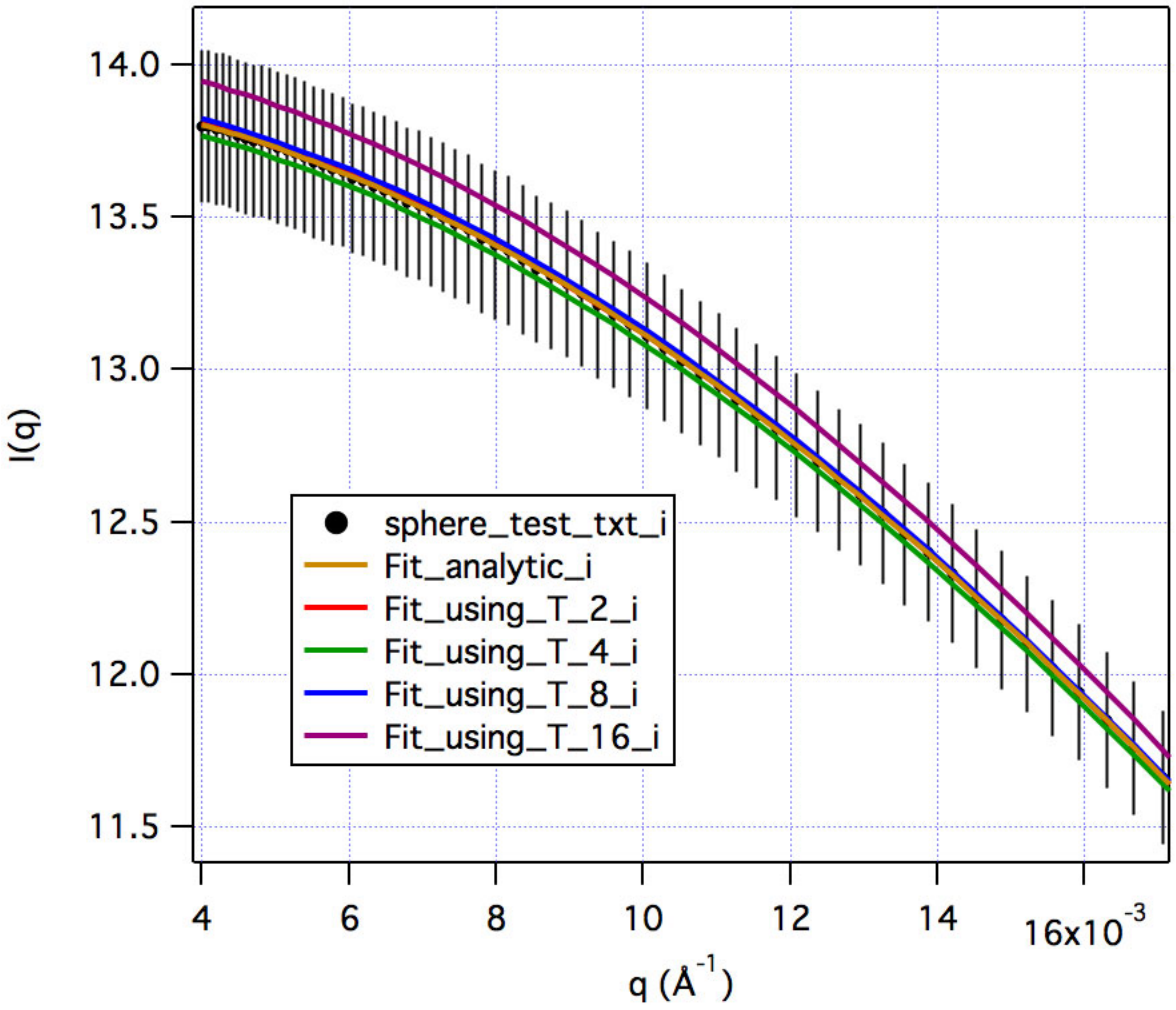
The time listed is the time for the fit to complete, using the threaded binned calculation, each starting from the same guess, constraints, and epsilon (=6 for the radius). The following graphs show the results of the fits. Overall, the fits look good, with the quality degrading as T starts to get very large. The fit looks crummy at the largest q-values, but what really contributes to the large chi-squared is probably the offset of the model at the low q values, where the error bars are smaller (artificially generated proportional to the intensity).

So there's a tradeoff here, as in all aspects of life. Using a relatively large T value, say 1/20 or so of the maximum dimension (110 A in this case) gives a reasonable tradeoff between speed and accuracy. Too coarse, and speed suffers as the model becomes an inaccurate "voxelated" representation of the smooth physical structure. Too fine of a grid, and the calculation is very accurate,

but the time for the fit can quickly become prohibitive.




sphere_test_txt_i
Fit_analytic_i
Fit_using_T_2_i
Fit_using_T_4_i
Fit_using_T_8_i
Fit_using_T_16_i
I(q)
0.01
0.001
2x10^-1
3
4
q (Å^-1)



Still to investigate:
- What is the "optimal" FFT_T?.
- Genetic Optimization. Does this still need some sort of epsilon values set to get the parameters to "move"?
- Re-doing some of the filling routines so that they don't need a matrix at all. If the points could be filled directly into the XYZRho waves, then there would (potentialy) be a speedup. One of the difficulties in filling this way is that there is no convenient way to check for overlap, and if there is overlap, duplicate points are written to the XYZRho columns. The I need to figure out how to get rid of these.
- Does having the Gizmo window open during the fit slow things down? No, not at all. I've done the test.
- I've built a function for cylinders that works just fine, but the optimization is rather fiddly. It seems to fail regularly with no dependence on the cylinder length. Now with the sphere results, a possibility may be that the small number of points at the lowest q that are sensitive to the cylinder length have such a little contribution to the chi-squared compared to the large nubmer of points at high q that the optimization sees no dependence, no matter how large of a step is taken. I am seeing a huge difference in chi-squared for the sphere.
- Test the cylinder fit as a function of T, to see if there is an optimal value. Or try the cylinder fit with a restricted q-range to determine the two lengths independently, but then a fit of the full q-range may still be insensitive to the length.

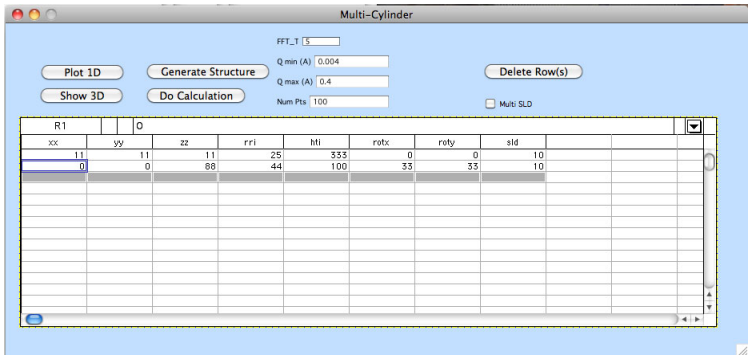
- **Multiple Oriented Cylinders**

FEB 2012 SRK

These procedures for calculating the scattering from multiple cylinders is based on a parsing routine written by Ken Rubinson that takes a list of cylinder specifications and parses that into a filled matrix suitable for calculation of the scattering pattern.

To start the calculation set N and T from the main FFT panel. This calculation will use either the binned distance or binned distance + SLD methods for the calculation rather than the FFT. The volume is used to set up the structure.

Macros -> Setup_KR_MultiCylinder opens the following panel:



The FFT_T is set on the main panel
The Qmin / Qmax / Number of points are set here as needed

Each row in the table defines a cylinder, using the columns as follows - apologies for the odd notation:

xx = xCtr
yy = yCtr
zz = zCtr
rri = radius (real distance, A)
hti = cylinder length (A)
rotx = rotation on X (degrees)
roty = rotation on Y (degrees)
sld = SLD (x 1e-7 A^-2)

To Use:

1) Enter in a row of values for the centers of each cylinder, its dimensions, rotation, and SLD. Note that the (x,y,z) center can be negative here, and will be rescaled to fit into the voxelgram.

2) [Generate Structure] will fill teh voxelgram with the structure as defined. If you use multiple SLDs, not all of the structure may be initially visible.

3) [Do Calculation] will actually do the calculation using the binned distance method. [Plot 1D will show the result. (ival_KR vs qval_KR)

This is a very simplistic description of the use of this rather complex structure generator and the instructions will be augmented in the future as use dictates.

As a Fit Function

With any model function based on filling a volume with shapes, the key to turning it into a fit function is defining the simplest set of lengths that change to enable the variation in the model function that you want (hope) to see. Then these must be translated into a list of input parameters.

Using Ken's parsing method and a simplified three-cylinder model, a fit function has been written: FFT_Fit_3Cyl_KR.ipf with the fit function ThreeCylKR()

The fit function uses three cylinders with a simplified geometry.

- the first cylinder is located at (0,0,0)
- the second cylinder is on "top" (Z)
- the third cylinder is on the "bottom" (-Z)

So then from the 10 input parameters and the simple geometric relationship, the "input" waves can be generated that Ken's parser is expecting. Then after parsing, the scattering is easily calculated. For this calculation, there is only a single SLD, and the diameter of the primary spheres is set as the global value FFT_T.

For this simple case, a single calculation using the distance-binned Debye method takes about 1.2 s, and the smeared calculation 28 s. The unsmeared fit takes approx. 360 s. A smeared fit would take at least 2.5 hr.

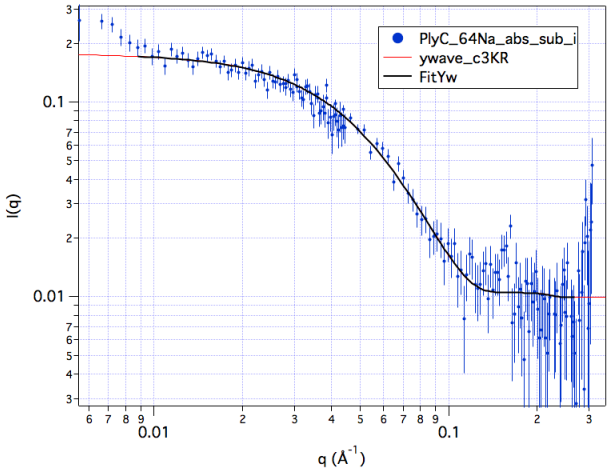
The results of a fit are shown below (epsilon was set to 5 for each of the variables, FFT_T = 4). The errors are large on the parameters, not due to the Debye method, but simply that the parameters are not well-determined. Fitting the data with a simple cylinder model gives large errors on the cylinder length. Adding two more cylinders appears to be too many parameters for the features in the data. But in principle, the three cylinder fit works very well.

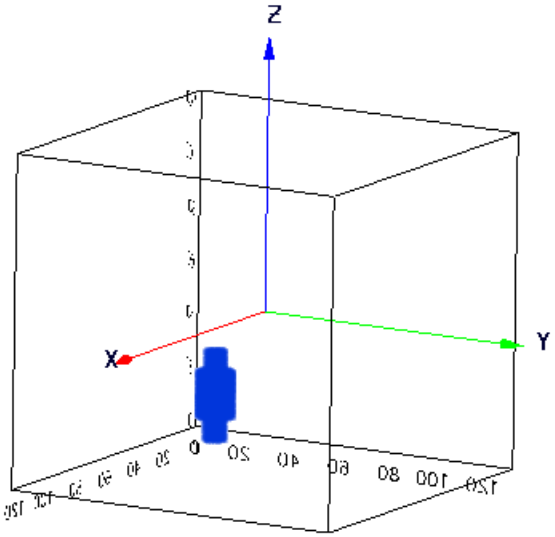
Fit to ThreeCylKR, 2/28/12, 4:05:51 PM

Data file: PlyC_64Na_abs_sub

scale =	0.122796	±	0.0468415
radius 1 (A) =	27.6482	±	1.62241
length 1 (A) =	71.649	±	53.2114
radius 2 (A) =	15.3849	±	121.757
length 2 (A) =	34	±	0
radius 3 (A) =	15.6558	±	159.064
length 3 (A) =	34	±	0
SLD cylinder (A^-2) =	1e-06	±	0
SLD solvent (A^-2) =	6e-06	±	0
incoh. bkg (cm^-1) =	0.00990405	±	0.000533592

chisq = 175.111
Npnts = 162 Sqrt(X^2/N) = 1.03968
Fitted range = [16,177] = 0.008777 < Q < 0.2685
FitError = No Error **FitQuitReason = No Error**



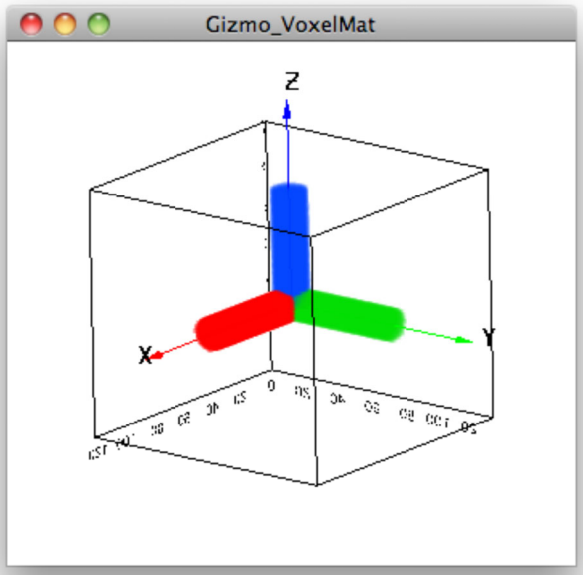


- **Transposing the Matrix**

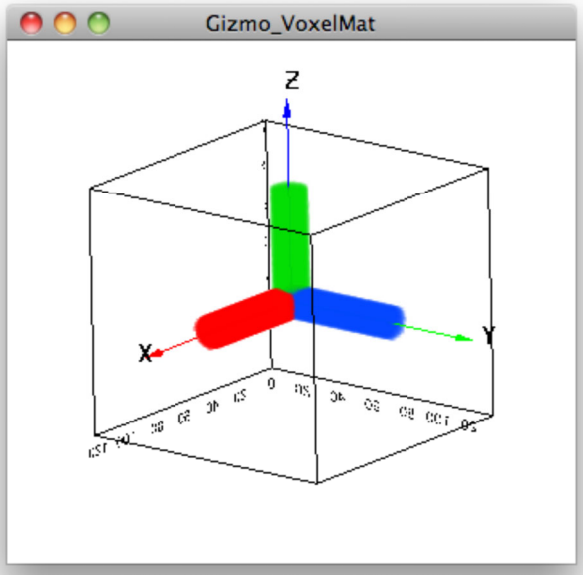
  MAR 2012 SRK

  Transposing the XYZ coordinates of the structure as drawn provides a quick way of re-orienting the structure to allow different views of the same structure to be seen in the scattering plane when viewing the 2D slice. In this example, cylinders are drawn along the positive axes, and the axes transformed to show the results of the transformation. Note that the starting condition is where the cylinder colors match the color of the axis arrow.

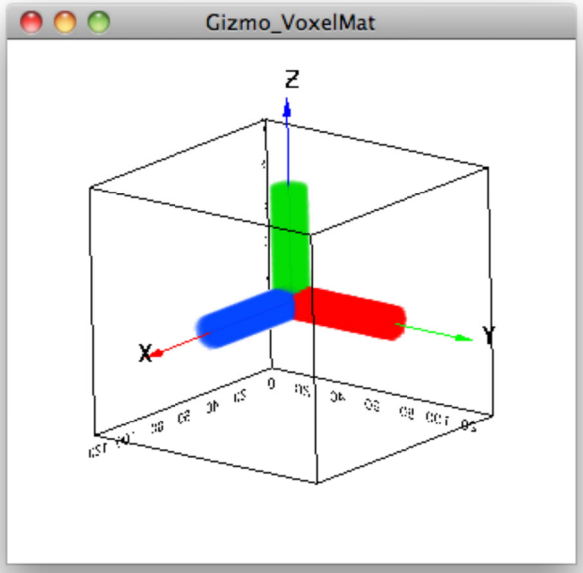
  Initially:



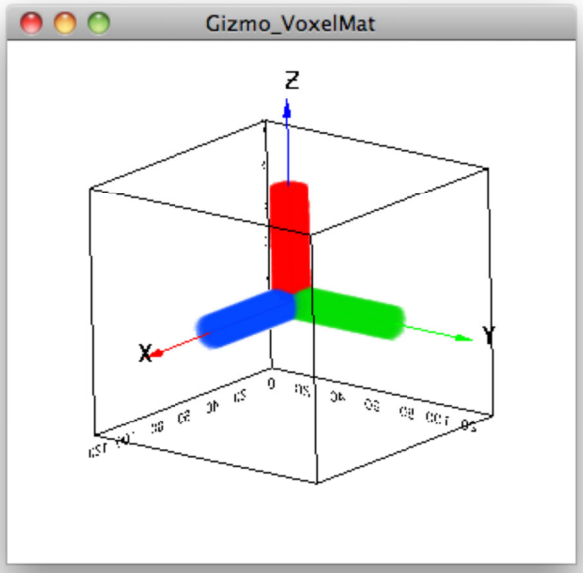
Initial XYZ -> XZY
swaps Y and Z
changes handedness



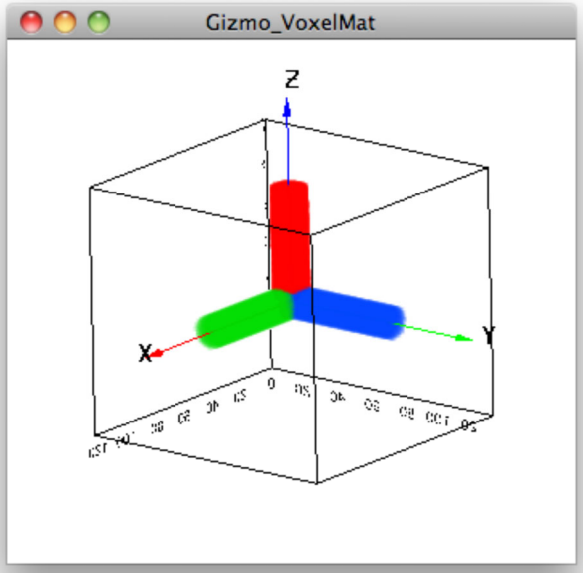
Initial XYZ -> ZXY
swaps all three
no change of handedness



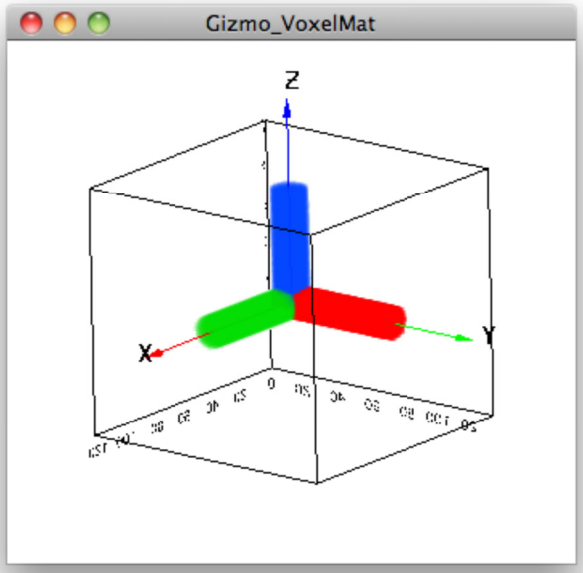
Initial XYZ -> ZYX
swaps X and Z
changes handedness



Initial XYZ -> YZX
swaps all three
no change of handedness



Initial XYZ -> YXZ
swaps X and Y
changes handedness



- **Input Matrix Format**

Generating an Igor-readable text file from a 3D matrix

If you have generated a 3D structure using a different program, you can output your structure in an Igor-readable format. What is outlined here is not the most efficient way to store the matrix, but

the most transparent for generating a simple file.

Each line has a single value (integer, one or two digits ONLY). This is the SLD of the voxel (/ 10-7 A-2). That means that for D2O, the proper value is 63. Negative integers are allowed for negative SLDs

Igor keeps track of a matrix with normal programming indices, that is the matrix dimensions from (0 -> N-1) and are displayed that way. (0,0,0) is at the bottom left corner of the box, not at the center. If your box is centered or non-integer locations, you'll need to translate and truncate.

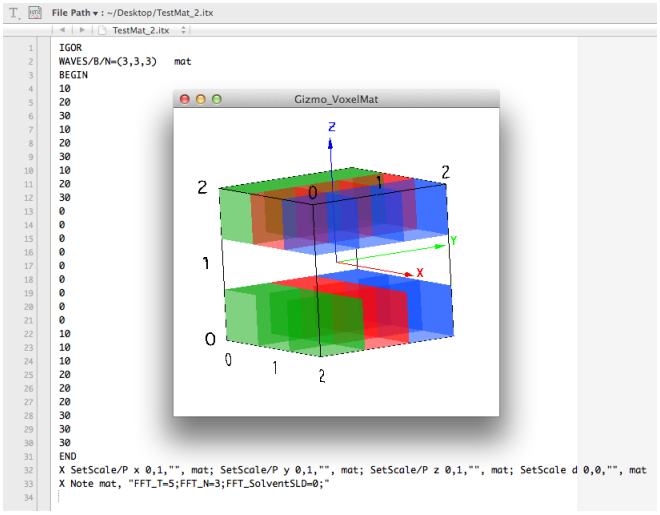
The file structure:

The first three lines:
- keep these the same, except change the 3's to be the number of cells in each direction of your matrix. All three dimensions must be equal. Typical dimensions are 128,128,128 and 256,256,256 but could be larger if needed
- Also on the second line - the matrix is named “mat”. Don't change this.

The values:
- one value per line works fine, starting at (0,y,0), then (1,y,0), etc. After x is exhausted, z is incremented and x returns to 0, and y is rastered again. If something is flipped, the matrix can always be transposed after loading to get the correct orientation.

In the 3x3 voxelgram, 10 = green, 20 = red, 30 = blue, and 0 = transparent (the solvent)

The last three lines
Keep these the same, with the exception that you want to set the numerical values in the wave note (the last line)
FFT_T=(the individual voxel edge dimension, in Angstroms. Does not need to be an integer)
FFT_N=(the number of voxels per edge. This is the same integer as in the matrix definition on line 2)
FFT_SolventSLD=(2 digit integer of solvent SLD as defined above)



Example file Structure:

IGOR
WAVES/B/N=(3,3,3) mat
BEGIN
10

```
20
30
10
20
30
10
20
30
0
0
0
0
0
0
0
0
0
10
10
10
20
20
20
30
30
30
END
X SetScale/P x 0,1,"", mat; SetScale/P y 0,1,"", mat; SetScale/P z 0,1,"", mat; SetScale d 0,0,"",
mat
X Note mat, "FFT_T=5;FFT_N=3;FFT_SolventSLD=0;"

------------------
```