

HSPICE Toolbox for Matlab and Octave

(also for use with Ngspice)

Michael H. Perrott

http://www.cppsims.com/download_hspice_tools.html

Copyright 1999 by Silicon Laboratories, Inc.

29 July 2011

Note: This software is distributed under the terms of the GNU Public License (see the included COPYING file for more details), and comes with no warranty or support.

The Hspice toolbox for Matlab and Octave is a collection of routines that allow you to manipulate and view signals generated by Hspice or Ngspice simulations within either the Matlab or Octave programs. The primary routine is a mex program called `loadsig` that reads binary output files of transient, DC, or AC sweep data generated by Hspice or Ngspice into Matlab or Octave. The remaining routines are used to extract particular signals and view them. For simplicity, we will focus on Matlab in the remainder of this document, but the instructions for use with Octave are identical.

We will begin this document by explaining how to include the Hspice toolbox in your Matlab session. A list of each of the current functions will then be presented. Finally, we will provide examples of using these routines to view and postprocess signals from Hspice output files.

Setup

To use the Hspice toolbox, simply place the included files into a directory of your choice, and then add that directory to your Matlab path. For example, inclusion of the path `'/home/user/matlab/bin'` in Matlab can be done by adding the line

```
addpath('/home/user/matlab/bin')
```

to the file `startup.m` located in your home directory. In addition, you can specify the plot background to be black by adding another line to `startup.m`:

```
colordef none;
```

Once you've made the above changes to `startup.m`, start Matlab as you normally would. Matlab will automatically read `startup.m` from your home directory and execute its commands.

Platform Compatibility

All files should work across different computer platforms, though the `loadsig` mexfile should generally be recompiled. It is currently compiled for 64-bit Redhat Linux and 64-bit Windows 7 machines. To compile the `loadsig` function for a different platform, go to the directory containing `loadsig.c` within Matlab, and then type `mex loadsig.c` within Matlab. Note that you can also compile `loadsig` in Octave in the exact same manner since the `loadsig.c` source code is compatible with both Matlab and Octave.

List of Functions

The following functions are currently included in the Hspice toolbox:

- `x = loadsig('hspice_output_filename');`
 - Returns a Matlab structure into variable `x` that includes all of the signals that are present in the Hspice binary output file, `hspice_output_filename`.
- `lssig(x)`
 - Lists all of the Hspice signal names present in the structure `x`.
- `y = evalsig(x, 'nodename');`
 - Pulls out the signal `nodename` from the structure `x` and places into variable `y`. The string `nodename` can be an expression involving several Hspice signals. If you only performed one sweep in the simulation (as is common), then `y` will contain one column. If you performed several sweeps, `y` will contain several columns that correspond to the data for each sweep. If you have set the global Matlab variable `sweep` to a nonzero number, however, then `y` will contain only one column corresponding to the value of `sweep`. If `sweep` equals zero, all the sweep columns are included in `y`.

- `plotsig(x, 'plot_expression', 'optional_plotspec')`
 - Plots signals from the structure `x` according to the expression given in `plot_expression`. The string `optional_plotspec` is used to create logscale plots; it can be specified as `logx`, `logy`, or `logxy`. The string `plot_expression` specifies the nodenames, and corresponding mathematical operations, that you would like to view. In this expression, commas delimit curves to be overlaid and semicolons delimit separate subplots on the same figure. All numeric node names should be prepended by '@' to distinguish them from constants. Some examples of using `plotsig` are:
 - * `plotsig(x, 'v1,v2;v3')`: overlays `v1` and `v2` on the same subplot, and plots `v3` on a separate subplot.
 - * `plotsig(x, '(v1+v2)^2; log(abs(v3))')`: plots the listed expressions on separate subplots.
 - * `plotsig(x, 'db(v1); ph(v1)', 'logx')`: plots the magnitude (in dB) and phase (in degrees) of `v1` on a semilogx axis.
 - * `plotsig(x, 'v1+@2+3')`: plots the addition of node `v1`, node 2, and the constant 3.
 - * `plotsig(x, 'integ(TIME,v1); avg(TIME,v2)')`: plots the integral of `v1` and average of `v2` on separate subplots.
- `tzoom`
 - Brings up buttons on the plot to allow nice zooming functions. Type `help tzoom` at the Matlab prompt for more info.
- `figname`
 - Allows easy labeling of figure windows. Type `help figname` at the Matlab prompt for more info.
- `xlima`
 - Sets the x-limits of all subplots in a figure. Three options are possible:
 - * `xlima`: sets all subplots to the same x-axis as the last subplot that was zoomed into,
 - * `xlima([xs xe])`: sets all subplots to the x-axis limits specified,
 - * `xlima('auto')`: resets all subplots back to autoscaling.
- `eyesig(x,period,start_off,'nodename')`

- Creates an eye diagram for nodename contained in `x` with the specified `period`. All data samples prior to `start_off` are ignored when creating the diagram (useful for removing the influence of transient effects from the eye diagram). The string `nodename` can be an expression involving several variables for the CppSim version (`eyesig`), but assumes a constant time step (which is invalid for Hspice simulations). NOTE: use instead `eyesig_old` for Hspice simulations — this version can only handle one variable and is more primitive than its CppSim counterpart, but does take into account the non-constant time step of Hspice simulations.

Examples

Viewing Signals Generated by Hspice

Use the Matlab command `cd` to go to a directory containing either transient, DC, or AC sweep data within a binary file generated from Hspice. We will assume a filename of `test.tr0`, and now list a series of Matlab commands that will be used to display nodes `q` and `qb` in that file.

- `x = loadsig('test.tr0');` %% loads Hspice signals into `x`
- `lssig(x)` %% verify that nodes `q` and `qb` are present
- `plotsig(x,'q; qb; q-qb')` %% plot expressions of interest

Viewing Signals Generated by Ngspice

Use the Matlab command `cd` to go to a directory containing either transient, DC, or AC sweep data within a binary file generated from Ngspice. We will assume a filename of `simrun.raw`, and now list a series of Matlab commands that will be used to display nodes `q` and `qb` in that file.

- `x = loadsig('simrun.raw');` %% loads Ngspice signals into `x`
- `lssig(x)` %% verify that nodes `q` and `qb` are present
- `plotsig(x,'q; qb; q-qb')` %% plot expressions of interest

Doing Postprocessing in Matlab

Use the Matlab command `cd` to go to a directory containing a binary transient, DC, or AC sweep file generated from Hspice. We will assume a filename of `test.tr0`, and now list a series of Matlab commands that will be used to postprocess nodes `q` and `qb` in that file.

- `x = loadsig('test.tr0');` %% loads Hspice signals into `x`
- `lssig(x)` %% verify that nodes `q` and `qb` are present
- `t = evalsig(x,'TIME');` %% loads time samples into Matlab variable `t`
- `q = evalsig(x,'q');` %% loads signal `q` into Matlab variable `q`
- `qb = evalsig(x,'qb');` %% loads signal `qb` into Matlab variable `qb`
- `qdiff = q-qb;` %% perform expressions in Matlab
- `plot(t,q,t,qb,t,qdiff)` %% plot variables using Matlab plot command