**Purpose of Code:**

Appropriate analytical models for inhomogeneous particles are limited to spherically symmetric particles except for a few cases. Often a simple and more versatile approach to model the scattering from a complicated structure is to directly calculate it from the known distribution of scattering length density. In order to derive the scattering from a complex structural construct, we have developed a Monte Carlo method using IGOR Pro to calculate the scattering profile from an arbitrary shape in a solvent and recover the scattering pattern in absolute scale. This code is fully functional, but is provided “as-is” and requires some knowledge of programming to use, and experience with IGORPro is preferred.

**Theory:**

Efficient Monte Carlo methods have been described previously for this approach, but a brief description is given below. First, a 3-D lattice of points is constructed in real space with x,y, and z coordinates and the value stored at each point is the scattering length density, \( SLD \), of the domain. A random number generator is then used to sample the box volume neglecting the solvent regions. The total number of points generated in each domain corresponds directly to the relative volume fractions of the components in the overall particle. If \( k \) is the total number of Monte Carlo points, then \( \frac{1}{2}k(k-1) \) point pairs’ distances are calculated and the product of \( \Delta SLD_i \) and \( \Delta SLD_j \) are binned into a histogram that corresponds to the maximum size of the box. This computation scales like \( \sim k^2 \) as a function of time, and therefore efficient implementation of the algorithm employs the averaging of multiple iterations of a small number of total Monte Carlo modelings.
Carlo Points. Iterated calculations scale as $k \log(k)$ for computation time. Ultimately, the binning process recovers the $p(r)$ histogram which is normalized such that equation E.1 is satisfied.

$$E.1 \quad I(0) = \int_0^\infty p(r) dr, \quad \text{where} \quad I(0) = \phi_{\text{part}} \frac{1}{V_{\text{part}}} \left[ \sum_i (SLD_i - SLD_{\text{avg}}) V_i \right]^2 = \phi_{\text{part}} (SLD_{\text{max}} - SLD_i)^2 V_{\text{part}}$$

After the normalization of $p(r)$, the scattering intensity is determined directly using equation E.2.

$$E.1 \quad I(Q) = \int_0^\infty p(r) \frac{\sin(Qr)}{Qr} dr$$

Using this approach, structures of arbitrary complexity can be generated and because of explicit inclusion of $SLD_s$, the effects of varying solvent contrast can be assessed for inhomogeneous particles.

**Understanding GUI Interface:**

The graphical user interface was developed to simplify the direct calculation of scattering data from a specified structure, and implementation of new models into the code. Hopefully it is intuitive enough to be used and we will be working continually to improve it. The control GUI is composed of one main panel that has the following functions

1. **Specify box dimensions:**
   - X_Dim, Y_Dim, Z_dim, are global variables which specify the number of pixels along each Cartesian coordinate, $(x,y,z)$
   - Lattice_Size is a global variable which determines the volume of each pixel
   - The values are set in the top left corner of the box and whatever shape you plan on constructing must fit within the dimensions of the box.
   - Using IGORPro’s “SetScale” function, the box dimensions are scaled such that the $[0][0][0]$ position of the matrix wave “BOX” occurs at its center, and all distances are calculated from this point.

2. **Select a Model**
   - By double clicking on a model in the list box, it specifies that model and updates the Fit_Coef, and w_coef waves in the table. Changing the w_coef values allows the user to change the default values for a desired calculation. **NOTE: The geometrical values must always be set such that the object fits within the BOX Volume**
b. Fit_coef wave lists the name of each parameter
c. W_coef specifies current value of each parameter

3. **Do Calculation**

   a. Once model is selected, select a number of MC points, by typing a new value into “# MC Points” tab
   b. Set your solvent scattering length density ($\rho_{\text{solv}}$).
   c. Press, “Generate MC Points”, this generates the number of monte carlo points desired and plots it in the Gizmo Window to show a 3-D representation of the object. The MC points are colorized by the value of their $\Delta SLD^2$.
   d. Select desired volume fraction and incoherent background value
   e. Press, “Calculate $p(r)$”, this is a computationally intense operation that uses $1/2 N(N-1)$ operations to bin the product of $SLD_i \times SLD_j$ into $p(r)$ histogram, where $N$ is the number of MC points.
   f. Once $p(r)$ is calculated, “Calculate $I(Q)$”, $I(Q)$ is generated using integration over a desired $Q$-range. The accuracy of the calculation at high-$Q$ is dependent on the lattice size.

**Creating a model:**

1. Model specification is accomplished using a series of **if…then** statements, shown below for a sphere:

   ```plaintext
   Threadsafe Function SphereFunc(x,y,z,w,SLD_Solv)
   Variable x,y,z //Do Not Change
   Wave w //Do Not Change
   Variable SLD_Solv //Do Not Change
   if(sqrt(x^2+y^2+z^2)\leq w[0]) //Radius_Sphere
       return w[1]//SLD_Sphere
   else
       return SLD_Solv
   endif
   End
   ```

   a. **Threadsafe** - is a modifier of the **Function** call, which allows this wave assignment to be executed on multiple processors, greatly improving the computation time for large box sizes.
   b. **Function Name** (Parameters List) – Use a name that is short, but sufficiently differentiated to be recognized
   c. $x,y,z,w,\text{SLD}_\text{Solv}$ – Parameters that are passed to function from external procedure. These can be a variable or wave, but must be declared as to their type in the order in which they appear in the parameters list.
   d. **In this example,**
i. \( x, y, \) and \( z \) are the physical coordinate values stored in dimensions the wave “BOX” with units [Å].

ii. \( w \) are the model parameters stored in \( w_{\text{coef}} \) wave. This wave is created by the “initialize model” function. In this example the parameters are the scattering length density, \( w[1] \), of the sphere and its sphere’s radius, \( w[0] \).

iii. The function literally reads: “if the pixel selected is a distance from the center of the box less than or equal to the particle’s radius then assign the scattering length density of the sphere, otherwise, assign the scattering length density of the solvent.”

iv. \( \text{SLD}_\text{Solv} \) is a global variable and is specified by the user as the scattering length density of the solvent. For \( \text{threadsafe} \) functions, global variables must be passed through the function call parameter list to be used locally. The \( \text{NVAR} \) function does not work to call it from external memory. This is unique feature of threadsafe functions as their memory is not stored within the IGORPro procedure.

Adding a new model:

1. Once a new model function is made it needs to be added to the function list which requires the modification of two functions and a wave.
   a. First, go to the data browser and right click on \( \text{Func List} \). This is the wave that is displayed in the drop down list. To add your function, right click on a new line in the wave and select insert point. This should create a blank space in the function list wave. \( \text{Func List} \) is a text wave and therefore, the name can be any string.
   b. Next, modify “\( \text{InitializeModel(i)} \)” and “\( \text{UpdateModel(i)} \)” by adding an additional else…if statement where the number value corresponds to the row in \( \text{Func List} \) that your new function name is stored.
   c. “\( \text{InitializeModel(i)} \)” –There are three lines of important code to modify, two “Make” commands and one wave assignment for your new function
      i. \( \text{Make/O/D/N=(# of parameters) Param}_\text{Val} = \{\text{Param1 Value,Param2 Value,…,} \} \)
         1. Add initial guesses for parameter values
      ii. \( \text{Make/O/T/N=(# of parameters) Fit}_\text{Coef} = \{\text{Param1 Value,Param2 Value,…,} \} \)
1. Specify Parameter names
   iii. **Multithread** Box=Function_Name(x,y,z,Param_Val,SLD_Solv)
   1. Additional parameters can be passed to “Function_Name”, but do not remove any of these parameters.

   d. **“UpdateModel(i)”** – the wave assignment function is the only one that needs to be changed.
   i. **Multithread** Box=Function_Name(x,y,z,Param_Val,SLD_Solv)
   1. Additional parameters can be passed to “Function_Name”, but do not remove any of these parameters.

**Some Notes about the Code:**

1. The code histograms the wave BOX into “W_HIST”, which is binned from $0 - 1 \times 10^{-5}$ [Å$^{-2}$] by $1 \times 10^{-7}$, this can easily be changed to represent a larger/smaller SLD range.
   a. This is used to obtain the number of non-solvent components without requiring additional user entry. Changing this value can affect a number of function calculations.
   b. Dependent functions >> **“CountComponents()”**
      i. This function uses W_Hist to create waves Comp_Num and SLD, which store the scattering length density and pixel number of each domain.
      ii. The SLD of solvent is rounded to an accuracy of $1 \times 10^{-7}$ for this calculation!!
      iii. Comp_Num is critical for the calculation of I0 >> see **“CalculateI0()”**
   c. Dependent functions >> **“GenerateRandomPoints(N)”**
      i. The for loop is dependent on the number of components in Comp_Num because MC_Num is a wave that is used to count sufficient monte carlo points such that the number of MC points in each unique SLD domain is proportional to the volume of that domain.

2. All units are assumed to be in Angstroms. The only unit conversion is for the scattering data to absolute scale in [cm$^{-1}$]

3. Q-range and number can be changed in the **“CalculateIvsQ(Pr)”** function by changing the local variables $Qnum, Qmin, and Qmax$

4. The calculations are currently not set up to do any fitting, nor is polydispersity implemented.
   a. It is in principle possible to implement both using iterations of small numbers of monte carlo points from a large number of successive boxes and summing the contributions to the scattering profiles