OVERVIEW

Slam Stick Viewer is a GUI application for viewing and exporting data from the Slam Stick™ portable vibration recorder and configuring the recorder. It's written in the Python programming language, and can be used either stand-alone, as part of your own data analysis s scripts, or interactively with SciPy's IPython shell.

A prepackaged Windows® executable is also provided for convenience. If you've never heard of Python, SciPy, etc. or don't wish to install them, this is the easiest way to get up and running. For users of other operating systems, please follow the instructions under 'Source Code' below.

INSTALLING

Win32 Executable

To install, just double-click on the installer and follow the instructions. The executable is simply a copy of the compiled Python scripts and libraries packaged together using the py2exe (http://www.py2exe.org/) utility. In the installer, select 'Source Code (Python)' if you would like a copy of the original Python scripts too. A Start Menu shortcut and uninstaller will be created automatically.

Source Code (Python)

If you already use Python (or wish to start!), need to modify the standard UI (e.g. to include custom plot types), or embed Slam Stick reading functionality in your own analysis scripts, installing the source code and packages is recommended. You will need:

- Python (http://python.org/)
- The NumPy package (http://numpy.scipy.org/)
- The Matplotlib package (http://matplotlib.sourceforge.net/)
- Slam Stick Viewer (source) (www.mide.com/products/slamstick/slamstick_vr001_software.php)

Note: If using the pre-built packages, be sure to choose a compatible version for each package (it may not necessarily be the latest version of each). At the time of this writing (10/18/2010), pre-built matplotlib packages are only available for Python 2.6 or earlier.

USING

File Menu

Slam Stick Viewer can open the following types of vibration data:

- Slam Stick recordings (native fomat produced by Slam Stick; normally 'data.dat')
- "Split" Slam Stick recordings (for compatibility with other Slam Stick tools, e.g. MATLAB scripts, which may split up a multi-recording Slam Stick file into individual recording files)
- Comma-Separated Value (.CSV) vibration data generated from Slam Stick or other sources. Refer to page 7 for formatting.

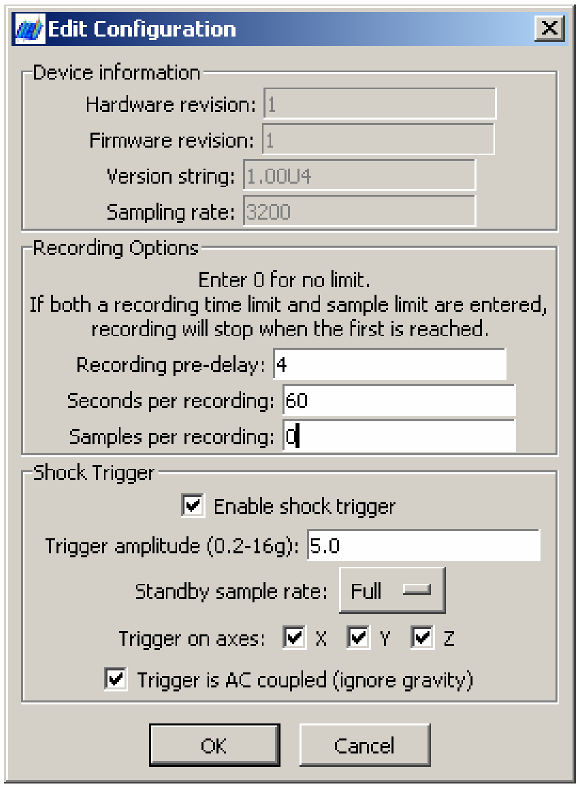
Files may be read from most any file source, including local/network drives or directly from the Slam Stick. Please note that reading from the Slam Stick is much slower than reading from a local hard drive. Additionally, depending on the speed of your computer and the size of the recording being analyzed, analysis may take a long time (up to several minutes) to complete. The GUI will appear unresponsive during this time.

Slam Stick Viewer can also export the recording (time history and/or FFT) as a .CSV file for use with other programs. Refer to page 7 for formatting.

Configure Menu

Use the Configure menu to configure the Slam Stick. The recommended approach is to select "Edit Configuration" and point to the configuration file (config.dat) of a Slam Stick that is currently plugged in. You can also create a new Slam Stick configuration file and save it to disk for later use. Figure 1 shows the configuration options. Refer to the Slam Stick manual for a detailed description of the configuration options available.

Note: Changing the Slam Stick configuration (or copying a changed configuration onto the Slam Stick) will cause an automatic erase to be performed when the Slam Stick is unplugged. Please save any data you wish to keep before doing so!


Edit Configuration

Device information
Hardware revision: 1
Firmware revision: 1
Version string: 1.00U4
Sampling rate: 3200

Recording Options
Enter 0 for no limit.
If both a recording time limit and sample limit are entered, recording will stop when the first is reached.
Recording pre-delay: 4
Seconds per recording: 60
Samples per recording: 0

Shock Trigger
[✓] Enable shock trigger
Trigger amplitude (0.2-16g): 5.0
Standby sample rate: Full
Trigger on axes: [✓] X [✓] Y [✓] Z
[✓] Trigger is AC coupled (ignore gravity)

Figure 2: Slam Stick configuration options

About Menu

This menu shows the Slam Stick Viewer version and release date, license and copyright info.

WORKING WITH PLOTS

Plot Toolbar

The Plot Toolbar provides common options for all plot types. Figure 3 shows the available options.

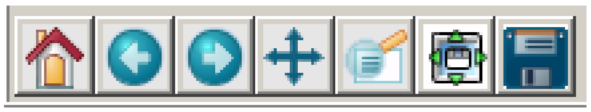


Figure 3: Plot Toolbar

The toolbar buttons, from left to right, do the following:

Home

This button returns the plot to its original view, adjusting the axes so that the entire plot is shown.

Back/Forward

If the currently displayed plot has been panned, zoomed, or otherwise adjusted, these buttons can be used to return to the previous view, or forward to the most recent view (if available).

Pan and zoom

With this control selected, click and hold with the left mouse button on the plot to pan (drag) the current view. Click and drag with the right mouse button to change the vertical and horizontal scale.

Zoom Rect

Click and drag to select a rectangular area of the plot to zoom in on.

Plot parameters

This button produces a dialog window where certain plot display parameters (plot size relative to main window, whitespace) can be adjusted. This may be useful to export an image of the plot at a specific aspect ratio.

Save

This button can be used to save the currently displayed plot as an image. Available image formats include .PNG, .EMF, Postscript (.PS, .EPS), .PDF, .SVG and raw (RGBA). Note the specific save-as formats available may vary depending on your operating system.

Plot Types and Plot-Specific Options

Use the tabs on the left hand side to switch between different plot types.

Time History

The Time History plot shows acceleration in Gs as a function of time for each axis. Use the tickboxes to the right of the plot to show/hide individual axes.

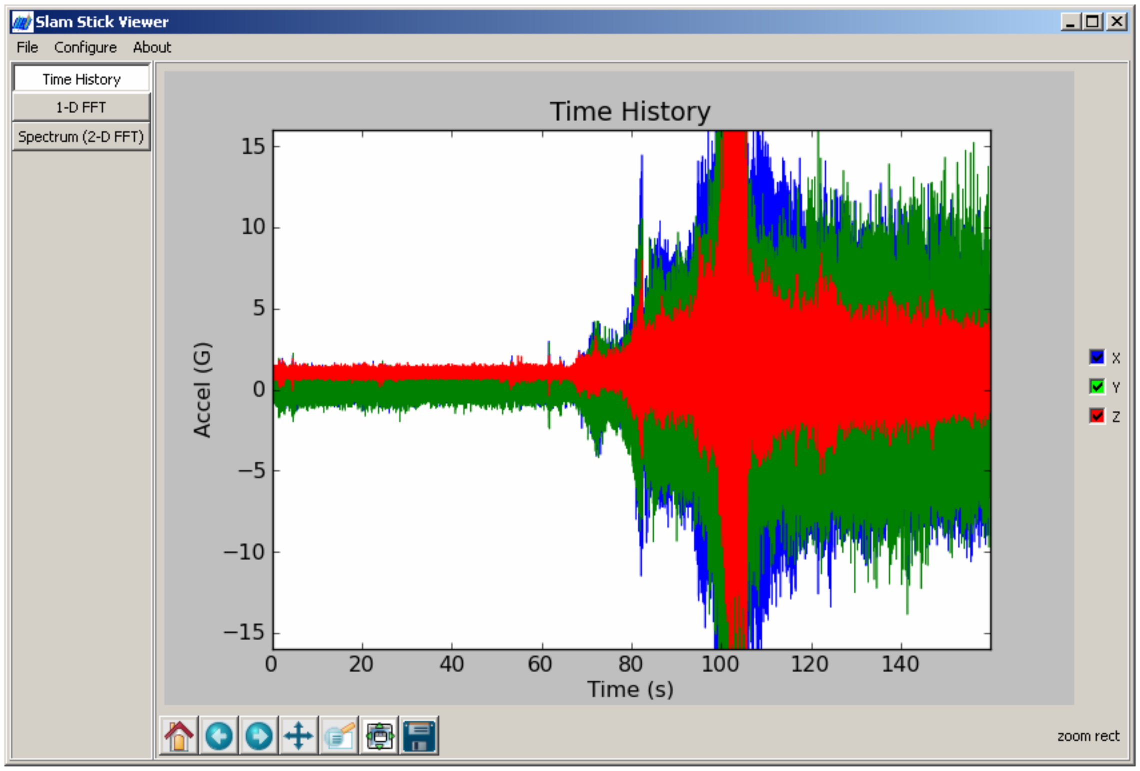


Figure 4: Time History view

1-D FFT

The 1-D FFT shows the frequency distribution in Gs (approximate) over the entire recording. This view provides the highest frequency resolution, and is best suited for recordings whose frequency content does not significantly change over time. Like the time history plot, use the tickboxes at right to show/hide individual axes. The data are shown in linear scale by default. Use the 'Log X' and 'Log Y' tickboxes to switch between linear and logarithmic representation for each axis.

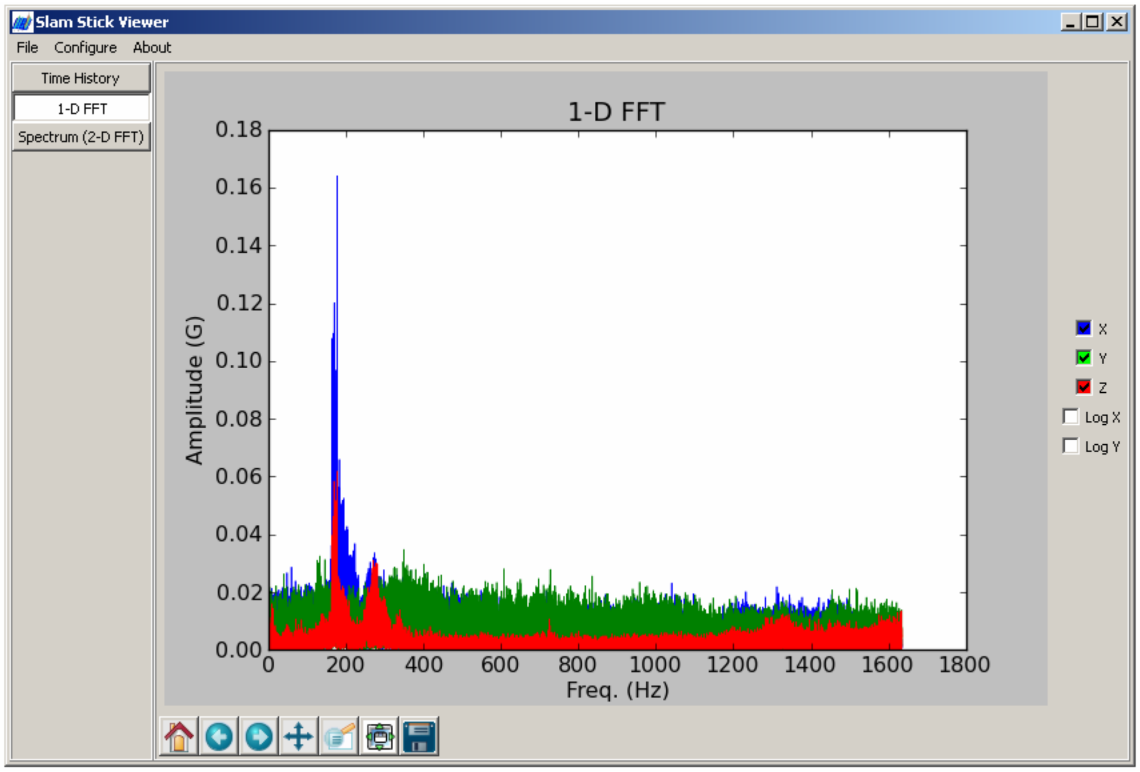


Figure 5: 1-D FFT view

Spectrogram (2-D FFT)

The Spectrogram plot shows the distribution of frequency content over time, sometimes called a modal survey. The recording data is divided into ¼-second increments by default. This view is most useful for identifying time-varying trends in the frequency content of a signal. Only one axis is shown at a time (the Z axis is shown by default). Use the radio buttons at right to switch the axis currently displayed.

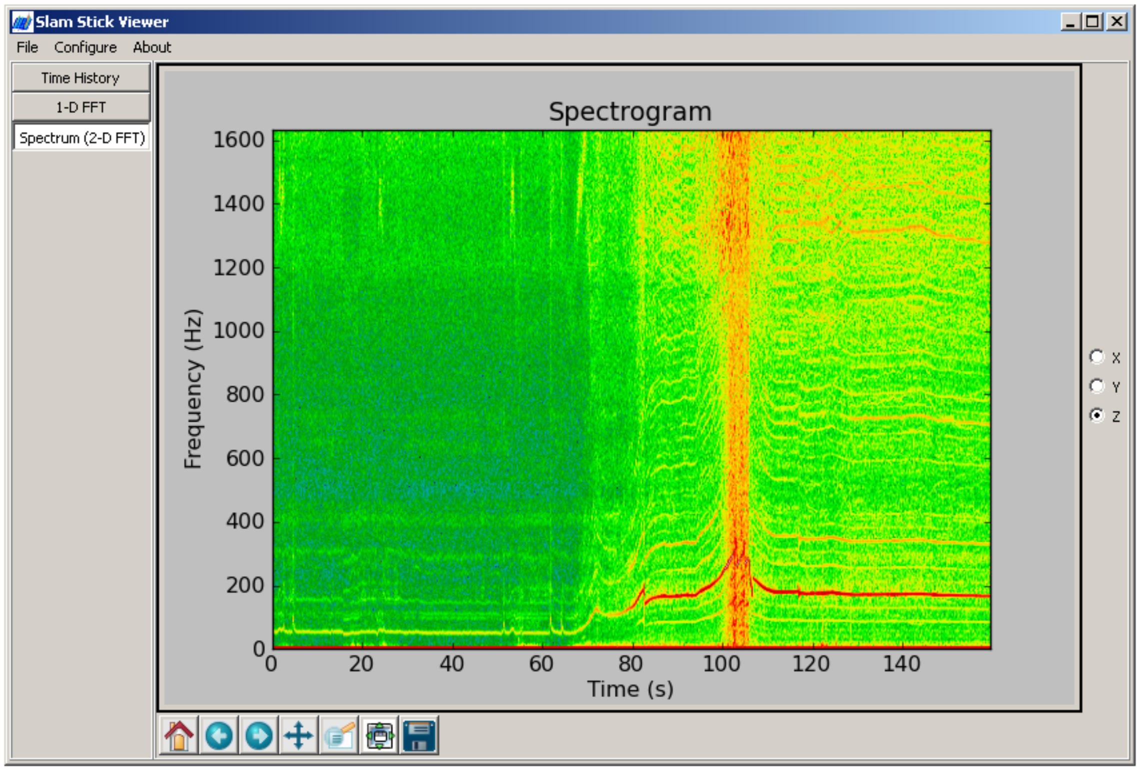


Figure 6: Spectrogram view

FILE FORMATS

Native Slam Stick recordings

For technical performance reasons, the Slam Stick records vibration data and its configuration in a custom binary format. Most users will not need to work with these data formats directly. Please refer to the "C" examples (available from the Slam Stick download page) for the structure of the binary recording file and configuration file.

CSV Data

CSV (Comma-Separated Value) is a plain-text format supported by common office and spreadsheet applications as well as most scientific computing packages. CSV time history data exported from the Slam Stick is saved as a list of samples in the following format:

```
t, X, Y, Z <newline>
```

where t is the time in seconds (the recording begins at t = 0), and X, Y, Z are the amplitudes in Gs (-16 ~ +16) for the respective axes. <newline> will be the appropriate end-of-line character(s) for your operating system.

Example:

```
0.0,                 0.140625,        1.1875,          0.0625

0.000305986308852,   0.140625,        1.1875,          0.0390625

0.000611972617703,   0.1171875,       1.15625,         0.046875
```

...

CSV data exported from Slam Stick Viewer will be formatted as shown; likewise, CSV data imported into Slam Stick Viewer is expected in this format. Loading arbitrary or incorrectly-formatted data may cause undesired operation.

The output format of FFT data is identical, except that the first column will consist of frequency steps in Hz instead of time steps.

Due to the limited decimal precision of the time step data, when using CSV formatted data in other applications, the following method is recommended for computing the period or sampling frequency:

Total_time: (last_timestep – first_timestep)

Period: Total_time / (number_of_rows – 1)

Frequency: 1/Period

This will also correctly handle files whose time does not necessarily start at t=0. Since the period is the time taken between any two samples, the total time is divided by the total number of samples (rows) *minus one*.

TROUBLESHOOTING

After opening a long recording, the main window is frozen!

The full FFT for all axes may take a while to complete (up to several minutes), especially for longer recordings or slower computers (e.g. netbooks). If the program is still unresponsive after several minutes, please see the next item.

Error messages referring to "side-by-side" or "WinSxS" configuration

This error means that required Microsoft run-time libraries are missing. This could occur on older Windows machines if automatic updating is disabled. Please install the Microsoft Visual C++ Run-Time Libraries (MSVCRT) version 8.0 (2008) and 9.0 (2010) from http://www.microsoft.com. At the time of this writing, the downloads of the current versions are called:

- Microsoft Visual C++ 2008 SP1 Redistributable Package
- Microsoft Visual C++ 2010 Redistributable Package

The program froze, crashed, or I may have found a bug - now what?

Please let us know with a bug report to the project maintainer (tgipson at mide.com). Please include:

- The debug log files generated by the program (debug.log, mpl-debug.log, and slamview.exe.log, if present)
- The action being performed when the error occurred, if any
- If the error occurred after loading a particular recording file, please send the recording file if possible.

The debug logs contain very basic information about the recording(s) displayed in the current session (e.g. length in seconds, sample rate), but cannot be used to reconstruct your original recording.

USING AND CUSTOMIZING SLAM STICK VIEWER (PYTHON SOURCE)

Executing

To run the Slam Stick Viewer GUI standalone, run `slamview.py` without any arguments.

Reading and Writing Files

Slam Stick file read and write functions are provided by `ss_read.py`, and can be called from an external script (requires NumPy).

`ss_reader(parent=None, filename=None, recording_num=None)`

Reads a vibration file in Slam Stick native, Slam Stick 'split', or CSV format (depending on file extension). parent is the parent Tk window (if any). If no filename was provided, a file selection dialog will first be spawned. If a Slam Stick file contains multiple recordings and recording_num is not specified, a selection dialog will be spawned.

Returns a tuple consisting of `(datalist, fActual, recording_time_actual)`, where `datalist` is a 4-by-n NumPy array consisting of t, X, Y, Z; `fActual` is the exact sampling frequency, and `recording_time_actual` is the exact elapsed time of the recording.

`ss_writer(parent=None, filename=None, theData=None, heading=None, dialect='excel')`

Writes a .CSV file consisting of the contents of `theData`, a list or NumPy array containing the numeric data to write. `parent` is the parent Tk window (if any). If no filename was provided, a file selection dialog will first be spawned. Please refer to the Python 'csv' module documentation for details.

Adding custom plots

A generic plotting class is available in `ssplot.py`; each of the builtin plot types is a subclass of `ssplot`. Each `ssplot` object expects an instance of '`notebook`' (`notebook.py`, providing a simple tabbed interface) as an input argument, and will add its plot and corresponding name to the notebook.

Its main methods are:

`Constructor (__init__)` – Creates a Tk frame to hold the plot and its controls. Creates a matplotlib Figure, axes (subplot), drawing canvas and matplotlib toolbar, and adds itself to the notebook. Calls `set_plot_type, create_line_controls, create_axis_controls`.

`set_plot_type(self)` – Sets the appropriate plot title and axis labels, plus any plot-specific options required by the subclass (for example, a default windowing function for FFT plots). Returns an appropriate tab name for this plot in the notebook.

`create_line_controls(self)` – Create plot-specific plot (line) controls. The default create_line_controls, for line plots, creates checkboxes to show or hide each line. For other plot types, this method should be overridden.

`create_axis_controls(self)` – Creates plot-specific axis controls. The default create_axis_controls, for line plots, creates checkboxes to switch between linear and logarithmic axis scales. For other plot types, this method should be overridden.

`update(self)` – Fully updates the plot window, including rescaling of the axes. This method should be called after changing or adding new data to a plot.

`redraw(self)` – Redraws the current plot window, but does not recompute axis scale (much faster than a full update). This method should be called after making minor changes to a plot, such as changing plot element colors or visibility, which do not affect the required axis limits.

Notably absent from the base ssplot class is a '`plot(…)`' method. An appropriate `plot(…)` method must be explicitly provided by the subclass. In most cases this plot method will accept an array containing the data to be plotted (plus any other needed arguments), and in turn call the appropriate matplotlib plotting functions on the class instance's Axes (named `self.myplot`). For example, the existing line-based plotting methods call `self.myplot.plot(…)`, and spectrogram plotting calls `self.myplot.specgram(…)`, etc.

With the appropriate `ssplot` subclass, the complete steps to add a plot to the GUI are:

- In the main script (`slamview.py`), instantiate your custom plot and add it to the 'notebook' (tabbed window):

  ```
  yourFrame = yourPlotType(tabsFrame, root)
  ```

  The plot window (plus its controls, etc.) and its corresponding tab will be added to the main window.
- In the main script's `populate_window()` function, add the appropriate yourPlotType.plot(…) call:

  ```
  yourFrame.plot(datalist, args)
  ```

  To keep the main script uncluttered, it is recommended for plot methods to accept the existing time-domain recording data (`datalist[…]`) and perform any necessary calculations internally.