APPENDIX C
GATORPLOT DOCUMENTATION

C.1 Overview

GatorPlot is a software package designed in the IDL programming language to interactively plot, fit, and manipulate data from one or more FITS or ASCII files or functions. I developed GatorPlot while working on my dissertation, adding features as I needed them, because the current available software was very inadequate. Beginning with chapter 4, everything for my dissertation was done using GatorPlot. Documentation and downloads for GatorPlot are available on the web at http://www.astro.ufl.edu/∼warner/GatorPlot/. To install GatorPlot, simply download the files gatorplot.pro and gatorplot_config.pro and place them anywhere in your IDL_PATH. The files are then executed by simply typing "gatorplot" or "gatorplot_config" at any IDL command prompt. GatorPlot can be used via an interactive graphical user interface (GUI) or a scripting language, GPScript. A large portion of the GUI is the graphing window and to the left of that window and below it are buttons and text areas that control all of the plot options (see Figure C-1). GPScript allows scripts to be written to do anything automatically instead of interactively.

The program gatorplot_config allows you to specify a default directory to select files from, the location to save the temporary file gatorplotvars.dat in, the location of the gatorplot.help file, and a command to start your web browser of choice. The default values are your current directory, $HOME, $IDL_DIR/contrib/astroLib/text/gatorplot.help, and none, respectively.

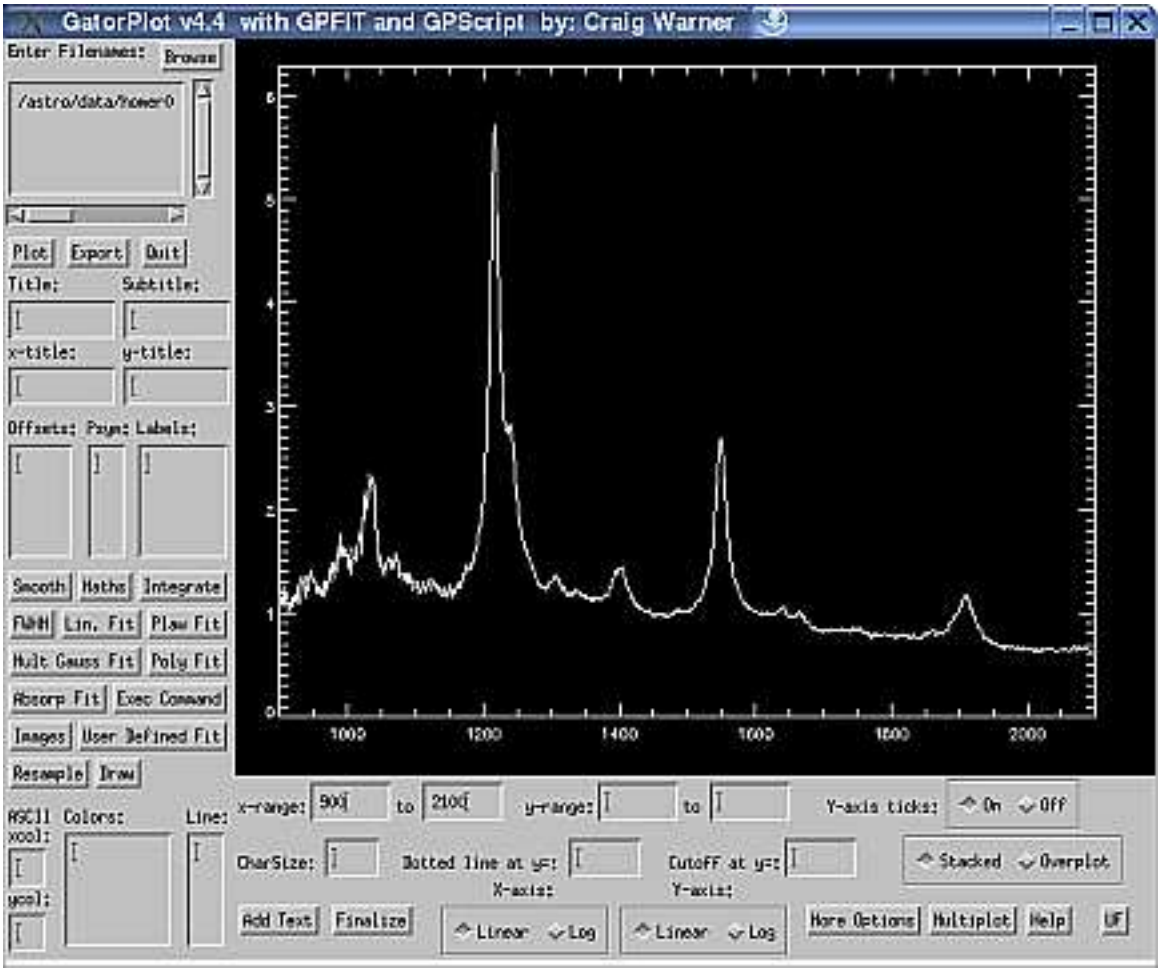

Figure C–1 GatorPlot v4.4 plotting the rest-frame UV region of a composite spectrum.

This appendix includes documentation of every major feature of GatorPlot. When a version identifier appears next to a feature (i.e., v2.0+), it marks the first version of GatorPlot in which the feature was implemented.

C.2 Plotting Files

At the top left corner of the GUI are the "Enter Filenames" text area and the "Browse" button. Enter as many filenames as desired by either typing them directly into the text area or by clicking the "Browse" button to bring up a file selector that allows multiple selections. These selections are appended to anything already in the text area, allowing files from multiple directories to be chosen by clicking browse multiple times.

The types of files accepted by GatorPlot are .fits, ASCII files, and .list files. If a filename ends in .fits, it is assumed to be a FITS file. Note that the read_fits, write_fits, and sxpar procedures need to be in your current directory or IDL_PATH. If a filename ends in .list, it is assumed to be an ASCII file containing a list of FITS and/or ASCII filenames, one per line. If neither .fits or .list, the file is assumed to be an ASCII file. When plotting an ASCII file, a window will pop up asking for the X-axis and Y-axis columns in the file. Multiple columns can be added or subtracted, e.g., column 1 is your x-axis scale, and you want to plot column 2 + column 3 as the y-axis. Enter 1 for x-axis column and 2+3 for y-axis column. Similarly you could enter 4-2 or 3+5-4+7. If no value is entered for the X-axis column, element # in the y-axis array is used as the x-axis. You can also optionally enter a number of lines to skip at the start of a file. Click "OK" to proceed or "OK for all" if you want the same columns plotted for every ASCII file entered so you don't have to re-enter the column values.

Beginning with version 2.0, functions can be entered into gatorplot in the form f(x)=sin(x) or f(x)=x^ 2+3*x-7 in the file list text area. When you enter a function, you will be prompted for Xmin, Xmax, and the # of points you wish to

use in calculating the function. All of these values are optional and if left blank will default to the min, max, and # of points of the first file in the list if there is another file.

The "Plot", "Export", and "Quit" buttons are located below the "Enter Filenames" box. Click the "Plot" button to generate the plot on the screen. Every time you click "Plot", it will redraw the plot, and lose all text added after plotting. Clicking anywhere in the plotting window returns the X and Y coordinates of the mouse pointer to your terminal window.

Click the "Export" button to export the plot. You will be prompted for a filename and given a choice of format (Figure C-2). You can also (in v3.0 and above) choose to export as greyscale or color. Note that IDL does not support true color plots in the postscript device so if your display is set to true color, you must use Screenshot .PS to get a color .PS file (the same goes for printing and .PDF files). The format choices are as follows.

- Postscript: After selecting Postscript, you are given the option of specifying its Xsize and Ysize in inches and whether you want .ps, or .eps (Encapsulated Postscript). Then you can click Finalize or Don't Finalize. If you click Finalize, the plot is sent to the specified file and is ready to be printed or opened in a PS viewer. If you click Don't Finalize, the plot is sent, but the file remains open so that you can add text to it using the "Add Text" button at the bottom. You must then click the "Finalize" button, also at the bottom, to close the file
- JPEG: Everything currently on the screen (including text added to the plot) is sent to a JPEG file when you click Export.
- FITS: The data currently being viewed is exported to a .FITS file. Note that only the data from the first file in the file list is exported to the .FITS file,

Figure C–2 GatorPlot's export window listing your choices of file formats.

and that only the portion of the data currently being viewed is sent to the .FITS file. So you can zoom in on a certain X-range and save only that part.

- Printer (works in Unix/Linux only): Sends the plot to your default printer. As with Postscript, you can specify the dimensions and click Finalize to print the plot, or click Don't Finalize and add text. Then when you click Finalize at the bottom, the plot with the text will be sent to the printer.
- PDF (works in Unix/Linux only): Same as Postscript except its sent to a .pdf file, viewable in Adobe Acrobat and Netscape.
- Screenshot .PS (v3.0): Screenshot .PS generates a Postscript file from the data currently in the plot window. Everything in the plot, including text that has been added is sent to the .PS file. To make it encapsulated, simply enter a .eps instead of .ps filename. The .PS file is automatically finalized so you can't add anything after exporting it.

- Print Screenshot (v3.0, Unix/Linux only): Same as Screenshot .PS except it sends the plot to your default printer.
- Screenshot PDF (v3.0, Unix/Linux only): Same as Screenshot .PS except its sent to a .pdf file.
- ASCII (v4.0): The x-axis and y-axis data are exported to a two-column ASCII file.

Clicking the "Quit" button quits GatorPlot.

C.3 Titles, Labels, Symbols, and Offsets

Under the "Plot", "Export", and "Quit" buttons are text fields labeled Title, Subtitle, x-title, y-title. Text entered in these fields will be displayed as the main title, subtitle, and corresponding axes titles on the plot.

If you are creating a stacked plot, use the Offsets text area to override the default offsets between the different plots. For n files, enter up to n-1 offsets. The first offset sets the y-value to be used as the "zero" value for the second file (overriding the default). The second line in offsets sets the y-value to be used for the third file, and so on.

In the Psym text area, you can enter a plotting symbol for each file. Leave blank or enter 0 for a line connecting the points. Enter 1-7 for different symbols (plus sign, asterisk, dot, diamond, triangle, square, and $\times$, respectively) or -1 to -7 for those symbols to be used and connected by a line. Enter 10 for histograms.

In the Labels text area, you can enter a short label for each file that will appear on the right side of the plot beyond the right y-axis. If creating a stacked plot, each label will appear next to the corresponding "zero" level. If overplotting data, the labels will appear from top (first file) to bottom (last file) of the right side.

C.4 Buttons

Click the "Smooth" button (v2.0+) to perform a smoothing function to your data. In the window that pops up, select Boxcar or Binomial smoothing, the number of points to bin by, and the number of iterations. For example, binning 3 points with boxcar smoothing would mean $x_2 = x_1/3 + x_2/3 + x_3/3$ while binning 3 points with binomial smoothing would result in $x_2 = 0.25x_1 + 0.5x_2 + 0.25x_3$.

The "Maths" button (v2.0+) can be used to perform arithmatic operations. A new window opens that has two text areas listing all the files, and next to each, text fields for scaling and x-offsets. You can select multiple files (or functions) in each text field and enter a scaling factor and an x-offset for each one. A constant can also be added. Then select the operation to perform between the two lists and click either plot or save (the result can be saved to a .fits or ASCII file). For example, if you wanted to plot File1 minus the constant 2, you could select File1 from the top text area and enter -2 in the +Constant box. If you wanted to plot (File1 - Function1) / (0.5 * File2), you would first select File1 and Function1 in the top text area and enter 1 and -1 in the first two rows of the Scale By box. Select "/" as your operation, select File2 in the lower text area, and enter 0.5 in the lower Scale By text field (Figure C-3). Finally, click either Plot or Save. X-offset is used if two files have different starting values so you can shift one relative to the other. Any arithmatic operation, from trivial to very complicated ones, can be performed on any set of files and functions with just a few clicks.

The "Integrate" button (v2.0+) performs an integration over a user-defined baseline. It can be used to integrate the flux of emission or absorption lines, to integrate functions numerically, or anything else. Clicking Integrate opens a new window that allows you to draw a baseline to integrate over. You can either enter a pair of (X,Y) coordinates for the start and finish of the line or left click on the graph window to obtain (X1,Y1) and right click to obtain (X2,Y2) coordinates by

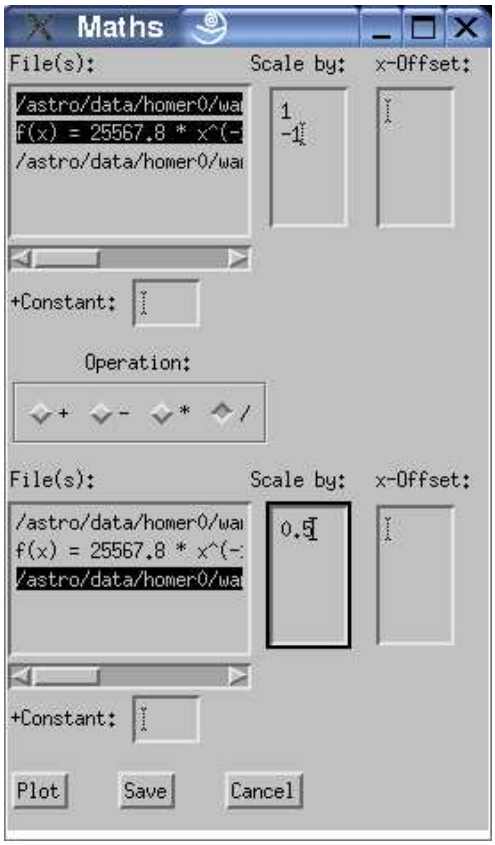

Figure C–3 GatorPlot's maths window demonstrating one of the above examples.

mouse. Leaving the Y values blank results in them defaulting to 0. If you have more than one file listed in gatorplot, in either stacked or overplot mode, the integration will automatically be done for all files listed. Since there are a finite number of points in these functions and files, the integration is simply a sum that approximates the integral. This can be very useful in estimating fluxes of emission lines. Clicking Integrate returns 5 lines of information to your terminal window: filename (or function), Integration (flux), center x-axis value, Average f (continuum or baseline), and Equivalent Width. The baseline is overplotted as a dotted line.

The "FWHM" button (v2.0+) estimates the Full Width at Half Maximum of an emission or absorption line. Clicking FWHM opens a window that asks for the X and Y coordinates of two continuum points, one on either side of the line to be measured. It also asks for the X and Y coordinates of the peak of the line. This is optional and will be computed automatically if you leave these blank. The continuum points can be set by left clicking in the plot window at point1 and holding the mouse button down while you drag the pointer to point2. Then release the mouse. They can also be entered manually into the text fields. Right clicking enters the coordinates into X and Y peak. When you click FWHM, notice that a smoothed spectrum appears. The spectra are automatically smoothed by 5-point boxcar smoothing. This reduces the effect of noise on the measurement. The continuum is obtained by averaging 5 (already smoothed) points at each of the specified locations. If you do not specify the peak location, that is determined to be the maximum (smoothed) value in between the two continuum points. The program then determines the width of the line at half maximum and returns 4 lines to your terminal window: filename (or function), FWHM (in x-axis units), FWHM in km/s, and the center in x-axis units. This is very useful in measuring the FWHMs of strong lines, but may fail if lines become very blended. In that case,

users must either guess at a local continuum or just click on either side of the line to obtain the X and Y coordinates and estimate the FWHM manually.

C.5 GPFIT

GPFIT was included with GatorPlot beginning with version 3.0. GPFIT refers collectively to all of GatorPlot's fitting routines. The "Lin. Fit" button (v3.0+) applies a linear fit of the form $Y = a + bX$ to your data using a chi-square minimization routine. You can select the ranges of data to be used in the fit by either entering them manually in the range box in the format low_x_value high_x_value. For instance, the example shown in Figure C-4 would use only data with x-axis values between 500 and 650 and between 712.5 and 842. Alternatively, you can just click on the screen to set the range. Click the left mouse button on the low x value, drag the mouse pointer to the high x value, and release the button. The range you selected will be appended to the range box. If the range box is left blank, it will use all the data for the fit. You can select no errors, poisson errors, or specify errors for each point from an input file. Note that the column of errors in the file must correspond point for point with Y_i, the individual y-axis values. Click Fit to do the fit. The results will be output to the screen and appended to a file linfit.gp in your current directory. A dotted line showing the fit will also be plotted on top of your data. A fit will be done for each file you have in the "Enter Filenames" box. Thus, you can automatically fit multiple files at once as long as you want to use the same range to fit each file.

The "Plaw Fit" button (v3.0+) fits a powerlaw of the form $Y = bX^a$ to your data using a chi-square minimization routine. You can select the ranges of data to be used in the fit as with the Linear Fit (see above). Choose no weighting (uniform) or Poisson weighting ($1.0/Y_i$). You may optionally enter your initial guesses for b and a. If either one or both are left blank, GatorPlot will generate initial guesses for you. Click Fit to perform the fit. The results will be output to

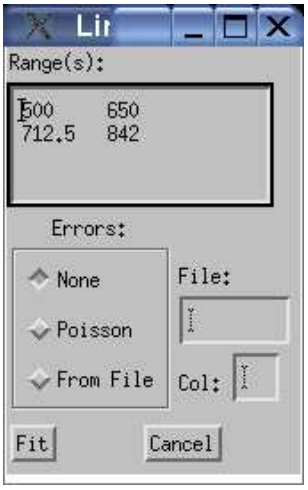

Figure C–4 An example of a linear fit to the range 500–650 and 712.5–842 Å.

the screen and appended to a file plawfit.gp in your current directory. A dotted line showing the fit will also be plotted on top of your data. Note that you can just copy the equation that is output to the screen right back into the "Enter Filenames box", substituting f(x) for Y and making sure to put parenthesis around a negative exponent, and plot the function yourself or enter Maths and divide the spectrum by the fit. The fit will be done for each file you have in the "Enter Filenames" box, so you can automatically fit multiple files at once as long as you want to use the same range to fit each file.

The "Mult Gauss Fit" button enables you to fit your data with one or more Gaussian profiles using a chi-square minimization routine. You can select the ranges of data to be used in the fit as with the Linear Fit (see above). Choose no weighting (uniform) or Poisson weighting ($1.0/Y_i$). If there is a constant offset from 0, you can enter that constant in the Constant box. For instance if you are fitting data that has been normalized to a continuum of 1, you would enter 1 in the Constant box. Initial guesses are mandatory, and there are two methods of inputting them. You can either give it a .gp filename that contains your initial guesses or enter the number of Gaussians you wish to fit in the #Gaussians box

and enter the initial guesses of the parameters of each Gaussian in the Enter Fit Parameters box. The .gp file should have the following format.

```
number_of_Gaussians
flux1           factor tie_to   optional comments
wavelength1     factor tie_to   you can write anything
FWHM1           factor tie_to   you want to out here
flux2           factor tie_to
wavelength2     factor tie_to
FWHM2           factor tie_to
...
```

The wavelength is in whatever scale the x-axis is in, and the FWHMs are in km/s. For free parameters, just enter 0 for both factor and tie_to. To fix a parameter at a particular value (i.e., wavelength has to be 1549.0 and cannot float), enter -1 for factor and 0 for tie_to. To tie a parameter to a constant times the value of another parameter (i.e., FWHM3 = FWHM1 or flux2 = 2*flux1), enter the constant for factor (i.e., 1 or 2 in the examples), and enter the parameter to tie it to for tie_to. This is **not** the Gaussian number, but the parameter number, which starts at 0 for flux1, 1 for wavelength1, 2 for FWHM1, 3 for flux2, 4 for wavelength2, and so on. You may find it helpful to keep track of the parameter numbers in your optional comments space in the file. So for the example of flux2 = 2*flux1, I would enter 2 for factor and 0 for tie_to. Below is an example file.

```
2
15.0            0       0       param0
1549.0         -1       0       param1 fixed at 1549
3000            0       0       param2
30.0            2.0     0       param3 tied to 2*flux1
1549.0          0       0       param4
```

```
3000            1       2       param5 tied to 1*FWHM1
```

Finally, you can tie a parameter to a variable ratio. This is useful if you want to use the profile of one emission line to constrain the fit to another emission line. For example, you want to use the profile of C IV to fit O III], and use 2 Gaussians to fit each line: 1 and 2 to fit C IV and 3 and 4 to fit O III]. In this case, you would want flux4/flux3 = flux2/flux1 in order to maintain the C IV profile in the fit to O III]. You would then let flux1, flux2, and flux3 float and tie flux4 to (flux2/flux1)*flux3. This is done by entering -3.00 for factor and 6 for tie_to. The wavelength4 could similar be tied to (wavelength2/wavelength1)*wavelength 3 by entering -4.01 for factor and 7 for tie_to. If factor is negative and not an integer, GPFIT takes the integer part (of the absolute value) as the parameter number for the numerator and 100 times the fractional part (of the absolute value) as the parameter number for the denominator. It then multiplies this ratio by the value in the parameter given in tie_to just as it would do with a constant factor. If you want to enter the parameters into the Enter Fit Parameters box instead of providing an input file, they are entered in the exact same way except that number of Gaussians is left off since it has its own box. So the first line in the Enter Fit Parameters box would be the second line of the input file (i.e., parameter number 0, which is the flux for the first Gaussian. Figure C-5 shows an example in which an input file is used to fit multiple Gaussians to the flux in the wavelength interval 1450–1720 Å of a spectrum with the continuum normalized to unity.

Click Fit to perform the fit and the final fit parameters will be output to the screen. They will also be appended to a .gpfit file. You can optionally specify this filename by entering it in the Output (.gpfit) File text field. If no filename is entered, it defaults to multgaussfit.gpfit in your current directory. A .dat file is also created with the plot data. It is an ASCII file with the format

```
Xi  Total_Fit  Const  Gauss1  Gauss2  ...  GaussN
```

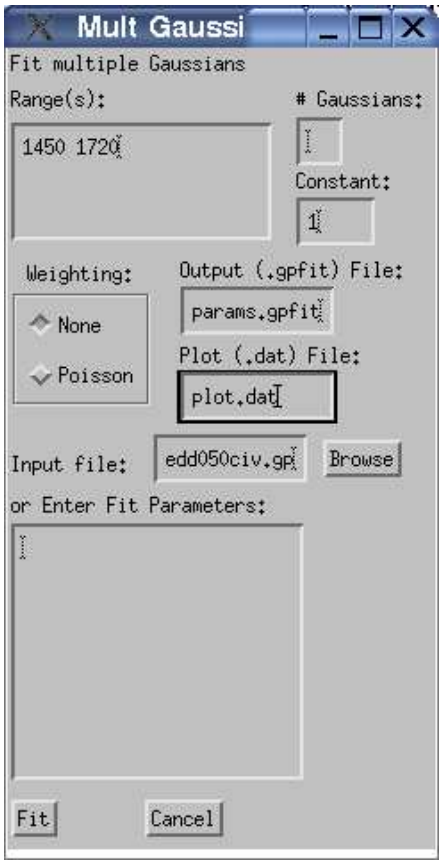

Figure C–5 An example of a multiple Gaussian fit to the wavelength range 1450–1720 Å. The continuum in the spectrum being fit has been normalized to unity, so 1 is entered as the constant. An input file contains the initial parameters.

and thus contains N+3 columns for a fit of N Gaussians. This filename can be specified in the Plot (.dat) File text field. If no filename is entered, it defaults to gpmultgaussfit.dat in your current directory. Your fit is then plotted over your data. Each individual Gaussian is plotted in one of five colors, with a dahsed line. The overall fit is then plotted in a solid blue-green line over top of your data. The fit will be done for each file you have in the Enter Filenames window, so you can automatically fit multiple files at once as long as you want to use the same range to fit each file and as long as the initial guesses are valid for each file. If the fit is unsatisfactory or the chi-square fails to converge, you should modify your initial guesses and try again. Once you have a satisfactory fit, you can export it to a postscript file using the Screenshot .PS option in the Export menu. Figure C-6 shows the result of a fit to the data shown in Figure C-5.

Click the "Poly Fit" button (v3.0+) to fit a polynomial of the form $Y = a_0 + a_1X + a_2X^2 + ... + a_nX^n$ to your data using a chi-square minimization routine. You can select the ranges of data to be used in the fit as with the Linear Fit (see above). Choose no weighting (uniform) or Poisson weighting ($1.0/Y_i$). Note that Poisson weighting can not be used if one of your data values is 0 since 1.0/0 is infinite. Enter the order of the polynomial you want to fit: 0 is a constant, 1 a linear fit, 2 quadratic, 3 cubic, and so on. Then you can optionally enter your initial guesses to the polynomial coefficients. Enter them one per line in the Init Coeff box, in the order a_0 (constant term), a_1 (linear term), a_2, and so on. Then click Fit to perform the fit. The results will be output to the screen and appended to a file polyfit.gp in your current directory. A green dashed line showing the fit will also be plotted on top of your data. The fit will be done for each file you have in the Enter Filenames window, so you can automatically fit multiple files at once as long as you want to use the same range to fit each file.

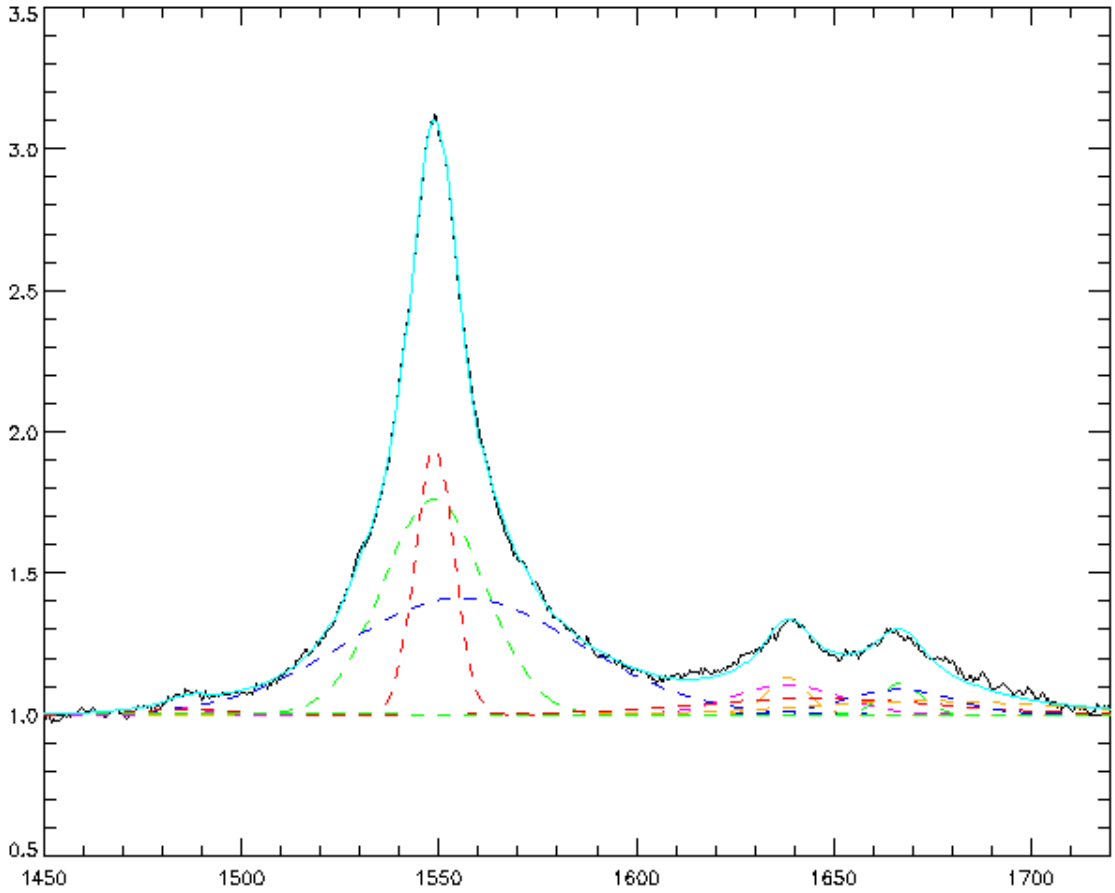

Figure C–6 An example of a multiple Gaussian fit to the wavelength range 1450–1720 Å of a normalized composite spectrum. C IV is fit with three Gaussian profiles, and N IV], He II, and O III] are all fit with C IV profiles. The plot has been exported using the Screenshot .PS option in the Export menu.

You can fit absorption lines using the "Absorp Fit" button (v4.0+). This is similar to the multiple Gaussian fit discussed above. You can fit your data with zero or more Gaussian profiles plus zero or more absorption line profiles of the form $I_\lambda = C_f I_0 e^{-\tau_\lambda} + I_0(1 - C_f)$, where $\tau_\lambda = \tau_0 e^{(-\Delta\lambda^2/\Delta\lambda_D^2)}$. Once again, a chi-square minimization routine is used to perform the fit. You can select the ranges of data to be used in the fit as with the Linear Fit (see above). Choose no weighting (uniform) or Poisson weighting ($1.0/Y_i$). If there is a constant offset from 0, you can enter that constant in the Constant box. For instance if you are fitting data that has been normalized to a continuum of 1, you would enter 1 in the Constant box. Initial guesses are mandatory, and there are two methods of inputting them. You can either give it a .gp filename that contains your initial guesses or enter the number of Gaussians and number of absorption lines you wish to fit in the #Gauss box and the #lines box, respectively, and enter the initial guesses of the parameters of each Gaussian and absorption line in the Enter Fit Parameters box. The .gp file should have the following format:

```
number_of_Gaussians
number_of_absorption_lines
flux1           factor tie_to   optional comments
wavelength1     factor tie_to   you can write anything
FWHM1           factor tie_to   you want to out here
flux2           factor tie_to   Note that there are 3 lines per
wavelength2     factor tie_to   Gaussian following the
FWHM2           factor tie_to   number_of_absorption_lines,
...
Tau_0 1         factor tie_to   then there are 5 lines
wavelength1     factor tie_to   per absorption line.
lambda_D 1      factor tie_to
```

```
I_0 1           factor tie_to
C_f 1           factor tie_to
Tau_0 2         factor tie_to
wavelength2     factor tie_to
lambda_D 2      factor tie_to
I_0 2           factor tie_to
C_f 2           factor tie_to
...
```

The wavelength is in whatever scale the x-axis is in, and the FWHMs are in km/s. τ_0 is an optical depth, λ_D is the Doppler broadening, I_0 is the continuum intensity, and C_f is the coverage factor (which ranges from 0 to 1). **It is very important that you have reasonable initial guesses for the fitting routine to properly work**. For free parameters, just enter 0 for both factor and tie_to. To fix a parameter at a particular value (i.e., wavelength has to be 1549.0 and cannot float), enter -1 for factor and 0 for tie_to. To tie a parameter to a constant times the value of another parameter (i.e., FWHM3 = FWHM1 or flux2 = 2*flux1), enter the constant for factor (i.e., 1 or 2 in the examples), and enter the parameter to tie it to for tie_to. This is **not** the Gaussian number, but the parameter number, which starts at 0 for flux1, 1 for wavelength1, 2 for FWHM1, 3 for flux2, 4 for wavelength2, and so on. Remember that the parameter number for τ_0 for the first absorption line will be 3 times the number of Gaussians. τ_0 for each subsequent absorption line will be n_{Gauss} * 3 + n_{lines} * 5, where n_{Gauss} is the number of Gaussians and n_{lines} is the number of previous absorption lines. You may find it helpful to keep track of the parameter number in your optional comments space in the file. Below is a sample input file.

```
2
1
```

```
15.0            0       0       param0
1549.0          -1      0       param1 fixed at 1549
3000            0       0       param2
30.0            2.0     0       param3 tied to 2*flux1
1549.0          0       0       param4
3000            1       2       param5 tied to 1*FWHM1
2               0       0       param6 - Absorption line Tau_0
1528            -1      0       param7 - fixed at 1528
5               0       0       param8
19              0       0       param9
0.5             0       0       param10 - C_f
```

Finally, you can tie a parameter to a variable ratio (see multiple Gaussian fit, above). If you want to enter the parameters into the Enter Fit Parameters box, they are entered in the exact same way except that the number of Gaussians and number of absorption lines are left off since they have their own boxes. So the first line in the Enter Fit Parameters box would be the third line of the input file (i.e., parameter number 0, which could be the flux of the first Gaussian or τ_0 for the first absoprtion line, depending on if you are fitting Gaussians in addition to absorption lines or not).

Click Fit to perform the fit and the final fit parameters will be output to the screen. They will also be appended to a .gpfit file. You can optionally specify this filename by entering it in the Output (.gpfit) File text field. If no filename is entered, it defaults to multgaussfit.gpfit in your current directory. A .dat file is also created with the plot data. It is an ASCII file with the format

```
Xi  Total_Fit  Const  Gauss1  ...  GaussN  Line1 ... LineN
```

and thus contains N+M+3 columns for a fit of N Gaussians and M absorption lines. This filename can be specified in the Plot (.dat) File text field. If no filename

is entered, it defaults to gpmultgaussfit.dat in your current directory. Your fit is then plotted over your data. Each individual Gaussian and absorption line is plotted in one of five colors, with a dahsed line. The overall fit is then plotted in a solid blue-green line over top of your data. The fit will be done for each file you have in the "Enter Filenames" box, so you can automatically fit multiple files at once as long as you want to use the same range to fit each file and as long as the initial guesses are valid for each file.

The "User Defined Fit" button (v4.0+) allows you to fit any user-defined function to your data using a chi-square minimization routine. You can select the ranges of data to be used in the fit as with the Linear Fit (see above). Choose no weighting (uniform) or Poisson weighting ($1.0/Y_i$). You must give the function a one-word name in the Function name box (GPFIT will automatically generate an IDL procedure based on the function you enter and save it in your current directory). Next, enter the function in terms of X and your parameters. Your parameters must be represented by A[0], A[1], A[2], and so on. For example, to fit the function $a\ sin(bX) + c$ to your data (finding values for a, b, and c), you would type: A[0]*sin(A[1]*X)+A[2] where A[0]=a, A[1]=b, and A[2]=c. Next, you must give partial derivatives with respect to each parameter as functions of X so that IDL can use its chi-square minimization routine. For the above example, you would enter sin(A[1]*X), A[0]*X*sin(A[1]*X), and (X>1)/(X>1), respectively. Note that the derivative with respect to A[2] is 1, but IDL needs numeric partial derivatives for the entire array, so the derivative needs an array of the same length as X with each value set to 1. An easy way to do this is of course to set it to X/X which is by definition 1, but you might run into a division by zero error. Thus, (X>1)/(X>1) is a simple way to represent an array of the proper length in which every element is 1 without creating any errors. It is mandatory to enter initial guesses for your parameters, one per line. See Figure C-7 shows the above example.

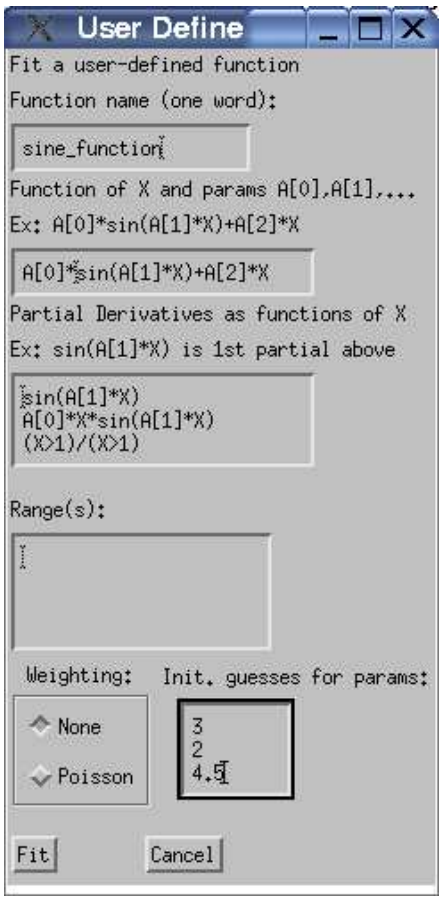

Figure C–7 An example of a user-defined fit of the function $Y = a\ sin(bX) + c$.

Click Fit to perform the fit. The results will be output to the screen and appended to a file udfit.gp in your current directory. A dotted line showing the fit will also be plotted on top of your data. The fit will be done for each file you have in the "Enter Filenames" box, so you can automatically fit multiple files at once as long as you want to use the same function and range to fit each file.

C.6 Commands, Images, and More

The "Exec Command" button (v4.0+) brings up a box that allows you to enter IDL and/or GPScript code to be executed immediately. Click Execute to execute the code or Exit to close the window without executing any code. The code can be as simple as the statement print,"Hello." or as complicated or moreso than the following program.

```
openr,5,'files.txt'
f=''
for j=0,4 do begin
   readf,5,f
   gpsload,f
   a=gpsDataAt(1442) AND gpsDataAt(1712) AND gpsDataAt(1549)
   if a then gpsfwhm,1442,1712,xpeak=1549,out='FWHM.dat'
endfor
close,5
```

Almost anything you can write in a program can be written here (although its doubtful you'll want to use this to write anything much longer than the above example, as it would be better to put anything long in its own file and execute it normally as either an IDL program or with GPScript).

GatorPlot also suports JPEG and FITS images. Click the "Images" button (v4.0+) to bring up the Image Menu which contains seven buttons: Load Image, New Window, Intensity, Export, Erase, Negative, and Cancel. The Image Menu will stay open until you click Cancel, which closes the menu and sends you back to the main GatorPlot. Load Image predictably loads an image. You can choose to load the image into either a new window that IDL will automatically size to fit the entire image, or into an existing window (created by either loading a previous image or with New Window). You can specify an X and Y position in pixels from the lower left hand corner of the window for the lower left hand corner of the image. By using this in an existing window, smaller images can be combined into larger images. In addition to Load and Cancel, there is a Get Info button, which, when clicked, will display on your screen the X and Y dimensions of the image and whether it is greyscale or true color. This can be useful when combining images of unknown sizes into a larger image because it allows you to determine the image size

before entering the position to display the image at in the window. New Window allows you to open a window of a custom X and Y size. For instance, if you wanted to combine four 80x80 images into one 4-panel 160x160 image, you would use New Window to open a 160x160 image, then Load all 4 images into the existing window with the proper X and Y position for each image. Intensity can only be used after an Image has been loaded. Clicking it brings up a window that has boxes labeled X, Y, R/gs, G, and B as well as a Cancel button that will close the window. Mouse over the image and X and Y will display the (X,Y) coordinate the mouse is over. If it is a greyscale image, the intensity (0-255) will be displayed in the R/gs box. If it is a true color image, then the R, G, and B intensities will all be displayed in the corresponding boxes. Erase erases the current open window. Negative takes a negative of the current image. Export allows you to export the image as a JPEG, FITS, Postscript, to the printer, or as a PDF in either greyscale or color. Enter a filename and select the file type, and color or greyscale. If you want to export as an encapsulated postscript, simply select postscript, but make sure your filename ends in .eps. Note that the Printer and PDF options only work on Unix and Linux systems.

The "Resample" button (v4.4) allows you to resample your data in various ways. You can enter a redshift to convert an observed spectrum to its rest-frame wavelengths. For instance, entering a redshift of 2 would cause the x-axis of the spectrum to be divided by 3. You can change the step size between pixels and the start and end wavelengths as well. The modified data can be plotted or saved as an ASCII or FITS file.

You can draw a line anywhere in the plot window using the "Draw" button (v4.4). Enter the X and Y coordinates of the two endpoints of the line either manually or by left-clicking on one point and right-clicking on the other. A color

can optionally be specified as an RGB triple separated by spaces. A linestyle can also be specified (see below).

C.7 Default Columns, Colors, and Linestyles

Beginning in v3.0, you can enter default column numbers for the x and y-axis data to be read from ASCII files. This is the same column you would normally enter in the popup dialogue box prompting you for X-axis column and Y-axis column when you plot an ASCII file. But if entered here, the popup box does not appear and gatorplot reads the column numbers from here instead. This saves time if you are plotting a file several times. GatorPlot will attempt to automatically detect any header the file may have and skip over it to get to the data.

In the Colors text area (v3.0+), you can optionally enter a color for each file to be plotted with. Colors can be entered in one of two ways. If your display is set to true color, you can enter the R, G, and B values (delimited by spaces) of the color you want to be plotted. For instance, 0 255 255 would produce a bright blue-green color or 255 160 0 would produce orange. R, G, and B values range from 0 to 255 each. If you would rather manually enter a color index yourself, you can do that instead. This can be used if you have 8-bit color or a color table already loaded (or if you have true color and are just adventerous). Enter for example 210 (or if you have true color try 3921453). If no color value is entered, it defaults to white on the screen and black in .PS files.

In the Line text area (v3.0+), you can optionally enter a linestyle for each file to be plotted with. Valid linestyles range from 0–5 and correspond to solid, dotted, dashed, dash dot, dash dot dot, and long dashes respectively. If left blank, it will default to solid lines if you are creating a stacked plot or the default linestyle corresponding to the file number for overplots.

C.8 More Options

Under the plot window are various other options that can be specified. Use the x-range and y-range text areas to specify the range you wish to view. Leave them blank for the default behavior of displaying the entire dataset. The Y-axis ticks button toggles between On and Off. Turn this off when creating a stacked plot if you don't want meaningless tick marks running all the way up the y-axis. Instead, there will only be tick marks at the "zero" level and, if specified, the level of each dotted line for the plot.

The CharSize text field allows you to set the character size. The default value is 1. Entering 2 would cause all characters to be twice as big and similarly entering 0.5 would cause all characters to be half the normal size. The "Dotted line at y=" text field allows you to overplot a dotted line at any specific y-value (for instance a continuum at y=1 if plotting a spectrum that has been normalized to a continuum level of 1). The "Cutoff at y=" text field allows you to cut off all data above this level for each plot. This can be useful when creating a stacked plot. For example, if you are plotting spectra that contain a strong emission line, use this to cut off the values above a certain level and zoom in on the details of weaker emission lines. This allows you to put a smaller offset between the plots and the smaller range shows more detail. The Stacked and Overplot buttons toggle the plotting behavior between stacking plots on top of each other and plotting multiple files directly over each other. The default behavior is Stacked and creates a plot where the files are stacked above each other with a y-axis offset for each. Click Overplot to plot the data directly on top of each other with the same "zero" level for all of the plots. If you choose Overplot, different linestyles will be used for the different files: solid for the first file, dotted for the second, dashed, dash dot, dash dot dot, and long dashes for the 6th file. If there are still more files, the pattern then repeats with the 7th being solid and so on.

Click the “Add Text” button to add text anywhere in the plot. It brings up a window prompting for X and Y coordinates, the text, character size, alignment, orientation, and color. If adding text to a plot on the screen, clicking on the screen will automatically fill in the X and Y coordinates for you. Alignment ranges from the default value of 0.0 (text begins at specified point) to 1.0 (text ends at specified point) and Orientation is from 0 to 360 degrees. Note that when adding text to a Postscript, or PDF file or a plot being sent to the printer, you are not able to click on the screen to obtain the X and Y coordinates because you are plotting to a different device. The easiest way to add text to these plots would be to first plot to the screen, then click everywhere you want to add text, and those coordinates will be printed out in the IDL window. Then export to the file, click “Don’t Finalize”, and add the text, typing in the coordinates that were printed out in the IDL window. Then click “Finalize” when you are done. Alternatively, beginning in v3.0, you can add text to the plot on the screen and then export the plot using Screenshot .PS, Print Screenshot, or Screenshot PDF.

If you are exporting a plot to a Postscript or PDF file or sending it to a printer and you clicked “Don’t Finalize” from the export menu, click the “Finalize” button after you are done adding text to the plot to close the file or send it to the printer.

The X-axis and Y-axis buttons allow you to toggle each axis between linear and logarithmic. The “More Options” button (v2.0+) allows you to set more IDL plot options, including Position, XYMargin, XYMinor, XYtickinterval, XYticklayout, XYticks, XYtickname, and XYtickv. See the IDL documentation for an explaination of what each of these options does.

Click the “Multiplot” button (v2.0+) to create a plot with multiple panels in one window. Set the number of rows and columns and specify auto or manual. In auto mode, each file or function in your file list is automatically plotted in its own panel. For instance if you have 4 files entered and you want to create a 4-panel

plot with 2 rows and 2 columns and have each file plotted, select 2 rows, 2 colums, and auto, click Apply, then click either Plot or Export. If you want to plot more than one file or function per panel, use manual mode. In manual mode, each time you click Plot, another panel is added with the files currently listed plotted just as they would normally be except within a panel. When exporting in manual mode to a Postscript or .PDF file or the printer, click Export after you set your Multiplot options, and enter the file info. But do not finalize if you have more panels to add. Add any text you want to this first window now, as this is your only chance (this applies to just plotting as well as exporting). Starting with the second plot, click Plot instead of export as you already entered a filename (or selected the printer). When finished, click Finalize at the bottom. To export to a JPEG, select Export–JPEG at any time and the current contents of the screen will be exported to a JPEG file. The same goes for the exporting options Screenshot .PS, Print Screenshot, and Screenshot PDF.

Clicking the "Help" button (v2.0+) enables the online help. A help file similar to this appendix is printed to your current terminal window, 23 lines at a time. Pressing the space bar goes to the next page, and pressing 'q' quits help. Beginning with v3.01, if you have correctly configured your web browser of choice using gatorplot_config, clicking help will open a browser window displaying the help file in HTML. Finally, the UF button is a little gadget added to make GatorPlot more Gator friendly.

C.9 GPScript

GPScript is a systemwide scripting language (implemented in v4.0) that can be used to automatically do almost anything that can be done interactively in GatorPlot. The procedures and functions of GPScript are an extension of IDL, so your scripts may contain any combination of IDL and GPScript functions/procedures. To run a script, at an IDL prompt, enter: gatorplot,script='scriptname', where

scriptname is the name of the file that contains the script. The script should be set up as and IDL procedure. Note that the file must be in your IDL_PATH or your current directory so that IDL can see it since it will not be precompiled. Also, the filename must match the name of the procedure. For instance, you could have script called data in a file named data.pro:

```
pro data
gpsload,'file.fits'
print,gpsDataAt(1549)
end
```

This simple script could be run by entering: gatorplot,script='data' at an IDL prompt. Below is a list of all GPScript functions/procedures, their calling sequences, and some examples.

- gpsload: Loads a file or several files into memory.

  `gpsload, files[, xcol=, ycol=, funmin=, funmax=, funpts=]`

  files is a string or string array containing one or more filenames or functions. The files can be either FITS or ASCII or you can enter a function just as you would in the Enter Filenames box in GatorPlot. The optional xcol and ycol keywords specify the columns in the ASCII file to read X and Y data from. The optional funmin, funmax, and funpts keywords specify the minimum and maximum X-values to be used in generating the function and the number of points to be used for the function. xcol and ycol are mandatory if loading an ASCII file, as are funmin, funmax, and funpts if loading a function. Examples:

  ```
  gpsload,['file1.fits','file2.dat'],xcol=1,ycol=3
  gpsload,'f(x)=sin(x)',funmin=0,funmax=6.28,funpts=100
  ```

- gpsmultiplot: Use this to setup a multi-panel plot.

  `gpsmultiplot, rows, cols [, /auto]`

rows and cols are the number of rows and columns in the multipanel plot. If auto is not set, then all files currently loaded will be plotted (either stacked or overplotted) in one panel of the multi-panel plot each time gpsplot is called. If the optional keyword auto is set, then each file currently loaded will be plotted in its own separate panel. Examples:

```
gpsmultiplot,2,2
gpsmultiplot,3,2,/auto
```

- gpsplot: Plot currently loaded files to the screen. If a multi-panel plot is desired, gpsmultiplot should be called first. gpsplot must be called before exporting any plots.

  ```
  gpsplot, [xmin=, xmax=, ymin=, ymax=, xtitle=, ytitle=, title=,
  /xlog, /ylog, /overplot, charsize=, linestyle=, color=, psym=,
  offset=, label=, xmargin=, ymargin=]
  ```

  In its simplest form, you can just call gpsplot without any keywords. The optional keywords xmin, xmax, ymin, and ymax of course allow you to specify the X and Y ranges to be displayed. xtitle, ytitle, and title allow you to enter strings that will be used as axis or plot titles. Specifying /xlog or /ylog makes that axis logarithmic instead of linear. Specifying /overplot causes files to be plotted over one another in the case that more than one file is plotted in a panel (or window). By default, the files will be shown stacked on top of each other. charsize sets the character size (default = 1), linestyle the linestyle (0-5, see IDL help), and psym the plotting symbol (again, see IDL help). color can be specified by either a string containing a single number for color index (e.g., color='1019') or as an RGB triple as in the GatorPlot GUI (e.g., color = '0 255 255' for blue-green). offset takes an array argument of length up to files-1 and overrides the default offset when displaying a stacked plot (e.g., if you have three files loaded, to be displayed

in a stacked plot, off=[2,5] would set the zero-level for file 2 at 2 and for file 3 at 5). label takes a string array as an argument and will label each data set on the right side of the display (e.g., label = ['File 1', 'File 2', 'File 3']). Finally, xmargin and ymargin each take vectors of length two and perform the same task as the xmargin and ymargin keywords in the IDL PLOT procedure. Examples:

```
gpsplot,xmin=1450,xmax=1720,ymax=5, xtitle='Wavelength',ytitle='Flux',
charsize=1.5, color='255 0 0'
gpsplot,title='Stacked Spectra',/xlog, linestyle=[0,1,2],
color=['255 0 0', '0 255 0', '0 0 255'], offset=[2,5], label=['File
1', 'File 2', 'File 3']
```

- gpsexport: Export plots that have been created with gpsplot. Plots can be exported to Postscript, JPEG, PDF files, the printer, or the data can be exported to a FITS or ASCII file.

  ```
  gpsexport, filename [, /printer, /color, /landscape, /multiplot,
  xsize=, ysize=]
  ```

  filename is the export file to contain the plot/data. The file type is automatically determined by the file extension (.jpg or .jpeg = JPEG file, .fit or .fits = FITS file, .ps or .eps = Postscript, .pdf = PDF, .dat or .txt = ASCII). Specify the /printer keyword to send the plot to the printer (UNIX/Linux only). Specify /color to export as true color instead of the default greyscale. Specify /landscape to export in landscape instead of portrait mode. Specifying /multiplot makes GPScript check if all panels have been created and only exports if they all have (e.g., if you're doing a 9-panel, 3x3 plot but you call gpsexport after only plotting 7 files and specify /multiplot, then nothing will happen. If you call it again after plotting all 9 files, again specifying /multiplot, then the plot will be exported. If you don't specify /multiplot

then GPScript exports without checking.) xsize and ysize can be set to the X and Y size of the Postscript, PDF, or printer file in inches. Again, using a .eps extension causes the plot to be exported to an encapsulated postscript file. Examples:

```
gpsexport,'image.jpg',/color
gpsexport,'figure.eps',xsize=7,ysize=9
gpsexport,/printer
```

- gpssmooth: Apply smoothing to loaded data.

  ```
  gpssmooth [, /boxcar, /binomial, npts=# of points, iter=#
  iterations]
  ```

  See Smoothing (above) for more details. If not specified, npts defaults to 5 and iter to 1. Example:

  ```
  gpssmooth, /boxcar, npts=3
  ```

- gpsmath: Perform arithmatic operations on data.

  ```
  gpsmath, files1 [, factor1=,const1=,oper=,files2=,factor2=,const2=,out=]
  ```

  filelist1 is an intarr containing the index of each file (starting with 0) to include in the arithmatic operation. factor1 is an intarr containing the factor to scale each corresponding dataset by eg. a0*x0+a1*x1+... const1 is a constant number to be added to this. oper can be '+', '-', '*', or '/'. files2 is an intarr similar to files1, with factor2 and const2 corresponding to factor1 and const1. Finally, out is a string containing a filename for the data to be written to after the arithmatic is performed. If the filename ends in '.fits' or '.fit', the data will be written as a FITS file, otherwise it will be written as an ASCII file. So there are many possible combinations of operations that can be performed by gpsmath: (a0*x0+a1*x1+a2*x2+...+c1) +, -, * , or / (b0*x0+b1*x1+b2*x2+...+c2) Examples:

  ```
  gpsload,['a1.dat','a2.dat','a3.dat','a4.dat'] ;load 4 files
  ```

`gpsmath, [0,1], factor1=[1,2], const1=2, out='test.fits'`

writes a FITS file with the result of a1.dat + 2*a2.dat + 2.

`gpsmath, [0,3], const1=1, oper='/', files2=[1], factor2=[2], out='test.dat'`

writes an ASCII file with (a1.dat + a3.dat + 1) / (2*a2.dat).

- gpsintegrate: Integrate over a user-defined baseline (see Integrate above).

  `gpsintegrate, x1, x2 [,y1, y2 , out=]`

  x1, x2, y1, and y2 are the X and Y coordinates of the start and end of the user-defined baseline Leaving off the y-values results in them being set to 0. out specifies a file to output the results to. If no file is specified, it defaults to gpsintegrate.dat. Examples:

  `gpsintegrate, 1500, 1600, out='results.dat'`

  integrates over a line drawn from (1500,0) to (1600,0) and outputs results to results.dat.

  `gpsintegrate, 1500, 1600, 0.5, 1.5`

  integrates over a line drawn from (1500,0.5) to (1600, 1.5) and outputs results to gpsintegrate.dat. gpsintegrate can also be used as a function:

  `result = gpsintegrate(x1, x2 [,y1, y2, out=])`

  When used as a function, the result returned is a string containing 4 space-delimited columns: the integration (flux), center, equivalent width, and filename. If more than one file is loaded when gpsintegrate is called, result returns as a string arrray with one element for each file. Example:

  `flux=strparse(gpsintegrate(1500,1600,1,1),1)+0.`

- gpsfwhm: Estimate the FWHM of an emission/absorption line without fitting (see FWHM above).

  `gpsfwhm, contin_x1, contin_x2 [, xpeak=, ypeak=, out=]`

contin_x1 and contin_x2 are the x-coordinates of two continuum points, one on each side of the line. xpeak and ypeak allow you to optionally specify the X and/or Y coordinates of the peak/trough of the line. If you do not specify these, they will be automatically calculated. out specifies an output file for the results (defaults to gpsfwhm.dat). Examples:

```
gpsfwhm, 1442, 1712
gpsfwhm, 1442, 1712, xpeak=1549, out='civ_fwhm.dat'
```

Like gpsintegrate, gpsfwhm can also be used as a function:

```
result = gpsfwhm(contin_x1, contin_x2 [, xpeak=, ypeak=, out=])
```

When used as a function, the result returned is a string containing 4 space-delimited columns: the FWHM in x-axis units, FWHM in km/s, center, and filename. If more than one file is loaded when gpsfwhm is called, result returns as a string arrray with one element for each file. Example:

```
fwhm=strparse(gpsfwhm(1450,1720),2)+0.
```

- gpslinfit: Linear fit to data (see GPFIT: Linear Fit).

  ```
  gpslinfit [, range=, /poission, out=, plotps=, plotxrange=]
  ```

  range is a string array representing the ranges in the data to be included in the fit. It is entered just as you would enter in in the Range box in GPFIT (i.e. ['1825 1850', '2000 2025'] would mean fit using data from 1825–1850 and 2000–2025). Leaving off range makes the range all X's. The /poisson option sets poisson errors. out specifies the output file for the fit parameters (default linfit.gp). plotps specifies a filename to plot (in postscript) the data and the fit. If no filename is specified, the plot is not created. plotxrange allows you to specify the x-range for the postscript plot. Example:

  ```
  gpslinfit, range=['1825 1850', '2000 2025'], /poisson, plotps='fit.ps'
  ```

- gpsplawfit: Powerlaw fit to data (see GPFIT: Powerlaw Fit).

  ```
  gpsplawfit [, range=, /poisson, coeff=, out=, plotps=, plotxrange=]
  ```

range is a string array (see gpslinfit) and /poisson sets poisson weighting. gpfcoeff takes your initial guesses for the coefficients of aX^b in the order [a,b]. out, plotps, and plotxrange as described in gpslinfit. out defaults to plawfit.gp if not specified. Example:

```
gpsplawfit, range=['1440 1460', '2000 2025'], /poisson, coeff=[700,
-0.8]
```

gpsplawfit may be called as a function as well:

```
result = gpsplawfit([range=, /poisson, coeff=, out=, plotps=,
plotxrange=])
```

When used as a function, the result returned is a two-element array containing the coefficients in the order [a,b].

- gpsmgfit: Multiple Gaussians fit to data (see GPFIT: Multiple Gaussian Fit).

  ```
  gpsmgfit in= [,range=,/poisson,const=,out=,plotfile=,plotps=,plotxrange=]
  ```

  in specifies the input file containing the fit parameters (as described in GPFIT: Multiple Gaussian Fit). range, poisson, out, plotps, and plotxrange as described above. out defaults to multgaussfit.gpfit. const is a constant offset from 0 (e.g., normalized continuum level). plotfile specifies a filename to receive the plotfile containing the data, overall fit, and value of each Gaussian at every X-value. It defaults to gpmultgaussfit.dat. Example:

  ```
  gpsmgfit, range=['1510 1680'], in='civ.gp', const=1, plotps='civfit.ps'
  ```

- gpspolyfit: Polynomial fit to data (see GPFIT: Polynomial Fit).

  ```
  gpspolyfit order= [, range, /poission, coeff=, out=, plotps=,
  plotxrange=]
  ```

  order specifies the order of the Polynomial to be fit (e.g., 2 = quadratic, 3 = cubic). range, poisson, out, plotps, and plotxrange as described above. out defaults to polyfit.gp. coeff is an array containing optional initial guesses at

the polynomial coefficients of $a_0 + a_1X + ... + a_nX^n$ in the order [a_0, a_1, ..., a_n]. Example:

```
gpspolyfit, order=3, coeff=[3, 0, -2, 1], out='poly.dat'
```

- gpsabsfit: Fit Absorption Lines (see GPFIT: Absorption Line Fit).

  ```
  gpsabsfit, in= [,range=,/poisson,const=,out=,plotfile=,plotps=,
  plotxrange=]
  ```

  in specifies the input file containing the fit parameters (as described in GPFIT: Absorption Line Fit). Everything else as described in gpsmgfit. out defaults to multabsfit.gpfit and plotfile defaults to gpmultabsfit.dat. Example:

  ```
  gpsabsfit, range=['1470 1600'], in='absparams.dat', const=19,
  plotps='abs.ps'
  ```

- gpsudfit: Fit User-Defined Function (see GPFIT: User-Defined Fit).

  ```
  gpsudfit, name, f=, pders=, coeff= [,range=,/poisson,out=,plotps=,
  plotxrange=]
  ```

  name specifies the name of the user-defined function for purposes of generating it (one word). f gives the function as a String in terms of X and parameters A[0], A[1], ... (e.g. A[0]*sin(A[1]*X)+A[2]). pders is a String array containing the partial derivatives with respect to each parameter (e.g. df/dA[0], df/dA[1], ...). coeff gives the initial guesses for each parameter. range, poisson, out, plotps, and plotxrange as described above. out defaults to udfit.gp. Example:

  ```
  gpsudfit, 'sinefit', f='A[0]*sin(A[1]*X)', pders=['sin(A[1]*X)',
  'A[0]*X*sin(A[1]*X)'], coeff=[4, 2], plotps='sine.ps'
  ```

- gpsDataAt: Checks if there is non-zero data at a specified X-value and returns 1 (true) or 0 (false).

  ```
  result = gpsDataAt(x)
  ```

  Examples:

```
print, gpsDataAt(1549)

if gpsDataAt(1549) then print,'There is data at 1549.'

a = gpsDataAt(1549)
```

- gpsMean: Returns the mean value of all data in a specified range.

  ```
  result = gpsMean(w1, w2)
  ```

  w1 and w2 are the X-values used to determine the range. Examples:

  ```
  print, gpsMean(1440, 1460)
  ```

  returns the mean value of the data in the 1440–1460 interval.

  ```
  if gpsDataAt(1450) then a=gpsMean(1440, 1460)
  ```

- gpsMedian: Returns the median value of all data in a specified range.

  ```
  result = gpsMedian(w1, w2)
  ```

  Example:

  ```
  print, gpsMedian(1440, 1460)
  ```

- gpsStdDev: Returns the standard deviation of all data in a specified range.

  ```
  result = gpsStdDev(w1, w2)
  ```

  Example:

  ```
  s = gpsStdDev(1440, 1460)
  ```

- gpsMax: Returns the maximum value in a specified range.

  ```
  result = gpsMax(w1, w2)
  ```

  Example:

  ```
  m = gpsMax(1500, 1600)
  ```

- gpsWhereMax: Returns the corresponding X-value to the maximum value in a specified range.

  ```
  result = gpsWhereMax(w1, w2)
  ```

  Example:

  ```
  x = gpsWhereMax(1500, 1600)
  ```

- gpsMin: Returns the minimum value in a specified range.

```
result = gpsMin(w1, w2)
```

Example:

```
m = gpsMin(1500, 1600)
```

- gpsWhereMin: Returns the corresponding X-value to the minimum value in a specified range.

  ```
  result = gpsWhereMin(w1, w2)
  ```

  Example:

  ```
  x = gpsWhereMin(1500, 1600)
  ```

- gpsloadimg: Loads an image into memory.

  ```
  gpsloadimg, filename
  ```

  filename is either a JPEG or FITS image. Example:

  ```
  gpsloadimg, 'picture.jpg'
  ```

- gpsexportimg: Exports a previously loaded image to JPEG, FITS, Postscript/.eps, PDF, or the printer.

  ```
  gpsexportimg, filename [, /printer]
  ```

  The file-type is automatically determined by the ending (.jpg/.jpeg, .fit/.fits, .ps/.eps, or .pdf). Specifying the printer keyword sends the image to the printer (UNIX/Linux only). Examples:

  ```
  gpsexportimg, 'picture.ps'
  ```

  ```
  gpsexportimg, /printer
  ```

- gpsnegative: Takes the negative of a previously loaded image.

  ```
  gpsnegative
  ```

- gpsIntensityAt: returns the intensity of a given pixel in an image.

  ```
  result = gpsIntensityAt(x, y)
  ```

  x and y are the x and y coordinates of the desired pixel. result returns as a number between 0 and 255 for greyscale images or a 3-element array

representing the R, G, and B intensities as 3 numbers between 0 and 255 for a true color image. Example:

```
a = gpsIntensityAt(50, 100)
```

- gpsexecom: Brings up a window that allows you to enter IDL and/or GPScript code to be executed immediately. See "Exec Command" button above.

  ```
  gpsexecom
  ```

- gpsresample: allows you to resample your data in various ways (see the "Resample" button above).

  `gpsresample [, z=, step=, xmin=, xmax=, outf=]` z is the redshift, step is the stepsize between pixels, and xmin and xmax are the start and end x-axis values. outf is an output file that can be either FITS or ASCII. If the filename entered ends in '.fit' or '.fits', then it will be a FITS file. Otherwise, it will be an ASCII file. Example:

  ```
  gpsresample, z=1.5, step=0.5, outf='resampled.dat'
  ```

- gpsdrawline: draws a line anywhere in the current plot (see the "Draw" button above).

  ```
  gpsdrawline, X1, X2, Y1, Y2 [,color=, linestyle=]
  ```

  X1, X2, Y1, and Y2 are the X and Y coordinates of the two endpoints of the line. color can be a number or a string with space-delimited RGB values of a color. linestyle can be from 0 to 5. Example:

  ```
  gpsdrawline, 1549, 1549, 3, 5, color='255 0 0', linestyle=1
  ```

- strparse: While technically not a part of GPScript, the strparse function is used by several procedures in GatorPlot and GPScript and is thus included with GatorPlot. strparse allows you to parse a string delimited by whitespace (tabs or spaces).

  ```
  result = strparse(string, column)
  ```

string is any string and column is the column number you want to extract. Examples:

```
s = 'column1 5 3 6 column5'
print,strparse(s,1)
```

prints 'column1' to the screen.

```
b = strparse(s,2)+0.*strparse(s,4)+0.
```

stores 30 (5*6) in the variable b.

- get_color: Also not technically a part of GPScript, but included with GatorPlot, get_color converts an RGB triple into a number between 0 and 2^{24} that IDL uses to represent that color.

  ```
  result = get_color(r, g, b)
  ```

  r, g, and b are all integers between 0 and 255. Examples:

  ```
  a = get_color(255, 112, 0)
  plot, x, y, color=get_color(0, 255, 255)
  ```