Python Program for Mitigating Radio Frequency Interference Observed in SpectraCyber Receiver Drift Scan Data Files

Jon Ayres

Deep Space Exploration Society

Abstract

The Colorado Springs Deep Space Exploration Society (DSES) is developing capability for collecting 21-cm HI Line drift scan data on the club's 60-foot parabolic dish antenna. Initial efforts have collected drift scans using a 9-foot parabolic dish interfaced to a SpectraCyber I receiver. These initial scans show that HI Line data is being collected; however, the desired data includes corrupted samples caused by numerous Radio Frequency Interference (RFI) events and also exhibits signal level offsets thought to be due to component temperature changes. This initial progress report demonstrates and describes a method for automated removal of transient RFI events using a Python program that is part of the DSES drift scan development effort. This report also outlines future goals.

Radio Frequency Interference Observed in Drift Scan Data

Numerous RFI events have been observed in the drift scan data files that were collected for mapping HI Line emissions of the Milky Way Galaxy. A Python program for removal of the RFI has been written and has yielded successful mitigation of one type of RFI. Figure 1 shows spectrogram plots of drift scan data; in this case, collected at +30 degrees of declination and over the complete range of right ascension. It also includes plots both before and after RFI mitigation has been applied. The SpectraCyber receiver was configured to scan in 5kHz steps from -600 kHz to +595 kHz HI Line offset frequency.

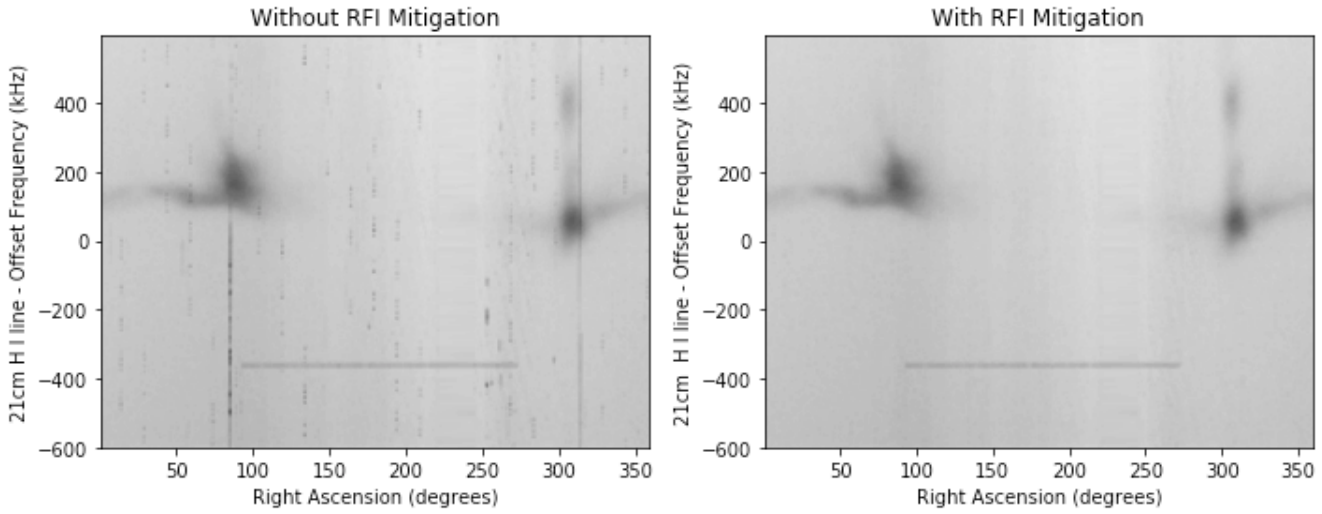

Figure 1 – Plots of: 2019-0923_9ftDishDriftScanData/DEC +30 Drift Scan/

The wispy, dark areas represent the desired HI Line signal; as areas go towards black, increasing HI signal levels are indicated over a range from 0-10 volts from the SpectraCyber receiver. RFI typically manifests as a streak of abnormally-high samples that show up as ragged line segments (streaks) above the background level. Numerous vertical RFI streaks are visible in the Figure 1 (without RFI Rejection) spectrogram. The brightest vertical streak RFI occurs at a right ascension of approximately

85 degrees and spans over the entire range of offset frequencies. For this particular RFI event, the strongest RFI spans from -500 to +50 kHz frequency offset; however, faint levels of RFI are visible over the entire range of offsets.

Vertical RFI streaks represent transient events that cover a range of offset frequencies for a fairly short period of time and then disappear. However, they may appear again at another right ascension coordinate and over a range of offset frequencies. The source of the RFI is not known and may be due to more than one source.

Another category of RFI appears over a longer duration and persists over a narrow range of offset frequencies. RFI in this category shows up as a horizontal streak, with an example shown in Figure 1 at -350 kHz frequency offset and occurring over 90 to 270 degrees of right ascension. As of this report, the Python program does not mitigate horizontal-streak RFI; this will be addressed in a future Python program version.

Yet another category of RFI appears to be caused by a signal that is too strong[1] for the receiver's front end amplifiers and pushes the input signal circuitry into compression. When this occurs, the noise floor can be pushed below the input range of the receiver's Analog/Digital Converter (ADC), resulting with zero values being recorded in the scan data file[2]. The interfering signal may occur at frequencies outside of the receiver's measurement Intermediate Frequency (IF) pass-band[3]. Compression / interference throws the measuring equipment into a non-linear response range and renders the data collected as invalid. Figure 2 shows spectrogram plots of scan data that is thought to have been corrupted by front end compression.

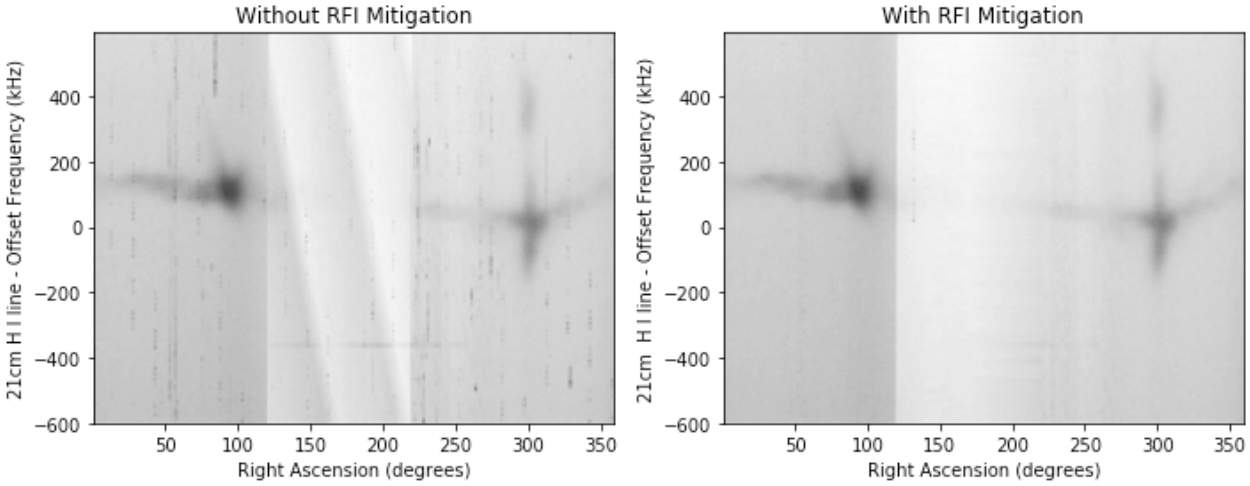

Figure 2 – Plots of: 2019-0923_9ftDishDriftScanData/DEC +20 Drift Scan/

1 The effects of front-end overload are under investigation; signal-compressing RFI could explain what is observed in some scan data, and its presence has yet to be confirmed.

2 The observed behavior might be caused by in-band high-level signals above the ADC input range that appear as near-zero values due to ADC output wrap-around.

3 The SpectraCyber receiver's measurement IF bandwidth is 30 kHz and the input signal is scanned across the -600 to +595 kHz measurement band by tuning a local oscillator in 5 kHz steps. The SpectraCyber scan file may not directly record nearby interference outside of its measurement IF bandwidth, but the receiver system could nonetheless be affected by strong signals.

The range from 120 to 220 degrees of right ascension as shown in in Figure 2 (without RFI Mitigation) has diagonal “ridges” that travel through the plot from upper-left to lower-right. Also, there are no high peaks depicted anywhere within the range of ridges. The interfering signal could be outside the -600 to +595 kHz offset frequency measurement range, yet might be a level sufficient to compress the noise floor below the minimum input range of the ADC. The presence of zero-valued samples in the dataset has been confirmed through examination of values over the range with compressing RFI.

Approaches for RFI Mitigation

Removal of RFI requires several approaches that are tailored to the characteristics of the specific RFI type to be mitigated. Current approaches implemented or considered are discussed in the following three Sections.

Short Duration RFI

Filtering of short-duration RFI requires samples of RFI-free data to be present over the same region containing RFI-corrupted samples. Collection of a suitable dataset requires multiple drift scans to be collected over the region to be filtered. The complete dataset may include numerous short-duration RFI events; however, if RFI-free data is also present over a given region, the RFI in that region may be identified and removed using a decision threshold for detection followed by rejection of the RFI-corrupted samples.

Frequency-Persistent RFI

A local RFI emitter near the 9-foot antenna has been identified and determined to be one source of RFI that occurs at a single frequency and over long duration. Frequency-persistent RFI will be re-assessed after capturing new sets of scan data, making sure that the known RFI emitter is disabled. If frequency-persistent RFI is still present after all accessible emitters have been silenced, it could possibly be removed by improvements to the Python program.

Front-End Overload

For the current overload mitigation approach, any 4-minute scan files that include data falling under a minimum threshold are rejected because data collected is likely to be under signal compression and will have compromised accuracy. This approach works for the existing 9-foot antenna data because its characteristic lowest value is typically above the (near-zero) threshold. However, this method would not be practical for systems in general, since threshold levels would need to be manually tailored for the system in use.

A better approach for detecting front-end overload will automatically establish the minimum level for the noise floor. It will also reject any scan file containing one or more samples that fall below an established threshold set in relation to the minimum-expected noise level. Minimum level characterization requires an analysis of the noise floor deviation with offset frequency; in the current 9-foot antenna system, signal level variation will need to be corrected before implementing minimum level characterization.

One set of scan data files at 0 degrees declination had values that were at a maximum level (near or at 10 Volts) for all samples in the affected scan data file. To address this overload case, the Python code

includes a decision threshold that rejects such a file whenever abnormally-high samples are found. Figure 3 shows spectrogram plots for the 0 degree declination dataset and the result of RFI mitigation. The vertical black bar in the first spectrogram plot has been removed.

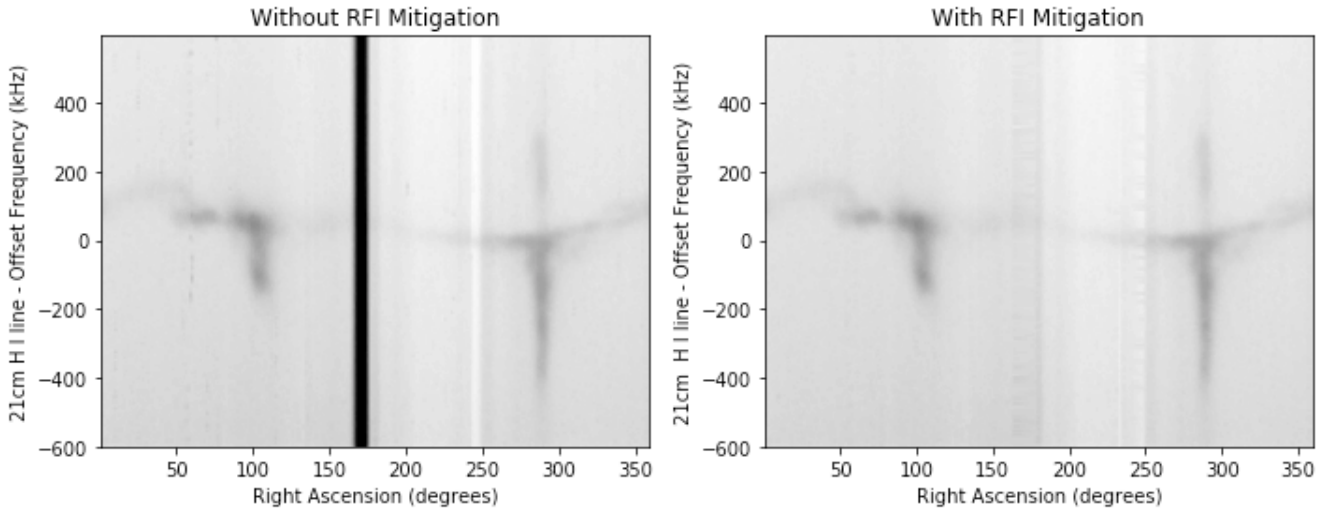

Figure 3– Plots of: 2019-0923_9ftDishDriftScanData/DEC +00 Drift Scan/

Signal Level Variation

In general, all plots of the drift scan data show considerable signal level variation, which is thought to be a result of component temperature changes. Signal level variation is easily observed as vertical banding within the background level of the typical spectrogram plot. DSES will implement hardware changes to address the variation. Dicke switching and thermo-electric cooling are currently being considered.

Python Program Output

The output of the program includes spectrogram plots for each declination, showing before and after RFI mitigation similar to those plots depicted in all Figures. In addition, the RFI-mitigated spectrogram's values are stored as an Enhanced Character Separated Values (ECSV) file for each declination's dataset. A set of ECSV files containing spectrogram data with RFI still present is also provided in a separate output folder. The ECSV (text) file may be used to post-process the spectrogram data using Astropy, and may also be easily imported into a Microsoft Excel spreadsheet for analysis.

One advantage of the spectrogram plot output is that the user can quickly assess the quality of the scan data files. For example, one can determine, through quick visual inspection of the spectrogram plot, that a region of the data should be re-scanned because high-valued vertical bars are present – indicating an overload condition. Finding the range of affected right ascension coordinates may be estimated from the spectrogram plot and the precise location may be determined though examination of ECSV file content. Once the files are identified, they may be deleted and replaced with new scan data files that cover the region.

Python Program Description

The Python program used to perform the RFI mitigation is provided separately from this paper and will be made available on github. A detailed code description of the program will also be provided on github as a readme file, along with an example dataset for use with the program. In its current form, the program is hard-coded to work with SpectraCyber scan files that are collected over multiple days and using fixed receiver parameters. Future updates to the program will add flexibility for more general use with the SpectraCyber receiver as well as improvements to RFI detection and removal.

The Python program was developed on a computer running the Ubuntu 18.04 Linux Operating System. The Anaconda Python distribution was used for code development and execution; Anaconda is freely available for MacOS, Linux and Windows, and greatly simplifies access to many useful Python libraries, including those required for the program.

The initial dataset used for analyses described in this paper includes the collection of drift scan data files for each of 11 declinations. A typical scan data file is collected over a four-minute period and includes 240 frequency offset samples. On average, 440 scan data files were collected for each declination, with a total of 4842 files processed. The program's RFI-mitigation processing of the 4842 files takes approximately 3 minutes using the development computer, which incorporates an Intel i7, 8th Generation processor with 32GB of RAM.

Future Goals

Of primary importance is the correction of the signal level variation that has been observed in the initial dataset. This correction needs to be implemented in order to successfully meet goals that are described below, in addition to the implementation of noise floor minimum characterizations that were discussed earlier.

Another goal is to package the multiple declination datasets into a single data file in Flexible Image Transport System (FITS) format. This spectral cube FITS file will consist of a three-dimensional representation of the data that will provide an easy means for data extraction along any desired set of axes. Once available, the FITS file will allow for a large number of Python's analytical tools to be applied for data analyses. A future version of the code will provide results as a FITS spectral cube in addition to the spectrogram plots and ECSV files already available.

Additionally, there is interest in developing a version of the Python program that will process wideband spectra from a Radio Astronomy Software Defined Receiver (RASDR). A system consisting of a RASDR interfaced to a 12-foot parabolic dish is available and this system offers the platform for development of the RASDR software version.

Conclusions

The Python program offers a method for automatically rejecting short-duration RFI from a dataset and provides useful spectrogram plots and ECSV data files for quality assessment and post processing. The original goal for the code was to greatly reduce the tedium involved in manually sifting through numerous scan data files to identify and remove RFI-corrupted samples.