

Generating Moderator Source Tables for NISP

P. A. Seeger, <http://PASeeger.com>, <mailto:PASeeger@losalamos.com>

August 5, 2001

Revised November 23, 2004

(report written: December 23, 2005)

Introduction

To make meaningful Monte-Carlo simulations for comparison of neutron instruments it is necessary to consider both intensity and resolution. For pulsed sources the resolution is generally dependent on the pulse shape; some instruments depend critically on details of the shape, beyond simple statistical parameters. Shapes for short and long wavelengths are fundamentally different. The best estimators we have for the pulse shapes are the detailed simulations made using the MCNP code that treats nuclear interactions microscopically. This report gives a procedure for converting the output data from MCNP into a form that is efficient for use in instrument simulations. The energy spectrum (properly normalized) is taken from a “point detector” output from MCNP, while the pulse shapes at a number of energies come from the equivalent of a two-dimensional E - t tally at the moderator surface. The procedure fits cubic spline curves between 33 nodes in the cumulative distributions. Since the reverse interpolation of the splines results in a scalloped appearance, the procedure has been modified to use splines only in the internal nodes, and to use decaying power laws in the lowest and highest 0.1% of each distribution.

Please be aware that the procedure in this report is a compromise between (a) making tables quick and efficient to use, and (b) precision of reproducing the MCNP “data” over its full range. If necessary for specific cases, the precision can be improved by limiting the dynamic range. There is *no* measure of accuracy; that is at the mercy on the MCNP simulations used as input.

I. Raw Data

MCNP “raw data” are available for four spallation neutron sources, from the following authors:

IPNS, Erik Iverson
ISIS, Stuart Ansell
Lujan Center, Guenter Muhrer
SNS, Erik Iverson

The example used in this report is from SNS. The current files (HPTS-sct, 7/31/02) are found at

http://www.sns.gov/users/instrument_systems/components/moderator/HPTS-sct.html

There are six files corresponding to the various moderator faces. Each file contains 1-D detector energy data, and a 2-D surface tally binned in energy and time. An essential feature of the data files is that both energy and time are binned logarithmically, and that in the E - t tally the number of time bins per decade is a multiple of 2 times the number of energy bins per decade; this latter requirement facilitates the conversion of bins from t to t/λ .

A. Edit the data files

The “standard” format includes many empty cells, and much of the high-energy data is not of interest in moderated spectra. The file needs to be cropped before use. Some of the header information must be made available in a fixed format at the beginning of the file. As an example, consider the file for the 25-mm poisoned water moderator (flight path 17, bottom-upstream moderator)

source_sct521_bu_17_1.dat

The text data at the beginning of the file is used to write 7 lines in the output file:

```
SNS-sct521_bu_17, 25-mm H2O  
EBI_020327_175548
```

```

0.034, 120.    /pulse energy (MJ), area (cm^2)
141           /points in energy spectrum
101, 90       /dimensions of E-t tally
10, 4         /first E to keep, # to bin together
(sct521_bu_17,25H2O)

```

- The first line is a 40-character descriptive title.
- The second line is a 17-character identifier, including the author's initials and the date and time that the file was generated, yymmdd_hhmmss.
- The third line has two normalization factors found in the text of the header:
- The fourth inserted line has the number of energy bins after deleting nulls, and the fifth line gives the number of energy bands and time bins retained in the 2-D tally.
- The sixth line allows energy bins of the E - t tallies to be combined to a maximum of 23 for the fitting procedure, to improve statistics.
- The seventh line has a blank in column 1 and a short identifier (enclosed in parentheses, no embedded blanks). This identifier will be associated with all files derived from this one; in combination with the date/time stamp on line 2, this will assure that versions are tracked.

A blank line must be inserted before the first energy data, and before each block of the E-t array.

B. Execute program `ARRANGE`

It is important to pay very close attention to the units of the data. The column headed "Flux $f(E)$ " is the number of recorded neutrons in the energy bin, divided by the energy difference; the units are $n/\text{ster}/\text{pulse}/\text{eV}$. The data in the time-energy array are also divided by the width of the time slice, so the units are $n/\text{ster}/\text{pulse}/\text{eV}/\mu\text{s}$. Also, the given values of E (eV) and t (μs) are at the geometric means of the histogram bin boundaries.

A version of the program `ARRANGE` specific to the SNS data is listed in Appendix A. It reads the edited MCNP file and creates three output files for plotting and for input into SigmaPlot[®]. The first file contains three data sets in the Block ASCII format used by NISP and associated analysis programs.

- 1-D histogram vs. logarithmic E (meV), with units $n / (\text{MW-s}) / \text{mm}^2 / \mu\text{ster} / \ln(E)$
- 2-D histogram vs. logarithmic E (meV) and t (μs), with units $n / \mu\text{s}$
- 2-D histogram vs. logarithmic t (μs) and λ (\AA), with units n per bin (*i.e.*, raw counts)

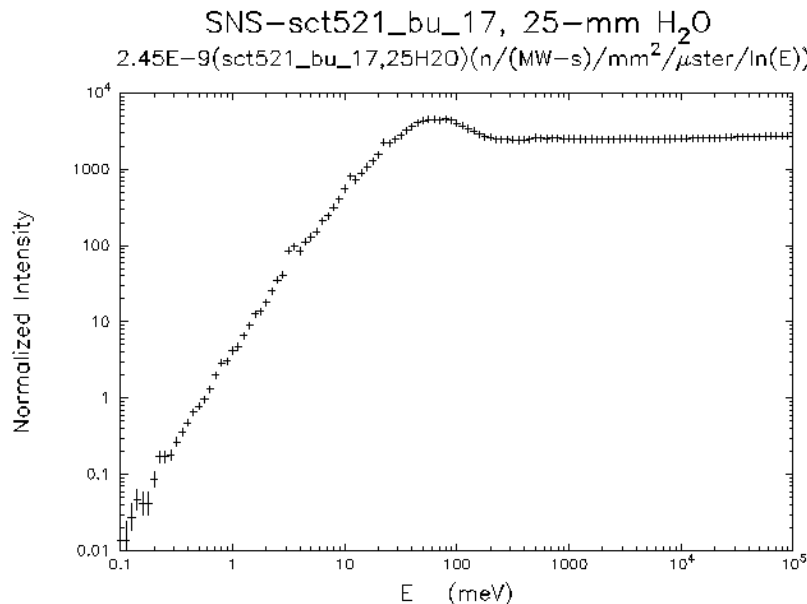


Figure 1. Energy spectrum from the point detector for the bottom-upstream 25-mm poisoned water moderator on the SNS High-Power Target Station.

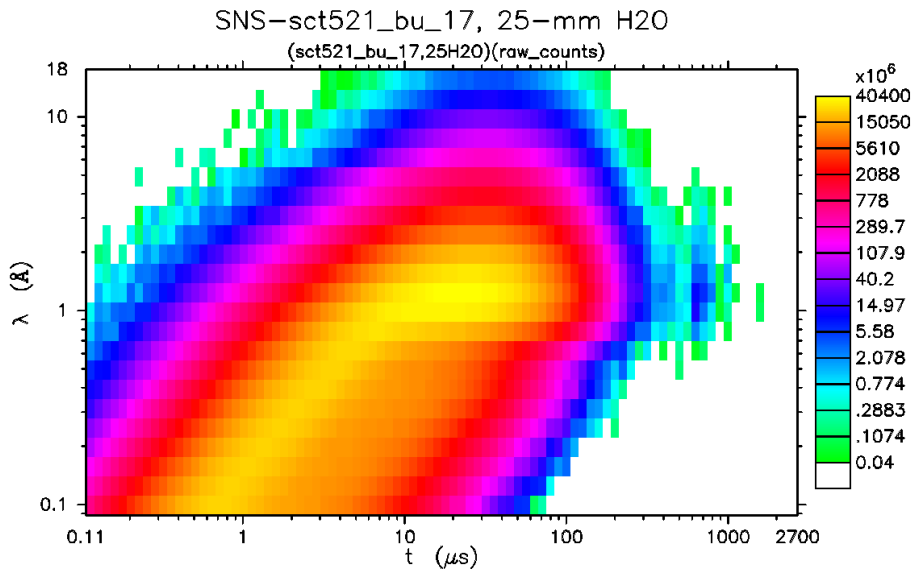


Figure 2. “Raw data” from the surface tally for the bottom-upstream 25-mm poisoned water moderator on the SNS High-Power Target Station. There are 23 wavelength bands. Data beyond about 500 μs are spurious.

These “raw” data may be plotted using the program `See_MC_Data`, as shown in Figs. 1 and 2 for the energy spectrum and for the 2-D t/λ histogram. In addition to statistical fluctuations, both data sets contain artifacts which must be omitted or smoothed over in the fitting procedure. A λ -weighted distribution will be used for energy, to increase the sampling at long wavelengths, and to provide a function that is decreasing at both limits. Neutrons sampled from the weighted distribution are given statistical weights proportional to $1/\lambda$ to compensate.

In the 2-D plot, the color intensity scale is logarithmic. At short wavelengths the pattern is consistent along a line of slope 1, implying a t/λ dependence of the pulse shape. At the longest wavelengths, the shape is roughly constant in the vertical direction, which implies a t dependence. There are clearly some bad data in the SNS file; all times $>500 \mu\text{s}$ must be deleted from the analysis.

The other two output files from `ARRANGE`, with extensions “.cols” and “.sigplt”, are formatted to be imported directly into spreadsheets for further analysis.

II. Energy Spectrum

The analysis is based on SigmaPlot[®], but presumably could be adapted to a spreadsheet application. Point-detector data are imported from the .col file to cell(3,1) as follows:

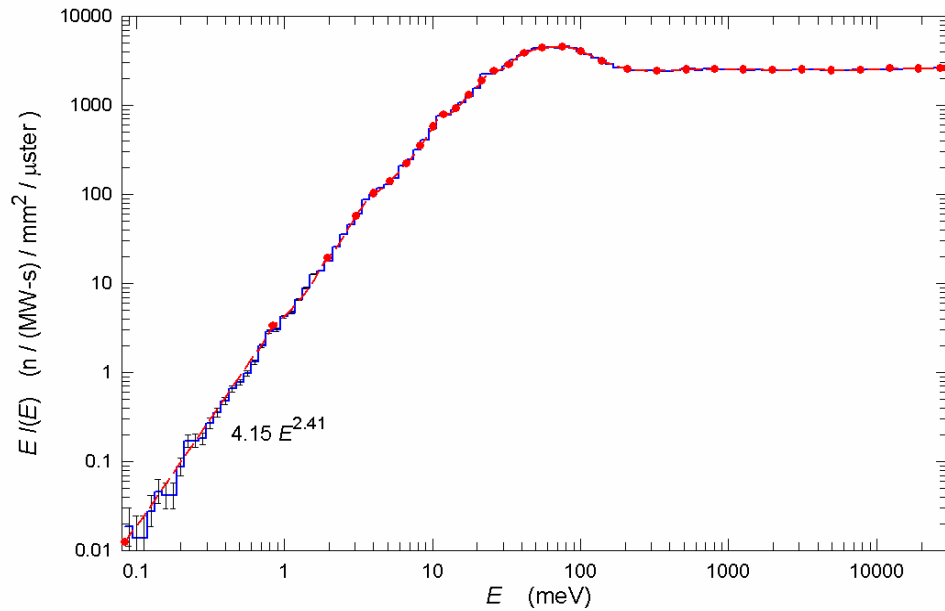
```
col(3) = E of bin centers (meV)
col(4) = data,  $E I(E)$  ( $n/(\text{MW-s})/\text{mm}^2/\mu\text{ster}/\ln(E)$ )
col(5) = standard deviation ( $n/(\text{MW-s})/\text{mm}^2/\mu\text{ster}/\ln(E)$ )
```

Running transform **Energy0** (see Appendix B) will generate columns for bin boundaries, the standard deviation of $\log_{10}(\text{data})$, and the bin width in lethargy units. The histogram will be plotted as the blue line on the upper plot of Fig. 3.

Choose a range of low-energy data for a linear regression, and copy the regression parameters to cell(8,1) and cell(8,2). Select the highest energy bin to keep and place its row number in cell(8,6); the default is to keep all energies, but an upper limit of 30 eV is more than adequate. Select the number of low-energy bins to be replaced by the regression line, if any, and place that number in cell(8,5). Then perform transform **Energy1** (see Appendix B). The cumulative λ -weighted distribution will be formed and plotted as the blue open circles on the lower plot of Fig. 3. The normalization factors will be saved as:

```
cell(8,7) = integrated source brightness,  $n / (\text{MW-s}) / \text{mm}^2 / \mu\text{ster}$ 
cell(8,9) =  $\lambda$ -weighted integral,  $n \cdot \text{\AA} / (\text{MW-s}) / \text{mm}^2 / \mu\text{ster}$ 
cell(8,10) = mean value of  $\lambda$  for this spectrum ( $\text{\AA}$ )
```

SNS-sct Bottom Upstream, 25-mm poison, H₂O
Energy Spectrum



Cumulative Energy Distribution (weighted by λ)

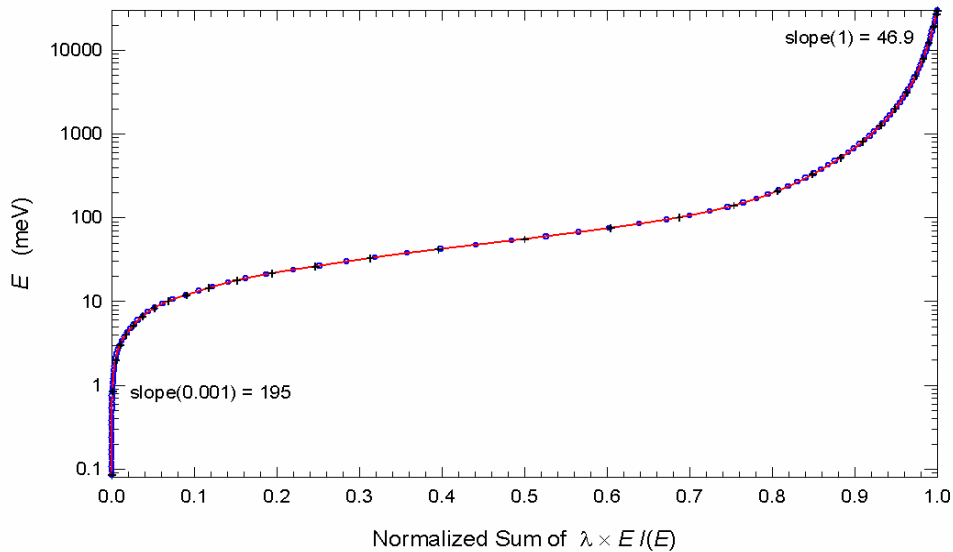


Figure 3. Analysis of point-detector energy spectrum. Lowest energy bins are replaced by a fitted power law and the cumulative distribution weighted by λ , $\Sigma[\lambda E I(E)]$, is formed as shown by the circles in the lower plot. 33 specific nodes are interpolated, and a cubic spline is calculated (+ symbols and continuous curve on lower plot). The reciprocal of the first derivative of the spline gives the nodes and curve in the upper plot.

The nodes X for the spline function are *always* at 33 specific values of the normalized cumulative distribution, and they are shown as the black + symbols in the lower plot of Fig. 3 (also tabulated in section IV below). The variable spacing of the nodes tracks the typical distribution very well. Note that the two outermost nodes each represent only 0.1% of the distribution, while the two intervals nearest the median are each slightly over 10% of the distribution. However, the cubic spline can **not** adequately reproduce the power law behavior at lowest energies, and because the reverse interpolation of the spline relies on the *reciprocal* of the first derivative the result has a scalloped appearance. Therefore the spline is fitted to nodes 2-33 and an analytic form is used to extrapolate the lowest 0.1% of the distribution.

The cubic spline algorithm is from “Numerical Recipes, 2nd ed.” [W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (Cambridge University Press, 1992), section 3.3]. It is coded inline in the **Energy1** transform (and twice in transform **5_spline** in Appendix C), but is much easier to read in the original Fortran (or even C!). The numerical results have been carefully compared to the Fortran version. Given the nodes X and the function $F(X) = \log_{10}[E(X) / (1 \text{ meV})]$ at the nodes, the algorithm finds d^2F/dX^2 at each node such that the first two derivatives are continuous and the function is cubic between nodes. The red line on the lower plot of Fig. 3 is interpolated using the spline function. The red points and dashed line on the upper plot are computed from the reciprocal of the first derivative of the spline.

There are two (*and only two*) adjustable parameters of the spline function, namely the slopes at the two end points. If cell(20,4) and cell(20,5) are blank, the transform tries to estimate the slopes such that the end nodes on the upper plot will be close to the “correct” values. The user then adjusts these estimates and reruns **Energy1**. Entering a larger value moves the node next to the end upward and *vice versa*. When you get close, there will be an up/down alternation of nodes. Adjust to make the function as smooth as possible. Note: because the sign of the 2nd derivative is used as a flag for whether to use the spline or a power law in the the end node, it is *essential* that the last d^2F/dX^2 , cell(18,33), be *positive*. Increase cell(20,5) if necessary to make this so. The chosen boundary conditions have been entered on the lower plot.

The values of X , $F(X)$, and d^2F/dX^2 are shown in Table 1 below. The first value in the d^2F/dX^2 column is actually the slope factor used to extrapolate the power law at very low energy.

III. Pulse Shapes

As might be expected, the analysis of the two-dimensional surface-tally data is more complex than the energy spectrum. The philosophy is to parametrize the pulse shapes so that after an energy has been selected, then the energy-specific pulse shape will be used to select the emission time of the neutron. The fundamental notion is that the high-energy part of the spectrum is dominated by the slowing-down process, and thus that the pulse shape can be expressed as a function $A(t/\lambda)$. (*Cf.* Ikeda & Carpenter.) On the other hand, low-energy pulse shapes are dominated by leakage of stored neutrons which is independent of wavelength, $B(t)$. For the present work, the pulse shape at intermediate energies is taken to be a simple linear combination of the two limits, using a linear or bi-linear switch function.

Import the .sigplt file (starting with row 2) into a new worksheet starting at cell(1,1). Delete the trailing 0s in col(2). If there are late time slices with null or spurious data, now is the best time to delete them. First shift col(2) down by the number of rows to be deleted, then shift the entire worksheet up to delete the rows. The analysis proceeds in numerical order through the transforms and regressions listed in Appendix C, but there are many choices and judgements to be made.

A. Initialization and normalization

Run transform **1_summ** (see Appendix C) to initialize and normalize the input shapes for each wavelength. This creates an array of neutron counts per bin from the input $n/\mu s$, and puts the sums of the energy band in col(72). It also places some parameters in col(82) and calculates λ (Å) in col(85). The procedure also finds the mean, standard deviation, and Gini statistic of each time distribution. Since the data bins are logarithmic, we are computing moments with respect to $\log(t)$ instead of t . To find the “linear” average,

$$\mu(t) = \exp[\mu(\ln(t))]$$

$$\sigma(t) = \mu(t) \sigma[\ln(t)]$$

For this reason, and especially because the distribution is very skew if plotted on a linear scale, my computed statistics from the SNS data file do *not* agree with the SNS “metrics” file.

B. Summation of late-time data

You must decide what wavelengths and what time range to include in the long-wavelength limit. From Fig. 2 (or the data array) select the wavelengths for which the pulse shape seems to be a function of t , and put the min and max index numbers in cell(82,20) and cell(82,21). (These were initialized to 1 and 9.) Also choose the earliest time slice to include, and put its row number in cell(82,9). Then run transform **2_late** to average the normalized shapes at long wavelength to find $B(t)$.

After executing transform **2_late** the data are plotted as the blue open circles (with statistical error bars) in the upper part of Fig. 4 (see section 5 below). The title of the plot should be edited to include the actual range of included wavelengths. The mean and standard deviation of the averaged data, found respectively in cell(82,30) and cell(83,30), should also be copied as text on the plot. Additional analysis of Fig. 4 will be discussed later, in section 5.

C. Average of early-time data

Run transform **3_avg_tl** to average normalized shapes at short wavelength to find $A(t/\lambda)$. This step relies on Erik's promise always to make the bins in the surface tally such that bins-per-decade of time are an *even* multiple (2 or greater) of the bins-per-decade of energy. Then it will be possible to shift columns up and down to make values of t/λ line up. The number of row shifts per column has been computed and is in cell(82,5). (It is 2 for the SNS simulations and 5 for the IPNS moderators.) In the transform the value of that cell is given the name "ratio".

The time-slice rows to be included in the t/λ conversion will be identified in col(76), "first t", and col(77), "last t". The first 23 cells in these columns correspond to the 23 energy bands. I chose to take some time slices from each of the highest 10 energies, so I entered numbers on rows 14 through 23. Looking at data col(71), which corresponds to the 23rd energy band, I decided to include rows (time slices) 19 through 72 which is the last row. In this case the slope of "first t" is equal to ratio, but sometimes it is not such a straight line. The "last t" is constant as long as there are non-zero cells.

	-76- first t	-77- last t
1		
2		
...		
14	1	72
15	3	72
16	5	72
17	7	72
18	9	72
19	11	72
20	13	72
21	15	72
22	17	72
23	19	72

After executing transform **3_avg_tl** the data are plotted as the blue open circles (with statistical error bars) in the upper part of Fig. 5. The title of the plot should be edited show the range of included wavelengths, and the mean and standard deviations entered as text from cell(82,32) and cell(83,33).

D. Optional functional fit to early-time data

The procedure previously used to parametrize the pulse shape for NISP *required* that a formula be fitted to the early data, and a number of regression equations were written. While it is no longer *necessary* to make such a fit, it is still interesting to see how well the early time data can be represented by an analytic expression. By far the most successful form we have tried is the Ikeda-Carpenter partially convoluted with an additional exponential, which is shown in Appendix C as regression **4_i_c_exp**. As the regression is non-linear, it may be necessary to adjust the starting values to get a successful convergence. After performing the regression, store the results as follows:

Parameters → col(83) "more params"

Predicted → col(84) "Early fit"

Residuals: not saved

The fitted curve is the black line in the upper plot of Fig. 5. The parameters of the fitted function have also been entered as text on the plot.

It is possible – but very seldom necessary or useful – to use the function in col(84) to smooth the early data when finding the cumulative distribution. To do so, enter the number of data points to be replaced in cell(82,11). In principle one could enter a numerical function in col(84).

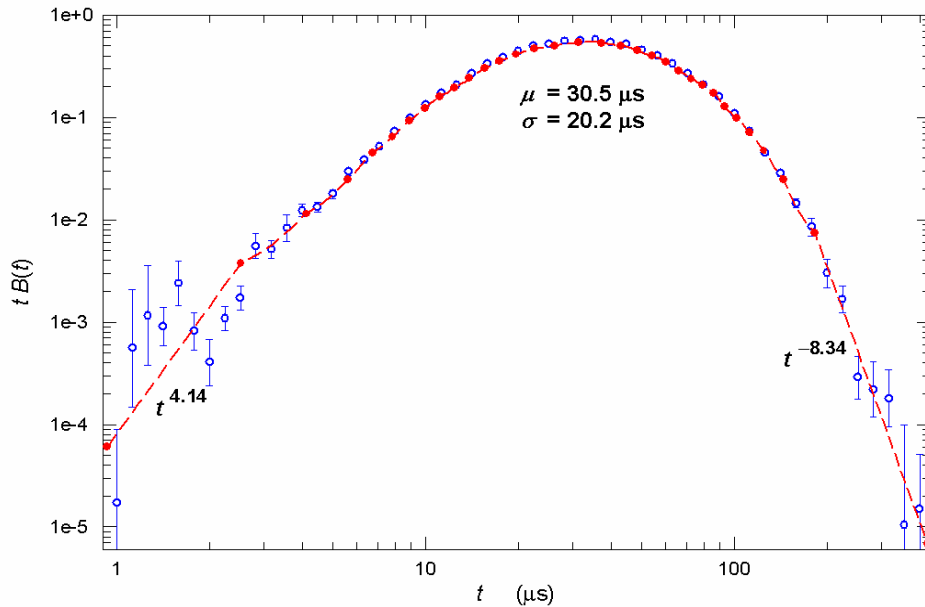
E. Fit spline curves

Transform **5_spline** will form the cumulative distributions of $A(t/\lambda)$ and of $B(t)$, and then generate spline curves defined as

$$G(X) = \log_{10}[(t/\lambda)(X) / (1 \mu\text{s}/\text{\AA})] \quad H(X) = \log_{10}[t(X) / (1 \mu\text{s})]$$

The spline algorithm used for both data sets is the same as for the energy above, using exactly the same nodes in the normalized cumulative distributions. The new fitting procedure uses power laws instead of splines at *both* limits, that is, below $F=0.001$ and above $F=0.999$. The slopes at $F=0.001$ and $F=0.999$ are free parameters chosen to make the spline as smooth as possible. Those slopes are subsequently used to determine the respective power laws. The cumulative distributions and fitted splines are shown in the

SNS-sct Bottom Upstream, H₂O, 25-mm poison
Late Time Pulse Shape (wavelengths 5.5–17 Å)



Cumulative Distribution of Late-time Pulse Shape

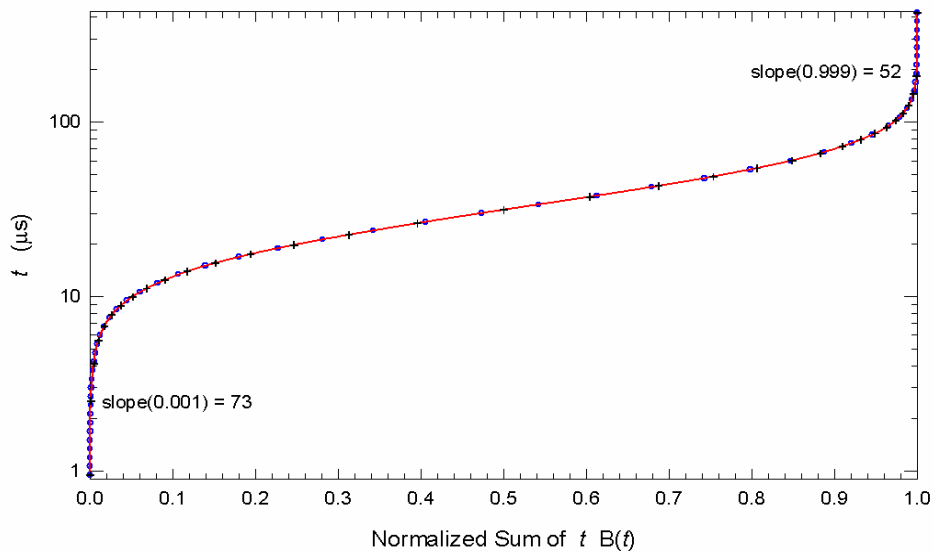


Figure 4. The long-wavelength limit, $B(t)$. The upper plot is the average of the normalized pulse shapes for several wavelengths, and the lower plot is the cumulative distribution. Analysis is the same as for Fig. 3.

lower plots of Figs. 4 and 5, and the dashed red lines in the upper plots show the reverse interpolation with the power-law extensions.

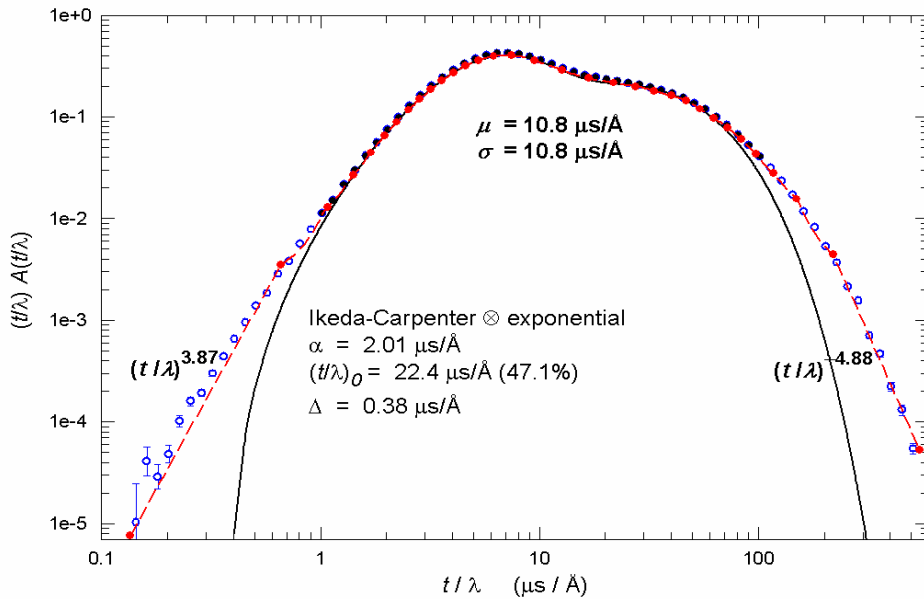
After the first execution of transform **5_spline**, the boundary conditions will have to be adjusted to smooth the plots in the same manner as was done for the energy spectrum. Note that

$$\text{cell}(100,8) = \text{slope}(0.001), \text{ Fig. 4} \quad \text{cell}(100,11) = \text{slope}(0.001), \text{ Fig. 5.}$$

$$\text{cell}(100,9) = \text{slope}(0.999), \text{ Fig. 4} \quad \text{cell}(100,12) = \text{slope}(0.999), \text{ Fig. 5}$$

Increasing any of these values will raise (or decreasing will lower) the node next to the corresponding end node. Repeat the transform until you are satisfied with the smoothness of the results, paying attention especially up/down alternations (ringing). Many iterations are usually necessary. If you can *not*

SNS-stc Bottom Upstream, H₂O, 25-mm poison
 Early Time Pulse Shape (wavelengths 0.09–0.7 Å)



Cumulative Distribution of Early-time Pulse Shape

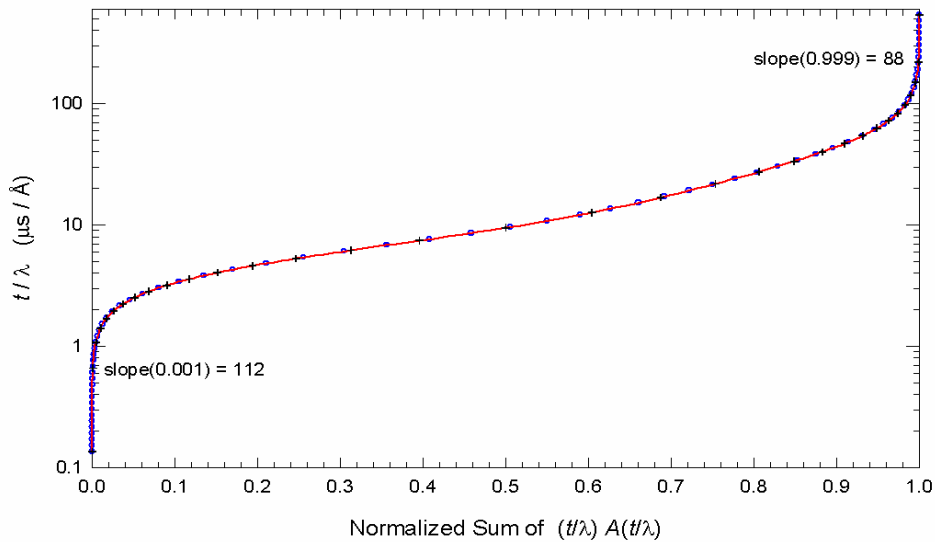


Figure 5. The short-wavelength limit, $A(t/\lambda)$. The upper plot is the average of the normalized pulse shapes for several wavelengths, and the lower plot is the cumulative distribution. An (optional) fitted function is shown on the upper plot, but is not necessarily used further. Analysis is the same as for Fig. 3.

achieve satisfactory end conditions, it may be because you are attempting to fit very low values at the end of the probability distribution. Better results may be possible if some data are omitted.

Finally, enter the slopes on the lower plots, and the power-law extensions on the upper plots.

E. The switch function

The most important part of the spectrum for many neutron scattering instrument is intermediate between the two limits discussed above. The model used in NISP assumes a linear combination of the two limiting cases:

$$I(t, \lambda) = (1 - R(\lambda)) A(t/\lambda) / \lambda + R(\lambda) B(t)$$

The switch function $R(\lambda)$ is 0 in the short-wavelength limit and 1 in the long-wavelength limit. In many cases a linear variation proportional to $\log(\lambda)$ between two limiting values of λ is sufficient; in other cases the function has two linear segments and a knee. The task is to find an optimal value of R for each value of λ and then fit the appropriate form of switch function to those values.

1. Initial estimate

The limiting wavelengths chosen for the long- and short-wavelength averages were previously stored in cell(83,16) and cell(83,17), and running transform **6_switch** will calculate the corresponding linear switch function. By generating this estimate of the switch function, there will be an additional curve on the pulse shape fitting plots to guide you in choosing R at each wavelength.

2. Plot the pulse shape for individual wavelengths

In order to see a plot of the pulse shapes for a particular value of λ from col(85), place its index number in cell(83,7) and execute transform **7_seefit**. (There is an indicator in cell(82,7) pointing to the cell where you put the λ number.) Plot templates are generally created for 7 wavelengths, namely #3 through #21 by 3s. Other data can be viewed on a neighboring template.

Four sample plots are shown in Figs. 6–9. Since a log t scale is used, the ordinate is $t I(t)$ so that the height of a point always represents the same number of neutrons. The input data points are black, the “early” function is red (hot), the “late” function is blue (cold), and the linear combination using the currently stored value for the switch function is brown. Compare the brown curve to the data points.

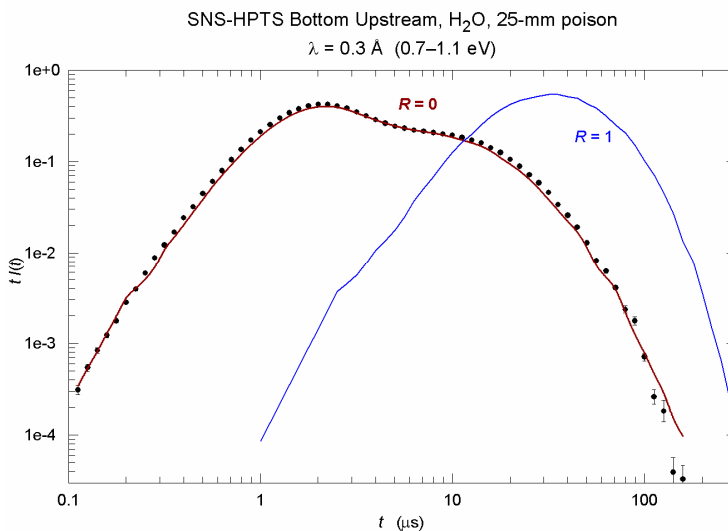


Figure 6. Pulse shape at short wavelength, 0.3 Å. The red curves on this and following plots are the short-wavelength limit, the blue curves are the long-wavelength limit, and the black points are the data from the surface tally for one energy bin. The solid brown lines (in this case coincident with the red line) are the final results using the switch function from Fig. 10. R is the fraction of the late-time data in the linear combination.

3. Regression analysis to find optimum R

Execute the regression procedure **8_ratio** for each wavelength, after transform **7_seefit** has been executed with the case number in cell(83,7). The result is a single parameter, the optimum value of the

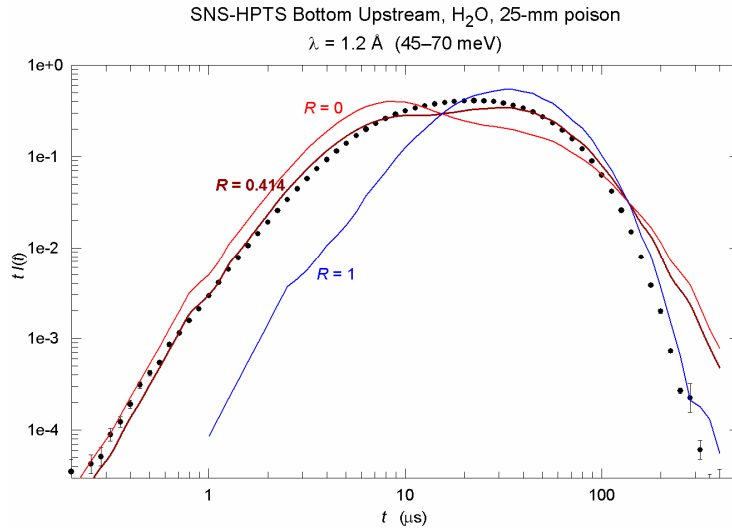


Figure 7. Pulse shape at intermediate wavelength 1.2 Å. See Fig. 6 for color key.

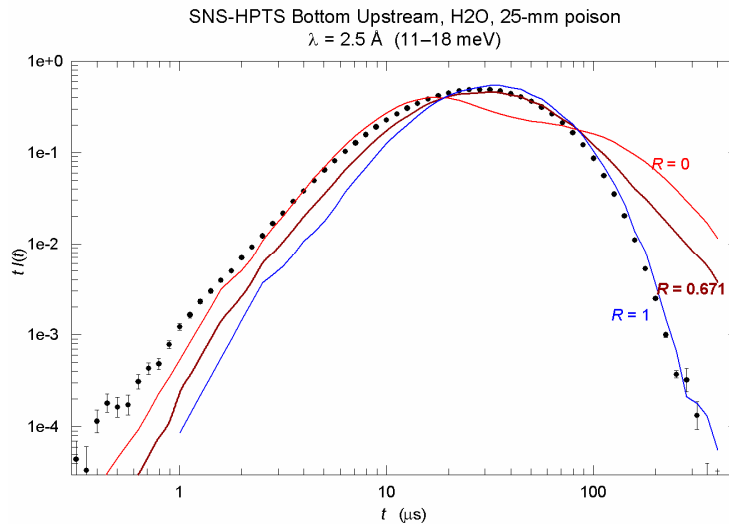


Figure 8. Pulse shape at intermediate wavelength 2.5 Å. See Fig. 6 for color key.

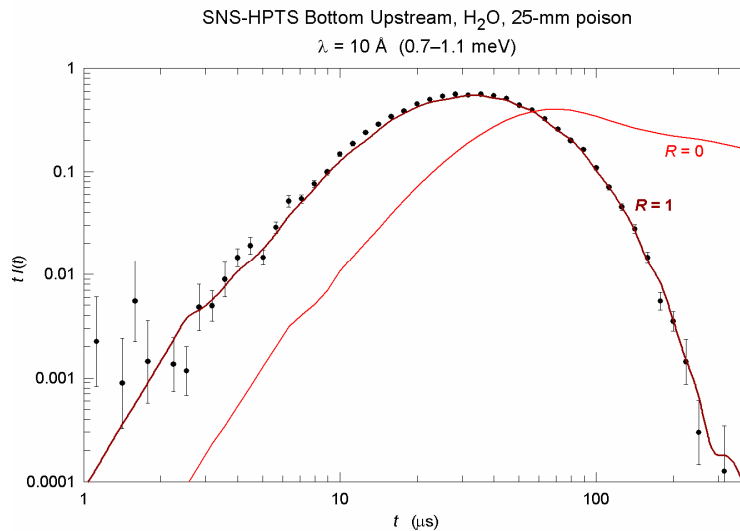


Figure 9. Pulse shape at long wavelength 10 Å. See Fig. 6 for color key.

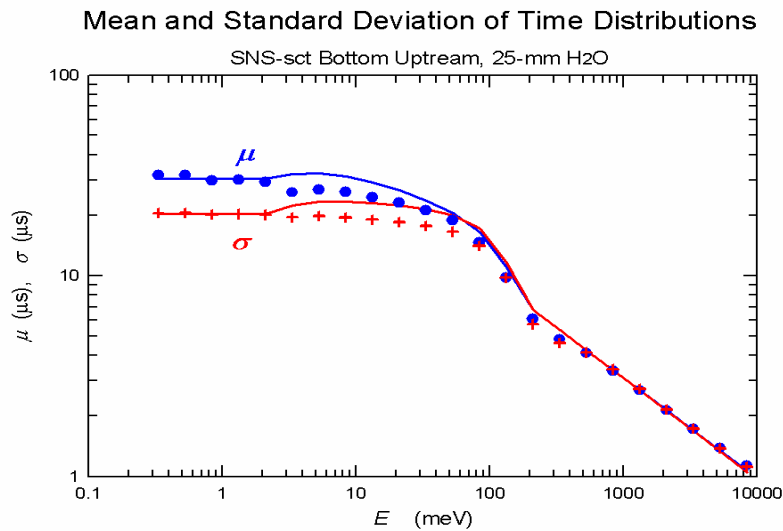
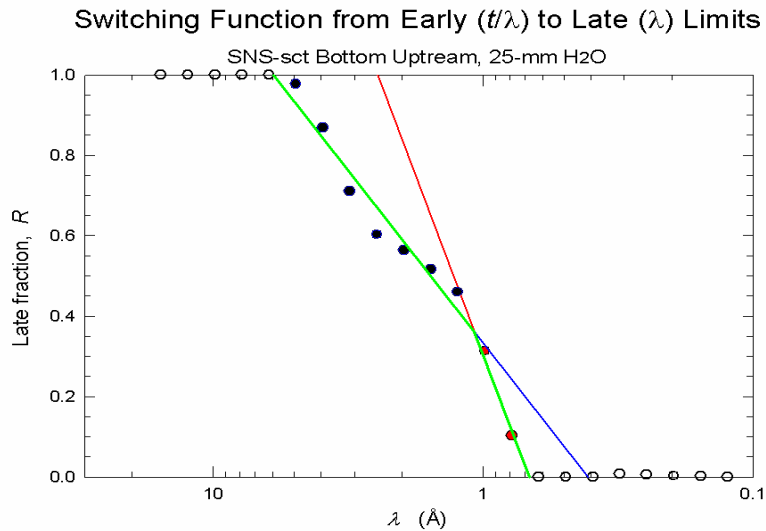


Figure 10. Top: Linear regressions of the “best fit” values of R for various wavelengths. The black line is a fit omitting the high points. The red and blue lines are fitted to the points of the same colors. The green line is the selected switch function; in this case two linear segments. Bottom: symbols are mean and standard deviation of the raw input distributions, and the lines are propagated as the linear combination of the early and late functions.

switch function R . Place that number in the corresponding cell in col(86); note that the adjacent column has the values of R from the currently stored switch function. Optionally, you may also store

Predicted \rightarrow col(90) “Best fit”

so that the best fit may be added to the plot as a *dotted* brown line.

Run these two procedures in turn for every wavelength. As each computed value of R is recorded, it will appear on the “Switch Function” plot, Fig. 10 top.

Use the linear regression line feature of SigmaPlot to draw straight lines through selected data points. The switch function may often be described by a single regression, but in this case a somewhat better fit is obtained with two linear segments and a knee. Place the coefficients of the black line (if single) or the red line (if two segments) in cell(83,16) and cell(83,17). For two segments the coefficients of the blue line go to cell(83,28) and cell(83,29); otherwise these cells must be empty. Then rerun transform **6_switch**.

Whenever transform **6_switch** is executed, the new switch function will be calculated and drawn on the plot (green). In addition, the means and standard deviations of the 23 computed time distributions are shown as lines in Fig. 10 bottom, compared to the input data plotted as symbols. This is a good test of

whether the ranges chosen for averaging the early and late data were appropriate: the statistics are constant at long wavelength, and proportional to $1/\sqrt{E}$ at short wavelengths.

Finally, re-run transform **7_seefit** for each desired plot (such as Figs. 6—9) IYou should write the value of the switch function on each plot.

IV. Output Tables

The final result of the fitting procedure is a .tbl file with the information in the following table:

SPLI SNS-sct521_bu_17, 25-mm H2O		EBI_020327_175548				
198, 0, 0, 21429.27e12 , -33, 0, 0, 0.2136, 0.3609, 0.6882, 0.6765, 5.9900, 14748.00e12/						
spline[\gS[fit[2.45E-9(sct521_bu_17,25H2O)(n/(MW-s)/m\u2\d/ster/ln(E))]*(\gl)]]						
X	F	d ² F/dX ²	G	d ² G/dX ²	H	d ² H/dX ²
0.0000	-1.0750	4.7875 *	-0.8675	8.9092 *	-0.0235	9.5435 *
1.0000e-3	-0.0873	-66080.9277	-0.1805	-41943.3195	0.4025	-34244.4174
4.5500e-3	0.2749	-8068.3008	0.0319	-4293.4689	0.6148	-4694.2022
0.0102	0.4813	-1891.2059	0.1497	-1143.3016	0.7484	-1689.1758
0.0173	0.6210	-792.0902	0.2294	-464.6968	0.8296	-346.7755
0.0262	0.7336	-340.6811	0.2926	-175.1827	0.8937	-239.6871
0.0373	0.8325	-178.6356	0.3490	-101.1657	0.9485	-91.0061
0.0512	0.9234	-98.3102	0.4016	-51.3619	0.9986	-54.7512
0.0686	1.0095	-53.5757	0.4525	-30.3638	1.0464	-24.6925
0.0903	1.0934	-31.9291	0.5029	-17.4222	1.0944	-16.3448
0.1174	1.1764	-19.3835	0.5540	-10.3677	1.1433	-13.0668
0.1513	1.2598	-11.2632	0.6068	-6.2786	1.1917	-6.0753
0.1937	1.3452	-6.9773	0.6626	-3.5323	1.2415	-3.1304
0.2466	1.4340	-4.1031	0.7230	-1.9616	1.2951	-2.5408
0.3129	1.5285	-2.2283	0.7909	-0.6868	1.3530	-0.7778
0.3956	1.6328	-0.8706	0.8714	0.2114	1.4191	-0.6242
0.5000	1.7556	0.0693	0.9768	1.9490	1.4965	-0.4571
0.6044	1.8823	2.7591	1.1026	3.2149	1.5708	0.7185
0.6871	2.0067	8.3711	1.2254	3.2612	1.6331	0.3631
0.7534	2.1500	18.8948	1.3390	2.2402	1.6866	2.3606
0.8063	2.3242	31.7264	1.4381	3.3666	1.7353	1.5474
0.8487	2.5220	29.0750	1.5246	6.4707	1.7788	5.5619
0.8826	2.7198	35.6589	1.6018	7.9813	1.8204	8.1989
0.9097	2.9118	79.1705	1.6707	14.5970	1.8605	11.4515
0.9314	3.1060	116.2666	1.7346	35.9766	1.8983	12.4616
0.9488	3.3022	190.6463	1.7972	49.7237	1.9337	30.7787
0.9627	3.4994	250.6818	1.8588	87.5329	1.9695	79.7305
0.9738	3.6957	507.3408	1.9210	175.7634	2.0082	90.3200
0.9827	3.8950	614.1943	1.9882	420.2107	2.0486	213.7365
0.9898	4.0913	979.7174	2.0680	975.2446	2.0952	626.3788
0.9955	4.2851	1629.3544	2.1737	2864.2027	2.1584	1709.1844
0.9990	4.4304	1561.1669	2.3429	32660.5104	2.2589	19168.1309
1.0000	4.4750	1943.9828	2.7325	-11.2280 *	2.6264	-19.2144 *

* These entries are power-law slopes, not 2nd derivatives

The first header line of the file has the keyword `SPLI` followed by the title and identifier that were originally edited into the data file. The format is (A4,1X,A40,1X,A17). The second line has 13 numeric parameters, seven of which are of use for normalization and the switch function:

parameter(4) = spectrum normalization from cell(8,7) of the energy worksheet,
 $\times 1.e12$ to give units of $n / (MW-s) / m^2 / ster$

parameter(8) = 0, or fractional position of knee from cell(82,26) of the pulse worksheet

parameter(9) = 0, or value of R at the knee from cell(83,24) of the pulse worksheet
parameter(10) = mean wavelength (\AA) from cell (8,10) of the energy worksheet
parameter(11) = switch function min λ (\AA) from cell(82,23) of the pulse worksheet
parameter(12) = switch function max λ (\AA) from cell(82,25) of the pulse worksheet
parameter(13) = weighted normalization from cell(8,9) of the energy worksheet,
 $\times 1.e12$ to give units of $n\text{-}\text{\AA} / (\text{MW-s}) / \text{m}^2 / \text{ster}$

The third header line indicates the operations that have been performed. The innermost level of parentheses is the identifier of the original MCNP data file. It has been multiplied by a constant with the indicated units. The operator “fit” refers to the power-law tail of the energy spectrum. The file was then multiplied by λ and summed. (Note that the characters “\g” in the string cause the following single character to be converted to Greek.) Finally, the “spline” operation was done.

The actual table format has the data in rows instead of columns, as in all NISP data files. The line above with column headings is replaced in the file by the keyword `DATA`, and the data for F , $F2$, G , $G2$, H , and $H2$ are given as a single comma-separated block of 198 values. The X nodes are shown here only for information; they are not included on the table file because they can be always the same and are built into the code. The spline function nodes and second derivatives are copied from the two worksheets:

$F \leftarrow$ col(17) of the energy worksheet
 $F2 \leftarrow$ col(18) of the energy worksheet
 $G \leftarrow$ col(105) of the pulse worksheet
 $G2 \leftarrow$ col(106) of the pulse worksheet
 $H \leftarrow$ col(117) of the pulse worksheet
 $H2 \leftarrow$ col(118) of the pulse worksheet

V. Using the Table

The primary use of the tables is to generate source particles for Monte Carlo simulations. Interpolating the spline function F at a uniform random value of X gives the appropriate random value of $\log_{10}(E / (1 \text{ meV}))$. After a value of E has been selected and λ calculated, the switch function is evaluated. Remember also that the statistical weight of the neutron must be proportional to $1/\lambda$. The pulse shape will be “early” if $\lambda < \lambda_{\text{min}}$, “late” if $\lambda > \lambda_{\text{max}}$, or determined by comparing R to a uniform random number for intermediate cases. If “early” then a random value of $\log_{10}((t\lambda) / (1 \mu\text{s}/\text{\AA}))$ is found by interpolating G , or if “late” then $\log_{10}(t / (1 \mu\text{s}))$ is found from H . In general three random numbers may be used: first, to find E ; second, to choose early or late based on the switch function; and third to select t either from the early or from the late distribution.

In the NISP code this process occurs in the subroutine `N_SOURCE`, which in turn calls `PLSPLINE` (version 23 Nov 2004) to sample each distribution. Note that a only version of `MC_Run` that has been linked to the `MCLIB` library *later* than 23 Nov 2004 will be able to read tables with power-law extrapolations for the end nodes. Although not needed for Monte Carlo simulations, a subroutine for reverse interpolation in the spline tables, `RSPLINT`, is also included in `MCLLIB`. This allows the probability density to be estimated for a given E and t .

Histograms of a test of 30,000,000 randomly selected neutrons are shown in Fig. 11 and 12, which should be compared to Figs. 1 and 2 respectively. Execution time (on a 2.8 GHz Pentium IV) was 40 s for neutron generation and binning (*i.e.*, 1.3 μs per neutron).

All source tables can be downloaded as part of the NISP package at <http://PASeeger.com>. The files are in the `tables/` folder, and will automatically appear in the list of available sources in NISP. Users must be aware that to use a file with this format they must have a version of `MC_Run` linked with the `MCLIB` library *later* than 23 Nov 2004 in order to have a version of `PLSPLINE` that recognizes that the limiting nodes are power laws of the logarithm instead of splines. The latest code versions are always included in the download from <http://PASeeger.com>. Additional documentation can also be downloaded from the same site.

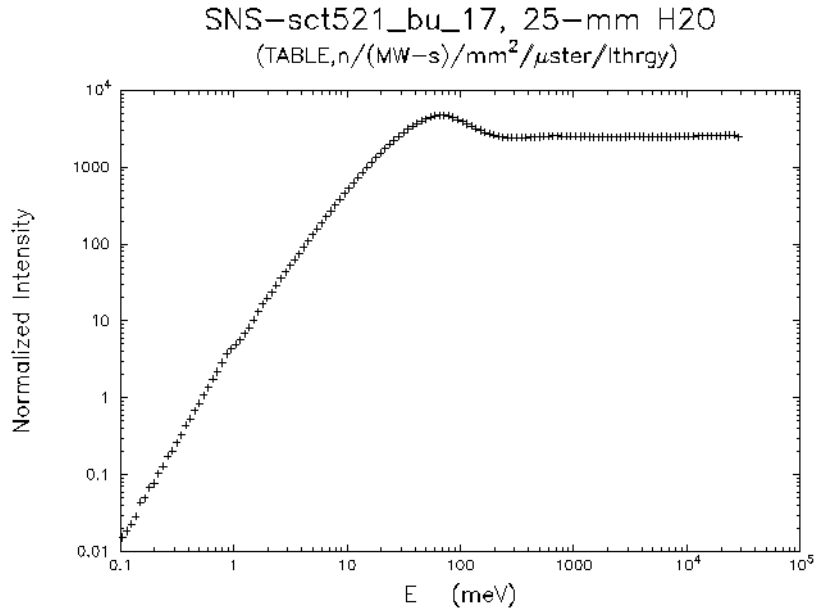


Figure 11. Computed energy spectrum using the output table. Compare to Fig. 1; artifacts and statistical fluctuations have been smoothed.

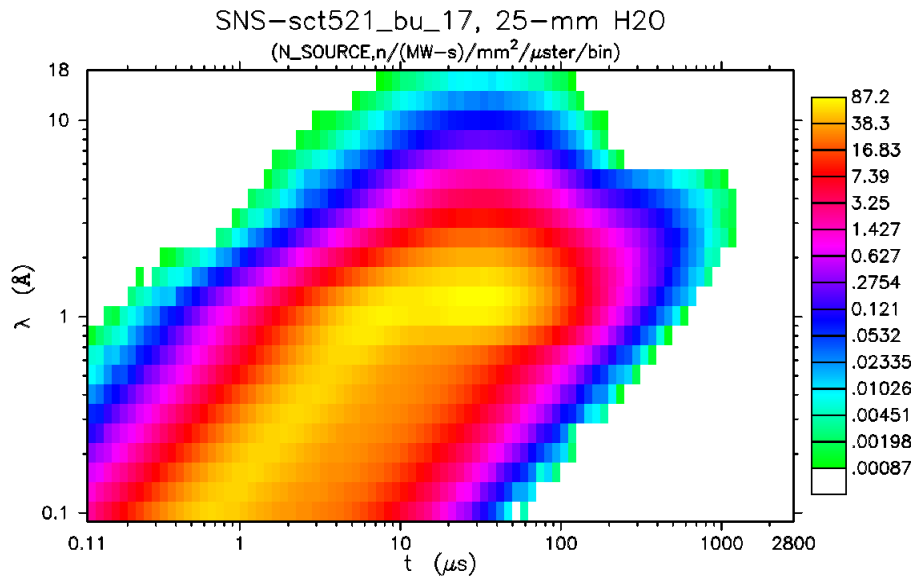


Figure 12. Histogram (sum of weights) of 3,000,000 samples from the table. Since the bins are logarithmic in both dimensions, the data have been divided by the logarithmic width in time and the logarithmic height in λ . Compare to Fig. 2.

As of this revision, the following source files have been generated using this procedure:

<code>SNS_sct_tu02_dH2.tb1</code>	SNS top upstream, f.p. 2, decoupled liquid Hydrogen
<code>SNS_sct_td05_cH2.tb1</code>	SNS top downstream, f.p. 5, coupled liquid Hydrogen
<code>SNS_sct_bu08_15H2O.tb1</code>	SNS bottom upstream, f.p.8, water poisoned at 15 mm
<code>SNS_sct_tu11_dH2.tb1</code>	SNS top upstream, f.p. 11, decoupled liquid Hydrogen
<code>SNS_sct_td14_cH2.tb1</code>	SNS top downstream, f.p. 14, coupled liquid Hydrogen
<code>SNS_sct_bu17_25H2O.tb1</code>	SNS bottom upstream, f.p.17, water poisoned at 25 mm

SNSL_fwdS.tb1	SNS LPTS front wing, decoupled Solid methane, 22K
SNSL_ssdL.tb1	SNS LPTS starboard slab, decoupled Liquid methane, 100K
ISIS1W_S2&N8_H2O.tb1	ISIS W target, f.p. S2 & N8, Water
ISIS1W_S8&N2_CH4.tb1	ISIS W target, f.p. S8 & N2, Methane
ISIS1W_N5_liqH2.tb1	ISIS W target, f.p. N5, liquid Hydrogen
ISIS2_H2-TaVSBBe.tb1	ISIS target 2, Ta vessel, solid Be, liquid Hydrogen
ISIS2_Narrow-TaVSBBe.tb1	ISIS target 2, Ta vessel, solid Be, "Narrow"
ISIS2_Groove-TaVSBBe.tb1	ISIS target 2, Ta vessel, solid Be, "Groove"
ISIS2_Broad-TaVSBBe.tb1	ISIS target 2, Ta vessel, solid Be, "Broad"
LujanII_01_HiResH2O.tb1	Lujan f.p. 1, high resolution water
LujanII_04_H2O.tb1	Lujan f.p. 4, high intensity water
LujanII_07_H2O.tb1	Lujan f.p. 7, high intensity water
LujanII_10_dcplH2.tb1	Lujan f.p.10, decoupled liquid Hydrogen
LujanII_12_cp1dH2.tb1	Lujan f.p. 12, upper coupled liquid Hydrogen
LujanII_14_upH2O.tb1	Lujan f.p. 14, upper Water
IPNS_C_groovedCH4.tb1	IPNS "Cold" moderator, grooved solid methane, 30K
IPNS_F_liqCH4.tb1	IPNS "Fast" moderator, poisoned liquid methane, 100K
IPNS_H_solCH4.tb1	IPNS "High Intensity" moderator, poisoned solid methane, 30 K

Please address any questions, comments, or problems to PASeeger@losalamos.com, and/or Luke Daemen at lld@lanl.gov.

Appendix A: Fortran program ARRANGE

```

PROGRAM ARRANGE
c
c Read (edited) output file from MCNP and convert to 1-D (point detector)
c and 2-D (surface tally) files in Block-ASCII for pictures and in
c text files for input to SigmaPlot.
c
c P. A. Seeger, August 1, 2000
c copied from Arrang_1D (v. 23 Aug 1995) and Arrang_2D (v. 27 May 1997)
c 13 Oct 2000: combine energy bins if necessary [PAS]
c 06 Jul 2001: keep 23 energy bins in spreadsheet output [PAS]
c 07 Jul 2001: don't combine unless more than 28 energies [PAS]
c 17 Jul 2001: include units of /eV for E-t input [PAS]
c 10 Jun 2003: changed for ISIS, explicit input of 1st energy to keep [PAS]
c 18 Jun 2004: option for E-t only by setting ne=0;
c variable for number of bins to combine [PAS]
c 24 Jul 2004: change for SNS, input 1st E & #bins to combine; E-t array
c in multiple E blocks [PAS]
c
implicit none
integer maxe, maxt, maxd
parameter (maxe=150, maxd=23, maxt=120)
real*4 ebins(maxe+1), edata(maxe), dedata(maxe), &
& tbins(maxt+1), ETHist(maxe,maxt), dETHist(maxe,maxt), &
& TLhist(maxt,maxd), dTLhist(maxt,maxd), lambda(maxd+1), &
& energy(maxe), time(maxt)
c
real*4 pulse, area, factor, param(32), f, e, t, col(48)
real*8 sumDET2
integer ne, me, mt, i, j, lhdrin, lhdrout, idot, i1, i2, ii, &
& ierr, initE, kombine, k
character filename*80, outfile*80, type*4, title*40, id*17, &
& hdrin*80, hdrout*80, units*25, line*80
logical flag
data units/'(n/(MW-s)/mm\u2\d/\gmster'/'

c
print *, ' Convert Erik Iverson's MCNPx output to Block ASCII'
10 continue
print *
print *, ' Enter input file name: '
read (*, '(a)') filename
if (filename.eq.' ' .or. filename(1:1).eq.'/') stop
idot = index(filename, '.')
if (idot .eq. 0) then
  idot = index(filename, ' ')
  filename(idot:) = '.dat'
end if
c
c Get initial seven lines of input (edited on to Erik's file)
open (1, file=filename, status='old', form='formatted', iostat=k)
if (k .ne. 0) then
  print *, '*** failed to open ',filename(1:len_trim(filename))
  go to 10
end if
read (1, '(a)') title
print *, ' Title: ',title
read (1, '(a)') id
print *, ' File id: ',id
read (1, *) pulse, area
c
c Convert area from cm^2 to mm^2, ster to uster
factor = pulse * area * 1.e8
print *, ' Point-detector normalization:',factor, ' MW-s mm^2 uster'
c
read (1,*) ne
if (ne .gt. 0) then
  print *, ' Points in energy spectrum:',int2(ne)

```

```

        if (ne .gt. maxe) then
            print *, ' *** Dimension exceeds array space in program:', &
&             int2(maxe)
            stop
        end if
    else
        print *, ' No Energy spectrum'
    end if
c
    read (1,*) me, mt
    print *, ' Surface tally:',int2(me),' energy x',int2(mt),' time'
    if (me.gt.maxe .or. mt.gt.maxt) then
        print *, ' *** Dimensions exceed array space in program:', &
&             int2(maxe),' x',int2(maxt)
        me = min(me, maxe)
        mt = min(mt, maxt)
    end if
    read (1,*) initE,kombine
    print *, 'First energy column will be',initE,', combine by', &
&             kombine,'s'
    read (1,'(a)') hdrin
    lhdrin = index(hdrin(2:), ' ')
    print *, ' Header:',hdrin(1:lhdrin)
c
c   Search for file name.extension that does not already exist
    outfile = filename(1:idot)//'et0'
    flag = .true.
    do while (flag)
        outfile(idot+3:idot+3) = char(ichar(outfile(idot+3:idot+3))+1)
        inquire(file=outfile, exist=flag)
    end do
    print *, ' ASCII output file: ',outfile
    open (3, file=outfile, status='unknown', form='formatted')
c
    if (ne .le. 0) go to 300
c
c   Search for beginning of point-detector data
    line = ' '
    do while (line(1:2) .eq. ' ')
        read (1,'(a)',end=900) line
    end do
    do while (line(1:2) .eq. '##')
        read (1,'(a)',end=900) line
    end do
c   Read point-detector data: E(eV), f(E) (n/pulse/eV/ster), sigma(E)
    do i=1,ne
        read (1,*,iostat=ierr) energy(i), f, edata(i), dedata(i)
        if (ierr .ne. 0) then
            print *, ' *** Energy data out-of-phase at line',int2(i)
            stop
        end if
c       New units will be n/(MW-s)/uster/mm^2/lethargy
        edata(i) = edata(i)/factor*energy(i)
        dedata(i) = dedata(i)/factor*energy(i)
c       Convert from eV to meV and generate bin boundaries
        energy(i) = energy(i)*1.e3
        if (i .gt. 1) ebins(i) = sqrt(energy(i-1)*energy(i))
    end do
c   Logarithmic extrapolation for 1st & last bins
    ebins(1) = ebins(2)**2 / ebins(3)
    ebins(ne+1) = ebins(ne)**2 / ebins(ne-1)
    print *, ' Energy range',ebins(1),' to',ebins(ne+1),' meV'
c
c   Output energy spectrum in Block ASCII format
    type = 'E '
    hdrout = hdrin
    lhdrout = lhdrin

```

```

call premul(1.d0/factor, hdROUT, lhdROUT)
hdROUT(lhdROUT+1:) = units//'/ln(E)) '
param(1) = float(ne)
param(11) = 0.
param(24) = 0.
12 write (3, 12) type,title,id
format (a4, 1x, a40, 1x, a17)
call real4out(param, 32, 3)
write (3, '(a)') hdROUT
write (3, '(a)') 'BINS'
call real4out(EBINS, ne+1, 3)
write (3, '(a)') 'DATA'
call real4out(EDATA, ne, 3)
write (3, '(a)') 'STDDEV'
call real4out(DEDATA, ne, 3)
write (3,*)

c
c Output energy spectrum in Column format
outfile(idot:) = '.cols'
open (4, name=outfile, status='UNKNOWN', form='FORMATTED')
write (4,12) type,title,id
call real4out(param, 32, 4)
write (4,'(a)') hdROUT
write (4,'(a)') ' POINTS          DATA          STDDEV      '
write (4, '(1p,3g14.6)') (energy(i),edata(i),dedata(i),i=1,ne)
write (4, *)
close (4)

C
300 continue
    if (me .eq. 0) go to 10

c
c Read the time array for the 1st energy
c Input data are: t (us), E (eV), counts (/us/eV), std.dev. (/us/eV)
do i=1,me
    line = ' '
    do while (line(1:12) .ne. ' 0.0000e+00')
        read (1,'(a)',end=900) line
    end do
    read (1,*) t,e
    energy(i) = e*1.e3
    if (i .gt. 1) EBINS(i) = sqrt(energy(i-1)*energy(i))
    do j = 1,mt-2
        read (1,*) t, e, ETHIST(i,j), dETHIST(i,j)
        if (i .eq. 1) then
c            times in the file are arithmetic bin centers
            time(j) = t
            if (j .gt. 1) TBINS(j) = (time(j-1) + time(j))/2.
        else if (t .ne. time(j)) then
            print *, '*** Data out of phase at e,t =',e,t
            stop
        end if
    end do
end do
c Extrapolate bin boundaries logarithmically
EBINS(1) = EBINS(2)**2 / EBINS(3)
EBINS(me+1) = EBINS(me)**2 / EBINS(me-1)
TBINS(1) = TBINS(2)**2 / TBINS(3)
TBINS(mt-1) = TBINS(mt-2)**2 / TBINS(mt-3)
c convert time to geometric mean
do j=1,mt-2
    time(j) = sqrt(TBINS(j)*TBINS(j+1))
end do
print *, ' E-t input has been read'
print *, ' Energy range:',EBINS(1),' to',EBINS(me+1),' meV'
print *, ' Time range:',TBINS(1),' to',TBINS(mt-1),' us'
c Convert from /eV to raw counts vs. E (still in units of /us)
do i=1,me

```

```

        f = (ebins(i+1) - ebins(i))/1000.
        do j=1,mt-2
            ETHist(i,j) = f*ETHist(i,j)
            dETHist(i,j) = f*dETHist(i,j)
        end do
    end do
C
C Combine energy bins to mamximum of maxd (=23)
me = min(maxd, (me-initE+1)/kombine)
i1 = initE
i2 = initE + kombine*me - 1
ebins(1) = ebins(initE)
do j=1,mt-2
    ii = 0
    do i=i1,i2,kombine
        ii = ii + 1
        ETHist(ii,j) = ETHist(i,j)
        dETHist(ii,j) = dETHist(i,j)
        if (kombine .gt. 1) then
            sumDET2 = dETHist(i,j)**2
            do k=1,kombine-1
                ETHist(ii,j) = ETHist(ii,j) + ETHist(i+k,j)
                sumDET2 = sumDET2 + dETHist(i+k,j)**2
            end do
            deThist(ii,j) = dsqrt(sumDET2)
        end if
        if (j .eq. 1) ebins(ii+1) = ebins(i+kombine)
    end do
end do
if (kombine.gt.1) print *, ' Reduced number of E bins to',int2(me)
print *, ' Energy range:',ebins(1),' to',ebins(me+1),' meV'
C
C Output file with "per us" E-t histogram
type = 'ET '
param(1) = float(me)
param(11) = 0.
param(24) = float(mt-2)
param(25) = 1./factor
write (3, 12) type,title,id
call real4out(param, 32, 3)
write (3, '(a)') hdrin(1:lhdrin)//'(n/\gms)'
write (3, '(a)') 'BINS'
call real4out(ebins, me+1, 3)
write (3, '(a)') 'BINS'
call real4out(tbins, mt-1, 3)
do j=1,mt-2
    write (3, '(a,i3)') 't = ',j
    call real4out(ETHist(1,j), me, 3)
    write (3, '(a)') 'STDDEV'
    call real4out(dETHist(1,j), me, 3)
end do
write (3,*)
C
C Flipped, vs. lambda instead of energy, "raw counts"
do i=1,me+1
    ii = me + 2 - i
    lambda(ii) = sqrt(81.8145347/ebins(i))
end do
do j=1,mt-2
    f = tbins(j+1) - tbins(j)
    do i=1,me
        ii = me - i + 1
        TLhist(j,ii) = f * ETHist(i,j)
        dTLhist(j,ii) = f * dETHist(i,j)
    end do
end do
type = 'TL '

```

```

param(1) = float(mt-2)
param(11) = 1.
param(24) = float(me)
write (3, 12) type,title,id
call real4out(param, 32, 3)
write (3, '(a)') hdrin(1:lhdrin)//'(raw_counts)'
write (3, '(a)') 'BINS'
call real4out(tbins, mt-1, 3)
write (3, '(a)') 'BINS'
call real4out(lambda, me+1, 3)
do i=1,me
  write (3, '(a,i3)') 'l = ',i
  call real4out(TLhist(1,i), mt-2, 3)
  write (3, '(a)') 'STDDEV'
  call real4out(dTLhist(1,i), mt-2, 3)
end do
write (3,*)
close (3)

c
c Output SigmaPlot format:
c   col(1) = mean time in bin (us) in decreasing order
c   col(2) = 23 energies (meV) in increasing order (zero filled)
c   col(3-25) = data (/us) for 23 energies at each time
c   col(26-48) = standard deviation (/us) for 23 energies at each time
do i=1,me
  energy(i) = sqrt(ebins(i)*ebins(i+1))
end do
outfile(idot:) = '.sigplt'
open (2, name=outfile, status='UNKNOWN', form='FORMATTED')
hdrout = hdrin(1:lhdrin)//'(/\gms)'
lhdrout = index(hdrout(2:), ' ')
write (2, '(a)') ' Spreadsheet: '//hdrout(1:lhdrout)
print *, ' Spreadsheet: '//hdrout(1:lhdrout)
ii = 1
do j=mt-2,1,-1
  col(1) = time(j)
  if (ii .le. me) then
    col(2) = energy(ii)
    ii = ii + 1
  else
    col(2) = 0.
  end if
  do i=1,me
    col(i+2) = EThist(i,j)
    col(i+25) = dEThist(i,j)
  end do
  call longout(col, 48, 2)
end do
write (2,*)
close (2)

c
goto 10
900 stop
end

```

Appendix B. Energy analysis transforms for SigmaPlot®

Worksheet column headings (can be copied to a worksheet row and “promoted”)

1: lower	upper	E (meV)	E n(E)	sig(E n)	sig(log10(En))
9: fit	10^fit		lambda	lamba E n(E)	Normalized sum
15: X	E(X)	F	F''	T1	T2
21: F'	F'''	nodes	more X	F(more X)	F''(more X)
27: E(more X)	function				

Energy0.xfm

```
jsv5D.,
; Initialize bin boundaries and compute logarithmic errors
;      24 Jul 2001
; rev. 28 Jun 2003 (added col 7, moved cell for n)
;
col(1,2) = sqrt(col(3,2) * col(3,1))
cell(1,1) = cell(1,2)^2 / cell(1,3)
n = size(col(1))
cell(8,6) = n
cell(1,n+1) = cell(1,n)^2 / cell(1,n-1)
col(2) = col(1,2)
col(6) = col(5) / col(4) / 2.302585
col(7) = ln(col(2)/col(1,1,n))
```

Energy1.xfm

```
jsv5D.,
; Generate cumulative distribution, and spline curve
;      29 Jul 2001
; rev. 28 Jun 2003 (dE in col(7), moved cell for nmax)
; rev. 23 Nov 2004 power-law instead of spline in lowest 0.1%
;
; Energy bins, in meV:
;   col(1) = lower, col(2) = upper, col(3) = geometric mean
; Data:
;   col(4) = neutrons/lethargy, col(5) = std.dev. neutrons/lethargy
; For error bars on log plot,
;   col(6) = col(5) / col(4) / ln(10)
; Widths of bins in lethargy units,
;   col(7) = ln(upper / lower)
; Coefficients of linear regression on log plot --> col(8)
;
; specify upper energy limit as a number of highest included bin
nmax = cell(8,6)
col(10) = col(4,1,nmax)
;
; copy part of regression line for low-energy fit
col(9,1,cell(8,5)) = cell(8,1) + log(col(3))*cell(8,2)
col(10) = 10.**col(9)
;
; determine normalization of (per lethargy) plot
cell(8,7) = total(col(10) * col(7))
;
; form lambda times E n(E), and Sum
col(12) = sqrt(81.8145/col(3,1,nmax))
col(13) = col(12)*col(10)*col(7)
; first part is exact integral of power-law fit
cell(8,4) = 10.**((cell(8,1)) * sqrt(81.8145) / (cell(8,2) - 0.5))
col(11) = cell(8,4) * 10.**((cell(8,2)-0.5) * log(col(1,1,cell(8,5)+1)))
col(14) = col(11) - cell(11,1)
; continue the sum with unsmoothed data
```

```

col(14,cell(8,5)+2,nmax+1) = cell(14,cell(8,5)+1) + sum(col(13,cell(8,5)+1))
; save normalization factor, and compute average lambda
cell(8,9) = cell(14,nmax+1)
cell(8,10) = cell(8,9) / cell(8,7)
; normalize the cumulative distribution
col(14) = col(14) / cell(8,9)
;
; interpolate specific points for spline nodes
nout = 33
cell(15,1) = 0.
cell(15,2) = 0.001
cell(15,(nout+1)/2) = 0.5
cell(15,nout) = 1.
cell(15,nout-1) = 0.999
for l=3 to nout/2 do
    cell(15,l) = 0.0182 * (1.25**(l-2) - 1)
    cell(15,nout-l+1) = 1. - cell(15,l)
end for
col(16) = interpolate(col(14), col(1,1,nmax+1), col(15))
col(17) = log(col(16))
;
; generate cubic spline curve for log(E) in col(17) vs. col(15),
; put y'' in col(18)
; temporary, x(i+1) - x(i) --> col(21),
; x(i+1) - x(i-1) --> col(22),
; (y(i+1) - y(i)) / (x(i+1) - x(i)) --> col(23)
col(21) = col(15,2) - col(15,1,nout-1)
col(22,2) = col(15,3) - col(15,1,nout-2)
col(23) = (col(17,2) - col(17,1,nout-1)) / col(21)
; lower boundary condition, at F=0.001
y2 = if(cell(20,4)>0., cell(20,4),
        cell(8,9)*0.43429/interpolate(col(3,1,nmax),col(13,1,nmax),cell(16,2)))
cell(20,4) = y2
cell(18,2) = -0.5
cell(19,2) = 3. / cell(21,2) * (cell(23,2) - y2)
; decomposition loop of tridiagonal algorithm
for o=3 to nout-1 do
    cell(20,1) = cell(21,o-1) / cell(22,o)
    cell(20,2) = cell(20,1)*cell(18,o-1) + 2.
    cell(18,o) = (cell(20,1) - 1.) / cell(20,2)
    cell(19,o) = (6.*(cell(23,o) - cell(23,o-1))/cell(22,o)
                - cell(20,1)*cell(19,o-1)) / cell(20,2)
end for
; upper boundary condition, at F=1
yn = if(cell(20,5)>0., cell(20,5),
        0.43429*cell(7,nmax)/(1. - cell(14,nmax)))
cell(20,5) = yn
cell(18,nout) = 0.5
cell(19,nout) = 3. / cell(21,nout-1) * (yn - cell(23,nout-1))
; backsubstitution loop
cell(18,nout) = (cell(19,nout) - cell(18,nout)*cell(19,nout-1))
                / (cell(18,nout)*cell(18,nout-1) + 1.)
for p=nout-1 to 2 step -1 do
    cell(18,p) = cell(18,p)*cell(18,p+1) + cell(19,p)
end for
cell(18,1) = 0.
;
; first & third derivatives of spline curve at nodes
cell(21,1) = "--"
for t=2 to nout-1 do
    cell(20,1) = cell(15,t+1) - cell(15,t)
    cell(20,2) = cell(20,1)**2 / 6.
    cell(21,t) = (cell(17,t+1) - cell(17,t) - cell(20,2)*(2.*cell(18,t)
                + cell(18,t+1)))/cell(20,1)
    cell(22,t) = (cell(18,t+1) - cell(18,t))/cell(20,1)
end for
cell(21,nout) = (cell(17,nout) - cell(17,nout-1))

```

```

        + cell(20,2)*(2.*cell(18,nout) + cell(18,nout-1))/cell(20,1)
col(19) = sqrt(81.8145/col(16))
cell(23,1) = cell(10,1)
col(23,2) = cell(8,9) / col(21,2) / col(19,2) / 2.302585
;
; spline interpolation between nodes
cell(24,1) = cell(15,1)
cell(24,11) = cell(15,2)
cell(25,11) = cell(17,2)
cell(26,11) = cell(21,2)
for u = 2 to nout-1 do
    cell(20,1) = cell(15,u+1) - cell(15,u)
    for v=1 to 9 do
        cell(20,2) = v/10. * cell(20,1)
        cell(24,10*u+v-9) = cell(15,u) + cell(20,2)
        cell(25,10*u+v-9) = cell(17,u) + cell(20,2)*(cell(21,u)
            + cell(20,2)*(cell(18,u)/2. + cell(20,2)*cell(22,u)/6.))
        cell(26,10*u+v-9) = cell(21,u)
            + cell(20,2)*(cell(18,u)+cell(20,2)*cell(22,u)/2.)
    end for
    cell(24,10*u+1) = cell(15,u+1)
    cell(25,10*u+1) = cell(17,u+1)
    cell(26,10*u+1) = cell(21,u+1)
end for
end for
;
; power law in lowest 0.1%
cell(20,1) = cell(8,2) - 0.5
cell(18,1) = 2.30259*cell(20,1)
cell(20,2) = 10.**(cell(17,1)*cell(20,1))
col(25,1,10) = data(cell(17,1), cell(17,2), (cell(17,2)-cell(17,1))/10.)
col(24,1,10) = cell(15,2)/(10.**(cell(17,2)*cell(20,1)) - cell(20,2))
    * (10.**(col(25,1,10)*cell(20,1)) - cell(20,2))
col(28,1,10) = cell(23,1)*(col(27,1,10)/cell(27,1))**cell(8,2)
;
; interpolated values for E, lambda, and I(lnE)
col(27) = 10.**col(25)
col(19) = sqrt(81.8145/col(27))
col(28,11) = cell(8,9) / col(26,11) / col(19,11) / 2.302585

```

Appendix C. Pulse-shape transforms and regressions for SigmaPlot®

Worksheet column headings (can be copied to a worksheet row and "promoted")

1: t(us)	E(meV)				
72: sum(n)	late t(us)	avg	sig(log)	first t	last t
78: t/lambda	early avg	sig log	(t/l) A(t/l)	Parameters	more params
84: Early fit	lambda (A)	R	switch	t*Data(case)	sig(log(data))
90: Best Fit	A(t/lambda)	t*B(t)	t*A(t)	Final Fit	t / lambda
96: X	deltaX	delta2X	T1	T2	(t/lambda)
102: A	Sum(A)	interp(A)	G	G2	deltaA/deltaX
108: spline(t/l)	spline(t/l)'	interp t/l	A(interp t/l)		(t)
114: B	Sum(B)	interp(B)	H	H2	deltaB/deltaX
120: spline(t)	spline(t)'	interp t	B(interp t)		

1_sumn.xfm

```

jsv5D.,
; Initialize and form sums of time slices
;   28 Jul 2001
;   17 Jul 2004, added col(126)=mu and col(128)=sigma
;   23 Aug 2004, added col(125)=Gini, moved sigma to col(127)
;   22 Nov 2004, use moments of log(t) instead of t, Gini-->col(128)
;
; col(1) and col(2) are geometric mean values of t(us) and E(meV) in bins
; col(3) - col(25) are input data (n/us) as functions of t for various E
; col(26) - col(48) are standard deviations (n/us) of the data
;
; compute relative time bin width
n = size(col(1))
nlogt = log(cell(1,1)/cell(1,n)) / (n - 1)
; since I know that Erik uses "bins per decade" I will round this number
dt = 10.**(1./round(1./nlogt,0)) - 1.
cell(82,4) = size(col(2))
cell(82,3) = 23
imax = min(col(82,3,4))
cell(82,3) = dt
cell(82,4) = imax
cell(82,9) = n
;
; also need to know ratio of lambda bin width to time
nlogE = log(cell(2,imax)/cell(2,1)) / (imax - 1)
cell(82,5) = round(nlogE / 2. / nlogt, 0)
;
; initialize some additional parameters
cell(82,10) = 0 ; no points used from late exponential
cell(82,11) = 0 ; no points used from early function fit
cell(82,12) = 0 ; no extrapolation of exponential
cell(82,20) = 1 ; longest wavelength for late average
cell(82,21) = 9 ; shortest wavelength for late average
;
; convert E (meV) to lambda (A)
col(85) = SQRT(81.8145 / col(2))
;
d = SQRT(1 + dt)
logt = log(col(1))
XX = logt*(d + 1/d)
X1 = logt*(2*d + 1/d)/3
for i=1 to imax do
; convert from n/us to n per bin, --> col(49)-col(71)
col(48+i) = col(2+i) * dt*col(1)
; sum of neutrons in each energy bin (column)
cell(72,i) = TOTAL(col(48+i))
; find first and second moments of log(time) for each energy
cell(126,i) = TOTAL(logt*col(48+i))/cell(72,i)
cell(127,i) = TOTAL(logt**2*col(48+i))/cell(72,i)

```

```

; find Gini statistic
cell(128,i) = TOTAL( 2*col(48+i)*(XX*SUM(col(48+i))-X1*col(48+i))/cell(72,i) -
                    XX*col(48+i) ) / cell(72,i)
end for
; means & standard deviations of time distributions
col(127) = sqrt(col(127) - col(126)**2)
col(126) = 10.**col(126)
col(127) = 2.30529*col(126)*col(127)
col(128) = -0.886227*2.30259*col(126)*col(128)

```

2_late.xfm

```

jsv5D.,
; Average the normalized pulse shapes for long wavelengths: B(t)
;   31 Jul 2001
;   17 Jul 2004, added mean & std.dev. of (t) --> cell(82,30),cell(83,30)
;   23 Aug 2004, added Gini --> cell(83,31)
;   22 Nov 2004, col(74) is t*B(t) instead of B(t), take moments of log(t)
;
; sum the "k2" latest time slices
k2 = cell(82,9)
; include appropriate lowest energy bins
j1 = cell(82,20)
j2 = cell(82,21)
nlate = j2 - j1 + 1
; uses the relative time bin width
dt = cell(82,3)
;
; initialize t (us) in col(73), sum (n) in col(74), variance in col(75),
;   count in col(81)
col(73) = col(1,1,k2)
col(74) = 0.*col(73)
col(75) = 0.*col(73)
col(81) = 0.*col(73)
;
; normalize time distribution for each energy bin to unit area, and sum
for i=j1 to j2 do
  for j=1 to k2 do
    if cell(25+i,j)>0 then
      cell(74,j) = cell(74,j) + cell(48+i,j)/cell(72,i)
      cell(75,j) = cell(75,j) + ( dt*cell(1,j)*cell(25+i,j)/cell(72,i) )^2
      ; tally number of non-zero cells
      cell(81,j) = cell(81,j) + 1.
    end if
  end for
end for
;
; average, change variance to sigma
col(74) = col(74) / nlate
col(75) = sqrt(col(75) / nlate / col(81))
; compute error of log10(data)
col(75) = col(75) / col(74) / 2.302585
;
; find mean & std.dev. of late-time average (us)
norm = TOTAL(col(74))
logt = log(col(73))
cell(82,30) = total(logt*col(74))/norm
cell(83,30) = total(logt**2*col(74))/norm
cell(83,30) = sqrt(cell(83,30) - cell(82,30)**2)
cell(82,30) = 10.**cell(82,30)
cell(83,30) = 2.30259*cell(82,30)*cell(83,30)
; Gini statistic
d = SQRT(1 + dt)
XX = logt*(d + 1/d)
X1 = logt*(2*d + 1/d)/3
cell(83,31) = -0.886227*TOTAL( 2*col(74)*(XX*SUM(col(74))-X1*col(74))/norm -

```

```

                                XX*col(74) ) / norm
cell(83,31) = 2.30259*cell(82,30)*cell(83,31)
; end-points of fits
;cell(128,1) = cell(83,31)
cell(129,1) = cell(82,30)
cell(130,1) = cell(83,30)
;
;convert from n/bin to n/lethargy
col(74) = col(74) / dt

```

3_avg_tl.xfm

```

jsv5D.,
; Average short-wavelength pulse shapes in t/lambda bins: A(t/lambda)
;   03 Aug 2001
;   17 Jul 2004, added mean & std.dev. of (t/lambda) --> cell(82,32),cell(83,32)
;   23 Aug 2004: added Gini --> cell(83,33)
;   09 Nov 2004, (t/lambda)*avg. in col(79), not col(81)
;
; uses relative time bin width and ratio of time/wavelength bins per decade
dt = cell(82,3)
imax = cell(82,4)
ratio = cell(82,5)
;
; determine shape of parallelogram to include all selected cells
; eqn. of upper edge is b + ratio*i
col(81) = ratio*data(1,imax)
b = min(col(76,1,imax) - col(81,1,imax))
cell(76,24) = b
; number of t/lambda cells is max height of parallelogram
k2 = max(col(77,1,imax) - col(81,1,imax)) - b + 1
cell(77,24) = k2
for k=1 to k2 do
    cell(78,k) = 0.
    cell(79,k) = 0.
    cell(80,k) = 0.
    cell(81,k) = 0.
end for
;
; include highest energies, count in cell(77,25)
cell(77,25) = 0.
for i=1 to imax do
    cell(77,25) = cell(77,25) + if(cell(76,i)>0, 1, 0)
    if cell(77,25)=1 then cell(77,26)=i end if
    ; for each included energy, limiting row numbers are in col(76) and col(77)
    for j=cell(76,i) to cell(77,i) do
        ;compute row number in output column, in cell(76,25)
        cell(76,25) = j - (b + ratio*i) + 1
        ;sum of normalized time distributions and errors (convert /us to /bin)
        cell(78,cell(76,25)) = cell(78,cell(76,25)) + cell(1,j) / cell(85,i)
        cell(79,cell(76,25)) = cell(79,cell(76,25)) + cell(48+i,j) / cell(72,i)
        cell(80,cell(76,25)) = cell(80,cell(76,25))
            + (dt*cell(1,j)*cell(25+i,j)/cell(72,i))^2
        ;count contributions
        cell(81,cell(76,25)) = cell(81,cell(76,25)) + 1
    end for
end for
;
; form average data, and sigma from variance
col(78) = col(78) / col(81)
col(79) = col(79) / col(81)
col(80) = sqrt(col(80)) / col(81) / col(79) / 2.302585
;
; find sigma of early-time average, us/A
norm = total(col(79))
logt1 = log(col(78))

```

```

cell(82,32) = total(logt1*col(79))/norm
cell(83,32) = total(logt1**2*col(79))/norm
cell(83,32) = sqrt(cell(83,32) - cell(82,32)**2)
cell(82,32) = 10.**cell(82,32)
cell(83,32) = 2.30259*cell(82,32)*cell(83,32)
; Gini statistic
d = SQRT(1 + dt)
XX = logt1*(d + 1/d)
X1 = logt1*(2*d + 1/d)/3
cell(83,33) = -0.886227*TOTAL( 2*col(79)*(XX*SUM(col(79))-X1*col(79))/norm -
                             XX*col(79) ) / norm
cell(83,33) = 2.30259*cell(82,32)*cell(83,33)

; end-points of fits
;cell(128,imax) = cell(83,33)*cell(85,imax)
cell(129,imax) = cell(82,32)*cell(85,imax)
cell(130,imax) = cell(83,32)*cell(85,imax)
;
; convert from per bin to per ln(t/lambda)
col(79) = col(79) / dt
;
; estimate switch function parameters based on cells included in averages
lambda1 = sqrt(col(85,cell(77,26))*col(85,cell(77,26)-1))
lambda2 = sqrt(col(85,cell(82,20))*col(85,cell(82,20)+1))
cell(83,17) = log(lambda2 / lambda1)
cell(83,16) = - log(lambda1)*cell(83,17)

```

4_i_c_exp (regression)

```

[Parameters]
b = 2.5      ; Ikeda-Carpenter parameter (us/A)
c = 20      ; late-time exponential (us/A)
R = 0.3     ; fraction in exponential
d = 0       ; initial delay (us/A)
a = 1       ; area of distribution

[Variables]
toverl = col(78)
intens = col(81)
wt = if(toverl<70., 1., 0.)

[Equations]
alpha = 1./(1/b - 1/c)
N0 = 0.5/b**3
N1 = (alpha/b)**3/c
x = if(toverl>d, toverl - d, 0.)
y = x/alpha
intfit = toverl*a*( (1.-R)*N0*x^2 * exp(-x/b)
                   + R*N1*exp(-x/c)*(1. - exp(-y)*(y^2/2.+y+1.)) )
fit intfit to intens with weight wt

[Constraints]
c > b
R < 1
R > 0
d > 0

```

5_spline.xfm

```

jsv5D.,
; Find cubic splines for A(t/lambda) and B(t)
; 06 Aug 2001
; 23 Nov 2004: input data per lethargy;
; power-law instead of spline in outermost intervals.
;

```

```

; uses relative time bin width, ratio of time bins to lambda bins
dt = cell(82,3)
lnw = 2.302585*log(1 + dt)
Emax = cell(82,4)
ratio = cell(82,5)
nt = cell(82,9)           ; number of "late" times
nt2 = nt + cell(82,12)   ; w/ extrapolation
jmax = cell(82,10)       ; earliest time for fitted late exponential
ntl = cell(77,24)        ; number of (t/lambda) bins
imin = ntl-cell(82,11)+1 ; latest t/lambda bin to use from Ikeda-Carpenter fit
cell(82,7) = " lambda # -->"
;
; col(79) is average of normalized distributions * t/lambda for shorter lambda
; col(74) is average of normalized distributions * t for longer wavelengths
; Note: both t and (t/lambda) bins are logarithmic, and spectra are per lethargy
;
; transfer (t/lambda)*A(t/lambda) to col(91), t*B(t) to col(92)
col(91) = col(79)
col(92) = 0.*col(1)
col(92) = col(74)
;
if imin>0 and imin<ntl then
    ;replace some early times with fit
    for ii=imin to ntl do
        cell(91,ii) = cell(84,ii)
    end for
end if
;
if jmax>1 then
    ;replace some late times with exponential fit
    for j=1 to jmax do
        cell(92,j) = cell(73,j)*10.**(cell(83,13)+cell(83,14)*cell(73,j))
    end for
end if
;
; transfer (t/lambda) bin boundaries to col(101) and
; col(91) to col(102) (in reverse order), sum in (103)
f = sqrt(1 + dt)
cell(101,1) = cell(78,ntl)/f
cell(103,1) = 0.
for m = 1 to ntl do
    cell(77,25) = ntl + 1 - m
    cell(101,m+1) = cell(78,cell(77,25))*f
    cell(102,m) = cell(91,cell(77,25))
    cell(103,m+1) = cell(103,m) + lnw*cell(102,m)
end for
cell(100,5) = cell(103,ntl+1)
col(103) = col(103) / cell(100,5)
;
; transfer t and B(t) to col(113), (114) in reverse order, sum in (115)
cell(113,1) = cell(73,nt)/f
cell(115,1) = 0.
for n=1 to nt do
    cell(77,25) = nt + 1 - n
    cell(113,n+1) = cell(73,cell(77,25))*f
    cell(114,n) = cell(92,cell(77,25))
    cell(115,n+1) = cell(115,n) + lnw*cell(114,n)
end for
if nt2>nt then
    ;extrapolate the exponential tail
    for nn=nt+1 to nt2 do
        cell(113,nn+1) = cell(113,nn) * (1. + dt)
        cell(114,nn) = cell(113,nn)*f*10.**(cell(83,13)+cell(83,14)*cell(113,nn))*f
        cell(115,nn+1) = cell(115,nn) + lnw*cell(114,nn)
    end for
end if
cell(100,6) = cell(115,nt2+1)

```

```

col(115) = col(115) / cell(100,6)
;
; interpolate specific points for spline nodes
nout = 33
cell(96,1) = 0.
cell(96,2) = 0.001
cell(96,(nout+1)/2) = 0.5
cell(96,nout-1) = 0.999
cell(96,nout) = 1.
for l=3 to nout/2 do
    cell(96,l) = 0.0182 * (1.25**(l-2) - 1)
    cell(96,nout-l+1) = 1. - cell(96,l)
end for
; save row differences and double differences from col(96)
col(97) = col(96,2) - col(96,1,nout-1)
col(98,2) = col(96,3) - col(96,1,nout-2)
; do the interpolations
cell(104,1) = cell(101,1)
col(104,2,nout-1) = interpolate(col(103), col(101), col(96,2,nout-1))
cell(104,nout) = cell(101,nt1+1)
col(105) = log(col(104))
cell(116,1) = cell(113,1)
col(116,2,nout-1) = interpolate(col(115), col(113), col(96,2,nout-1))
cell(116,nout) = cell(113,nt2+1)
col(117) = log(col(116))
;
; generate cubic spline curve for log(t/l) in col(105) vs. fraction in col(96),
; put y'' in col(106). NO spline in two outermost intervals.
col(106,1) = 0.
col(106,nout) = 0.
; save deltaA/deltaX
col(107) = (col(105,2) - col(105,1,nout-1))/col(97)
; may need t/l at bin centers
t1 = sqrt(col(101,1,nt1-1)*col(101,2))
; lower boundary condition
A1 = if(cell(100,8)>0, cell(100,8),
    0.43429/interpolate(t1,col(102),cell(104,2)))
cell(100,8) = A1
cell(106,2) = -0.5
cell(99,2) = 3. / cell(97,2) * (cell(107,2) - A1)
; decomposition loop of tridiagonal algorithm
for o=3 to nout-2 do
    cell(100,1) = cell(97,o-1) / cell(98,o)
    cell(100,2) = cell(100,1)*cell(106,o-1) + 2.
    cell(106,o) = (cell(100,1) - 1.)/cell(100,2)
    cell(99,o) = (6.*(cell(107,o) - cell(107,o-1))/cell(98,o)
        - cell(100,1)*cell(99,o-1))/cell(100,2)
end for
; upper boundary condition
An = if(cell(100,9)>0, cell(100,9),
    0.43429/interpolate(t1,col(102),cell(104,nout-1))*cell(100,5))
cell(100,9) = An
cell(106,nout-1) = 0.5
cell(99,nout-1) = 3. / cell(97,nout-2) * (An - cell(107,nout-2))
; backsubstitution loop
cell(106,nout-1) = (cell(99,nout-1)-cell(106,nout-1)*cell(99,nout-2))
    / (cell(106,nout-1)*cell(106,nout-2) + 1.)
for p=nout-2 to 2 step -1 do
    cell(106,p) = cell(106,p)*cell(106,p+1) + cell(99,p)
end for
;
; generate cubic spline curve for log(t) in col(117) vs. fraction in col(96),
; put y'' in col(118). NO spline in outermost intervals.
col(118,1) = 0.
col(118,nout) = 0.
; save deltaB/deltaX
col(119) = (col(117,2) - col(117,1,nout-1))/col(97)

```

```

; may need t at bin centers
t2 = sqrt(col(113,1,nt2-1)*col(113,2))
; lower boundary condition
B1 = if(cell(100,11)>0, cell(100,11),
        0.43429/interpolate(t2,col(114),cell(116,2)))
cell(100,11) = B1
cell(118,2) = -0.5
cell(99,2) = 3. / cell(97,1) * (cell(119,2) - B1)
; decomposition loop of tridiagonal algorithm
cell(100,22) = 0
for q=3 to nout-2 do
    cell(100,1) = cell(97,q-1) / cell(98,q)
    cell(100,2) = cell(100,1)*cell(118,q-1) + 2.
    cell(118,q) = (cell(100,1) - 1.) / cell(100,2)
    cell(99,q) = (6.*(cell(119,q) - cell(119,q-1))/cell(98,q)
                - cell(100,1)*cell(99,q-1))/cell(100,2)
end for
; upper boundary condition
Bn = if(cell(100,12)>0, cell(100,12),
        0.43429/interpolate(t2,col(114),cell(116,nout-1))*cell(100,6))
cell(100,12) = Bn
cell(118,nout-1) = 0.5
cell(99,nout-1) = 3. / cell(97,nout-2) * (Bn - cell(119,nout-2))
; backsubstitution loop
cell(118,nout-1) = (cell(99,nout-1) - cell(118,nout-1)*cell(99,nout-2))
                  / (cell(118,nout-1)*cell(118,nout-2) + 1.)
for r=nout-2 to 2 step -1 do
    cell(118,r) = cell(118,r)*cell(118,r+1) + cell(99,r)
end for
;
;interpolate 5 intermediate points between nodes
cell(99,1) = cell(96,1)
cell(99,2) = cell(96,1) + cell(97,1)/36.
cell(99,3) = cell(96,1) + cell(97,1)/9.
cell(99,4) = cell(96,1) + cell(97,1)/4.
cell(99,5) = cell(96,1) + cell(97,1)/2.25
cell(99,6) = cell(96,1) + cell(97,1)/1.44
for s=2 to nout-2 do
    cell(99,6*s-5) = cell(96,s)
    cell(99,6*s-4) = cell(96,s) + cell(97,s)/6.
    cell(99,6*s-3) = cell(96,s) + cell(97,s)/3.
    cell(99,6*s-2) = cell(96,s) + cell(97,s)/2.
    cell(99,6*s-1) = cell(96,s) + cell(97,s)/1.5
    cell(99,6*s)   = cell(96,s) + cell(97,s)/1.2
end for
cell(99,6*nout-11) = cell(96,nout-1)
cell(99,6*nout-10) = cell(96,nout) - cell(97,nout-1)/1.44
cell(99,6*nout-9)  = cell(96,nout) - cell(97,nout-1)/2.25
cell(99,6*nout-8)  = cell(96,nout) - cell(97,nout-1)/4.
cell(99,6*nout-7)  = cell(96,nout) - cell(97,nout-1)/9.
cell(99,6*nout-6)  = cell(96,nout) - cell(97,nout-1)/36.
cell(99,6*nout-5)  = cell(96,nout)
;
for t=2 to nout-2 do
    ;first derivative of (t/l) spline curve at node
    cell(100,1) = (cell(105,t+1) - cell(105,t))/cell(97,t)
                - cell(97,t)*(cell(106,t)/3. + cell(106,t+1)/6.)
    ;second derivative
    cell(100,2) = cell(106,t)
    ; third derivative between nodes
    cell(100,3) = (cell(106,t+1) - cell(106,t))/cell(97,t)
    ;interpolate 5 intermediate points between nodes
    cell(108,6*t-5) = cell(105,t)
    cell(109,6*t-5) = cell(100,1)
    for u=1 to 5 do
        cell(100,4) = cell(99,6*t+u-5) - cell(96,t)
        cell(108,6*t+u-5) = cell(105,t) + cell(100,4)*(cell(100,1)

```

```

                + cell(100,4)*(cell(100,2)/2. + cell(100,4)*cell(100,3)/6.)
cell(109,6*t+u-5) = cell(100,1) + cell(100,4)*(cell(100,2)
                + cell(100,4)*cell(100,3)/2.)
end for
;
;first derivative of (t) spline curve at node
cell(100,1) = (cell(117,t+1) - cell(117,t))/cell(97,t)
                - cell(97,t)/6.*(2.*cell(118,t) + cell(118,t+1))
;second derivative
cell(100,2) = cell(118,t)
;third derivative between nodes
cell(100,3) = (cell(118,t+1) - cell(118,t))/cell(97,t)
;interpolate 5 intermediate points between nodes
cell(120,6*t-5) = cell(117,t)
cell(121,6*t-5) = cell(100,1)
for v=1 to 5 do
    cell(100,4) = cell(99,6*t+v-5) - cell(96,t)
    cell(120,6*t+v-5) = cell(117,t) + cell(100,4)*(cell(100,1)
                + cell(100,4)*(cell(100,2)/2. + cell(100,4)*cell(100,3)/6.))
    cell(121,6*t+v-5) = cell(100,1) + cell(100,4)*(cell(100,2)
                + cell(100,4)*cell(100,3)/2.)
end for
end for
cell(108,6*nout-11) = cell(105,nout-1)
cell(109,6*nout-11) = (cell(105,nout-1) - cell(105,nout-2))/cell(97,nout-2)
                + cell(97,nout-2)/6.*(2.*cell(106,nout-1) + cell(106,nout-2))
cell(120,6*nout-11) = cell(117,nout-1)
cell(121,6*nout-11) = (cell(117,nout-1) - cell(117,nout-2))/cell(97,nout-2)
                + cell(97,nout-2)/6.*(2.*cell(118,nout-1) + cell(118,nout-2))
;
; convert log(t/lambda) to t/lambda and log(t) to t
col(110) = 10.**col(108)
col(122) = 10.**col(120)
; function values proportional to reciprocal of 1st derivatives of splines
col(111) = cell(100,5) / col(109) / 2.302585
col(123) = cell(100,6) / col(121) / 2.302585
;
;four power-laws: A0, An, B0, Bn. Iterate once, n-->col(100,1,4)
A2 = cell(111,7)/cell(100,5)
cell(100,1) = A2/cell(97,1)
cell(100,1) = A2/cell(97,1)*(1. - 10.**((cell(105,1)-cell(105,2))*cell(100,1)))
cell(106,1) = 2.30259*cell(100,1)
;
An1 = cell(111,6*nout-11)/cell(100,5)
cell(100,2) = An1/cell(97,nout-1)
cell(100,2) = An1/cell(97,nout-1)*(1. - 10.**((cell(105,nout-1)-
cell(105,nout))*cell(100,2)))
cell(106,nout) = -2.30259*cell(100,2)
;
B2 = cell(123,7)/cell(100,6)
cell(100,3) = B2/cell(97,1)
cell(100,3) = B2/cell(97,1)*(1. - 10.**((cell(117,1)-cell(117,2))*cell(100,3)))
cell(118,1) = 2.30259*cell(100,3)
;
Bn1 = cell(123,6*nout-11)/cell(100,6)
cell(100,4) = Bn1/cell(97,nout-1)
cell(100,4) = Bn1/cell(97,nout-1)*(1. - 10.**((cell(117,nout-1)-
cell(117,nout))*cell(100,4)))
cell(118,nout) = -2.30259*cell(100,4)
;
;interpolate the power laws
col(108,1,6) = cell(105,2) + log(1. - cell(100,1)/A2*(cell(97,1)-
col(99,1,6)))/cell(100,1)
col(110,1,6) = 10.**col(108,1,6)
col(111,1,6) = cell(111,7) - cell(100,1)*cell(100,5)*(cell(96,2)-col(99,1,6))
;

```

```

col(108,6*nout-10) = cell(105,nout-1) - log(1. + cell(100,2)/An1*(cell(96,nout-1)-
col(99,6*nout-10)))/cell(100,2)
col(110,6*nout-10) = 10.**col(108,6*nout-10)
col(111,6*nout-10) = cell(111,6*nout-11) + cell(100,2)*cell(100,5)*(cell(96,nout-1)-
col(99,6*nout-10))
;
col(120,1,6) = cell(117,2) + log(1. - cell(100,3)/B2*(cell(97,1)-
col(99,1,6)))/cell(100,3)
col(122,1,6) = 10.**col(120,1,6)
col(123,1,6) = cell(123,7) - cell(100,3)*cell(100,6)*(cell(97,1)-col(99,1,6))
;
col(120,6*nout-10) = cell(117,nout-1) - log(1. + cell(100,4)/Bn1*(cell(96,nout-1)-
col(99,6*nout-10)))/cell(100,4)
col(122,6*nout-10) = 10.**col(120,6*nout-10)
col(123,6*nout-10) = cell(123,6*nout-11) + cell(100,4)*cell(100,6)*(cell(96,nout-1)-
col(99,6*nout-10))

```

6_switch.xfm

```

jsv5D.,
; Evaluate switch function from regression parameters
; 05 Aug 2001
; 17 Jul 2004, added mean & std.dev. of t-->col(129),col(130)
; 22 Nov 2004, added Gini-->col(131)
;
; find min and max lambda for single (or first) ramp
min = 10.**(-cell(83,16) / cell(83,17))
max1 = 10.**((1.-cell(83,16)) / cell(83,17))
cell(82,23) = min
cell(82,24) = max1
cell(82,25) = max1
cell(83,23) = 0.
cell(83,24) = 1.
cell(83,25) = 1.
;
; now see if second ramp defined
if cell(83,29)>0. then
    max = 10.**((1.-cell(83,28)) / cell(83,29))
    knee = 10.**((cell(83,16)-cell(83,28))/(cell(83,29)-cell(83,17)))
    cell(82,24) = knee
    cell(83,24) = cell(83,16) + cell(83,17)*log(cell(82,24))
    cell(82,25) = max
    cell(82,26) = log(knee/min) / log(max/min)
end if
;
; evaluate at each wavelength
col(87) = if(col(85)<min, 0.,
            if(col(85)>cell(82,25), 1.,
              if(col(85)>cell(82,24),
                cell(83,28)+cell(83,29)*log(col(85)),
                cell(83,16)+cell(83,17)*log(col(85))))))
;
; linear combination of mu(t) & sigma(t) with switch function
col(129) = col(87)*cell(82,30) + (1.-col(87))*col(85)*cell(82,32)
col(130) = sqrt(col(87)*cell(83,30)**2 +
                (1.-col(87))*(col(85)*cell(83,32))**2 +
                col(87)*(1.-col(87))*(cell(82,30)-col(85)*cell(82,32))**2)
; Gini(t) combined same as standard deviation
col(131) = sqrt(col(87)*cell(83,31)**2 +
                (1.-col(87))*(col(85)*cell(83,33))**2 +
                col(87)*(1.-col(87))*(cell(82,30)-col(85)*cell(82,32))**2)

```

7_seefit.xfm

```

jsv5D.,
; Load specific columns with data for one wavelength

```

```

;    04 Aug 2001
;    22 Nov 2004 modified because cols(111) and (123) have changed;
;                compute Gini of log(t) instead of t, put in col(132)
;
case = cell(83,7)
dt = cell(82,3)
;
; move logarithmized data to col(88) with log10 error in col(89)
col(88) = col(1) * col(2+case) / cell(72,case)
col(89) = col(25+case)/col(2+case) / 2.302585
;
; illustrate fit in columns 92-95
; convert A(t/lambda) to A(t) for this lambda
col(95) = col(1)/cell(85,case)
col(93) = interpolate(col(110), col(111)/col(110), col(95))
; replace "missing" with "0", and change units to /us
col(93) = if(col(93)>0, col(93), 0)
col(93) = col(93)/cell(85,case)
; use interpolated B(t) from spline
col(92) = 0.*col(1)
col(92) = interpolate(col(122), col(123)/col(122), col(73))
;
; convert to logarithmic form, t I(t), and combine
; "best" à col(90), "final" à col(94)
col(92) = col(1) * col(92)
col(93) = col(1) * col(93)
col(90) = (1.-cell(86,case))*col(93) + cell(86,case)*col(92)
col(94) = (1.-cell(87,case))*col(93) + cell(87,case)*col(92)
;
; check normalizations
cell(83,9) = dt*total(col(88))
cell(83,10) = dt*total(col(93))
cell(83,11) = dt*total(col(92))
;
; Gini statistic for this fitted case
d = SQRT(1 + dt)
XX = log(col(1))*(d + 1/d)
X1 = log(col(1))*(2*d + 1/d)/3
norm = TOTAL(col(94))
cell(132,case) = -0.886227*TOTAL( 2*col(94)*(XX*SUM(col(94))-X1*col(94))/norm -
                                XX*col(94) ) / norm
cell(132,case) = 2.30259*cell(129,case)*cell(132,case)

```

8_ratio (regression)

[Parameters]

R = 0.5

[Variables]

A = col(93)

B = col(92)

data = col(88)

normA = cell(83,10)

normB = cell(83,11)

wt = if(col(88)>0., 1., 0.)

[Equations]

$N = (1 - R) * \text{normA} + R * \text{normB}$

$\text{calc} = ((1 - R) * A + R * B) / N$

fit calc to data with weight wt

[Constraints]

R > 0

R < 1