

ULFEM TIME SERIES ANALYSIS PACKAGE

K. Neil Kappler

August 2, 2010

Contents


1	Overview	2
2	QUICK START GUIDE	2
2.1	Calling ULFEM	2
2.1.1	typhoon	2
2.1.2	Other Machines	3
2.2	Downloading 1 Hz Data	3
2.3	Downloading 40 Hz Data	4
2.4	Plotting Data	5
2.5	Saving Plots	5
3	INSTALLATION	6
3.1	Windows	6
3.2	Unix/Linux	6
4	FAQ	7
5	DOWNLOADING DATA	7
6	PLOTTING TIME SERIES	7
6.1	Description of Individual Routines	8
6.1.1	PlotManager.m	8
6.1.2	loadPlotCB_kk.m	8
6.1.3	plotTScbk.m	8
6.1.4	mkShort.m	8
6.1.5	Control of Individual Yaxes Lims in TSplotter	8
7	METADATA STRUCTURE and HANDLING	9
7.1	Automated Metadata Handling	9
7.2	Spreadsheet (manual) Metadata Handling	10
7.3	COILS	10

1	OVERVIEW	2
8	KNOWN BUGS	12
9	FREQUENCY DOMAIN DATA	12
9.1	Decimation	15
10	ACKNOWLEDGEMENTS	23
11	APPENDIX	23
11.1	PUBLIC-PRIVATE KEY SETUP	23
11.2	CODE	24
11.2.1	ulfemToolbox Remote Directory Contents	24
11.3	Signals and Instruments	25

1 Overview




This manual describes how to use the ULFEM software package. Casual users can read the quick start guide and will probably not need any more information than this. For users who may wish to modify the code, further description of the routines are provided.

2 QUICK START GUIDE

This guide lists commands entered by the user to run ULFEM. Text to be entered on the command line is prefaced by >. The  symbol means press Enter or Return.

2.1 Calling ULFEM

2.1.1 typhoon

1. Sit at typhoon and log in as ULFEM or
Remotely login to typhoon using ReflectionX or similar package which emulates unix xwindows.
2. Right-click the mouse and select “Open Terminal” from the drop down menu. A terminal window will appear.
3. > cd /gp/ulfem/ULFEM/matlab/ 
4. > matlab & 
5. In the MATLAB window
> ULFEM 
The ULFEM master GUI will appear in the lower left of your screen. The matlab window will display a few messages about the paths it is setting. Disregard these for now.

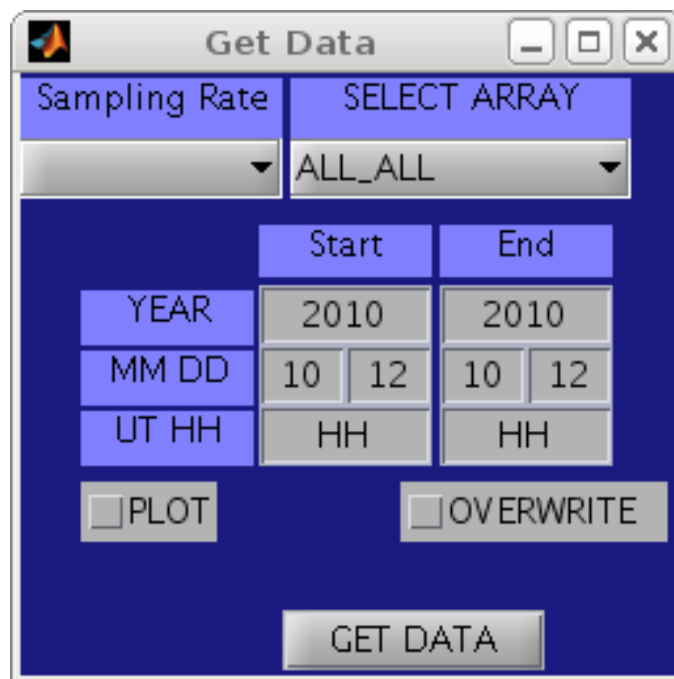


Figure 1: The Get_Data GUI

2.1.2 Other Machines

1. Open a terminal window (or a cygwin window if using MSWindows)
2. cd to the ULFEM/MATLAB/ directory.
3. Call Matlab: `> matlab &`
4. In the MATLAB window
`> ULFEM`

The ULFEM master GUI will appear in the lower left of your screen. The matlab window will display a few messages about the paths it is setting. Disregard these for now.

2.2 Downloading 1 Hz Data

1. From the ULFEM master GUI, push the “GET NCEDC” button.
 A GETTS window appears. like the one shown in Figure 1.
2. In the GETTS window, from the “Sampling Rate” dropdown menu select “L”
3. From the ARRAYS dropdown menu, select an array. There are preloaded arrays for each individual site, as well as an array called BAYAREAMAG which are the nine coils at JRSC, BRIB, and MHD. Selection of an individual array is the way

to specify the collection of channels you wish to download. Alternatively, one can download any channel individually, but usually channels are downloaded en masse according to a standard list (Array).

4. In the YEAR, MM, DD edit boxes, enter the start time. In the MM box use 1-12 for month corresponding to Jan-Dec. In the DD box use 1-31 for day of month. For 1 Hz data the UTHHMM field will default to 00:00 and 24:00 since whole days are downloaded by default. Select the PLOT checkbox to see the data plotted as they download. Push the “**Get-Data**” Button. The code will download data, one-day-at-a-time from your start time to your end time. If you only want one day of data, leave the end day equal to the start day. 1Hz data always download from midnight to midnight UT. To allow variation in starttime, comment out lines 210, 212 of GetNCEDC.m where the edit box is set to a text box.

If a download has been started in error and you wish to abort the process, place cursor in the matlab command window, press Ctrl, and then hold the C key down. The matlab command line should be restored. In the event that the process has runaway, or is hanging and matlab is unresponsive, open a unix window and type: `ps -e — grep MAT`. A line will appear which starts with the process ID of the matlab window. Enter the command: `kill -9 PID`, where PID is the process ID output from the previous command. You will need to restart matlab.

2.3 Downloading 40 Hz Data

1. From the ULFEM master GUI, push the “GET NCEDC” button. A GETTS window appears.
2. In the GETTS window, from the “Sampling Rate” dropdown menu select “B”
3. From the ARRAYS dropdown menu, select an array. There are preloaded arrays for each individual site, as well as an array called BAYAREAMAG which are the nine coils at JRSC, BRIB, and MHDL.
4. In the YEAR, MM, DD edit boxes, enter the start time. In the MM box use 1-12 for month corresponding to Jan-Dec. In the DD box use 1-31 for day of month. The user enters integers corresponding to the starting and ending hour of the desired data window in the UT HH field. Data are downloaded in 2-hour windows. If a specified window is less than 2 hours, a two-hour time series will still be delivered, from *start* UT HH to *start* UT HH+2. Thus if 0,1 are entered as the start and end time, a time series spanning 00:00-02:00 will be downloaded. To download a whole day at a time, enter 0, 23 as the UT HH start and end respectively, or enter 0,24. In order to avoid confusion about open and closed set boundaries, a tiny amount (0.01) is subtracted from the end HH, so that downloading for example from 4 to 6 HH is not taken as two sets. The code will interpret 4,6 as 4, 5.99.

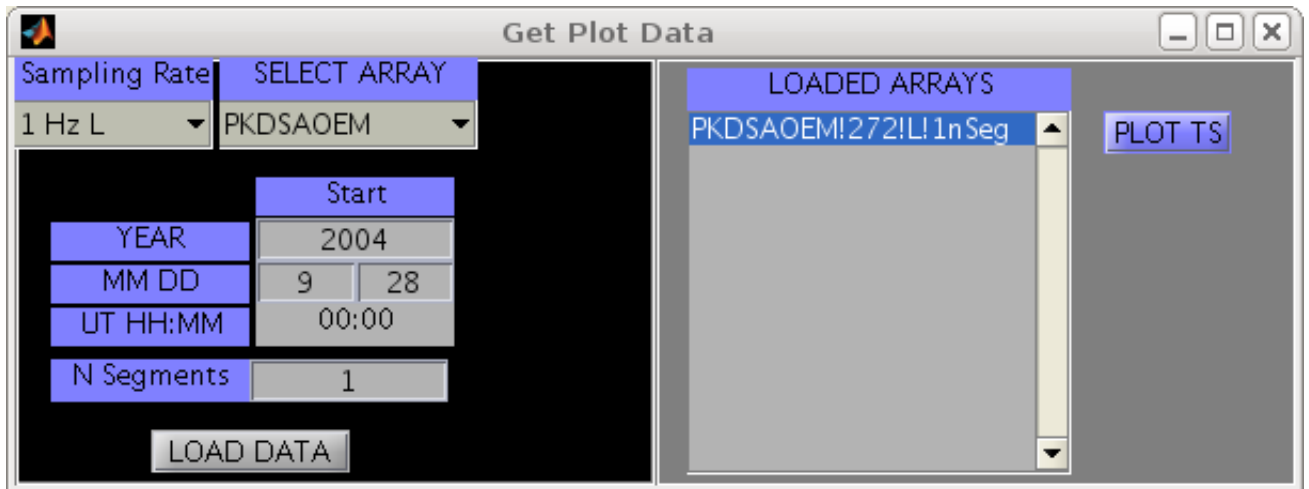


Figure 2: The GUI for calling data time series plots.

If one wants to download the same time window over many days (for example, from 2-4am), then just select 2,4 as the HH start and end, and then select the range of applicable days. After pushing “**Get-Data**”, the code will download data, one-2hr-window-at-a-time from your start time to your end time. If you only want one day of data, leave the end day equal to the start day.

2.4 Plotting Data

1. From the ULFEM master GUI select **Plot Time Series** button.
2. In the “Get Plot Data” window which appears, select your sampling rate, array, and day to load.
3. For nSegs enter “1”. Push enter while the cursor is in the nSegs window (see Known-Bugs).
4. Push the Load Data Button. Wait a few seconds. An array name should appear in the Right hand side window.
5. Select the array name and push the “**PLOT TS**” button.

2.5 Saving Plots

1. Select as active the window you wish to plot.
2. At the MATLAB command line, type:`pdfPlot('FILENAME');`
3. Plot will be saved to the FIGURES directory in the top level of your ULFEM folder.

3 INSTALLATION

PREINSTALL CHECKLIST

1. You will need an account at BSL (seismo.berkeley.edu)
2. Set up Public-Private Key Encryption between the machine you will install on, and BSL.

3.1 Windows

There are two ways to run the ULFEM code on a windows machine.

1. Using a unix emulator such as ReflectionX and working remotely.
2. Installing Cygwin and Matlab on the Windows Machine and working locally.

3.2 Unix/Linux

1. Choose a directory as the home directory. For example: /home/kappler/ULFEM/. Inside of this directory, place the file ULFEMMAT.tar. Then call


```
> tar -xvf ULFEMMAT.tar
```

 A MATLAB directory will be created.


```
> cd MATLAB
```

```
> matlab &
```
2. Check that paths are set properly:

Open the script setPaths.m, and add a new location. Just copy the 'GUMP' settings. At minimum you will need to set the variables: ULFEMpath, username, binStageDir, ascStageDir, metaStageDir, ulfemToolbox.

For the local directories, (ULFEMpath), follow the examples set on GUMP. For username enter your BSL login. The three staging directories and ulfemToolbox are folders on your BSL account. You need to create these three folders. Place them in your home directory or in a scratch directory for which you have read and write privileges. You do not need to place anything in the staging directories, but in ulfemToolbox, place the python scripts: *cleanAsciiStage.py*, *cleanBinStage.py*, and *cleanMetaStage.py*. Examples of these files are on the ULFEM website under CODE/AUXILIARY_FILES/ and in the Appendix of this document.
3. From the matlab command line, call setPaths.m.


```
>setPaths 
```
4. Place the file sr_list.mat in the "ULFEM/sys/" directory.

5. From the matlab command line, run `configureDataDirectories.m`
`>configureDataDirectories`
6. unpack `sitesFolderContents.tar` in `ULFEM/sys/SITES`.

4 FAQ

Q:Where are the data stored after I download them?

A:In `/gp/ulfem/ULFEM/data/`. Folders are organized by sampling rate and year. Inside the `SRYYYY` folder is a folder named `TS` (*Time Series*). The raw data are in there.

5 DOWNLOADING DATA

Section 2 explains the basics of data download. In some cases however, the user may request data when no data are available. In order to keep track of this, a 'dummy' TS file is created when no data are returned. Loading such a TS file yields a single data structure: *y*. *y.data*=NaN and *y.flag*="No Binary Data from Que".

6 PLOTTING TIME SERIES

The time series plotting software is based on the PKDSAO plotting package written in Matlab by Gary Egbert. It has been modified for more channels for the Bay Area USGS array. The matlab files used for the plotting software are listed below:

1. `PlotManager.m`
2. `loadPlotCB_kk.m`
3. `mkshort.m`
4. `plotPage.m`
5. `plotParams.m`
6. `plotTScbk.m`
7. `plotTSwin.m`
8. `rePltCh.m`
9. `rePltPage.m`

6.1 Description of Individual Routines

6.1.1 PlotManager.m

The *PlotManager* program is the GUI which allows the user to select the data to plot. The user chooses an ARRAY, sampling rate, start time, and the number of contiguous segments to load. A segment is 1 day when dealing with 1Hz and 2 hours when dealing with 40 Hz. Once these parameters are specified, pushing the 'LOAD DATA' button executes a call which loads the specified data into a matlab structure for plotting. Once the data are que-ed into structure, a line will appear in the righthand side of the GUI under the heading "LOADED ARRAYS". Select a particular loaded array with the mouse and press "PLOT TS" to generate a plot. Calls: *loadPlotCB_kk.m*.

6.1.2 loadPlotCB_kk.m

This is the Callback when an array is being loaded. It is comprised of a case-switch statement having cases: *Load* and *Plot*. These are callbacks from the *PlotManager.m* GUI. The *Load* portion of the code performs the operation of reading in the collection of channels which the user specified under PlotManager, and then storing the data as a temporary array in the folder SCRLOTpath. // The *Plot* portion of the code reads the temporary array and plots it using a variant of Gary Egbert's time series plotting package. Important variables in this routine are: TSd1, plotStart, and TSw1. Important subroutines which are invoked here are plotTSwin and plotPage.

6.1.3 plotTScbk.m

This code has around 630 lines and is mostly comprised of case-switch loops. The switch is on *action* which is the Tag of the calling button. The possible values for *action* are: chnumpt, Centered, replot, reset, zoomOut, zoomIn, zoomOutCh, zoomInCh, chonoff, set_mCH, set_MCH, PrevPage, NextPage, gotoPage, Quit, t0, tUnits, yUnits, Uniform ylim, Mark, Unmark, Zoom plot, Clear marks, McohPlot, pctOverlap, mVkm.

6.1.4 mkShort.m

This function simply cuts the yAxesLims down to 'short' numbers so that they will fit inside the edit box on the TS plot window.

6.1.5 Control of Individual Yaxes Lims in TSplotter

When the initial time series is read in in loadPlotCB_kk, the yAxes ranges are determined by an automated process, and stored as TSd1.range. The two functions to pay attention to for understanding the setting of yLims are the callbacks to the **REPLOTT** and **RESET** buttons. The **RESET** button reinitializes the yAxes Limits to their initial settings, and the **REPLOTT** button should set the yAxes lims to the user-specified values in the edit boxes. The callback to both buttons is *plotTScbk*. The actions which are input to plotTScbk

which are relevant are: `replot`, `set_mCH`, `set_MCH`, and `Uniform ylim`. Note `plotTSwin.m` creates `MCH` and `mCH`, but doesn't actually assign the values to the edit boxes. This is done in `plotPage.m`. The trouble with manually adjusting axes limits was addressed. It turns out that `TSw` was being treated as a `dataStruct` with `{ii}` following it in `set_mCH` and `set_MCH`. I am concerned that the `ii` may be because if running more than one timeseries plotter simultaneously, there may be some confusion on matlabs part... The actual handle for the editBoxes into which the `yAxesLims` are written is

7 METADATA STRUCTURE and HANDLING

Metadata handling was set up to be automated via a Metadata Download utility. This utility has been omitted in the short term because the metadata that we require to process the data are not all stored at BSL. Instead, a set of metadata spreadsheets have been created in `ULFEM/metaData/SPREADSHEETS/` which store the required data for simple processing.

7.1 Automated Metadata Handling

Not currently supported by BSL for all stations, the metadata handling is called from ULFEM using the "Update Metadata" pushbutton. The `UpdateMetaDataGUI` then appears. Once the sampling rates and Arrays are chosen from this GUI, the user pushes the "Start Update" Button, which calls "UpdateMetadata.m" subroutine. The `UpdateMetadata` program flow consists of 2 stages. For each channel (`ch`) and sampling rate (`sr`), the NCEDC database is queried for a list of EPOCHS where that particular `ch-sr` combo was in use. For example, some sensors were never changed since installation, and so there is only one epoch, but other sensor data were modified from time to time, either as a part of site maintainance or retrofit. Modifications can include switching gains, sensor components, dataloggers, filtering programs, etc. Each change in configuration corresponds to a different set of metadata parameters used for interpreting the data. Blocks of time during which metadata do not change are referred to as EPOCHS. Information about epochs are stored on a webpage, which is downloaded and searched. Once the individual epochs are identified, then a second program loops over each EPOCH and builds a metadata file for each epoch. When a given time series is to be converted into SI units, for example when calculating FC files, the routine `associateEpochNumber.m` is called. This routine takes as input a SITE, SR, CH, combination, together with a year and date. The file `metaData/EPOCHS/SITE_SRCH_EPOCHS.mat` day is loaded and the epochs are searched until the correct one is identified.

<http://www.ncedc.org/ftp/pub/doc/BP.info/BP.responses/RESP.BP.LCCB..LQ1>

RECORDED FIELD	SENSOR TYPE
Epoch #	ALL
Epoch Start Time (4 fields: Year, Julian Day, Hour and Minute)	ALL
Counts Per Volt conversion factor of DataLogger	ALL
COIL ID (Serial number of BF4)	Magnetics
Sensor Gain (dB)	Electrics
Sensor Gain (V)	Electrics
Sensor Length (m)	Electrics
Sensor Azimuth	ALL
Sensor Latitude	ALL
Sensor Longitude	ALL
Sensor Elevation	ALL
Sensor Depth	ALL
Sensor Dip	ALL

Table 1: Metadata Spreadsheet Format

7.2 Spreadsheet (manual) Metadata Handling

The spreadsheets are stored as *.ods* files in *ULFEM/metaData/SPREADSHEETS/*. There is one spreadsheet for each site in the array. Inside the spreadsheet there is a different page for each sensor. For example, the spreadsheet BRIB.ods has seven pages, one for each of { Q2, Q3, Q5, Q6, T1, T2, T3}. For a given sensor the metadata listed in Table 1 is recorded columnwise:

If a new epoch is started for a sensor, a new line is to be created in that sensors file. To port the spreadsheet to matlab, just paste the spreadsheet into the appropriate *.txt* file in the *ULFEM/metaData/SPREADSHEETS/* folder. Then call *updateMetaMat.m* to reinitialize the metadata files. This only needs to be done after a site has been modified, and only the text file corresponding to the modified channels need be updated.

7.3 COILS

The coil calibration files are stored in the *sys* directory. The coils known to have been in use are listed in Table 2.

Right now the metadata handling is split, half between the way that calls BSL, and half using the spreadsheet method. The *getESiteScaleFactor.m* code will need to be modified to use the spreadsheet method, but will work for now.

SITE	CH	Coil ID	EPOCHS
BRIB	T1	unknown	all
BRIB	T2	unknown	all
BRIB	T3	unknown	all
JRSC	T1	unknown	all
JRSC	T2	unknown	all
JRSC	T3	unknown	all
MHDL	T1	unknown	all
MHDL	T2	unknown	all
MHDL	T3	unknown	all
PKD	T1	bf4-9816	1-4
PKD	T1	unknown	5-6
PKD	T2	bf4-9819	1-4
PKD	T2	unknown	5-6
PKD	T3	unknown	all
SAO	T1	unknown	1-10
SAO	T1	bf4-9520	11-14
SAO	T1	bf4-9718	15-16
SAO	T1	bf4-9204	17-18
SAO	T1	unknown	19
SAO	T2	unknown	1-7
SAO	T2	bf4-9519	8-11
SAO	T2	unknown	12
SAO	T3	unknown	all

Table 2: Coils present in ULFEM array

8 KNOWN BUGS

1. Edit Boxes do not instantly update.

When entering a value into an edit box, if the next thing you will be doing is clicking a pushbutton, press enter with the cursor in the edit box first. Otherwise the value you placed in the edit box will not be in the GUI structure, even though it is on the screen.

2. Data Download Hangs This has only happened on AMPERE. Suspect the connection to BSL is too fast for seismo server to keep up. The short term fix is to press Ctrl+C and download will resume. Any corrupt files can be caught in a second pass when “Overwrite” is not selected as a download option. The longer term fix will be to add some appropriate “wait-loops” to the download software so that BSL is given time to finish its calls to database before carrying on. This will involve a “check completion daemon” which will keep track of the last command sent, will hold form sending the next command until confirmation of the first command is returned. Probably solvable wholly within the context of [s w] 0 looping in the matlab “unix” command. One of the places where the hang has happened is “Moving data from BSL to Local Machine”.
3. There are some data files when instrument changes occur during the file and so more than one Epoch is associated. Make sure that there are Flags on the TS files associated with these days.
4. Last EPOCH of PKD (used middle day 277 of 2008) has no metadata under the Update Metadata button. Used day 100 instead and got data by hand. Bug was at BSL since terminal asking for day 100 works but 277 does not.

9 FREQUENCY DOMAIN DATA

First Major Edit: September 6, 2010 A discussion of frequency domain data, motivated by Egbert’s RRRMMT software can be found later in this section. Since writing the initial version of that frequency domain code, it has occurred to me that there is a more general way to proceed. Previous versions of ULFEM, as well as Egbert’s RRRMMT code require a so-called ‘band setup file’ before Fourier transforming is initiated. The band setup file contains lists of harmonics which are returned by FFT of time series which will eventually be stacked together to estimate properties of the signal in particular bands. I point out that such bands *do not need to be defined before FFT*. There are certain benefits to defining these bands up front, as the processing flow requires no more spectral processing parameters to be defined between FFT and presentation of SDMs or Apparent resistivities. The other benefit, is that once the subset of harmonics (Fourier coefficients) which will be used in further processing are identified, the rest of the FCs may be discarded. Disk space

however, at our sampling rates, is not a major issue in the age of \$70 TB hard drives. By keeping all harmonics and placing them in FC files for later processing, and calculating Robust SDMs for each harmonic independently (as is required to hold phase information), we can defer the choice of bands until later. All older notes regarding frequency domain considerations remain relevant, it is just a matter that the bands (or bins as they are sometimes called) do not need to be defined yet. The individual harmonics are determined entirely by the FFT window length, and the sampling rate (only one of which we can control in post processing). By processing all harmonics 'up front' we can use the techniques from the old notes later to choose bands, but in this case, we can rapidly experiment with different band averaging schemes, since reprocessing a new band scheme (assuming it is compatible with the chosen window length) is simply a matter of re-averaging the SDMs, or harmonics.

Thus, frequency domain processing in the current version of ULFEM (post 9/6/2010) is initiated simply by choosing the FFT window length.

The code must, therefore choose the number of decimation levels, or the user must. It seems to me that the number of decimation levels should be user entered to keep things general, and if the user exceeds a reasonable nDL then the code will return a warning or error.

Old version of Notes begins here Much of the ULFEM data analysis is done in frequency domain. This section describes the codes and methods involved in transforming the data to frequency domain. One underlying idea is to use cascading decimation so that the same time-domain windowing scheme can be repeatedly applied at varying levels of resolution. The fundamental considerations involved in setting up such a scheme, which are tied directly to generation and sorting of Fourier coefficients are:

- A Global range of periods or frequencies of interest
- B Number of frequency bins in the range
- C Width of the time-domain window (frequency resolution)
- D Number of decimation levels required to cover A given C
- E Low frequency threshold on number of cycles in a time domain window

Secondary considerations also include:

- (i) Prewhitening scheme to use.
- (ii) Window (tapering) function
- (iii) Sequential time window overlap

The items in the above list are not mutually exclusive, which is to say that by specifying some of them, others become constrained. Choices are partially dictated by theoretical considerations and partly by 'rule of thumb' considerations. The script *HarmonicAndDecimationSchemes.m* was written to help the user choose appropriate values for these parameters. Once selections are made, a bFC.mat file is generated and stored in sys.

Switching between band-averaging and decimation schemes is supported through this script, but it is unlikely that the average user will ever wish to modify the files. To this end, the FC files are not stored according to the bFC file which generated them. Rather, if the user wished to change the band-setup structure, they are encouraged to save the old FC files and bFC files in a backup directory, and reprocess the data, which will generate new FCs. Should the user wish to tag FC files with bFC metadata, this should be fairly simple to do, but it is currently not supported.

Given the exploratory nature of this project, for item A, we would like to look at as broad a range of frequencies as is practical. Since BF-4 coils are extremely poor instruments at periods greater than 10000s, which is less than 1/8 of a day, so there is no reason to do magnetotelluric analysis with the ULFEM arrays at periods longer than this. Considering that a variant of Egberts robust multiple-station processing will be applied to these data at some point, which requires sample statistics about the FCs such as the standard deviation of a block of them, one must make sure that there are enough time windows in a day that meaningful variances can be calculated. A statistical rule of thumb here is that you need more than 10 samples in order to generate meaningful variances using the RMEV method. For item E, some processors claim that a minimum of 10 cycles of a harmonic of interest must be present in a time window before a spectral peak can be 'trusted', where as other processors insist that since theoretical information exists at the Nyquist period, no Fourier coefficients ought to be discarded. We decimate the data by a factor of 2 at each decimation level. This results in considerable overlap between the frequency bands estimated at each decimation level. Because the $i+1$ th decimation level, contains as a high frequency band, the low frequency bands of the i th decimation level, we can choose to omit the lowest frequencies at any decimation level except the highest. This allows us to be conservative in the FCs we keep, insisting on 20 or more cycles per window. For item C, the window length L should be 2^N where N is an integer since the Fast Fourier Transform (FFT) routines are optimally efficient at these widths. The more narrow the window, the finer the time-resolution of the time series of Fourier coefficients. The time-resolution however comes at the cost of frequency resolution. If the ten-cycle criterion is respected, then no window widths of 16 points or smaller make sense to use, since any wavelengths which can oscillate ten times in such a window are above the Nyquist frequency. For example, consider 1Hz data, then ten oscillations in a 16 second window must have a period of 1.6 seconds or shorter, but the Nyquist period is 2s.

The following practical values are chosen as default parameters: Periods of interest: [3-2048]s Number of Frequency bins: 32 (log-spaced) Width of Time Domain window: 256 points Number of Decimation levels Applied: 6 Low Frequency threshold: 6

For secondary factors, Hamming Windows, ARMA(3,4) prewhitening, and 50% sequential time window overlap are selected.

These parameters are all set in the *bandSetup.m* file in the *sys* directory. Figures 3 show the distribution of the Fourier Coefficients amongst the bands.

9.1 Decimation

The fundamental limit of spectral imaging is the so-called Nyquist limit. When sampling at uniform intervals Δt , the highest frequency which can be calculated is $f_{Nyquist} = \frac{1}{2\Delta t}$, or half the sampling frequency. The spacing in frequency domain of the Fourier coefficients is dependent on the spacing in time domain of the sampling, as well as on the length of the time series being transformed. The relationship between these quantities is:

$$df = \frac{1}{N * dt} \quad (1)$$

When decimating a data set sampled at $(\Delta t)_0$, say by a factor of two, then each level of decimation will alter the time interval by a factor of two, so that for the N^{th} level of decimation, we have $(\Delta t)_N = 2^N(\Delta t)_0$, where the zeroth level of decimation is taken as the raw time series. Note that all recorded data are anti-alias filtered in before digitization, and no discussion of that process is given here. We simply assume that all frequency content above $f_{Nyquist}$ has been previously removed from the BSL-stored timeseries. The relationship between Δf , the number of time series observations (N), and the sampling period (Δt) is shown graphically in Figure 9.1, which is made from the program *freqSpacing.m*.

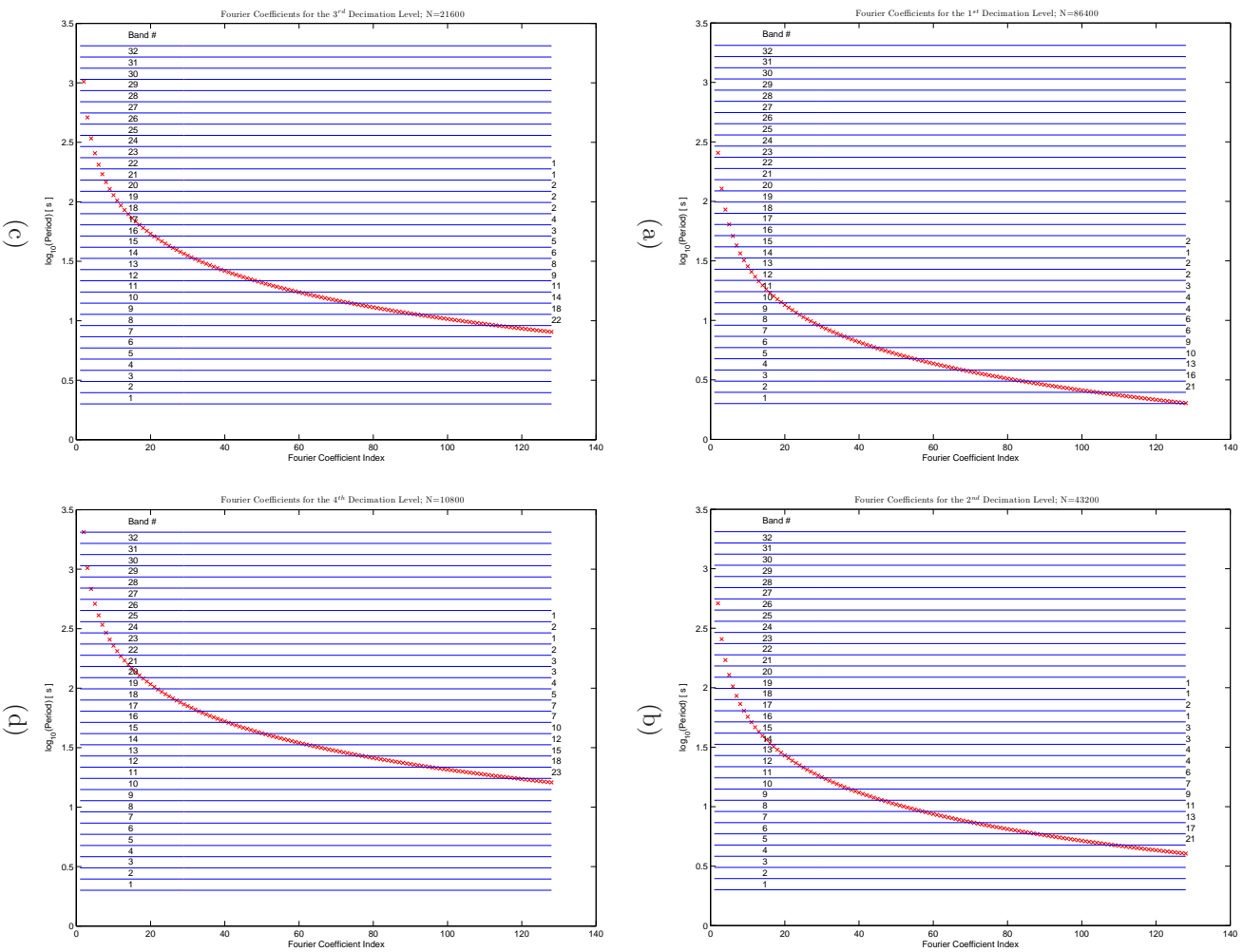


Figure 3: Band Choices (b) ID2

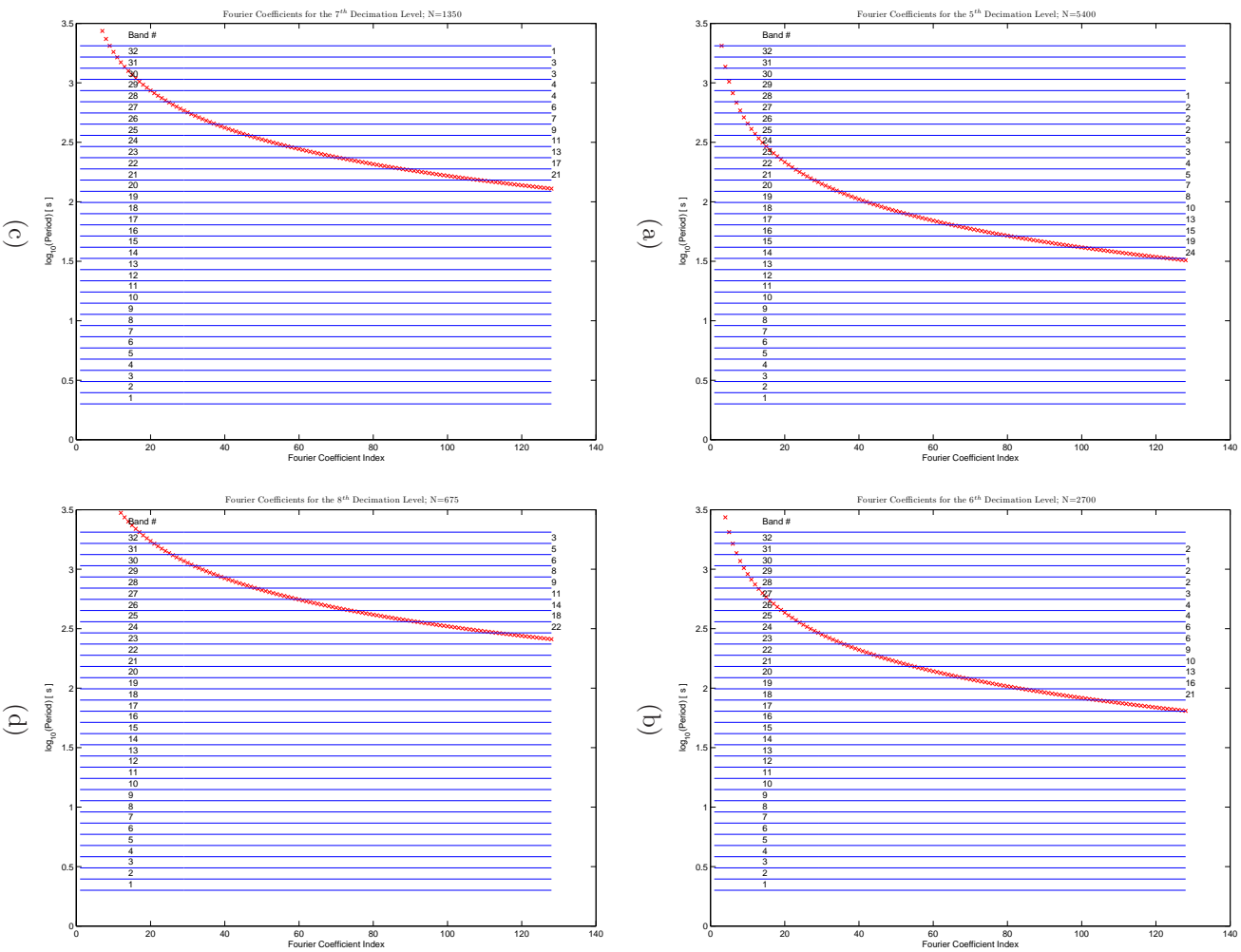
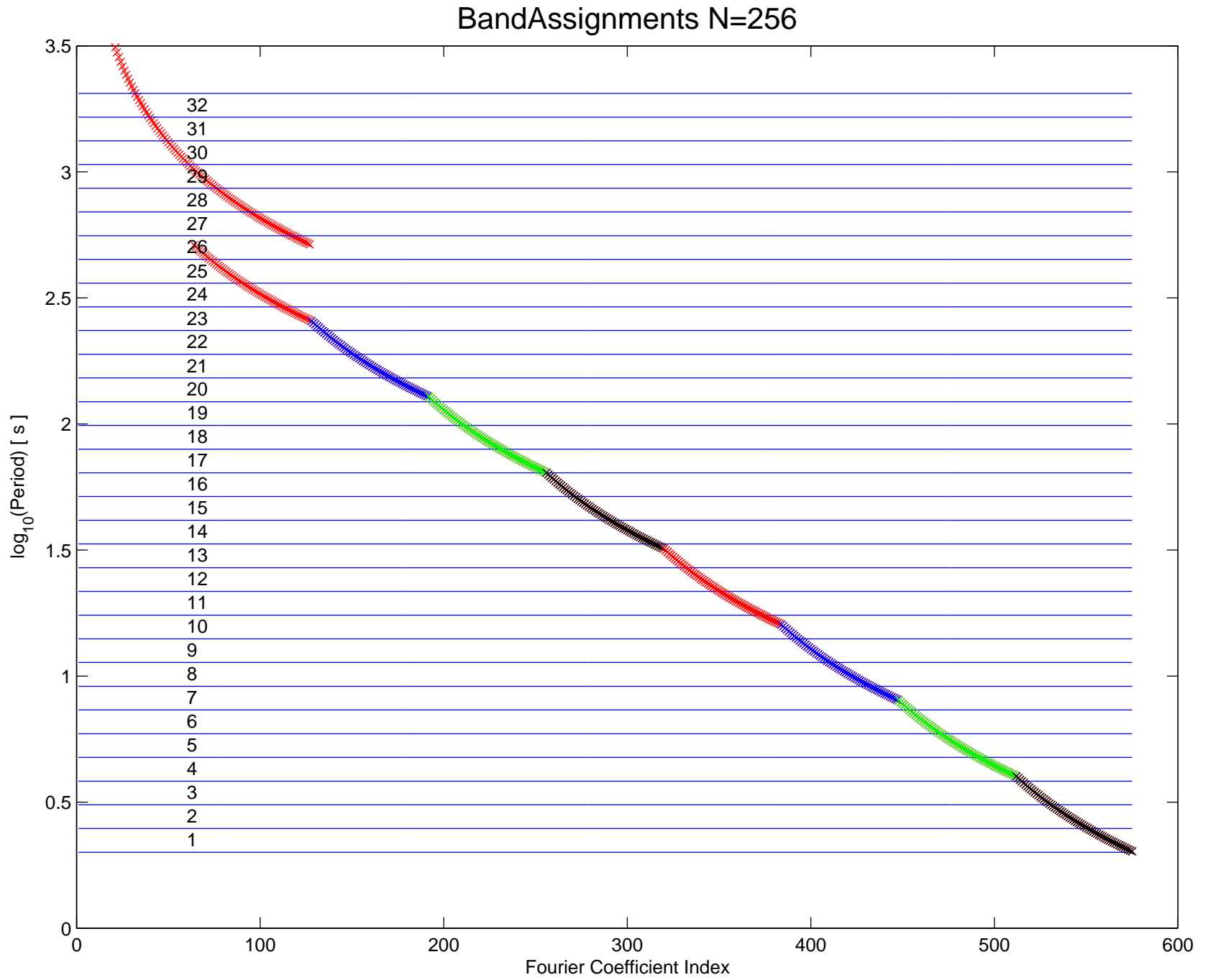
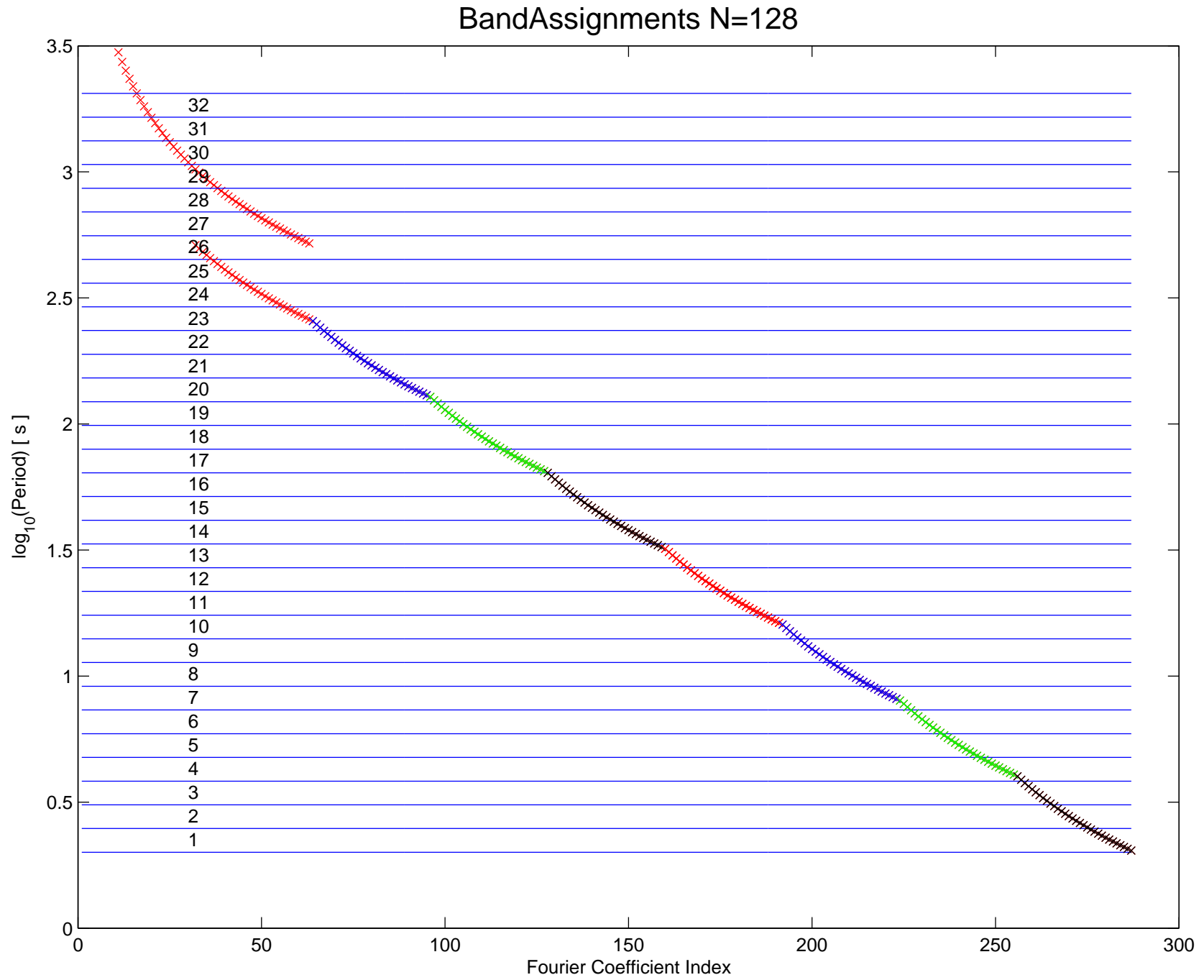
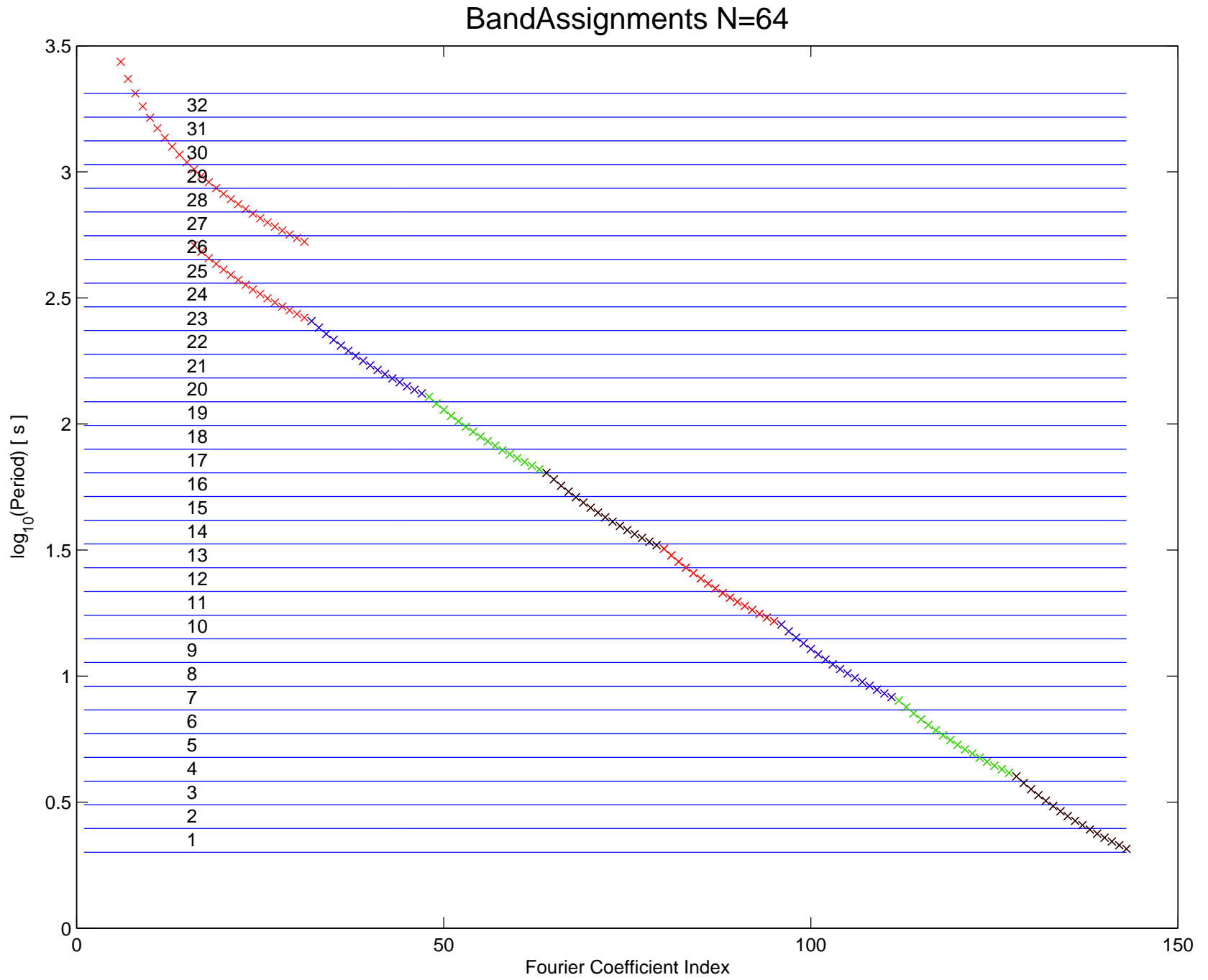
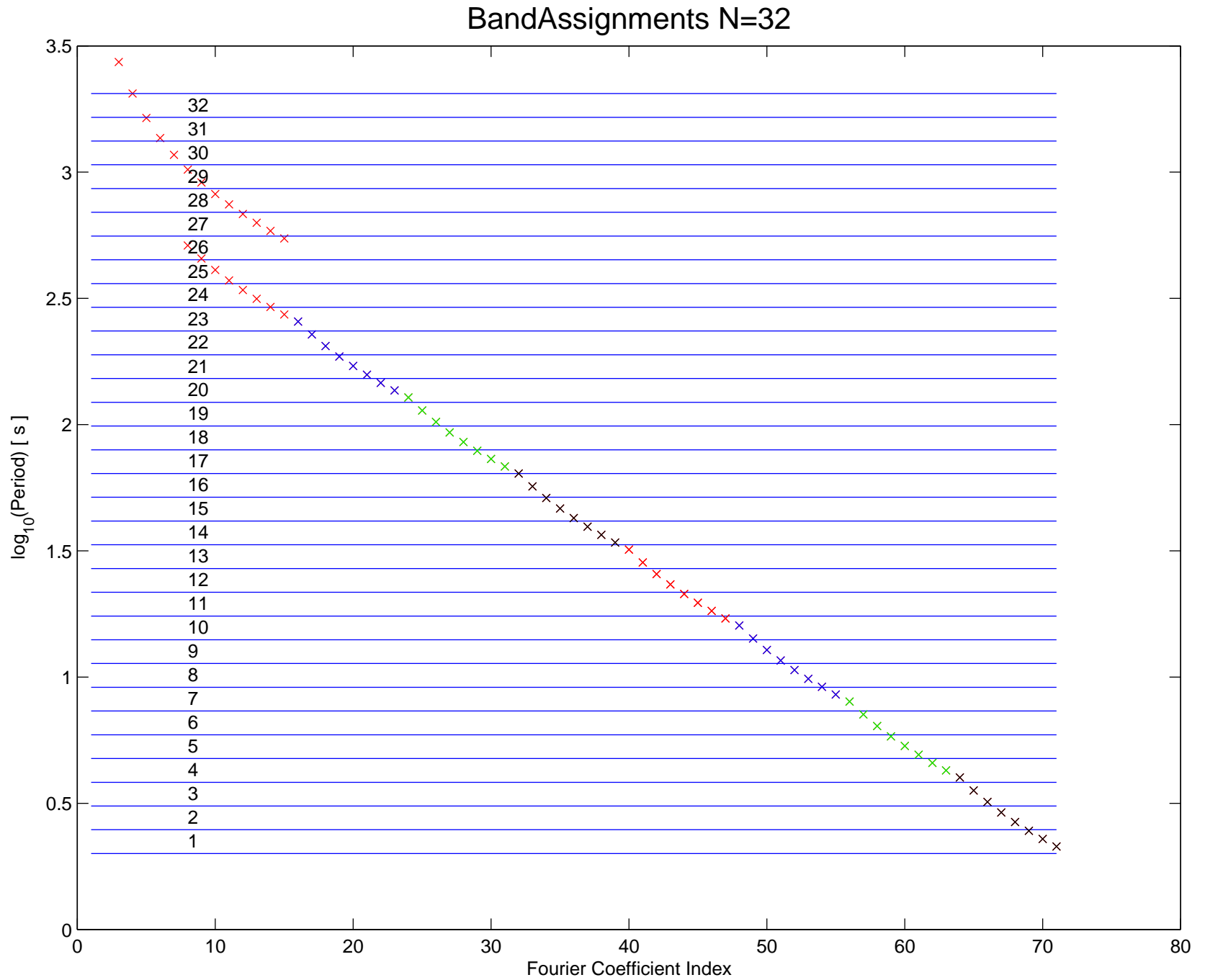


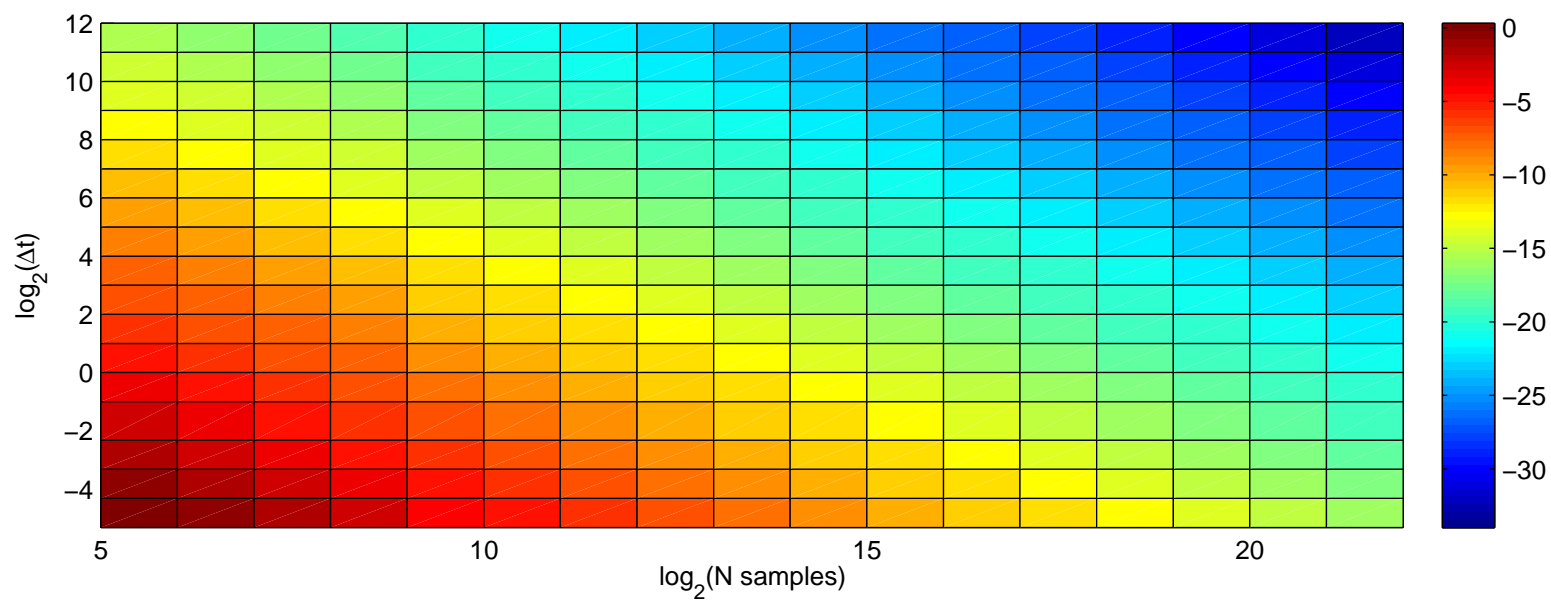
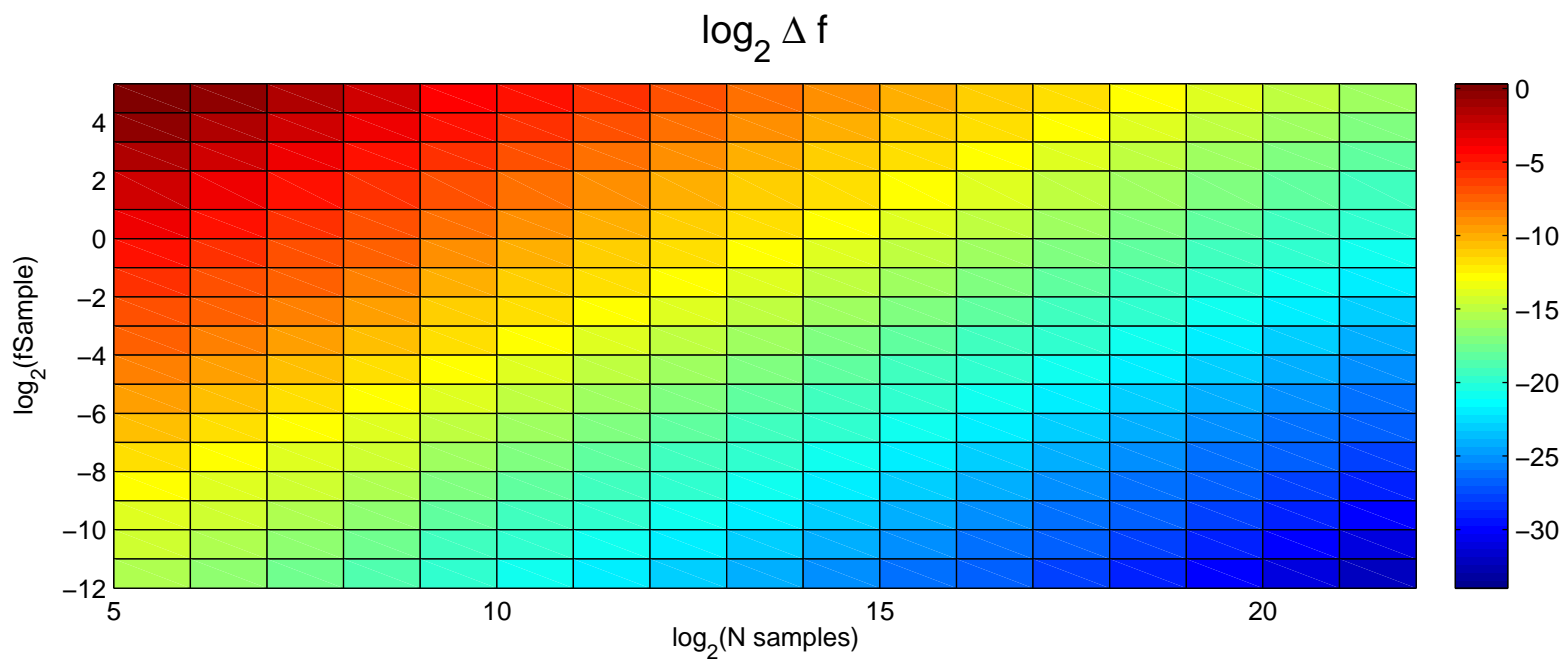
Figure 4: Band Choices (a) ID5











10 ACKNOWLEDGEMENTS

Professor Frank Morrison of UC Berkeley originally conceived the ULF monitoring array concept and installed the early networks at Parkfield and Hollister CA in 1995. Professor Gary Egbert of Oregon State University wrote the seed for the time series plotting software. The development of the ULFEM Matlab package has been made possible by support by researchers Darcy Karakelian-McPhee and Jonathan Glen of the USGS. Darcy and Jonathan, together with Professor Simon Klemperer of Stanford University have established more sites and continue funding the array project. Professor Barbara Romanowicz and the staff at the Berkeley Seismological Laboratory have supported our efforts with data storage, and processing facilities over the past 1.5 decades. Without the contributions of these people our research would not be possible.

11 APPENDIX

11.1 PUBLIC-PRIVATE KEY SETUP

The following text was excerpted from <http://sial.org/howto/openssh/publickey-auth/>.

Secure Shell (SSH) public key authentication can be used by a client to access server

Definition of terms used in this documentation:

- * Client: the system one types directly on, such as a laptop or desktop system.
- * Server: anything connected to from the client. This includes other servers acce

Never allow root-to-root trust between systems. If required by poorly engineered lega

SSH public keys should be periodically rotated, just as X.509 keys are.

First, confirm that OpenSSH is the SSH software installed on the client system. Key g

```
$ ssh -V
```

```
OpenSSH_3.6.1p1+CAN-2003-0693, SSH protocols 1.5/2.0, OpenSSL 0x0090702f
```

If OpenSSH is running on a non-standard port, consult running OpenSSH on a custom por

Key Generation

A RSA key pair must be generated on the client system. The public portion of this key

```
client$ mkdir ~/.ssh
```

```

client$ chmod 700 ~/.ssh
client$ ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

```

Do not use your account password, nor an empty passphrase. The password should be at

The file permissions should be locked down to prevent other users from being able to

```

$ chmod go-w ~/
$ chmod 700 ~/.ssh
$ chmod go-rwx ~/.ssh/*
Key Distribution

```

The public portion of the RSA key pair must be copied to any servers that will be acc

```

# first, upload public key from client to server
client$ scp ~/.ssh/id_rsa.pub server.example.org:

```

```

# next, setup the public key on server
server$ mkdir ~/.ssh
server$ chmod 700 ~/.ssh
server$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
server$ chmod 600 ~/.ssh/authorized_keys
server$ rm ~/id_rsa.pub

```

Be sure to append new public key data to the authorized_keys file, as multiple public

11.2 CODE

11.2.1 ulfemToolbox Remote Directory Contents

cleanAsciiStage.py

```

import sys
import os

asciiStageDir='/scr/01/kappler/dotDeeStage/ascii/'
rmCmd='/bin/rm '+asciiStageDir+'*'
os.system(rmCmd)

```

cleanBinStage.py


```
import sys
import os

binStageDir='/scr/01/kappler/dotDeeStage/bin/'
rmCmd='/bin/rm '+binStageDir+'*'
os.system(rmCmd)
```

cleanMetaStage.py

```
import sys
import os

metaStageDir='/scr/01/kappler/ulfemstage/'
rmCmd='/bin/rm '+metaStageDir+'*'
os.system(rmCmd)
```

11.3 Signals and Instruments

The natural spectrum of megnetotelluric noise has what is known as a $\frac{1}{f^\alpha}$ -type shape. That is to say, the low frequency signals are dramatically larger than the high frequency signals. This has several implications for researchers wishing to record these signals. First, the number of bits recorded on the data loggers must be high. Traditionally (until 2010) the data have been recorded on Quanterra dataloggers which have XX-bits. If one listens to only a short burst of time, the record will not have much drift, and a lower bit rate is acceptable, but as the periods of monitoring increase in length, the signals will drift significantly with progressively lower frequency signals, and the need for a high number of bits becomes important.

References