acidbourbon

a digital & analog audio electronics blog by Michael Wiebusch

LTspice & NumPy – Part 1: seamless integration of LTspice in python signal processing

November 26, 2019 · by acidbourbon · in software hacks and fixes. ·

Motivation

Might this article be interesting for you? Answer the following questions:

- Do you like signal processing with python and numpy? – Correct answer: **Yes!**
- Do you like simulating analog circuits (i.e. filters) with LTspice? – Correct answer: **Yes!**
- On Linux? – Correct answer: **Yes!** (It works on Windows, too.)
- Don't you wish you could just plug in an analog circuit simulation into your python script and use it as a digital filter? – Correct answer: **OMG** that would be great!

Okay, read on.

There are python libraries available for reading the LTspice binary output format (.raw).
E.g. there is "ltspice" (https://pypi.org/project/ltspice/ (https://pypi.org/project/ltspice/)) which you can install via pip. But somehow it crashed when I tried to use it.
I had more luck with the "pyspicer" utilities written by Nuno Brum (https://www.nunobrum.com/pyspicer.html (https://www.nunobrum.com/pyspicer.html)).

Update: Nuno has packaged his stuff in a PyPi (pip) module called "PyLTSpice", so reading raw files is a piece of pie now! Yaay!

But this article is not about just slurping the output of an LTspice simulation and plotting it with python (which is nice, of course),
but about creating the seamless experience of using LTspice just like another python module – without clicking on anything, without even opening the LTspice window when applying the filter circuit. This involves calling LTspice from the command line in a non-graphical "batch mode" and then using Nuno Brum's pyspicer/LTspice_RawRead module.

What it does

I wrote a little wrapper around "LTspice_RawRead" and the LTspice system call, called apply_ltspice_filter (https://github.com/acidbourbon/numpy_ltspice_filter/raw/master/apply_ltspice_filter.py). No magic, it's all about convenience for you, really. The desired functionality is that you input a numpy data vector into the Spice simulation and get an equal sized data vector back with the circuit's reaction. LTspice usually decides on its own which time steps it does at which point during the calculation. So apply_ltspice_filter has to resample the Spice output to the same time quantization as the input signal. Also the script *forces* LTspice to make fine enough time steps, based on the time quantization of the input signal. This way you have an unbroken workflow of time discrete signal processing.

Prerequisites

- **Python3**
- **Matplotlib**
- **Numpy**

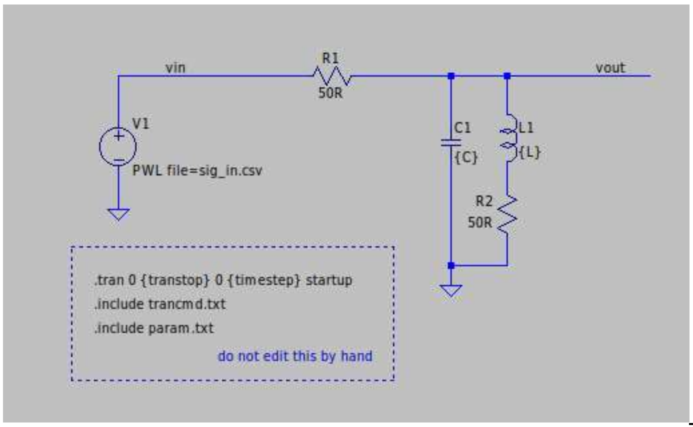
- **Scipy**
- **PyLTSpice** (install via pip)
- **LTspiceXVII**
  - installed with wine (or natively on windows)
  - Download the LTspice installer from http://ltspice.analog.com/software/LTspiceXVII.exe (http://ltspice.analog.com/software/LTspiceXVII.exe)
  - The main executable should reside at "C:\Program Files\LTC\LTspiceXVII\XVIIx64.exe" (windows/wine path), i.e. "~/.wine/drive_c/Program\ Files/LTC/LTspiceXVII/XVIIx64.exe" (unix path)

Example

We will:

- Create a signal in python
- Run it through an LTspice filter circuit, while we control some of the filter's adjustable parameters in python
- Plot the results within python

However, let's design the filter circuit first. I assume you have LTspice installed (it installs and runs perfectly fine in Linux with wine).
We open LTspice and create a circuit like below and save it as "filter_circuit.asc (https://github.com/acidbourbon/numpy_ltspice_filter/raw/master/filter_circuit.asc)" in a folder of your liking.



```
.tran 0 {transtop} 0 {timestep} startup
.include trancmd.txt
.include param.txt
```

(https://acidbourbon.files.wordpress.com/2019/09/selection_009.png)
filter_circuit.asc

The important bits here are:

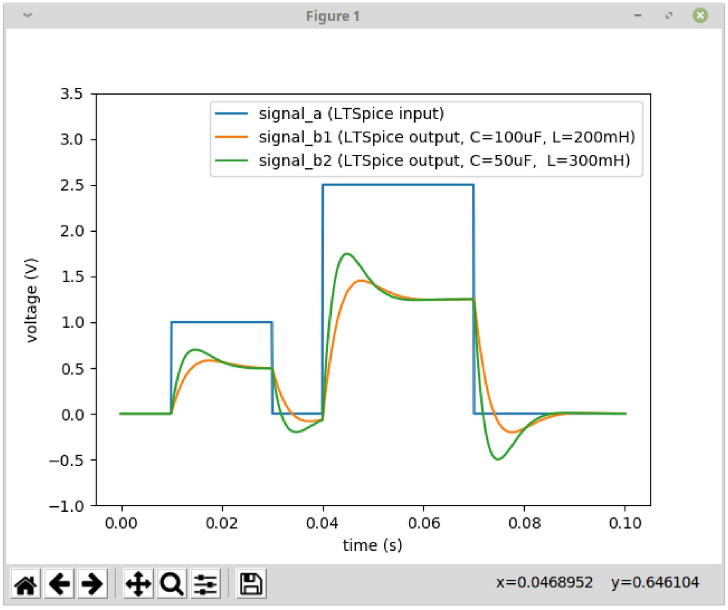
- The voltage source which is configured to "play back" a signal defined in a not-yet-existing, soon to be auto-generated input file ("sig_in.csv")
- The voltage output of the circuit has to be labeled "vout"
- The commands in the box, telling LTspice to include parameters from external .txt files and the .tran command defining the stop time and the time step of the transient simulation

The rest of the circuit can later be changed to your own needs. Note that the capacitor and the inductor in our example circuit have no defined values yet but their values are set to variables "C" and "L" (which we will set in the python code).

Okay. We designed our filter. Now we close LTspice and do the rest in python. Let's create a script like this (filter_demo.py (https://github.com/acidbourbon/numpy_ltspice_filter/raw/master/filter_demo.py)), which uses "apply_ltspice_filter.py (https://github.com/acidbourbon/numpy_ltspice_filter/raw/master/apply_ltspice_filter.py)" (see Download section):

```python
#!/usr/bin/env python3

import numpy as np
from apply_ltspice_filter import apply_ltspice_filter
import matplotlib.pyplot as plt

##################################################
##              generate test signal            ##
##################################################

# our samples shall be 100 ms wide
sample_width=100e-3
# time step between samples: 0.1 ms
delta_t=0.1e-3
samples = int(sample_width/delta_t)

time = np.linspace(0,sample_width,samples)

# we want 1 V between 10 ms and 30 ms, and 2.5 V between 40 and 70 ms
signal_a = 0 + 1*((time > 10e-3) * (time < 30e-3)) + 2.5*((time > 40e-3) * (time < 70e-3))

##################################################
##        apply filter - configuration 1        ##
##################################################

# all values in SI units
configuration_1 = {
  "C":100e-6, # 100 uF
  "L":200e-3 # 200 mH
}

dummy, signal_b1 = apply_ltspice_filter(
      "filter_circuit.asc",
      time, signal_a,
      params=configuration_1
      )

##################################################
##        apply filter - configuration 2        ##
##################################################

configuration_2 = {
  "C":50e-6, # 50 uF
  "L":300e-3 # 300 mH
}

dummy, signal_b2 = apply_ltspice_filter(
      "filter_circuit.asc",
      time, signal_a,
      params=configuration_2
      )

##################################################
##            plot input vs output(s)           ##
##################################################

plt.plot(time,signal_a, label="signal_a (LTSpice input)")
plt.plot(time,signal_b1, label="signal_b1 (LTSpice output, C=100uF, L=200mH)")
plt.plot(time,signal_b2, label="signal_b2 (LTSpice output, C=50uF,  L=300mH)")
plt.xlabel("time (s)")
plt.ylabel("voltage (V)")
plt.ylim((-1,3.5))

plt.legend()
plt.show()

```

This creates the following plot:



Et voila, there you go. Numpy->LTspice->Numpy. Same time quantization before and after.

Download

find above example on GitHub: https://github.com/acidbourbon/numpy_ltspice_filter.git (https://github.com/acidbourbon/numpy_ltspice_filter.git)

or download as zip: https://github.com/acidbourbon/numpy_ltspice_filter/archive/master.zip (https://github.com/acidbourbon/numpy_ltspice_filter/archive/master.zip)

Just download, unpack and run "filter_demo.py"

Speeding things up?

If you are interested in a faster, more efficient way to process signals with your LTspice filter, check out LTspice & NumPy – Part 2: Fast Convolution Filter (https://acidbourbon.wordpress.com/2019/12/04/ltspice-numpy-part-2-fast-convolution-filter/)

Thank you

A warm thank you note to Nuno Brum, who wrote the amazing LTspice RawReader python module and to Alex Stallman and Henk who tested "apply_ltspice_filter" for windows compatibility.

Sponsored Content
[Bilder] 20 coole Fotos, die während der Dreharbeiten von Star Wars geschossen wurden DailySportX | Sponsored (https://rfvtgb.dailysportx.com/worldwide/strwrs-cp-gu-ob-ge?utm_source=outbrainjk&utm_campaign=ds-strwrs-s15-win-gr-mm-11112&utm_term=$section_name$&utm_medium=$section_id$&utm_bid=cpc&obOrigUrl=true)

Your IQ Is Over 140 If You Get Even 5/20 On This Quiz TooCool2BeTrue | Sponsored (https://www.toocool2betrue.com/random-knowledge-quiz/?utm_source=ob-quiz&utm_template=quiz&utm_medium=$publisher_name$-$section_name$&utm_term=$publisher_id$_$section_id$&utm_content=00a971c69db92c5 ob-t1-d-randomquiz-22.11.10-10711-cnr_roi125_sauto_quiz_chrome_1019&utm_cpc=cpc&obOrigUrl=true)

[Quiz] How Many Movies Can You Identify From a Single Frame? https://themoneytime.com/ | Sponsored (https://themoneytime.com/movie-scene-quiz/?utm_source=ob-quiz&utm_template=quiz&utm_medium=$publisher_name$-$section_name$&utm_term=$publisher_id$_$section_id$&utm_content=00c917c324671f39 ob-t1-d-moviesquiz-22.11.08-10705-cnr_roi125_sauto_quiz_chrome_2691&utm_cpc=cpc&obOrigUrl=true)

Tags: automation, ltspice, numpy, python, signal processing

14 responses to "LTspice & NumPy – Part 1: seamless integration of LTspice in python signal processing"

1. Pingback: Circuit Simulation in Python ~ Cornay Britz·

2. Pingback: seamless integration of LTSpice in python/numpy signal processing – ElexHere·

3. *Alex Stallman* December 2, 2019 at 15:31 · · Reply →
   Brilliant. I’ve basically got this working on Windows. I just can’t get both plots showing. Using the filter_demo.py from Git, I only get 1 output plot, with the shape of the b1 in the example, but labelled b2. I get the error “‘C:\Program’ is not recognized as an internal or external command, operable program or batch file.” as if it’s struggling to go back to the OS path, although it clearly worked perfectly the first time!

4. *Henk* December 2, 2019 at 16:53 · · Reply →
   I was able to run your example on Windows, when using subprocess for calling LTSpice. Replacing “os.system(ltspice_command+” {:s}.asc”.format(simname))” with “subprocess.run([*ltspice_command.split(), “{:s}.asc”.format(simname)])”. Don’t forget to import subprocess.

5. *acidbourbon* December 3, 2019 at 19:30 · · Reply →
   Hey Alex, Henk. Thanks a lot for testing the script in windows. Henk, I commited your proposed changes to the repository on GitHub. Can either of you pull the newest version from GitHub, run it and confirm that it works on windows? Cheers!

6. Pingback: LTspice & NumPy – Part 2: Fast Convolution Filter | a c i d b o u r b o n·

7. Pingback: ICYMI CircuitPython Newsletter: MP3 decoding, CircuitPython snakes its way to Fomu, NXP, and more! #Python #Adafruit #CircuitPython #ICYMI @circuitpython @micropython @ThePSF @Adafruit « Adafruit Industries – Makers, hackers, artists, designer·

8. *David B* February 17, 2020 at 11:59 · · Reply →
   Great job ! Worked perfectly on the first try.
   Thanks a lot !

   - *acidbourbon* February 22, 2020 at 11:37 · · Reply →
     Thanks for the feedback! Nice to hear that it has become a useful tool for someone else 🙂 Good day to you, Sir!

9. *Booysen, MJ* September 1, 2020 at 03:50 · · Reply →
   Hi, thanks for an awesome tool! Please note, we had to make the following change when we got LTSpice timestep too small errors:
   From: f = interpolate.interp1d(vout_x,vout_y)
   To: f = interpolate.interp1d(vout_x,vout_y, fill_value=”extrapolate”)

   - *acidbourbon* September 1, 2020 at 08:58 · · Reply →
     Thank you for the feedback! I added the modification to the code on github! All the best, Micha

10. *Henrique Miguel Rosa Santana da Cruz* February 17, 2021 at 15:39 · · Reply →
    Hello first of all, what a great job,
    I was wondering if there was a way to not use this sig_in.csv file but instead use a SINE with certain fix values

    - *acidbourbon* February 19, 2021 at 22:36 · · Reply →
      Hi. Thank you 🙂 So, yes … you can right click on the voltage source and select “sine wave” instead of “PWL file”. And there you go.

11. *pollytaiwan* May 13, 2021 at 13:53 · · Reply →
    hello, why can’t I download : ” filter_circuit.asc”?

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Create a free website or blog at WordPress.com.