Instructions for Installation and Setup of August 2022 check scripts

Accessing code:
These instructions come with the files saved at: https://github.com/autonomous-phase-sensitive-radar/processing.

To access the latest version of this code, go to the link above, click the button that says “Code”, and click the option to download ZIP. Unzip the folder, and then you will have the code accessible.

Code requirements:
These scripts require the latest version of MATLAB installed, and that MATLAB needs to work offline. This means making sure that the license is properly renewed (for more information regarding these licenses, go to: https://www.cuit.columbia.edu/content/matlab). Additionally, MATLAB needs to have the Signal Processing Toolbox along with the Statistics Toolbox installed. Furthermore, these scripts call upon additional utility scripts from Keith Nicholls. They are included in the Github folder with the processing scripts as of July 18, 2022. When opening MATLAB, the folder containing these utility scripts (named nicholls_utils) should be added to the path.

The directory structure should look like:
… whatever directory the processing folder is stored in
    /processing
        …field processing scripts
        /nicholls_utils
            … utility scripts from Keith in here

All the relevant files outside of the main field processing scripts (of which this instruction document is a part of) should already be on the toughbook used in the May 2022 field trip to Ilulissat. Consequently, MATLAB and its relevant toolboxes should not need to be reinstalled, though it is important to check that it still works offline and that the license is valid.

Running the scripts:
There are 3 in-field scripts to run: `check_dates_vs_time.m`, `check_clipping_attenuation.m`, and `check_strain_rate.m`. They are listed in order of importance/order to run. Their individual instructions are as follows.

check_dates_vs_time.m: This script checks to make sure there are no glaring data gaps. It has two settings that allow for either a quick coarse check (in field) or a longer fine check (in town).
- After inserting the SD card, open the script, and on line 8, set the variable “`myFolder`” to the full path to the SD card (including the SD card itself).
- Specify if you want to do a quick check or thorough check by changing the value of the “`resolution`” variable (in the field it would be set at 1 for the quick check; in town, 0)
- Click run, and then the script will generate a plot marking the dates (and times if it’s the thorough check) of each file created (or burst for thorough check). One can use this plot to identify if there’s any lapses in measurements.

check_clipping_attenuation.m: This script checks to make sure there is no clipping or to make sure that the signal is not too highly attenuated. It would be run twice (ideally) in the field, once for clipping and once for high attenuation.
- First, we insert the SD card and specify its path in the same way as the check_dates_vs_time.m script (this time the “`myFolder`” variable is on line 12).
- Specify the max number of plots to generate (line 15). This is to prevent potentially hundreds of plots popping up in the case that many measurements are flagged. It is preset to 20.
- Specify how many files (roughly 1 file a day) and how many bursts (roughly 96 bursts in a file) to skip over when iterating through all the bursts. This is to speed up the script runtime in the field. These specifications are done by changing “`file_spacing`” (line 19) and “`burst_spacing`” (line 20).
- Specify whether we want to detect clipping or high attenuation (prioritizing clipping, since we want conservative attenuation settings for night and winter). This is done at line 23 with the “`clipping`” variable

(setting it to 1 means we're flagging clipped data; 0 means flagging highly attenuated data).
- If we are flagging highly attenuated data, specify the amplitude that serves as a cutoff with the variable "`amplitude`" (line 27). Note that clipping happens when the amplitude is greater than 1.25. The preset for this is 0.25.
- The code will display a warning for which file and burst contains flagged data, and also generate a plot showing the raw voltage, a histogram of voltages, and a range plot generated by that specific measurement. We can use those plots to verify if the measurements are indeed clipped.

check_strain_rate.m: This script does a rough check to make sure that vertical velocity measurements make sense. It will be run in the field and returns the vertical velocities of specified files and bursts.
- After inserting the SD card, specify the two .DAT files to compare (lines 12-13). The simplest comparison is the first and last files created. Make sure to include the full path to the files.
- The code will then compare the first and last file. It generates 3 plots: the first one being a vertical velocity between the first and second burst of the first file, the second one being between the first and second burst of the last file, and the third one being between the average of the first file and the average of the last file.
- The main things to look at are the first two plots - we want to make sure that those vertical velocities are essentially zero throughout the range, as the measurements are done very close to one another. The third plot of vertical velocities should be nonzero, but there is not really an expected trend.
- Some additional comparisons/fine tuning can be done (though probably off the ice) by using commands of similar structure to lines (16-23) specifying which bursts to compare and whether or not to average the files that contain those bursts.
- Furthermore, the file fmcw_process_config_vsr.m contains many tuning settings (explained in the file itself) that can modify the results. I have left it as is for the time being.