Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

ASpect: A Multi-Wavelength Spectrum Analysis Package for IRAF

S. J. Hulbert, J. D. Eisenhamer, Z. G. Levay, and R. A. Shaw

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21218

Abstract. ASpect is spectrum analysis package being developed at ST ScI. We discuss the implementation of a GUI using the IRAF Object Manager, the ASpect task design, and the mechanics of performing a fit to a spectral feature with ASpect.

1. Introduction

ASpect is a spectrum and line analysis package being developed at ST ScI. ASpect is designed as an add-on package for IRAF and incorporates a variety of analysis techniques for astronomical spectra. ASpect operates on spectra from a wide variety of ground-based and space-based instruments, allowing simultaneous handling of spectra from different wavelength regimes. It accommodates non-linear dispersion relations. ASpect provides a variety of functions, individually or in combination, with which to fit spectral and continuum features. It allows for the masking of known bad data. Most importantly, this tool provides a powerful, intuitive graphical user interface (GUI) implemented using the IRAF Object Manager and customized to handle the burden of data input/output (I/O), on-line help, selection of relevant features for analysis, plotting and graphical interaction, and database management. ASpect is scheduled for release in late 1995.

2. ASpect GUI

The functional need for a full set of graphical I/O capabilities originally led us to implement the ASpect GUI with the public-domain software, Tcl/Tk. While Tcl/Tk provided the GUI functionality required for the task, this was accomplished at the expense of working outside of the IRAF environment. With the availability of the IRAF Object Manager, however, we now have access to the necessary GUI elements within IRAF. As originally envisioned, ASpect is an IRAF application–it is written is SPP and uses the IRAF Object Manager to provide the GUI. In this scenario the ASpect task is the client; the Object Manager is the server. At startup, the client application downloads a text file that defines the GUI. The Object Manager interprets this GUI file using the Tcl interpreter and then creates the GUI objects (each widget is an object). For the duration of the task execution, the Object Manager manages the messages between the graphical objects and the client application.

3. ASpect Task Design

The ASpect task has been organized as a group of objects: each object is implemented by a separate code library. Figure 1 shows a graphical representation of the relationships between the ASpect objects and code libraries. In this figure, the connections show which objects are available for "use" by any given object, i.e., the information "flows" towards the small circles at the ends of the connections. For example, the fit library "knows" about the component and spectrum objects but a given spectrum or component object knows nothing of fitting. This approach enforces modularity and by requiring a set of well-defined interfaces makes reuse of code more practical.

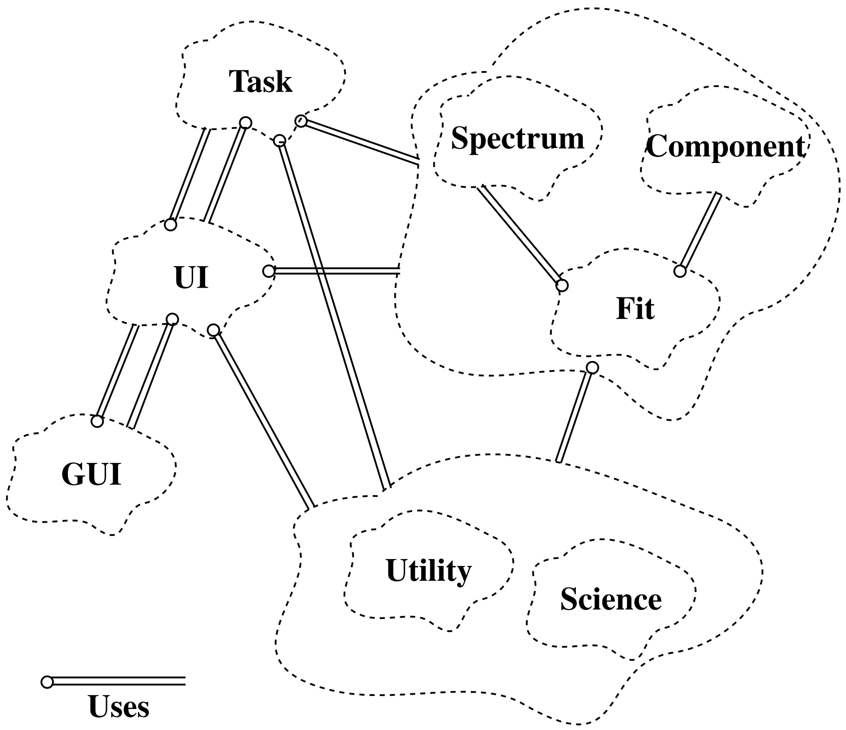

Figure 1. Relationships between ASpect Objects and Code Libraries.

The highest level of objects are designed to bridge the gap between the user and the actual task. The GUI object, which is created by the IRAF Object Manager, manages all information passed between the UI object and the GUI. The UI object consists primarily of the "event loop", which handles events from the GUI. The UI object is solely responsible for setting UI parameters and all GIO calls are isolated here.

The basic task functionality is implemented in the task object, which is in turn dependent on a host of other objects and libraries. The task object is the main procedure and all global variables are maintained here. The task object manages a series of spectrum and component objects in the course of

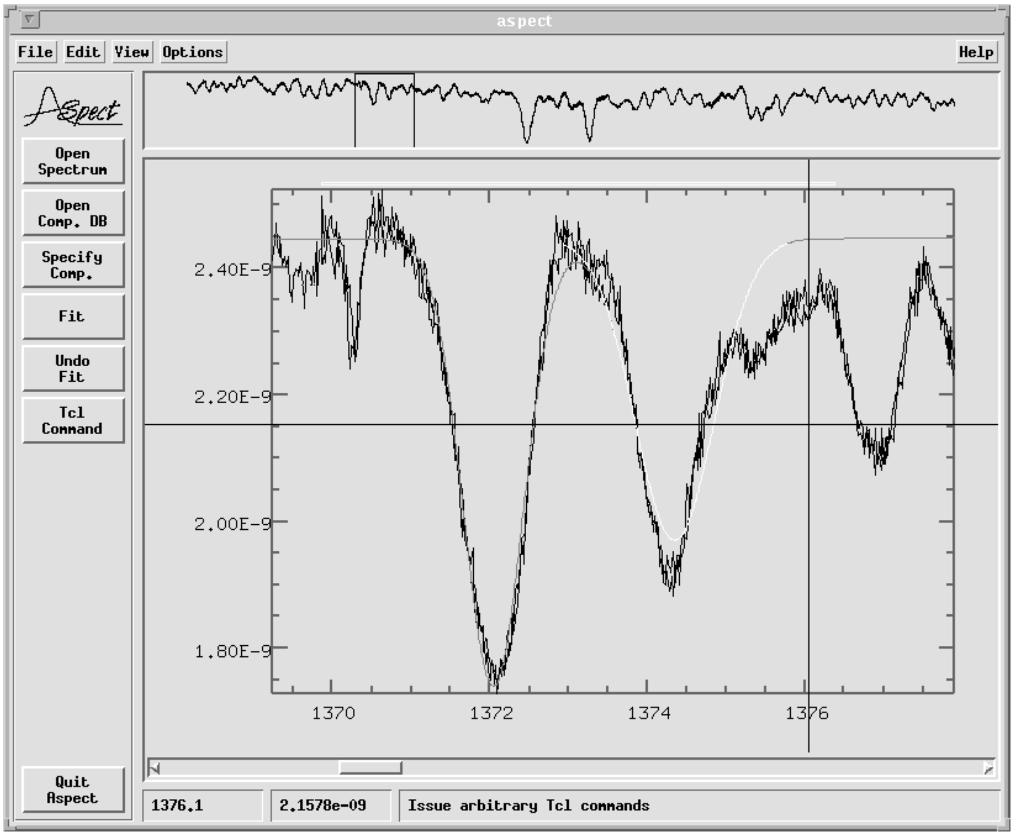

Figure 2. The ASpect workspace consists of a main window showing a zoomed portion of the spectrum bordered by a menu bar and spectrum overview window across the top, a button bar to the left, and a scroll bar, cursor readback display, and context-sensitive help box along the bottom.

the analysis. The spectrum object merely defines a spectrum: a flux array, a wavelength array, an error array, and a data mask array. A component object is a collection of models for various features in a spectrum which are to be fit. The actual fitting process is controlled by the data contained in the fit object.

The lowest level of objects provide support for all other parts of the ASpect task. The science libraries consist of a set of routines that implement the basic scientific functionality. In the case of fitting these consist of the actual fitting functions, e.g., Gaussian absorption line or black-body continuum. The utility library is a set of generic routines used by more than one part of ASpect.

4. Fitting Procedure

The set of actions that results in a fit to features in a spectrum consists of: (1) selecting the region of the spectrum to be analyzed, (2) selecting the type of component, (3) marking the region of the spectrum where the component is

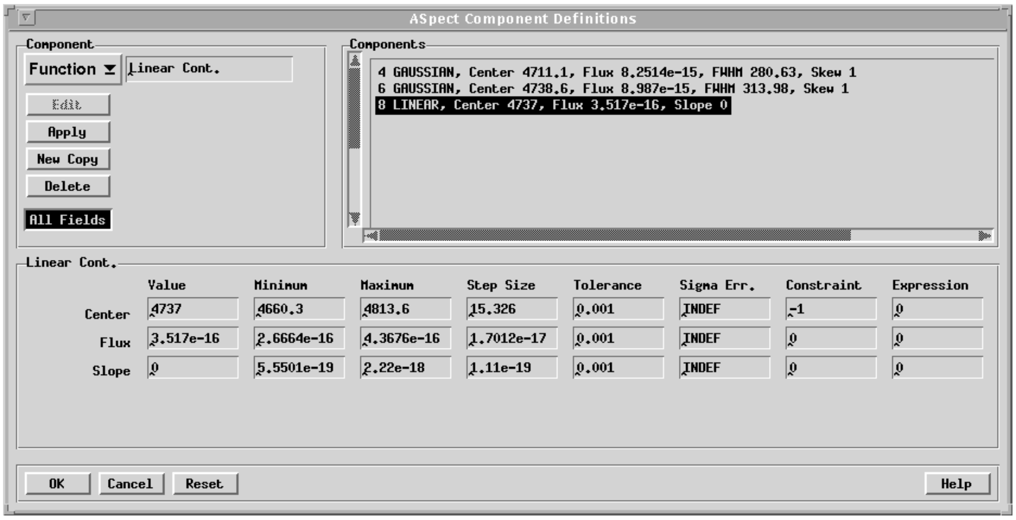

Figure 3. The component dialog box is used to display and edit components and the parameters that define them.

relevant, (4) marking the region of the spectrum over which to do the fit, and (5) initiating the fit. The initial spectral region selection is accomplished with the pan and zoom capabilities of ASpect. The marker in the overview window (see Figure 2) can be moved or resized. Alternately, a marker can be drawn in the main window to zoom in on a particular part of the spectrum. The type of component is selected in the component dialog box which is shown in Figure 3. A pull down menu labeled “Function” is used to reveal a set of line and continuum functional models. This function is mapped to a particular feature of the spectrum by drawing out a box around the feature. The size and placement of this marker is used to generate an initial guess for the parameter values of the component. Once defined, the component is added to the component list (seen in the top right-hand corner of the component dialog box). Additionally, the guess is displayed graphically in the main window as a fit (of sorts). At this point the user may edit the parameter values of an individual component as well as certain fit-related constraints and limits. Any number of components can be defined in this manner. When all components have been defined, one or more fit regions are marked. These fit regions are the parts of the spectrum that will be used in evaluating the fit. Finally, the fit is initiated using the fit dialog box. Generic fit parameters, such as number of iterations and type of fit algorithm, are also controlled from the fit dialog box. Once the fit is complete, the fit is overplotted in the main window. As the fitting process is iterative, the user may decide to repeat the fit after modifying any of the components or fit regions.

Acknowledgments. This ASpect project is funded under contract with the NASA Astrophysical Data Program.