- **Event Mode Data**

Package version 7.40

05 JUL 2013 SRK

20 MAY 2016 SRK

Event mode data is neutron scattering data that is collected in a mode where every neutron event is recorded with its XY position on the detector and its arrival time. This results in a file that is a stream of events, each an encoded 32-bit binary word. Data in this format are essential for techniques such as TISANE, time-of-flight, time-sliced data such as oscillatory rheology data, kinetic processes, and more. This package allows processing of the event stream into any number of time bins, spaced as needed for the experimental conditions.

In general, there are two types of event data: Oscillatory or continuous (stream). For oscillatory data, the sample environment sends a "T0" signal to the detector to restart the clock at the beginning of each measurement cycle. Then the resulting event times range from 0 to t, repeating through the data collection. For continuous event data, there is no T0 signal and the time increases monotonically from 0 to the total collection time. Since these two types of data are processed slightly differently, be sure that the proper checkbox is selected to signify the data type.

See:

Event Mode Quick Start

Splitting Really Large Event Files

Types of Event Data

Correcting for things that go wrong

Setting up Custom Bin Widths

EventLoadWave XOP

Event Mode Quick Start

1) Load in the Reduction Macros

2) Choose Macros/Event Mode Processing

3) Select radio button for oscillatory or stream mode

4) Select the check box to remove bad events (this is highly recommended)

This option will do its best to remove:

-misencoded points after a rollover

-misencoded time points at the beginning of the file

-PP events that are XY (XY-time events are kept)

-Zero events

5) Determine the size of your event file as-is on disk. If it is larger than ≈ 150 MB, you'll need to split it up for processing. Skip to Splitting Really Large Event Files

6) If the file is small enough, click the Load Event Log file button (select the ".hst" file)

be patient, this is the slow step if the XOP is not present

7) Inspect the data, to be sure it's good.

7.1) Look at the results of the load in the information panel at the bottom. numXYevents is the number of neutron counts. PP is the number of "Pulse Pileup" events, ZeroData is the number of events that are, well, zero. Rollover is the number of 26-bit time encoding rollover events. Bad events are the number of events following a rollover that have mis-encoded time values.

7.2) After the data is loaded, two graphs will be drawn, one of the rescaled time vs. events, and one of the differentiated time vs. events. See below for the explanation of what these graphs might mean. The differential will show most of the problems, if any are present.

7.3) If something looks odd, Click on "Adjust Events" and proceed to:

Correcting for things that go wrong

8) Max time is filled in automatically, or you can change it.

9) Set the number of slices

10) Set the bin spacing from the popup (Custom is described later)

11) Bin Event Data - click the button

This is relatively quick and can be repeated with different binning without re-loading the data file.

12) Select the time slice to display

13) Click on "Show Bin Details". This brings up a bar graph of the number of events in each bin as a function of the bin width. There is also the tabulated data showing the numbers and the bin edges. Note that the single zero values at the opposite ends are for display purposes only and are not binning errors.

14) Export the slices as VAX data by:

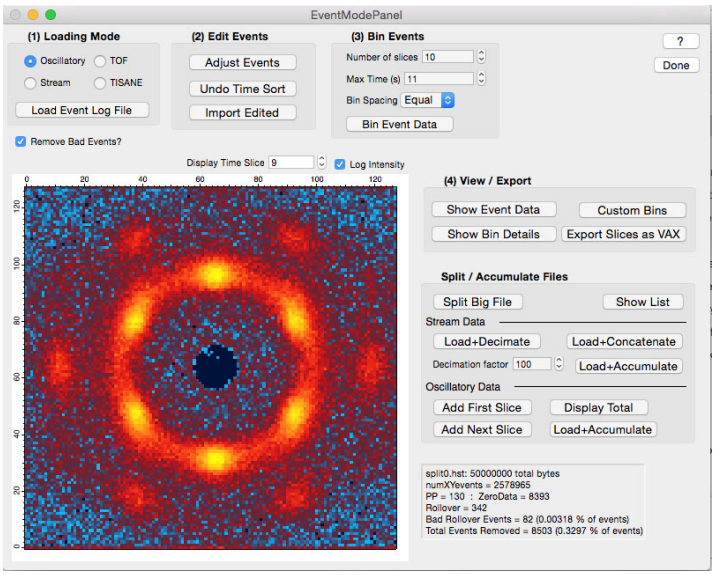
14.1) Load in the "full" VAX file with the good header information by using Display Raw Data from the main (yellow) control panel

14.2) Use the "Export Slices As VAX" button

14.3) Enter the starting "run number", and the 5-character prefix

14.4) You'll be asked if you've loaded the VAX file with the proper header, and given the chance to get out. If you haven't saved the Igor experiment, you'll be prompted to save the experiment, so the VAX files can be saved in the same place as the whole experiment.

The binned slices will be exported as separate VAX files, distributing the monitor counts, count time, and of course, counts as specified by the bins.



-- from the panel --

(1) Loading Mode

Using the radio buttons, select the "mode" of the data you have collected. This will set how the data is processed and presented. With the proper mode selected, click:

-- Load Event Log File: to load the XY neutron events versus time for processing. Select the raw".hst" event file. If you are attempting to load previously edited event data, skip to "Import Edited" in the next section.

** Currently, TOF and TISANE "modes" do nothing different than Oscillatory, but may be different in the future.

(2) Edit Events

These sections allow you to adjust the stream of the events, correcting for mis-encoded data. This can be a very slow step, but it's a very necessary step to ensure the quality of the final data.

-- Adjust Events: opens a panel to allow you to inspect all of the event versus time data and make "adjustments" to the data, correcting for mis-encoded times, noise, and other obvious instrumental artifacts. See Correcting for things that go wrong. The corrected data can then be saved to disk and re-loaded later (see Import Edited).

-- Undo Time Sort: will "un-sort" the data into the original timing of the events as they arrived. If the data is oscillatory, you'll see the oscillations again. Only the first 1500 events are shown, but you can change this by modifying the graph. But be careful if you try to plot everything - as there can be millions of points, making things slow and obscuring all of the oscillations.

-- Import Edited: will load in data that you have saved from the "Adjust Events" panel. This is saved in Igor text format, not the binary event stream. Once imported, proceed with binning.

(3) Bin Events

-- Number of Slices: enter the number of slices that you want to use. You can change this later after you

see what results you get. Using slice=1 is a valid choice and puts all of the events into a single bin.

-- Max Time: you can manually enter a Maximum time on the panel, and this will be the maximum time binned.

-- Bin Spacing: use this popup to select the type of bin spacing. Currently the spacing can be linear(equal), Fibonacci, or Custom.

-- Bin Event Data: processes the data based on the mode checkbox, number of slices, maximum time, and bin type.

(4) View / Export

-- Show Event Data: Displays the RescaledTimeGraph, showing the first 1500 points of the event file by default (for speed). It should be monotonically increasing. For oscillatory data, if you've processed the data and want to see the oscillations, you need to "un-do" the sorting. For either mode, it is very instructive to see all of the data, since there may be millions of events and the first 1500 are likely to look good. Ctrl-A or Cmd-A will show all of the data. Be patient as re-drawing all of the data is sometimes slow.

-- Show Bin Details: This brings up a bar graph of the number of event in each bin as a function of the bin width. There is also the tabulated data showing the numbers and the bin edges. Note that the single zero values at the opposite ends are for display purposes only.

-- Custom Bins: opens the panel to set up custome bin widths for the currently loaded data set

-- Export Slices As VAX: allows export of all of the current slices as "RAW" data files in the VAX binary format, suitable for use with the NCNR SANS reduction. The VAX files can also be then exported to different file formats. To export the slices, enter the starting "run number", and the 5-character prefix. Then, you'll be asked if you've loaded the VAX file with the proper header, and given the chance to get out. If you haven't saved the Igor experiment, you'll be prompted to save the experiment, so the VAX files can be saved in the same place as the whole experiment.

(5) Split / Accumulate Files

Splitting Really Large Event Files

This section is for the processing of data files that are too large to be loaded all at once into system memory. I don't know yet what the limit is on the physical file size that can be read in and processed. "Too big" appears to be files that are larger than about 200 MB -- generating an out-of-memory error. Currently, there is a global variable set at 150 (MB) as the default limit. If you'd like to increase the limit, execute the following at the command line to re-set the value:

root:Packages:NIST:Event:gEventFileTooLarge = 150

(As of mid-2016, Igor 7 and the 64-bit version are under beta testing, and could potentially allow loading of huge event files)

Currently, the options on the panel are:

-- Split Big File:

This command will read in a large event file and split it into smaller pieces. You will be prompted for the size of each chunk (in MB, 100 MB recommended) and the base name for the chunked files. A number and the ".hst" extension will be appended to each chunk. This function splits the raw (hst) files only.

-- Show List: After the data is loaded, a table will be presented with a list of files, as saved to disk. This is the list of files that will be loaded, in order, for any later processing. If you want to load a subset of the files or different ones, edit the entries in the table. There is a menu option:

SANS->Event Processing->Get List of ITX or Split Files:

which is a crude way to get a new (filtered) list of files to load based on the entered search string or extension. This is especially useful to get a list of the edited/saved splits.

Then the processing options for stream data and oscillatory data are slightly different. These are the suggested methods for processing the most commonly enountered scenarios and the flow of steps are detailed below. Your mileage may vary. If you have different needs for your experiment, please discuss this with your local contact or contact me directly.

(for stream data)

-- Load+Decimate: Using the decimation factor, the split files are decimated as they are loaded from the list, providing a manageable data set to work with.

-- Load+Concatenate: Loads a list of files (processed), concatenating the data and time values.

-- Load+Accumulate: The preferrred method, will load and bin the data as the list is loaded. The full data set is used, no decimation is done. Once done, the data is ready for immediate export.

(for oscillatory data)

-- Add First Slice: After the first split of a large file has been processed and sliced, this command moves the current set of slices into memory so that further splits can be loaded, processed, and added together.

-- Add Next Slice: Adds the current set of slices to the accumulated slices.

-- Display Total: Copies the accumulated slices back to be the displayed slices. Does not disrupt the accumulated slices, so more can be added. Useful for a check of the progress and this step is NECESSARY before exporting the slices as RAW data.

-- Load+Accumulate: The preferrred method, will load and bin the data as the list is loaded. As the list is processed, it automatically goes through the steps of loading, binning, and adding the slices. Finally, the total is displayed for immediate export.

If the data is Stream data, my suggestion is to do a "pre-screening" of the data set, to get an overview of the full time course of the data, which will allow you to decide on the binning scheme on a decimated version of the data. Then a second pass on the full data set is then much more easily accomplished.

To process stream data:

(1) Spit the file into chunks. You'll get a list of chunked files as saved to disk.

(2) Set the Decimation factor. 100 is a good choice.

(3) Click "Load From List". This will load each file in sequence, decimating the data as it is loaded.

(4) Proceed with normal processing of the decimated data, adjust the events, save the adjusted and decimated data. This will allow you to make decisions about the time behavior of the data, proper bin spacing, etc. on a data set that is representative of the full data set, but is very manageable in size.

Note that any export of the decimated data will be incorrect - since the export does not take the decimation factor into account (although this could be patched in the exported VAX files).

(5) Once decisions have been made about the proper binning for the data, *Write down the bin settings that you want for the final, full data set.* Now the tedious part begins. The task now is to work thorugh all of the split files (.hst), one-by-one, loading, adjusting, and saving each one. Be sure to clean up all of the garbage events in each. Be sure that they are saved with the ".itx" extension to mark them as edited event data.

(6) Using the menu option, get a list of these edited event files. The old list will be overwritten.

(7) With this new list of edited (.itx) files, choose "Load+Accumulate" (under the Stream Data section). The files will be loaded and binned as they are loaded, accumulating data inthe bins as the time increases. Once all of the files are loaded, the data is ready for immediate export. DO NOT click "Bin Event Data" -- or you will simply bin the last (partial) set of data that was loaded, and you'll have to repeat the Load+Accumulate.

If data is Oscillatory data, my suggestion is to split the data set, and during the process of correcting the errors in each of the splits, determine the binning that is best for your data. Write this down, you'll want it later.

To process oscillatory data:

(1) Split the event file into chunks.

(2) Go through the tedious part of editing all of the split files. The task now is to work thorugh all of the split files (.hst), one-by-one, loading, adjusting, and saving each one. Be sure to clean up all of the garbage events in each. Be sure that they are saved with the ".itx" extension to mark them as edited event data.

(3) With the last split file, determine the binning that you want. Set the values, and bin the data. This is only a partial set, and is simply to set the number of bins and bin timing.

(4) Now, using the menu option, get a list of these edited event files. The old list will be overwritten.

(5) With this new list of edited (.itx) files, choose "Load+Accumulate" (under the Oscillatory Data section). The files will be loaded and binned as they are loaded, accumulating data inthe bins as set. Once all of the files are loaded, the total of the slices is displayed and the data is ready for immediate export. DO NOT click "Bin Event Data" -- or you will simply bin the last (partial) set of data that was loaded, and you'll have to repeat the Load+Accumulate.

These commands are to assist in accumulating the slices internally, so that large numbers of exported files don't need to be added together later. But both ways are equivalent. In either scheme, The processing must be identical. Same time bins, same maximum time, same number of bins. Otherwise it doesn't make any sense.

This file splitting is equivalent to the unix command:

split -b100m Event20121119170652.hst e0652

where in this example, -b100m sets the pieces to be approximately 100 MB each, and named "e0652a", "e0652b", "e0652c", etc. as needed. See 'man split' for more details.

Each of the split files can be processed in the same way - using the same binning, and exporting to

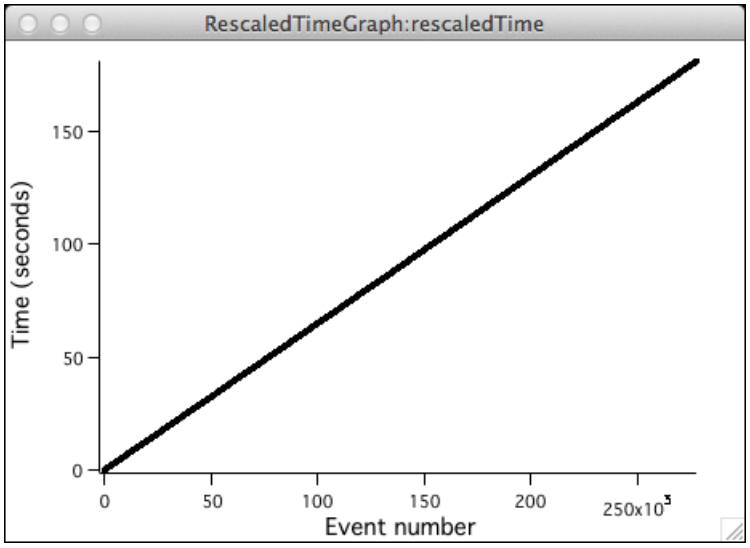
VAX format. Then the data files for each bin can be summed up for the data reduction.

-- Using the unix command can mangle the split by cutting apart a single event, or splitting XY-time events. Just clean up the ends, throwing away a few data points.

Types of Event Data

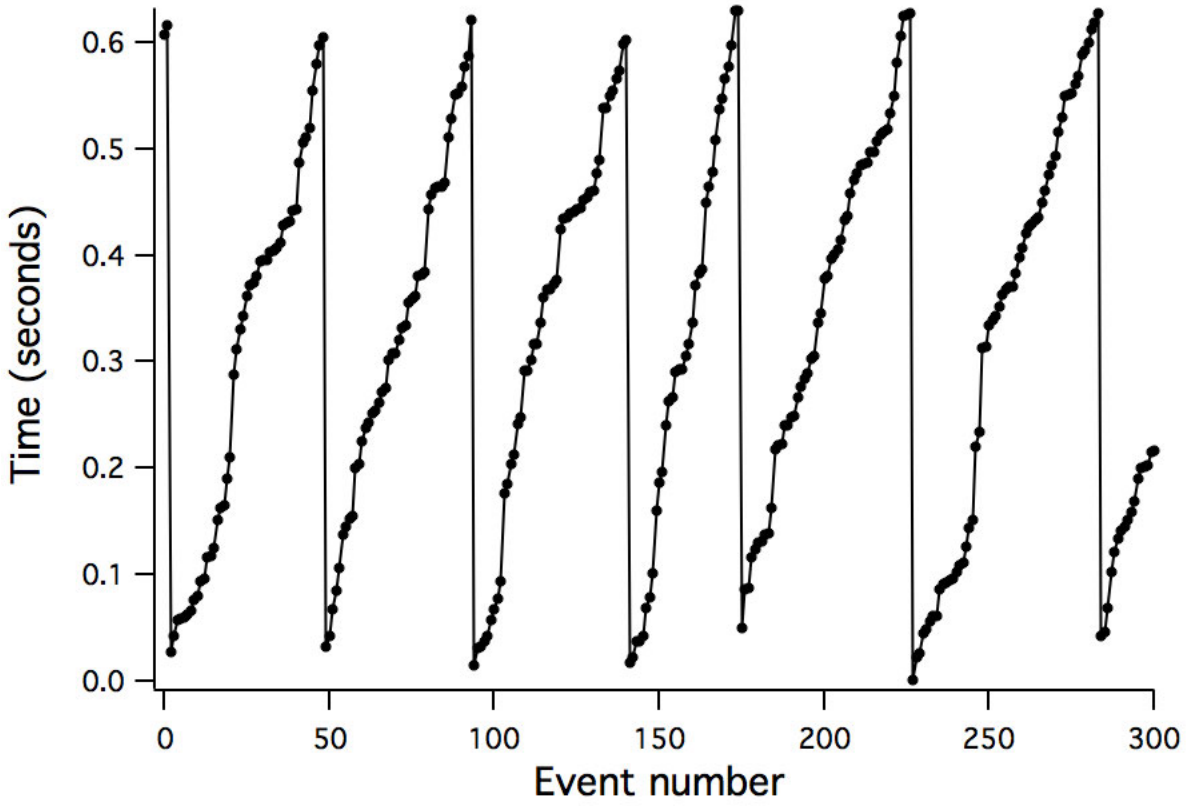
-- Stream data details:

This is a "stream" of data events where no T0 signal is sent. A possible application of this file type is a single, long relaxation process over several minutes or longer. The full time is collected in a single file, and then sliced into time bins as appropriate for the kinetics of the process.

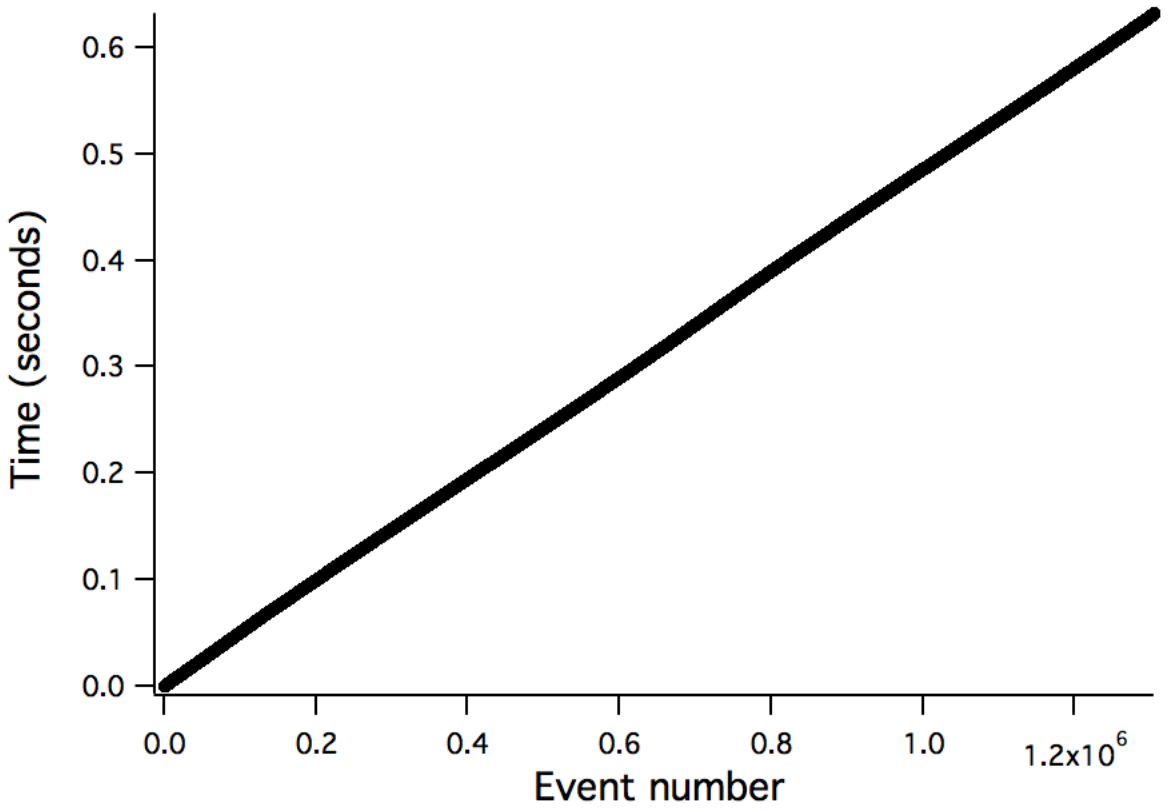


-- Oscillatory data details:

At the T0 signal, the clock is reset on the hardware side, so there is nothing to interpret when reading the data (as long as the period of the oscillation is less than 6.7s. In this example, it's about 0.6s. So time only ever reads a value between zero and 0.6s. In the graph, only the first few hundred events are shown, to show the sawtooth nature of the data. So upon loading, the data looks like:



Then in processing, the data is sorted by time to collapse all of the XY events into a single time "frame" of 0s -> 0.6s. Then it's easily binned from here. It can also be "un-sorted" to return to the original event stream. After processing, the rescaled time graph looks like: (and this is now all 1,300,000+ events)

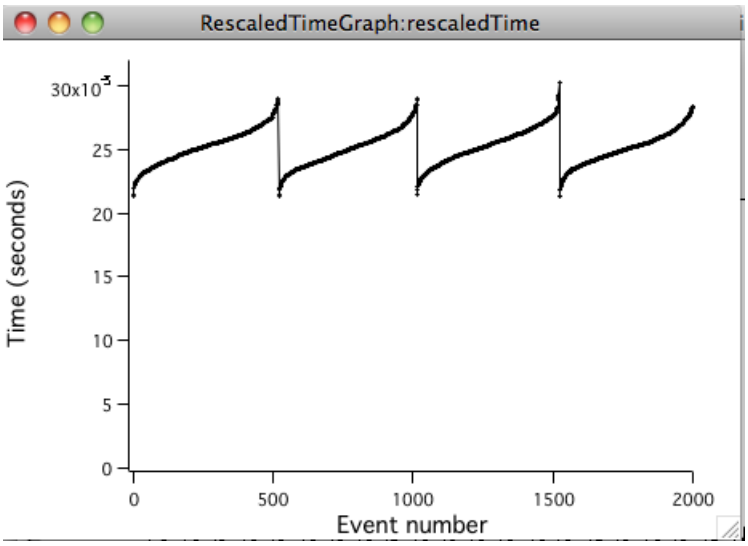


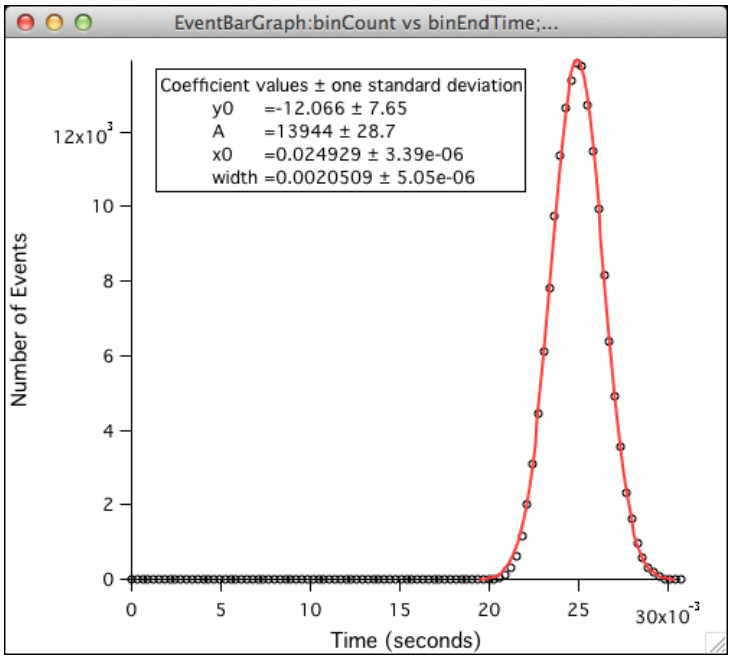
-- TISANE data details:

Details and an example to be added.

-- Time-of-flight data details:

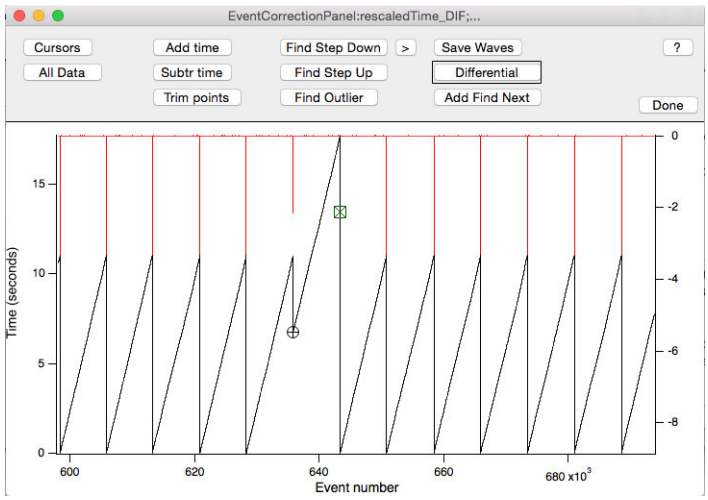
An example of TOF data is shown below. I don't know the actual setup here, but there is a chopper in the beam, chopping the beam into non-overlapping pulses. The event stream looks like below with regular oscillations. Note that the mean time for an event is around 25 ms and is in a narrow band, set well away from zero. Note also the sigmoidal shape of the event arrival times in each pulse. When histogrammed (following graph), the result is a gaussian(ish) distribution of arrivals. This example file may be a TOF measurement of the wavelength distribution, which could be calculated if the sample to detector distance were known.





Correcting for things that go wrong

Not all of the "bad" time events can be removed or even detected on loading. To remove bad events, or adjust sections of mis-encoded time, use the Event Correction Panel:



This panel allows you to interactively adjust or trim data points from the file. The rescaled time is plotted versus the event number. Clicking "All Data" will show the entire wave.

-- Click on "Cursors" to add cursors (a circle and a square) to the data.

-- Click on "Differential" to add the derivative to the plot. This aids greatly in locating the "bad" spots in the data.

Move (drag) the cursors where you want them - to the beginning and end (inclusive) of points that need to be adjusted or trimmed.

The panel buttons are:

-Cursors: adds cursors to the graph for selection of data points and ranges

-All Data: shows the full data set of time and events

-Add Time: adds 2^26*10^-7 s = 6.7s = one rollover to the selected points

-Subtr Time: subtracts 6.7s = one rollover from the selected points

-Trim Points: trims (deletes) the selected points from the data set

-Find Step Down: finds a step down in the data, and attempts to place the cursors at the beginning of the step and the last data point. If good, follow with "Add Time", then "Differential" (to be able to find the next step)

-Find Step Up: finds a step up in the data, and attempts to place the cursors at the beginning of the step and the last data point. If good, follow with "Subtr Time", then "Differential" (to be able to find the next step)

-Find Outlier: finds the single data point that is the largest deviation from the average of the whole data set. Puts both cursors there. If good, follow with "Trim Points" and that single point will be removed.

- >: This little button nudges the cursor a bit to the right after you have corrected a bad spot in the data set. This keeps the search from "re-finding" the already-correct spot again. Also useful for bypassing a small blip in the data that is flagged with a bad (enough) derivative.

-Save Waves: saves the modified data for later import. The modified data must be imported through this panel, not loaded from the main event mode panel

-Differential: Recalculates the derivative. This is not automatically done, since it can be time

consuming. Recalculate this after every adjustment to the event data.

-Add Find Next: adds 2^26*10^-7 s = 6.7s = one rollover to the selected points, recalculates the derivative, and moves the cursor to the next step down.

Adding or subtracting 6.7s is the only time option currently. It is a result of the time being misencoded, and can be corrrected. If you find a time step of other than 6.7s, then something else is going on with your data that merits a deeper investigation.

If the "steps" that are being located are the little ones, not the big ones of 6.7s, you can change the tolerance from the command line. Also, remember to re-calculate the differential after adjusting the event data. Steps are located from the differential of the event data. The default step tolerance is set for 5 (standard deviations from the average derivative value). Execute the following command to reset the global variable. Try 10 (standard deviations) to exclude some of the smaller steps.

root:Packages:NIST:Event:gStepTolerance = 10

NOTES - adjusting events

Zoom and shrink as needed:

- drag a marquee and right-click (horiz expand) (repeat)

- double-click on the bottom axis and on the "Axis Range" tab, enter the min and max values (you may want to make sure that the "Live update" checkbox is NOT checked).

- "All Data" will show you all of the data again.

Move cursors faster:

- click and drag them. If you place them on an "edge" they'll and up at one end of the step or the other.

- if a cursor is "solid" in the information bar at the bottom of the graph, if can be moved with the arrow keys, or the "slider". if both are "solid", both cursors move in tandem. To make a cursor not solid - and then not move with the arrows. click on its icon in the status bar at the bottom of the graph to toggle its solid/open state.

Read out where the cursors are:

- the status bar at the bottom of the graph shows where the cursors are, reporting the point number, x and y values, and also the delta(x) and delta(y). when there is a step in the time, set the cursors at each edge. Then the delta(y) should be the magical 6.7 seconds. If it's not, then something else is going on. The answer then is to trim the data from the set.

There is NO UNDO

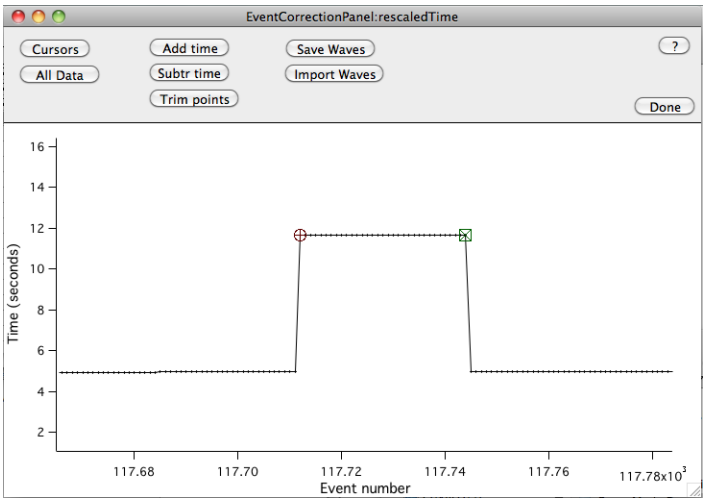
Changing the time values is reversible - just do the opposite operation before moving the cursors. Trimming the points is not reversible. Once they're gone, they're gone.

- Any Adding, Subtracting time, or Trimming of data points will automatically update the maximum time for the data set. So after adjusting the data - and Saving it! - you can bin the data.

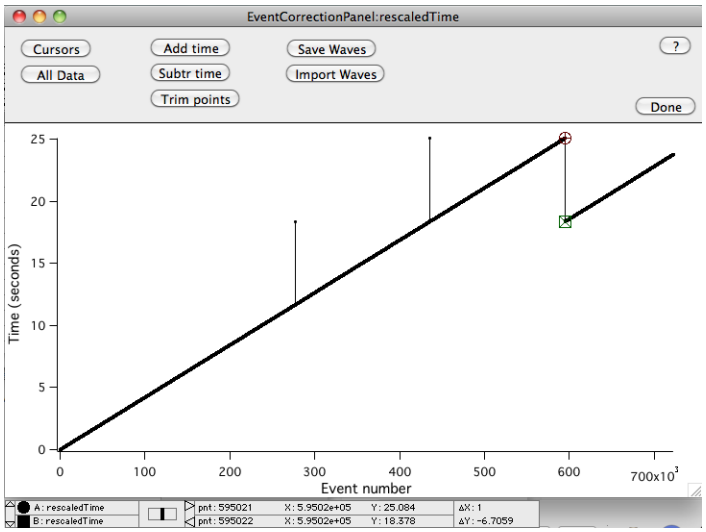
- Once the data has been "retouched" to where you are happy with it, you can save the waves for later recall. The waves are saved in Igor text format, not the event format (they are relatively equivalent in the disk space that they take up) and are much more logical to read. "Import Waves" will re-read in the saved waves as if they were just loaded, and are ready to process.

- Be sure that you are adjusting and saving un-sorted data.

- For example, subtract the time from here to correct the time in the bad "spike" (highly magnified):



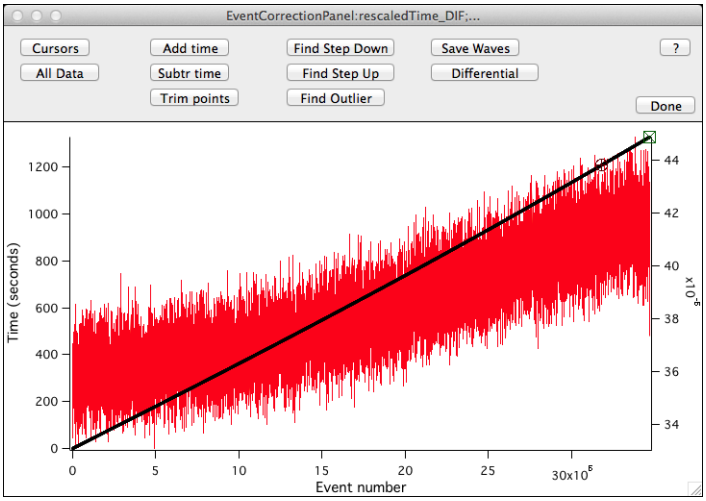
The step shown below is a delta(t) of 6.7059s. Add time from the step to the end of the set to fix the problem, which was (maybe) a rollover that was never written to the event file (but the bits were cleared). Then trim or adjust the remaining two spikes, and the set is good to process. Save it first, of course.



** You'll find out very quickly that the fewer data points that you have displayed, the faster the graph updates.

** If the rescaledTime graph is also open, it will update also as you modify the data set, increasing (doubling) the time it takes for anything to redraw. So close any unnecessary graphs. You can always re-open them later if you like.

An example of a final, touched-up data set looks like this:



On close inspection of the event time versus event number, which looks linear, it is not (it actually is curving up). The differential highlights this better. The differential shows the usual noise of counting, but with an upward trend. This indicates that delta(t) between event arrivals is increasing, that is - the countrate is decreasing. A decreasing differential indicates an increasing count rate. This changing rate (for stream data) will be evident in the binned data as well.

Setting up Custom Bin Widths

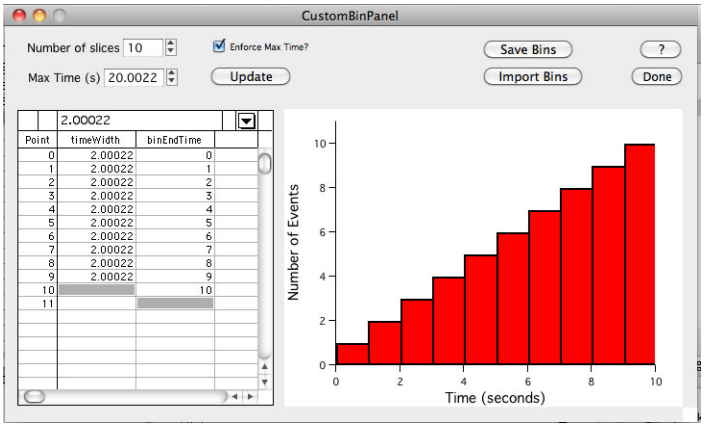
Currently, there are three choices for the bin spacing: Equal, Fibonacci, and Custom.

Equal - splits the time from 0->t_max into equal bin widths.

Fibonacci - sets the bins from 0->t_max based on the Fibonacci series (neglecting the first zero)

Custom - lets you set the number of bins and the width of each bin. Details are below.

When you select "Custom" from the popup, a new panel is presented:

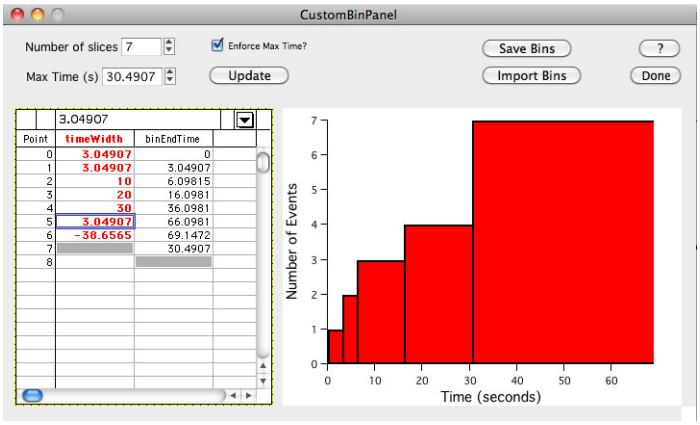


On this panel you can set the number of slices and the maximum time to use. These are the same values that are on the main event loader panel. The table contains two columns. timeWidth is the width of each time bin in seconds. binEndTime is the ending time of each bin, also in seconds. There is one extra point in the binEndTime column. The first point here is zero, simply for display in the bar graph. Therefore the delta(t) for bin[p] is (binEndTime[p+1]-binEndTime[p]). A bar graph of the bin widths is also shown, with dummy values for the number of events. (the # of events = the bin number)

Now you can simply enter the number of bins (slices) that you want. The length of the waves in the table automatically adjusts. Next, be sure to enter a timeWidth for all of the bins.

--Then click "Update" to recalculate the binEndTime and update the bar graph.

By default, bins of equal widths are filled in, and the maximum time is enforced. What this check box means is that the calculation of the binEndTime will use the entered timeWidths up until the last one - then it will CHANGE what you have entered to make the delta(t) work out so that the last bin ends at the maximum time. Note that this WILL CHANGE the value that you entered, and if the total of the bins exceeds the maximum time, then the last bin width will be negative - which is bad - and the column will turn red and bold to warn you to fix the timing.

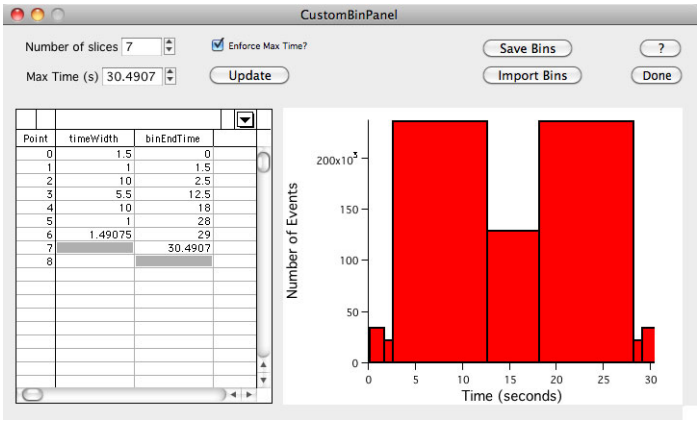


You must click "Update" to recalculate every time you make a change. It doesn't update automatically.

Now, when you process the data (and Custom is still selected in the Bin Spacing popup), the bins that you made will be used, and the bar graph of bins will now reflect the actual number of counts.

You can save the bins that you generated by clicking "Save Bins". This will save and Igor text file that

you can read back in using "Import Bins"



- **EventLoadWave XOP**

EventLoadWave /A[=*baseName*]/D/N[=*baseName*]/O /P=*pathName* /Q /I /R /W *fileNameStr*

EventLoadWave is an Igor operation that loads in event files (.hst) as generated by NISTO at the NCNR SANS instruments. The EventLoadWave operation loads data from the named event file into waves. EventLoadWave is intended to be used for good. Never for evil purposes.

Parameters

If *fileNameStr* is omitted or is "" or the /I flag is used, EventLoadWave displays an open file dialog. It also displays the dialog if the /P=*pathName* and *fileNameStr* parameters do not fully specify the file to be loaded.

If you use a full or partial path for *fileNameStr*, see Path Separators for details on forming the path.

Flags

/N=*baseName*	Automatically assigns wave names of the form *baseName* 0, *baseName*1. Overwrites existing waves. *This is coded as *EventWave* as the base name. Do not change this in the code.*
/O	Overwrite existing waves in case of a name conflict. This currently does not seem to work correctly - so the /N flag is used instead
/P=*pathName*	Specifies folder to look in for *fileNameStr.*
/I	Specifies interactive mode — you get the open file dialog.
/Q	Suppresses the normal messages in the history area.
/R	Tells the loader to remove the "bad" events whenever it can. Without this flag, all of the events are read in.
/W	Tells EventLoadWave to only "pre-scan" the file. This simply does

a count of the various types of events, and sets the output variables.

// don't use these flags... you'll get odd behavior

/A	Automatically assigns arbitrary wave names. Skips names already in use.
/A=*baseName*	Automatically assigns wave names of the form *baseName* 0, *baseName*1. Skips names already in use.
/D	Creates double precision waves. This is ignored by the XOP
/N	Automatically assigns arbitrary wave names. Overwrites existing waves.

Details

The /N flag instructs Igor to automatically name new waves “wave” (or *baseName* if /N=*baseName* is used) plus a digit. The digit starts from zero and increments by one for each wave loaded from the file. If the resulting name conflicts with an existing wave, the existing wave is overwritten.

The /A flag is like /N except that Igor skips names already in use.

The call to the function to load and remove bad points should be:

EventLoadWave/R/N=EventWave fileNameStr

EventLoadWave sets the standard Igor file-loader output variables:

S_path Set to the path leading to the folder containing the loaded file. This is a system file path (e.g., "hd:FolderA:FolderB:"), not an Igor symbolic path. The path uses Macintosh path syntax, even on Windows, and has a trailing colon.

S_fileName Set to the name of the loaded file.

V_flag Set to the number of waves loaded.

S_waveNames Set to a semicolon-separated list of the loaded waves.

It also generates the following output variables:

V_num0 Set to the number of type 0 events (= XY).

V_num1 Set to the number of type 1 events (= Minor rollover).

V_num2 Set to the number of type 2 events (= XY-Time).

V_num3 Set to the number of type 3 events (= Major rollover).

V_numBad Set to the number of points following a major rollover that have mis-encoded time values, and are discarded if the /R flag is set.

V_numPPSet to the number of type PP (Pulse pileup) events. Currently, if a type 0 event is also PP, it is discarded (if /R set). Type 2 events that are PP are not discarded.

V_numRemoved Set to the number events that have been discarded (if /R set).

V_numT0 Set to the number of T0 (trigger) events

V_numZero Set to the number of events that are completely zero (all bits). These are discarded if /R is set.

V_nXYevents Set to the number of XY events (= type 0 + type 2)

When called from a procedure, these are created as local variables. When called from the command line, it creates global variables, which are then visible in the data browser.

NOTE

1) To avoid a proliferation of LARGE waves - the operation is called with the /N=EventWave flag, so that the waves generated are ALWAYS EventWave0, EventWave1, EventWave2. The waves are then renamed xLoc, yLoc, and timePt. The X and Y waves are generated as SP and the timePt is DP. Byte waves would save space for the X and Y locations, but all integer waves must be internally converted to FP for display or analysis, so SP is used instead.

2) In the XOP, the time is calculated in double precision. This is to accommodate long collection times. Unsigned integers are 32 bits. Therefore the longest time that can be represented is 2^32=4,294,967,296 ticks, where each tick is 0.1 microsecond. So the maximum time represented as an integer is 429 seconds. In the Igor code, all variables and calculations are done in double precision. In c, these type conversions must be explicitly done. Double precision is necessary since only about 7 digits of precision can be represented in SP (32-bit) while about 17 digits can be represented in DP (64-bit) values.

Revision Notes

1.51	First pass. Equivalent to the functionality in NCNR package version 7.13d

---