Graphical User Interface Design for the X-Ray Crystallography Data Processing Program - Scaling

K. C. Van Zandt
Dan C. Marinescu
Robert Bolotovsky
Michael Rossman

Report Number:
98-005
GRAPHICAL USER INTERFACE DESIGN FOR
THE X-RAY CRYSTALLOGRAPHY DATA
PROCESSING PROGRAM - SCALING

K. C. Van Zandt
Dan C. Marinescu
Robert Bolotovsky
Michael Rossmann

CSD-TR #98-005
March 1998
Graphical User Interface Design for the X-Ray Crystallography Data Processing Program - Scaling

K. C. Van Zandt, Dan C. Marinescu
Department of Computer Science

Robert Bolotovsky, Michael Rossmann
Department of Biological Sciences

Purdue University
West Lafayette, IN 47907

Abstract

X-ray crystallographers depend on software to process and analyze their data. These programs were often developed with emphasis on mathematical algorithms rather than input and output control. As a result, these programs can be difficult to set up. This paper discusses a specific data processing program, scaling, which is the last step in crystallography data processing. It computes the scale factors for the different X-ray images and merges them to obtain a list of the Bragg reflection intensities. A graphical user interface is introduced to simplify the input and output of the scaling step. This graphical user interface reduces the complexities of the scaling process by providing default values to input control whenever possible, presenting a list of choices when appropriate, and displaying the extensive statistical output in table and graph form for easier viewing.
1. Introduction

Macromolecular crystallography relies heavily on software to help process the voluminous amount of images collected during an experiment. The data processing programs take a set of raw diffraction data and convert it into a list of Bragg reflections with intensities. Much of the crystallographic software developed in the past was introduced for mini computers running the UNIX operating system in a university environment. These programs were initially designed to run on ASCI terminals. They usually lack a graphical user interface, and depends on text input files for control parameters. Often, these control parameters must be entered in a particular format to be read in successfully.

With the increasing power of the personal computers, the user is no longer satisfied with the user-unfriendly form of input where control data must be entered in a fixed format. Everyday desktop programs all have sophisticated graphical user interface. The tools for building graphical user interface have also matured. More and more of the scientific programs now are spruced up with a graphical front end. In addition, with the advance of hardware, raw images can now be collected at a fast rate making it adamant that data processing and data analysis be automated to some extent.

In this paper we briefly describe the different stages of macromolecular X-ray crystallography experiment. We discuss the detailed steps of the data processing stage. We then describe the specific step of scaling - defined as finding the scale factors for the integrated intensities of the different images. We present the design principles of a graphical user interface for a scientific program and introduce the actual graphical user interface design for scaling. In conclusion, we present a methodology for taking a scientific program and developing a graphical user interface to make it more usable.

2. X-Ray Crystallographic Data Collection, Processing and Analysis

A detailed description of the data collection, data processing, and data analysis procedures in macromolecular crystallography is given in Drenth [1]. Figure 1 shows the different stages of a crystallography experiment as well as the finer steps within the Data Processing stage. The goal of a crystallographic experiment is to find the electron density map which characterizes the molecular structure of the given crystal. In order to calculate the electron density map though, one needs to find the structure factors and the phase angles. During the data collection stage, a set of two-dimensional detector images is collected and digitized as the crystal is oscillated. The data processing stage leads to the most accurate set of experimental structure factors necessary for mapping the electron density distribution inside the crystal. The data analysis stage uses the intensity measurements derived from the data processing stage, compare or combine them with those from related experiments to derive the phase information. Finally, during the structure determination stage of the experiment, the structure factors and the phase information are combined in a Fourier transform to calculate the electron density.
Data collection in macromolecular crystallography involves mounting a crystal on an X-ray diffractometer and taking successive measurements of the X-rays scattered by the crystal while the crystal is oscillated about an axis. The images, also referred to as frames, are recorded on a two-dimensional area-detector, digitized and transferred to a computer.

The images of the crystal diffraction feature X-ray reflections, or peaks, which carry two kinds of information: the relative positions and the intensities. From the relative positions of the peaks one can derive the dimensions of the crystal unit cell and its orientation with respect to the diffractometer axes, commonly referred to as the orientation matrix. The crystal orientation matrix allows the prediction of the reflection pattern on each image. The intensities of all reflections on all images are then evaluated, or integrated, usually using a peak profile-fitting technique. All frames of data need to be put on a common scale, so that intensities of reflections, recorded on different images, could be merged into a single data set. If a reflection is recorded more than once (redundant), its intensity should be averaged over all the measurements to increase the statistical accuracy of the data.

The three major stages in crystallographic data processing are:

1. Indexing and determination of the crystal unit cell and orientation matrix, using one or several images;
2. Theoretical prediction of the reflection positions on each image. Evaluation of the reflection intensities (peak profile fitting) on each image;
3. Scaling images and post-refinement of the crystal mosaicity and orientation matrix. Averaging
intensities of redundant reflections and merging all reflections into a single data set.

The first step, the indexing procedure, determines the unit cell parameters. It does so by first searching an image file to find the location of the centroids based on a user specified peak-to-noise ratio. Using these peaks, together with information about the X-ray source, crystal attributes and detector properties, the indexing program runs Fourier analysis to determine the dimension and orientation of the crystal unit cell. The theory is presented in a separate paper by Steller, Bolotovsky and Rossmann [2].

The integration step leads to the determination of the intensity values for the Bragg diffraction spots from each image. In principal, one uses the orientation matrix derived from the previous step to predict where the peaks should be. This set of orientation matrix can further be refined by minimizing the deviation between the predicted peaks and the observed peaks. Using the refined orientation matrix, one can predetermine the positions of the spots where the intensities should be integrated. The crystallographer can choose the size and shape of a region within which the integration is to be performed. The integration step should produce a file listing the Miller indices (hkl) of the reflections, the estimated intensities, the intensity errors for each image. The file should also contain the crystal oscillation range, orientation matrix, mosaicity and wavelength corresponding to the particular image. A detailed discussion of this step can be found in a paper by Rossmann [3].

The third and final step is commonly referred to as scaling. The scaling procedure looks for reflections, recorded on more than one frame, and, by comparing their intensities, derives the scale factor for every frame. The older versions of the scaling program used to take into account only the full reflections. However, with the advent of the crystal freezing technology, most reflections collected are partial rather than full. Therefore there is a need to design new scaling algorithms to make use of the partial reflections. The mathematical foundations for one such enhancement are presented in a paper by Bolotovsky, Steller and Rossmann [4]. The scaling algorithms, described in that article, have been implemented as a C program. A graphical user interface has been developed independently as a front end to the scaling program to make it easy for the user to specify the many control parameters needed by the program. This graphical user interface is the subject of interest for this paper.

3. Scaling and Merging Images

X-ray diffraction images produced by frozen biological macromolecules often have a large mosaic spread - defined to be the smallest angle through which the crystal can rotate about some axis while a reflection continues to be observable. Therefore, a scaling method must be capable of utilizing intensities from partial reflections as well as full reflections. A least-squares procedure, based on Hamilton, Rollett and Sparks [5], forms the basis of the scaling program. This procedure has been generalized to handle the partial reflections. Two methods are used to handle the partial reflections. One method allows the intensities of partial reflections from consecutive frames to be summed to obtain the full intensity. The other method uses a theoretical estimate of the reflection partiality to estimate the full intensity of a reflection. As mentioned before, the details of the theory behind the scaling program is presented in a separate paper [4].
From the user's point of view, the scaling program takes a set of integrated intensity files and a set of user directives as input and refines the scale factor, the B factor, the mosaicity and the orientation matrix for each image according to some user specified strategy. The final goal is to produce a merged reflection intensity file where the intensity and error estimate for each reflection are listed. The scaling program also produces an output file containing the statistics generated from the computations. This statistics can be used by the user of the scaling program to modify the directives and start another run to produce a data set of better quality.

The scaling program runs through 4 distinct phases. First, it sorts the individual integrated intensity files and combines the sorted files into a single file, called a combine file. This step requires certain knowledge about the crystal symmetry. The user needs to provide the crystal point group of symmetry. The specified point group implies a set of symmetry operators. If the user does not claim that the crystal had an anomalous scattering effect, the inversion operator will be added to the set of point group symmetry operators, making it a Laue group of symmetry operators. The symmetry operators are used to produce the symmetry-reduced Miller indices for each original Miller indices found in the integrated intensity file. The whole data set is then sorted on the reduced Miller indices, the original Miller indices and oscillation ranges of the frames. The result is the combine file.

After one run of the scaling program, it can produce a file that contains a list of reflections which are found to be outliers or discordant measurements, based on some user specified criteria. In subsequent runs of the scaling program, the user can specify the name of a file which contains a list of reflections to be rejected. The combine file will have to be reconstructed to exclude these reflections. The user can also specify the resolution limits within which to process the reflections. Finally, the values of the scale factor, the B factor and the mosaicity for each image are initialized. The user can edit these initial values if he/she so chooses.

The second phase of the scaling program uses the least-squares procedures to refine the scale factor, the B factor and the mosaicity for each of the images. The least-squares procedures are non-linear. All the refinable parameters can not be optimized in one step. To find the global minimum, the parameters should be refined in the proper order. Therefore, the user must define a strategy of refinement. For example, 3 default cycles are defined for the scaling phase. Only the scale factor is refined during the first cycle. The scale factor and the B factor are both refined during the second cycle. The scale factor, the B factor and the mosaicity are all refined during the third cycle. This strategy ensures that the most volatile parameter is refined first. In addition, the user can also set these parameters used in the least-squares procedures on a cycle by cycle basis - the number of iterations, whether to use method 1 or method 2 to interpret the partial reflections, the partiality tolerance if method 1 is chosen, the partiality limit if method 2 is chosen, the weighting schemes to be used in the least-square procedure, and finally the sigma cutoff which is used to reject certain insignificant intensities from being used in the scaling procedure.

The third phase of the scaling program does post-refinement which uses least-squares procedure to refine the crystal mosaicity and the orientation matrix. The orientation matrix is a combination of the crystal unit cell parameters and the setting angles with respect to the diffractometer axes. Again, the user must define a strategy for the post-refinement. He/She can define multiple cycles for the post-refinement operation. For each cycle, he/she can define exactly which parameters to
refine. The setting angles can be individually refined. The cell dimensions must be refined as a set. The mosaicity can also be refined during post-refinement. In addition, the user can specify the number of iterations to run through, the sigma cutoff, the weighting scheme to be used during the least-squares procedure, and whether all groups of redundant reflections should be used for the least-squares or only every n-th group. During the post-refinement, intensities of partial reflections are always corrected to estimate the full reflection intensities according to method 2. Therefore, the user must also specify the partiality limit which may bar the inclusion of low-partiality reflections.

The fourth phase of the scaling program performs the averaging of intensities of redundant reflections and the merging of the intensity information from all frames of data. Here again the user can specify the method used to utilize partial reflections, two methods being available. The user can also specify the sigma cutoff, the weighting scheme to be used for calculating the averaged intensities, the coefficients to be used in the statistical screening of outliers and discordant pairs, and whether to down-weight or reject the outliers while averaging.

After processing through these four phases, the scaling program produces a log file which contains many statistical tables to give the user insight into the quality of the scaled data. If rejections have been made, the scaling program produces a file containing a listing of the rejected reflections. The scaling program also produces an output file with the scaled and merged intensities for each unique Miller index available in the data. This file can serve as input to the data analysis stage of the X-ray crystallographic experiment where phase information is derived.

4. User Interface Design Principles

The description of the scaling step indicates that this is a complex iterative process which requires the specification of many control parameters. In designing the user interface for capturing these parameters, we observe the following user interface design principles formalized in Foley, van Dam, Feiner and Hughes (6):

- Reduce complexity
- Minimize memorization
- Prevent user errors and provide error recovery
- Provide visual feedback
- Be consistent
- Make use of the object-action paradigm

For example, to reduce complexity, we divide up the parameters logically according to the four phases of the scaling operation. The parameters that control one phase are placed into one folder. Each folder can be brought up to the forefront with a click of the folder tab. This hides the complexity of the overall process and allows the users to deal with parameters for one phase at a time. We also provide default values wherever they are appropriate so that the user only need to make occasional changes to the parameters instead of defining every parameter from scratch.

To minimize a user's memory load, we present a predefined list of choices in terminology familiar to the crystallographers whenever possible. For example, we present a list of the available crystal
systems for user selection. When the user has chosen the crystal system, there is an associated list of space group choices for that system. This allows the user to concentrate on running the scaling process rather than trying to remember the names the 230 space groups that are listed in the International Table. In addition to removing the burden of memory from the user, the predefined list also prevents the user from typing errors while entering the required information.

The object-action paradigm is a well established model in a desk top environment. The design of the scaling graphical user interface follows the paradigm so that a user can put his/her experience from other software package to immediate use. One would select an object and then click on a command button to invoke an action on the selected object. To prevent errors, a command button may be grayed out when no object is selected. For example, to defined a scaling cycle, one must define the parameters for one cycle. Since we provide a default definition of three cycles, the user may want to modify the parameters for one cycle. He must click on the row that contains the cycle to highlight it to enable the Edit command button so that the Edit action can be applied to the selected cycle.

The following section describes the details of the graphical user interface for the scaling program. The visual components we make use are: tabbed folders, list boxes, choice boxes, editable text fields, check boxes, multiple column tables and command buttons. These components provide the visual feedback and guidance for the user to construct a set of control parameters to run the scaling program.

5. Graphical User Interface for Scaling

5.1 Sort and Combine

The scaling program needs the following control parameters in order to process the sort and combine step:

- A set of the integrated intensity files.
- The initial values of scale factor, B factor and mosaicity for each file.
- The space group this crystal belongs to.
- The corresponding symmetry operators.
- The resolution limits.
- Whether there is anomalous scattering data to be processed.
- Whether some of the intensity files need to be reindexed.
- Whether to reject some reflections specified in a file produced by a previous run.

The scaling GUI presents the Sort/Combine folder for the user to specify these parameters. Figure 1 shows the layout of this folder. On the upper left hand corner is a listbox showing all files with the .x or .mx extension in the working directory. The .x files are the integrated intensity files generated by the DENZO package (7) which performs the integration step. The .mx files are the integrated intensity files generated by the MOSFILM package (8). Initially all .x and .mx files are selected for the scaling process. The user can toggle the "selected" status of a particular intensity file by clicking on the file.
Figure 2: Screen for the Sort/Combine control parameters.
The crystal system and the space group for the given crystal have usually been determined during the previous steps of the data processing stage. This information can be passed into the scaling GUI through a command line argument. If no such information is provided, the system defaults to Triclinic and the space group defaults to P1. Notice that the system selection and the space group selection are implemented with two inter-dependent choice boxes. When the user clicks on the choice button, the predefined list is displayed for selection. Once the selection has been made on the crystal system, the space group choice box is updated with those space groups that fall within the designated crystal system. When the user has specified the system and the space group, the corresponding symmetry operators are displayed in a text area where advanced users may choose to edit some of the operators.

By default, it is assumed that the user is not processing any anomalous scattering data. However, if the user indeed wants to process anomalous scattering data, the screen provides a check box labeled “Processing Anomalous Scattering Data” whose status can be toggled by clicking on the box. The resolution limits are also parameters which usually have been determined by the previous steps of data processing. This information can be passed to the scaling GUI in some file along with the crystal system and space group. If this information is not provided, a set of default values will be displayed.

Several command buttons are provided in the Sort/Combine folder. One command button is labeled “Define Reindex Group”. This function is discussed in a later section under “Advanced Features”. Another command button is labeled “View Point Group Symmetry Ops”. It can be pressed to display the corresponding Laue group symmetry operators which are used to generate the symmetry reduced Miller indices. A third command button is labeled “Reject Reflections Listed in File”. This button is useful in subsequent runs of the scaling program. Pressing this button brings up a dialog box through which the user can selected a list of rejection files. Rejection files contain reflections to be rejected based on some rejection criteria. It is written by the scaling program from a previous run. Finally, there is the command button labeled “Initialize” which can be pressed to display the initial values of the scale factor, the B factor and the mosaicity of each intensity file that has been selected. Figure 2 shows the screen displaying the initial values of the selected frames.

There is a default selection of the “middle” frame as the reference frame. A reference frame is defined to be the frame for which the scale factor and the B factor are fixed. If the user wants to select his own reference frame, or modify any of the initial values, he can click on the table entry to highlight it, click on the Edit command button to bring up the dialog box for editing the values for the selected frame. An “X” in the “Reference” column of the table marks the frame to be a reference frame. Figure 3 shows the dialog box for editing one set of initial values.
Figure 2: Screen displaying the initial values

Figure 3: Dialog box for editing one set of initial values
5.2 Scaling (Refinement)
The scaling program needs a set of strategies to process the refinement step. It will cycle through the least-squares procedure based on some directives set by the user. These directives are:

- How many cycles to go through?
- During one cycle, which parameters to refine?
- During one cycle, how many iterations to run through?
- During one cycle, which method to use for processing partial reflections?
- During one cycle, include every one of the symmetry reduced (h k l)? Or every nth?
- During one cycle, what is the sigma cutoff?
- During one cycle, which weight scheme to use?
- During one cycle, if method 1 is used to add up some partial reflections, what is the partiality tolerance? Partiality tolerance is defined to be the maximum allowed difference between the sum of the calculated partialities and 1. If method 2 is used to estimate the full reflection, what is the partiality limit? Partiality limit is defined to be the lowest partiality allowed in the least-squares procedure.

Figure 4 shows the screen layout of the refinement control parameters. 3 cycles are defined by default. During cycle one, only the scale factor is refined. During cycle 2, both the scale factor and the B factor are refined. During cycle 3, the scale factor, the B factor and the mosaicity are refined. Each cycle runs through 5 iterations, uses method 2 to estimate the full reflections from the partial reflections, applies a partiality limit of 0.05, and includes all groups of redundant reflections in the least-squares process.
Figure 4: Screen layout for the refinement parameters
If the user wants to modify the control directives for a given cycle, he can click on the cycle to select it and press the Edit command button to bring up the dialog box for editing. Figure 5 shows the dialog box for editing the refinement parameters for one cycle.

Figure 5: Dialog box for editing refinement directives

Note that a few of the directives have limited choices. These are presented in choice boxes such as the one for selecting a method for processing the partial reflections. The weight options are also implemented with a choice box where these three weight options are selectable:

- Apply unit weights
- Apply $\frac{1}{\text{sigma}_\text{Squared}}$ weights
- Apply calculated partiality/$\text{sigma}_\text{Squared}$ weights

Note also that if the user selects method 1, he will be prompted for the partiality tolerance, whereas if he chooses method 2, he will be prompted for the partiality limit.

Method 1 deals with the partial reflections by adding up the partial reflections that correspond to the same $(h k l)$. The user must define groups of frames for this operation so that only partial reflections from frames belonging to the same group should be added together. On the scaling folder there is a command button labeled "Add Partials Group" which can be used to define the groups. Figure 6 shows the dialog box that is displayed when the user presses the "Add Partials Group" button.
Group” command button. The angles Phi1 and Phi2 are displayed for user information. They are the starting and ending angles of the crystal oscillation during the data collection process. The fourth column of the table is used to display the group numbers. An entry of “-” indicates that the frame is in a group by itself.

![Dialog box for defining groups for adding the partial reflections](image)

Figure 6: Dialog box for defining groups for adding the partial reflections

If the user wants to define a group, he can click on one frame to highlight it and use it as the anchor. The user then moves down to the row which is to be the last frame in the group (one may have to scroll down the rows to get to the frame) and presses down on the SHIFT key while clicking the mouse button. This action highlights the entire block of frames. The user then presses the command button labeled “Define Group” to define the block of frames to be in one group. Figure 7 shows the selection of frames s1300.x through s1310.x to be in one group.
Another command button of interest in the Scaling folder is the "Constrain Mosaicity" button which is used to define groups of frames where the mosaicity value is constrained to be the same for all frames within one group during the refinement. The steps for defining groups to constrain mosaicity are the same as those for defining groups for adding partials.

5.3 Post-Refinement
The scaling program needs a set of strategies to process the post-refinement step. It will cycle through the least-squares procedure based on some directives set by the user. These directives are:

- How many cycles to go through?
- During one cycle, which parameters to refine?
- During one cycle, how many iterations to run through?
- During one cycle, include every one of the symmetry reduced (h k l)? Or every nth?
- During one cycle, what is the sigma cutoff?
- During one cycle, which weight scheme to use?
- During one cycle, what is the partiality limit?

Figure 8 shows the screen layout of the post-refinement control directives. One cycle has been defined by default. During this cycle, the rotation angle around the X axis and that around the Y axis will be refined. The cell dimension and the mosaicity will also be refined. This cycle runs through 5 iterations, uses method 2 to estimate the full reflections from the partial reflections, set the partiality limit to 0.05 and includes every one of the symmetry reduced Miller indices. It uses a sigma cutoff of 2 and applies the 1/sigma_squared weights to the least-squares procedure.
Figure 8: Screen layout for the post-refinement control parameters
For post-refinement, the user can define groups to constrain mosaicity. He can also define groups for constraining setting angles. And finally he can define groups to constrain cell dimensions. All three groups are independently defined. Again, the steps of defining groups are similar to those described for defining groups for adding partial reflections. Figure 9 shows the dialog box for editing/defining one cycle for post-refinement:

![Dialog box for editing/defining post-refinement parameters](image)

**Figure 9: Dialog box for editing the post-refinement parameters**

### 5.4 Merging (Averaging)

The scaling program needs the following control parameters to run the averaging/merging step:

- Which method to use in processing the partial reflections?
- If method 1, what is the partiality tolerance? If method 2, what is the partiality limit?
- Which weighting scheme?
- Which outlier weighting option?
- What is the rejection criterion?
- What is the sigma cutoff?
- Which coefficients to use in the statistical rejection of outliers?
- Which coefficients to use in the statistical rejection of discordant pairs?
Figure 10: Screen layout for the averaging/merging control parameter
Figure 10 shows the layout of the screen which presents the averaging parameters. Note that default values are provided for each one of the parameters. Method 2 is used to process partial reflections. Partiality limit is set to 0.05. 1/sigma_squared weights are applied. The outliers are down-weighted. No reflections are rejected. The sigma cutoff is set to -3. The coefficients used in the statistical rejection of outliers are 9.0, 0.05, 0, 0 and 1 respectively. The coefficient used in the statistical rejection of discordant pairs is 9.0.

5.5 Output Parameters

The scaling program collects many statistics automatically. For example, there are the reflection statistics with respect to intensities, the reflection statistics with respect to resolutions, R factor statistics for symmetry operators, R factor statistics for special reflections and so on. The reflection statistics are collected into bins. The default number of bins is set to 10. This can be modified by the user in the output/statistics folder.

Figure 11 shows the screen layout for the control parameters that effect the output of the scaling program. The only statistics that is not collected automatically is the frame vs. frame statistics. Since the user may be attempting to scale hundreds of frames at the same time, collecting this statistics may be time consuming. It is therefore left to the user to decide whether to collect this information. A check box labeled “Collect Frame-vs-Frame Statistics” is provided.

The scaling program generates four output files. One file contains the scaled and averaged intensity for each (h k l) reflection. A second file contains the log of the scaling computations. This file would contain the diagnostic messages if any error occurred during scaling. It also contains the statistics compiled during scaling. The third file contains a list of the rejected reflections. The fourth file contains the outlier analysis statistics which is kept in a separate file due to its large volume.
Figure 11: Screen layout for output control parameters
5.6 The Utility Functions

After browsing through all five folders, the user will note that all control parameters have default values associated with them. The minimum action that the user has to take is to check the selected frames (remember all .x and .mx frames in the working directory are displayed and selected) and make sure that they should be scaled together. If he chooses to change the selections by removing some frames from consideration, then he must press the “Initialize” command button to reinitialize the scale factors, B factors and mosaicities for the selected frames. The user must also define the add partials groups if he chooses to apply method 1 to the partial reflections. Other than these operations, if the user is satisfied with all the other parameters, then he can press the “Run” command button to execute the scaling program using the current control parameters.

In addition, we provide the “Load” and “Save” functions so that the user can save a set of control parameters into a file and later load the parameters from the file. This allows the user to start with a template which he may prefer over the default values that are provided. Making minor edits in some of the control parameters after loading from a previously saved parameter file should make it much easier to run another set of refinement and post-refinement.

5.7 Advanced Features

Occasionally a group of integrated frames may need to be reindexed. The scaling GUI provides the command button labeled “Define Reindex Group” so that the user can define the groups of frames that need reindexing along with the corresponding reindex matrix. Figure 12 shows the dialog box displayed when the “Define Reindex Group” button has been pressed.
To define a group of frames to be reindexed, the user clicks on one frame to highlight it and use it as the anchor, he then moves down to the last frame that should be in the group and press down the shift key while clicking the button to define the end of the group. The entire block of selection will be highlighted. The user can press the "Define Group" command button to define the selected frames as belonging to one group. Figure 13 shows that frames s1300.x through s1305.x are in group 1. The user will also be prompted to enter the reindex matrix. By default the identity matrix is provided.

![Dialog box showing one group of frames has been defined](image)

Figure 13: Dialog box showing one group of frames has been defined

6. Output Presentation

The scaling program generates many statistical tables for user information. After one run of the scaling program, the user can view these tables by pressing the "View Output" command button. This button brings up a menu of many tables which can be viewed. Figure 14 shows the output menu.
For example, if the user clicks on the "View" button next to "Scale Factor vs Frame Number", he will get a table listing the refined scale factors as well as a graphical representation of the table. The tabular presentation of a sample run is given in Figure 15.
The table shows the refined results of scale factor, B factor and mosaicity for each frame as well as the error and the shift with respect to these values. The shift measures the difference between the refined result of two successive iterations, while the error is the estimated standard deviation of the value of the refined result. The corresponding graph in Figure 16 shows the scale factors versus the frame numbers.
7. Software Requirements and Installation

The scaling program is written in the standard C language. It has been shown to execute properly on an SGI workstation, a DEC workstation, and an IBM workstation. Porting to other UNIX environment may involve only recompilation. However, it is up to the user to verify that the ported version executes properly. The scaling program takes a complex control input parameter file and runs through the least-squares refinement procedure according to the specified strategies. A user can type in the control parameters by hand. Or, he can use the graphical user interface to specify these control directives.

The graphical user interface for the scaling program is developed in Java (9) using JDK 1.0.2. The reasons for choosing Java are its portability, the easy prototyping and the potential to execute across the Internet. Due to the large number of control parameters needed for scaling, these parameters are grouped into logical sets according to the different phases of the scaling program. Tabbed folders proved to be a space saving widget which makes the screen design more compact. The main Java class for the graphical user interface is the one that handles the processing of the events that occur in the Scale Window. The name of the class is ScaleWindowEvents. To invoke the scaling graphical user interface, one therefore runs the java interpreter on the ScaleWindowEvents class:

```
java ScaleWindowEvents
```

Since the scaling process follows the cell determination and intensity integration steps, it is more
than likely that the crystal system and space group have already been determined. The user can invoke the graphical user interface by running:

```
java ScaleWindowEvents init.param
```

where `init.param` would be a file that contains the initialization information for the crystal. A sample initial parameter file may look like the following:

```
Crystal_System_Name   Cubic
Space_Group_Name      I213
Resolution            1.90
```

The user interface should execute on any platform where JDK 1.0.2 has been installed. Please note though that JDK 1.0.2 on different platforms may behave differently.

The scaling graphical user interface makes use of a Java Two Dimensional Graph Package developed by Leigh Brookshaw (10) as well as a MultiColumnListbox widget developed at Taligent, now part of IBM (11).

The class files can be downloaded from the following web page:

http://www-structure.bio.purdue.edu/~viruswww/Rossmann_home/dps/download.shtml

8. Methodology Review

Adding a graphical user interface to a complex scientific program is a gratifying undertaking. It presents a front end to a program which is often essential in some scientific experiment but daunting for a novice user. For example, the scaling program requires the specification of some fifty or so control parameters. Some of these parameters are intimately tied to the mathematical algorithm used in the processing. The user may be quite at a loss to provide an initial value. The number of parameters also tend to increase as the program migrates toward general use by students as well as researchers.

Often, such a program is executed within a context. That is, the program is part of a collection of programs that make up the complete data processing or analysis. It may depend on output from a preceding program. It may also generate data which the succeeding program must use. The design of the user interface should take into consideration the issue of consistency. If a proper tool is chosen, one can build a synergic environment which is aware of the multiple programs which form the data processing package.

To build a user interface front end for such a program, one must have a fairly clear understanding of the major logic flow of the program as well as complete knowledge of the input and the output parameters. Understanding the process can help to define the logical grouping of parameters. Designing the user interface then involves the effort of selecting the most effective graphical component to capture a parameter and finding the most natural and logical way to prompt for that parameter from the user. However, before one can choose a graphical component, one must first
determine which language and/or graphics tool to use.

We chose to develop the user interface for the scaling program using the Java language and the Java Development Kit, JDK. A major part of the decision was to achieve synergy within the data processing stage so that scaling can be executed from an environment where the indexing step and the integration step can also be executed. The decision was also based on the premise that the user interface will run as an applet across the Internet so that crystallographers from all over the world can access the interface. An applet is defined as a "limited" Java application which can be downloaded from a server and executed on a local machine through the Internet.

The original design called for a Java applet which runs as a Data Processing Manager. The different data processing programs such as the indexing program, the integration program and the scaling program all have their own user interface for collecting the input parameters. When the input parameters have been established, then the actual data processing program can be started, perhaps on a remote workstation somewhere on the network, to run that phase of the data processing. If output is expected from the program, then the Data Processing Manager is responsible for collecting the output and writing the information into a central database so that it can be piped into the succeeding program.

However, because the ownerships of the different data processing programs are spread out among different crystallography laboratories, we could not get a consensus to proceed with the Data Processing Manager concept. The user interface design work became more narrowly focused to be just a Java application that serves as the front end of the scaling program. When one takes away the Internet connectivity, then Java, used strictly as a tool to develop graphical user interfaces, is found somewhat lacking in many of the fancy widgets that make a user interface truly friendly yet stunning. This is due to the fact that Java is a relatively young language. One example of Java’s immaturity is the shortage of anything but the most basic graphical widgets. We use a multiple column list box to store parameters for many refinement cycles in the form of a table. However, it could not handle the direct editing of individual cells. The information had to be collected into a dialog box before editing can commence. This lack of direct manipulation capability makes it difficult to satisfy some usability requirements.

In conclusion we believe that the methodology of adding a graphical user interface to a scientific program should include the following:

- Wait for the program parameters to stabilize
- Understand the program’s input and output requirements
- Understand the program’s logical flow
- Understand and define the program’s operating environment
- Select the proper language and tool
- Define the usability requirements
- Follow through by prototyping and gathering user feedback
- Ensure consistency between the program’s user interface and that of the preceding and succeeding programs
9. References