BUTTERWORTH HEINEMANN

A visual data flow environment for macromolecular crystallographic computing

D.L. Wild,* P.A. Tucker,† and S. Choe*

*Structural Biology Laboratory, The Salk Institute, La Jolla, California, †European Molecular Biology Laboratory, Heidelberg, Germany

This article describes the integration of programs from the widely used CCP4 macromolecular crystallography package into a modern data flow visualization environment (application visualization system [AVS]), which provides a simple graphical user interface, a visual programming paradigm, and a variety of 1-, 2-, and 3-D data visualization tools for the display of graphical information and the results of crystallographic calculations, such as electron density and Patterson maps. The CCP4 suite comprises a number of separate Fortran 77 programs, which communicate via common file formats. Each program is encapsulated into an AVS macro module, and may be linked to others in a data flow network, reflecting the nature of many crystallographic calculations. Named pipes are used to pass input parameters from a graphical user interface to the program module, and also to intercept line printer output, which can be filtered to extract graphical information and significant numerical parameters. These may be passed to downstream modules, permitting calculations to be automated if no user interaction is required, or giving the user the opportunity to make selections in an interactive manner.

Keywords: AVS, CCP4, dataflow toolkits, macromolecular crystallography, graphical user interfaces, visualization

INTRODUCTION

X-Ray crystallography is the primary experimental technique for obtaining the atomic coordinates on which molecular visualization depends. In the macromolecular field, rapid progress has been made in the last two decades, driven partly by advances in the instrumentation and computer power available to crystallographers.[1] The user interface to macromolecular crystallographic software has, however, been neglected until fairly recently. Much of the crystallographic software used for macromolecular structure determination still retains the "card image" input format common in the early 1970s when it was first written, although, as Frenz[2] points out, "mainstream" software has since advanced to dialog, menus, and windows as better means of user–computer interaction. A number of projects have been undertaken either to develop completely new packages based on the X-window system,[3] or to develop X-window-based interfaces to existing crystallographic software.[4,5] This article describes an alternative approach to providing a graphical user interface to the Collaborative Computing Project 4 (CCP4) suite of programs for macromolecular crystallography, which have been widely used since the establishment of CCP4 in 1979.[6]

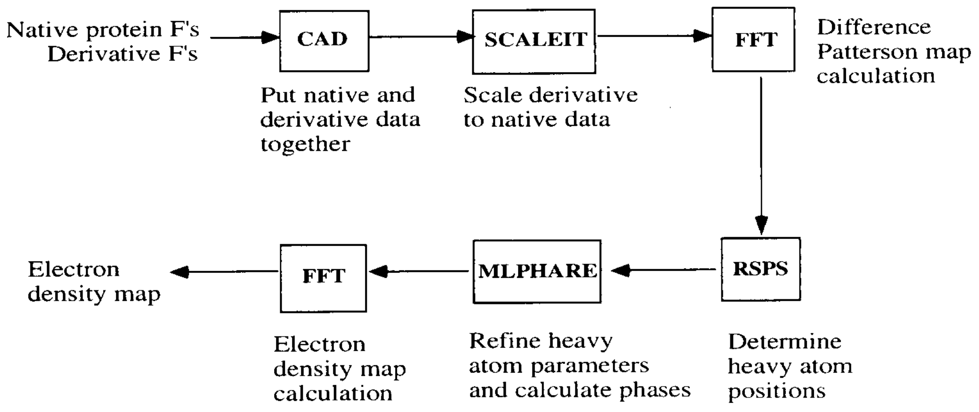
Unlike some other systems for macromolecular crystallography, which attempt to encompass all the required functions into one monolithic program, the CCP4 suite comprises a number of separate programs, written by different authors in standard Fortran 77, which communicate via common formats for structure factor, electron density map, and atomic coordinate data files. The suite contains programs that cover most aspects of macromolecular structure determination: data reduction, phasing by isomorphous and molecular replacement, least-squares refinement, and structure analysis. Although it is beyond the scope of this article to describe these procedures in detail, and the interested reader is referred to the many excellent texts on macromolecular crystallography,[7–10] Figure 1 illustrates the main steps involved in solving a crystal structure by isomorphous replacement, and the principal CCP4 programs used. A complete list of programs comprising the suite is given in Ref. 6.

An important feature of the CCP4 suite is that individual programs may be combined in different ways to accomplish

Color Plates for this article are on pages 299–300.

Address reprint requests to Dr. Wild at the Structural Biology Laboratory, The Salk Institute, 10010 N. Torrey Pines Road, La Jolla, California 92037.

Received 5 June 1995; revised 14 July 1995; accepted 22 July 1995.



Native protein F's
Derivative F's
CAD
Put native and derivative data together
SCALEIT
Scale derivative to native data
FFT
Difference Patterson map calculation
RSPS
Determine heavy atom positions
MLPHARE
Refine heavy atom parameters and calculate phases
FFT
Electron density map calculation
Electron density map

Figure 1. The main steps involved in the solution of a crystal structure by multiple isomorphous replacement using the CCP4 suite, together with the principal CCP4 programs used (boxed). In practice, some of these steps may involve iterative procedures and subnetworks of programs.

particular tasks. Input for each program is provided by a series of "control cards" comprising a leading keyword plus additional keyword/value pairs or numerical parameters. Because many crystallographic calculations involve a series of steps, using different programs, these are often chained together in a script or command file. An example, taken from the CCP4 distribution, is given in Figure 2 of a script to search for locations of possible water molecules around a protein by looking for peaks in the difference ($F_{obs} - F_{calc}$) map that lie within 3.5 Å of atoms in the protein molecule. A number of CCP4 programs are used: *sfcalc* (calculate structure factors from atomic coordinates), *rstats* (scale F_{calc} to F_{obs}), *fft* (calculate difference map), *peaksearch* (search for peaks in difference map), and *watpeak* (locate peaks within certain distance of protein atoms). The final output of the job is a pdb format file containing the coordinates of the selected peaks; the intermediate files are needed only for communication between the various programs in the pipeline.

As Evans[11] points out, the advantage of this type of loose organization is that it is easy to add functionality or modify programs without interfering with the rest of the suite. Disadvantages include lack of consistency between programs, both in programming style and user interface, and the fact that the user must decide which operation to perform next. The latter disadvantages may be overcome by integrating the individual programs into an environment that presents the user with a consistent graphical user interface (GUI), and provides a number of prepackaged scripts for common tasks, while retaining the flexibility that allows the user to recombine programs in new ways. The approach presented in this article uses a modern data flow visualization environment to perform these tasks. Although one particular environment (AVS version 5.02[12]) has been used, the approach described should be equally applicable to other data flow visualization environments on the market.

DATA FLOW VISUALIZATION ENVIRONMENTS

Data flow visualization environments such as AVS, Data Explorer,[13] Iris Explorer,[14] and Khoros[15] allow modular software components to be connected together into executable data flow networks or directed acyclic graphs to construct a data visualization application. Each node (or module) in the graph represents an operation on the data and each directed arc represents a path over which data

```
############################################################
#  Add Fc and Phic to the reflection file

sfall XYZIN toxd.pdb HKLIN toxd_phase_mir.mtz \
    HKLOUT toxd_mir_fc.mtz <<-eof-sfall
        TITLE Fc and Phic calculation from toxd.pdb
        MODE SFCALC HKLIN XYZIN
        RESO 10.0 2.7
        GRID 100 60 32
        SFSG 19
        LABIN FP=FTOXD3 SIGFP=SIGFTOXD3
        LABOUT FC=TOXFC PHIC=TOXPHIC
        END
eof-sfall

# Scale the Fc's to the Fo's

rstats HKLIN toxd_mir_fc.mtz HKLOUT toxd_fofc.mtz <<-eof-rstats
        LABIN FP=FTOXD3 SIGFP=SIGFTOXD3 FC=TOXFC PHIC=TOXPHIC
        TITLE Fc scaled to Fo
        RESO 10.0 2.7
eof-rstats

# Calculate the difference map for an asymmetric unit

fft HKLIN toxd_fofc.mtz MAPOUT toxd_fofc.map <<-eof-fft
        LABIN F1=FTOXD3 SIG1=SIGFTOXD3 F2=TOXFC PHI=TOXPHIC
        TITLE Toxd Fo-Fc difference map using FTOXD3 data
        END
eof-fft

# Extend the difference map to cover the molecule - 4 Ang

extend MAPIN toxd_fofc.map MAPOUT toxd_fofc_ext.map \
  XYZIN toxd.pdb <<eof-extend
        border 4
eof-extend
# Find position of significant peaks (more than 3.5 sigma) in extended map

peakmax MAPIN toxd_fofc_ext.map XYZOUT toxd_fofc.peaks <<-eof-peakmax
NUMPEAKS 200
THRESHOLD RMS -3.5 NEGATIVES
eof-peakmax

# Which residues are near (within 3.5 Ang) these peaks?

watpeak XYZIN toxd.pdb PEAKS toxd_fofc.peaks XYZOUT toxd_wats.brk <<-eof-
atpeak
        TITLE Peak positions using FTOXD3 data and final pdb
        DISTANCE 3.5
eof-atpeak
```

Figure 2. The CCP4 script for a calculation to locate possible water peaks in a difference Fourier map.

flows.[16,17] Data appearing on the output port of one module are passed to the input port of one or more downstream modules. Typically, an operation on the data is executed if and only if all the required input values to the particular module have been presented. Data flow networks are built interactively using a visual programming interface, by choosing icons representing individual modules from a palette and connecting them together in a workspace. Graphical user interfaces for individual modules or complete networks of modules may be constructed with the help of built-in user interface builders. In addition to a number of built-in modules, repositories of user-written modules are available for most of the environments listed above (for AVS, this is the International AVS Center [IAC], ftp address; avs.ncsc.org, 128.109.178.38).

The use of data flow toolkits in molecular visualization has been described by a number of authors,[18–20] where the graphical rendering tools provided by these environments have been used as an alternative to a traditional program written with graphics libraries. In addition to being useful for visualization, the data flow model also reflects the nature of crystallographic computing using a loosely organized package such as CCP4. Each module represents a single CCP4 program, and networks of programs may be constructed to perform particular tasks. These networks are equivalent to the scripts or command files of traditional batch-oriented processing. The data passed between modules in our implementation are usually strings describing the names of input and output files, which have been retained as the means of interprogram communication. Figure 3 shows an AVS network constructed to perform the same task as the script shown in Figure 2.

METHODS

An approach to the modernization of "legacy" Fortran programs with card image input has been described by Albury.[21] The program can be considered as a "compute object" in a client–server environment. It receives a series of "messages" (input cards), processes the requests according to some algorithm, and replies with "response messages" (lines to an output file). On the basis of this approach, a number of CCP4 programs have been integrated into the AVS environment through the use of named pipes as an interprocess communication mechanism under the Unix operating system.

Each CCP4 program is encapsulated into an AVS "macro module," which, in turn, comprises a network of lower level modules and connections. A macro module appears in the module palette as a single module icon, allowing an encapsulated program to be used individually or linked to other modules in a data flow network. On-line help on each macro module may be obtained by clicking on the "dimple" on the right-hand side of the module icon. This displays a standard Module Editor window, which contains a help button. Help text based on the CCP4 "man" pages may be displayed in a scrollable Help Browser window by clicking on this button.

Each macro module is made up of six basic linked modules, as shown in Figure 4. An interface module provides the GUI, and passes input parameters from the on-screen widgets as strings to an output port. A combination of two public domain modules (*create_unix_pipe* and *string_to_file*, obtainable from the International AVS Center) create a named pipe and pass the strings output by the GUI to this pipe. The Fortran CCP4 program is incorporated into an AVS module and opens the named pipe as its standard input (unit 5). Only minor changes to the CCP4 source code were necessary, because file names are now passed as subroutine parameters from the AVS "wrapper" routine, rather than being picked up from the command line.

Command strings are thus passed from the GUI via the named pipe to the Fortran program. A further pair of modules creates another named pipe (opened by the CCP4 For-

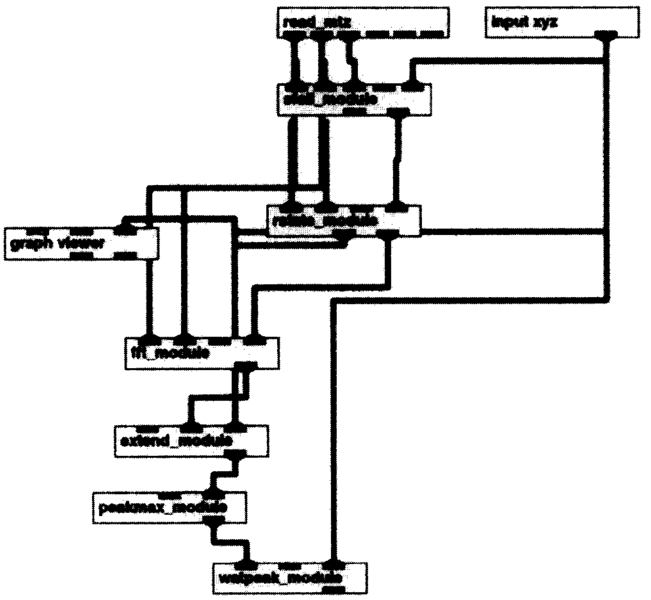


Figure 3. An AVS network constructed to perform the same calculation as the script shown in Figure 2. The graphical user interface for this network is shown in Color Plate 1.

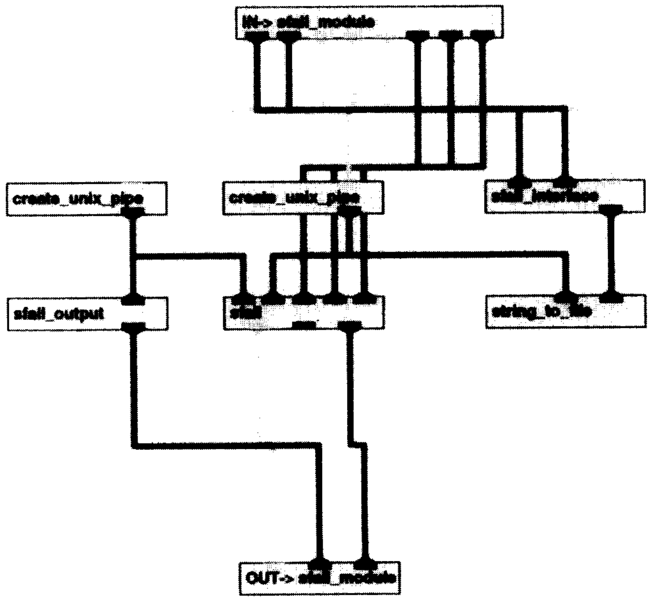


Figure 4. The internal structure of an AVS/CCP4 macro module.

tran program as standard output [unit 6]), and allow an ''output filter'' module to read the line printer output strings produced by the CCP4 program. Additional connections allow parameters to be passed into and out of the macro module.

EXAMPLES

Each macro module representing a single CCP4 program has its own attached graphical user interface, which is presented when the macro module is pulled into the AVS workspace. An example is shown in Figure 5 for the program *sfall* (structure factor calculation). A certain amount of dynamic behavior may be programmed into the user interface; related parameters may be grouped together and displayed only when certain radio buttons or toggles are pressed. The AVS Layout Editor tool gives the user the possibility of

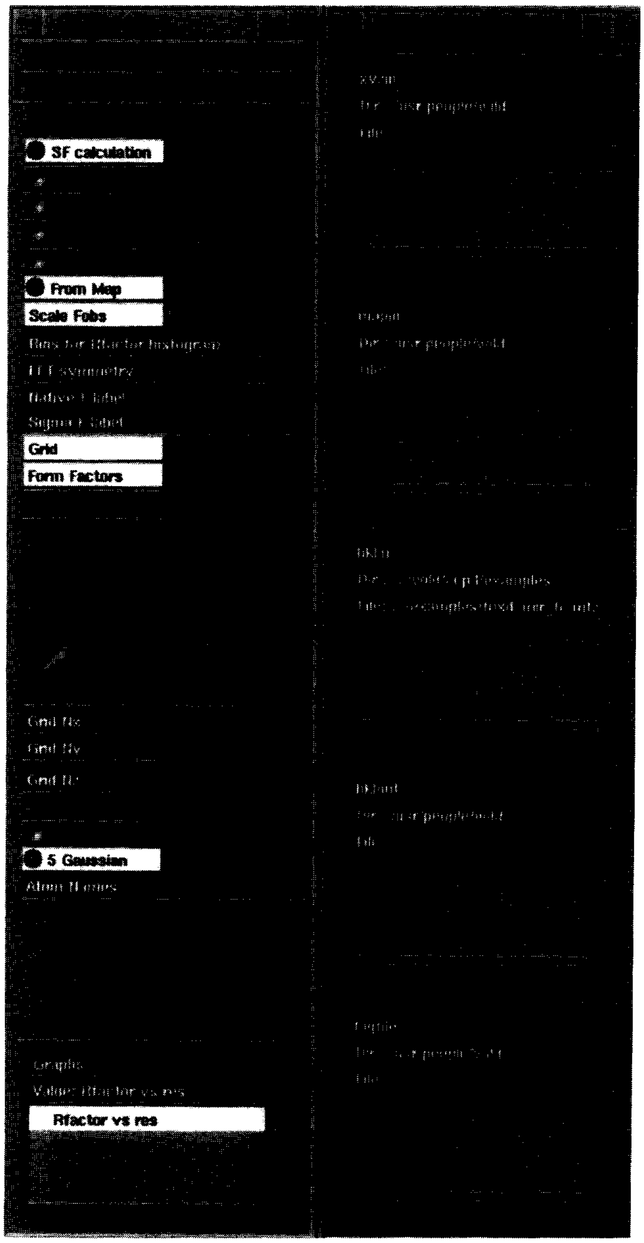


Figure 5. The user interface for the program sfall.

reconfiguring the position of widgets in the interface, or even of changing the type of widget, for instance from a dial to a slider. The Layout Editor also allows composite user interfaces to be built for networks of modules. A user interface for the network shown in Figure 3 is presented in Color Plate 1. This composite interface uses elements from the interfaces of the individual programs in the network, while hiding those parameters that are repeated or unnecessary.

One feature of the CCP4 structure factor file format (MTZ) is that reflection data are stored as columns of real numbers, each referred to by an alphanumeric label (the ''data label''). Programs using these data require the user to assign these ''data labels'' to a set of ''program labels'' by the use of the LABIN statement (see Figure 2). The GUI presented in Color Plate 1 allows the user to make these assignments in a point-and-click fashion. The *read_mtz* module opens the MTZ file and reads the ASCII header, which contains the data column labels. These are then presented in a scrollable ''choice browser'' widget (labeled ''Data Label'' in Color Plate 1), together with a second choice browser containing a list of the possible data items. By clicking on a choice in the ''Data Item'' browser, followed by a choice from the ''Data Label'' browser, the user may set up the necessary correspondences between these items. In the network shown in Figure 3, the chosen data labels are then passed as the input parameters ''Native F label'' and ''Sigma F label'' for the three modules that require this input: *sfall, rstats,* and *fft*. Because these data are repeated, the user interface presents only one ''typein'' widget for each of these items, which are then passed to all the modules that require them. The data labels chosen in the choice browser widget are immediately displayed in these ''typein'' widgets.

While the AVS Network Editor and Layout Editor provide a flexible development environment for the construction of networks and composite user interfaces, they may not be appropriate for the end user. The AVS Command Language Interpreter (CLI) may be used to construct an application in which these elements of the development environment are hidden and the end user presented with a pull-down menu bar from which the GUIs for particular task-based networks can be displayed.

As mentioned above, the function of the GUI is not only to simplify program input, but also to present program output to the user. Line printer output strings, read by the ''output filter'' module, may be displayed directly in a text browser widget on screen (as in Color Plate 1), or filtered to extract summary information, which can then be presented to text widgets in the GUI. The output from a number of CCP4 programs already has tabular statistical information marked with keywords to allow extraction and display as graphs by the *Xloggraph* utility, which is distributed with the suite. Graphical information from these or other unmarked tables may be extracted from the output by the ''output filter'' module and passed directly to the AVS Graph Viewer, providing visual monitoring of the progress of the calculations. An example is also shown in Color Plate 1, where graphical output from the *rstats* program (scaling of $F_{\rm calc}$ to $F_{\rm obs}$) is displayed in the AVS Graph Viewer window.

A further possibility is the extraction of significant numerical parameters from the output. These may be passed from the output port of the filter module to the input ports of downstream programs that require them as input, allowing calculations to be automated if no user interaction is required, or giving the user the opportunity to make selections in an interactive manner.

An example is given in Figure 6, which shows the network for a molecular replacement calculation using the program *Amore*.[22] Here the network consists of connected instances of the same module, *Amore*, where each instance of the program calls a different function (from top to bottom: *Sorting* [packs and sorts reflection data], *Tabling* [transforms coordinates and prepares table of continuous Fourier coefficients], *Rotation* [calculates rotation function], *Translation* [calculates translation function], *Translation* [calculates translation function with some molecules fixed to search for extra molecules], *Fitting* [performs rigid-body refinement for selected solutions of translation search], and *Reorientation* [applies appropriate rotation and translation parameters to initial model]). For instance, information contained in the output from the *Tabling* step is used in later stages of the calculation, either to set default values (in the *Rotation* step), or as input parameters (for example, Center of Mass and Eulerian Angles in the *Reorientation* step). These numbers are extracted from the program output during the *Tabling* step, and passed via the output port of the *Amore* macro module to the downstream modules that require them. Repeated input parameters, such as the minimum and maximum resolution, are also passed downstream. Figure 7 shows the user interface for the translation function step. The solutions from the rotation function, expressed as Eulerian angles, are extracted from the output of the rotation function step and displayed in a "choice browser" widget, where they may be selected by clicking on the appropriate line. The selected strings are passed to the next module and displayed in a text "typein" widget, where they may be added to the input for the translation function calculation by clicking on the "Accept Solution" button. A similar procedure may be followed for the solutions of the translation function calculation.

A further advantage of integrating the CCP4 programs into an environment such as AVS is that a number of prepackaged and user-written tools are available for the visualization of two- and three-dimensional (2D and 3D) data types. The results of crystallographic calculations, such as electron density and Patterson maps, may be visualized with these graphical tools; for example, as 3D solid or wireframe isosurfaces or as 2D contoured sections, and combined with renderings of the molecular structure. Crystal Eyes stereo is supported by AVS on Silicon Graphics platforms. In Color Plate 1 the positions picked from the difference Fourier map are displayed as red spheres around a stick figure of the protein. A number of user-written modules are available from the IAC that facilitate the interactive display of any

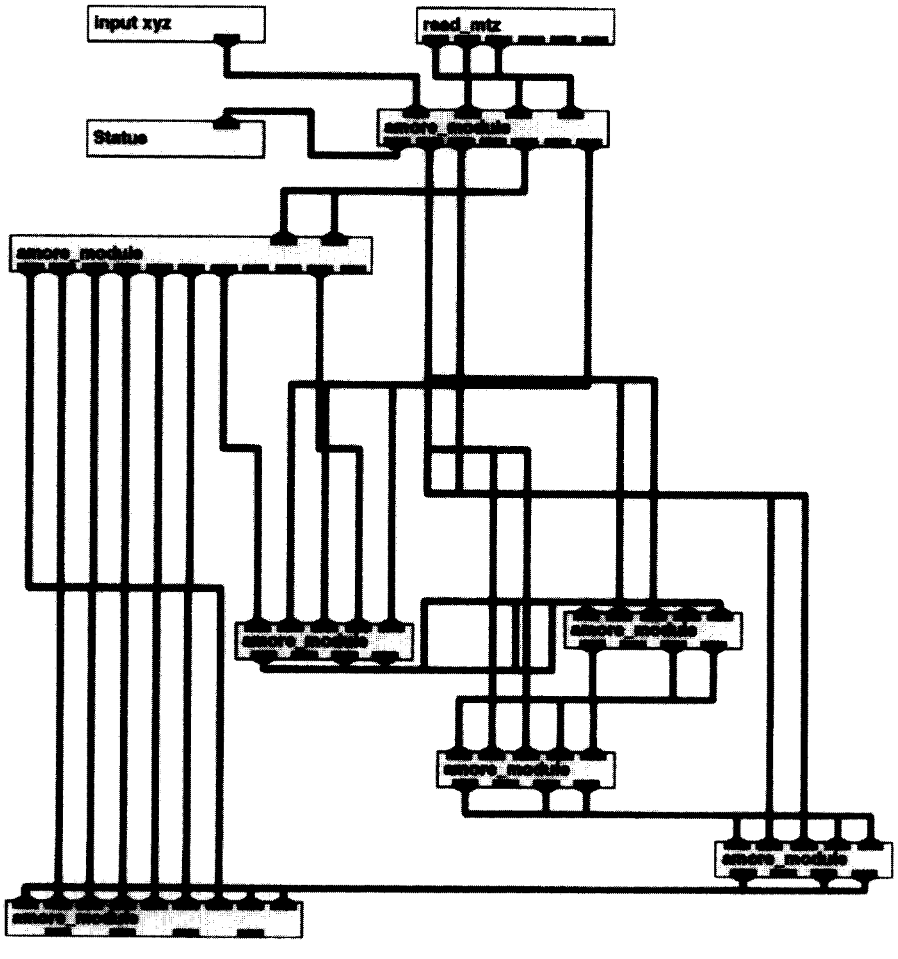


Figure 6. An AVS network for molecular replacement calculations using the CCP4 program Amore.

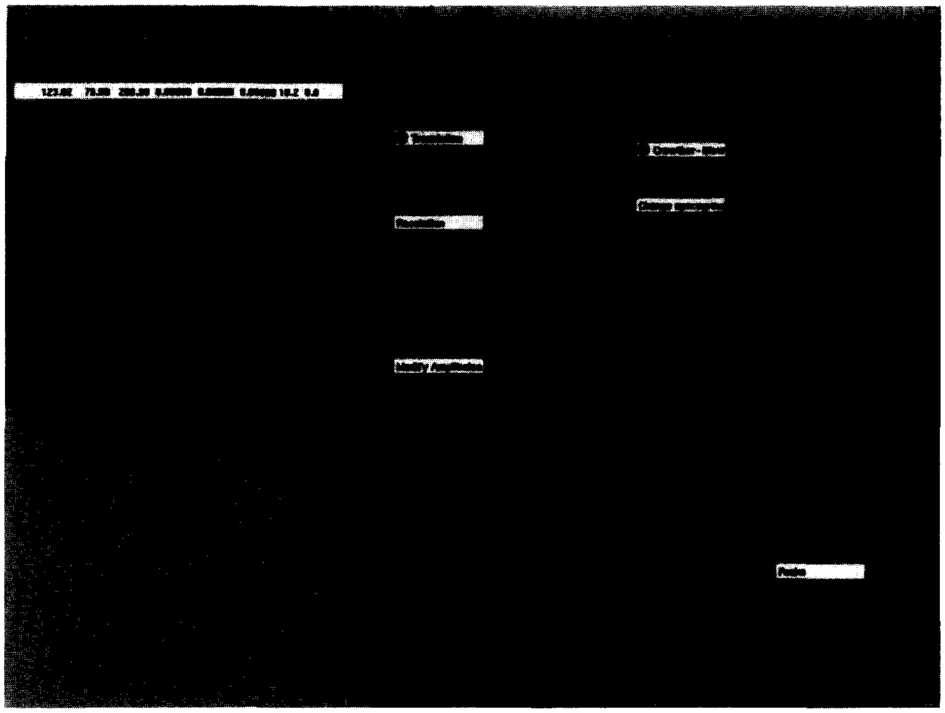


Figure 7. The user interface for the translation function step of Amore, together with the solutions output from the rotation function step.

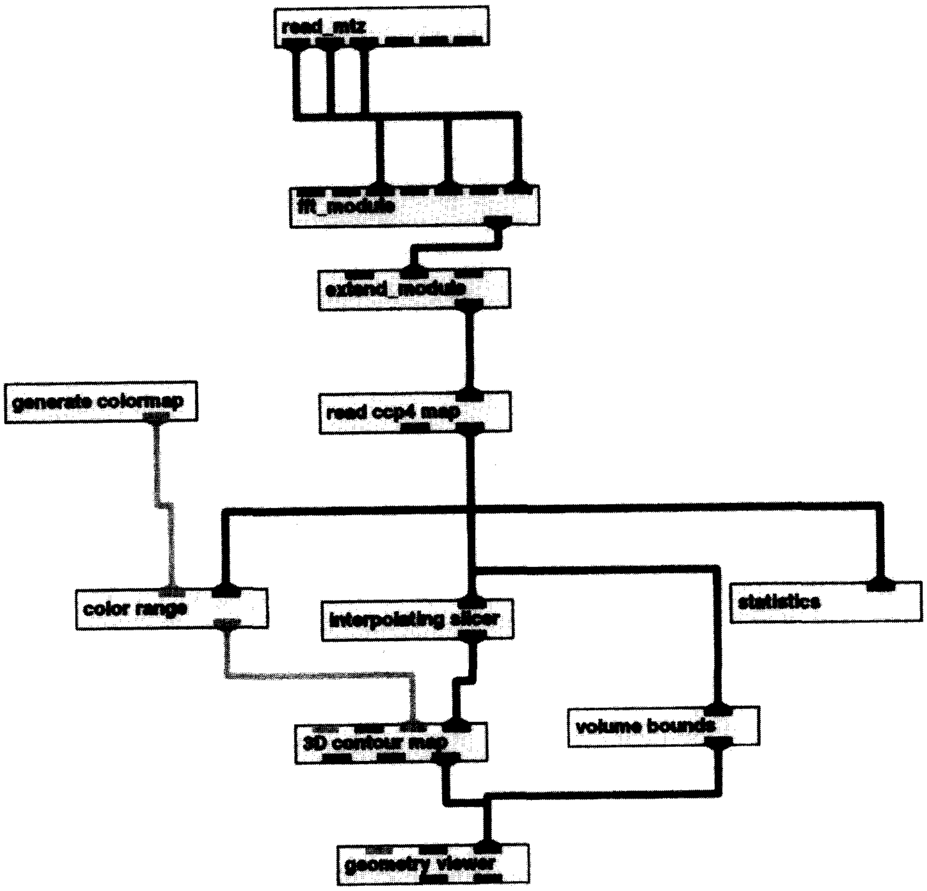


Figure 8. An AVS network to calculate Patterson or electron density maps from an MTZ file, and contour and display sections.

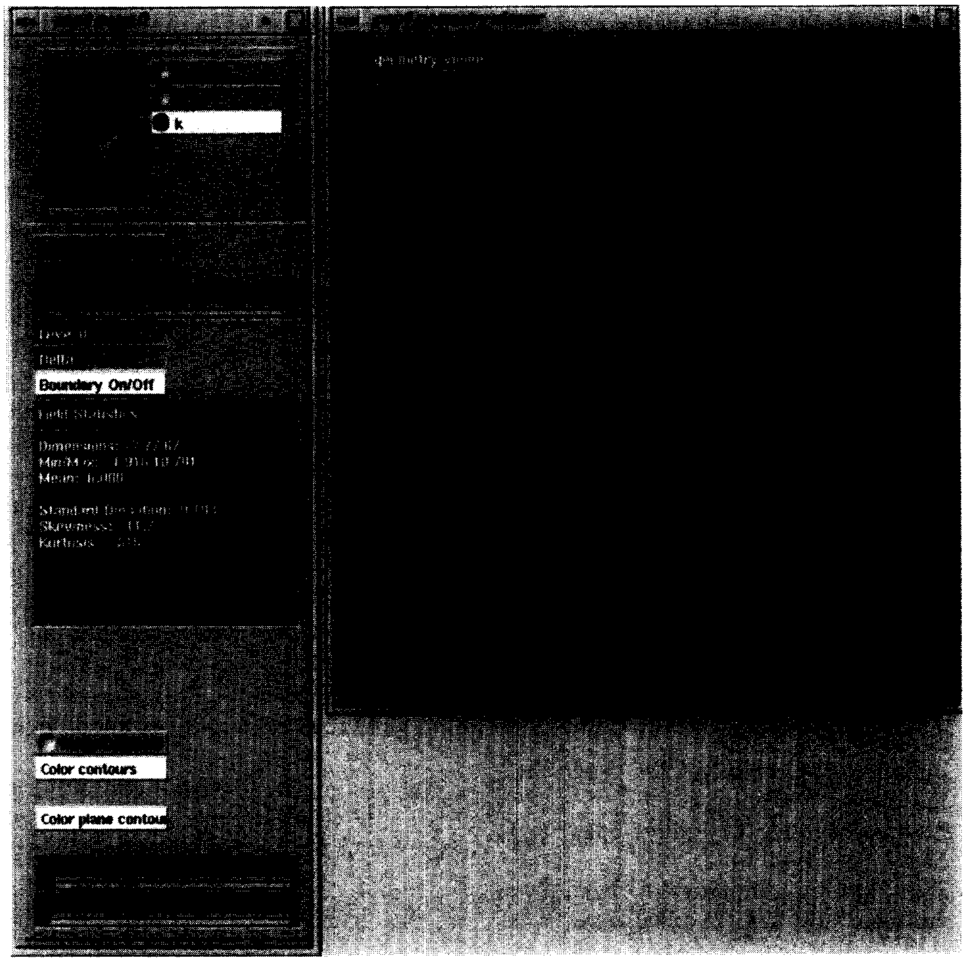


Figure 9. Part of the user interface of the network shown in Figure 5, with output displayed in the AVS Geometry Viewer window.

arbitrary contoured section of an electron density or Patterson map (*contour_3D_map, interpolating_slicer, new_arbitrary_slicer*). An example of a network to calculate and display sections of a difference Patterson map, which uses these modules, is given in Figure 8. The section number and contour levels may be varied interactively, by the use of the interface shown in Figure 9, which also shows the output displayed in the AVS Geometry Viewer window. Hard copy PostScript output may also be obtained.

CONCLUSIONS AND FURTHER WORK

It has proved relatively straightforward to integrate programs from the CCP4 crystallographic package into the AVS environment, with only minor changes to the original source code. The data flow environment provides a natural framework for constructing CCP4 program networks. The provided interface builder tools facilitate the construction of graphical user interfaces for individual programs or networks of related programs, and prepackaged graphical tools facilitate the visualization of the results of calculations. Most of the coding efforts goes into programming the dynamic behavior of the user interface and into the extraction of significant numbers from the program output. We expect the majority of our module code and networks to be portable to the new AVS/Express and AVS version 6 environments, by virtue of the AVS 5 compatibility kit provided with these products, with the exception of the user interfaces, which will need to be modified to use the Motif widget set rather than the "LUI" widgets supported by AVS 5. Although the use of the AVS environment has been described in this article, other data flow environments could be used, such as the freely available Khoros 2.0 environment. The method of using named pipes to communicate between the GUI and Fortran program could also be used to integrate other crystallographic programs that expect card image input data and produce line printer output (such as SHELX,[23] TNT,[24] MERLOT,[25] and GLRF[26]) into the same environment.

Details of how to obtain modules and networks are available from D.L.W. (wild@sbl.salk.edu).

ACKNOWLEDGMENTS

The authors thank M. Harrington for help with the illustrations, and the following for helpful comments and suggestions: A. Bilwes, P. Kanellopoulos, D. Love, J. Noel, J. Postma, and L. Ten Eyck. D.L.W. acknowledges the support of EMBL during part of this work.

REFERENCES

1 Hall, S.S. Protein images update natural history. *Science* 1995, **267,** 620–624

2 Frenz, B. Improved productivity through crystallographic packages. In: *Crystallographic Computing 5* (Moras, D., Podjarny, A.D., and Thierry, J.C., eds.). O.U.P., Oxford, 1991, pp. 126–135

3 McRee, D.E. A visual protein crystallographic software system for X11/XView. *J. Mol. Graphics* 1992, **10,** 44–46

4 Bourne, P.E., Marquess, P.M., and Hendrickson, W.A. The crystallographic workbench. In: *Collected Abstracts, 15th Congress of the IUCr,* 1990, MS-02.01.06

5 Wild, D.L. An X-windows based user interface for protein crystallographic software (abstract). In: *Crystallographic Computing 5* (Moras, D., Podjarny, A.D., and Thierry, J.C., eds.). O.U.P., Oxford, 1991, p. 471

6 Collaborative Computational Project, Number 4. The CCP4 suite: Programs for protein crystallography. *Acta Crystallogr.* 1994, **D50,** 760–763

7 Blundell, T.L. and Johnson, L.N. *Protein Crystallography.* Academic Press, London, 1976

8 Drenth, J. *Principles of Protein X-Ray Crystallography.* Springer-Verlag, New York, 1994

9 Rhodes, G. *Crystallography Made Crystal Clear.* Academic Press, San Diego, 1993

10 McRee, D. *Practical Protein Crystallography.* Academic Press, San Diego, 1993

11 Evans, P.R. The CCP4 package program. In: *Crystallographic Computing 5* (Moras, D., Podjarny, A.D., and Thierry, J.C., eds.). O.U.P, Oxford, 1991, pp. 136–144

12 AVS, Inc., Waltham, Massachusetts

13 IBM, Yorktown Heights, New York.

14 NAG, Ltd., Oxford, U.K.

15 Khoral Research, Inc., Albuquerque, New Mexico

16 Upson, C., Faulhaber, T. Jr., Karnins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., and vanDam, A. The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graphics Appl.* 1989, **9**(4), 30–42

17 Konstantinides, K. and Rasure, J. The Khoros software development environment for image and signal processing. *IEEE Trans. Image Processing* 1994, **3,** 243–252

18 Duncan, B.S., Pique, M., and Olson, A.J. AVS for Molecular Modelling. Proceedings of 2nd International AVS User Group Conference, 1993

19 Casher, O., Green, S.M., and Rzepa, H.S. EyeChem 1.0: A modular chemistry toolkit for collaborative molecular visualization. *J. Mol. Graphics* 1994, **12**(3), 226–234

20 Walton, J. Visualization of sphere packs using a dataflow toolkit. *J. Mol. Graphics* 1994, **12**(4), 275–281

21 Albury, R. Dusting off applications. *Unix Rev.* 1991, **9**(1), 40–45

22 Navaza, J. AMoRe: An automated package for molecular replacement. *Acta Crystallogr.* 1994, **A50,** 157–163

23 Sheldrick, G.M. Phase annealing in SHELX-90: Direct methods for larger structures. *Acta Crystallogr.* 1990, **A46,** 467–473

24 Tronrud, D.E., Ten Eyck, L.F., and Matthews, B.W. An efficient general-purpose least-squares refinement program for macromolecular structures. *Acta Crystallogr.* 1987, **A43,** 489–501

25 Fitzgerald, P.M.D. MERLOT, an integrated package of computer programs for the determination of crystal structures by molecular replacement. *J. Appl. Crystallogr.* 1988, **21,** 273–281

26 Tong, L. and Rossmann, M.G. The locked rotation function. *Acta Crystallogr.* 1990, **A46,** 783–792