# **ngfmVis Update**

GitHub link: https://github.com/cooperbell/ngfmVis

## **Overview**

ngfmVis has been updated to include some new features and address performances issues. Below is a breakdown of what’s been changed and any extra info that’s needed (i.e. how to use it).

Overall, the general flow of the program is the same as before. You still invoke ngfmVis with the same arguments as before, for example:

ngfmVis('file','capture09102019.txt','null');

ngfmVis('serial','COM6','null');

With this update I tried to make the program as similar to what is was before while still providing meaningful enhancements. So, some old features that you put in are still there along with new implementations of them (i.e. key pressing to send hardware commands but also a new GUI page for sending them) so you don’t have to learn a completely new way to do the same thing you did before.

What’s been changed:

* Input parameters GUI page
* Reading the serial port on an async worker
* Real-time parsing
* Modular plot
* HK data GUI page
* Hardware Commands GUI page
* Logging update
* Faster, more responsive GUI
* Misc. data now on the bottom
* Heartbeat
* Documentation for every function

\*For a more comprehensive and detailed look at what was changed and my process for doing it, look through my pull requests and issues on my GitHub.

## **Input parameters GUI page**

This is a GUI that is for supplying arguments to the ngfmVis call. Invoke it by typing in ngfmVis() into the console without arguments and it’ll come up. I imagine you won’t use this much, but it may be helpful for people who are new to this program.

## **Reading the serial port on an async worker**

To solve the issue of dropping packets on the serial port, reading data from the serial port has been moved to run on an asynchronous worker. It invokes sourceMonitor.m to read the port at whatever rate you specify for targetSamplingHz in ngfmLoadConstants.m. To see in real-time what the port is sampled at, look at the **Src Hz** label on the bottom bar of data on the main GUI page.

**\*\*\*NOTE**: In order to properly close the async worker and serial port, close ngfmVis by one of these three ways:

* Click the ‘Quit Program’ button in the top left of the main GUI page
* Press the ‘q’ key while the GUI is active (i.e. click anywhere on it)
* Close the figure

**DO NOT** click the red square ‘Quit Debugging’ in MATLAB. This prevents the main program from telling the worker to close itself up and it’ll cause issues with running the parallel pool and possibly connecting to the serial port.

**If you do experience problems with the worker or serial port locking up:**

* If the program doesn’t seem to be doing anything, then most likely the issue is with the parallel pool getting stuck. If you click the pause button and a file named JavaBackedFuture.m comes up, then it’s definitely a parallel pool issue. Fix it by resetting the parallel pool:
  + Type into the console: delete(gcp);
  + Once the pool is closed, type gcp;
* If you get an error like:

Open failed: Port: /dev/tty.usbserial-6961\_0\_0080 is not available. Available ports: /dev/tty.SOC, /dev/cu.SOC, /dev/tty.MALS, /dev/cu.MALS, /dev/tty.Bluetooth-Incoming-Port, /dev/cu.Bluetooth-Incoming-Port. Use INSTRFIND to determine if other instrument objects are connected to the requested device.

then there’s a chance that the worker didn’t properly close up the port before terminating and the port is locked up.

* + This one is a bit annoying to deal with but plugging the USB into a different USB slot works and using a different port, otherwise restarting MATLAB will release all the ports it has a hold on.

## **Real-time parsing**

Since the main program and the async worker communicate by way of pollable queues, the program quickly stops being ‘real-time’ as the number of packets in the queue grows, and it stops showing “what’s happening now” and starts showing “What has happened”. To fix this, each iteration in the main program all of the packets in the queue are processed at once – whether there are 2 packets or 2000 – and plots the final result. It also does this with the heartbeat messages.

## **Modular plot**

The big right-hand plot is now more modular. What you can do with it:

* Use the dropdown to switch what plot is being displayed currently. At the beginning of the program, it pulls the scripts from the /spectraPlots folder.
* Adding a script
  + Click ‘Add Plot’ button
  + Select .m file
  + Choose to add permanently or not
    - If you choose to add permanently, it gets stored into the /spectraPlots folder
    - If you choose to not add permanently, it gets stored into a temporary folder on your system and disappears once the session is done.
* Deleting a script
  + Brings up a GUI that has a dropdown of the currently used scripts to delete.

**\*\*Note:** There is minimal error checking against what scripts are run. A script will only get kicked out if it throws an error, otherwise it’ll run whatever is loaded. So, you can import in a file that doesn’t plot and does something else entirely and it’ll run it as long as it doesn’t throw an error. My suggestion is to test out your script in a different MATLAB program to make sure it works then throw it in.

## **HK data page**

Located on the Housekeeping Data tab, it displays a graph for each of the housekeeping data points. It shows a recent history by the number of packets, specified by hkPacketsToDisplay in ngfmLoadConstants.m. Default is 100 packets.

Pressing the ‘`’ key (debugData toggle) changes some of the titles for the graphs. This is a feature you had before.

## **Hardware Commands page**

Located on the Hardware Commands tab, it provides an easier way to send commands to the magnetometer over serial. Most of the functionality is the same as before, so it should be pretty self-explanatory.

For example, if you want to send Feedback, Static, increase of 5 on channels X and Y then you:

* Check channels X and Y at the top
* Specify Static on the radio button
* Type in 5 for the Inc/Dec text field
* Click ‘Send Feedback’
* It’ll send 'X' 'S' 'I' 'I' 'I' 'I' 'I' 'Y' 'S' 'I' 'I' 'I' 'I' 'I'

For the table load and table file, it sends the data in the format ‘0x0000’ character by character, address then data.

For the table file, it sends the channel, then address, data, address, data, etc.

The ‘Enter a command’ will send whatever you type into there.

## **Logging update**

Logging is largely the same, either normally or the debug data (which has been consolidated into one file logData.m). Now, if you leave the third argument to the ngfmVis() call as an empty string it’ll generate a log file with the filename as a timestamp. This is the recommended way to log the files since it won’t overwrite old log files that you may want to see later.

Three different logging scenarios:

1. ngfmVis('file','capture09102019.txt','null');
   1. 'null', Does not log
2. ngfmVis('file','capture09102019.txt','');
   1. empty string, generates log file with timestamp in format log\_date-time.txt
      1. ex. log\_20200106-133750.txt
3. ngfmVis('file','capture09102019.txt','log.txt');
   1. Logs to a file named log.txt

Pressing the ‘`’ (debugData flag) logs it differently like before.

## **Faster, more responsive GUI**

The GUI has been rewritten to be faster and more responsive. It looks exactly the same as it did before. However, more GUI elements can be added without much more performance overhead. This increase in speed should also allow you to interact with the plots more easily, something that I don’t think was very doable before. For more information, check out my pull requests #26 and #31 on the project GitHub which goes into much more detail on how I set up the GUI and manage the data.

## **Misc. data on the bottom**

The data on the left column have been moved to the bottom, minus the house keeping data which has been moved to its own page. Pressing the ‘`’ (debugData flag) changes the labels like before.

## **Added heartbeat**

There is now a heartbeat for the asynchronous worker since we have limited information about what it’s doing and what’s happening with it. The **Heartbeat** label on the bottom bar of misc. data on the main GUI page displays how long ago it got a heartbeat message. If all is well, then most of the time it should display ‘<1s’, meaning it got the message less than a second ago. If it climbs above a few seconds, then typically something is wrong.

\*\*Note: If you block execution of the main program – i.e. browsing for a file – then the heartbeat label will temporarily show a value of a few seconds. This is expected behavior and will quickly return to ‘<1s’ if all is well with the async worker.

Every second, the async worker sourceMonitor.m starts a stopwatch timer and sends it over to the main program. The main program takes that timer and stops it, display the elapsed time on the GUI.

## **Documentation for every function**

Added documentation to every function in the program. If you don’t know what a function is doing or want to develop on it, that comment block plus the inline comments should be sufficient enough to easily add more code. It follows MATLAB’s help feature, meaning you can type in help functionName and it’ll display the comment block I put there.