SNR Snow Outlier Removal Code

Introduction

The purpose of this code is to process a set of daily average SNR data in order to detect and remove data outliers that are possibly caused by snow, for example snow covering the antenna or otherwise interfering with the SNR data. To remove snow outliers, this code processes a set of SNR data by removing both user prescribed data points and by calculating outliers to remove through a series of methods. After removing user prescribed points, applying a mask to remove points that are too high or too low, and removing offsets caused by firmware changes or other sources, three types of outliers are removed. First a simple standard deviation mask is applied to the whole data set to remove obvious outliers. Second, the summer data points for each year are used to remove outliers throughout the remainder of the year that are too low. Finally, the code removes statistical outliers as calculated by using a least squares solution to determine a sinusoidal fit to the yearly temperature and SNR oscillations.

At each intermediate step, the SNR data and segment parameters are stored to allow for each different data set to be plotted and compared to the original raw data.

Contents

1. Getting Started

1.1. Running Demo Mode

To run the code with the included data sets and default parameters, simply run the demonstration mode. To run the demo mode, perform the following command in MATLAB from the subfolder containing the main code:

snr_outliers_main('Mode','Demo')

This will cause the code to enter the demonstration mode and use the included lists and data sets to run through the code. If other parameters to change the code performance are desired, they can also be input with the demo mode as described in section 3.4.4 below.

1.2. Demo Mode Examples

In the demonstration mode, the code is run on three different stations for two different sources of SNR data each. The first set of data included is the L1 SNR data set while the second set is the L2 data set. The L1 data sets are not processed by the code (but this can be changed by changing input parameters), and the L2 data sets are processed by using each of the different outlier removal segments. For each station in the demo list, there are two plots created for each station. The first plot compares the raw SNR data for the two different SNR data types as shown in Figure 1.

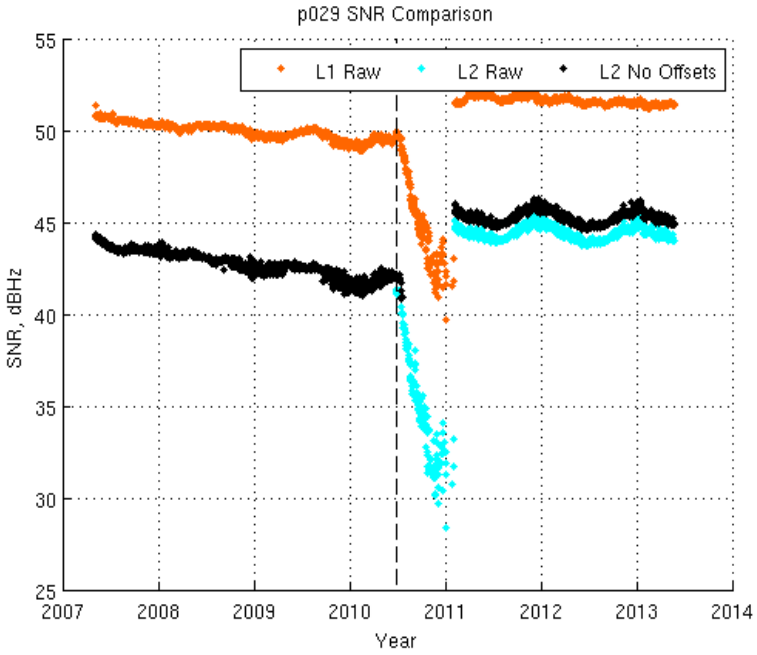

Figure 1 P029 data set comparison

The first plot also shows all the offsets removed from the raw data before it is processed by the remainder of the code. Each offset that is removed is marked in the first plot by a vertical line and the results of removing all the offsets for the L2 data set are shown on top of the raw data. The user prescribed bad data points and the points removed by the SNR mask are removed before removing the offsets so they are also not included in the data set with no offsets.

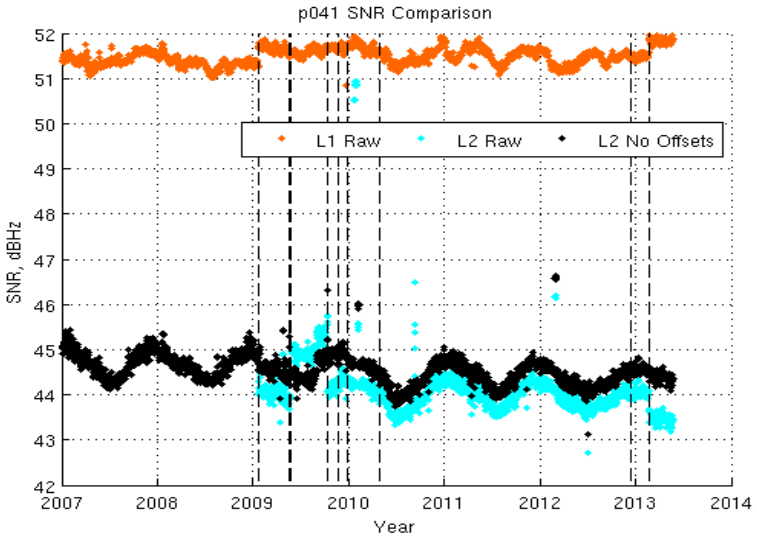

Figure 2 P041 data set comparison

Note that for *P029* there is only one offset that is removed and in the case of *P041* in Figure 2, there are nine different offsets that are removed. The different offsets are due to firmware changes or in the case of *P041* there is one additional offset that does not come from a firmware change.

After removing the user prescribed bad data points, and offsets, the L2 data are processed by performing each of the three main outlier removal segments. In the second plot created for each station, each segment has a separate subplot showing the bad points that are removed based on that segment's removal criteria.

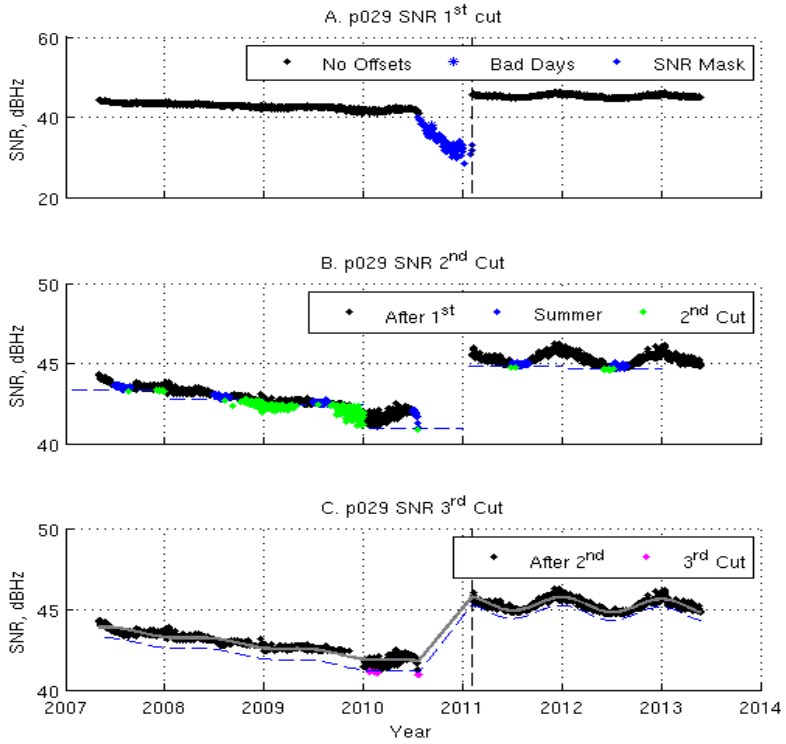

Figure 3 P029 outlier removal segments

Each subplot first shows the SNR data at the beginning of the segment and shows the different

bad data points prescribed by each segment as well as showing some parameters used for each segment. For the first and third segments, each section of data between antenna and receiver changes is read separately. These sections are marked by a vertical line in both the first and third subplots. The second subplot shows the points determined to be "summer" as well as showing a horizontal line representing the minimum value in the summer. *P041* and *P150* are good examples of summer calculation and outliers. The first four years of *P029* show that the SNR are steadily decreasing so that many data points are removed at the end of the 2008 and 2009.

2. What the Code Does

2.1. Loading Parameters

Before processing any sites, the system first loads in any input variables and the remaining default constants and format parameters used to process a site. The optional input variables allow for a specific instance of the code to be performed with different parameters. The constants and format parameters are each loaded in a wrapper function that allows for quick changes to the more permanent default values used by the code.

2.2. Loading Lists

After the parameters have been loaded, the code uses a set of wrapper functions to load a list of bad days, a list of sites not to use, a list of sites to use, and information about each site's hardware.

First, the code loads a list of bad days. The list should include both bad global days to be used for all stations and bad local days that correspond to problems with one particular station for a small period of time. Second, the code loads a list of sites that should not be used. This list is used to screen out sites that are unwanted when loading the list of all the sites. Third, the code loads a list of all the sites to use. Any site that is not in the bad sites list will be included and processed. If a specific list of sites is passed in, it is used exactly as it is passed in. If a bad site is passed in it will still be used. Fourth, the code loads information about the changes made to the antenna, receiver and firmware for each site that will be used. Finally, the code loads in a list of offset days that are not tied to firmware but are still present

2.3. Process All Sites

If the list of sites has at least one site in it, the code will loop through each site and process a single site at a time and return the results of the outlier removal. For each site, the code will loop through all possible SNR data set types and is the process toggle is on then the data type will be processed with the following steps.

2.3.1. Load SNR data

First, the code loads all the raw SNR data for each site. The code will attempt to load data for each year in the entire SNR year span. If there is at least one data point, the code will attempt to process the data set.

2.3.2. Main Processor

The main process performed on each site consists of six overall segments. The first two remove data points based on specified dates and an SNR mask, the third adjusts for firmware offsets, followed by the three main data filters used to remove snow outliers. The different segments can all be toggled on/off individually and are all described below.

Remove bad days

The code first compares the data set to the list of bad days that was previously loaded. If there

are any days that are on either the global list or the local list, the daily data point will be removed.

Remove SNR Mask

The code removes SNR values that are above or below the specified max and min values respectively.

Remove offsets

Next, the code removes all the offsets in the offset list. There are two types of offsets in the list: firmware changes and other offsets. The firmware offset days are taken directly from the site change information. The other offsets are taken from the list of additional offsets. The two lists are combined then all offsets are removed. The offsets will be removed by adding the correct bias to the SNR data that are after the switch. No data points are removed in this stage of the outlier removal.

1st cut (Sigma outliers)

The first of the three main outlier removal steps involves simply removing statistical outliers. The code calculates the mean and standard deviation of the SNR data and removes data points that are too many standard deviations away from the mean.

2nd cut (Summer defined outliers)

The second outlier removal step involves calculating the summer period for each year and finding the minimum SNR value for each summer. The SNR values are lowest during the summer when the heating of the receiver and antenna both cause increased noise. The outliers calculated for a particular year are then determined to be those points in a year that are lower SNR than the lowest value during the summer.

3rd cut (Sinusoidal Sigma outliers)

The final outlier removal step involves fitting the yearly data to a sinusoid. The seasonal temperature change cycle is estimated using a sinusoid with a period of one year and this approximation is used to create a least squares estimate for the seasonal trend of the SNR. With the estimated fit, the difference between the data points and the estimate are calculated and the standard deviation is calculated. Those data points which vary from the estimate more than the desired standard deviation tolerance are removed.

2.3.3. Plot Data

After the site has had the three different types of outliers removed, the results of any and all steps can be plotted to show the effects the outlier removal had on the raw data. The raw data and all the bad data points are shown as determined by each different segment. The final data set without any outliers is shown for comparison.

3. How to Use the Code

3.1. Data Set Requirements

The SNR data set used by the code must have one SNR value for a single day of the year. The data should match the format of the included demo data sets. To load a different format, the wrapper *load_snr_data* must be modified to ensure that the data are loaded correctly.

3.2. Extra List Information

The local and global bad days lists, site change list, and offsets list are all extra lists that are not required for successful operation of the code. If these lists are empty, the code will still perform the SNR outlier removal but the results may be unexpected. Any information should be included in these

lists if possible.

3.3. Flexibility

The code has many different flexible options that allow different portions to be changed or removed altogether by changing inputs to the code rather than modifying the code itself. These options come in the form of different constants and format type inputs. Different segments can have their parameters like the sigma tolerance or summer day range changed. The segments can also be turned on or off completely with segment toggles. The different segments update a running data variable to keep the most up to date SNR data as well as storing the segment outputs for later plotting. Because of this, turning a segment on or off does not affect the functionality of the following segments. These inputs can be changed at run time by using the *constants* and *format* inputs for a one time change or the default can be changed in the constants wrapper.

3.4. Global Variables

There are three global variables that are used throughout the code suite. These global variables allow for information to be passed between functions without the use of a series of inputs and outputs. The global variables are also accessible after the code has finished processing data and therefore allows the data to be retrieved after the code has completed. The three global variables are *constants*, *format*, and *sites_list*. The *constants* global stores all the constants used for data processing as described in 3.4.1. The *format* global stores all the formatting variables like the verboseness of the output and the different style variables used for plotting the data.

3.4.1. Constants

There are a number of different constants that can be input into the system to change the performance of one or more segments without changing the code itself. Constants include SNR data constants, segment toggles, and segment tolerances. The list of variables in the *constants* structure is shown in Table 1.

Table 1 Code Suite Constants

Constant	Default	Description
demo_mode	false	Logical toggle to run demo_mode
Filenames		
bad_sites_file	/demo/bad_sites_list	List of sites not to process
global_bad_days_file	/demo/global_bad_days	List of days that are bad for all sites
local_bad_days_file	/demo/local_bad_days	List of days that are bad for individual sites
site_change_info_file	/demo/site_change_info	List of equipment changes for individual sites
sites_list_file	/demo/sites_list	List of sites to process
offset_file		List of offsets at individual sites
Segment Toggles		
remove_bad	true	Remove bad days(global and local)
remove_mask	true	Remove SNR mask
remove_offsets	true	Remove offsets (firmware and other)
remove_cut1	true	Remove 1st cut (sigma outliers)
remove_cut2	true	Remove 2nd cut (summer outliers)
remove_cut3	true	Remove 3rd cut (sinusoid outliers)
save_data	True(false for demo)	Save data to an output file
make_plots	True	Create plots

save_plots	True(false for demo)	Save plots to an output directory
Preferences		
Sid_length	4	Number of characters in a site ID
case_sensitive	false	Case sensitivity for comparing site IDs
equal_yrs	True	Treat all years as average
yr_length	365.25	Length of an average year
SNR Data Properties		
snr_types	2	Number of different data sets to use
snr_yr_min	2007	Earliest data year to process
snr_yr_max	2013	Latest data year to process
snr_yrs	2007:2013	Range of years to process
Predeclared Lists		
bad_sites	{}	Cell string of bad sites
global_bad_days	[]	[year doy year.dec] bad
local_bad_days	{[];[]}	Cell array of bad
local_bad_sids	{}	Cell string of sites IDS corresponding to bad days
Tolerances		
bad_snr_min	40	Minimum mask value
bad_snr_mak	50	Maximum mask value
offset_range	10.5/366	Range of days to use for calculating offsets
offset_min_pts	3	Minimum number of points needed for an average
cut1_outliers	[true;false]	[low;high] remove offsets below/above min
cut3_outliers	[true;false]	[low;high] remove offsets below/above min
cut1_sigma	3	Number of standard deviations away from the mean
cut3_sigma	3	Number of standard deviations away from the mean
summer_range	[180;240]	Doy values for the summer end points
summer_min_pts	2	Minimum number of points for a summer
summer_min_index	2	Index of sorted snr values to use as the min

3.4.2. Format

The *format* variable structure contains different output and style variables to alter the way that the code results are displayed.

Table 2 Code Suite Format

Format		Default	Description
text_out		1000	Integer describing how much to print to standard out. 0,1,10,100,1000 (1000 is everything)
font_size		10	Font size on plots
plot_x_lim		[]	Input x axis limits
line_style		‘.’	Default line style for data points
marker_size		6	Default size of data points marker size
color			Structure of different colors to use for plotting
	.raw	[1;.4;0],[0;1;1]	3x2 array of RGB values for the 2 different raw

			data sets
	.bad	[0;0;1] [blue]	Default color for all bad data points
	.bad_days	[0;0;1] [blue]	Color for days marked as bad
	.bad_mask	[0;0;1] [blue]	Color for days removed by the SNR mask
	.bad_cut1	[1;0;0] [red]	Color for days removed by the 1st cut
	.bad_cut2	[0;1;0] [green]	Color for days removed by the 2nd cut
	.bad_cut3	[1;0;1] [magenta]	Color for days removed by the 3rd cut
	.no_offsets	[0;0;0] [black]	Color for data after offsets are removed
	.summer	[0;0;1] [blue]	Color for all data points used for each summer
	.summer_min	[0;0;1] [blue]	Color for line marking the minimum value for each summer
	.cut3_fit	[.5;.5;.5] [gray]	Color for the sinusoidal fit line
	.cut3_sigma	[0;0;1] [blue]	Color for the sinusoidal outlier line
	.data	[0;0;0] [black]	Color for the final processed data set
	.offset_marks	[0;0;0] [black]	Color for lines marking offsets
	.equip_marks	[0;0;0] [black]	Default color for lines marking equipment changes
	.bad	[0;0;1] [blue]	Default color for all bad data points
	.bad_days	[0;0;1] [blue]	Color for days marked as bad
	line_style_bad	'.'	Default line style for bad data points
	line_style_bad_days	'*'	Line style for bad SNR days
	line_style_summer	'.'	Line style for the summer points
	line_style_summer_min	'--'	Line style for the summer min line
	line_width_summer_min	1	Line width for the summer min line
	line_style_cut3_fit	'-'	Line style for the sinusoidal fit line
	line_width_cut3_fit	2	Line width for the sinusoidal fit line
	line_style_cut3_sigma	'--'	Line style for the sinusoidal outlier line
	line_width_cut3_sigma	1	Line width for the sinusoidal outlier line
	line_style_offset_marks	'-.'	Line style for the offsets lines
	line_width_offset_marks	1	Line width for the offsets lines
	line_style_equip_marks	'--'	Line style for the equipment change lines
	line_width_equip_marks	1	Line width for the equipment change lines

3.4.3. Sites List

The *sites_list* variable stores all the data correlating to each of the sites that are processed in a single structure array. The SNR data are stored in a substructure throughout the entire process allowing for later plotting and retrieval of the data.

3.4.4. Inputs

Inputs for the code come in pairs and the pairs can be passed in any order desired. There are four different input pairs that can all be passed or not. Each pair has a string identifier first and the expected input second. By passing in inputs the code can, enter demo mode, use a non-default format or constant, and use a specified list of sites.

Demo Mode

Input 'Mode' followed by 'Demo':

snr_outliers_main('Mode','Demo',...)

The Demo mode changes the filenames of files to load and even load site change information

differently. Until the code has been modified to fit with the desired architecture as described below in 4, demo mode is a necessary input.

Constants

Input 'Constants' followed by a constants structure that has any or all of the desired constants to change.

constants.remove_cut3=false;
snr_outliers_main('Constants',constants,…)

Any value from the Table 1 can be input as part of the structure and the input value will override the default. This is useful to re run the code with a single parameter changed without having to modify the *load_constants* function for a single run of the code.

Format

Input 'Format' followed by a format structure.

format.text_out=1;
snr_outliers_main('Format',format,…)

Similar to the constants input, the format input can take any or all of the values from Table 2 and have them override the defaults. This is useful for changing the verbosity of the text output or changing colors used for plot formatting.

Sites List

Input 'Sites' followed by a cell string of sites to use:

list={'p041';'p150'};
snr_outliers_main('Sites',list,…)

The cell string input *list* is a cell string of sites that override the site list and the bad site list. The code will only process the sites in the input list and no others.

Combining Inputs

The inputs can be combined in any order as long as the pairs remain together. For instance the two following function calls produce the same effects.

snr_outliers_main('Mode','Demo','Sites',list)
snr_outliers_main('Sites',list,'Mode','Demo')

Both of the preceding function calls would process only *p041* and *p150* and would look for data in the demo mode default location.

snr_outliers_main('Mode','Demo','Sites',list,'Constants',constants,'Format',format)

The above call would process only the two stations but not remove the third segment and would print less information to the screen than normal.

3.5. Outputs

After processing the SNR data, the code has the option to save the output data to a simple file and saving the plots to

4. Modifying the Code to Integrate with an Existent System

The code was written to use the large amount of data available for a system in place while still keeping in mind the need for adaptation to new systems. Different function groups have been created to help ease this transition and allow for different systems to successfully use the code suite.

4.1. Wrappers

Different wrapper functions have been created to help make integration easier. These wrappers take the different system interactions into a separate function so that only the function code needs to be

changed while still providing the same functionality for the entire code suite. The different wrapper functions allow for the MATLAB path to be changed, different methods of returning SNR data, loading information about different stations, and reading lists of bad days for individual stations. Any of the wrappers can be modified to change the way the code suite interacts with the system.

4.1.1. Modify Path

The *modify_path* function allows for the MATLAB path of different system directories to be added to and removed from the path. If they are not already included in the path, this function adds the different subdirectories inside the suite code folder. To change which path the MATLAB suite uses for external MATLAB codes, simply change the system directory variable or if that directory is permanently in the path simply remove the system directory segment.

4.1.2. Load Sites List

The *load_sites_list* function allows the system to read a list and setup the list of all the sites that are to be processed. If a desired list of stations already exists in a different format, this wrapper should be modified to use the existing list. Care should be taken to ensure the output list still has the same expected format but the steps taken to generate a list can be changed.

4.1.3. Load SNR Data

The *load_snr_data* function allows the system to read SNR data from a specified file. If SNR data are stored differently, then this wrapper should be updated to load the data for a given station. Data can be loaded by year or all at once as long as the output format matches the prescribed format structure.

4.1.4. Load Site Change Info

The *load_site_change_info* function allows the code suite to load site change information from a list or from some other system source. If there is already a preferred method to load information about the antenna, receiver, and firmware changes for a station, then this wrapper should be modified to include those changes. The output structure should match the specified output structure.

4.1.5. Load Bad Days List

The *load_bad_days* function allows the code suite to load a list of both bad local and bad global days for SNR data. The *read_local_bad_days* function should match to the cell string and cell array formats specified and the *read_global_bad_days* should match the simple array format specified. However, neither of these two additional reader functions are necessary as long as the outputs for the *load_bad_days* function is matched.

4.1.6. Load Bad Sites List

The *load_bad_sites* function loads a list of bad sites that should not be included when reading the overall list of sites. This allows a much larger system database list of sites to be used for the overall list but for the undesired stations to be excluded from processing. As with the other wrappers, if there is a more convenient method to load this information, this function should be modified.

4.2. Plotters

The plotting functions are designed to plot different steps in the code process but for many cases are basically the same. However, any of the codes can be modified to change how different sections are plotted if the desired change is more than a simple formatting change as described in section 3.3. The overall process of plotting is performed by the *main_plotter* function.

4.2.1. Main Plotter

The *main_plotter* function creates the plots for each station and should be modified to change what gets plotted and how it appears when all the plot routines are combined. In the demo mode, the first SNR data set is plotted against the raw for the second set and this should be changed if the number of data types is changed. If the subplot outputs are desired to be individual plots, that should be changed in this function as well.

5. Version History

Version	Date	Description	Author
1.0	June 13, 2013	Initial Release	Kyle Wolma