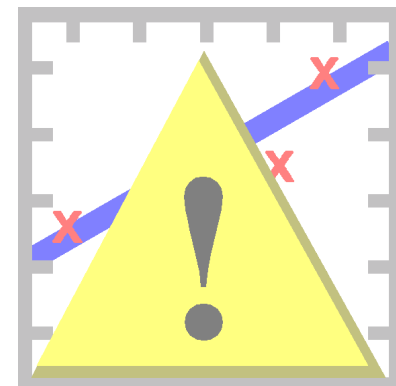




*a data analysis program*

Version 4.5  
September, 2007

## **User's Guide**



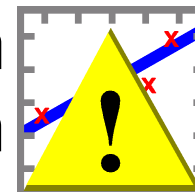
## Contents:

1. Program Description	3
1.1 Sample Splat! Session	3
1.2 Program Uses	8
1.3 Program Organization	9
1.3.1 Command Line Interpreter	9
1.3.2 Data Storage & Int. Flags	10
2. Command Overview	14
2.1 Entering & Moving Around Data	15
2.1.1 File Input	15
2.1.2 Keyboard Entry	17
2.1.3 Create Data from a function	17
2.1.4 Moving Data Around	18
2.2 Analyzing Data	19
2.3 Processing Data	20
2.4 Plotting Data	21
3. Command Reference	23
4. Installation & Setup	46
4.1 Windows version	46
4.2 DOS version	52
4.3 Linux version	55
4.4 SPLAT.INI configuration file	57
4.5 Macro commands	58

### **New features in version 4.5**

- ☐ New fitting functions (Morse, Maxwell) and options (fixed width Gaussian, /Sub)
- ☐ Derivative command
- ☐ Added functionality to Point and Line commands

# 1 Program Description



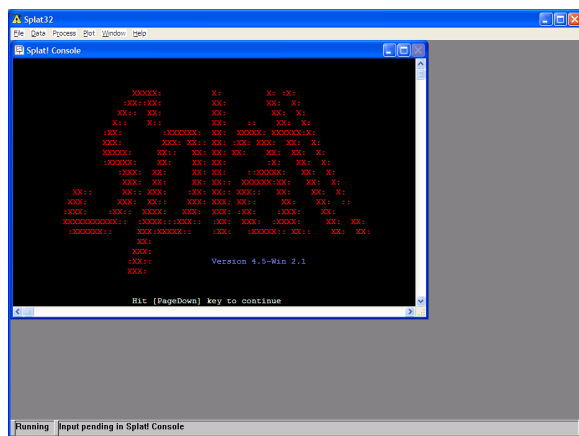
Splat! is a general purpose data analysis program. It uses a simple, intuitive command line interface to input, manipulate, and display data. Think of it as a scientific calculator for processing computer files- it can preform many of the same data manipulation functions, and can produce primitive graphical output, but like a calculator the output options are a bit limited.

Splat! is not a spreadsheet program like Excel (<sup>TM</sup>Microsoft). It is also not a graphical plotting program like Origin (<sup>TM</sup>OriginLab) or IGOR (<sup>TM</sup>Wavemetrics). Nor is it a symbolic mathematical program like Mathematica (<sup>TM</sup>Wolfram), Maple (<sup>TM</sup>Maplesoft), and Mathcad (<sup>TM</sup>Mathsoft). To illustrate what Splat *is*, rather than spend a lifetime describing what it is not, it is probably best to run through a quick example.

## 1.1 Example Splat! Session

Let us assume we are interested in finding out how the cost of textbooks scale with how big the book is. Our data set is a set of X,Y values for each book: the number of pages (X) and the cost (Y). Our data is stored in two text files (standard human readable text files) called `soft.txt` (for soft cover books)

and `hard.txt` (for hardcovers). We start off our analysis by firing up Splat!<sup>1</sup>. Here is what the Windows XP version looks like:



After pressing the <Pg Dn> key, we have Splat read in the two files using the **READ**<sup>2</sup> command: at the '>' prompt we enter: `read soft.txt`, and `read hard.txt`. The first file contains 20 data points, the second file contains an additional 18 data points (for a total of 38 points). Now that we have read the data into Splat!, we can take a quick look at the data using the **PLOT** command. This is as simple as entering `plot` at the '>' prompt, or by selecting 'Plot' from the Plot pull-down menu (in the Windows version). However,

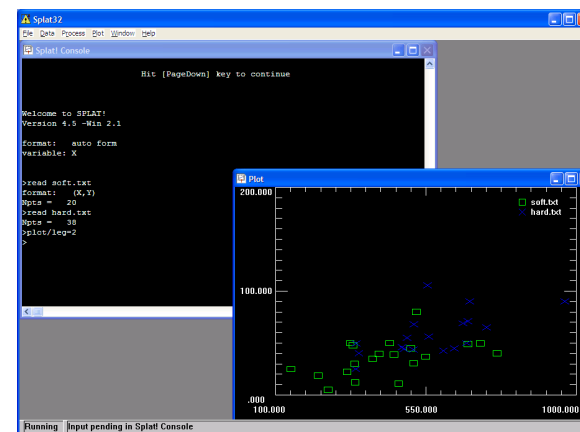
---

<sup>1</sup>See section 4 on the details of how to install Splat! for your operating system.

<sup>2</sup>In this manual Splat! commands are shown in bold all caps (so they stand out) when described in the text. The Splat! command line interpreter is case insensitive. Sample input/output are displayed in a fixed width font.

to make things a bit more complicated we also want to display a plot legend in the upper right corner, this is accomplished with `plot/leg=2`.

In the DOS version the text command line display is replaced by a full screen graphics display until the user presses <enter>. In the Windows version the plot shows up in a new window behind the command console window. Pressing <tab> switches the focus between the two sub-windows<sup>3</sup>.

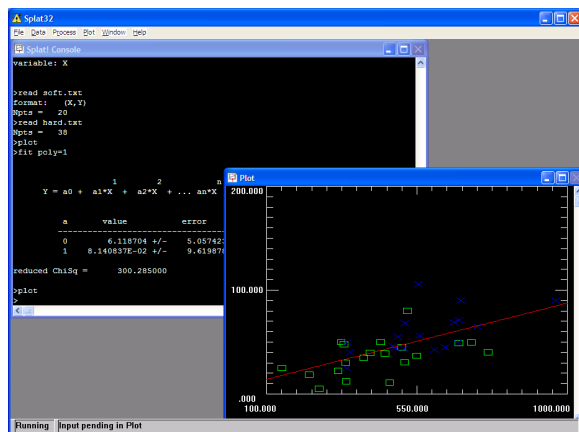


Note that the data points from the two different input files are displayed with different symbols. This looks nice, and makes the output easier to understand, but as far as Splat! is concerned all data points are treated equally regardless of their source. Only in plotting is there any distinction made (and only as long as the order of the data points is maintained).

---

<sup>3</sup>You might be tempted to use the mouse to select the plot window- not recommended. If you do, you must use the mouse to also re-select the Splat! Console (<Tab> will not work).

Looking at the data, there is only a slight dependence on text book price with page count. We can further quantify this by fitting a line to the data and extracting the slope (with units of \$/page). To do this we **FIT** a first order polynomial: `fit poly=1`. The *a1* coefficient is the slope we are after. In addition to the fitted coefficients displayed in the table, Splat! also produced a line output for the plot. If we replot the data by typing `plot` we can see how well this line fits the data.



In addition to polynomials, Splat! has a number of other functions that can be used in fitting, but they don't seem warranted in this case. Splat! was designed to handle experimental data with error bars, but our text book data doesn't have any, so the uncertainties reported in the fitted coefficients and the goodness of fit parameter (reduced  $\chi^2$ ) have little meaning.

In addition to fitting, we can also get statistics on a single variable. For example, if we are interested in the cost of books, we can get a number of statistical number with the

**STAT** command: `stat y` (recall the y-column is the cost).

```
>stat y
```

Statistics for 38 data points:

```

Average   =      46.655790
Avg. Dev. =      15.935350
Std. Dev. =      22.039090
Variance  =      485.721300
Skew      =      5.434262E-01
Kurtosis  =      2.412953E-01
(mesokurtic)

```

```

Minimum =      4.950000
Maximum =     105.750000
Median  =      44.950000
Sum     =     1772.920000

```

Since most of the sampled books were purchased long ago, the average price is quite low compared to present textbook prices. Good thing Splat! is free.

When we are all done with our analysis, we **EXIT** the program with the `quit`<sup>4</sup> command. Other than outputting a new data file, there is no way to save your work in Splat!. Hence, when you quit there is no prompt to save your work before exiting.

---

<sup>4</sup>An attempt at verbal judo. Both `exit` and `quit` close the program. Many Splat! commands use multiple aliases.

## 1.2 Program Uses

Splat! is generally used to either manipulate 'raw' data into a format for another program (such as a fancy plotting program), or to extract some useful information from a data set.

### □ Data Manipulation

One of the (few) strong points of Splat! is its averaging capabilities. While averaging, Splat! can create statistical error bars or do weighted averaging of data with pre-existing error bars. If so directed, Splat! can also recognize and discard outliers/bad data points while averaging. Complex mathematical transformations can also be applied to the data. Simple commands can also be used to rescale data.

### □ Data Analysis

As illustrated in the previous section, Splat! has a number of commands used to extract information from *xy* and single variable data sets via fitting and statistical analysis.

To support these primary functions, Splat! has a number of tools for importing and exporting data, and displaying data. Additional commands control how the program displays and interprets data.

## 1.3 Program Organization

Splat! has three main components that you the user should be aware of:

- the command line interpreter
- the data storage arrays
- internal configuration flags

### 1.3.1 Command Line Interpreter

You interact with Splat! via a command line interface. The Windows version of Splat! is essentially a shell using menus and dialog boxes to create the equivalent text-based commands. Commands are of the form:

verb[/optional qualifiers] parameter

Most basic operations in Splat! can be accomplished without any qualifiers, and even the parameters are often unnecessary for 'default' conditions. While these default conditions are appropriate in most cases, they may not be the best for your particular application in which case you have a bit more typing to do.

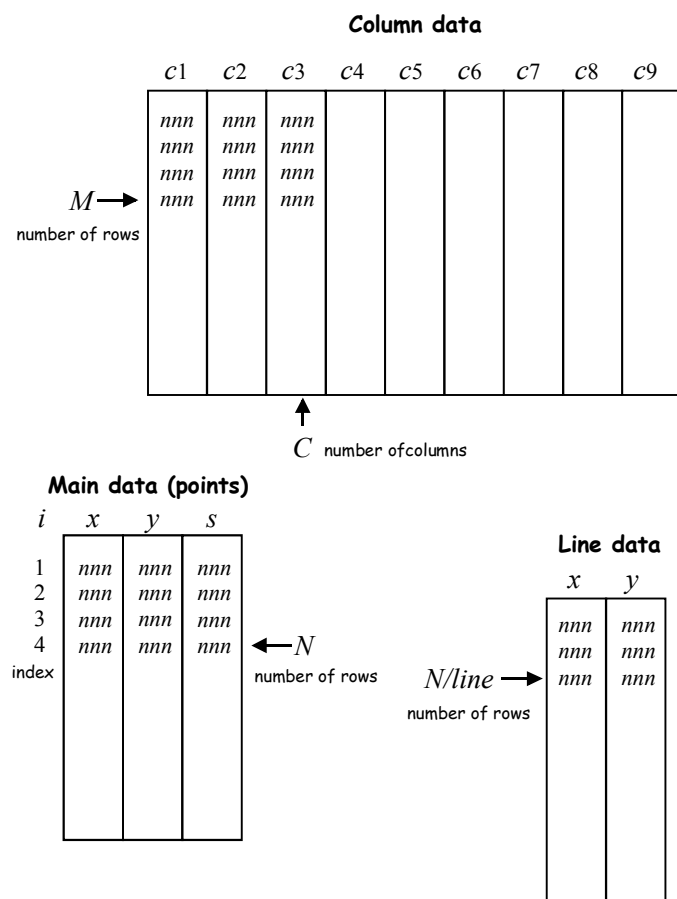
#### Note of caution:

Splat! assumes you (the user) is reasonably intelligent and knows what you are doing. Hence, there is little error checking built into the program. It is very easy to have Splat! attempt to analyze non-existent data, or have it attempt to divide by zero (which occurs in many incarnations). Naturally this will crash the program. It is recommended that you save your work before attempting complicated operations such as fitting or math operations.

Splat!'s command line interpreter is case insensitive, and most commands can be abbreviated significantly. For example, the **AVERAGE** command will be executed for any command that begins with an AV, thus all of the following are acceptable: AVERAGE, AVE, AVG, AV, av, aVerySilly, ...

### 1.3.2 Data Storage Arrays & Internal Flags

Splat! has three main data storage arrays: (i) the standard *xy*s array, (ii) the column data array, and (iii) the *xy* line data array.



Most commands either work exclusively on the *xy*s data or use these as the default. The *xy*s data is plotted as points. A

separate *xy* array holds data that will be plotted as a line. While it is possible to transfer data between these two arrays, this is somewhat awkward. It is much easier to transfer data between the standard *xy*s data array with the column data worksheet.

Each of the three array has a maximum size of about 50,000 points (you can find out this size with **HELP**). Each array has a flag that tells how many rows are currently being used. For the *xy*s array this is called *N*, for the column array it is called *M*. A further variable *C* keeps track of how many columns are filled. The number of values in the *xy* line data is also called *N*, but this *N* is different than the *N* of the point data. Confused? Don't worry, Splat! generally keeps track of all these variables for you. When you reset the program using the **CLEAR** command the program doesn't erase your old data, it just sets the number of points *N* to zero. Using the **MAKE** and **SET** commands you can directly set the values of these counters.

The **FORM** flag informs Splat! about the type of data in the *xy*s array. In form=1, only a single column of data contains useful data. In form=2 either the *x* (or *y*) column contains data and the *s* column contains the uncertainty/error bar for the data column. More typically one uses either form=3 which is for *xy* data or form=4 for *xy* data with the uncertainty in *y* stored in column *s*.

How does this all work? Recall the example in section 1.1 of a data set consisting of the data pairs of page counts and book costs. When you **READ** in a file with two columns of data Splat! defaults to assuming the columns correspond to the values for *x* and *y*. If you use the **LIST** command to look at the data Splat displays only the *x* and *y* columns from point 1

to  $N$ . To be more concrete, let us consider an example where our data consists of the following 9 values:

```
1 10
2 15
3 8
1 12
2 13.5
3 16
1 11
2 11
3 17
```

If we **READ** this into Splat! and **LIST** the data, Splat! displays only the x and y columns of data from 1 to 9:

```
Welcome to SPLAT!
Version 4.5 -Win 2.1
```

```
format:  auto form
variable: X
```

```
>read sample.dat
format:  (X,Y)
Npts =   9
>list
```

1	1.000000	10.000000
2	2.000000	15.000000
3	3.000000	8.000000
4	1.000000	12.000000
5	2.000000	13.500000
6	3.000000	16.000000
7	1.000000	11.000000
8	2.000000	11.000000
9	3.000000	17.000000

If we now proceed to **AVERAGE** the data, the three values at each x value will be averaged together. If we list the data now, we only have three data points. When Splat! averaged the data, however, it calculated the error in mean for each value and stored the value in the s column. Splat! doesn't know if we want to use this information, so it is 'hidden'

when we now **LIST** the data.

```
>avg
Npts =    3
>list
```

1	1.000000	11.000000
2	2.000000	13.166670
3	3.000000	13.666670

But if we switch to **FORM 4**, the newly created error bars are also listed,

```
>
>form 4

format:  (X,Y,S)
variable: X
>list
```

1	1.000000	11.000000	5.773503E-01
2	2.000000	13.166670	1.166667
3	3.000000	13.666670	2.848001

While not shown, if we now plotted the data each data point would include a y error bar.

## 2 Command Overview



As mentioned in section 1.3.1, Splat! commands use the following syntax:

Verb/qualifier(s) parameters

All qualifiers and most parameters are optional. If you omit a required parameter, Splat! will prompt you for it. Unless the parameter required is obvious, you can usually type in a ? to get a list of available options. In this documentation, optional parameters and qualifiers are enclosed within brackets.

One qualifier that is supported by many commands is the range option, /n1:n2. This limits the Splat! command to only the points between n1 and n2. For example, LIST/4:8, would only list data points 4 through 8. If the first number is omitted, the lower limit defaults to 1, while if the second number is omitted, the upper limit defaults to the number of data points. Except when used with PLOT, the numbers can be replaced with an expression including the variables N and M. For example, if you only want to look at the last 11 data points in a list of 100, you could use any of the following:

```
LIST/90:100
LIST/90:
LIST/n-10:n
LIST/n-3!-2*pi+e:n
```

There are a lot of Splat! commands. A few are far more useful than others. Here is the big picture:

Loading/Saving data:	<b>READ, WRITE, ENTER</b>
Examining data:	<b>LIST, BIN, PLOT</b>
Plot control:	<b>PLOT, POINT, LINE, MARK, LABEL</b>
Analyzing data:	<b>STAT, FIT, D, DD, INTEGRATE</b>
Processing data:	<b>MATH, FFT, SORT, AVERAGE, NORM, SMOOTH</b>
Moving data:	<b>ADD, PORT, EDIT, MATH</b>
Controlling Splat!:	<b>FORM, SET, MAKE, CLEAR, CLS</b>
Worthless features:	<b>CALC, YO, ECHO</b>
Worthwhile features:	<b>HELP, CD, DIR, \$</b>

Detailed information on each command can be found in the Section 3. Here we present a brief overview of useful commands in each main area.

### 2.1 Entering Data and Moving it Around

There are three main routes of getting some data into Splat!, (i) reading it in from a file, (ii) entering it from the keyboard, or (iii) creating it internally from a function. Let us look at all three routes a bit more closely.

#### 2.1.1 File input

Splat! can read in most standard ASCII text files. Numbers on a line of data can be separated by spaces, tabs, or commas. Typically, Splat! starts up in form=5, auto format. In form 5 Splat! starts scanning your input file for a line starting with numbers (hence it skips over most lines starting with



characters) and figures out how many separate numbers are on each line. So for example, if the first line of the file is

```
12.00 452 400 7.5 1.0 20
```

Splat! sets *C* (the number of columns) to 6. And reads the data into *c1* through *c6* of the column worksheet. It then transfer the first three columns to the *xy*s data sheet (*c1*→*x*, *c2*→*y*, *c3*→*s*), and sets *N*, the number of *xy*s data points, equal to *M*, the number of column data points (which would depend upon how many of lines of data are in the file). Finally, Splat! changes the format from form=5 (auto format) to form=4 (*xy*s). Note that Splat! only reads data into the column worksheet when it is in form 5 (auto format) or form 6 (column format). Since Splat! switched from form 5 to form 4 after the first read operation in the above example, any additional read commands will only attempt to read in three columns of data into the *xy*s data array.

If the first line of data instead consisted of only two numbers, ie.,

```
12.00 452
```

Splat! sets *C* to 2. And reads the data into *c1* and *c2* of the column worksheet. It then transfer the first two columns to the *xy*s data sheet (*c1*→*x*, *c2*→*y*), and sets *N*, the number of *xy*s data points, equal to *M*, the number of column data points. Finally, Splat! changes the format from form=5 (auto format) to form=3 (*xy*).

It all seems confusing, but Splat! is simply trying to make life easy for you. Further information can be found under the **READ** and **FORM** commands in section 3. Since Splat!'s **READ** command supports wildcards and multiple input files in a single operation, it is possible to avoid a lot of problems (especially with multi-column data files) by reading in all the input files in a single step.

Alternatively, if you know how many columns of data you have, you can specify the format before reading in the data. For multi-column data input you specify **FORM 6** (multi column) and fix the number of columns with the **SET** command (i.e. SET C=6).

### 2.1.2 Keyboard Entry

A second way of getting data into Splat! is to type the data in by hand using the **ENTER** command. Typically you specify the form beforehand (with **FORM**), otherwise Splat! defaults to assuming you want to enter *xy* data. This form of data entry is pretty basic, and not recommended for entering more than a handful of points. You are far better off typing the numbers into a file using your system's text editor. In fact, Splat! uses the system's default text editor for the **EDIT** function which allows you to manually edit the input.

### 2.1.3 Creating Data from a Function

Another option for generating data is to specify the values using a function. As an example, let us assume we want 500 points of data ranging from *x*=0 to *x*=10 with *y* values of sin(*x*). To do this in Splat! we first create 500 points of data (from thin air) with the **MAKE** or **SET** commands, i.e. MAKE 500 or SET N=500. Next we need to set the *x* values. To do this we use the **MATH** function with the index *i* (which runs from 1 to 500) using MATH X=10\*(I-1)/(N-1). With the *x* values set, we next turn our attention to the *y* values, MATH Y=SIN(X). Finally, we switch to Form 3 so that Splat! knows to display only the *xy* values, FORM 3. With the Windows version of Splat! this whole process is automated for you by using the 'Create from function...' selection under the Data pull down menu. Column or line data can be generated in a similar fashion.

### 2.1.4 Moving Data Around

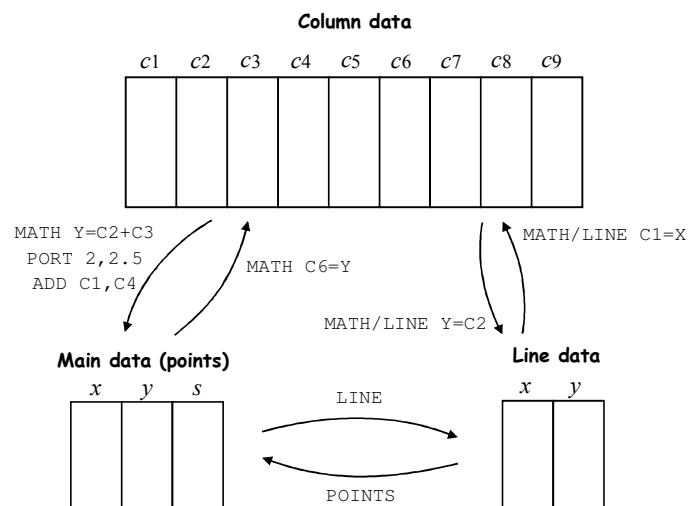
Suppose we had a data file with the following format

$x$ -value  $y_1$ -value  $y_2$ -value  $y_3$ -value.

This data should be interpreted as three  $xy$  pairs, i.e.  $(x, y_1)$ ,  $(x, y_2)$  and  $(x, y_3)$ . Unfortunately, Splat! doesn't know that and has instead (in auto format mode) read the  $x$ -value into  $c1$  (and  $x$ ), the  $y_1$ -value into  $c2$  (and  $y$ ), the  $y_2$ -value into  $c3$  (and  $s$ ), and the  $y_3$ -value into  $c4$ . It also picked form 4 ( $xy$ s) for the data format, what a mess. Relax, Splat! has some simple commands to move data between the three data arrays. To fix up this particular example case we do the following,

FORM 3	Switch to $xy$ format
SET N=0	Erase the existing messed up $xy$ s data
ADD C1, C2	transfer the $x, y_1$ pair
ADD C1, C3	transfer the $x, y_2$ pair
ADD C1, C4	transfer the $x, y_3$ pair

In addition to the **ADD** command, both the **MATH** function and the **PORT** command can be used to transfer data from the column array to the  $xy$ s data set. Only the **MATH** command can be used to transfer data back to the column array from the  $xy$ s data set. The **POINT** and **LINE** functions are likewise used to shuffle data between the  $xy$ s data set and the  $xy$  line data. See the appropriate entries in the Command Reference (section 3) for detailed information. The **EDIT** command, with the correct qualifier, can be used to edit any of the three data sets.



## 2.2 Analyzing Data

Once you have read some data into Splat! you generally want to do one of three things with it, (i) look at it (i.e. plot it), (ii) modify it in some way (i.e. average it) or (iii) extract some information from it without modifying it. The following two sub-sections deal with the first two options, this sub-section deals with the latter.

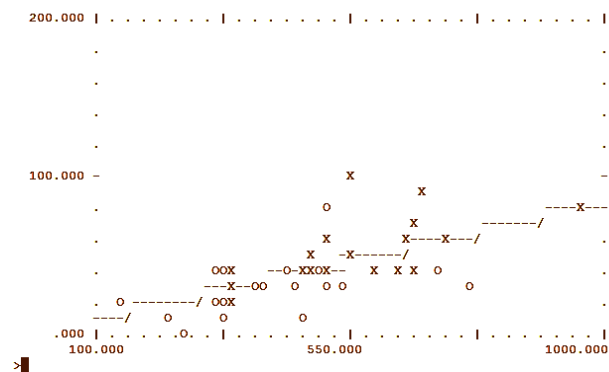
Splat! can analyze single variable data with either the **STAT** command or the **HISTOGRAM** command. Both are pretty lame. You are far more likely to use the **FIT** command to analyze  $xy$  or  $xy$ s data sets. The **FIT** command can be used to perform either a linear least squares fit or a non-linear least squares fit to the data, the choice depends on the function being fit. Fitting a polynomial is a linear fit (i.e. the fitted coefficients are linear combinations of the fitted basis), whereas fitting a Gaussian peak is a non-linear fit. With the important exceptions of problems with round-off errors linear

Some basic calculus operations can also be performed on the data without modifying it (too much) including **IN**TeGratation and derivatives (**D** for the first derivative, **DD** for the second derivative). These are pretty lame, the integration routine does sort the data, and only uses the simple trapezoid rule. The derivative functions are highly sensitive to noise (and round off error).

Splat! includes many commands to modify the *xy*s data in some fashion. Many of these commands can also be applied to the column data of *xy* line data with the appropriate qualifier. By far the most useful of these commands is the **MATH** function. This function can be used with wide variety of standard mathematical functions and operations to express any one variable (*X*, *Y*, *S*, *C1*, *C2*, . . . *C9*) as a function of itself and any combination of the others as well as the index *I*, the total number of points (*N* or *M*), and the mathematical constants *E* ( $e=2.718\dots$ ) and *PI* ( $\pi=3.14159\dots$ ). See the

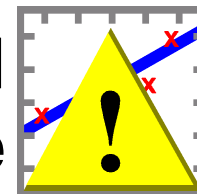
A related function is the **NORM** command which can be used to rescale data in one easy step (including error bars if in *xyz* form 4). Both **MATH** and **NORM** modify values without changing either the total number of points or the order of points in the data arrays. In contrast, **SORT** can be used to reorder the data. **AVERAGE** both sorts the data and reduces the number of data points. **SMOOTH** is the less intellectually honest, bastard offspring of **AVERAGE** that does some averaging of noisy data without decreasing the number of data points. The fourth circle of Hell is reserved for those who use this function. Also destined for trouble are those who venture to use the Fast Fourier Transform (**FFT**) function that sort-of works.

Ah, back in olden times the primary output device for Splat! was a text only VT-52 terminal. Splat!'s **PLOT**ting functions have progressed quite a bit since then, but the text output plotting routine can still be accessed with `PLOT/T`. Here for example is the text version of the plot from page-6:



The modern version of **PLOT** has oodles of optional qualifiers. These fall into three broad classes, (i) system-dependent, graphics-mode selection controls, (ii) options that control plot symbols and line types, and (iii) other qualifiers that affect how the plot looks. Category (i) qualifiers are best left to the detailed command reference (Section 3) and the program installation/setup section (Sec. 4). Category (ii) qualifiers are very user unfriendly and best not accessed directly. Instead, it is easier to adjust these parameters through the **MARK** and **LABEL** commands. Or, even easier, let Splat! worry about them with the `I_MARKER=1` option selected in the `SPLAT.INI` file (see Section 5). Indeed, considering the only one to see the output of Splat! is the user, there is little reason to mess around with how points and lines are displayed and labeled. Far more useful are parameters in category (iii) such as `/XLOG` and `/YLOG` that select a log scale instead of the usual linear scales. The `/DATA` qualifier prevents Splat! from rounding the plot limits to nice 'round' numbers and instead forces the limits to the actual data limits. This same qualifier can also be used with the **FIT** command to limit the fitted line (useful when 'round' numbers are unphysical (i.e. would be zero on a log scale)).

## 3 Command Reference



### \$command

In the DOS version, this spawns a DOS process to execute the given command. In the Linux version this spawns a process to execute the given unix command. In the Windows version, this command is no longer used since you can switch between windows easy enough.

### ADD[/n1:n2] Cn1 [,Cn2, Cn3]

Transfers data from column data set to the points (X,Y,S) data set. The exact destination is set by the format and default options. For example, in `form=3`, `ADD C1, C5` will append the C1 data to the X data, and append the C5 data to the Y data. See also the **PORT** command.

### AVERAGE[/SUM,/SD,/D] [delta\_x]

Averages all Y values with the same X values (or within `delta_x` wide bins). In `form=3` the S vector is set equal to the uncertainty in the mean ( $\sigma/\sqrt{N}$ ). If you instead want the 'real' standard deviation, use the `/SD` option. In `form=4`, S is the weighted average. When using the `delta_x` parameter, the X coordinate is set equal to the simple average of the data points. In `/SUM` mode, all Y values within a bin are added together instead of averaging. If the `/D` option is included Splat! will attempt to discard 'bad' data points. Suppose you had four points of 5, 5.1, 4.9 and 100. Clearly the 100 point is wrong, but this one bad point will shift the average from 5 to

29. With the /D option Splat! corrects for this by finding the median value and the standard deviation. Then Splat! discards all values more than 2 standard deviations from the median (a robust indicator of the mean). You can specify how many points to throw out by altering the good value range measured in standard deviations (ex: /D=3).

**BIN** [/n1:n2] num\_bins,[var]

A simple text histogram of the default variable or the specified variable. Same as **HIST**.

**CALC**

Reverse Polish Notation (RPN) calculator mode. **CALC** has it's own help command (or see **MATH**). In **CALC** mode, only the N variable is defined, X,Y,S, and I are not valid. Use QUIT, EXIT or control-Z to return to the Splat! prompt.

**CD** directory

Changes the default directory (same as the MS-DOS/Linux command).

**CLEAR**

Sets the number of points to zero, no line, and default format and variable.

**CLS**

Clears the screen.

**D**[/ADD, /n1:n2] [xwidth]

Numerical derivative of the Y data ( $dy/dx$ ). The derivative is found by fitting a line to all points within a sliding window of width [xwidth], if this is not specified it defaults to a few percent of the maximum x range. If there are

not enough points to fit a line within an interval, the size of the interval is increased. The x-spacing need not be constant, and can contain repeats. The results are saved as the line replacing any existing line unless the /ADD switch is included.

**DD**[/ADD, /n1:n2] [xwidth]

Numerical second derivative of the Y data ( $d^2y/dx^2$ ). The derivative is found by first fitting a line to all points within a sliding window of width [xwidth], if this is not specified it defaults to a few percent of the maximum x range (see first derivative function **D**). Next the derivative of this intermediate line is found, i.e.  $d/dx(dy/dx)$ . The final spacing between points is typically about twice xwidth. The results are saved as the line replacing any existing line unless the /ADD switch is included.

**DIR**

Disk directory.

**ECHO** text\_string

Prints text\_string on the screen.

**EDIT**[/LINE or /COL]

Edit data using the standard system editor. For the DOS version this is the DOS editor, for the Windows version this is Notepad, for the Linux version this is the vi editor. Data is written to and then read from the file SPDATA.TMP. Use /LINE to edit the line data, or /COL to edit the column data. Note that the column data is written out with a maximum of four numbers on a line. So if you have 6 columns of data, the first four numbers will be on line-1, the last two on line-2. The next record will start with four numbers on line-3,...

**ENTER**

Enter data into Splat! from the keyboard. Enter a '/' to exit.

**FFT [/INV] [var]**

Takes the Fast Fourier Transform of a set of equally spaced data points. Splat! will round down the number of data points to the nearest power of two, and find the magnitude of the Fourier components. The transform is applied to: the default variable if in form 1 (X), or form 2 (X,S); the Y variable if in form 3 (X,Y) or form 4 (X,Y,S); or these defaults can be overridden by supplying a variable name. If in X,Y or X,Y,S format, the X-axis is transformed to the proper frequency scale. The /INV option reverses the FFT out of the frequency domain. Unfortunately, this doesn't reproduce the initial function since phase information is lost in the forward FFT, as well as truncations in the number of data points.

**EXIT**

Exit program. You can also use QUIT, or hit Control-Z (End-Of-File).

**FIT[/options][n1:n2] function=n**

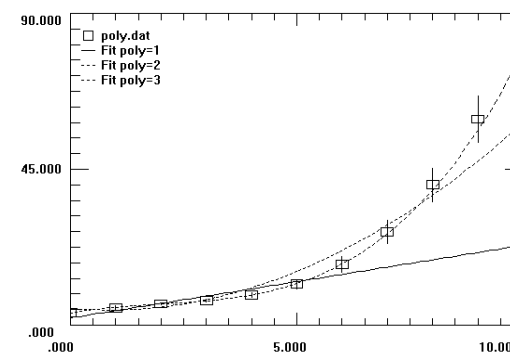
Options include:

/D	take line limits from data (instead of nice round numbers)
/ADD	append output line instead of replacing existing line
/NL	do not produce an output line
/SUB	subtract fit from data without producing an output line
/n1:n2	limit fit to selected range of data

Linear (or non-linear) least square fit a function to the data. Possible functions include:

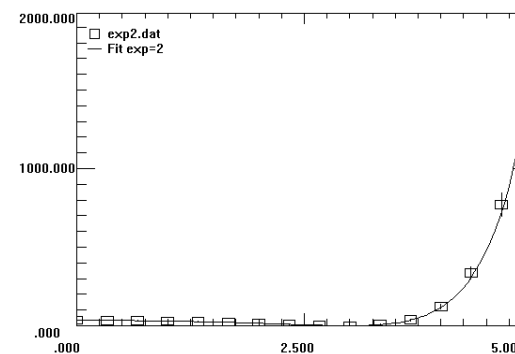
POLY = n      n<sup>th</sup> order polynomial      (linear fit)

$$y = \sum_{i=0}^n a_i x^i$$



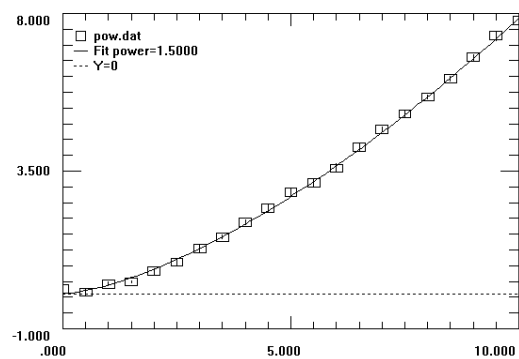
EXP = n      sum of exponential powers      (linear fit)

$$y = \sum_{i=0}^n a_i e^{i x}$$



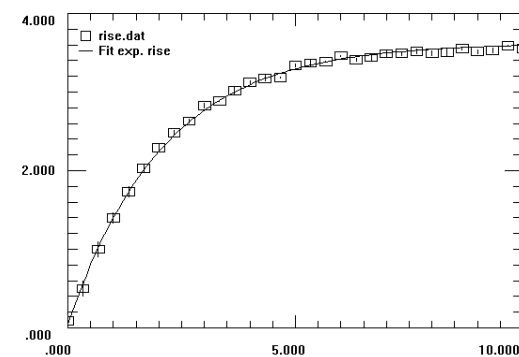
POW = p      n<sup>th</sup> power of x      (linear fit)

$$y = a_0 x^p$$



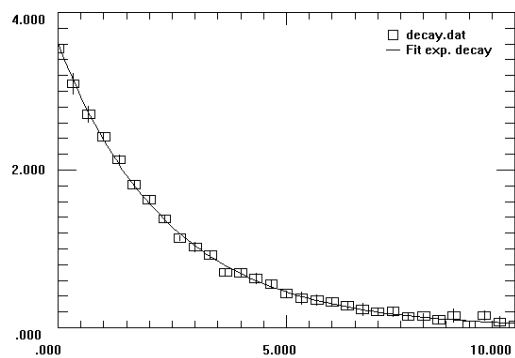
RISE      exponential rise      (non-linear fit)

$$y = a_0 \left( 1 - e^{-(x - a_1)/a_2} \right)$$



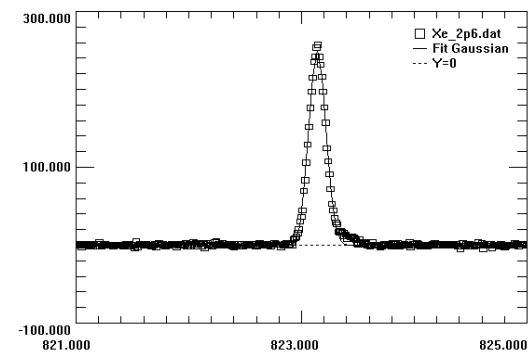
DECAY      single exponential decay      (linear fit)

$$y = a_0 e^{a_1 x}$$



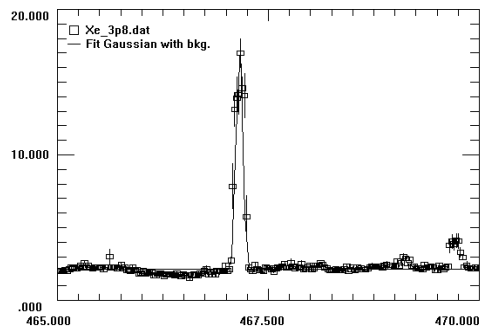
GAUSSIAN[=fwhm]      Gaussian distrib.      (non-linear fit)

$$y = a_0 e^{-(x - a_1)^2 / a_2^2}$$



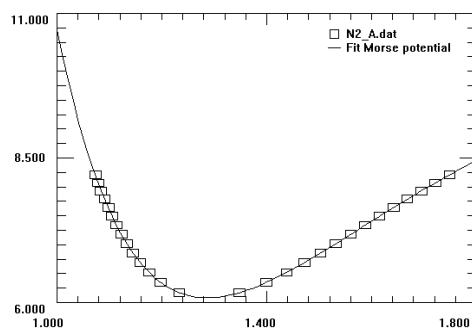
GAUSSb[=fwhm]      Gaussian dist. with (non-linear fit)  
constant background

$$y = a_0 e^{-(x - a_1)^2 / a_2^2} + a_3$$



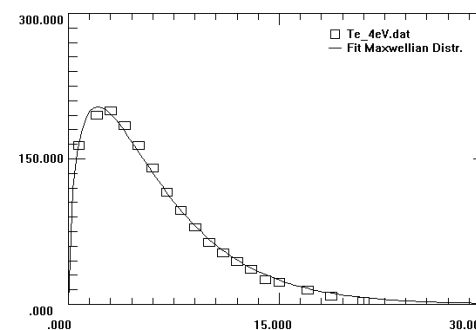
MORSE      Morse Potential      (non-linear fit)

$$y = a_0 \left[ 1 - e^{-a_1(x - a_2)} \right]^2 + a_3$$



MAXWELL      Maxwellian Distrib. (non-linear fit)

$$y = \frac{2}{\sqrt{\pi}} a_0 \sqrt{x} \frac{1}{a_1^{3/2}} e^{-x/a_1}$$



Two forms of output are produced. A text listing of the fitted parameters along with uncertainties; and a 100 point line of the fitted function. Use the /D option to limit the output line to the range  $x_{\min}$  to  $x_{\max}$  of the data. To suppress the generation of the output line, use the /NL option. To append the fitted line output to your existing line data use the /ADD option.

With the /SUB option specified, the fitted function is subtracted from the data and no line output is produced. By default, the fit is subtracted from all of the XY data, even if only a select range of the data (by using the /n1:n2 option) was used in the fit. To limit the subtraction to only the fitted range, use the /D option.

The Gaussian fits also output the FWHM of the curve, and the area of the Gaussian portion of the peak. By default, the width of the Gaussian peak is allowed to vary as a free parameter. If, instead, you specify the width (i.e. FIT GAUSS=1 . 1) this is held constant.



**FORM** format,[var]

Change the default data format and variable. In general, the format number tells how many data columns are being used, while the variable tells Splat! what the default choice is for operations such as sorting. Valid options for format are:

<u>format #</u>	<u>data format</u>
0	user defined
1	X
2	X,S
3	X,Y
4	X,Y,S
5	auto format
{6}	multi column

Valid choices for the variable are X, Y, and S. To change the variable with form, you must also provide the format. Examples:

FORM=4	change data format to form 4 (X,Y,S)
FORM 3	change to form 3 (X,Y)
FORM 2, Y	change to form 2 (X,S) with Y as default variable (hence this is really (Y,S) mode...

Note that changing forms does not delete/reset the data into the X,Y,S vectors. It simply controls how many are displayed, and whether error bars should be used for some operations (such as fitting or averaging).

The user defined format (0), auto format (5), and multi-column format (6) are special formats used to control how files are read into Splat!, see the **READ** command for more details.

**HELP**[/MATH] [command]

Help screen of Splat! commands. The /math option provides assistance on available math functions and variables. The help screen also lists the maximum number of data points your version of Splat! can support. The standard Help screen lists most basic commands and options. More detailed information for some commands, particularly those with lots of options like **AVERAGE**, **FIT**, **PLOT**..., is also available. For example, type `HELP FIT` for a list of options & parameters unique to the **FIT** command.

**HIST**

Histogram of data (See **BIN**)

**INCLUDE**

Include data from file (see **READ**).

**INT** [x1,x2]

Numerically integrates the Y data, using the trapezoidal rule. The x-spacing need not be constant, and can contain repeats (which are averaged together). With no arguments, it integrates the entire data range. If both x1 and x2 are included, the integration is only performed over the points nearest the interval specified. Splat! will inform you of the actual range of integration. Note that this command also sorts the data.

$$\text{Answer} = \int_{x1}^{x2} Y_i(x) dx$$

**LABEL**[/LINE][/N=num] text\_label

Used with **MARK** and **PLOT** to create a plot legend. With no options, the last marker set is labeled with text\_label. Or use the /N=num option to replace the label for marker num. The /LINE option is used to select the line labels. Use **MARK/****LIST** to view what label is associated with what marker number. By default, Splat! sets labels to input filenames, etc...

**LEGEND** [0]

In the *Windows version only*, opens a separate window with the plot legend. The window closes when you hit a key. **LEGEND** 0 creates a static legend window that must be closed manually.

**LINE**[/options] [filename]

Options include:

- /NO
- /CLEAR
- /ZERO
- /ADD
- /n1:n2

With no parameters or qualifiers, this function converts the X,Y data to a line and sets the number of points to zero. If you include a filename, the line is read in from the file instead. Use the /NO or /CLEAR qualifiers to clear an existing line, /ZERO to add a Y=0 line. By default the present line is erased; to append the new line to the old line use /ADD.

**LIST**[/options]

Options include:

- /LINE
- /COLUMN
- /FILE
- /MARKERS

/n1:n2

With no options, this command lists the data on screen. Use /LINE to see the line data, or /COL to see the column data. Use /FILE to see the filename used in the last read/write operation. The /MARK option lists the data markers and labels (see also **MARK/****LIST**). Same as the **SHOW** command.

**LOAD**

Load data from file (see **READ**).

**LS**

Disk directory.

**MAKE** nsize

Increases the number of data points by nsize. Use a negative number for nsize to eliminate points. See also the **SET** command.

**MARK** [/options] [n1, n2, n3...]

Options include:

- /NO
- /CLEAR
- /LINE
- /LIST
- /ADD

Marks are simply points in the data list where you wish to change plot symbols. It is a simple way of automatically appending /NSET1=n1, n2, n3... onto each **PLOT** command. Marks are entered either automatically by SPLAT! (if I\_MARKER=1 in the SPLAT.INI file), or by entering a list with the **MARK** command. Use the /NO or /CLEAR options to delete all marks, or use the /LIST option to view the currently set marks. If you want to add a new mark, without retyping in the whole list again, use the /ADD

command. Any command that can alter the order of the data points (**AVERAGE**, **EDIT**, **SORT**) will automatically clear all marks. Associated with each mark is a text label that can be altered with the **LABEL** command.

A similar list is also kept for the line data (for appending /NLINE=n1,n2,n3... onto each **PLOT** command). Use the /LINE option to enter/alter/clear these values.

**MATH**[/LINE][n1:n2][/J] expression

Manipulate data. The expression must be of the form var=..., where var is either X,Y, S.or C1 to C9. With the /LINE option, you can alter the X,Y line data (and C1 to C9). Note that you can alter a variable which is not in the current format (for example, if in form 3 (X,Y) you can still do **MATH** S= Y/10). The part on the right side of the equal sign can be a rather complicated expression involving constants (numbers,  $\pi$ ,  $e$ ), variables (X,Y,S,I,N,M, C1 to C9) and a variety of functions (sin, cos...). Note that starting with version 4.1, Splat! can now correctly handle most negative numbers, such as **SIN**(-X), whereas in the past this could cause serious problems (since the minus sign looked like subtraction).

With the /n1:n2 option the data is only modified for points between n1 and n2. By default, the index variable I is the absolute index value, i.e. **MATH**/4:7 Y=I will set Y=4 for point 4. Use the /J option for relative indexing, i.e. **MATH**/J/4:7 Y=I will set Y=1 for point 4.

#### variables:

X Y S I N M

-the X,Y,S vectors, index I, and the Number of data points

E PI RAN

-standard numerical values for  $e$  and  $\pi$ , a random number between 0 and 1

#### functions:

CHS

-convert to negative number

+ - \* / ( ) INV ^ \*\* Sqrt

-algebraic functions

ABS INT NINT

-more algebraic functions

MIN MAX

-Min/Max of two arguments

! FACT

-factorial / natural log of gamma function

SIN COS TAN ASIN ACOS ATAN

-trigonometrical functions (radians)

SIND COSD TAND ASIND ACOSD ATAND

-trigonometrical functions (degrees)

EXP LN SINH COSH TANH

-natural log & hyperbolic functions

LOG ALOG

-base-10 log

DUP DROP SWAP CLEAR

-stack commands (**CALC** mode only)

Some examples:

MATH X=X^2	$x_i \rightarrow x_i^2$
MATH Y=X-4	$y_i = x_i - 4$
MATH Y=(C2-C3)/C4	$y_i = (c_2^i - c_3^i) / c_4^i$
MATH X=3*I-0.5+RAN	$x_i = 3 i \pm 0.5$
MATH Y=SIN X	$y_i = \sin(x_i)$
MATH Y=EXP ( CHS X)	$y_i = e^{-x_i}$
MATH/LINE Y=Y+4*X	$y_i^{line} = y_i^{line} + 4x_i^{line}$

**NORM[/LINE][n1:n2] [I=pt\_num] [var=val]**

With no options, this sets the peak of the Y data to one. Optionally a new value other than one can be supplied. By including a variable with the optional value, a different variable can be normalized. To choose the normalization point to be a value different from the peak, use the I=pt\_num option. The same transformation is performed to the S data if in form 2 or form 4. With the /LINE option, the line data is normalized.

Examples:

NORM	set peak Y value to 1
NORM 12	set peak Y value to 12
NORM Y=12	set peak Y value to 12
NORM I=30 Y=12	sets Y value for point num 30 to 12

**OUT**

Same as **WRITE**.

**POINT[/R]**

Adds the existing line data to the regular x,y data points. This does not clear the line. With the /R option the x,y data is replaced with the line data.

**PORT[/C,/X] x1,x2**

Transfers data, for a selected range, from the column data set to the points (X,Y,S) data set. Only points with C1 values in the range  $x1 \leq C1 \leq x2$  are transferred. The number of columns transferred is determined by the format: in form=3, C1→X, C2→Y; in form=4, C1→X, C2→Y, C3→S. By default, the points are appended to the existing (X,Y,S) data set, to replace the existing data use the /C option. The /X option replaces the existing data (as with /C) and deletes any line data. See also the **ADD** command.

**PLOT[/options]**

Options include:

/DATA	force x&y plot limits to min/max of actual data
/n1:n2	plot only points n1 to n2 (no expressions allowed)
/NSET1=n1,n2,...	plot first n1 points as squares, then all points up to n2 as 'x's, etc ...
/NLINE=n1,n2,...	starts drawing a new line after n1, n2...
/XLOG	plot x-axis on a log scale
/YLOG	plot y-axis on a log scale
/LEG[=position]	draw a plot legend in upper left corner (or other position)
/TEXT	text mode output (from the early days of Splat!)

/PRINT	produces HPGL output file (see below)
/OVERWRITE	overwrite existing HPGL output file (for use with /PRINT)
/MODE=video_mode	change the graphics mode (see below)

A simple 2D plot of all data (points and lines). Plot will normally choose appropriate x&y data ranges (from both the line and point data), and plots the data points as squares. If there are a large number of data points, the size of each symbol is automatically reduced. The /DATA switch forces the plot limits to the actual data limits instead of using round numbers. To plot data on a base-ten log scale, use the /XLOG and/or /YLOG options— note that only positive values are allowed for a log scale. The /DATA switch is ignored on any axis plotted on a log scale.

Different sets of data can be plotted with separate symbols using the /NSET1 and /NLINE options. With the /NSET1=n option, only the first n points are plotted as squares, remaining points as 'X's, etc... To have all your data points be 'X's, use /NSET1=0. With the I\_MARKER option in the SPLAT.INI file (see section 2.2), Splat! can be configured to automatically append /NSET1 and /NLINE options onto each plot command.

To include a legend in the upper left corner of the plot, use the /LEGEND option. If this overwrites some data points, you can select another location by using /LEG=*position* instead, where *position*=1 is the upper left corner, 2 is the upper right corner, 3 is the lower left corner, and 4 is the lower right corner.

The /TEXT, /PRINT and /MODE qualifiers select what form of output device is used. /TEXT uses the original SPLAT! version 1.0 character cell plotting routine. Note that this text output routine does not support many of the more advanced PLOT options.

System specific notes: DOS version

/MODE is only applicable to the DOS/PC version of SPLAT! It selects one of the standard VGA/SVGA graphics modes:

<u>Mode</u>	<u>screen size</u>
/M=VGA	640 × 480
/M=640	640 × 480
/M=800	800 × 600
/M=SVGA	1024 × 768
/M=1024	1024 × 768
/M=1280	1280 × 1024
/M=AUTO	best CGA, EGA or VGA available

/PRINT produces a HPGL output file called SPLATn.HPG (n=1,2,3...), and then issues the command HPSPLAT SPLATn.HPG. Typically, this is a HPGL file viewer. For actual hardcopy output, this file can be imported into a 'real' program such as most popular graphics and word processing programs. With the /OVERWRITE option SPLAT! overwrites any preexisting SPLAT1.HPG file.

System specific notes: LINUX version

/PRINT produces a HPGL output file called SPLATn.HPG, and then issues the command:

```
HP_PRINT1 SPLATn.HPG HP_PRINT2
```

where the HP\_PRINTns are defined in the .splat.ini file. Typically, this is a HPGL file viewer, such as hp2xx. Under Linux, the HPGL file output is the default output mode as well, but Splat! issue the command HP\_PLOT SPLAT1.HPG to view the output. In addition, in plot (versus

print mode) the /OVERWRITE option is used to prevent your working directory from filling up with SPLATn.HPG files. See section 2.2 for more details on editing the .splat.ini file.

### System specific notes: WINDOWS version

Two output modes can be used in the Windows version of Splat!. A standard HPGL graphics file can be created with the /PRINT option. For actual hardcopy output, this file (called SPLATn.HPG, with n as a running index) can be easily imported into most popular vector-based graphics programs and many word processing programs. Alternatively, you can use the Windows <alt>-<prt screen> to copy the current Splat! window to the clipboard. This can then be pasted into Paint as a standard Windows Bitmap file.

You can use the TAB key to switch between the plot window and the command console being in focus. The LEGEND command is used to create a separate plot legend in a new window, instead of placing the legend in the plot.

## QUIT

Exit program.

## READ filename

Read in data set from a file. The file can include blank lines. Lines that begin with non-numeric characters are skipped. If you enter '?' for the filename, a directory listing is provided before the further prompting for a filename. Splat! also supports wildcards ( \* and ?) in the filename.

If in form=0 (user defined format), a format statement is also required. The format statement is of the type /'0,0,X,Y'. Where the format statement consists of a line telling which column in the data file to interpret as each

variable (X,Y,S), and which column entries to ignore (0). The entire statement must be enclosed in single quotation marks. In the preceding example, the numbers in column three will be read into the X vector, and the numbers in column four will be read into the Y vector. Note that you are still limited to only three total values per line that can be read in (X,Y,S). Splat! will switch to the appropriate format after it has read in the data. Some examples of the user defined format (assuming you start in Form=0),

READ/' 0, 0, 0, Y, X' filename	Reads the 5 <sup>th</sup> column as X, 4 <sup>th</sup> col. as Y
READ/' 000YX' filename	same as above
READ/' , , , Y, X' filename	same as above
READ/0/' 0, S, Y, X' filename	Reads the 2 <sup>nd</sup> col as S, 4 <sup>th</sup> col as X, 3 <sup>rd</sup> col as Y

In form=5 (auto format), Splat! can read in up to nine columns of numbers. Splat! will find the first row of numbers in the first file it opens, and read in that number of columns for each data point. This data is stored in the C1 to C9 data array. The number of points in the C1 to C9 array is called M, while the number of data columns used is called C. If you have no existing X,Y,S data when you read in with auto format, then Splat! copies as many columns as possible (1 to 3) into X,Y,S vectors, and sets the format to the appropriate value. For example, suppose you have a file with seven columns of data called lots.dat. The command READ/5 lots.dat will cause Splat! to read the seven columns of data into C1 through C7. Then Splat! will copy C1 into X, C2 into Y and C3 into S. Finally, Splat! will change the format to form 4 (X,Y,S format).

Form=6 (multi column) is a user unfriendly way to read data into the column data. To use form 6, you must first

set the number of columns to read with **SET C=n**. All *n* 'columns' need not be on the same line. Splat! will start reading the next line to fill in all the columns.

### **SET[/LINE] N=expression**

Set the number of data points (N) using some expression. With the /LINE option, you can alter the number of line segments. Some examples:

```
SET N=12      set the number of data points to 12
               (regardless of current N)
SET N=N+6     add 6 data points (same as MAKE 6)
SET N=N/2     cut the number of existing data points in half
SET N=N*2     double the number of existing data points.
SET M=8       set the number of column data points to 8
SET/LINE N=N/2 cut the number of existing line
               segments in half
```

The expression is evaluated the same as with **CALC** and **MATH**, so you can use all the wacky functions you want. The finally result is rounded to the nearest integer.

**SET** is also used to control the column data variables, M and C. M is the number of data points stored in the column array. C is the number of columns that contain data. The value M can be used in any expression (i.e. with **SET**, **MATH**, the range option /n1:n2), but C can not.

### **SHOW**

List data on screen. (Same as **LIST**)

### **SMOOTH[/G or /M] window**

Smooths the data without changing the number of data points. There are two smoothing functions. The default choice (/G) is to convolute the data with a Gaussian with

FWHM=window. The other choice (/M) is to use a rolling median of width equal to window. While smooth does not change the number of data points, it does sort the data.

### **SORT[/D] [variable]**

Sorts the data in ascending order either using the default variable, or the optional parameter. Use the /D qualifier to sort in descending order.

### **STAT[/n1:n2] [variable]**

Statistics (average, mean, min, max...) on a single variable. If in form 2 or form 4, the S vector is used to find the weighted mean.

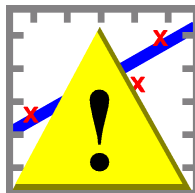
### **WRITE filename**

Writes data out to a file (standard ASCII text file). If the I\_CLOBBER value in the .INI file is set to one, the program will check to see if you are overwriting a file. If a file exists, it allows you to abort the write.

### **YO**

Hands out a worthless nugget of wisdom. Type YOYO for twice the fun, or YO! for even more 'advice'.

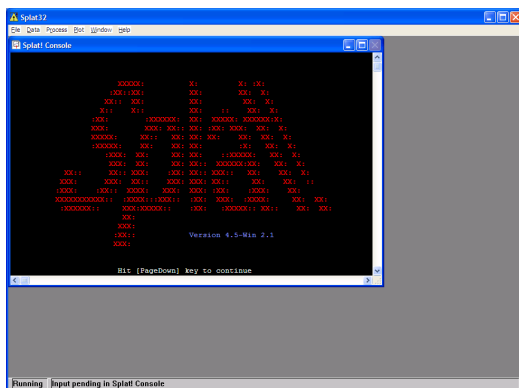
## 4 Installation & Setup



Splat! comes in three varieties: a 16-bit DOS version, a 32-bit Windows version, and a Linux version. The DOS version works fine with the various Windows operating systems, it just runs in its own DOS/command prompt process. Unfortunately there is no handy installation utility for Splat!, so it requires a bit of work to set it up properly.

The following three subsections describe how to set up each different version of Splat! with different operating systems. The last two sections (4.4 and 4.5) describe how to describe to configure how Splat operate (common to all versions) using the splat.ini initialization/configuration file.

### 4.1 Windows (32bit) version



Windows XP / 2000 / 95 / 98 / Vista?

#### 4.1.1 Installation

The Windows version of Splat! only 'needs' two files:

SPLAT.EXE

SPLAT.INI

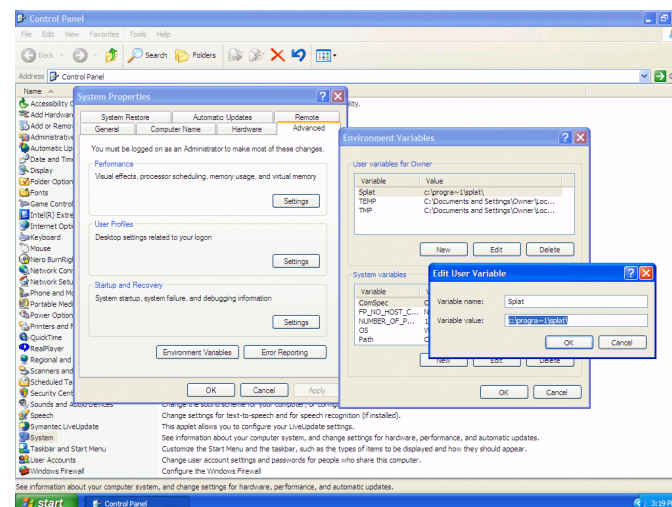
Copy these two files to a directory such as

C:\Program Files\Splat or C:\SPLAT.

The executable should be in a directory listed in your path statement. The .INI file is optional, but is used to set some common user options. To use the .INI file you must also define a SPLAT environment variable.

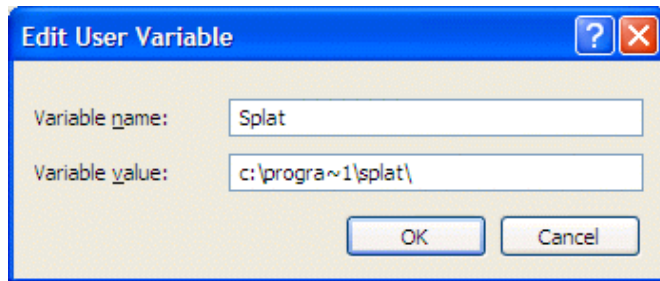
#### Windows XP/2000/Vista

You add the path and environmental variable via the System control panel. Click on the "Advanced" tab, then press the "Environment Variables..." button.

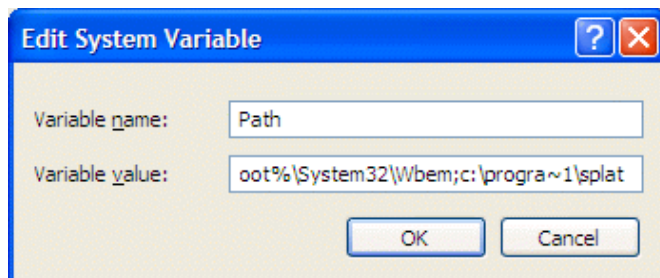




Let us assume you have put the two files in C:\Program Files\Splat , in the top User Variables part hit the 'New..' button. , type in SPLAT for the name of the new variable and set the value to the Splat! directory. Note that this variable should end with a "\". It is also required to use the short 8 character path name for the directory, thus for example the value is C:\Progra~1\Splat\.



It is also advisable (but not required) to add Splat! to your path. In the System Variables section (the lower portion of the Environmental Variables window), select Path and click the 'Edit...' button. Append the Splat! directory to the existing Path variable. For the path value, do not include the trailing '\'.



### **Windows 95/98**

To set environmental variables in Windows 9x, you edit the AUTOEXEC.BAT file. For example, if Splat! is in the C:\SPLAT directory you would have something like:

```
PATH=C:\DOS;C:\WINDOWS;C:\SPLAT
```

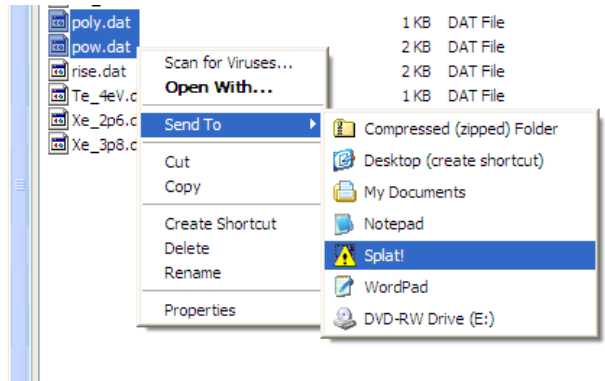
```
SET SPLAT=C:\SPLAT\
```

Note the extra ' ' in the SPLAT variable versus the PATH variable.

### **4.1.2 Running the Program**

Splat! can be run just like any other windows program, i.e.: (i) enter Splat in the Run... box, (ii) click on the Splat! program icon, or (iii) create a shortcut to the program (by right clicking on the program and selecting create program shortcut), and put a copy on your desktop, or in your Start menu, whatever you want. You can Drag and drop files onto the program icon. While the windows version of Splat! supports long filenames in general, command line arguments (for some reason) get converted to DOS compatible 8.3 style filenames. This applies to drag and drop files. If you create a shortcut, it may be worthwhile to edit its properties and change the 'Start In' directory to something short and sweet like C: or C:\temp.

One of the most useful ways to run Splat! is to place a Shortcut to Splat! in the Send To folder. In Windows XP (etc) this folder is located somewhere like C:\Documents and Settings\User\Send To. In Windows 95/98 this folder is located in C:\Windows\Send To. You can then send files directly to Splat! by right clicking on them in Explorer.



### 4.1.3 Issues/Bugs

Splat! was written as a command line program in the days of stones knives and bear skins. As a result, trying to run Splat! as a pull-down, menu-driven, Windows program is going to have some problems. Here are a few:

#### (1) Long Filenames

Splat! can read in long filenames of up to 70 characters (including the path). But Splat! can only save files in the “old-school” 8.3 character format. Note that the 70 character limit can cause serious problems with long Windows-XP directories of the form “C:\DOCUMENTS AND SETTINGS\OWNER\MY DOCUMENTS\...” which eats up around 50 of the 70 possible characters. This difficulty can be easily overcome by using directories with short names.

#### (2) Menu/Keyboard problems

The pull down menus in the Windows version look nice, but on some Win-XP systems some menu items don't work (most importantly Open...). Another

problem on some systems is that accessing the menus causes the keyboard to lock up. General advice is not to use the menus, you don't really need them any ways.

#### (3) Plot Window problems

When the color depth is too high (millions of colors / high/ 32 bits), the windows Splat! interface may not read the display resolution settings correctly, so an empty plot window appears. To fix this, you can either right click on SPLAT.EXE, then select Properties, under the compatibility tab check the "Run in 256 color" mode; or you can lower the Color Quality of your display to (medium / 16 bit / thousands of colors). The first option causes weird pallet re-mapping effects, the second lowers the color quality of your display for all programs. I can't tell too big a difference between the medium and high color depths, but you might...

#### (4) HandleTEMPORARY files

On Windows 9x systems, HandleTEMPORARY files get left behind when Splat! closes. An annoyance, but not a real problem.

## 4.2 DOS (16bit) version

```

Loaded macro command: @pc
Loaded macro command: @ion

Welcome to SPLAT!
Version 4.5 -PC 1.1

format:  auto form
variable: 8

reading data from ***File_List***
file : C:\TEMP\SOFT.TXT
file : C:\TEMP\HARD.TXT
format:  <X,Y>
Npts = 38
>_

```

DOS / Windows 3.1, 95/95/Me/2000XP /...

The DOS version of Splat! might be more properly called Splat! in 'full screen' mode, it works just fine under Windows as a full screen application. As a result, instructions are included for running this version of Splat! in Windows XP (etc...) as well as with legacy DOS computers.

### 4.2.1 Installation

The DOS version of Splat! requires the following files:

```

SPLAT.EXE
DOSXMSF.EXE or DOSXNT.EXE
HPSPLAT.BAT or HPSPLAT.EXE
SPLAT.INI

```

All the executables should be located in a directory listed in your path statement. The HPSPLAT files are not required unless you want to automatically process HPGL plot output. The .INI file is also optional, but is used to set some common user options and is highly recommended. To use the .INI file you must also create a SPLAT environment variable. For DOS and early versions of Windows (3.1,95, 98) this is done with settings in the AUTOEXEC.BAT file. For example, if

Splat! is in the C:\SPLAT directory you would have something like:

```

PATH=C:\DOS;C:\WINDOWS;C:\SPLAT
SET SPLAT=C:\SPLAT\

```

Note the extra '\' required at the end of the SPLAT variable versus the PATH value. In later versions of Windows (XP / 2000 / ...) the path and environment variable are setup differently. For these OS follow the directions under the Windows Splat! installation (section 4.1.1)<sup>1</sup>.

### Configuring the Plot mode

Before using Splat! on a regular basis, it is worthwhile to optimize the screen resolution used for plotting. The default plot mode for Splat! is a very low resolution mode, a much better resolution setting may work better with your computer. To find out, create some sample data in Splat! (or read in the sample.dat file included in the distribution file). Now try out various plot modes (see the **PLOT** entry in section 3 for all possible options); i.e. PLOT, PLOT/m=640, PLOT/m=1024,... 'Bad' plot modes manifest themselves in many possible ways: you may see nothing<sup>2</sup>, the screen may freak out<sup>2</sup>, only the text axis labels are displayed, only the data is displayed (with no labels), etc...

Once you find a 'good' plot mode edit the SPLAT.INI file to use this choice as the default plot mode. The first few lines of the SPLAT.INI file should look something like this,

---

<sup>1</sup>But note that for the DOS version of Splat! you must add Splat! to your Path.

<sup>2</sup>Hitting <return> should fix things up by returning the display to text mode.

&DEFAULTS

PLOT\_MODE\_DEF = 'M=SVGA'

:

Simply replace the 'M=SVGA' in the above example with what works best for your computer.

#### 4.2.2 Running the Program

While in DOS the only way to start Splat! is with the command line, there are a number of ways to start up DOS Splat! under Windows as described in Sec. 4.1.2. Before turning to the later, let us briefly examine some of the options of the former.

Splat! allows a few possible command line arguments. In particular, you can set the default form and variable, and specify a filename to read in. For example,

```
SPLAT/FORM=3/VAR=Y STUFF.DAT
```

This starts Splat! up in data format 3 (X,Y), with Y as the default variable, and immediately reads in file STUFF.DAT.

There is little reason to pre-specify the format since Splat!'s auto-format capability should handle this for you. The filename could include wildcards (\*,?), or could be replaced by a list of files separated by spaces. For example,

```
SPLAT SOME.DAT MORE.DAT
```

```
or SPLAT ARGON_*.DAT
```

To use the DOS version of Splat! with Windows, we first want to create a shortcut to the program. From Explorer right click on the Splat! shortcut and chose properties. Under the Screen Tab, select Full Screen mode. Under the program tab, check the 'close on exit' box. You may want to change the 'Starts in' directory to something simple. It, is also nice to change the Splat! icon. Click the 'change icon...' button, and browse to the Splat! directory. Select the Splat icon, or find

your own icon for Splat!. You should now have a Windows shortcut to Splat!; put a copy on your desktop, in the Send To folder, or in your Start menu, whatever you want (see Sec. 4.1.2 for more detailed instructions).

Note that the DOS version of Splat! uses the DOS 8.3 style filename.ext form for filenames, and generally can not read in long filenames. However, by either dragging & dropping files onto the Splat! program icon, or by using the Send To method, you can read in long filenames, but you are still limited to 8.3 filenames on output. Send To can also be used to select multiple files at once to be read into Splat!, but the number of files is limited by the length of the full path names.

#### 4.2.3 Simultaneous Use of DOS Splat and Windows Splat

Can't make up your mind? Do you want to run both the 16bit and 32bit versions of Splat! on the same computer? No problem. Set up the DOS version of Splat as described here. Rename the Windows Splat! program something like WinSplat.exe or Splat32.exe and copy it into the Splat! directory. You could also rename the DOS version, but you are more likely to access the DOS version directly from the command line where the program name makes a difference versus a Windows shortcut that can be labeled whatever you like. Both versions can use the same .INI configuration file.

### 4.3 Linux version

#### 4.3.1 Installation

The Linux version of Splat! uses the following files:

splat	Splat! program file
hp2xx	HPGL file viewer program (Optional)
~.splat.ini	default program values (Optional)

All the executables should be in a directory listed in your path statement (such as /usr/bin/). The .splat.ini file is optional, but is used to set some common user options. It is generally a hidden file in your home directory. Typical values for the files contents are:

```
&DEFAULTS
PLOT_MODE_DEF = ' '
IFORM_DEF     = 5
IVAR_DEF      = 1
I_MARKER      = 1
I_CLOBBER     = 1
HP_PLOT       = 'hp2xx -q'
HP_PRINT1     = 'hp2xx -q'
HP_PRINT2     = '; ls'
```

The default plot method for the Linux version of Splat! is to create an HPGL output file and then hope the user has some way of displaying it, i.e. hp2xx (which you need to supply). Check the hp2xx documentation for further help on that program's parameters and qualifiers. Splat!'s printer output also relies on hp2xx. The output HPGL filename is sandwiched between the user defined HP\_PRINTn commands. For example, the following settings produce output on a LaserJet printer:

```
HP_PRINT1 = 'hp2xx -q -c11111 -m pcl -i -F -f- -O50'
HP_PRINT2 = ' | lpr'
```

For these settings, a PLOT/PRINT command corresponds to a Linux command line of

```
hp2xx -q -c11111 -m pcl -i -F -f- -O50
SPLAT1.HPG | lpr
```

Alternatively, the only other plot mode under Linux is the old text-based system (see example p.21). To set this as the default plot mode set PLOT\_MODE\_DEF = 'TEXT'.

### 4.3.2 Running the Program

Splat! allows a few possible command line arguments. In particular, you can set the default form and variable, and specify a filename to read in. For example,

```
splat -form=3 -var=Y Stuff.dat
```

This starts Splat! up in data format 3 (X,Y), with Y as the default variable, and immediately reads in file Stuff.dat. The filename could include wildcards (\*,?), or could be replaced by a list of files separated by spaces. For example,

```
splat -form=3 some.dat more.data
```

## 4.4 SPLAT.INI Configuration File

The SPLAT.INI file contains two things, (1) the default start-up mode information for Splat! and (2) macro definitions. The macro functions are described in section 4.5. As for the default start-up mode stuff, you do not absolutely need an INI file since Splat! has up to three sets of default parameters for most options. These sources are (in order of precedence):

- 1) Splat!'s own internal defaults
- 2) SPLAT.INI values
- 3) Values supplied on the command line (i.e. SPLAT/FORM=3)
- 4) SPLAT! Command qualifiers (i.e. PLOT/M=VGA, or FORM 4)

Nonetheless, the .INI file is the only way to turn on/off the marker function (for automatic generation/management of marks and labels), the control of file overwrite protection, and define macros to change the flags for automatic data marking and for.

The exact contents of the SPLAT.INI file depend upon what operating system you are using. The SPLAT.INI file distributed with your particular version of Splat! contains lists of all valid values for the setting for your system.

For the DOS/Windows version of Splat!, typical SPLAT.INI contents and meanings are:

```
&DEFAULTS      ←Required header
PLOT_MODE_DEF = '/M=SVGA'  ←Sets the default plot
                             mode, see PLOT
IFORM_DEF = 3      ←Sets the default data format see
                     FORM entry in section 3
IVAR_DEF  = 1      ←Sets the default variable see FORM
I_MARKER  = 1      ←Set to 1 to automatically plot each
                     data file read in with a new symbol,
                     0 for same symbol
I_CLOBBER  = 1      ←Set to 1 to check for overwriting
                     output files, 0 to allow overwriting
                     without checking
```

The PLOT\_MODE\_DEF option is appended onto every **PLOT** command. It is typically used to set the display resolution for the DOS version of Splat!. However, any valid **PLOT** options can be specified. For example, if you always want a plot legend, you can set it to '/LEG/M-SVGA'.

## 4.5 Macro Commands

Up to ten command macros can be defined in the SPLAT.INI file. Each macro can be composed of up to 15 Splat!

commands<sup>3</sup>. A macro command is accessed in Splat! by typing @macro\_name [parameter] at the Splat! prompt. All macro names must start with a '@'. One optional parameter is allowed. The macro definition block in the SPLAT.INI file is of the following form:

```
@macro_name      the macro name
3                number of SPLAT!
                  statements in the macro
                  SPLAT! commands...
```

```
statement 1
statement 2
statement 3
```

Anytime a '%1%' appears in the macro statements the user supplied parameter is substituted (if provided). For example, suppose you want a macro that creates a ten point (X,Y) data set for a user supplied function on the interval from 0 to 1. In the SPLAT.INI this would be defined as

```
&MACRO
@makeit
5
form 3
set n=10
label user function: %1%
math x=(i-1)/9
math y=%1%
/
```

To use this to create values for sin(x) one would enter @makeit sin(x) at the Splat! prompt. Macro calls can be nested (one macro invokes another macro), but not recursive.

---

<sup>3</sup>15 is more a recommendation than a firm limit. Up to 25 commands *may* be allowed.