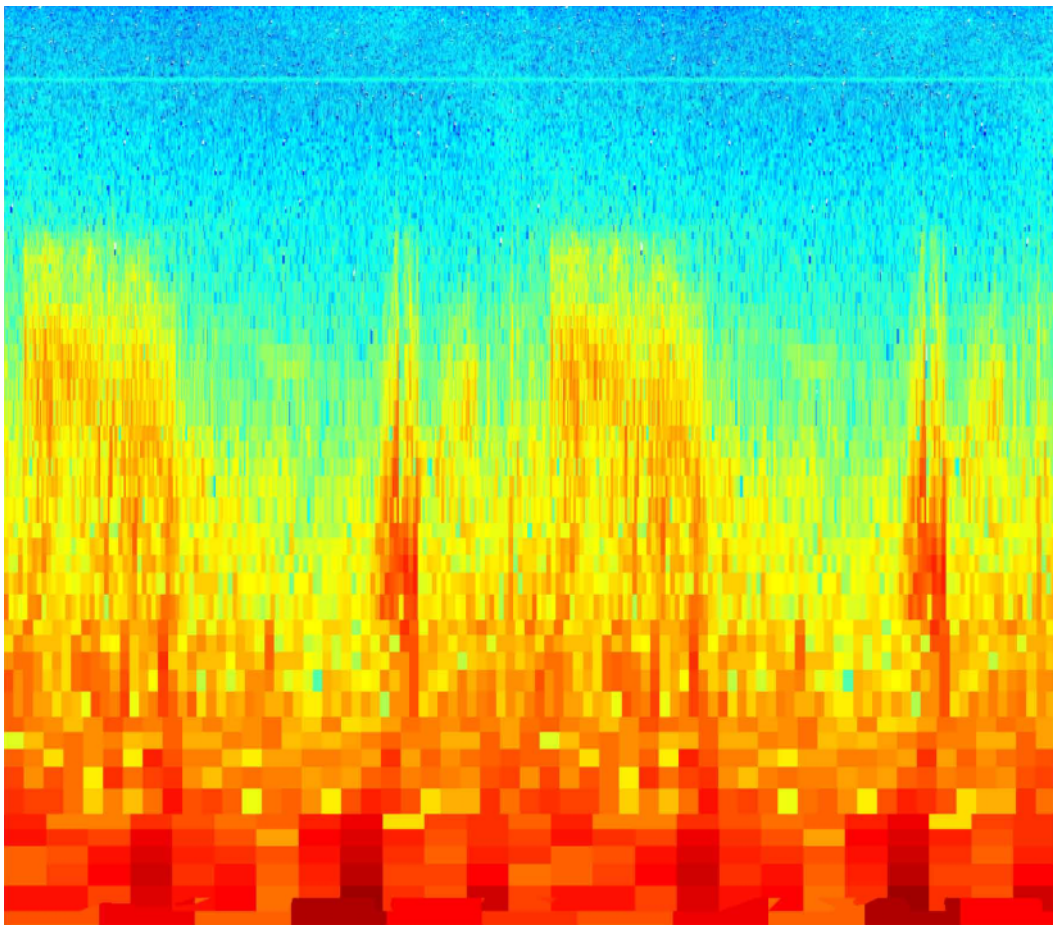
USGS
science for a changing world

ULFEM Time-Series-Analysis Package



Open-File Report 2013–1285

U.S. Department of the Interior
U.S. Geological Survey

ULFEM Time-Series-Analysis Package

By Karl N. Kappler, Darcy K. McPhee, Jonathan M.Glen, and Simon L. Klemperer

Open-File Report 2013–1285

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
Sally Jewell, Secretary

U.S. Geological Survey
Suzette Kimball, Acting Director

U.S. Geological Survey, Reston, Virginia 2013

For more information on the USGS—the Federal source for science about the Earth,
its natural and living resources, natural hazards, and the environment:
World Wide Web: *http://www.usgs.gov*
Telephone: 1-888-ASK-USGS

For an overview of USGS information products, including maps, imagery, and publications, visit
http://www.usgs.gov/pubprod

Suggested citation:
Kappler, K.N., McPhee, D.K., Glen, J.M., and Klemperer, S.L., 2013, ULFEM Time-Series-Analysis Package: U.S. Geological Survey Open-File Report 2013-1285, 326 p.,
http://dx.doi.org/10.3133/ofr20131285

ISSN 2331-1258 (online)

Contents

1 Overview

This manual describes how to use the Ultra-Low-Frequency ElectroMagnetic (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, we provide further description of the routines.

2 Quick-start Guide

This guide lists the commands entered by the user to run ULFEM. Text to be entered on the command line is prefaced by '>'; the Enter symbol means press Enter or Return. This guide assumes that you have access to the *typhoon* machine at the U.S. Geological Survey (USGS) in Menlo Park, California, but the code will run on any Unix/Linux machine or a Windows machine with a Unix emulator such as Cygwin.

2.1 Installing and Calling ULFEM

1. You will need a research account at the Berkeley Seismological Laboratory (BSL) — contact *seismo.berkeley.edu.*

2. Set up Public-Private Key Encryption between BSL and the machine on which you will install the software (see appendix).

2.1.1 Unix/Linux

1. Choose a directory as the home directory, for example: /home/username/ULFEM/. Inside this directory, place the file ULFEM.tar. Then run
   > tar -xvf ULFEM.tar Enter
   A directory named "MATLAB" will be created. Then enter
   > cd MATLAB Enter
   > matlab & Enter

2. Check that paths are set properly:
   Open the script setPaths.m and add a new location; just copy the GUMP settings. At a minimum, you will need to set the variables ULFEMpath, username, binStageDir, ascStageDir, metaStageDir and ulfemToolbox.
   For 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 will 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. In ulfemToolbox, place the python scripts *cleanAsciiStage.py, cleanBinStage.py* and *cleanMetaStage.py* (see appendix for examples of these files.)

3. From the matlab command line, run
   >setPaths Enter

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

5. From the matlab command line, run
   >configureDataDirectories Enter

6. Unpack sitesFolderContents.tar in ULFEM/sys/SITES.

2.1.2 Windows

There are two ways to run the ULFEM code on a Windows machine: (1) by using a UNIX emulator such as ReflectionX and working remotely, or (2) by installing Cygwin and Matlab on your Windows machine and working locally.

2.1.3 typhoon

1. Sit at typhoon and log in as 'ULFEM' or remotely login to typhoon, using ReflectionX or a similar package that emulates UNIX xwindows.

2. Right-click the mouse and select "Open Terminal" from the drop-down menu; a terminal window will appear.

3. Enter
   > cd /gp/ulfem/ULFEM/matlab/ Enter

4. Run
   > matlab & Enter

5. In the MATLAB window, enter
   > ULFEM Enter
   The ULFEM master GUI will appear in the lower left corner of your screen (fig. 1). The matlab console will display a few messages about the paths it is setting. Disregard these for now.

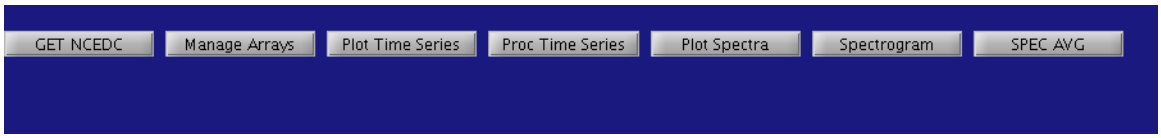


Figure 1. ULFEM program master GUI

2.1.4 Other Machines

1. Open a terminal window (or a cygwin window if using Windows).
2. Change to the ULFEM/MATLAB/ directory.
3. Run
   > matlab & Enter
4. In the MATLAB window, enter
   > ULFEM Enter
   The ULFEM master GUI will appear in the lower left corner of your screen (fig. 1). The matlab console will display a few messages about the paths it is setting. Disregard these for now.

2.2 Downloading Data

2.2.1 1-Hz Data

1. From the ULFEM master GUI, push the "GET NCEDC" button.
   A GETTS window appears, like the one shown in figure 2.
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 that consists of the nine coils at JRSC, BRIB, and MHDL. You can specify the collection of channels you wish to download by selecting one of these preloaded arrays. Alternatively, you can download any channel individually or create your own array as described in sec. 3.3
4. In the YEAR, MM, DD edit boxes, enter the start time. In the MM box use '1-12' for the month corresponding to January through December. In the DD box use '1-31' for the day of the month. For 1-Hz data, the UTHHMM field will default to 00:00

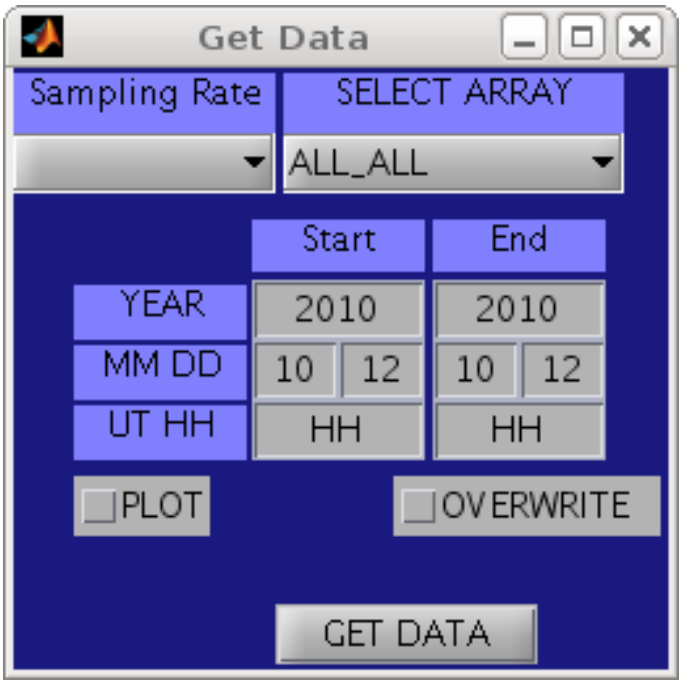


Figure 2. The Get_Data GUI

and 24:00 because whole days are downloaded by default. Select the PLOT checkbox if you wish 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 want only one day of data, leave the end date equal to the start date. 1-Hz data download by default from midnight to midnight GMT.

If a download has been started in error you can abort the process with "Ctrl-C". That is place the cursor in the matlab command window, hold down Ctrl, and then press the C key. The matlab command line should be restored. In the event that the process has runaway or is hanging and matlab is unresponsive, open a terminal window and enter 'ps -e | grep MAT'. A line will appear that 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; then restart matlab.

2.2.2 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 that consists of the nine coils at JRSC, BRIB, and MHDL. You can specify the collection of channels you wish to download by selecting one of these preloaded arrays. Alternatively, you can download any channel individually or create your own array as described in sec. 3.3

4. In the YEAR, MM, DD edit boxes, enter the start time. In the MM box use '1-12' for the month corresponding to January through December. In the DD box use '1-31' for the day of the month. Enter integers corresponding to the starting and ending hours of the desired data window in the UT HH field. Data are downloaded in 2-hour segments. If a specified segment is less than 2 hours, a 2-hour time series will still be delivered, *start* UT HH to *start* UT HH+2. For example if '0','1' are entered as the start and end times respectively, 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 times respectively, or enter '0,24'. 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 interpretted as two data sets. Thus, the code will interpret '4,6' as '4, 5.99'.

   To download the same time window over many days (for example, from 2 to 4 a.m.), just enter '2,4' as the HH start and end times, and then select the range of applicable days. After pushing [Get-Data] , the code will download data in 2-hour segments from the specified start date to your end date. If you want only 1 day of data, leave the end date equal to the start date.

2.3 Plotting Time Series

1. From the ULFEM master GUI, select [Plot Time Series] .

2. In the "Get Plot Data" window that appears (fig. 2), select your sampling rate, array, and day to load.

3. For nSegs, enter the integer number of data segments desired. For 1-Hz data, n segments corresponds to n days of data; for 40-Hz data, n corresponds to the number of 2-hour sections to read in.

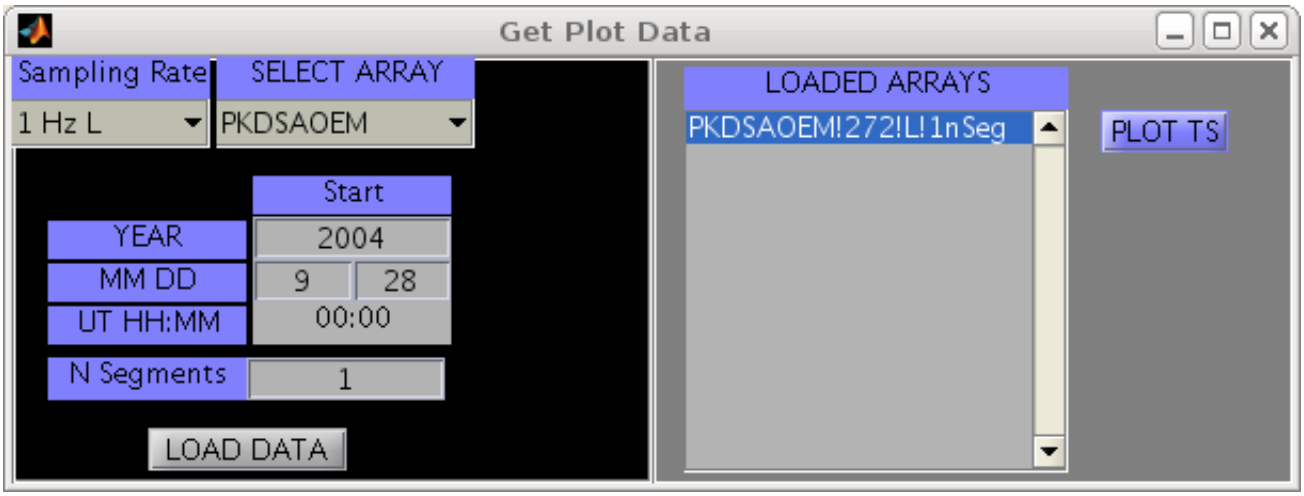


Figure 3. GUI for calling data time-series plots.

4. Push Load Data . Wait a few seconds. An array name will appear in the right-hand side frame.

5. Select the array name by clicking it with the mouse, and push PLOT TS .

2.4 Time-Series Plotter

An example multivariate time series plot is shown in figure 4. The plotter has two main panels and two drop-down control menus. The panel on the left provides edit boxes and controls that allow the user to adjust the length of the displayed time series, to advance the time-series plots by a fixed amount of time with a given overlap, to scale the Y-axis of the displayed plots, and to turn on/off the display of any collection of channels. The default behavior of the plotter is to remove the mean from each plot before drawing, so that each time series will have mean zero in the viewer. This behavior can be adjusted by checking/unchecking the "Centered" option under the Axis-Scaling/Centering drop-down menu.

The time series plotter has several easy-to-use built-in controls, making the following tasks possible:

1. Select the number of points plotted in the current window by changing the value in the edit box in the left panel.

2. Adjust the vertical scale on the plots. To scale all channels at once, click the '+/-' pushbuttons in the left panel to zoom in and out, respectively. To scale an individual

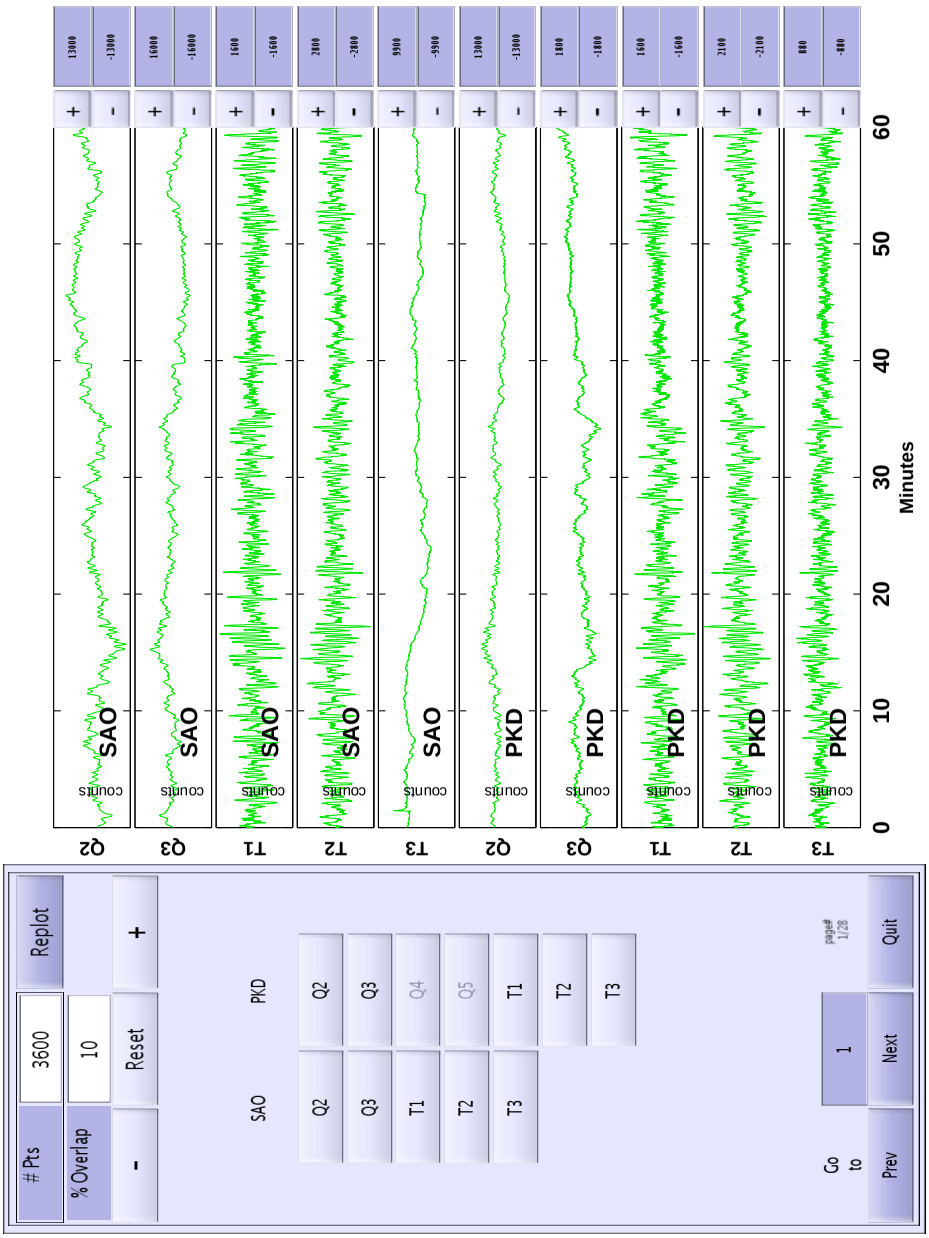


Figure 4. Sample time-series plot from Parkfield-Hollister array.

channel, click the '+/-' pushbuttons immediately to the right of the time series you wish to scale. You can also set the vertical axes limits manually by editing the values in the edit boxes immediately to the right of the individual channel pushbuttons; these boxes indicate the upper and lower bounds of the Y-axes.

3. Add or Remove a channel by pressing the pushbutton that corresponds to the channel you wish to add/remove. The channel pushbuttons are located in the center of the middle frame; they are organized into columns, one column per site, and labeled according to channel. Pushing a particular channel button will cause the time series to be redrawn without that channel, and the label on the button will turn from black to gray. Pushing a gray button will cause the time series to be redrawn, and the text will turn black again.

4. Adjust the X-axes limits. The time-series plotter was designed so that a user can scan the data in adjustable length segments. Once you have selected the number of points displayed, you can advance the time series by pushing the Next button, or regressed by pushing Prev . If the loaded file is larger than the value in "#pts", then the time series is subdivided into "pages"; the number of pages is displayed at the bottom of the left panel. You may select a page by entering a page number in the GoTo edit box near the bottom of the left panel. The overlap between pages can be controlled with the %Ovlp edit box. Setting the value %Ovlp=0 means that the pages partition the time series, whereas setting to non-zero (say, x%) means that the first x% of any page will be the last x% of the previous page. This condition does not apply to page 1.

2.5 Saving Plots

1. Select as active the window you wish to plot.

2. At the MATLAB command line, enter:*pdfPlot('FILENAME');*

3. Enter

4. The plot will be saved to the 'figures' directory in the top level of your ULFEM folder.

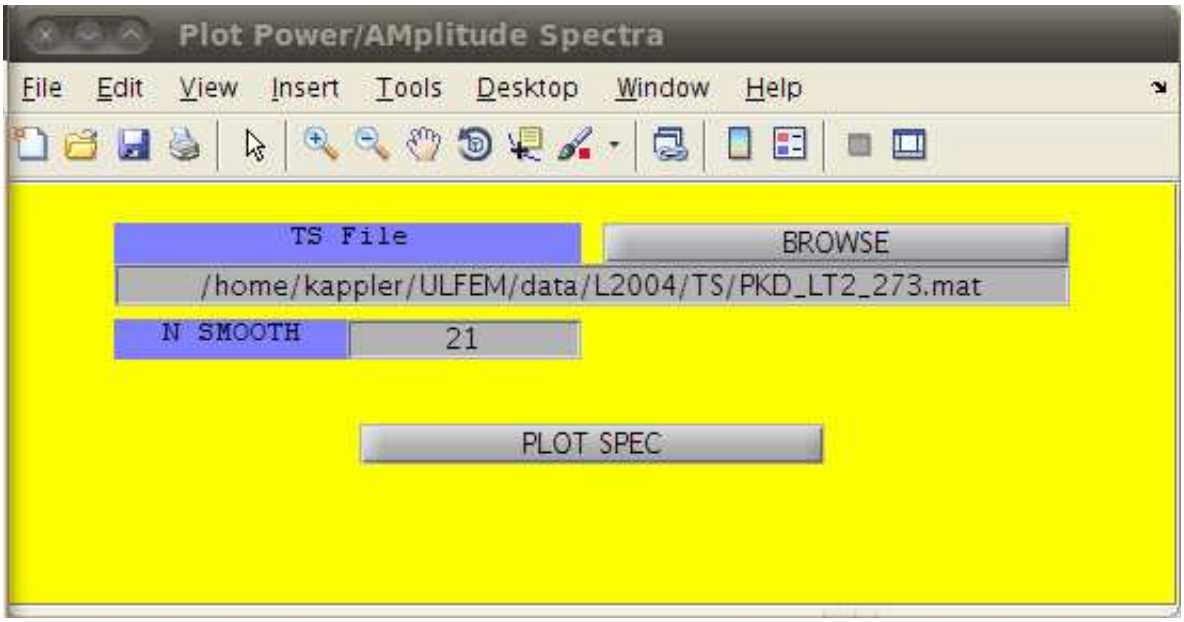


Figure 5. Spectral plotter interface

2.6 Converting Time Series to Fourier-Coefficient Files

To convert raw time series to Fourier coefficient (FC) files, you need to first download all the time series by using the "Get-Data" interface described in section 2.2. To convert TS to FC files, push Proc TS button on the ULFEM master GUI; fill in the metadata specifications for sampling rate, array, and time interval; and leave the other parameters set to default. Push Process Data , and the FC files will be created provided the TS files exist.

2.7 Plotting Simple Spectra

It is often helpful to look at crude plots of time series' spectra without applying significant processing. The PLOT SPECTRA button on the ULFEM GUI offers the user this utility. The user directly specifies a time-series (TS) file to convert to a spectral plot. Raw spectral plots are produced in SI units, unless the user specifies a smoothing filter to be applied, which can be specified as a variable-length median filter. After pushing PLOT SPECTRA the spectral plotter GUI appears as shown in figure 5.

To use the GUI to create a spectral plot:

1. Enter the full path to the TS file you wish to plot. You can push the BROWSE button to look through the data directories for the appropriate file, double-clicking

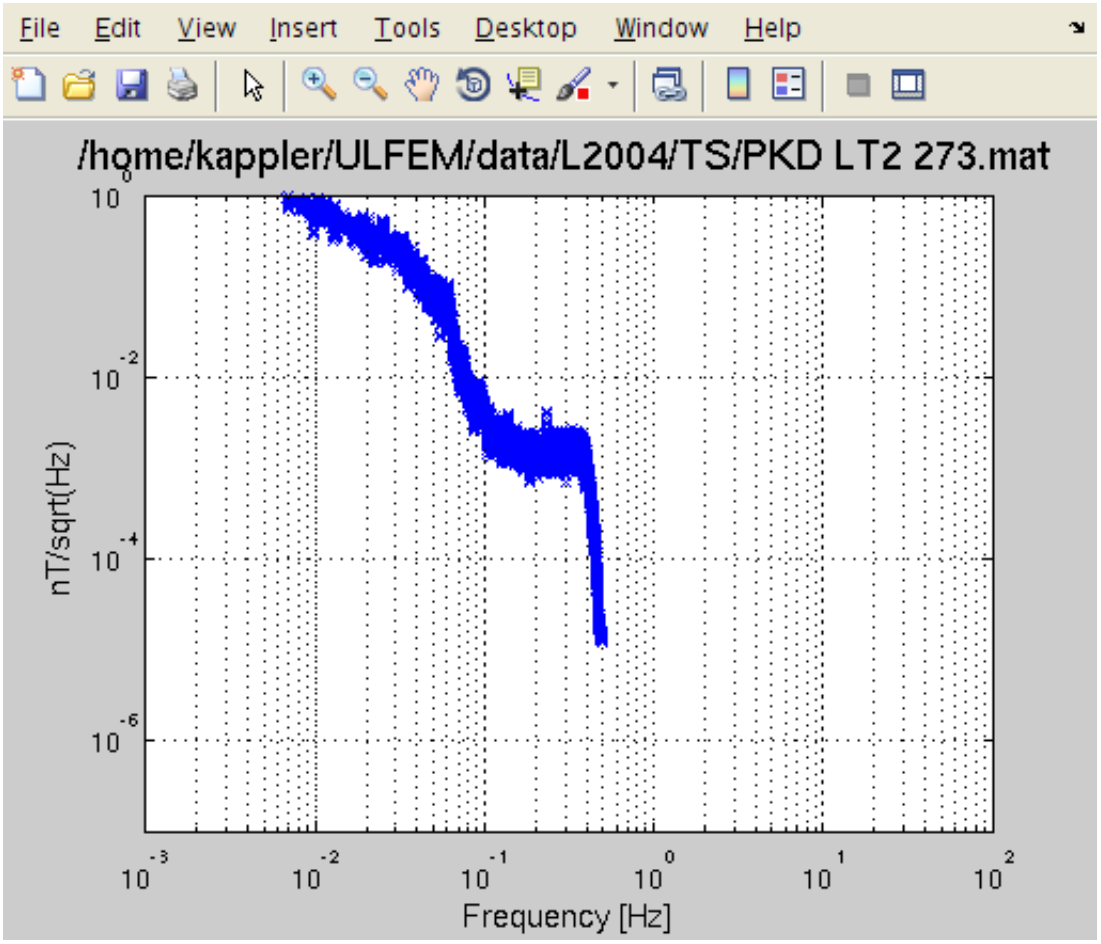


Figure 6. Simple spectral plot created by ULFEM user interface. Note non-physical rolloff due to anti alias filters at high frequency.

the file name once it is found.

2. Select a median smoothing-filter order. Setting to 1 means no smoothing. Use an odd integer, typically not much greater than 100.

3. Push PLOT SPEC . A plot similar to the one in figure 6 should appear.

2.8 Plotting Spectrograms

To plot a spectrogram for a particular instrument over some time interval, first make sure that the FC files are created for that instrument/interval (see sec. 2.6). From the ULFEM master GUI, push the Spectrogram button; a window for metadata entry appears that is similar to the Time Series Plotter (see sec. 2.4). Fill in the fields for sampling rate, array, and time interval as you would for TS plots, and push Load FCs . The FC file loaded for plotting will appear in the listbox on the right-hand side. Selecting the FC file to plot and pushing Plot Spcgm results in a spectrogram plot. An example spectrogram is shown in figure 7.

2.9 Plotting Averaged Spectra

The averaged-spectral plotter is designed with the idea of looking at spectra from many days during the window from around 2-4 a.m. — when the Bay Area Rapid Transit (BART) system is typically offline and data are less noisy. You need to prepare a configuration file (*specAvg.cfg*) before calling the program. The first line of the configuration file specifies station, sampling rate, and sensor. The second line specifies the start of the time interval in HHMM form, and the interval duration in minutes. The remaining lines specify the days from which the data are drawn in YYYY DDD form, where 'DDD' is ordinal day. An example configuration file is shown below:

```
SAO LQ2  #Station  SR SENS
0222 130  #t0 (HHMM)  dt (in minutes)
2004 260 #YYYY DDD
2004 261 #YYYY DDD
2004 262 #YYYY DDD
2004 263 #YYYY DDD
2004 264 #YYYY DDD
2004 265 #YYYY DDD
2004 266 #YYYY DDD
2004 267 #YYYY DDD
2004 268 #YYYY DDD
2004 269 #YYYY DDD
2004 270 #YYYY DDD
2004 271 #YYYY DDD
```

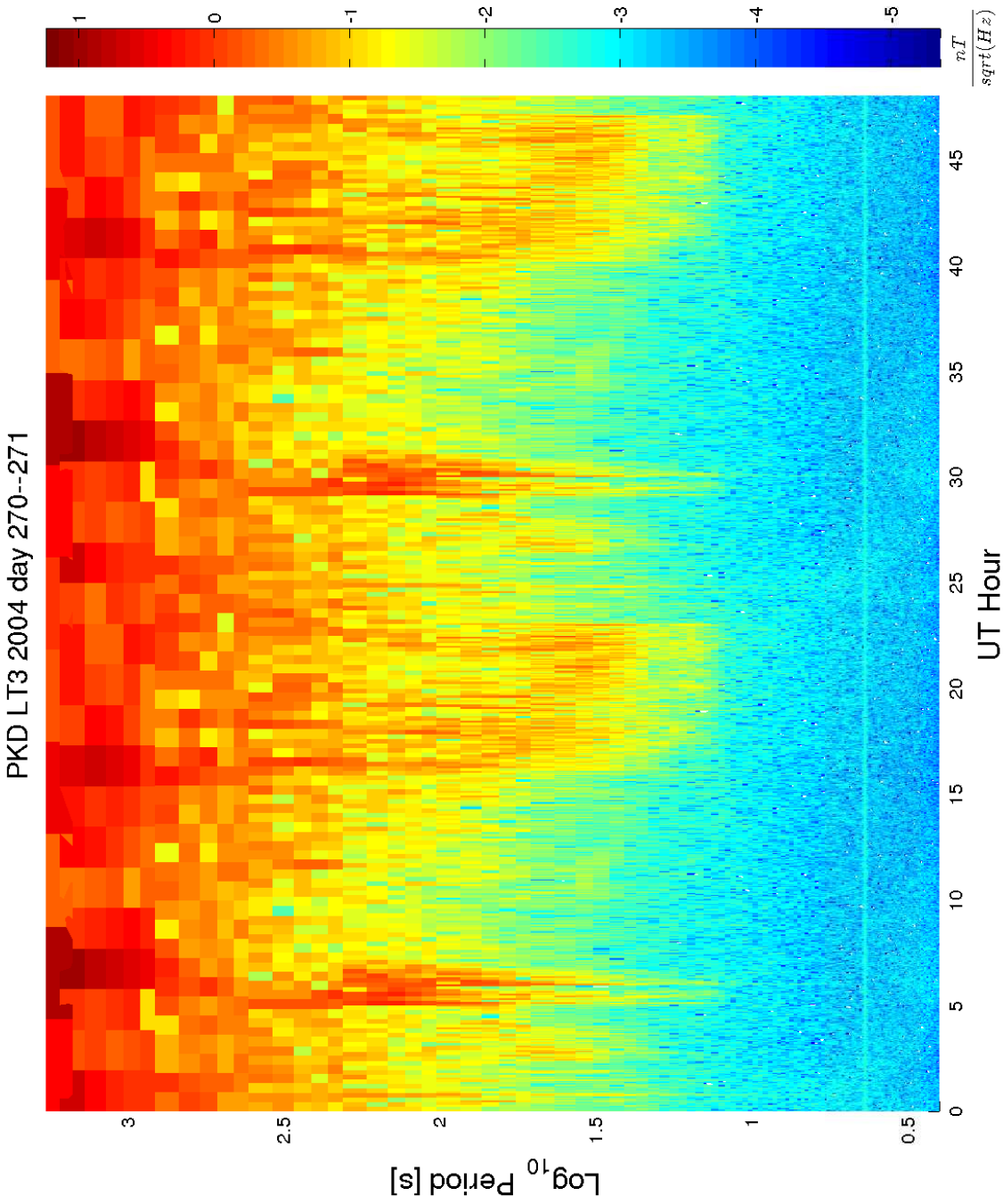


Figure 7. Sample spectrogram created by ULFEM user interface.

```
2004 272 #YYYY DDD
<eof>
```

To use the program, once the configuration file is generated, push the Spec Avg button in the ULFEM GUI; a window will appear like the one shown in figure 8. To compare spectra, select checkboxes for those days to be included in the average, set the smoothing parameter, and click on a day. The time series will appear in the upper box, and the spectra in the lower box. To view spectra without the averaged curve, uncheck the box labeled "SHOW AVG". An example of the GUI with plots is shown in figure 9.

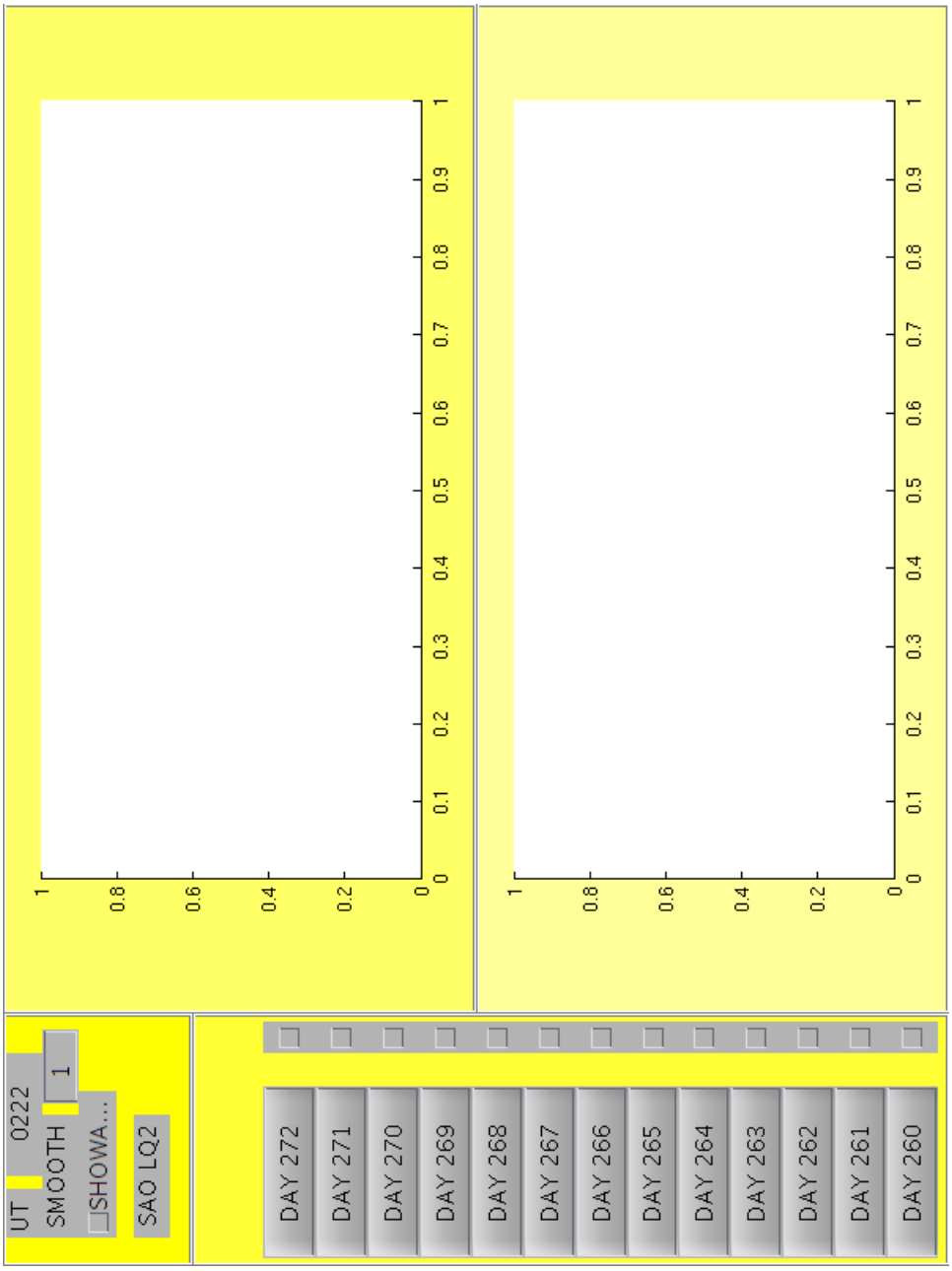


Figure 8. Spectral-average callback window.

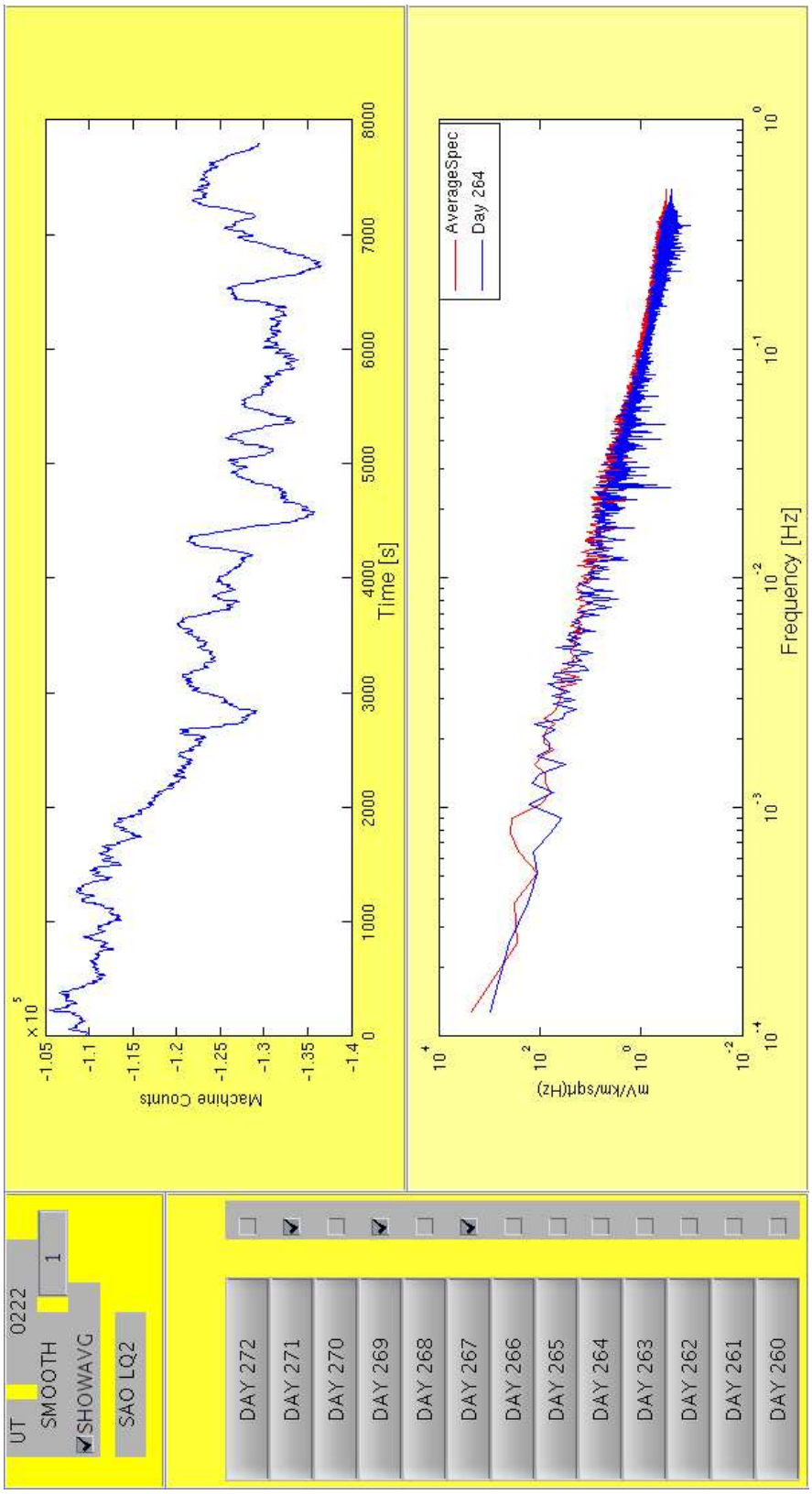


Figure 9. Spectral-average plotting utility.

3 Advanced User's Guide

Overview

This advanced user's guide is a more detailed version of the quick-start-guide. We discuss each part of the ULFEM program in more detail, with advice is on how to customize various parts of the code. When a method is being referred to by its name, I will often append parentheses () to the method name, for example the setRCcbk method is refererred to as getSRcbk().

3.1 Downloading Data

If you wish to allow variation in the starting time of files to download, comment out lines in setSRcbk() where the edit box is set to a text box. The setSRcbk() method is currently in the Get_NCEDC_GUI module in MATLAB/GET_DATA/.

Section 2 explains the basics of data download. Sometimes however, you may request data during a time period when no data are available. To keep track of this situation, a 'dummy' TS file is created when no data are returned. Loading such a TS-file into Matlab yields a single data structure *y*. Where typically *y.data* is an array, in this case *y.data* has the value NaN and *y.flag*="No Binary Data from Que". The process flow for calling data from NCEDC is illustrated in figure 10.

3.2 Updating MetaData

An automated system has been implemented to keep track of the changes in array instrumentation. Metadata were pulled from BSL web pages using the Linux "curl" command and a data parser; however at the time of this writing, the web pages lack several key pieces of information, such as sensor orientation and location coordinates, critical to metadata handling. These fields are slated to be added to the Web pages in the future; meanwhile a system for metadata handling using spreadsheets has been implemented. This system is slaated for migration to SQLite3 databases in a future code version.

3.2.1 Automated Metadata Handling

An interface similar to that for "Get Data" has been developed that allows the user to select combinations of array and sampling rate as input to the "UpdateMetadata.m" sub-

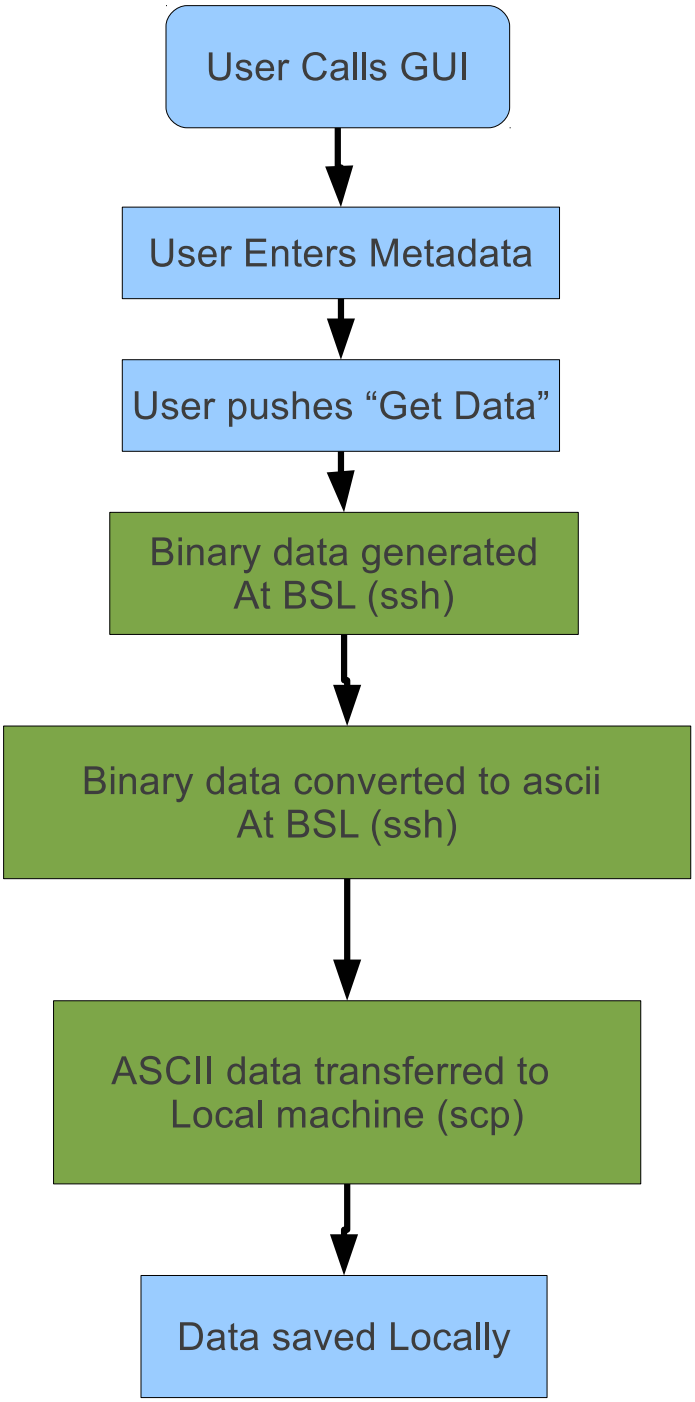


Figure 10. Process flow for calling data from NCEDC to local database. Green boxes refer to processes taking place on the remote BSL server.

routine. The UpdateMetadata program flow consists of two stages. For each channel (ch) and sampling rate (sr), the NCEDC database is queried for a list of EPOCHS when that particular ch-sr combo was in use. For example, some sensors have never been changed since installation, and so there is only one epoch, whereas 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, and so on. 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 Web page, 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 the tuple (SITE, SR, CH), 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.

3.2.2 Spreadsheet (Manual) Metadata Handling

Spreadsheets are stored as *ULFEM/metaData/SPREADSHEETS/* as Libré Office documents (.ods). There is one spreadsheet for each site in the array. Inside the spreadsheet 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 are recorded columnwise.

If a new epoch is started for a sensor, a new line is to be created in that sensor's file. To port spreadsheet to matlab, save it as the the appropriate *.txt* file in the *ULFEM/metaData/SPREADSHEETS/* folder, then call *updateMetaMat.m* to reinitialize the metadata files. This procedure needs to be done only after a site has been modified, and only the text file corresponding to the modified channels needs to be updated — although it is good practice to keep all the spreadsheets current. In hindsight, the export process could have been more cleanly implemented via the Libré Office export-to-csv (comma-separated-value) functionality. Support for this more generic format and SQL databasing utilities exist only in the development version of the ULFEM package however.

Table 1. Metadata Spreadsheet Format

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

3.2.3 Coils

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

3.3 Instrument-Array Manager

A key element of the processing flow is the definition of a list of instruments with which the user will be working. From the collection of all possible instruments, a list of instruments that will be processed together is referred to as an instrument array. The key variable involved in instrument-array selection is *ARRAYS.mat*.

The ULFEM code package can in theory handle any number of sensors, provided that they are all specified with unique names. Because the number of sensors can be large, and often the user may wish to process only a subset of these sensors, the option was created to work with "arrays" of channels that might have have been more appropriately termed "sub-arrays", but for historical reasons are called ARRAYs. An ARRAY is then just a collection of sensors you may wish to process. You can create their own array made up of any collection of sensors by using the Array Manager.

For convenience, several arrays come preloaded; these are the individual sites with an

Table 2. Coils present in ULFEM array

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

array called “ALL_ALL” that contains every channel available. Here are a couple examples of ARRAYS elements:

```
 ARRAYS{1}

ans =

        name: 'ALL_ALL'
    chrArray: {1x38 cell}
         UIC: [38x2 double]
```

or

```
 ARRAYS{7}

ans =

        name: 'BRIONES'
    chrArray: {1x7 cell}
         UIC: [36x2 double]
```

The first element of the ARRAYS.mat data structure is always ALL_ALL, which is hard coded; all other arrays can be added or subtracted at the user’s discretion. The main data structure in the ArrayManager is *handles*. The key fields are all loaded with the command

```
[handles.SITES handles.NETWORKS handles.CHANNELS]=readSiteNetCh
```

The function *readSiteNetCH* loops over the sites in siteNetworks.lst and generates an array for indexing all channels while keeping track of which instruments are part of which array. Note that the indexing depends on the order in which the various sites are listed in siteNetworks.lst, and on the order in which channels are listed in “.sns” files. Ideally, this would have been done by using a dictionary-like data structure such as Matlab’s “container map” so that indexing could have been by instrument name. Because container maps were not available in older versions of Matlab an indexing scheme needed to be created. Of course, all raw TS and FC files are named by individual instrument, and so there will

be no problem with raw-data examination, although array style processing needs to be done with care if the siteNetworks.lst is modified. This file will probably never need to be modified for the Bay Area Network as it exists today. If sites are to be added, or channels added to a site, sites and channels are to be appended to the bottom of the lists. Future code should be written using container maps or, better yet, Python dictionaries to avoid the need for .lst files.
The process flow for using the Array Manager software is shown in figure 11

3.4 Plotting Multivariate Time Series

The multivariate time-series plotter code is based on the analysis software of Gary Egbert at Oregon State University, modified for more channels for the USGS Bay Area array. This code is slated to be replaced by Python in the next version. The process flow for using the time-series plotter is shown in figure 12.

3.4.1 Description of Individual Routines

The main matlab files used for the plotting software are PlotManager.m, loadPlotCB_kk.m, mkshort.m, plotPage.m, plotParams.m, plotTScbk.m, plotTSwin.m, rePltCh.m, and rePltPage.m.

3.4.2 PlotManager.m

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

3.4.3 loadPlotCB_kk.m

The loadPlotCB program is the callback when an array is being loaded. It consists of a case-switch having two cases, *Load* and *Plot*, which are callbacks from the *PlotManager.m* GUI. The *Load* portion of the code performs the operation of reading in the collection

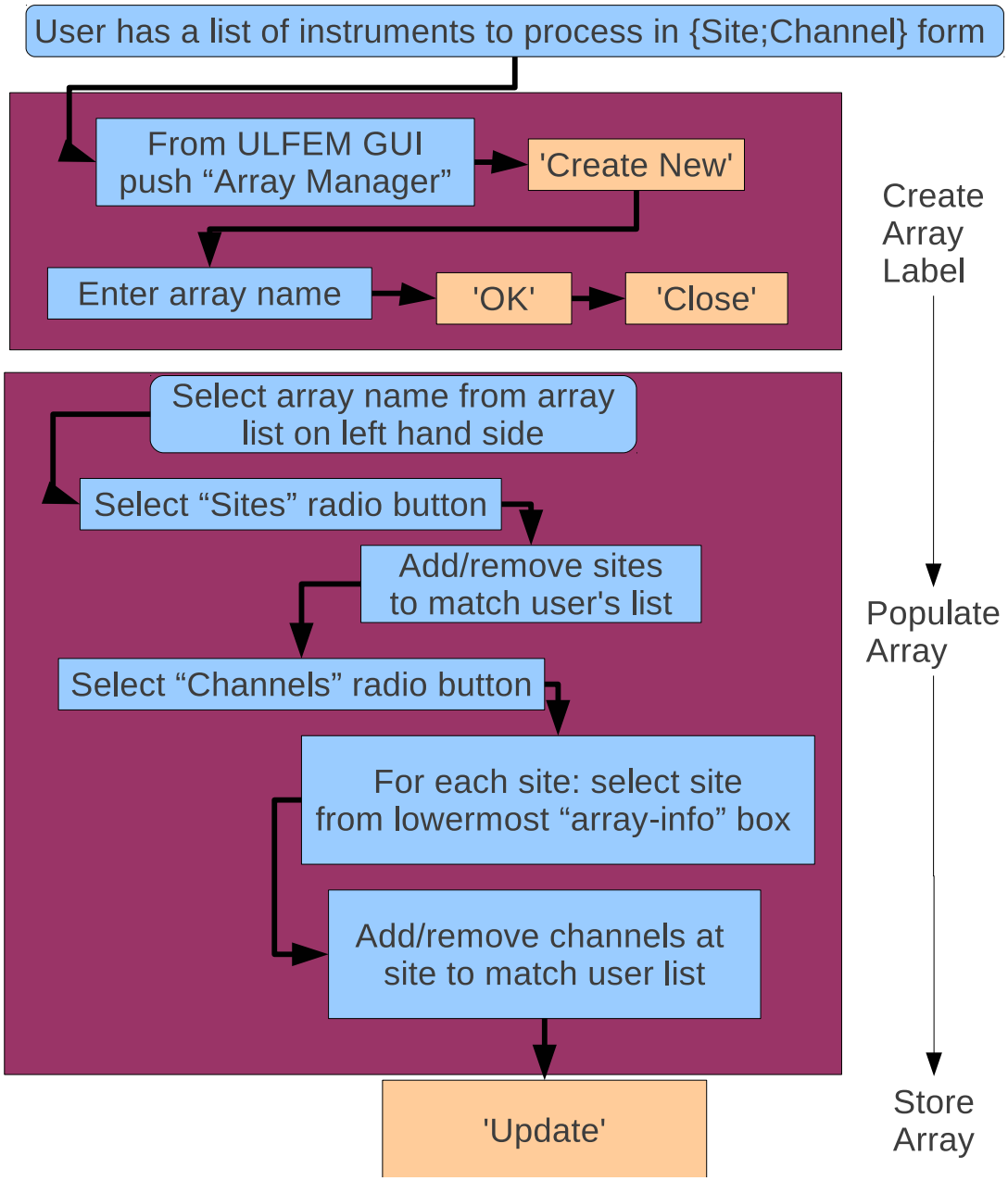


Figure 11. Process flow for adding instrument arrays. Orange boxes refer to pushbuttons on array manager GUI.

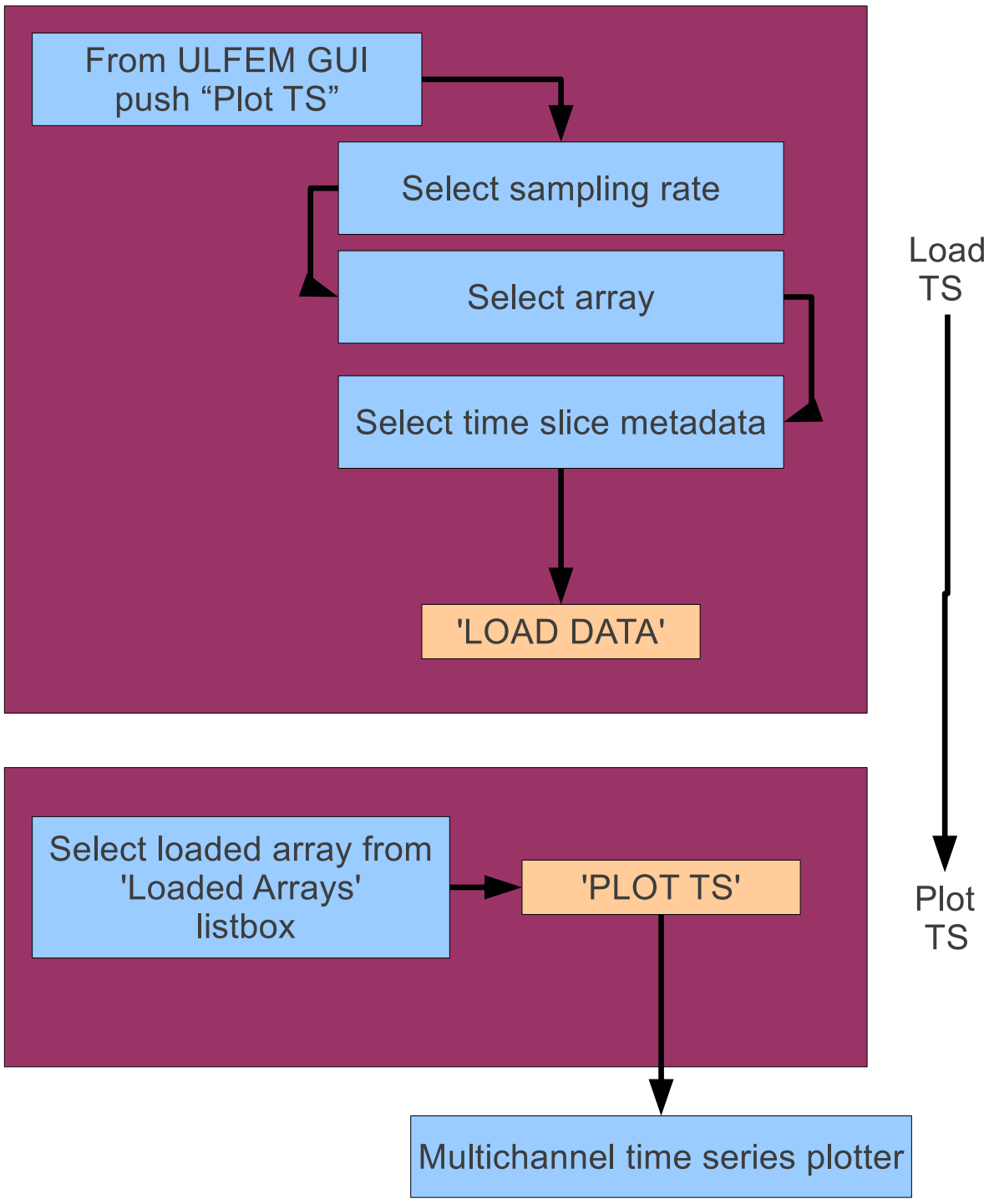


Figure 12. Process flow for calling time-series plots. Orange boxes refer to pushbuttons on Plot TS GUI.

of channels that the user has specified under PlotManager, and then storing the data as a temporary array in the folder SCRPLOTpath. The *Plot* portion of the code reads the temporary array and plots by using a variant of Gary Egbert's time-series-plotting package. Important variables in this routine are TSd1, plotStart, and TSw1; important subroutines that are invoked here are plotTSwin and plotPage.

3.4.4 plotTScbk.m

The plotTScbk.m program has around 630 lines mostly consisting of case-switch loops. The switch is on *action*, which is the Tag of the calling button. 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, and mVkm.

3.4.5 Scaling of E-Field Units

The plotTScbk.m program handles the switch of E-field units from digitizer counts to electric field units (V/m). Under the old system of automated metadata handling, this switch was accomplished with the function *getESiteScaleFactor.m*, which takes as input the four arguments {yStr, dStr, sta, srCH}. Under the spreadsheet method of metadata handling, this was replaced by *getESiteScaleFactor_ssht.m*. Because the response of the digitizer is effectively flat over the frequency range of interest, this conversion is simply done with the product of the three numbers: the counts-to-volts (cpv) conversion factor from the digitizer, the EFSC gain (G), and the electric dipole length (L). The value of cpv is typically about 400,000 counts per volt for Quanterra units, G is either 10, 20, 30 or 40 dB (corresponding to a scale factor of $\sqrt{10}$, 10, $10^{1.5}$, or 100), and L is typically 100 to 200 m. The conversion factor is then given by:

$$E_{V/m} = \frac{E_{counts}}{cpv \times G \times L} \tag{1}$$

The required variables are all stored in each epoch for a particular sensor. The routine associateEpochNumber2.m is required to get the metadata. This routine takes as inputs cfg{iDay}.yyyy, cfg{iDay}.ddd, cfg{iDay}.site, and cfg{iDay}.sens and returns epochMeta{iDay}.

3.4.6 mkShort.m

The mkShort.m function simply cuts the yAxesLims down to "short" numbers so that they will fit inside the edit box on the TS plotting window.

3.4.7 Control of Individual y-Axes Limits in TSplotter

When the initial time series is read into loadPlotCB, the yAxes ranges are determined by an automated process and stored as TSd1.range. The two key functions to understanding the setting of yLims are the callbacks to the **REPLOT** and **RESET** buttons. The **RESET** button reinitializes the yAxes limits, and the **REPLOT** button sets the yAxes lims to the user-specified values in the edit boxes. The callback to both buttons is *plotTScbk*. The relevant actions that are inputted to plotTScbk are replot, set_mCH, set_MCH, and Uniform ylim. Note that plotTSwin.m creates MCH and mCH but doesn't actually assign the values to the edit boxes, which is done in plotPage.m.

3.5 Processing Time Series to Spectral Data

The spectral processing of time series is done by a cascading-decimation short-time-Fourier-transform (STFT) scheme typical to magnetotelluric (MT) processing. The STFT is popular for MT because the processing results are spectral ratios of electric-to-magnetic time series. Thus, spikes in time series cause erroneous results, which are handled by breaking the time series into short "ensembles" to perform statistics on many spectral ratio estimates. For monitoring purposes, we also want to break long time series into ensembles. The high-frequency data available in each ensemble are redundant in longer ensembles, containing little new information besides more time averaging. To avoid such redundant information, time series are decimated. The TS files are converted to FC files in a collection of bands and decimation levels.

3.6 Spectral Plotter

The process flow for using the spectral plotter is shown in figure 13. The overarching routine to plot spectra is SpecPlotter.m which generates the GUI and buttons for controlling which spectra to plot. The key program that actually converts time series to spectra and plots is plotTS2Spec.m, which is a standalone spectral calculator that directly converts time series to spectral representation, with accounting for SI units and consideration built in for the use of various window functions and smoothers.

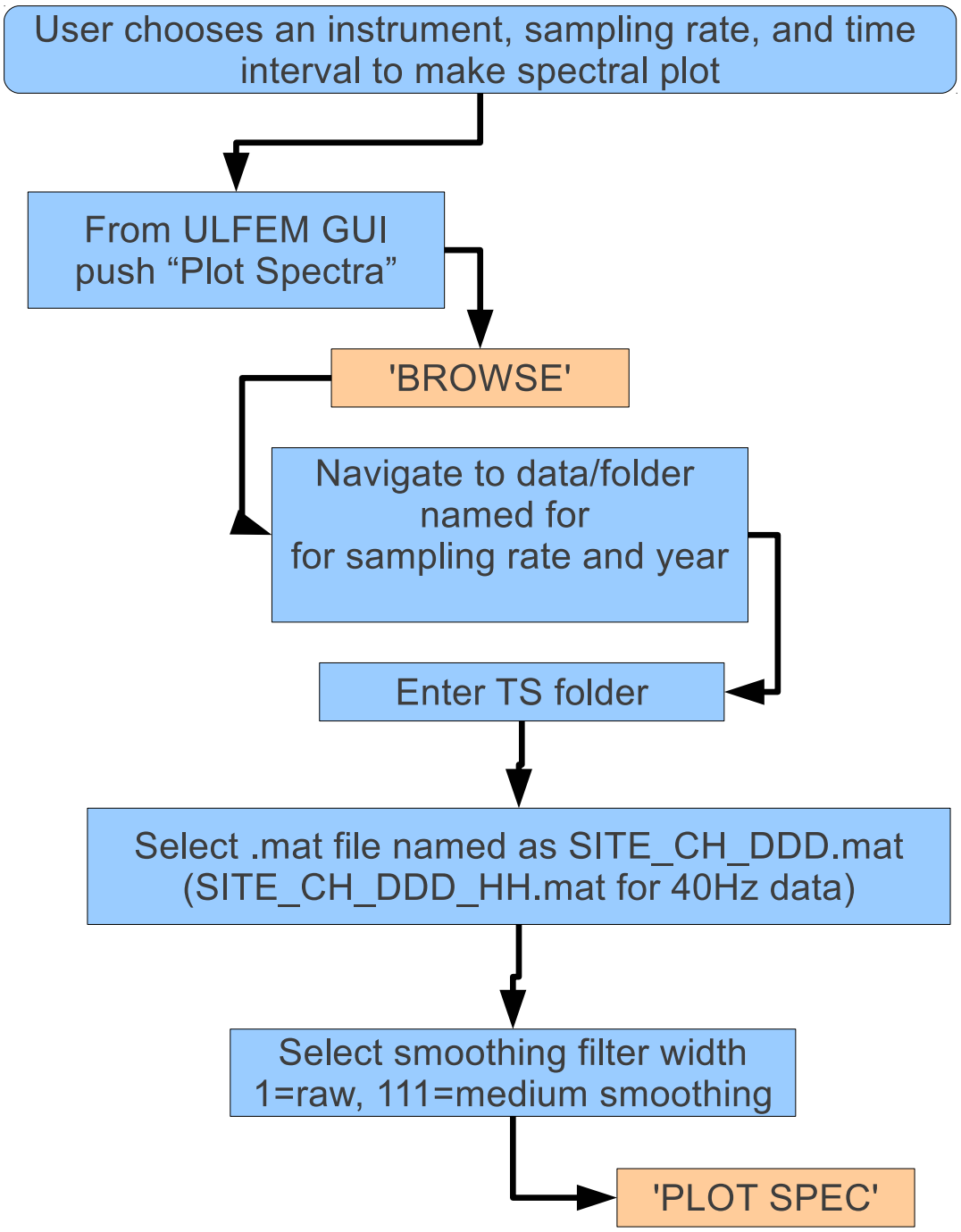


Figure 13. Process flow for using the spectral plotter. Orange boxes refer to pushbuttons on GUI.

3.7 Spectrogram Plotter

The process flow for using the spectrogram plotter is shown in figure 14. The GUI for choosing channels and dates of spectrograms is almost identical to that used to plot raw time series, but the callbacks from the GUI are totally different. Once the user selects an array and a collection of days for spectral plots, and pushes "LOAD FCs", the code flow is to load the FC Files (rather than TS files) for each channel and day and store them as daily files, then concatenate the daily array files into a multiple segment array file, which is stored separately as a temporary file. Pushing the PLOT button calls a loop over all array channels, generating the spectrogram. The structure of FC files is described in section 4 below.

3.8 Spectral-Average Plotter

The spectral-average configuration file is stored in *specAvg.cfg*. The philosophy for the spectral-averaging interface is to read in all the time series and store them in the handles.daysOfData structure upon initialization. Then power spectra for each time series are calculated and stored in the same data structure. When a particular combination of days is selected as the spectral average, the power spectra are then averaged, and the square root is taken just before plotting. Thus the plotted quantity is amplitude spectra. The process flow for comparing plots of a given day versus an averaged spectrum is shown in figure 15.

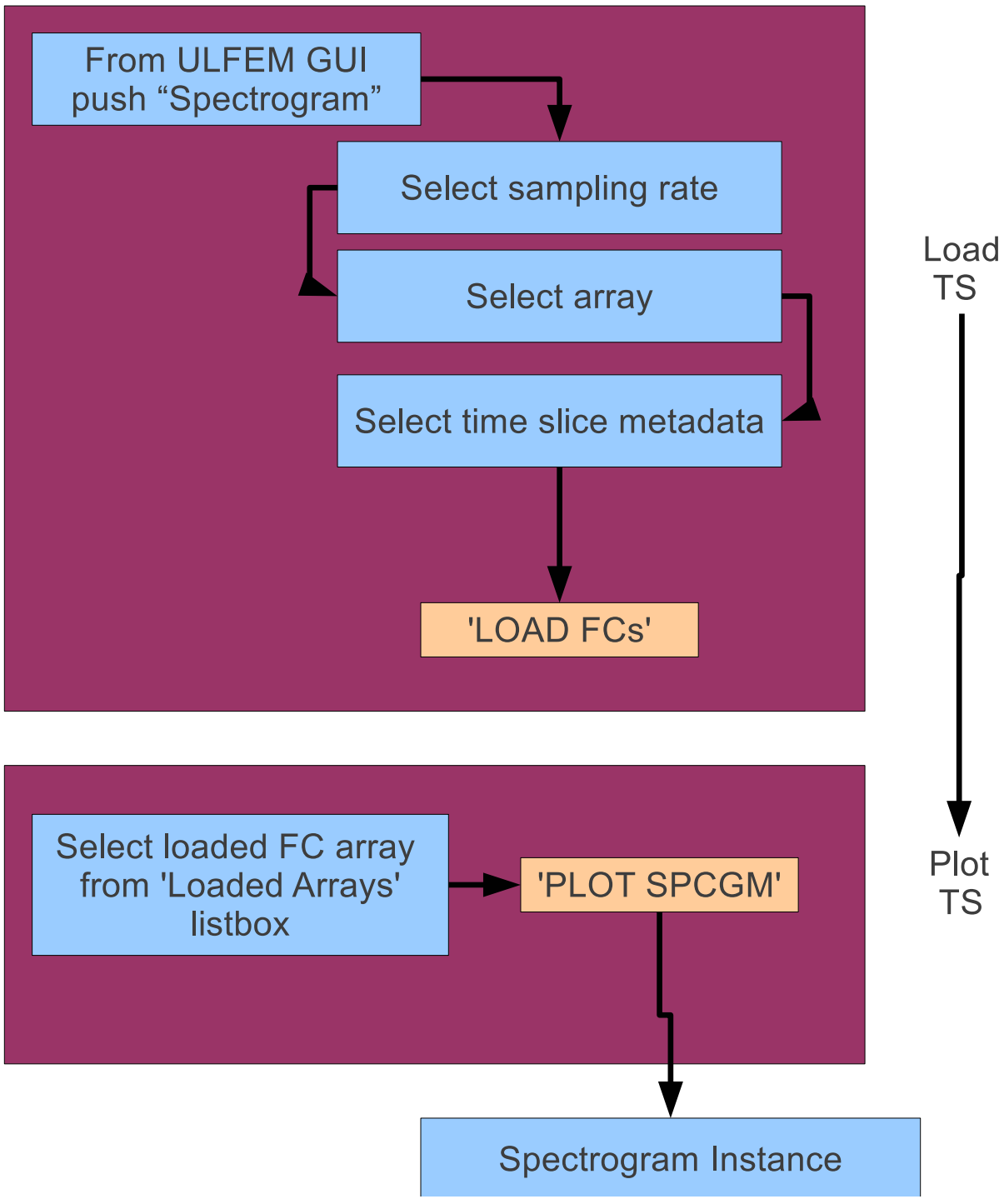


Figure 14. Process flow for using spectrogram plotter. Orange boxes refer to pushbuttons on GUI.

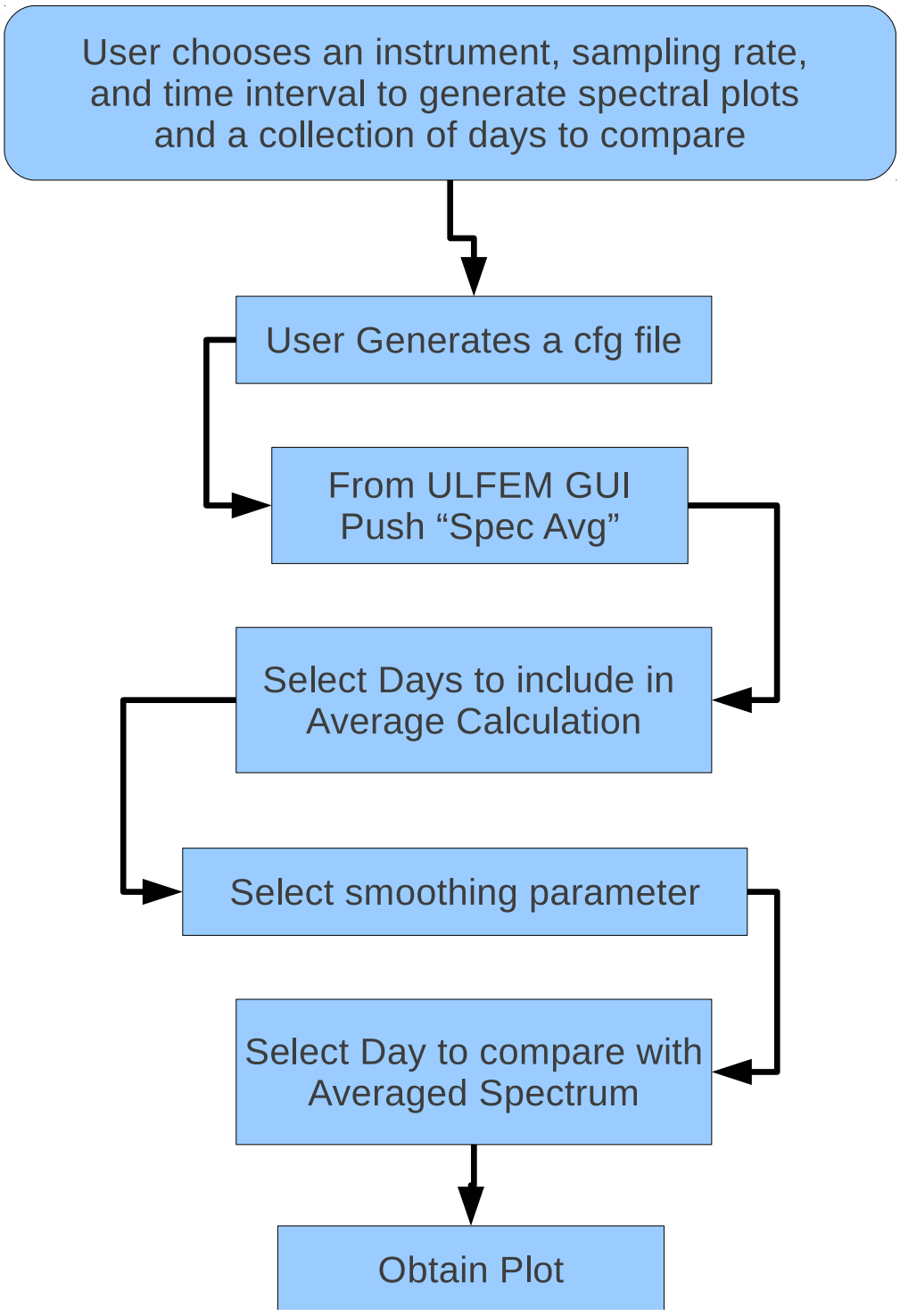


Figure 15. Process flow for using spectral-average plotter.

4 Fundamental Data Structures

This section documents the data structures that are used repeatedly in the data handling, as well as underlying configuration files on which the ULFEM code depends.

4.1 Site and Lists Channel

Adding or removing sites from the ULFEM package is controlled within the directory *sys/SITES*, which contains three types of files: siteNetworks.lst, ARRAYS.mat, and ".sns", the most fundamental of which is siteNetworks.lst. Here is the file text:

```
SAO   BK
BRIB  BK
PKD   BK
MHDL  BK
JRSC  BK
CCRB  BP
LCCB  BP
SCYB  BP
```

which is simply a list of sites in the left-hand column, with the network to which each site belongs in the right-hand column. If a sensor that is not at one of these sites is to be added to ULFEM, the user needs to add a line with the site and network. For each site-network pair there is an ".sns" (sensor) file, which is a list of the sensors at the site — not necessarily a list of all tha sensors physically present, but just those ULFEM will read and work with. An example .sns file from Parkfield is:

```
Q2
Q3
Q4
Q5
T1
T2
T3
HE
HN
HZ
```

In the above example, we have four electric dipoles (Q2-Q5), three BF4 coils (T1-T3), and a three-component seismometer (HE, HN, HZ).

4.2 FC Files

The FC files are created from the TS files stored in the TS folders. The FC files are generated by using a cascading-decimation scheme. A typical FC file X, from a 1-Hz data segment, has the following structure:

```
>> X

X =

  Columns 1 through 4

    {1x14 cell}    {1x15 cell}    {1x15 cell}    {1x15 cell}

  Columns 5 through 8

    {1x15 cell}    {1x14 cell}    {1x12 cell}    {1x9 cell}

  Column 9

    {1x6 cell}
```

where X itself is a cell array, with one entry for each decimation level. The first decimation level has, for example, 14 bands:

```
 X{1}

ans =

  Columns 1 through 3

    [21x449 double]    [16x449 double]    [13x449 double]
```

```
Columns 4 through 6

  [10x449 double]    [9x449 double]    [6x449 double]

Columns 7 through 9

  [6x449 double]    [4x449 double]    [4x449 double]

Columns 10 through 12

  [3x449 double]    [2x449 double]    [2x449 double]

Columns 13 through 14

  [1x449 double]    [2x449 double]
```

where each band has the shape $nFC \times nT$, where nFC is the number of Fourier coefficients in a band, and nT is the number of time windows in a day (or segment). The cells in X{i} are complex-valued matlab arrays.

4.2.1 Frequency-Domain Data in ULFEM

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. Cascading decimation is used so that the same time-domain windowing scheme can be repeatedly applied at varying levels of resolution. The fundamental items involved in setting up such a scheme tied directly to generation and sorting of Fourier coefficients are (1) global range of periods or frequencies of interest, (2) number of frequency bins in the range, (3) width of the time-domain window (frequency resolution), (4) number of decimation levels required to cover 1, given 3, and (5) low frequency threshold on number of cycles in a time domain window.

Secondary items also include (6) prewhitening scheme to use, (7) apodization window (tapering) function, and (8) sequential time window overlap.

These items are not mutually exclusive — specifying some of them constrains others. Choices are dictated partly 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 items. Once selections are made, a bFC.mat file is generated and stored in the *sys* directory.

Switching band-averaging and cascading-decimation schemes is supported through this script, but it is unlikely that the average user will wish to modify the files. Should the user wish to tag FC files with bFC metadata, this should be fairly simple to implement, but it is currently not supported. Instead, if the user wishes to change the band-setup structure, the old FC files and bFC files can be moved to a backup directory and nef FC files generated by reprocessing the data. A more prototype FC file is a class which includes specifications of the listed processing parameters exists in the development codes. However, the code associated with this documentation store the bFC and FC files separately.

For item 1 abve, we would like to look at as broad a range of frequencies as is practical. Since BF-4 coils respond poorly at periods greater than 10,000s, ($\approx$1/8 of a day), the default bFC file does not extend analysis to longer periods than this. Considering that we will wish to calculate, at some point, sample statistics about the Fourier coefficients (for example, the standard deviation of the amplitude of some harmonic over particular time intervals), we must ensure enough time windows in a segment (day) that meaningful variances can be calculated. For item 5, we require a certain minimum number of cycles of a harmonic to be present in a time window before a spectral peak can be "trusted". The default decimation of the data is by a factor of 2 at each level, resulting 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 procedure allows us to be conservative in the Fourier coefficients we keep on the low-frequency end, insisting on 20 or more cycles per window. Note that the harmonics on the high end are affected by antialias filters and should be interpreted with this in mind, especially in the raw (zeroth decimation level) data. For item 3, choosing L should be 2^N (where N is an integer) is optimal because the fast Fourier transform routines are most efficient at these widths. The narrower the window, the finer the time-resolution of the time series of Fourier coefficients at the cost of frequency resolution. In general a lower bound on L is twice the minimum number of oscillations required in a window.

The following practical values are chosen as default parameters for 1-Hz data: period of interest, 3-2,048 s; number of frequency bins, 32 (log-spaced); width of time domain window, 256 points; number of decimation levels, 9; and low frequency threshold, 6.

For items 6-8, Hamming windows, ARMA(3,4) prewhitening, and 50-percent sequential time-window overlap are selected. Note that when there are sharp spectral peaks, prewhitening can cause severe errors in spectral estimates, and so prewhitening in these cases is not recommended.

These parameters are all set in the *bandSetup.m* file in the *sys* directory. The distribution of Fourier coefficients amongst the bands is plotted in figures 16 and 17.

4.2.2 Decimation

The fundamental limit of spectral imaging is the so-called Nyquist limit. When sampling at uniform intervals Δt, the highest calculable frequency is $\frac{1}{2\Delta t}$, or half the sampling frequency. The spacing in frequency domain of the Fourier coefficients depends on the spacing in time domain of the sampling, as well as on the length of the time series being transformed. The relation between these quantities is given by:

$$\Delta f = \frac{1}{N * \Delta t} \tag{2}$$

When decimating a dataset sampled at $(\Delta t)_0$, say, by a factor of 2, then each level of decimation will alter the time interval by a factor of 2, so that for the Nth 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 antialias filtered before digitization, and so no discussion of that process is given here. We assume that all frequency content above the Nyquist frequency has been previously removed from the BSL-stored time series. The relation between Δf, the number of time-series observations (N), and the sampling period (Δt) is plotted in figure 18, which is generated from the program *freqSpacing.m*.

4.2.3 FCRA Data Structure

The FCRA data structure which is generated by *readFCRA.m* (usage: *FCRA = readFCRA(RA, segmentSpecifier)*), contains all the FC files from a particular collection of instruments. It has the same structure as an FC file but with one added dimension in the arrays of Fourier coefficients to cover the various instruments.

A single segment, for example X=FCRA{1}, using 1-Hz data and the standard *bFC*-file has the following structure when there are three instruments in the collection for site MHDL.

```
 X{1}

ans =

  Columns 1 through 3

    [3x21x449 double]    [3x16x449 double]    [3x13x449 double]

  Columns 4 through 6

    [3x10x449 double]    [3x9x449 double]    [3x6x449 double]

  Columns 7 through 9

    [3x6x449 double]     [3x4x449 double]    [3x4x449 double]

  Columns 10 through 12

    [3x3x449 double]     [3x2x449 double]    [3x2x449 double]

  Columns 13 through 14

    [3x1x449 double]    [3x2x449 double]
```

4.3 bFC File Structure

A bFC file is a cell array with one cell for each decimation level. In the following example there are 9 levels of decimation.

```
 bFC

bFC =

  Columns 1 through 4
```

```
    {1x14 cell}    {1x15 cell}    {1x15 cell}    {1x15 cell}

  Columns 5 through 8

    {1x15 cell}    {1x14 cell}    {1x12 cell}    {1x9 cell}

  Column 9

    {1x6 cell}
```

Each decimation level is broken into "bands". For example, decimation level 2 has 15 bands:

```
 bFC{2}

ans =

  Columns 1 through 4

    [1x1 struct]    [1x1 struct]    [1x1 struct]    [1x1 struct]

  Columns 5 through 8

    [1x1 struct]    [1x1 struct]    [1x1 struct]    [1x1 struct]

  Columns 9 through 12

    [1x1 struct]    [1x1 struct]    [1x1 struct]    [1x1 struct]

  Columns 13 through 15

    [1x1 struct]    [1x1 struct]    [1x1 struct]
```

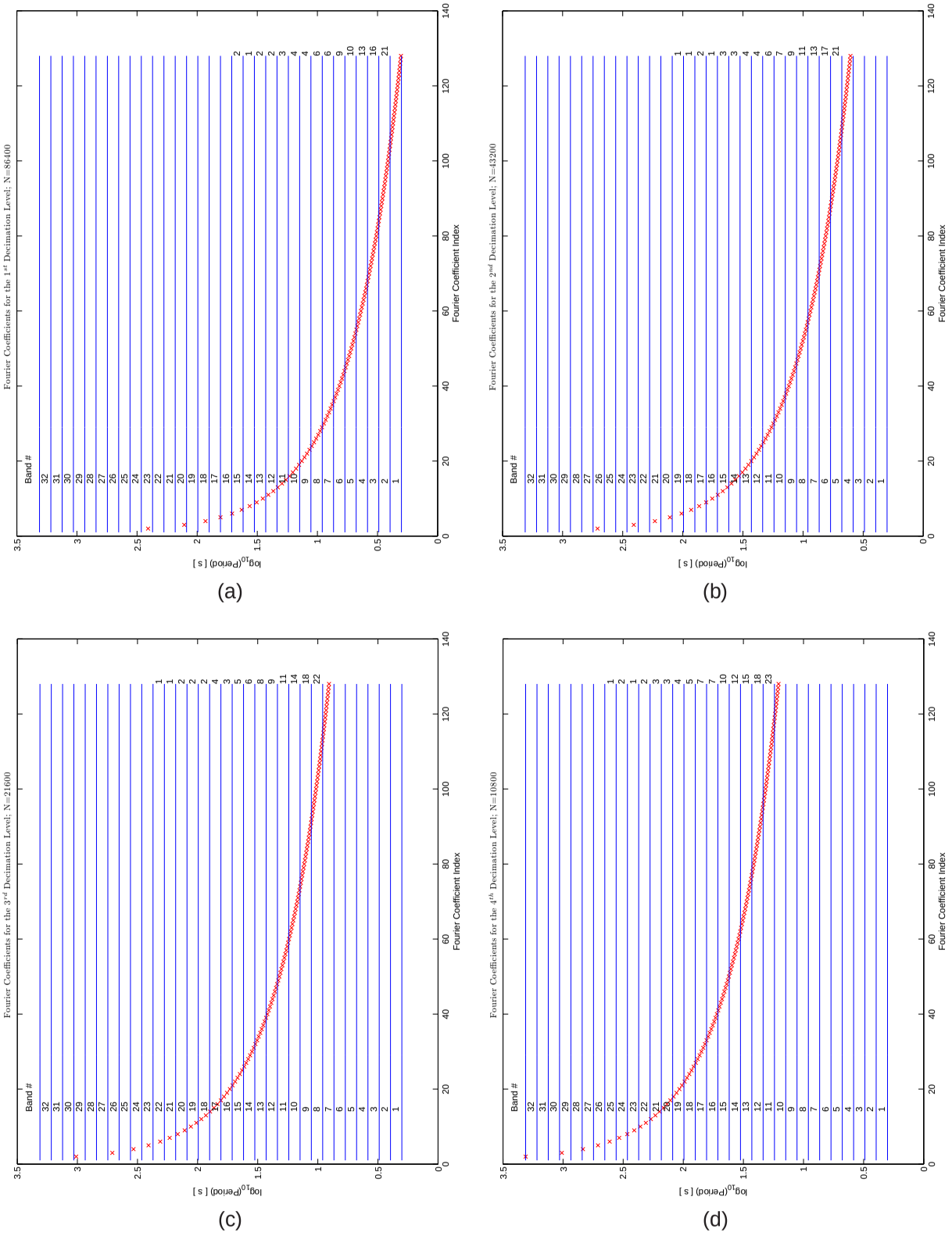


Figure 16. Band choices for decimation levels 1 through 4 with windows width of 256 (a-d correspond to decimation levels 1 through 4 respectively).

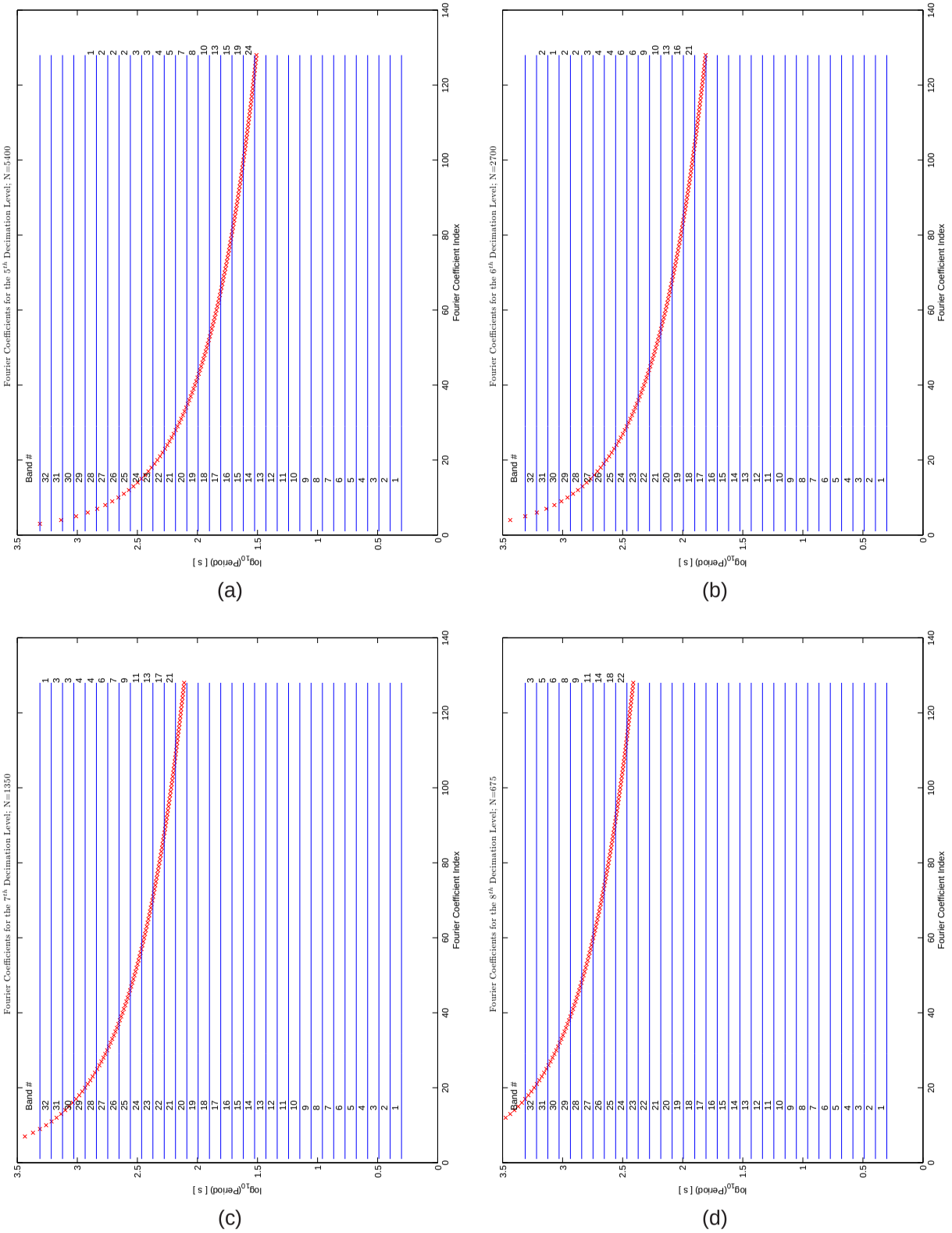


Figure 17. Band choices for decimation levels 5 through 8 with windows width of 256 (a-d correspond to decimation levels 5 through 8 respectively).

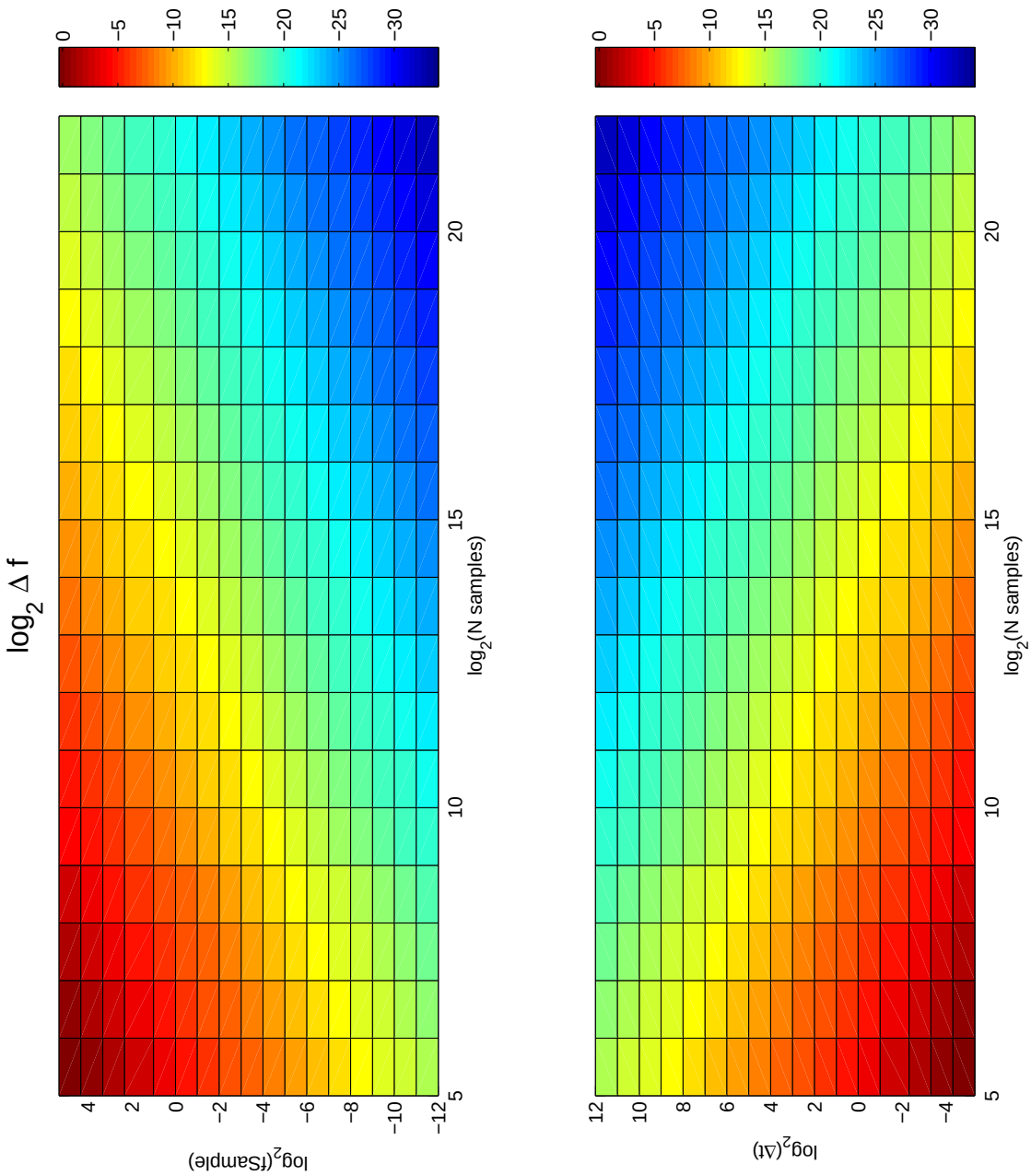


Figure 18. Δf shown on the color (Z) axis as a function of sampling rate (upper plate) or dt (lower plate) and number of points in time series on X axis.

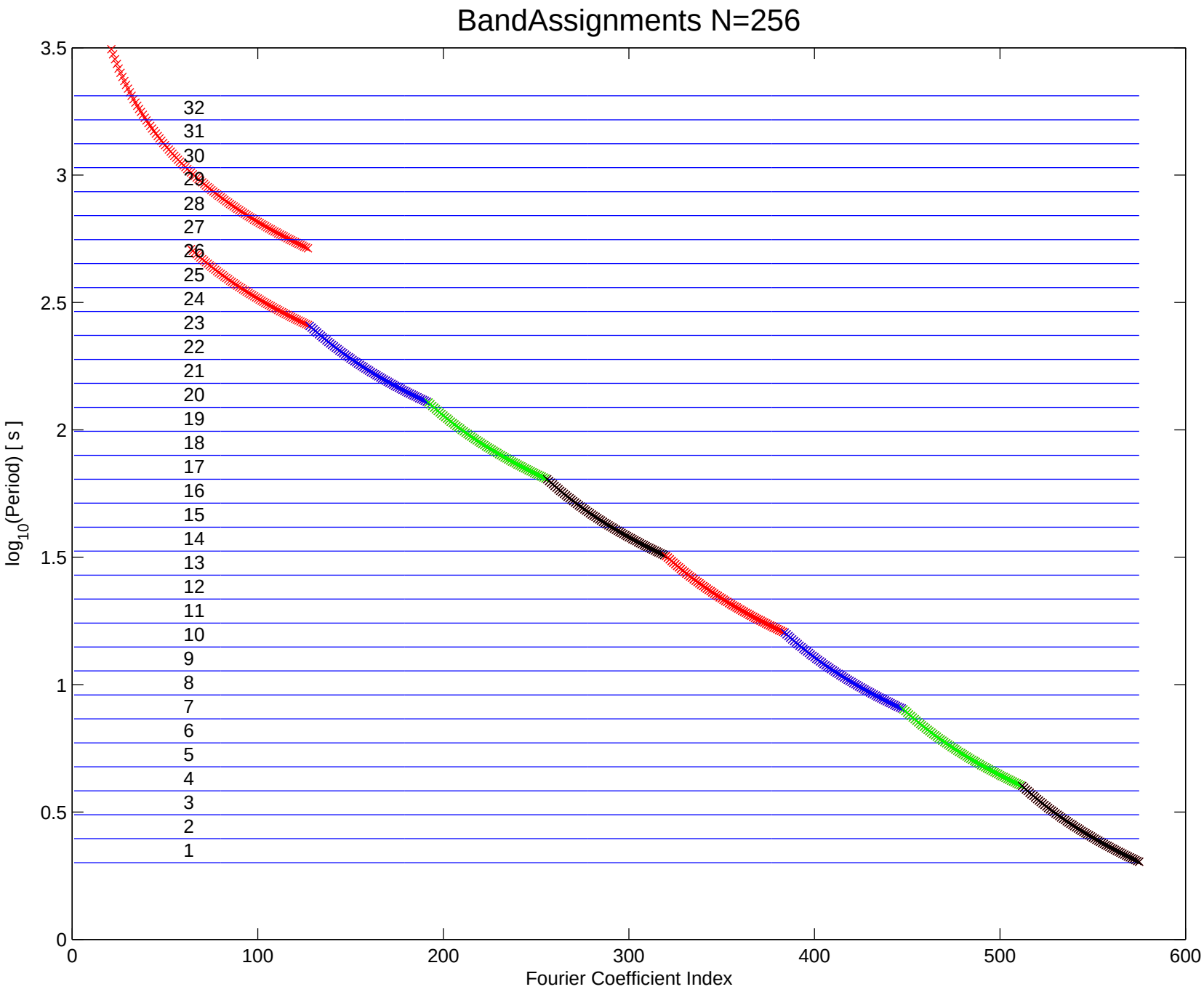


Figure 19. Distribution of Fourier coefficients among bands for window length of 256.

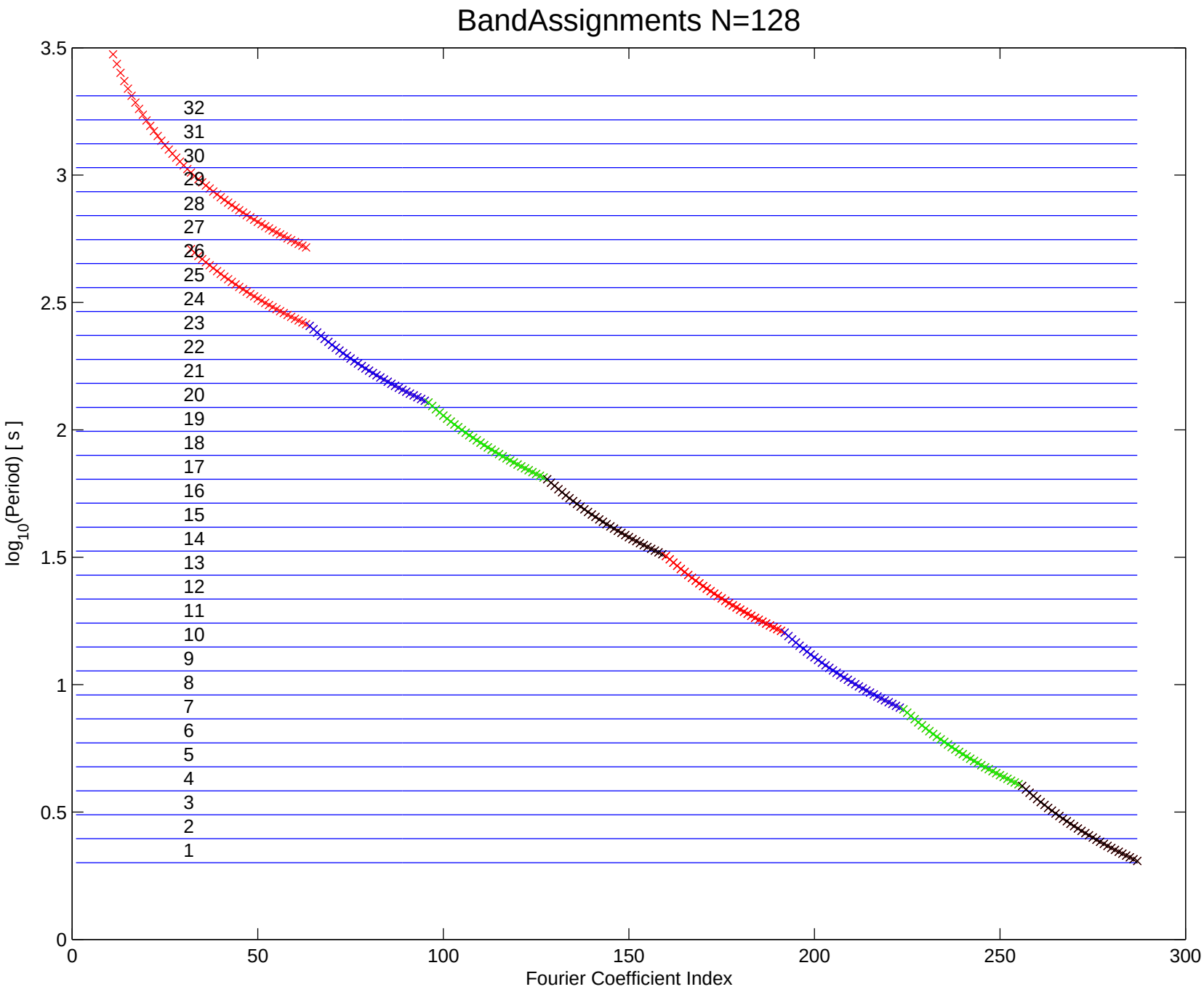


Figure 20. Distribution of Fourier coefficients among bands for window length of 128.

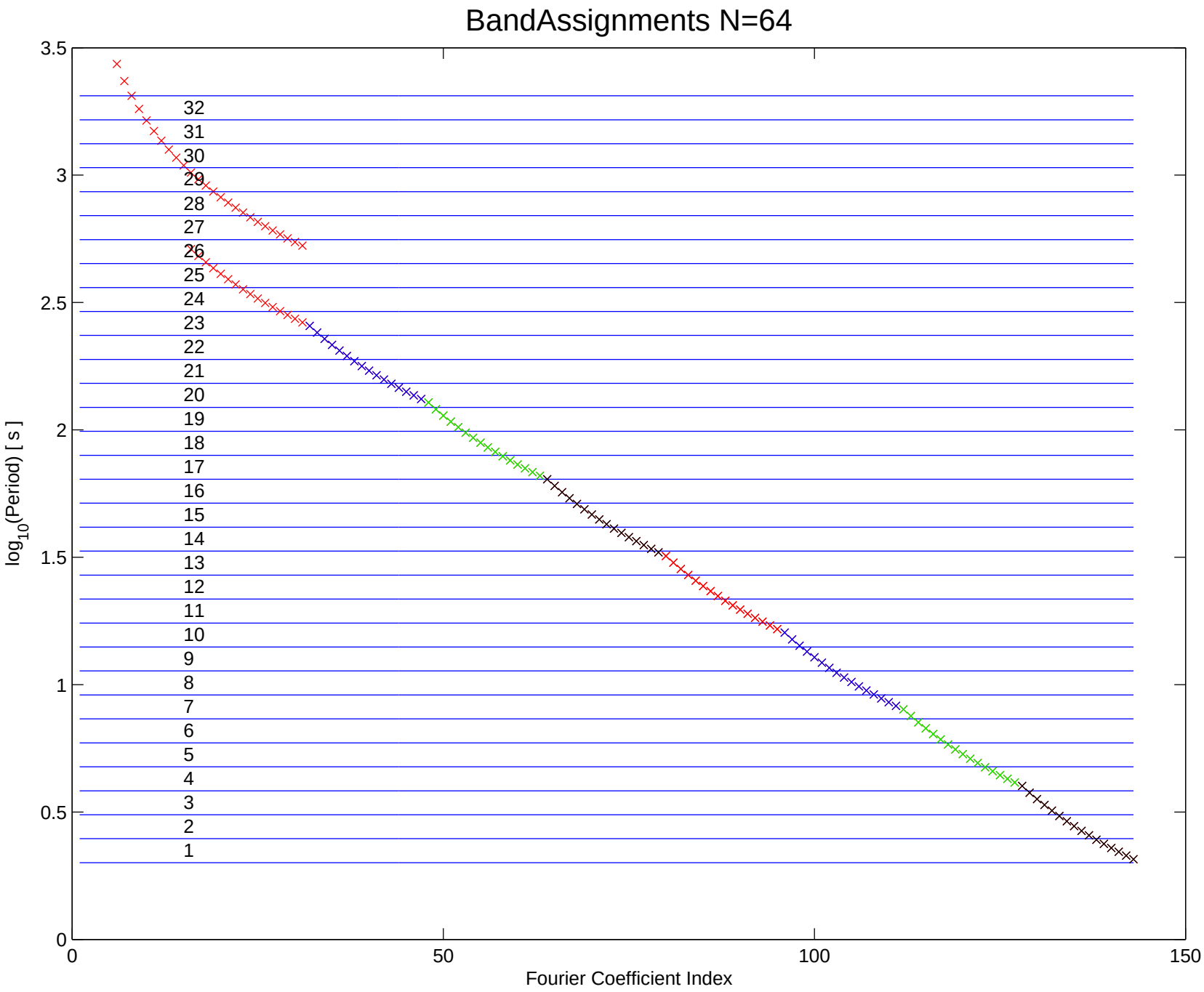


Figure 21. Distribution of Fourier coefficients among bands for window length of 64.

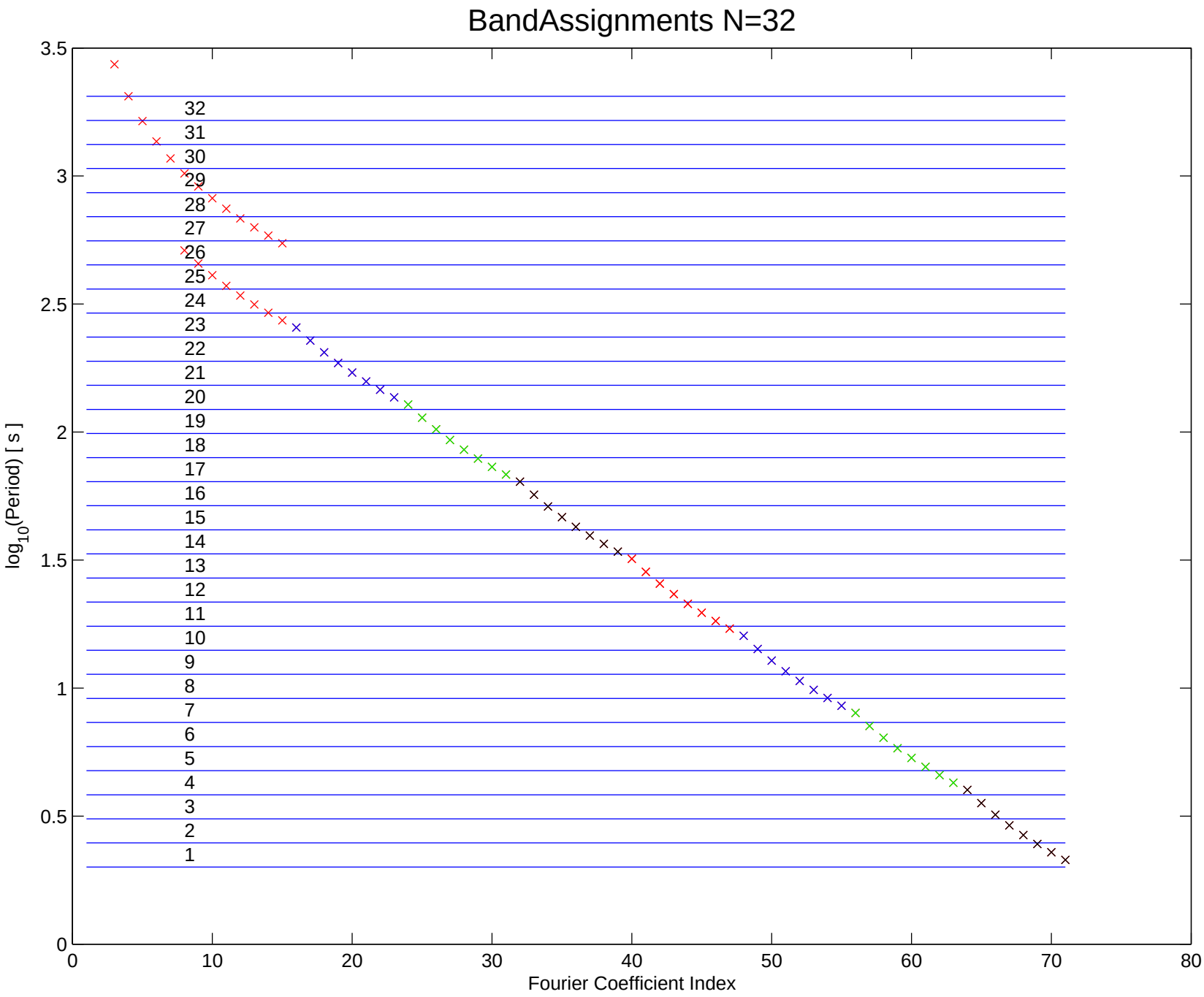


Figure 22. Distribution of Fourier coefficients among bands for window length of 32.

5 FAQs

Q: What are the codes for the various sampling rates?
A: V: 0.01 Hz; L 1 Hz; B; 40 Hz

Q:Where are the data stored after I download them?
A: In your data directory. At GUMP, this is /gp/ulfem/ULFEM/data/. Otherwise, the data directory can be found in setPaths.m. Inside the data folder are a collection of folders named by sampling rate and year; for example, 1-Hz data from 2004 would be inside the folder L2004/. Inside each of these "sampling rate-year" folders is a folder named TS (***T***ime ***S***eries); the raw data are stored in the TS folder as .mat files. The naming convention for these files is SITE_SR-CH_DDD.mat, where SITE is a three or four character code for the site; for example, PKD or JRSC, SR is the sampling-rate code, CH is the two character channel code used to queue the data from BSL (T1 for magnetometer, Q2 for electrode and so on), and DDD is the Julian date.

Q:How would a user distribute *.mat* files to other users?
A:Once the data are downloaded, they are stored in the **data** folder, in subfolders tagged by sampling rate and year. The data in these folders are "standalone" and can be distributed to any user with a licensed version of Matlab.

5.1 Known Bugs

1. TIn some data files, instrument changes occur during recording, and so more than one epoch is associated. Ensure that flags on the TS files are associated with these days.

2. The 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 because terminal asking for day 100 works but not for 277.

3. Subscript indices must be real or positive integers. When porting from a Windows to a Linux system, you may find an error in the download in loadBroken y.data=loadBroken(X,N); DATA(tIndex:tIndex+length(temp)-1)=temp;

   subscript indices must be real or positive integers. If this error occurs, check the value

of the variable hrShift in loadBroken.m. If it is negative, then most likely the scratch directory is not being cleaned out. Replace /bin/rm with rm in cleanScratch.m.

6 Acknowledgments

Frank Morrison of the University of California, Berkeley originally conceived of the ULF monitoring array and installed the early EM networks at Parkfield and Hollister Calif., in 1995. Gary Egbert of Oregon State University wrote the seed code for the time-series-plotting software. The development of the ULFEM Matlab package has been made possible by support from researchers Darcy K. McPhee and Jonathan Glen of the USGS, who together with Simon Klemperer of Stanford University have established more sites and continue to fund the array project. Barbara Romanowicz and the BSL staff have supported our efforts with data storage and processing facilities over the past 15 years. Without the contributions of all these people, our research would not be possible. We also thank Clark Dunson and Bruce Chuchel for critical reviews of this manuscript.

7 Appendix

7.1 Public-Private Key Setup

The following text is excerpted

http://bose.utmb.edu/Compu_Center/ssh/SSH_HOWTO.html}

Secure Shell (SSH) public key authentication can be used by a client to access servers, if properly configured. These notes describe how to configure OpenSSH for public key authentication, how to enable a ssh-agent to allow for passphrase-free logins, and tips on debugging problems with SSH connections. Password free logins benefit remote access and automation, for example if administering many servers or accessing version control software over SSH.

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 accessed through the first server connected to.

Never allow root-to-root trust between systems. If required by poorly engineered legacy scripts, limit the from access of the public keys, and if possible only allow specific public keys to run specific commands. Instead, setup named accounts for users or roles, and grant as little root access as possible via sudo.

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 generation varies under different implementations of SSH. The ssh -V command should print a line beginning with OpenSSH, followed by other details.

```
$ 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 port for the appropriate client configuration necessary to access the port.

Key Generation

A RSA key pair must be generated on the client system. The public portion of this key pair will reside on the servers being connected to, while the private portion needs to remain on a secure local area of the client system, by default in ~/.ssh/id_rsa. The key generation can be done with the ssh-keygen(1) utility.

```
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 least 16 characters long, and not a simple sentence. One choice would be several lines to a song or poem, interspersed with punctuation and other non-letter characters. The ssh-agent setup notes below will reduce the number of times this passphrase will need to be used, so using a long passphrase is encouraged.

The file permissions should be locked down to prevent other users from being able to read the key pair data. OpenSSH may also refuse to support public key authentication if the file permissions are too open. These fixes should be done on all systems involved.

```
$ 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 accessed by the client. The public key information to be copied should be located in the ~/.ssh/id_rsa.pub file on the client. Assuming that all of the servers use OpenSSH instead of a different SSH implementation, the public key data must be appended into the ~/.ssh/authorized_keys file on the servers.

```
# 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 keys may be in use. Each public key entry must be on a different line.

7.2 Code

7.2.1 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)
```

```
function [] = ArrayManger(hObject, eventdata, HHH)
%{
@type hObject:
@param hObject: handle to pushbutton (see GCBO)
@type eventdata:
@param eventdata: reserved - to be defined in a future version
   of MATLAB
@type HHH:
@param HHH:


@calls: prepArrayManagerGUI.m


%OUTPUTS: None, save the list of ARRAYS to disk.  For each
   array save the
%list of associated site-channel pairs.
%Key Variables: handles.UIC, one row for every possible site/
   channel combo
%First row is site id, 2nd row is chID, use +ve if selected, -
   ve if not
%There are 5 Frames in this GUI.
%Frame 1 allListbox; Shows choices of sites or channels
%Frame 2 Contains the Add/Remove Site Channel Buttons
%Frame 3 selectedListbox; Shows the selected sites/channels
%Frame 4 siteChListbox; The listbox on the bottom which shows
   all
%Frame 5 Contains The Array List, and the Add/Remove Array
   Buttons

%}
global ULFEMpath DATApath verbose SYSpath
prepArrayManagerGUI;
if verbose
```

```
    disp 'hobject'
    %get(hObject)
    disp 'hobject'
end
end

%% Called in prepArrayManager; FRAME 2
function handles = selUnselPBs_CreateFcn(hObject, eventdata,
   handles)
%{

@type hObject:
@param hObject: handle to pushbutton (see GCBO)
@type eventdata:
@param eventdata: reserved - to be defined in a future version
   of MATLAB
@type handles: struct
@param handles: structure with handles and user data (see
   GUIDATA)

@type homeFrame: integer
@param homeFrame: the integer which indexes the homeFrame in
handles.framePostions.  The homeFrame ?is the frame? where the
   selUnselPBs are
created
@type edgeWidth: the with of space left between frameEdge and
   button edge.

%}
homeFrame=2;
edgeWidth=0.1*handles.framePosns(homeFrame,3);
bttnWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
bttnHeight=0.1;
bttnH0=0.25;
```

```
handles.rmAllBttn=uicontrol('Parent',handles.ArryMgrFig,...
    'Style', 'pushbutton','units','normalized',...
    'Position',[handles.framePosns(homeFrame,1)+edgeWidth,...
    handles.horzLine+bttnH0+0*bttnHeight bttnWidth,bttnHeight
      ],...
    'BackgroundColor',[1 1 0],'Tag', 'rmAllBttn','tooltipstring
      ',...
    'rmAllBttn','String','Remove All <<--',...
    'callback',{@rmAllBttn_Callback,handles});
handles.addAllBttn=uicontrol('Parent',handles.ArryMgrFig,...
    'Style', 'pushbutton','units','normalized',...
    'Position',[handles.framePosns(homeFrame,1)+edgeWidth,...
    handles.horzLine+bttnH0+1*bttnHeight bttnWidth,bttnHeight
      ],...
    'BackgroundColor',[1 1 0],'Tag', 'addAllBttn',...
    'String','Add All -->>','callback',{@addAllBttn_Callback,
      handles});
handles.rmBttn=uicontrol('Parent',handles.ArryMgrFig,...
    'Style', 'pushbutton','units','normalized',...
    'Position',[handles.framePosns(homeFrame,1)+edgeWidth,...
    handles.horzLine+bttnH0+2*bttnHeight bttnWidth,bttnHeight
      ],...
    'BackgroundColor',[1 1 0],...
    'Tag', 'rmBttn','String','Remove <--',...
    'callback',{@rmBttn_Callback,handles});
handles.addBttn=uicontrol('Parent',handles.ArryMgrFig,...
    'Style', 'pushbutton','units','normalized',...
    'Position',[handles.framePosns(homeFrame,1)+edgeWidth,...
    handles.horzLine+bttnH0+3*bttnHeight bttnWidth,bttnHeight
      ],...
    'BackgroundColor',[1 1 0],'Tag', 'addBttn',...
    'String','Add-->','callback',{@addBttn_Callback,handles});
end

```

```
%% FRAME 2 CALLBACK
function  addBttn_Callback(hObject, eventdata, handles)
handles=guidata(handles.ArryMgrFig);
if get(handles.siteMode,'Value')
    addNdxs=get(handles.allListbox,'Value');
    siteCodes=handles.SITES(addNdxs);
    UIDlist=siteCodes2UID(siteCodes,handles.SITES);
    for iSite=UIDlist
        siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
        handles.UIC(siteSubArrayIndex,1)=abs(handles.UIC(
           siteSubArrayIndex,1));
    end
elseif get(handles.channelMode,'Value')
    iSite=get(handles.siteChListbox,'Value');
    siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
    subArray=handles.UIC(siteSubArrayIndex,:)
    if find(subArray(:,1)<0)
        display(['this site is not available'])
    else
        addNdxs=get(handles.allListbox,'Value');
        chCodes=handles.CHANNELS{iSite}(addNdxs)
        UIDlist=chCodes2UID(chCodes,handles.CHANNELS{iSite});
        subArray(UIDlist,2)=abs(subArray(UIDlist,2))
        handles.UIC(siteSubArrayIndex,:)=subArray;
    end
end
handles=updateSelected(hObject, eventdata,handles);
end

%% FRAME 2
function  handles=rmBttn_Callback(hObject, eventdata, handles)
handles=guidata(handles.ArryMgrFig);
if get(handles.siteMode,'Value')
    selectedSitesList=get(handles.selectedListbox,'String');
```

```
    if numel(selectedSitesList)==0
        set(handles.selectedListbox,'Value',[]);
    end
    rmNdxs=get(handles.selectedListbox,'Value');
    if length(rmNdxs)>0
        siteCodes=selectedSitesList(rmNdxs);
        UIDlist=siteCodes2UID(siteCodes,handles.SITES);
        for iSite=UIDlist
            siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite
               );
            handles.UIC(siteSubArrayIndex,1)=-abs(handles.UIC(
               siteSubArrayIndex,1));
        end
    end
elseif get(handles.channelMode,'Value')
    iSite=get(handles.siteChListbox,'Value');
    siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
    subArray=handles.UIC(siteSubArrayIndex,:);
    if find(subArray(:,1)<0)
        display(['this site is not available'])
    else
        selectedChsList=get(handles.selectedListbox,'String');
        rmNdxs=get(handles.selectedListbox,'Value');
        chCodes=selectedChsList(rmNdxs);
        UIDlist=chCodes2UID(chCodes,handles.CHANNELS{iSite});
        subArray(UIDlist,2)=-abs(subArray(UIDlist,2));
        handles.UIC(siteSubArrayIndex,:)=subArray;
    end
end
handles=updateSelected(hObject, eventdata,handles);
end

%% FRAME 2
```

```
function handles=addAllBttn_Callback(hObject, eventdata,
   handles)
handles=guidata(handles.ArryMgrFig);
if get(handles.siteMode,'Value')
    handles.UIC(:,1)=abs(handles.UIC(:,1));
    guidata(handles.ArryMgrFig,handles);
    handles=updateSelected(hObject, eventdata,handles);
elseif get(handles.channelMode,'Value')
    iSite=get(handles.siteChListbox,'Value')
    siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
    subArray=handles.UIC(siteSubArrayIndex,:);
    if find(subArray(:,1)<0)
        display(['this site is not available'])
    else
        handles.UIC(siteSubArrayIndex,2)=abs(handles.UIC(
           siteSubArrayIndex,2));
        guidata(handles.ArryMgrFig,handles);
        handles=updateSelected(hObject, eventdata,handles);
    end
end
end

%% FRAME 2
function rmAllBttn_Callback(hObject, eventdata, handles)
handles=guidata(handles.ArryMgrFig);
if get(handles.siteMode,'Value')
    handles.UIC(:,1)=-1*abs(handles.UIC(:,1));
    guidata(handles.ArryMgrFig,handles);
    handles=updateSelected(hObject, eventdata,handles);
    %guidata(handles.ArryMgrFig,handles);

elseif get(handles.channelMode,'Value')
    display(['rmAll in Channel Mode'])
    iSite=get(handles.siteChListbox,'Value');
```

```
182     display(['Sitenum for rmAll is ',num2str(iSite)])
183     siteSubIndex=find(abs(handles.UIC(:,1))==iSite);
184     subArray=handles.UIC(siteSubIndex,:);
185     if find(subArray(:,1)<0)
186         display(['this site is not available'])
187     else
188         handles.UIC(siteSubIndex,2)=-abs(handles.UIC(
              siteSubIndex,2));
189         display(['Rows ',num2str(siteSubIndex(1)),':',num2str(
              siteSubIndex(end)),' set to negative'])
190         guidata(handles.ArryMgrFig,handles);
191         handles=updateSelected(hObject, eventdata,handles);
192     end
193 end
194 end
195
196 %% Called in prepArrayManager FRAME 2
197 function handles=siteChRadio_CreateFcn(hObject, eventdata,
      handles)
198 homeFrame=2;
199 edgeWidth=0.1*handles.framePosns(homeFrame,3);
200 bttnWidth=(handles.framePosns(homeFrame,3)-2*edgeWidth);
201 bttnHeight=0.1;
202 bttnH0=0.06;
203 handles.siteChRadio = uibuttongroup('Position',[0 bttnHeight+
      bttnH0 1 2*bttnHeight],...
204     'BackgroundColor',[1 0.8 0.8],...
205     'units','normalized',...
206     'Tag','SITEorCHradio',...
207     'Parent',handles.Frame{2});
208 handles.siteMode = uicontrol('Style','Radio','String','SITES'
      ,...
209     'Position',[1 4 58 30],'parent',handles.siteChRadio,'
          HandleVisibility','on');
```

```
210 handles.channelMode = uicontrol('Style','Radio','String','
      CHANNELS',...
211     'pos',[59 4 80 30],'parent',handles.siteChRadio,'
          HandleVisibility','callback');
212 set(handles.siteChRadio,'SelectionChangeFcn',{@selcbk,handles})
      ;
213 set(handles.siteChRadio,'SelectedObject',[]);
214 set(handles.siteChRadio,'Visible','on');
215 set(handles.siteMode,'Value',1);
216 end
217
218
219 %% Called in prepArrayManager FRAME 5
220 function handles=addArrayPB_CreateFcn(hObject, eventdata,
      handles)
221 homeFrame=5;
222 edgeWidth=0.1*handles.framePosns(homeFrame,3);
223 bttnWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
224 bttnHeight=0.1;
225 bttnH0=0.05;
226 handles.addArrayPB=uicontrol( ...
227     'Parent',handles.ArryMgrFig,...
228     'Style', 'pushbutton',...
229     'units','normalized',...
230     'Position',[handles.framePosns(homeFrame,1)+edgeWidth,
          bttnH0+0*bttnHeight bttnWidth,bttnHeight],...
231     'BackgroundColor',[1 1 0],...
232     'Tag', 'addArrayBttn',...
233     'String','Create New',...
234     'callback',{@createArrayPB_Callback,handles});
235
236 end
237
238
```

```
%% Called in prepArrayManager FRAME 5
function handles=rmArrayPB_CreateFcn(hObject, eventdata,
   handles)
homeFrame=5;
edgeWidth=0.1*handles.framePosns(homeFrame,3);
bttnWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
bttnHeight=0.1;
bttnH0=0.05;
handles.rmArrayPB=uicontrol( ...
    'Parent',handles.ArryMgrFig,...
    'Style', 'pushbutton',...
    'units','normalized',...
    'Position',[handles.framePosns(homeFrame,1)+edgeWidth,...
    bttnH0+1*bttnHeight, bttnWidth,bttnHeight],...
    'BackgroundColor',[1 1 0],...
    'Tag', 'rmArrayBttn',...
    'String','Remove',...
    'callback',{@removeArrayPB_Callback,handles});

end

%%
function handles=createArrayPB_Callback(hObject, eventdata,
   handles)
%create a window to prompt the user for a name of a new array
global ULFEMpath
namerPosn=[380,550,200,100];
namerFig=figure('MenuBar','none','Name', 'NAMER', ...
    'NumberTitle','off','Position',namerPosn,...
    'units','normalized','tag','NAMER');
nameBoxPosn=[0.1 0.5,0.8, 0.2];
OKPBPosn=[0 .1 .30 .20];
cancelPBPosn=[.45 .10 .30 .20];

```

```
nameBox=uicontrol('Parent',namerFig,'Style','edit',...
    'units','normalized','position',nameBoxPosn,...
    'String','ARRAY_ID');
OKPB=uicontrol('Parent',namerFig,'Style','pushbutton',...
    'Units','Normalized','Position',OKPBPosn,...
    'String','OK','Callback',{@OKfcn, handles});
CANCELPB=uicontrol('Parent',namerFig,'Units','Normalized',...
    'Position',cancelPBPosn,'Style','pushbutton',...
    'String','CLOSE','Callback',{@cancelfcn, handles});
OKtxt=uicontrol('units','normalized','position'
   ,[0.1,0.8,0.8,0.2],...
    'style','text','string','Assign a Name to Array','fontsize'
       ,12,'Parent',namerFig);
%guidata(handles.ArryMgrFig,handles);
%%
function []=OKfcn(hObject, eventdata, handles)
        %update the ARRAY LIST
        %first make sure new array name is not already used.
        %then add to handles.ARRAYs and write to sys/SITES/
           ARRAYS
        handles=guidata(handles.ArryMgrFig);
        newName=get(nameBox,'String')
        nArrays=numel(handles.ARRAYList);
        for iArray=1:nArrays
            match=strmatch(handles.ARRAYList{iArray},newName,'
               exact');
            if match
                warnBox(['Name "',newName,'" already used'])
                break
            end
        end
        if numel(match)==0
            ndx=get(handles.arrayListbox,'Value');
            handles.ARRAYS{nArrays+1}=handles.ARRAYS{ndx};
```

```
301             handles.ARRAYS{nArrays+1}.name=newName;
302             for rA=1:nArrays+1
303                 handles.ARRAYList{rA}=handles.ARRAYS{rA}.name;
304                 display(['ARRAY #',num2str(rA),' ',handles.
                        ARRAYList{rA}])
305             end
306             set(handles.arrayListbox,'String',handles.ARRAYList
                    );
307             ARRAYS=handles.ARRAYS;
308             arrayFilename = fullfile(SYSpath,'SITES','ARRAYS.
                    mat');
309             cmd=['save ',arrayFilename,' ARRAYS']
310             eval(cmd)
311         end
312         guidata(handles.ArryMgrFig,handles);
313     end
314     function []=cancelfcn(hObject, eventdata, handles)
315         close(namerFig)
316     end
317 end
318 %%
319 function handles=removeArrayPB_Callback(hObject, eventdata,
       handles)
320 %create a window to ask the user if they wish to remove array
321 global ULFEMpath
322 handles=guidata(handles.ArryMgrFig);
323 aiqNdx=get(handles.arrayListbox,'Value');
324 aiq=handles.ARRAYList{aiqNdx};
325 rmerPosn=[380,550,200,100];
326 rmerFig=figure('MenuBar','none','Name', 'REMOVE', ...
327     'NumberTitle','off','Position',rmerPosn,...
328     'units','normalized','tag','RMER');
329 rmBoxPosn=[0.1 0.5,0.8, 0.2];
330 OKPBPosn=[0 .05 .30 .20];
```

```
cancelPBPosn=[.35 .05 .30 .20];
OKPB=uicontrol('Parent',rmerFig,'Style','pushbutton',...
    'Units','Normalized','Position',OKPBPosn,...
    'String','OK','Callback',{@OKfcn, handles})
CANCELPB=uicontrol('Parent',rmerFig,'Units','Normalized',...
    'Position',cancelPBPosn,'Style','pushbutton',...
    'String','CANCEL','Callback',{@cancelfcn, handles});
OKtxt=uicontrol('units','normalized','position'
   ,[0.1,0.3,0.8,0.6],...
    'style','text','string',['Are you sure you want to remove
       the array: ',aiq],...
    'fontsize',12,'Parent',rmerFig);

    function d=OKfcn(hObject, eventdata, handles)
        handles=guidata(handles.ArryMgrFig);
        ndx=get(handles.arrayListbox,'Value');
        if ndx>2
            defaultArray=1;
            set(handles.arrayListbox,'Value',defaultArray)
            handles.UIC=handles.ARRAYS{defaultArray}.UIC;
            handles.ARRAYS(ndx)=[];
            handles.ARRAYList=[];
            for rA=1:numel(handles.ARRAYS)
                handles.ARRAYList{rA}=handles.ARRAYS{rA}.name;
            end
            set(handles.arrayListbox,'String',handles.ARRAYList
               );
            ARRAYS=handles.ARRAYS;
            arrayFilename = fullfile(SYSpath,'SITES','ARRAYS.
               mat');
            cmd=['save ',arrayFilename,' ARRAYS']
            eval(cmd)
        else
            warnBox('Cannot remove ARRAY ALL or NONE')
```

```
361         end
362         guidata(handles.ArryMgrFig,handles);
363         %handles=updateSelected(hObject, eventdata,handles);
364         close(rmerFig)
365     end
366     function []=cancelfcn(hObject, eventdata, handles)
367         close(rmerFig)
368     end
369 
370 end
371 
372 
373 %% Called in prepArrayManager FRAME 5
374 function handles=arrayListCreateFcn(hObject, eventdata, handles
      )
375 homeFrame=5;
376 roomForBttn=0.3;
377 edgeWidth=0.1*handles.framePosns(homeFrame,3);
378 listWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
379 listHeight=handles.framePosns(homeFrame,4)-handles.topEdge*
      edgeWidth;
380 handles.arrayListbox=uicontrol( ...
381     'Parent',handles.ArryMgrFig,...
382     'Style', 'listbox',...
383     'units','normalized',...
384     'Position',[handles.framePosns(homeFrame,1)+edgeWidth,
          edgeWidth+roomForBttn,listWidth,listHeight-roomForBttn
          ],...
385     'BackgroundColor',[1 1 0],...
386     'ToolTipString','This Name represents a collection of
          instruments called an Array',...
387     'Tag', 'array_Listbox',...
388     'String',handles.ARRAYList,...
389     'callback',{@arraySel_Callback,handles},...
```

```
    'max',1);
end

%%
function handles=arraySel_Callback(hObject,eventdata,handles)
%This is the callback when you tooggle between array names in
   arrayListbox
%For each array, need a list of sites which are REJECTED.
   These need to be
%given negative indices in UIC
    sm0=get(handles.siteMode,'Value');
    cm0=get(handles.channelMode,'Value');
    set(handles.siteMode,'Value',1);
    set(handles.channelMode,'Value',0);
    handles=guidata(handles.ArryMgrFig);
    ARRAY_uid=get(handles.arrayListbox,'Value');
    handles.UIC=handles.ARRAYS{ARRAY_uid}.UIC;
    handles.ARRAYS{ARRAY_uid}.UIC;
    handles=updateSelected(hObject, eventdata,handles);
    set(handles.siteMode,'Value',sm0);
    set(handles.channelMode,'Value',cm0);
end

%% FRAME 2
%Callback for changing the Site/Channel mode radiobutton
function handles=selcbk(source,eventdata,handles)
handles=guidata(handles.ArryMgrFig);
SorC=get(get(source,'SelectedObject'),'String')
set(handles.allListbox,'Value',[]);
set(handles.selectedListbox,'Value',[]);
if regexp(SorC,'SITES')
    set(handles.allListbox,'String',handles.SITES)
    set(handles.siteChListbox,'Enable','off');
    set(handles.selectedListbox,'String',handles.selSites);
```

```
422     set(handles.allText,'string','All Sites')
423     set(handles.selText,'string','Selected Sites')
424 elseif regexp(SorC,'CHANNELS')
425     set(handles.siteChListbox,'Value',1);
426     set(handles.siteChListbox,'Enable','on');
427     set(handles.siteChListbox,'Callback',{
          @siteChListbox_Callback,handles});
428     set(handles.allListbox,'String',handles.CHANNELS{1})
429     siteSubArrayIndex=find(abs(handles.UIC(:,1))==1);
430     if find(handles.UIC(siteSubArrayIndex,1)<0)
431         set(handles.selectedListbox,'String',{});
432     else
433         selChs=find(handles.UIC(siteSubArrayIndex,2)>0);
434         set(handles.selectedListbox,'String',handles.CHANNELS
              {1}(selChs));
435     end
436     set(handles.allText,'string','All Channels');
437     set(handles.selText,'string','Selected Channels');
438 end
439 end
440
441 %%
442 function handles=updatePB_CreateFcn(hObject, eventdata,handles)
443 homeFrame=2;
444 edgeWidth=0.1*handles.framePosns(homeFrame,3);
445 bttnWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
446 bttnHeight=0.1;
447 bttnH0=0.01;
448 handles.nextBttn=uicontrol( ...
449     'Parent',handles.ArryMgrFig,...
450     'Style', 'pushbutton',...
451     'units','normalized',...
452     'Position',[handles.framePosns(homeFrame,1)+edgeWidth
          bttnH0+handles.horzLine bttnWidth bttnHeight],...
```

```
453     'BackgroundColor',[1 0 0],...
454     'Tag', 'updatePB',...
455     'String',['UPDATE'],...
456     'callback',{@updatePB_Callback,handles});
457 end
458 %%
459 function updatePB_Callback(hObject, eventdata, handles)
460 %Write the ARRAYS data structure to .mat file
461 global ULFEMpath verbose SYSpath
462 handles=guidata(handles.ArryMgrFig);
463 ARRAYS=handles.ARRAYS;
464 arrayFilename = fullfile(SYSpath,'SITES','ARRAYS.mat');
465 cmd=['save ',arrayFilename,' ARRAYS']
466 eval(cmd)
467 end
468 %%
469 function UIDlist=siteCodes2UID(siteCodes,siteList)
470 s=siteCodes;
471 UIDlist=[];
472 for ess=1:numel(s)
473     UIDlist=[UIDlist find(strcmp(s{ess},siteList)==1)];
474 end
475 UIDlist=sort(UIDlist);
476 end
477 %%
478 function UIDlist=chCodes2UID(chCodes,chList)
479 c=chCodes
480 UIDlist=[];
481 for see=1:numel(c)
482     UIDlist=[UIDlist find(strcmp(c{see},chList)==1)];
483 end
484 UIDlist=sort(UIDlist);
485 end
486 %%
```

```
function handles=siteChListbox_Callback(hObject, eventdata,
   handles)
handles=guidata(handles.ArryMgrFig);
iSite=get(handles.siteChListbox,'Value');
siteIsSelected=find(handles.UIC(:,1)==iSite)
if siteIsSelected
    handles=updateSelected(hObject, eventdata,handles);
else
    set(handles.allListbox,'String',handles.CHANNELS{iSite});
    set(handles.selectedListbox,'Value',[]);
    set(handles.selectedListbox,'String','X');%handles.
       siteChList{iSite});
end
end
%%
function handles=updateSelected(hObject, eventdata,handles)
%{
For each possible site, identify the appropriate bookeeping
   subarray and
determine if the site is selected (are one or more channels
   selected) and which
channels are selected.  This routine is run after just about
   anything the user
does to keep a current list of what the selected sites and
   channels are. I want
this routine to take as input only the UIC object
%}
global ULFEMpath verbose
display(['Entering the Update Selected Subroutine'])
handles.UIC;
%get(handles.siteMode,'Value')
%get(handles.channelMode,'Value')
if get(handles.siteMode,'Value')
    if get(handles.channelMode,'Value')
```

```
515         display(['ERROR both SITE MODE and channel mode are
               somehow active'])
516     end
517     if verbose
518         display(['SITE MODE'])
519     end
520     selSites=[];
521     handles.siteChTextList=[];
522     for iSite=1:numel(handles.SITES)
523         siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
524         %Site is part of array if its num positive somewhere
525         if find(handles.UIC(siteSubArrayIndex,1)==iSite)
526             selSites=[selSites iSite];
527         else
528             display(['site ',handles.SITES{iSite},' is not
                   selected']);
529         end
530         chListIndex=find(handles.UIC(siteSubArrayIndex,2)>0);
531         handles.siteChList{iSite}=handles.CHANNELS{iSite}(
               chListIndex);
532         handles.siteChTextList{iSite}=[handles.SITES{iSite},'
               -- {' ];
533         if find(selSites==iSite)
534             numch2list=numel(chListIndex);
535             for ch=1:numch2list
536                 handles.siteChTextList{iSite}=[handles.
                       siteChTextList{iSite},' ',handles.siteChList{
                       iSite}{ch}];
537             end
538         else
539             handles.siteChTextList{iSite}=[handles.
                   siteChTextList{iSite},'XXX'];
540         end
```

```
541         handles.siteChTextList{iSite}=[handles.siteChTextList{
               iSite},'}'];
542     end
543     handles.selSites=handles.SITES(selSites);
544     handles.siteChTextList;
545     if isfield(handles,'siteChListbox')
546         set(handles.siteChListbox,'String',handles.
               siteChTextList);
547         if numel(handles.siteChTextList)>0
548             set(handles.selectedListbox,'Value',1);
549         end
550     end
551     if isfield(handles,'selectedListbox')
552         display('updating selected box')
553         selSites;
554         handles.selSites;%=selSites;
555         set(handles.selectedListbox,'String',handles.selSites);
556     end
557     guidata(handles.ArryMgrFig,handles);
558 
559 elseif get(handles.channelMode,'Value')
560     display(['Channel Mode\n'])
561     handles.siteChTextList;
562     iSite=get(handles.siteChListbox,'Value');
563     siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
564     subArray=handles.UIC(siteSubArrayIndex,:);
565     chListIndex=find(handles.UIC(siteSubArrayIndex,2)>0);
566     handles.siteChList{iSite}=handles.CHANNELS{iSite}(
          chListIndex);
567     handles.siteChTextList{iSite}=[handles.SITES{iSite},' -- {'
          ];
568     numch2list=numel(chListIndex);
569     for ch=1:numch2list
```

```
570         handles.siteChTextList{iSite}=[handles.siteChTextList{
               iSite},' ',handles.siteChList{iSite}{ch}];
571     end
572     handles.siteChTextList{iSite}=[handles.siteChTextList{iSite
           },'}'];
573     set(handles.selectedListbox,'Value',1);
574     set(handles.allListbox,'Value',1);
575     if isfield(handles,'selectedListbox')
576         display('updating selected box')
577         handles.siteChList{iSite};
578         set(handles.selectedListbox,'String',handles.siteChList
               {iSite});
579     end
580     if isfield(handles,'siteChListbox')
581         set(handles.siteChListbox,'String',handles.
               siteChTextList);
582         subArray(:,2);
583         find(subArray(:,2)>0)
584         if find(subArray(:,2)>0)
585             disp('a3')
586             set(handles.selectedListbox,'Value',1);
587         else
588             disp('a4')
589             set(handles.selectedListbox,'Value',[]);
590         end
591     end
592     set(handles.allListbox,'String',handles.CHANNELS{iSite});
593 end
594
595 %update the seleted ARRAY
596 if isfield(handles,'arrayListbox')
597     ndx=get(handles.arrayListbox,'Value');
598     handles.ARRAYS{ndx}.UIC=handles.UIC;
```

```
599      handles.ARRAYS{ndx}.chrArray=num2chr_relDB(handles.UIC,
            handles.SITES,handles.CHANNELS,handles.NETWORKS);
600  end
601  guidata(handles.ArryMgrFig,handles);
602  %handles.siteChTextList
603  end
```

```
function [varargout]=Get_NCEDC_GUI(varargin)
%{
A GUI TO GET DATA from NCEDC.
This function is the callback to the Get NCEDC button on the
   ULFEM GUI.

%}

global ULFEMpath ulfempath DATApath ARRAYS syspath verbose
   PYTHONpath
%Get metadata needed to setup GUI about Arrays and Sampling
   Rates
ULFEM_environment()
%% Initiate the GUI and specify its geometery
for GUI_specs=1:1
handles.GET_GUI_POSN=[20,550,450,230];
getDataGui=figure('MenuBar','none','Name', 'Get Data', '
   NumberTitle','off',...
                  'Position',handles.GET_GUI_POSN,'units','
                     normalized',...
                  'tag','GETTSGUI', 'Color', [0.1 0.1 0.5]);
wd1=0.30;
wd2=0.66*wd1;
ht1=0.1;
epsln=0.01;
xx=0.041;%x and y shifts for date seelction text and edit boxes
yy=-0.15;%x and y shifts for date seelction text and edit boxes
scl=0.66;
textBoxColour = [0.5 0.5 1];
SRtxtpsn=[0 1-ht1-epsln wd1 ht1];
SRpoppsn=[0 1-2*ht1-epsln wd1 ht1];
arraySeltxtpsn=[wd1+epsln 1-ht1-epsln 1.3*wd1 ht1];
arraySelpoppsn=[wd1+epsln 1-2*ht1-epsln 1.3*wd1 ht1];

```

```
starttxtpsn=[xx+wd2+epsln yy+1-2*ht1+epsln wd2 0.9*ht1]; %row1
endtxtpsn=[xx+2*wd2+2*epsln yy+1-2*ht1+epsln wd2 0.9*ht1];%row1

YYtxtpsn=[xx+0 yy+1-3*ht1 wd2 0.9*ht1];%row2
YYstarteditpsn=[xx+wd2+epsln yy+1-3*ht1 wd2 ht1];%row2
YYendeditpsn=[xx+2*wd2+2*epsln yy+1-3*ht1 wd2 ht1];%row2

MMDDtxtpsn=[xx+0 yy+1-(4*ht1) wd2 0.9*ht1];%row3
MMstarteditpsn=[xx+wd2+epsln yy+1-4*ht1 wd2/2 ht1];
MMendeditpsn=[xx+2*wd2+2*epsln yy+1-4*ht1 wd2/2 ht1];
DDstarteditpsn=[xx+1.5*wd2+epsln yy+1-4*ht1 wd2/2 ht1];
DDendeditpsn=[xx+2.5*wd2+2*epsln yy+1-4*ht1 wd2/2 ht1];%row3

HRtxtpsn=[xx+0 yy+1-(5)*ht1 wd2 0.9*ht1];%row4
HRstarteditpsn=[xx+wd2+epsln yy+1-5*ht1 wd2 ht1];
HRendeditpsn=[xx+2*wd2+2*epsln yy+1-5*ht1 wd2 ht1];%row4
PLOT_X = xx+0*wd2;
PLOT_Y = yy+1-6.3*ht1;
PLOT_WIDTH = wd1/2-epsln;
PLOT_HEIGHT = ht1;
PLOTcbpsn=[PLOT_X PLOT_Y PLOT_WIDTH PLOT_HEIGHT];
%STTcbpsn=[xx+1*wd2-2*epsln yy+1-6.3*ht1 wd1/2 ht1];
OVERWRITE_X = xx+1*wd2-1*epsln;
OVERWRITE_Y = PLOT_Y;
OVERWRITE_WIDTH = wd2;
OVERWRITE_HEIGHT = ht1;
OVERWRITEcbpsn=[OVERWRITE_X OVERWRITE_Y OVERWRITE_WIDTH
   OVERWRITE_HEIGHT];
NPY_X = OVERWRITE_X + OVERWRITE_WIDTH + 3*epsln;
NPYcbpsn=[NPY_X PLOT_Y PLOT_WIDTH ht1];


GETDATApbpsn=[wd1 1-9.9*ht1 wd1 ht1];
```

```
63 [YYYY MM DD]=yesterday;
64
65 handles.Parent=getDataGui;
66
67
68 %make the buttons and boxes
69 handles.SRtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text',...
70     'String','Sampling Rate','units','normalized','Position',
          SRtxtpsn,...
71     'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
72 handles.SRpopup = uicontrol('Style', 'popupmenu','String',
      SRSTRINGS,...
73                     'Units','normalized','Position',SRpoppsn);
74 handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
      , 'text',...
75     'String','SELECT ARRAY','units','normalized','Position',...
76     arraySeltxtpsn, 'BackgroundColor', textBoxColour,'FontName'
          ,...
77     'FixedWidth');
78 handles.ArraySelPopup=uicontrol('Parent', handles.Parent, '
      Style', ...
79     'popupmenu','String',arrayList,'units','normalized',...
80     'Position',arraySelpoppsn);
81 handles.starttxt=uicontrol('Parent', handles.Parent, 'Style', '
      text',...
82     'String','Start','units','normalized','Position',
          starttxtpsn, ...
83     'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
84 handles.endtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text', ...
85     'String','End','units','normalized','Position',endtxtpsn,
          ...
86     'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
```

```
handles.YYtxt=uicontrol('Parent', handles.Parent, 'Style', 'text',...
    'String','YEAR','units', 'normalized','Position',YYtxtpsn,...
    'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
handles.YYstartedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',YYYY,'units','normalized','Position',
      YYstarteditpsn, ...
    'Tag','YYeditTag');
handles.YYendedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',YYYY,'units','normalized','Position',YYendeditpsn,
       ...
    'Tag','YYeditTag');
handles.MMDDtxt=uicontrol('Parent', handles.Parent, 'Style', 'text',...
    'String','MM DD','units','normalized','Position',MMDDtxtpsn,...
    'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
handles.MMstartedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',MM,'units','normalized','Position',MMstarteditpsn);
handles.DDstartedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',DD,'units','normalized','Position',DDstarteditpsn);
handles.MMendedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',MM,'units','normalized','Position',MMendeditpsn);
handles.DDendedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String',DD,'units','normalized','Position',DDendeditpsn);
```

```
handles.HRtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text',...
    'String','UT HH','units','normalized','Position',HRtxtpsn
       ,...
    'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
handles.HRstartedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit',...
    'String','HH','units','normalized','Position',
       HRstarteditpsn);
handles.HRendedit=uicontrol('Parent', handles.Parent, 'Style',
   'edit',...
    'String','HH','units','normalized','Position',HRendeditpsn)
       ;
handles.PLOTcb=uicontrol('Parent', handles.Parent, 'Style', '
   Checkbox',...
    'String','PLOT','units','normalized','Position',PLOTcbpsn);
handles.NPYcb=uicontrol('Parent', handles.Parent, 'Style', '
   Checkbox',...
    'String','NPY','units','normalized','Position',NPYcbpsn);
%handles.STTcb=uicontrol('Parent', handles.Parent, 'Style', '
   Checkbox','String','STT','units',...
%    'normalized','Position',STTcbpsn);
handles.OVERWRITEcb=uicontrol('Parent', handles.Parent, 'Style'
   , ...
    'Checkbox','String','OVERWRITE','units','normalized',...
    'Position',OVERWRITEcbpsn);
handles.GETDATApb=uicontrol('Parent', handles.Parent, 'Style',
   ...
    'pushbutton','String','GET DATA','units','normalized','
       Position',...
    GETDATApbpsn,'callback',{@GETDATAFCN,handles}, '
       tooltipstring',...
    'Gets Data from NCEDC');
set(handles.SRpopup, 'callback', {@setSRcbk,handles});
```

```
end
end

%%

function [] = GETDATAFCN(hObject, eventdata, handles)
    %Collect Parameters Needed to Call GetAscii.m, i.e. the metadata needed to
    %tell NCEDC what time series we are interested in.  start time, end time,
    %sampling rate, then call data from BSL.

    %@note; Sep11, 2012; Encapsulate generation of a list of all segments
    %before performing downloads - reduce indentation in main
    global ULFEMpath DATApath ARRAYS verbose PYTHONpath
    GUIstate = readGetDataGUI(handles)
    metaDataCells = generateMetaDataCells(GUIstate);
    for iMeta =1:length(metaDataCells)
        meta=metaDataCells{iMeta};
        if exist(meta.fullName,'file') & GUIstate.OVERWRITEdata ==0
            display([meta.ext, 'File ',meta.fName,' already exists, skipping download'])
        else %meta.ext == 'mat'
            y=getAscii(meta); %currently deals with mat and npy
            display(['Got ASCII file "y.data" and its size is:',num2str(length(y.data))])
            if meta.ext == 'mat'
                if GUIstate.plotAsYouDownload
                    plotDownloadedData(y,meta)
                end
                svCmd=['save ',meta.fName,' y'];
                eval(svCmd);
```

```
            end
        end
    end
    display('Data Download Complete')
    displayDoneDownloadBox()
end %End FUnciton Definition


function []=setSRcbk(hObject, eventdata, handles)
    SR = getSR(get(handles.SRpopup,'Value'));
    if SR=='L' | SR=='V'
        set(handles.HRstartedit,'String','00:00');
        set(handles.HRstartedit,'style','text');
        set(handles.HRendedit,'String','24:00');
        set(handles.HRendedit,'style','text');
    else
        set(handles.HRstartedit,'String','HH');
        set(handles.HRstartedit,'style','edit')
        set(handles.HRendedit,'String','HH');
        set(handles.HRendedit,'style','edit');
    end
end

function displayDoneDownloadBox()
%create a window to prompt the user download is finished
uiPosn=[380,550,200,100];
finitoFig=figure('MenuBar','none','Name', 'Finished!', ...
    'NumberTitle','off','Position',uiPosn,...
    'units','normalized','tag','fin');
msgBoxPosn=[0.1 0.5,0.8, 0.5];
msgBox=uicontrol('Parent',finitoFig,'Style','text',...
    'units','normalized','position',msgBoxPosn,...
    'String','Download Complete','fontsize',16);
end
```

```
190
191 function [yStr, dStr] = StringsFromDatenum(dayteNum)
192     %
193     ds=split('-',datestr(dayteNum));
194     yStr=ds{3};
195     ddd=dayteNum-datenum([str2num(ds{3}),1,1])+1;
196     dStr=zeroPadStr(ddd,3);
197
198 end
199
200
201 function meta = populateTSmeta(yStr,dStr,hStr,durn,siteID,ch,SR
       ,network,NPY)
202 %This function should be generalized to a class metadata object
203 %which initializes with fields set to None, then populated
       based on kwargs
204     meta.yStr=yStr;
205     meta.dStr=dStr;
206     meta.hStr=hStr;
207     meta.durn=durn;
208     meta.site=siteID;
209     meta.ch=ch;
210     meta.SR=SR;
211     meta.network=network;
212     meta.fName = genFilenameFromMeta(meta);
213     if NPY == 1
214         meta.ext = 'npy'
215     elseif NPY == 0
216         meta.ext = 'mat'
217     end
218     meta.fullName = [meta.fName,'.',meta.ext]
219 end
220
221 function [] = plotDownloadedData(y,meta)
```

```
%plot data recently downloaded
    if length(y.data)
        t=1:length(y.data);
        if length(y.data)==86400
            t=linspace(0,24,length(y.data));
            figure; plot(t,y.data);
            xlabel('Hours','fontsize',15);
        else
            figure; plot(t,y.data);
        end
    else
        warning('there do not seem to be any data downloaded
           for this channel')
        figure; plot(t,y.data);
    end
    w = ' ';
    ttl=[meta.site,w,meta.ch,w,meta.yStr,w,meta.dStr,w,meta.
       hStr,w,meta.durn];
    ylabel('Machine Counts')
    title(ttl);
    pause(3)
end


function fName = genFilenameFromMeta(meta)
% generate file handle from metadata
%the same handle will be used for both .mat and .npy files
%
    global DATApath verbose
    if verbose
        display(['GET ',meta.yStr,' ',meta.dStr,' ',meta.hStr,'
            ',meta.durn, ' ',meta.site,' ',meta.SR,meta.ch])
    end
```

```
252     fName = fullfile(DATApath,[meta.SR,meta.yStr],'TS',[meta.
           site,'_',meta.SR,meta.ch,'_',meta.dStr])
253     if regexp(meta.SR,'B')
254         fName=[fName,'_',meta.hStr];
255     end
256 end
257
258 function GUIstate = readGetDataGUI(handles)
259 %Method to read the fields of the GetNCEDC GUI
260 %and store in a struct
261     global ARRAYS
262     GUIstate.SR = getSR(get(handles.SRpopup,'Value'));
263     arrayNameIndex=get(handles.ArraySelPopup,'Value');
264     GUIstate.RA=ARRAYS{arrayNameIndex};
265     GUIstate.YYYYstart=str2num(get(handles.YYstartedit,'String'
           ));
266     GUIstate.YYYYend=str2num(get(handles.YYendedit,'String'));
267     GUIstate.MMstart=str2num(get(handles.MMstartedit,'String'))
           ;
268     GUIstate.MMend=str2num(get(handles.MMendedit,'String'));
269     GUIstate.DDstart=str2num(get(handles.DDstartedit,'String'))
           ;
270     GUIstate.DDend=str2num(get(handles.DDendedit,'String'));
271     GUIstate.dayNumStart=datenum(GUIstate.YYYYstart,GUIstate.
           MMstart,GUIstate.DDstart);
272     GUIstate.dayNumEnd=datenum(GUIstate.YYYYend,GUIstate.MMend,
           GUIstate.DDend);
273     GUIstate.plotAsYouDownload=get(handles.PLOTcb,'Value');
274     GUIstate.OVERWRITEdata=get(handles.OVERWRITEcb,'Value');
275     GUIstate.NPY=get(handles.NPYcb,'Value');
276     GUIstate.SITECH=GUIstate.RA.chrArray;
277     if GUIstate.SR == 'B'
278         GUIstate.HHstart=str2num(get(handles.HRstartedit,'
              String'));
```

```
279         GUIstate.HHend=str2num(get(handles.HRendedit,'String'))
280         %STT=get(handles.STTcb,'Value');
281     else
282         GUIstate.HHstart=0;
283         GUIstate.HHend=00;
284     end
285
286     GUIstate.HHlist=GUIstate.HHstart;
287     %in case its 1Hz or 0.1Hz, just call the data in 1-day
          chunks by datestr
288     %in case its 40Hz, check is specified sametimetomorrow
289     if GUIstate.SR == 'B'
290         GUIstate.durn='2H';
291         GUIstate.epsln=0.01;
292         GUIstate.HHlist=GUIstate.HHstart:2:GUIstate.HHend-
              GUIstate.epsln;
293         %add note in documentation, for multiple day download,
              start at
294         %00:00
295         GUIstate.nSections=length(GUIstate.HHlist);
296     else
297         GUIstate.durn='1d';
298     end
299
300 end
301
302
303
304 function metaDataCells = generateMetaDataCells(GUIstate)
305 %Create a list of each instance of a data download,
306 %Store the list a cellArray of metaData objects
307     metaDataCells = {};
308     iMetaCell = 1;
309     for dayte=GUIstate.dayNumStart:GUIstate.dayNumEnd
```

```
310         [yStr, dStr] = StringsFromDatenum(dayte);
311         for sc=1:numel(GUIstate.SITECH)
312             siteID=GUIstate.SITECH{sc}.locn;
313             network=GUIstate.SITECH{sc}.net;
314             ch=GUIstate.SITECH{sc}.chn;
315             for HH=GUIstate.HHlist
316                 hStr=zeroPadStr(HH,2);
317                 meta = populateTSmeta(yStr,dStr,hStr,GUIstate.
                        durn,siteID,ch,GUIstate.SR,network,GUIstate.
                        NPY);
318                 metaDataCells{iMetaCell}=meta;
319                 iMetaCell = iMetaCell + 1;
320             end
321         end
322     end
323 end
```

```
function atBSL=amIatBSL()
%{
Checks the name of local host vs a list of BSL machines I am sometimes
logged in on, and returns a boolean (0/1), true if at BSL, false otherwise
%}

BSLHOSTLIST={'crust','seismo54.geo.berkeley.edu','seismo56.geo.berkeley.edu'};

unixCmd='hostname';
[s,w]=unix(unixCmd);
hostName=w(1:end-1);
atBSL=0;
for i=1:length(BSLHOSTLIST)
    if strmatch(hostName,BSLHOSTLIST{i},'exact')
        atBSL=1;
        display(['Detected that I am at BSL, hostname: ', hostName]);
    end
end
if atBSL==0
    display('Not at BSL: replacing cp commands scp - make sure public-private keys are setup')
else
end
```

```
function [y]=getAscii(X)
%{
A Method to retrieve ASCII data from NCEDC
@inputs:
X - an object of metaData class
@FLOW
-check whether you are at BSL - if so this can be made to run much faster
with internal networking.

@changelog
%10Dec2009 Staging Clean Job Modified becasue of ssh wildcard issues
%remoteCmd=['ssh ',me,' rm -f ',ascStageDir,'*'];

@note Sep 11, 2012, Modifying to accomodate .npy files.
or first iteration, can just convert ascii to npy using a python script
locally, but we will want to convert from ascii to python at BSL later so
that data transfer rate is higher.

%}

global ULFEMpath ulfempath SCRpath scrpath me binStageDir ascStageDir ...
    ulfemToolbox PYTHONpath
atBSL=amIatBSL();

%specially handle sites with .10 .20 channel specifiers
tenTwentySites={'BRIB','MHDL','JRSC'};  %ULFEMenvironment?

%set default values
```

```
%@note: this should be made into an object of class recording,
   and handle
%units
%y.flag=''; Decided field will only exist if there is a problem
y.data=nan;

N=idNumExpectedPts(X.durn,X.SR);

cleanBSL_StagingDirectories()  %clean out staging directories
   at BSL
cleanScratch();  %clean local dirs


display(['Queing Data at BSL'])
%@note: This should be a function: localCmd = genQdataCmd(X,
   binStageDir)
%and then wrapped as an ssh with another command wrapAsSSH(user
   , localCmd)
%possibly including submethods like
remoteCmd=['ssh ',me,' qdata -f ',X.yStr,'.',X.dStr,'.',X.hStr,
   ':00 -o ',binStageDir,X.site,'.',X.network,...
    '.',X.SR,X.ch,'.',X.yStr,'_',X.dStr,'_',X.hStr,'.bin -s ',X
       .durn,' ',X.site,'.',X.network,'.',X.SR,X.ch]

for siteHandle=tenTwentySites
    if regexp(X.site,siteHandle{1})
        if regexp(X.ch,'Q')
            app='.10';
        elseif regexp(X.ch,'T')
            app='.20';
        end
        remoteCmd=[remoteCmd,app];
    end
end
```

```
[s w]=unix(remoteCmd);
if s~=0
    mssg=['Error in Queing Data'];
    warning(mssg);
end

display(['Getting list of que-ed data file(s)'])
remoteCmd=['ssh ',me,' ls ',binStageDir];
[s w]=unix(remoteCmd);
if length(w)==0
    warning(['Looks like there were no binaries cued up, check
       \n 1) the site-net-channel combo \n 2) That site was
       reporting at the specified time','],...
        '');
    y.flag='No Binary Data from Que';
    return
end
fileList=ksplit(w,'\n');

%remove lines which have to do with ssh garbage text, BSL
   problem)
fileListFlagLines=[];
FlagLines={'ssh_keysign:', 'key_sign failed', 'Warning: No
   xauth data; using fake'};
for i=1:numel(fileList)
    for j=1:numel(FlagLines)
        if strmatch(FlagLines{j},fileList{i})
            fileListFlagLines=[fileListFlagLines i];
        end
    end
end

fileList(fileListFlagLines)=[];
display(['Converting File(s) to Ascii'])
```

```
for file=1:numel(fileList)
    remoteCmd=['ssh ',me,' quan2asc  -o ',ascStageDir,' ',
       binStageDir,fileList{file}];
    [s w]=unix(remoteCmd);
    if s~=0
        mssg=['Error in Converting Binary Files to ASCII'];
        warning(mssg);
    end
end

display(['Getting list of ascii file(s) at BSL:'])
[s w]=unix(['ssh ',me,' ls ' ascStageDir]);
if s~=0
    mssg=['Error getting list of cued ASCII files at BSL'];
    warning(mssg);
    s
    w
end

%generate matlab TS
%case 1: no data
if length(w)==0
   mssg=['Error Queing Data - No Data:',X.yStr,'.',X.dStr,'.',X
      .hStr,':00  ',X.site,'.',X.SR,X.ch,' is empty'];
   warning(mssg);
   if regexp('1d',X.durn) & regexp('L',X.SR)
        y.flag='No Ascii Data... but';
        return
   end
else
    %Case 2:there is data
    fileList=ksplit(w,'\n');
    fileListFlagLines=[];
    for i=1:numel(fileList)
```

```
119         for j=1:numel(FlagLines)
120             if strmatch(FlagLines{j},fileList{i})
121                 fileListFlagLines=[fileListFlagLines i];
122             end
123         end
124     end
125     fileList(fileListFlagLines)=[];
126     fileListText=[];
127     for iFl=1:length(fileList)
128         fileListText=[fileListText,' ',fileList{iFl}];
129     end
130     display(['FileList: ', fileListText])
131    %<transfer the file(s)>
132     display(['Transferring data from BSL to Local Machine'])
133     for iFile=1:length(fileList)
134         remoteCmd=['scp ',me,':',ascStageDir,fileList{iFile},'
              ',SCRpath];
135         if atBSL
136             remoteCmd=['cp ',ascStageDir,'* ',ulfempath,'scr/'
                 ];
137         end
138         [s w]=unix(remoteCmd);
139         if s~=0
140             display(['Error scp-ing ASCII data from BSL!'])
141             display(sprintf('Failed Command: %s returned with
                  exit status %d',remoteCmd,s))
142             %display(remoteCmd)
143             %display('returned with exit status = 0'])
144         end
145     end
146     %<\transfer the file(s)>
147 
148     %first npy/mat modification can be done here; loadBroken
       handles mat
```

```
149     %right now, but can be modified to handle NPY based on X.
           ext
150     y.data=loadBroken(X,N);
151 end
152 end
153
154
155 function [] = cleanBSL_StagingDirectories()
156     %execute script cleanAsciiStage.py at BSL
157     global ulfemToolbox me
158     display(['Cleaning out Staging Directories at BSL'])
159     remoteCmd=['ssh ',me,' python ',ulfemToolbox,'
           cleanAsciiStage.py'];
160     [s w]=unix(remoteCmd);
161     if s~=0
162         display(['Error cleaning Ascii Statging Dir at BSL']);
163         s
164         w
165     end
166
167     remoteCmd=['ssh ',me,' python ',ulfemToolbox,'cleanBinStage
           .py'];
168     [s w]=unix(remoteCmd);
169     if s~=0
170         display(['Error cleaning Binary Statging Dir at BSL']);
171         s
172     end
173 end
```

```
function DATA=loadBroken(X,N)
%{
USAGE: DATA=loadBroken(X,N);
@type X: struct
@param X: metadata object
@type N: integer
@param N: number of points anticipated in file

%Loads data from BSL; the "broken" refers to the idea that the
   data may be
%coming in segments when there are interruptions in the
   datalogger

%Would be good to write an example time/channel which is an
%Example of Broken FileStream here for debugging purposes
%}
global SCRpath ULFEMpath PYTHONpath samplingRateContainerMap

fNamePrefix=[X.site,'.',X.SR,X.ch];
display(['Preparing to load datafile ',fNamePrefix]);
sr = samplingRateContainerMap(X.SR)
fileList = dir([fullfile(SCRpath,fNamePrefix),'*'])
DATA=nan(N,1); %instantiate a data vector of NaN

if X.ext == 'mat'
    for iFile=1:length(fileList)
        fName=fullfile(SCRpath,fileList(iFile).name);
        [PATHSTR,NAME,EXT] = fileparts(fName);
        HHMMSS = NAME(end-5:end);
        Hdiff=str2num(HHMMSS(1:2)) - str2num(X.hStr)
        %check to make sure HH returned is HH requested
        hrShift=(sr*3600*(Hdiff));
        tIndex=1+hrShift+(sr*60*str2num(HHMMSS(3:4)))+(sr*
           str2num(HHMMSS(5:6)));
```

```
32          clear temp;
33          temp = textread(fName,'','headerlines',1);
34          display(['filling in DATA vector from ',num2str(tIndex)
               ,' to ',...
35              num2str(tIndex+length(temp)-1),' with ',num2str(
                   length(temp)),' points'])
36          DATA(tIndex:tIndex+length(temp)-1)=temp;
37      end
38  elseif X.ext == 'npy'
39      %Initally only support the case where there is a lone file,
40      %and wait for a broken recording to handle multifile; kk
           9/28/12
41      ascfName = fullfile(SCRpath,fileList(1).name)
42      unixCmd = ['python ',fullfile(PYTHONpath,'asc2npy.py'),' ',
           ascfName,' ', X.fullName]
43      unix(unixCmd)
44  end
```

```
function metaData = readMeta(SITE,CH)
%{
%MAY 8, 2010 spreadsheet metadata reader

%Oct 30, 2010: Change so .csv compatible


%}

global METADATApath

SAVE=1;

metaData={};

%Identify the file to read in
fName=[SITE,'_',CH,'.txt'];
fName = fullfile(METADATApath,'SPREADSHEETS',fName);
textMeta={}; %data structure to hold the lines of text read in

%Read in the meta data.
if exist(fName,'file')
    fid = fopen(fName,'r');
    a=fgetl(fid);
    nEpochs=str2num(a);
    fields=fgetl(fid);
    for iEpoch=1:nEpochs
        textMeta=fgetl(fid);
        cellMeta{iEpoch}=strsplit('##',textMeta);
        for iField=1:numel(cellMeta{iEpoch})
            cellMeta{iEpoch}{iField}=ddewhite(cellMeta{iEpoch}{
               iField});
        end
        cellMeta{iEpoch};
```

```
    end
else
    warnMssg=['File ',fName,' DNE!  Check metadata SPREADSHEETS
       ']
    warning(warnMssg);
end

%Assume that the ORDER of the fields is as described here:
%ELECTRIC:
% EPOCH, YYYY, DDD, HH, MM, COUNTS/V, GAIN(dB), GAIN(V), LENGTH
    (m),
% AZIMUTH, LATITUDE, LONGITUDE, ELEVATION, DEPTH, DIP
%MAGNETIC:
% EPOCH, YYYY, DDD, HH, MM, COUNTS/V, COIL ID,
% AZIMUTH, LATITUDE, LONGITUDE, ELEVATION, DEPTH, DIP
ELECTRIC_META={'EPOCH','YEAR','DYR','HH','MM','COUNTS/V','
   GAIN_dB',...
    'GAIN_V','LENGTH_m','AZIMUTH', 'LATITUDE', 'LONGITUDE', ...
    'ELEVATION', 'DEPTH', 'DIP'};
MAGNETIC_META={'EPOCH','YEAR','DYR','HH','MM','COUNTS/V','COIL
   ID',...
    'AZIMUTH', 'LATITUDE', 'LONGITUDE','ELEVATION', 'DEPTH', '
       DIP'};
SEISMIC_META={'EPOCH','YEAR','DYR','HH','MM','COUNTS/V','INST
   ID',...
    'AZIMUTH', 'LATITUDE', 'LONGITUDE','ELEVATION', 'DEPTH', '
       DIP'};
%COnfirm that the fields in the file being read-in match the
   fields
%That I think they are.
fields=strsplit('##',fields);
display(['readMeta.m calling channel ',CH])
if regexp(CH,'Q')
    metaFields=ELECTRIC_META;
```

```
60      instType='Electric';
61  elseif regexp(CH,'T')
62      metaFields=MAGNETIC_META;
63      instType='Magnetic';
64  elseif regexp(CH,'H')
65      metaFields=SEISMIC_META;
66      instType='Seismic';
67  elseif regexp(CH,'P')
68      metaFields=SEISMIC_META;
69      instType='Seismic';
70  end
71  display(['Identified as ',instType])
72  possError=0;
73  for iField=1:numel(fields)
74      fields{iField}=ddewhite(fields{iField});
75      if regexp(fields{iField},metaFields{iField}) |  regexp(
           metaFields{iField},fields{iField})
76          %then we are OK
77      else
78          display(['Format of .txt file is unexpected for field #
               ',num2str(iField)])
79          display(['Field Name read-in from txt file is: ',fields
               {iField}])
80          display(['Which should be a string of length: ',num2str
               (length(fields{iField}))])
81          display(['Whereas the metaFields template indicates
               that field #',num2str(iField),' should be the string:
                ',metaFields{iField}])
82          display(['Which has length ',num2str(length(metaFields{
               iField}))]);
83          possError=1;
84          regexp(fields{iField},metaFields{iField})
85          regexp(metaFields{iField},fields{iField})
86      end
```

```
87  end
88  if possError
89      warnMssg=['METADATA FIELD ORDERING IS NOT AS EXPECTED.
           CHECK SETUP OF METADATA SPREADSHEETS'];
90      warning(warnMssg)
91  end
92  fclose(fid);
93
94  %Now Have Identified Field Names and Values.  Build these into
       a .mat
95  %datasturcture;
96  for iE=1:nEpochs
97      switch instType
98          case 'Electric'
99          metaData{iE}.iEpoch=cellMeta{iE}{1};
100         metaData{iE}.yyyy=cellMeta{iE}{2};
101         metaData{iE}.dyr=zeroPadStr(cellMeta{iE}{3},3);
102         metaData{iE}.hh=zeroPadStr(cellMeta{iE}{4},2);
103         metaData{iE}.mm=zeroPadStr(cellMeta{iE}{5},2);
104         metaData{iE}.cpv=cellMeta{iE}{6};
105         metaData{iE}.gainDB=cellMeta{iE}{7};
106         metaData{iE}.gainV=cellMeta{iE}{8};
107         metaData{iE}.length=cellMeta{iE}{9};
108         metaData{iE}.azim=cellMeta{iE}{10};
109         metaData{iE}.latitude=cellMeta{iE}{11};
110         metaData{iE}.longitude=cellMeta{iE}{12};
111         metaData{iE}.elevation=cellMeta{iE}{13};
112         metaData{iE}.depth=cellMeta{iE}{14};
113         metaData{iE}.dip=cellMeta{iE}{15};
114         case 'Magnetic'
115         metaData{iE}.iEpoch=cellMeta{iE}{1};
116         metaData{iE}.yyyy=cellMeta{iE}{2};
117         metaData{iE}.dyr=zeroPadStr(cellMeta{iE}{3},3);
118         metaData{iE}.hh=zeroPadStr(cellMeta{iE}{4},2);
```

```
        metaData{iE}.mm=zeroPadStr(cellMeta{iE}{5},2);
        metaData{iE}.cpv=cellMeta{iE}{6};
        metaData{iE}.coilID=cellMeta{iE}{7};
        metaData{iE}.azim=cellMeta{iE}{8};
        metaData{iE}.latitude=cellMeta{iE}{9};
        metaData{iE}.longitude=cellMeta{iE}{10};
        metaData{iE}.elevation=cellMeta{iE}{11};
        metaData{iE}.depth=cellMeta{iE}{12};
        metaData{iE}.dip=cellMeta{iE}{13};
        case 'Seismic'
        metaData{iE}.iEpoch=cellMeta{iE}{1};
        metaData{iE}.yyyy=cellMeta{iE}{2};
        metaData{iE}.dyr=zeroPadStr(cellMeta{iE}{3},3);
        metaData{iE}.hh=zeroPadStr(cellMeta{iE}{4},2);
        metaData{iE}.mm=zeroPadStr(cellMeta{iE}{5},2);
        metaData{iE}.cpv=cellMeta{iE}{6};
        metaData{iE}.InstID=cellMeta{iE}{7};
        metaData{iE}.azim=cellMeta{iE}{8};
        metaData{iE}.latitude=cellMeta{iE}{9};
        metaData{iE}.longitude=cellMeta{iE}{10};
        metaData{iE}.elevation=cellMeta{iE}{11};
        metaData{iE}.depth=cellMeta{iE}{12};
        metaData{iE}.dip=cellMeta{iE}{13};
    end
end

if SAVE
    saveFileName = fullfile(METADATApath,'SPREADSHEETS',[SITE,'
       _',CH])
    svCmd=['save ',saveFileName,' metaData'];
    eval(svCmd)
end
```

```
%for iE=1:nEpoch
%    L{iE}=strsplit('##',l{iE});
%    for iFld=1:numel(L{iE})
%        L{iE}{iFld}=ddewhite(L{iE}{iFld});
%    end
%    timeStr{iE}=[L{iE}{2} L{iE}{3} L{iE}{4} L{iE}{5}];
%end
```

```
%update matlab metaData
clear all
close all
setPaths;
load(fullfile(syspath,'SITES','ARRAYS.mat'));
ARRAYS
if regexp(ARRAYS{1}.name,'ALL')
    arrayList=ARRAYS{1}.chrArray;
    for iRA =1:numel(arrayList)
        SITE=arrayList{iRA}.locn;
        CH=arrayList{iRA}.chn;
        display(['Getting SITE: ',SITE,' CH:', CH])
        M=readMeta(SITE,CH);
    end
else
    warnMssg=['First Array in ARRAYS.mat does not appear to be
       the array of ALL sensors.  It should be.']'
end
```

```
% calculates # of columns/rows for channel buttons on
% control panel of time series figure
function [ncol,nrow,staN]=NcolNrow(sta,ch_id);
[nchan,dum]=size(sta);
ncol=1;
staN=sta(1,:);
k1=1;
while k1<nchan,
for k2=k1+1:nchan
  s=(sta(k1,:)~=sta(k2,:));
  if sum(s)>0,
   ncol=ncol+1;
   staN=[staN;sta(k2,:)];
   k1=k2+1;
   break
  end
  k1=k2;
end
end
%%%%%%%%%%
[nsta,dum]=size(staN);
nrow=zeros(1,nsta);
for k1=1:nsta
 for k2=1:nchan
  if sta(k2,:)==staN(k1,:),nrow(k1)=nrow(k1)+1;end
 end
end
return
```

```
function [varargout]=PlotManager(hObject, eventdata)
%GUI for selecting data to plot as timeSeries.
%There are two main frames in the GUI.
%Frame #1 is used to select segments of data to load.  The interface in
%frame 1 shares much in common with the getData GUI, and should be
%merged with that code to avoid reproduction of work, and to ease
%maintainance.
%Frame #2 executes the call to plot the data, based on the user selected
%loaded data segment.
%
%
display(' Plot Manager -- Entry')
global ULFEMpath DATApath ARRAYS rect0 arrayList syspath
ULFEM_environment;%gen SRSTRINGS,arrayList, ARRAYS
handles.GET_GUI_POSN=[20,550,600,200];
getPlotDataGui=figure( 'MenuBar','none','Name', 'Get Plot Data',...
    'NumberTitle','off','Position',handles.GET_GUI_POSN,'units',...
    'normalized','tag','GETPLOTTSGUI', 'Color', [0.1 0.1 0.5]);
handles.Parent=getPlotDataGui;
guidata(handles.Parent,handles);

nFrames=2;
widFrame1=0.5;%percentage of the whole gui spent on frame 1;
handles.framePosns(1,:)=[0 0 widFrame1 1];
handles.framePosns(2,:)=[widFrame1 0 1-widFrame1 1];
for f=1:nFrames
    frmCols=[0 5]/10;
    handles.Frame{f}=uipanel('Parent',handles.Parent,...
```

```
        'units','normalized','Position',handles.framePosns(f,:)
          ,...
        'BackgroundColor',[1 1 1]*frmCols(f),'tag',['frame',
          num2str(f)]);
end

homeFrame=1;
wd1=0.31*handles.framePosns(homeFrame,3);
%wd2=0.66*wd1; %controls YYMMDDHH BOXES
wd2=wd1;
ht1=0.1;
epsln=0.01*handles.framePosns(homeFrame,3);
xx=0.057*handles.framePosns(homeFrame,3);%x,y shifts for date
   selction text and edit boxes
yy=-0.15*handles.framePosns(homeFrame,3);%x,y shifts for date
   selction text and edit boxes
textBoxColour = [0.5 0.5 1];
SRtxtpsn=[0 1-ht1 wd1 ht1];
SRpoppsn=[0 1-2*ht1 wd1 ht1];
arraySeltxtpsn=[wd1+epsln 1-ht1 1.3*wd1 ht1];
arraySelpoppsn=[wd1+epsln 1-2*ht1 1.3*wd1 ht1];
starttxtpsn=[xx+wd2+epsln yy+1-3*ht1+epsln wd2 0.9*ht1];
YYtxtpsn=[xx+0 yy+1-4*ht1 wd2 0.9*ht1];
YYstarteditpsn=[xx+wd2+epsln yy+1-4*ht1 wd2 ht1];
MMDDtxtpsn=[xx+0 yy+1-(5)*ht1 wd2 0.9*ht1];
MMstarteditpsn=[xx+wd2+epsln yy+1-5*ht1 wd2/2 ht1];
DDstarteditpsn=[xx+1.5*wd2+epsln yy+1-5*ht1 wd2/2 ht1];
HRtxtpsn=[xx+0 yy+1-(6)*ht1 wd2 0.9*ht1];
HRstarteditpsn=[xx+wd2+epsln yy+1-6*ht1 wd2 ht1];
nSegeditpsn=[xx+1.0*wd1 yy+1-7.3*ht1 wd2 ht1];
nSegtxtpsn=[xx+0.0*wd1 yy+1-7.3*ht1 wd1 ht1];
SITECHpbpsn=[wd1/2 1-8*ht1 1.5*wd1 ht1];
LOADDATApbpsn=[xx+0.5*wd1 1-9.7*ht1 wd1 ht1];

```

```
homeFrame=2;
wd2=0.20*handles.framePosns(homeFrame,3);
edgeWidth=0.05*handles.framePosns(homeFrame,3);
ht1=0.1;
epsln=0.01*handles.framePosns(homeFrame,3);
xx=0.1*handles.framePosns(homeFrame,3);;%x and y shifts for
   date seelction text and edit boxes
yy=-0.15*handles.framePosns(homeFrame,3);;%x and y shifts for
   date seelction text and edit boxes
LOADEDMVTSlistBoxpsn=[handles.framePosns(homeFrame,1)+edgeWidth
    edgeWidth 3*wd2, 0.85];
LOADEDMVTStxtpsn=[handles.framePosns(homeFrame,1)+edgeWidth
   1-5*edgeWidth 3*wd2, ht1];
PLOTTSpbpsn=[handles.framePosns(homeFrame,1)+2*edgeWidth+3*wd2
   1-ht1-5*edgeWidth wd2, ht1];

YYYY='YYYY';
dv=datevec(date);
YYYY=zeroPadStr(dv(1),4);

handles.SRtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text',...
    'String','Sampling Rate','units','normalized','Position',
       SRtxtpsn, ...
    'BackgroundColor', textBoxColour);
handles.SRpopup = uicontrol('Style', 'popupmenu','String',
   SRSTRINGS,...
    'Units','normalized','Position',SRpoppsn);
handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
   ,'text',...
    'String','SELECT ARRAY','units','normalized','Position',
       ...
    arraySeltxtpsn, 'BackgroundColor', textBoxColour);
```

```
82 handles.ArraySelPopup=uicontrol('Parent', handles.Parent, '
     Style', ...
83     'popupmenu','String',arrayList,'units','normalized',...
84     'Position',arraySelpoppsn);%,'Callback',{@array_choose,
         handles});
85 handles.starttxt=uicontrol('Parent', handles.Parent, 'Style', '
     text',...
86     'String','Start','units','normalized','Position',
         starttxtpsn, ...
87     'BackgroundColor', textBoxColour);
88 handles.YYtxt=uicontrol('Parent', handles.Parent, 'Style', '
     text',...
89     'String','YEAR','units','normalized','Position',YYtxtpsn
         ,...
90     'BackgroundColor', textBoxColour);
91 handles.YYstartedit=uicontrol('Parent', handles.Parent, 'Style'
     , 'edit',...
92     'String',YYYY,'units','normalized','Position',
         YYstarteditpsn, 'Tag',...
93     'YYeditTag');
94 handles.MMDDtxt=uicontrol('Parent', handles.Parent, 'Style', '
     text',...
95     'String','MM DD','units','normalized','Position',MMDDtxtpsn
         ,...
96     'BackgroundColor', textBoxColour);
97 handles.MMstartedit=uicontrol('Parent', handles.Parent, 'Style'
     , 'edit',...
98     'String','MM','units','normalized','Position',
         MMstarteditpsn);
99 handles.DDstartedit=uicontrol('Parent', handles.Parent, 'Style'
     , 'edit',...
100    'String','DD','units','normalized','Position',
         DDstarteditpsn);
```

```
handles.HRtxt=uicontrol('Parent', handles.Parent, 'Style', 'text',...
    'String','UT HH:MM','units','normalized','Position',HRtxtpsn,...
    'BackgroundColor', textBoxColour);
handles.HRstartedit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String','HH:MM','units','normalized','Position',HRstarteditpsn);
handles.nSegtxt=uicontrol('Parent', handles.Parent, 'Style', 'text',...
    'String','N Segments','units','normalized','Position',nSegtxtpsn,...
    'BackgroundColor', textBoxColour);
handles.nSegEdit=uicontrol('Parent', handles.Parent, 'Style', 'edit',...
    'String','nSeg?','units','normalized','Position',nSegeditpsn,...
    'tooltipstring',...
    'How many segments after the start time do you wish to load');
handles.LOADDATApb=uicontrol('Parent', handles.Parent, 'Style',...
    'pushbutton','String','LOAD DATA','units','normalized','Position',...
    LOADDATApbpsn,'callback','loadPlotCB_kk','tooltipstring',...
    'Loads Data for Plotting','Tag','Load');

handles.LOADEDMVTSlistbox=uicontrol('Parent', handles.Parent, 'Style', ...
    'listbox','units','normalized','Position',LOADEDMVTSlistBoxpsn,...
```

```
    'tooltipstring','LoadeD arrays','Tag','LoadedArraysListbox'
      );
handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
   , 'text',...
    'String','LOADED TS','units','normalized','Position',
      LOADEDMVTStxtpsn,...
    'BackgroundColor', textBoxColour);
handles.PLOTTSpb=uicontrol('Parent', handles.Parent, 'Style',
   ...
    'Pushbutton','String','PLOT TS','units','normalized','
      Position',...
    PLOTTSpbpsn, 'BackgroundColor', textBoxColour,'callback'
      ,...
    'loadPlotCB_kk','Tag','Plot');
set(handles.SRpopup, 'callback', {@setSRcbk,handles});
guidata(handles.Parent,handles);
end


function []=setSRcbk(hObject, eventdata, handles)
SR = getSR(get(handles.SRpopup,'Value'));
if SR=='L'
    set(handles.HRstartedit,'String','00:00');
    set(handles.HRstartedit,'style','text');
    %set(handles.HRendedit,'String','24:00');
    %set(handles.HRendedit,'style','text');
else
    set(handles.HRstartedit,'String','HH:MM');
    set(handles.HRstartedit,'style','edit')
    %set(handles.HRendedit,'String','HH:MM');
    %set(handles.HRendedit,'style','edit');
end
%get(handles.HRstartedit)
end
```

```
function [varargout]=ProcManager(varargin)
%GUI TO PROCESS TS files.  This function is the callback to the
    Proc TS button on the ULFEM GUI.  The Main task of this
   program is to convert timeseries to FC files, where FCs have
   units of mV/km/sqrt(Hz) and nT/sqrt(Hz)

%This version relies on a bFC- bands of Fourier coefficients -
   setup file


global ULFEMpath DATApath ARRAYS SLaSH syspath
%Get metadata needed to setup GUI about Arrays and Sampling
   Rates
load([syspath,'sr_list']);
SRSTRINGS{1}=' ';
for i=1:length(sr_list.letters)
    SRSTRINGS{i+1}=[num2str(sr_list.numbers(i)),' Hz ',sr_list.
       letters(i)];
end
[arrayList ARRAYS]=readArrays();
%PWSTRINGS:, AWSTRINGS
PWSTRINGS={'NONE','First Differece','ARMA'};
AWSTRINGS={'BOXCAR','HAMMING','HANN'};
%Initiate the GUI and specify its geometery

handles.GUI_POSN=[20,550,250,530];
handles.procTS2FCGUI=figure('MenuBar','none','Name', 'Process
   Time Series to Fourier Coefficients', 'NumberTitle','off',...
                'Position',handles.GUI_POSN,'units','
                   normalized',...
                'tag','TS2FCGUI', 'Color', [0.1 0.1 0.5]);

%embed buttons in a frame for portability
handles.framePosns(1,:)=[0 0.9 1 0.1];
```

```
handles.framePosns(2,:)=[0 0.6 1 0.3];
handles.framePosns(3,:)=[0 0.4 1 0.2];
handles.framePosns(4,:)=[0 0.2 1 0.2];
frmCols=[0 2 4 6 8]/10;
for f=1:size(handles.framePosns,1)
handles.Frame{f}=uipanel( ...
        'Parent',handles.procTS2FCGUI,...
        'units','normalized',...
        'Position',handles.framePosns(f,:),...
        'BackgroundColor',[1 1 frmCols(f)],...
        'tag',['frame',num2str(f)]);
end

wd1=0.40;
wd2=0.66*wd1;
ht1=0.1;
tbh=0.05;%text box height
epsln=0.01;
xx=0.091;%x and y shifts for date seelction text and edit boxes
yy=-0.15;%x and y shifts for date seelction text and edit boxes
scl=0.66;
textBoxColour = [0.5 0.5 1];
%%%FRAMEWISE FRAME1
nWidgets{1}=[2 2]; %number of buttons of equal size
%when it is 2x2, use 9% pad top and bottom, 40% for widgets and
    1% epsilon
pad=0.09;
widgetH=0.4;
wE=0.01;%widgetEpsilon
SRtxtpsn=[0 1-pad-widgetH wd1 widgetH];
SRpoppsn=[0 1-widgetH-SRtxtpsn(4)-wE wd1 widgetH];
arraySeltxtpsn=[wd1+wE 1-widgetH-pad 1.3*wd1 widgetH];
arraySelpoppsn=[wd1+wE 1-widgetH-arraySeltxtpsn(4)-wE 1.3*wd1
   widgetH];
```

```
%%FRAME2 NOT QUITE FRAMEWISE YET
starttxtpsn=[xx+wd2+epsln yy+1-2*ht1+epsln wd2 0.9*ht1]; %row1
endtxtpsn=[xx+2*wd2+2*epsln yy+1-2*ht1+epsln wd2 0.9*ht1];%row1
YYtxtpsn=[xx+0 yy+1-3*ht1 wd2 0.9*ht1];%row2
YYstarteditpsn=[xx+wd2+epsln yy+1-3*ht1 wd2 ht1];%row2
YYendeditpsn=[xx+2*wd2+2*epsln yy+1-3*ht1 wd2 ht1];%row2
MMDDtxtpsn=[xx+0 yy+1-(4*ht1) wd2 0.9*ht1];%row3
MMstarteditpsn=[xx+wd2+epsln yy+1-4*ht1 wd2/2 ht1];
MMendeditpsn=[xx+2*wd2+2*epsln yy+1-4*ht1 wd2/2 ht1];
DDstarteditpsn=[xx+1.5*wd2+epsln yy+1-4*ht1 wd2/2 ht1];
DDendeditpsn=[xx+2.5*wd2+2*epsln yy+1-4*ht1 wd2/2 ht1];%row3
HRtxtpsn=[xx+0 yy+1-(5)*ht1 wd2 0.9*ht1];%row4
HRstarteditpsn=[xx+wd2+epsln yy+1-5*ht1 wd2 ht1];
HRendeditpsn=[xx+2*wd2+2*epsln yy+1-5*ht1 wd2 ht1];%row4
%%%FRAME3 Go FRAMEWISE
widgetH=0.25;pad=0.05;wE=0.01;
PWtxtpsn=[0 1-pad-widgetH wd1 widgetH];
PWpoppsn=[0 1-widgetH-PWtxtpsn(4)-wE wd1 widgetH];
ApodWndwtxtpsn=[wd1+wE 1-widgetH-pad 1.3*wd1 widgetH];
ApodWndwpoppsn=[wd1+wE 1-widgetH-ApodWndwtxtpsn(4)-wE 1.3*wd1
   widgetH];
widgetH=0.15;
WndwWdthtxtpsn=[0 1-0.6-widgetH wd1 widgetH];
WndwWdtheditpsn=[0 WndwWdthtxtpsn(2)-widgetH-pad wd1 widgetH];
Overlaptxtpsn=[wd1+wE 1-0.6-widgetH wd1 widgetH];
Overlapeditpsn=[wd1+wE WndwWdthtxtpsn(2)-widgetH-pad wd1
   widgetH];
%%%FRAME4 Go FRAMEWISE
pad=0.15;cbh=0.20;%chekboxheight
pbh=0.35;
STTcbpsn=[0.15 1-pad-cbh wd1/2 cbh];
OVERWRITEcbpsn=[0.15+STTcbpsn(3)+pad 1-pad-cbh  wd1 cbh];
GETDATApbpsn=[0.25 pad 0.5 pbh];

```

```
91
92  [YYYY MM DD]=yesterday;
93
94  handles.Parent=handles.Frame{1};
95
96  %make the buttons and boxes
97  handles.SRtxt=uicontrol('Parent', handles.Parent, 'Style', '
       text','String','Sampling Rate','units',...
98      'normalized','Position',SRtxtpsn, 'BackgroundColor',
           textBoxColour,'FontName', 'FixedWidth');
99  handles.SRpopup = uicontrol('Parent',handles.Parent,'Style', '
       popupmenu','String', SRSTRINGS,...
100                     'Units','normalized','Position',SRpoppsn);
101 handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
       , 'text','String','SELECT ARRAY','units',...
102     'normalized','Position',arraySeltxtpsn, 'BackgroundColor',
           textBoxColour,'FontName', 'FixedWidth');
103 handles.ArraySelPopup=uicontrol('Parent', handles.Parent, '
       Style', 'popupmenu','String',arrayList,'units',...
104     'normalized','Position',arraySelpoppsn);
105 %%
106 handles.Parent=handles.Frame{2};
107 handles.starttxt=uicontrol('Parent', handles.Parent, 'Style', '
       text','String','Start','units',...
108     'normalized','Position',starttxtpsn, 'BackgroundColor',
           textBoxColour,'FontName', 'FixedWidth');
109 handles.endtxt=uicontrol('Parent', handles.Parent, 'Style', '
       text','String','End','units',...
110     'normalized','Position',endtxtpsn, 'BackgroundColor',
           textBoxColour,'FontName', 'FixedWidth');
111 handles.YYtxt=uicontrol('Parent', handles.Parent, 'Style', '
       text','String','YEAR','units',...
112     'normalized','Position',YYtxtpsn,'BackgroundColor',
           textBoxColour,'FontName', 'FixedWidth');
```

```
handles.YYstartedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit','String',YYYY,'units',...
    'normalized','Position',YYstarteditpsn, 'Tag','YYeditTag');
handles.YYendedit=uicontrol('Parent', handles.Parent, 'Style',
   'edit','String',YYYY,'units',...
    'normalized','Position',YYendeditpsn, 'Tag','YYeditTag');
handles.MMDDtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text','String','MM DD','units',...
    'normalized','Position',MMDDtxtpsn,'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.MMstartedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit','String',MM,'units',...
    'normalized','Position',MMstarteditpsn);
handles.DDstartedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit','String',DD,'units',...
    'normalized','Position',DDstarteditpsn);
handles.MMendedit=uicontrol('Parent', handles.Parent, 'Style',
   'edit','String',MM,'units',...
    'normalized','Position',MMendeditpsn);
handles.DDendedit=uicontrol('Parent', handles.Parent, 'Style',
   'edit','String',DD,'units',...
    'normalized','Position',DDendeditpsn);
handles.HRtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text','String','UT HH:MM','units',...
    'normalized','Position',HRtxtpsn,'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.HRstartedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit','String','HH:MM','units',...
    'normalized','Position',HRstarteditpsn);
handles.HRendedit=uicontrol('Parent', handles.Parent, 'Style',
   'edit','String','HH:MM','units',...
    'normalized','Position',HRendeditpsn);
%%
handles.Parent=handles.Frame{3};
```

```
handles.PWtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text','String','PREWHITENING','units',...
    'normalized','Position',PWtxtpsn, 'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.PWpopup = uicontrol('Parent',handles.Parent,'Style', '
   popupmenu','String', PWSTRINGS,...
                    'Units','normalized','Position',PWpoppsn);
handles.ApodWndwtxt=uicontrol('Parent', handles.Parent, 'Style'
   , 'text','String','APODIZATION','units',...
    'normalized','Position',ApodWndwtxtpsn, 'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.AWpopup = uicontrol('Parent',handles.Parent,'Style', '
   popupmenu','String', AWSTRINGS,...
                    'Units','normalized','Position',
                       ApodWndwpoppsn);
handles.WndwWdthtxt=uicontrol('Parent', handles.Parent, 'Style'
   , 'text','String','Window Width','units',...
    'normalized','Position',WndwWdthtxtpsn,'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.WndwWdthedit=uicontrol('Parent', handles.Parent, 'Style
   ', 'edit','String',256,'units',...
    'normalized','Position',WndwWdtheditpsn, 'Tag','WWeditTag')
       ;
handles.Overlaptxt=uicontrol('Parent', handles.Parent, 'Style',
    'text','String','Overlap','units',...
    'normalized','Position',Overlaptxtpsn,'BackgroundColor',
       textBoxColour,'FontName', 'FixedWidth');
handles.Overlapedit=uicontrol('Parent', handles.Parent, 'Style'
   , 'edit','String',64,'units',...
    'normalized','Position',Overlapeditpsn, 'Tag','OvlpeditTag'
       );
handles.Parent=handles.Frame{4};
handles.STTcb=uicontrol('Parent', handles.Parent, 'Style', '
   Checkbox','String','STT','units',...
```

```
    'normalized','Position',STTcbpsn);
handles.OVERWRITEcb=uicontrol('Parent', handles.Parent, 'Style'
   , 'Checkbox','String','OVERWRITE','units',...
    'normalized','Position',OVERWRITEcbpsn);
handles.PROCDATApb=uicontrol('Parent', handles.Parent, 'Style',
    'pushbutton','String','PROC DATA','units',...
    'normalized','Position',GETDATApbpsn,'callback',{
       @PROCDATAFCN,handles}, 'tooltipstring','Process TS to FC'
       );
set(handles.SRpopup, 'callback', {@setSRcbk,handles});
end


function [] = PROCDATAFCN(hObject, eventdata, handles)
    %Convert TS to FC.
    global ULFEMpath DATApath ARRAYS SLaSH syspath
    SLaSH=loadSLASH;

    %Collect Parameters SR,start time, end time,
    SR = getSR(get(handles.SRpopup,'Value')); %sampling rate
    load([syspath,'bFC_',SR]);
    load([syspath,'T_',SR]);
    arrayNameIndex=get(handles.ArraySelPopup,'Value');
    RA=ARRAYS{arrayNameIndex}; %Array Name
    YYYYstart=str2num(get(handles.YYstartedit,'String'));
    YYYYend=str2num(get(handles.YYendedit,'String'));
    MMstart=str2num(get(handles.MMstartedit,'String'));
    MMend=str2num(get(handles.MMendedit,'String'));
    DDstart=str2num(get(handles.DDstartedit,'String'));
    DDend=str2num(get(handles.DDendedit,'String'));
    dayNumStart=datenum(YYYYstart,MMstart,DDstart);
    dayNumEnd=datenum(YYYYend,MMend,DDend);
    OVERWRITEdata=get(handles.OVERWRITEcb,'Value');
    if SR=='B'
```

```
        HHstart=str2num(get(handles.HRstartedit,'String'));
        HHend=str2num(get(handles.HRendedit,'String'));
        STT=get(handles.STTcb,'Value');
        dT=1/40;
    else
        HHstart=0;  HHend=00;
        dT=1;
    end
    clear SITECH
    SITECH=RA.chrArray;
    OUTVARS=['Y0=',num2str(YYYYstart),'; Y1=',num2str(YYYYend),
       '; M0=',num2str(MMstart),'; M1=',num2str(MMend)...
    ,'; D0=',num2str(DDstart),'; D1=',num2str(DDend),'; H0=',
       num2str(HHstart),'; H1=',num2str(HHend),'; SR=',SR];
    display(OUTVARS)

    prewhiteningMethodValue=get(handles.PWpopup,'Value');
    prewhiteningChoices=get(handles.PWpopup,'String');
    prewhiteningMethod=prewhiteningChoices{
       prewhiteningMethodValue};

    apodWndwValue=get(handles.AWpopup,'Value');
    ApodWndwChoices=get(handles.AWpopup,'String');
    ApodizationWndw=ApodWndwChoices{apodWndwValue};

    Nwndw=str2num(get(handles.WndwWdthedit,'String'));
    Lwndw=str2num(get(handles.Overlapedit,'String'));
    nDL=numel(bFC);

    HHlist=HHstart;
    if SR=='B'
        durn='2H';
        if STT==0
            epsln=0.01;
```

```
            HHlist=HHstart:2:HHend-epsln
        end
        nSections=length(HHlist);
    else
        durn='1d';
    end
   %Loop Over individual time windows
    for dayte=dayNumStart:dayNumEnd
        ds=split('-',datestr(dayte));
        yStr=ds{3};
        ddd=dayte-datenum([str2num(ds{3}),1,1])+1;
        dStr=zeroPadStr(ddd,3);
        for sc=1:numel(SITECH)
            siteID=SITECH{sc}.locn;
            network=SITECH{sc}.net;
            ch=SITECH{sc}.chn;
            for HH=HHlist
                hStr=zeroPadStr(HH,2);
                matfName=[DATApath,SR,yStr,SLaSH,'TS',SLaSH,
                   siteID,'_',SR,ch,'_',dStr];
                meta.yStr=yStr;
                meta.dStr=dStr;
                meta.hStr=hStr;
                meta.durn=durn;
                meta.site=siteID;
                meta.ch=ch;
                meta.SR=SR;
                meta.network=network;
                if regexp(meta.ch,'Q')
                    fieldType='ELECTRIC';
                elseif regexp(meta.ch,'T')
                    fieldType='MAGNETIC';
                end

```

```
247             if regexp(SR,'B')
248                 matfName=[matfName,'_',hStr];
249             end
250             if exist([matfName,'.mat'],'file')
251                 clear y
252                 load(matfName);
253                 if isfield(y,'flag')
254                     warnmssg=['Requested Time Series File:
                           ',matfName,' is Flagged: ',y.flag];
255                     warning(warnmssg)
256                     %Proceed to a case/switch if types of
                           flags become
257                     %numerous/complex, for now just skip
                           flagged files
258                 else
259                     display(['File:',matfName,' loaded'])
260                     display(['Identify appropriate EPOCH'])
261                     %iDedEpoch=associateEpochNumber(yStr,
                           dStr,siteID,[SR,ch]);
262                     epochMeta=associateEpochNumber2(yStr,
                           dStr,siteID,ch); %modernized 01 May
                           2010 for use with spreadsheets;
263                     FC= TS2CFC(y.data, epochMeta,ch,
                           prewhiteningMethod, ApodizationWndw,
                           Nwndw,Lwndw,nDL, dT, fieldType,bFC);
264                 end
265             else
266                 warnmssg=['Requested Time Series File: ',
                       matfName,' does not exist.  Try
                       downloading it.'];
267                 warning(warnmssg)
268             end
269             %save the FC file in this loop
```

```
                FCmatfName=[DATApath,SR,yStr,SLaSH,'FC',SLaSH,
                   siteID,'_',SR,ch,'_',dStr];
                if regexp(SR,'B')
                    FCmatfName=[FCmatfName,'_',hStr];
                end
                svCmd=['save ',FCmatfName,' FC'];
                eval(svCmd)
            end
        end
    end
end


function []=setSRcbk(hObject, eventdata, handles)
SR = getSR(get(handles.SRpopup,'Value'));
if SR=='L' | SR=='V'
    set(handles.HRstartedit,'String','00:00');
    set(handles.HRstartedit,'style','text');
    set(handles.HRendedit,'String','24:00');
    set(handles.HRendedit,'style','text');
else
    set(handles.HRstartedit,'String','HH:MM');
    set(handles.HRstartedit,'style','edit')
    set(handles.HRendedit,'String','HH:MM');
    set(handles.HRendedit,'style','edit');
end
%get(handles.HRstartedit)
end
```

```
function [varargout]=PlotManager(hObject, eventdata)
%{ GUI for user to choose data time series of Fourier
   Coefficients for
%Spectrogram Plotting.
%}Based on Time series plotter.  Plots One channel at a time
   though.

global ULFEMpath DATApath ARRAYS rect0 arrayList syspath
load([syspath,'sr_list'])
SRSTRINGS{1}='ALL';
for i=1:length(sr_list.letters)
    SRSTRINGS{i+1}=[num2str(sr_list.numbers(i)),' Hz ',sr_list.
       letters(i)];
end
[arrayList ARRAYS]=readArrays();
handles.GET_GUI_POSN=[20,550,800,200];
getPlotDataGui=figure( 'MenuBar','none','Name', 'Spectrogram
   Manager',...
 'NumberTitle','off','Position',handles.GET_GUI_POSN,'units','
    normalized',...
 'tag','SPCGMMGR', 'Color', [0.1 0.1 0.5]);
handles.Parent=getPlotDataGui;
guidata(handles.Parent,handles);

nFrames=2;
widFrame1=0.5;%percentage of the whole gui spent on frame 1;
handles.framePosns(1,:)=[0 0 widFrame1 1];
handles.framePosns(2,:)=[widFrame1 0 1-widFrame1 1];
for f=1:nFrames
    frmCols=[0 5]/10;
    handles.Frame{f}=uipanel('Parent',handles.Parent,...
        'units','normalized','Position',handles.framePosns(f,:)
           ,...
```

```
28          'BackgroundColor',[1 1 1]*frmCols(f),'tag',['frame',
              num2str(f)]);
29  end
30
31  homeFrame=1;
32  wd1=0.31*handles.framePosns(homeFrame,3);
33  %wd2=0.66*wd1; %controls YYMMDDHH BOXES
34  wd2=wd1;
35  ht1=0.1;
36  epsln=0.01*handles.framePosns(homeFrame,3);
37  xx=0.057*handles.framePosns(homeFrame,3);;%x and y shifts for
       date seelction text and edit boxes
38  yy=-0.15*handles.framePosns(homeFrame,3);;%x and y shifts for
       date seelction text and edit boxes
39  textBoxColour = [0.5 0.5 1];
40  SRtxtpsn=[0 1-ht1 wd1 ht1];
41  SRpoppsn=[0 1-2*ht1 wd1 ht1];
42  arraySeltxtpsn=[wd1+epsln 1-ht1 1.3*wd1 ht1];
43  arraySelpoppsn=[wd1+epsln 1-2*ht1 1.3*wd1 ht1];
44  starttxtpsn=[xx+wd2+epsln yy+1-3*ht1+epsln wd2 0.9*ht1];
45  YYtxtpsn=[xx+0 yy+1-4*ht1 wd2 0.9*ht1];
46  YYstarteditpsn=[xx+wd2+epsln yy+1-4*ht1 wd2 ht1];
47  MMDDtxtpsn=[xx+0 yy+1-(5)*ht1 wd2 0.9*ht1];
48  MMstarteditpsn=[xx+wd2+epsln yy+1-5*ht1 wd2/2 ht1];
49  DDstarteditpsn=[xx+1.5*wd2+epsln yy+1-5*ht1 wd2/2 ht1];
50  HRtxtpsn=[xx+0 yy+1-(6)*ht1 wd2 0.9*ht1];
51  HRstarteditpsn=[xx+wd2+epsln yy+1-6*ht1 wd2 ht1];
52  nSegeditpsn=[xx+1.0*wd1 yy+1-7.3*ht1 wd2 ht1];
53  nSegtxtpsn=[xx+0.0*wd1 yy+1-7.3*ht1 wd1 ht1];
54  SITECHpbpsn=[wd1/2 1-8*ht1 1.5*wd1 ht1];
55  LOADDATApbpsn=[xx+0.5*wd1 1-9.7*ht1 wd1 ht1];
56
57  homeFrame=2;
58  wd2=0.20*handles.framePosns(homeFrame,3);
```

```
edgeWidth=0.05*handles.framePosns(homeFrame,3);
ht1=0.1;
epsln=0.01*handles.framePosns(homeFrame,3);
xx=0.1*handles.framePosns(homeFrame,3);;%x and y shifts for
   date seelction text and edit boxes
yy=-0.15*handles.framePosns(homeFrame,3);;%x and y shifts for
   date seelction text and edit boxes
LOADEDARRAYSlistBoxpsn=[handles.framePosns(homeFrame,1)+
   edgeWidth edgeWidth 3*wd2, 0.85];
LOADEDARRAYStxtpsn=[handles.framePosns(homeFrame,1)+edgeWidth
   1-5*edgeWidth 3*wd2, ht1];
PLOTTSpbpsn=[handles.framePosns(homeFrame,1)+2*edgeWidth+3*wd2
   1-ht1-5*edgeWidth 1.3*wd2, ht1];
%SPCGRMpbpsn=[handles.framePosns(homeFrame,1)+2*edgeWidth+3*wd2
    1-2*ht1-7*edgeWidth 1.3*wd2, ht1];

YYYY='YYYY';
dv=datevec(date);
YYYY=zeroPadStr(dv(1),4);

handles.SRtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text','String','Sampling Rate','units',...
    'normalized','Position',SRtxtpsn, 'BackgroundColor',
       textBoxColour);
handles.SRpopup = uicontrol('Style', 'popupmenu','String',
   SRSTRINGS,...
                    'Units','normalized','Position',SRpoppsn);
handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
   , 'text','String','SELECT ARRAY','units',...
    'normalized','Position',arraySeltxtpsn, 'BackgroundColor',
       textBoxColour);
handles.ArraySelPopup=uicontrol('Parent', handles.Parent, '
   Style', 'popupmenu','String',arrayList,'units',...
```

```
80      'normalized','Position',arraySelpoppsn);%,'Callback',{
          @array_choose, handles});
81  handles.starttxt=uicontrol('Parent', handles.Parent, 'Style', '
      text','String','Start','units',...
82      'normalized','Position',starttxtpsn, 'BackgroundColor',
          textBoxColour);
83  handles.YYtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text','String','YEAR','units',...
84      'normalized','Position',YYtxtpsn,'BackgroundColor',
          textBoxColour);
85  handles.YYstartedit=uicontrol('Parent', handles.Parent, 'Style'
      , 'edit','String',YYYY,'units',...
86      'normalized','Position',YYstarteditpsn, 'Tag','YYeditTag');
87  handles.MMDDtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text','String','MM DD','units',...
88      'normalized','Position',MMDDtxtpsn,'BackgroundColor',
          textBoxColour);
89  handles.MMstartedit=uicontrol('Parent', handles.Parent, 'Style'
      , 'edit','String','MM','units',...
90      'normalized','Position',MMstarteditpsn);
91  handles.DDstartedit=uicontrol('Parent', handles.Parent, 'Style'
      , 'edit','String','DD','units',...
92      'normalized','Position',DDstarteditpsn);
93  handles.HRtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text','String','UT HH:MM','units',...
94      'normalized','Position',HRtxtpsn,'BackgroundColor',
          textBoxColour);
95  handles.HRstartedit=uicontrol('Parent', handles.Parent, 'Style'
      , 'edit','String','HH:MM','units',...
96      'normalized','Position',HRstarteditpsn);
97  handles.nSegtxt=uicontrol('Parent', handles.Parent, 'Style', '
      text','String','N Segments','units',...
98      'normalized','Position',nSegtxtpsn,'BackgroundColor',
          textBoxColour);
```

```
handles.nSegEdit=uicontrol('Parent', handles.Parent, 'Style', 'edit','String','nSeg?','units',...
    'normalized','Position',nSegeditpsn,'tooltipstring','How many segments after the start time do you wish to load');
%handles.LOADDATApb=uicontrol('Parent', handles.Parent, 'Style', 'pushbutton','String','LOAD DATA','units',...
%    'normalized','Position',LOADDATApbpsn,'callback',{@LOADDATAFCN,handles},'tooltipstring','Loads Data for Plotting','Tag','Load');
handles.LOADDATApb=uicontrol('Parent', handles.Parent, 'Style', 'pushbutton','String','LOAD FCs','units',...
    'normalized','Position',LOADDATApbpsn,'callback','loadFCRA_CB','tooltipstring','Loads Data for Plotting','Tag','Load');
%used to be loadPlotCB_kk,
handles.LOADEDARRAYSlistbox=uicontrol('Parent', handles.Parent, 'Style', 'listbox','units',...
    'normalized','Position',LOADEDARRAYSlistBoxpsn,'tooltipstring','LoadeD arrays','Tag','LoadedArraysListbox');%'String','LOAD DATA',
handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style', 'text','String','LOADED ARRAYS','units',...
    'normalized','Position',LOADEDARRAYStxtpsn, 'BackgroundColor', textBoxColour);
handles.PLOTTSpb=uicontrol('Parent', handles.Parent, 'Style', 'Pushbutton','String','PLOT SPECGM','units',...
    'normalized','Position',PLOTTSpbpsn, 'BackgroundColor', textBoxColour,'callback','loadFCRA_CB','Tag','Plot');
%handles.SPCGRMpb=uicontrol('Parent', handles.Parent, 'Style', 'Pushbutton','String','SPECTROGRAM','units',...
%    'normalized','Position',SPCGRMpbpsn, 'BackgroundColor', textBoxColour,'callback','loadPlotCB_kk','Tag','SPCGRM');

set(handles.SRpopup, 'callback', {@setSRcbk,handles});
```

```
guidata(handles.Parent,handles);
end


function []=setSRcbk(hObject, eventdata, handles)
SR = getSR(get(handles.SRpopup,'Value'));
if SR=='L'
    set(handles.HRstartedit,'String','00:00');
    set(handles.HRstartedit,'style','text');
    %set(handles.HRendedit,'String','24:00');
    %set(handles.HRendedit,'style','text');
else
    set(handles.HRstartedit,'String','HH:MM');
    set(handles.HRstartedit,'style','edit')
    %set(handles.HRendedit,'String','HH:MM');
    %set(handles.HRendedit,'style','edit');
end
%get(handles.HRstartedit)
end
```

```
function [] = SpecAvgGUI(hObject, eventdata, handles)
%OUTPUTS:
%Key Variables:
%There are 4 Panels in this GUI.
%Panel 1 Date, time, smoothing, update button
%Panel 2 One pushbutton and one checkbox for each Day in
   SpecAvg.cfg
%Panel 3 Time Series Plotter
%Panel 4 Spectral Plotter

global ULFEMpath DATApath verbose

prepSpecAvgGUI;
handles=guidata(handles.SpecAvgFig);
handles = populateSpecAvgGUI(hObject, eventdata, handles);
guidata(handles.SpecAvgFig,handles);
end
```

```
function [varargout]=SpecPlotter(varargin)
%A GUI TO PROCESS TS files.  This function is the callback to the Proc TS
%button on the ULFEM GUI.  The Main task of this program is to convert
%timesereis to FC files, where FCs have units of mV/km/sqrt(Hz) and
%nT/sqrt(Hz)


global ULFEMpath DATApath ARRAYS SLaSH syspath
%Get metadata needed to setup GUI about Arrays and Sampling Rates
%%
%THIS SUFF NEEDED IF ADDING A FRAME FOR PLOTTING WHOLE ARRAYS AT ONCE
% load([syspath,'sr_list']);
% load([syspath,'bFC']);
% SRSTRINGS{1}=' ';
% for i=1:length(sr_list.letters)
%     SRSTRINGS{i+1}=[num2str(sr_list.numbers(i)),' Hz ', sr_list.letters(i)];
% end
% [arrayList ARRAYS]=readArrays();
% %PWSTRINGS:, AWSTRINGS
% PWSTRINGS={'NONE','First Differece','ARMA'};
% AWSTRINGS={'BOXCAR','HAMMING','HANN'};

%%
%Initiate the GUI and specify its geometery
load([syspath,'bFC']);
handles.bFC=bFC;
handles.GUI_POSN=[20,550,550,200];
```

```
handles.plotSpecGUI=figure('Name', 'Plot Power/AMplitude Spectra', 'NumberTitle','off',...
                'Position',handles.GUI_POSN,'units','normalized',...
                'tag','plotSpecGUI', 'Color', [0.1 0.1 0.5]);

%embed buttons in a frame for portability
handles.framePosns(1,:)=[0 0 1 1];
%handles.framePosns(2,:)=[0 0.6 1 0.3];
%handles.framePosns(3,:)=[0 0.4 1 0.2];
%handles.framePosns(4,:)=[0 0.2 1 0.2];
frmCols=[0 2 4 6 8]/10;
for f=1:size(handles.framePosns,1)
handles.Frame{f}=uipanel( ...
        'Parent',handles.plotSpecGUI,...
        'units','normalized',...
        'Position',handles.framePosns(f,:),...
        'BackgroundColor',[1 1 frmCols(f)],...
        'tag',['frame',num2str(f)]);
end



wd1=0.40;
wd2=0.66*wd1;
ht1=0.1;
tbh=0.05;%text box height
epsln=0.01;
xx=0.091;%x and y shifts for date seelction text and edit boxes
yy=-0.15;%x and y shifts for date seelction text and edit boxes
scl=0.66;
textBoxColour = [0.5 0.5 1];
%%%FRAMEWISE FRAME1
nWidgets{1}=[2 2]; %number of buttons or thingys of equal size
```

```
%when it is 2x2, use 9%pad top and bottom, 40% for widgets and 1% epsilon
pad=0.09;
widgetH=0.4;
wE=0.01;%widgetEpsilon
TSFILEtxtpsn=[pad 1-pad-ht1 wd1 ht1];
BROWSEpbpsn=[pad+wd1+2*epsln 1-pad-ht1 wd1 ht1];
TSFILEeditboxpsn=[pad 1-pad-2*ht1 1-2*pad ht1];
nSmoothtxtpsn=[pad 1-pad-3*ht1-3*epsln wd1/2 ht1];
nSmootheditboxpsn=[pad+wd1/2 1-pad-3*ht1-3*epsln wd1/2 ht1];
PLOTpbpsn=[(1-wd1)/2 1-3*pad-4*ht1 wd1 ht1];



handles.Parent=handles.Frame{1};

%make the buttons and boxes
handles.TSFILEtxt=uicontrol('Parent', handles.Parent, 'Style', 'text','String','TS File','units',...
    'normalized','Position',TSFILEtxtpsn, 'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
handles.TSFILEedit=uicontrol('Parent', handles.Parent, 'Style', 'edit','Units','normalized','Position',TSFILEeditboxpsn,...
    'callback', {@lbox2_OpeningFcn,handles});%'create','/home/kappler/')
handles.BROWSEpb=uicontrol('Parent', handles.Parent, 'Style', 'pushbutton','Units','normalized','Position',BROWSEpbpsn,...
    'callback', {@loadEdit,handles},'String','BROWSE')
handles.nSmoothtxt=uicontrol('Parent', handles.Parent, 'Style', 'text','String','N SMOOTH','units',...
    'normalized','Position',nSmoothtxtpsn, 'BackgroundColor', textBoxColour,'FontName', 'FixedWidth');
handles.nSmoothedit=uicontrol('Parent', handles.Parent, 'Style', 'edit','Units','normalized','Position',nSmootheditboxpsn,'
```

```
   String','1');
handles.PLOTpb=uicontrol('Parent', handles.Parent, 'Style', '
   pushbutton','Units','normalized','Position',PLOTpbpsn,...
    'callback', {@plotSpec,handles},'String','PLOT SPEC')
guidata(handles.plotSpecGUI,handles)
end
function loadEdit(hObject, eventdata, handles, varargin)
global DATApath
[x y]=uigetfile(DATApath);
set(handles.TSFILEedit,'String',[y,x])
end

function [] = plotSpec(hObject, eventdata, handles)
    %Convert TS to FC.  Program Flow:
    %1. Identify the file list to be processed.
    global ULFEMpath DATApath ARRAYS SLaSH
    SLaSH=loadSLASH;
    %Collect Parameters SR,start time, end time,
    fName=get(handles.TSFILEedit,'String');
    nSmoothStr=get(handles.nSmoothedit,'String');
    nSmooth=str2num(nSmoothStr);
    a=strfind(fName,'_')
    SRCH=fName(a+1:a+3)
    SR=SRCH(1);
    CH=SRCH(2:3);
    dStr=fName(a+5:a+7)
    a=strfind(fName,'data')
    yStr=fName(a+6:a+9)
    a=strfind(fName,'TS/')
    b=strfind(fName,SRCH)
    siteID=fName(a+3:b(1)-2)
    %prewhiteningMethodValue=get(handles.PWpopup,'Value');
    %prewhiteningChoices=get(handles.PWpopup,'String');
```

```
%prewhiteningMethod=prewhiteningChoices{
   prewhiteningMethodValue}
prewhiteningMethod='NONE';
%apodWndwValue=get(handles.AWpopup,'Value');
%ApodWndwChoices=get(handles.AWpopup,'String');
%ApodizationWndw=ApodWndwChoices{apodWndwValue}
ApodizationWndw='HAMMING';
%Nwndw=str2num(get(handles.WndwWdthedit,'String'))
%Lwndw=str2num(get(handles.Overlapedit,'String'))
%nDL=numel(handles.bFC)

if regexp(SR,'L')
    dT=1
elseif regexp(SR,'B')
    dT=1/40;
end


if regexp(SRCH,'Q')
    fieldType='ELECTRIC';
elseif regexp(SRCH,'T')
    fieldType='MAGNETIC';
end
clear y
load(fName);
if isfield(y,'flag')
    warnmssg=['Requested Time Series File: ',fName,' is
       Flagged: ',y.flag];
    warning(warnmssg)
    %Proceed to a case/switch if types of flags become
    %numerous/complex, for now just skip flagged files
else
    display(['Identify appropriate EPOCH'])
```

```
147         epochMeta=associateEpochNumber2(yStr,dStr,siteID,CH); %
              modernized 01 May 2010 for use with spreadsheets;
148         plotTS2Spec(y.data, epochMeta,CH,prewhiteningMethod,
              ApodizationWndw, dT, fieldType,fName,nSmooth);
149     end
150 end
151
152
153 function []=setSRcbk(hObject, eventdata, handles)
154 SR = getSR(get(handles.SRpopup,'Value'));
155 if SR=='L' | SR=='V'
156     set(handles.HRstartedit,'String','00:00');
157     set(handles.HRstartedit,'style','text');
158     set(handles.HRendedit,'String','24:00');
159     set(handles.HRendedit,'style','text');
160 else
161     set(handles.HRstartedit,'String','HH:MM');
162     set(handles.HRstartedit,'style','edit')
163     set(handles.HRendedit,'String','HH:MM');
164     set(handles.HRendedit,'style','edit');
165 end
166 %get(handles.HRstartedit)
167 end
```

```
function [] =pdf811(svfName,varargin)
%usage:pdf811('/home/oper/ux060blahblah');
%saves gcf as /home/oper/ux060blahblah.pdf
%usage: save811('x','orient','landscape','figNum','2')
%svfName, e.g. save811('/home/oper/ux060blahblah');
%saves gcf as /home/oper/ux060kjsf.eps
%vararging looks like:
%'figNum','4','orient','portrait','pdf','1'
figureHandle='gcf';
lv=length(varargin);
paperPosn=[0 0 8.5 11];
ornt='landscape';
if lv > 0
    for i=1:lv
        if regexp('figNum',varargin{i})
            cmd=['figure(',varargin{i+1},');'];
            eval(cmd)
        end;
    end
    for i=1:lv
        if regexp('orient',varargin{i})
            ornt=varargin{i+1};
        end
    end
end

    set(gcf, 'PaperPositionMode','manual');
    set(gcf, 'PaperUnits','inches');
    set(gcf, 'PaperPosition',paperPosn);
    orient(ornt);

    cmd=['print -dpdf ',svfName,'.pdf',];
    eval(cmd);
```

```
function FC= TS2CFC(TS, epochMeta,CH, prewhit, apodWindow,
   Nwndw, Lwndw,nDL, dT, fieldType,bFC)


%If E, B, V? figure typ of TF
global COILpath

PREWHT=0;ordr=5; %not used now
SAVE=1;
N=Nwndw;%256; %window length
L=Lwndw;%64; %number of overlapping points
nDL=numel(bFC); %number of decimation levels
C2V=str2num(epochMeta.cpv); %apply digitzer counts to volts
yV=TS/C2V;

%correct electric channels for EFSC gains, and normalize by 1e6
   /trodelength to get mV/km TS
if regexp(fieldType,'ELECTRIC')
    gainV=str2num(epochMeta.gainV);
    trodeLength=str2num(epochMeta.length);
    yV=(1e6)*yV/(gainV*trodeLength);
end

%APODIZATION WINDOW SELECT:  FOR NOW DEFAULT TO HAMMING
apodWndw=ones(1,N); %Boxcar
cohGain=1.0;
equivNoiseBndwdth=1.0;
if regexp(apodWindow,'HAMMING')
    apodWndw=hamming_ulfem(N);
    cohGain=0.54;
    equivNoiseBndwdth=1.36;
elseif regexp(apodWindow,'HANN')
    apodWndw=hanning(N);
    cohGain=0.5;
```

```
33      equivNoiseBndwdth=1.5;
34  end
35
36  nanCheck=sum(sum(isnan(yV)));
37  if nanCheck>0
38      display(['There are still nans in this TSD data! DOI=',
           num2str(doi)])
39      return
40  end
41
42  if regexp(fieldType,'MAGNETIC')
43          %USE regexpi to load coil file
44          if regexp(epochMeta.coilID,'unknown')
45              display(['Unknown Coil ID, using 9519'])
46              coilID='bf4-9519';
47          else
48              coilID=epochMeta.coilID;
49          end
50          [COILpath,coilID,'.rsp']
51          bf4=textread([COILpath,coilID,'.rsp'],'','headerlines'
              ,6);
52  end
53
54  %%  loop over decimation levels
55  for iD=1:nDL
56      clear yVDec
57      display(['Treating Decimation Level ',num2str(iD-1)]);
58      dt=2^(iD-1)*dT;  %decimation-corrected sampling rate
59      display(['Current Time Step for data initially sampled at $
           \Delta$ t=',num2str(dT),' is ', num2str(dt)]);
60      df=1/(N*dt);
61      freqs=0:df:(N-1)*df;
62      %now decimate by appropriate amount of times:
63      yVDec=nan(round(length(yV)/(2^(iD-1))),1);
```

```
64      display(['size of decmated data is ',num2str(size(yVDec))])
           ;
65      temp=yV;
66      for decL=1:iD-1
67          temp=decimate_ulfem(temp,2);
68  %        display(['size of temp is ',num2str(size(temp))])
69      end
70      yVDec=temp;
71      display(['size of yVDec is',num2str(size(yVDec))]);
72      APOD=mkWM(yVDec,N,L);
73      APOD(end,:)=[]; %get rid of last window so no worry nans,
           integral mulitples of N etc
74      for i=1:size(APOD,1)
75          APOD(i,:)=apodWndw;
76      end
77      %APOD
78      %now generate an instrument transfer function for each coil
           :
79      if regexp(fieldType,'MAGNETIC')
80          display(['MAG'])
81          clear phs
82          phs=interp1(bf4(:,1),exp(j*deg2rad(bf4(:,3))),freqs);
83          TF=interp1(bf4(:,1),bf4(:,2),freqs).*phs;
84      else
85          TF=ones(1,size(APOD,2));
86      end
87      %may need to  add the EFSC PHASE data
88      %display(['size of yV mtx is',num2str(size(yV))]);
89      WM=mkWM(yVDec,N,L);%window the data.  Data are in ROWS
90      WM(end,:)=[]; %get rid of the last window; nB this can nuLL
           out the final decimation level!
91      WM=detrend(WM'); %DATA are now in COLUMNS
92      if PREWHT
93          for wdw=1:size(WM,2)
```

```
94              sig=WM(:,wdw)';
95              [B,A]=prony(sig,ordr,ordr);
96          end
97          WM=WM.*APOD'; %Window the data
98          sig2=conv(sig,(A));
99          sig2=sig2(ceil(ordr/2):end-ceil(ordr/2));
100         ft2=fft(sig2);
101         zpA=[zeros(1,length(ft2)-ordr-1) A];
102         fA=fft(zpA);
103         spc=ft2./fA;
104         sclSpc=sqrt((2*dt)/(1.36*N))*spc/(0.54);
105         sclSpc=sclSpc./squeeze(ITF(ch,1,:))';
106     else
107         WM=WM.*APOD'; %Window the data
108         fWM=sqrt((2*dt)/(equivNoiseBndwdth*N))*fft(WM)/cohGain;
109         %fWM=fWM';
110         ITF=zeros(size(fWM));
111         %loop over the windows, each window should have size
              Nwndw
112         for i = 1: size(ITF,2)
113             ITF(:,i)=TF;
114         end
115         sclSpc=fWM./ITF;
116
117     end
118     for ib=1:numel(bFC{iD})
119         FC{iD}{ib}(:,:)=sclSpc(bFC{iD}{ib}.FCI,:);
120     end
121 end
```

```
function FC= TS2FC_single(TS, epochMeta, prewhit, apodWindow,
   dt, fieldType)

%If E, B, V? figure typoe of TF
global COILpath
PREWHT=0;ordr=5;
C2V=str2num(epochMeta.cpv)
yV=TS/C2V;
N=length(yV);
%correct electric channels for EFSC gains, and normalize by
%1e6/trodelength to get mV/km TS
if regexp(fieldType,'ELECTRIC')
    gainV=str2num(epochMeta.gainV);
    trodeLength=str2num(epochMeta.length);
    yV=(1e6)*yV/(gainV*trodeLength);
end
hamm=hamming_ulfem(N);
nanCheck=sum(sum(isnan(yV)));
if nanCheck>0
    display(['There are still nans in this TSD data! DOI=',
       num2str(doi)])
    return
end
if regexp(fieldType,'MAGNETIC')
    %USE regexpi to load coil file
    if regexp(epochMeta.coilID,'unknown')
        display(['Unknown Coil ID, using 9519'])
        coilID='bf4-9519'
    else
        coilID=epochMeta.coilID;
    end
    [COILpath,coilID,'.rsp']
    bf4=textread([COILpath,coilID,'.rsp'],'','headerlines',6);
end
```

```
df=1/(N*dt);
freqs=0:df:(N-1)*df;
if regexp(fieldType,'MAGNETIC')
    clear phs
    phs=interp1(bf4(:,1),exp(j*deg2rad(bf4(:,3))),freqs);
    TF=interp1(bf4(:,1),bf4(:,2),freqs).*phs;
else
    TF=ones(1,N);
end
%may need to  add the EFSC PHASE data
fWM=sqrt((2*dt)/(1.36*N))*fft(yV)/(0.54);
sclSpc=fWM./TF;
sclSpc=sclSpc(1:round(N/2));
FC.spec=sclSpc;
FC.frqs=freqs(1:length(sclSpc));
```

```
function []=ULFEM()
%{
The call to initialize the ULFEM package: Creates the main GUI
   from
which all ULFEM applications are called.
%}

clear all; close all

global ULFEMpath verbose rect0
verbose=1;
setPaths;
configureDataDirectories;
initializeSRList;
rect0=[100 100 850 600];
ULFEM_GUI_POSN=[30,50,900,100];
ulfemGui=figure('MenuBar','none','Name', 'ULFEM', ...
    'NumberTitle','off','Position',ULFEM_GUI_POSN,...
    'units','normalized','tag','ULFEMGUI', 'Color', [0.1 0.1
       0.5]);

%<settings to control pushbutton widths and spacings>
wd1=0.13;
ht1=0.2;
epsln=0.01;
Get_NCEDCpbpsn=[0 1-2*ht1 wd1 ht1];
%Update_METApbpsn=[wd1+epsln 1-2*ht1 wd1 ht1];
Array_Configpbpsn=[1*(wd1+epsln) 1-2*ht1 wd1 ht1];
Plot_Mgrpbpsn=[2*(wd1+epsln) 1-2*ht1 wd1 ht1];
Proc_Mgrpbpsn=[3*(wd1+epsln) 1-2*ht1 wd1 ht1];
Plot_Spcpbpsn=[4*(wd1+epsln) 1-2*ht1 wd1 ht1];
SPCGRM_pbpsn=[5*(wd1+epsln) 1-2*ht1 wd1 ht1];
specAvg_pbpsn=[6*(wd1+epsln) 1-2*ht1 wd1 ht1];
%<\settings to control pushbutton widths and spacings>
```

```
33
34
35 %<generate pushputtons and link them to functional callbacks>
36 handles.Get_NCEDCpb=uicontrol('Parent', ulfemGui, 'Style', '
      pushbutton',...
37     'String','GET NCEDC','units','normalized','Position',
         Get_NCEDCpbpsn,...
38     'callback',{@Get_NCEDC_GUI});
39
40 %handles.Update_METApb=uicontrol('Parent', ulfemGui, 'Style',
      ...
41 %    'pushbutton','String','Update Metadata','units','
      normalized',...
42 %    'Position',Update_METApbpsn,'callback',{@UpdateMeta});
43
44 handles.Array_Configpb=uicontrol('Parent', ulfemGui, 'Style',
      ...
45     'pushbutton','String','Manage Arrays','units', 'normalized'
         ,...
46     'Position',Array_Configpbpsn,'callback',{@ArrayManager});
47
48 handles.Plot_Configpb=uicontrol('Parent', ulfemGui, 'Style',
      ...
49     'pushbutton','String','Plot Time Series','units','
         normalized',...
50     'Position',Plot_Mgrpbpsn,'callback',{@PlotManager});
51
52 handles.Proc_Configpb=uicontrol('Parent', ulfemGui, 'Style',
      ...
53     'pushbutton','String','Proc Time Series','units','
         normalized',...
54     'Position',Proc_Mgrpbpsn,'callback',{@ProcManager});
55
```

```
handles.Plot_Spectrapb=uicontrol('Parent', ulfemGui, 'Style',...
    'pushbutton','String','Plot Spectra','units', 'normalized',...
    'Position',Plot_Spcpbpsn,'callback',{@SpecPlotter});

handles.SPCGRMpb=uicontrol('Parent', ulfemGui, 'Style', 'pushbutton',...
    'String','Spectrogram','units','normalized','Position',SPCGRM_pbpsn,...
    'callback',{@SpcgrmManager});

handles.SpecAvgpb=uicontrol('Parent', ulfemGui, 'Style', 'pushbutton',...
    'String','SPEC AVG','units','normalized','Position',specAvg_pbpsn,...
    'callback',{@SpecAvgGUI});
end
%<\generate pushputtons and link them to functional callbacks>

%function UpdateMeta(hObject, eventdata)
%    %UpdateMetaDataGUI();
%    display('This Function has been replaced by a manual metadata updating
%proceedure.  See User Manual')
%end
```

```
function [varargout]=UpdateMetadataGUI(varargin)
%A GUI TO Update Metadata from NCEDC.
%Use drop down menu for sampling rate.
%
global ULFEMpath DATApath ARRAYS SLaSH syspath scrpath sr_list
   metadatapath metaStageDir SYSpath
metaStageDir
load([syspath,'sr_list'])
SRSTRINGS{1}='ALL';
for i=1:length(sr_list.letters)
    SRSTRINGS{i+1}=[num2str(sr_list.numbers(i)),' Hz ',sr_list.
       letters(i)];
end
[arrayList ARRAYS]=readArrays();
handles.GET_META_GUI_POSN=[420,550,220,100];
updateMetadataGui=figure('MenuBar','none','Name', 'Get MetaData
   ', 'NumberTitle','off',...
                  'Position',handles.GET_META_GUI_POSN,'units',
                     'normalized',...
                  'tag','GETMETAGUI', 'Color', [0.1 0.1 0.5]);
wd1=0.40;
wd2=0.66*wd1;
ht1=0.1;
epsln=0.01;
xx=0.091;%x and y shifts for date seelction text and edit boxes
yy=-0.15;%x and y shifts for date seelction text and edit boxes
scl=0.66;
textBoxColour = [0.5 0.5 1];
SRtxtpsn=[0 1-ht1-epsln wd1 ht1];
SRpoppsn=[0 1-2*ht1-epsln wd1 ht1];
arraySeltxtpsn=[wd1+epsln 1-ht1-epsln 1.3*wd1 ht1];
arraySelpoppsn=[wd1+epsln 1-2*ht1-epsln 1.3*wd1 ht1];


```

```
OVERWRITEcbpsn=[xx+1*wd2 yy+1-5.8*ht1 1.5*wd1 2*ht1];
UPDATEpbpsn=[wd1 1-9.7*ht1 wd1 2*ht1];
[YYYY MM DD]=yesterday;
handles.Parent=updateMetadataGui;
handles.SRtxt=uicontrol('Parent', handles.Parent, 'Style', '
   text','String','Sampling Rate','units',...
    'normalized','Position',SRtxtpsn, 'BackgroundColor',
       textBoxColour);
handles.SRpopup = uicontrol('Style', 'popupmenu','String',
   SRSTRINGS,...
                    'Units','normalized','Position',SRpoppsn);
handles.ArraySelTxt=uicontrol('Parent', handles.Parent, 'Style'
   , 'text','String','SELECT ARRAY','units',...
    'normalized','Position',arraySeltxtpsn, 'BackgroundColor',
       textBoxColour);
handles.ArraySelPopup=uicontrol('Parent', handles.Parent, '
   Style', 'popupmenu','String',arrayList,'units',...
    'normalized','Position',arraySelpoppsn);

handles.OVERWRITEcb=uicontrol('Parent', handles.Parent, 'Style'
   , 'Checkbox','String','OVERWRITE','units',...
    'normalized','Position',OVERWRITEcbpsn);
handles.UPDATEpb=uicontrol('Parent', handles.Parent, 'Style', '
   pushbutton','String','Start Update','units',...
    'normalized','Position',UPDATEpbpsn,'callback',{
       @UpdateMetadata,handles}, 'tooltipstring','Updates
       Metadata Files');
end


function [] = UpdateMetadata(hObject, eventdata, handles)
    global ULFEMpath DATApath ARRAYS SLaSH syspath scrpath
       sr_list metadatapath metaStageDir
    %metaStageDir
```

```
54      [arrayList ARRAYS]=readArrays();  %%added 22Nov2009 this
          will generalize update metadata.
55      SLaSH=loadSLASH;
56      [SR HZ] = getSR(get(handles.SRpopup,'Value'));
57      if strmatch(SR,'ALL')
58          SRs=sr_list.letters;
59          HZs=sr_list.numbers;
60      else
61          SRs=SR;
62          HZs=HZ;
63      end
64      nSR=length(SRs);
65      arrayNameIndex=get(handles.ArraySelPopup,'Value');
66      RA=ARRAYS{arrayNameIndex};
67      overwriteAsYouDownload=get(handles.OVERWRITEcb,'Value');
68      clear SITECH
69      SITECH=RA.chrArray;
70      for iSR=1:nSR
71          clear meta
72          meta.localStagingDir=scrpath;
73          meta.SR=SRs(iSR);
74          meta.Hz=HZs(iSR);
75          display(['Treating sampling rate ', num2str(iSR) ,' of
              ',num2str(nSR),' :',meta.SR,' = ',num2str(meta.Hz),'
              Hz'])
76          display(' ');display(' ');
77
78          for sc=1:numel(SITECH)
79              siteID=SITECH{sc}.locn;
80              meta.site=siteID;
81              network=SITECH{sc}.net;
82              meta.Network=network;
83              ch=SITECH{sc}.chn;
84              meta.ch=ch;
```

```
85              display(' ');
86              display(['Retrieving ASCII Header for ',meta.site,'
                   .',meta.Network,'.',meta.SR,meta.ch]);
87              OK=getAsciiHeader(meta);
88              if OK
89                  display(['ASCII Header Retrieved'])
90                  [L startT endT]=idEpochs(meta);
91                  display([num2str(numel(L)),' distinct Epoch(s)
                       Identified'])
92                  cleanScratch();
93                  %%FLOW PART 3
94                  for epch=1:numel(L)
95                      %make sure each epoch is at least a day
                           long,
96                      txt1=['Epoch #',num2str(epch),' starts at:
                           ' startT{epch}.textStr,' and ends at: ',
                           endT{epch}.textStr];
97  %                   txt2=[' And has Matlab datenum width:',
    num2str(endT{epch}.datenum-startT{epch}.datenum)];
98                      disp([txt1])
99  %                   display([txt1,' ',txt2])
100                     dayWidth=endT{epch}.datenum-startT{epch}.
                           datenum;
101                     if dayWidth>2
102                         midTimeDateNum=startT{epch}.datenum+(
                               endT{epch}.datenum-startT{epch}.
                               datenum)/2;
103                         datestr(midTimeDateNum);
104                         XX=meta;
105                         XX.durn='1d';  %set meta durn
106                         XX.yStr=datestr(midTimeDateNum,'yyyy');
107                         XX.dStr=zeroPadStr(num2str(floor((
                               midTimeDateNum+1)-datenum(str2num(XX.
                               yStr),1,1))),3);
```

```
                    if str2num(XX.dStr)==0
                        XX.dStr='001';
                    end
                    XX.hStr=zeroPadStr(datestr(
                       midTimeDateNum,'HH'),2);
                    XX.epch=epch;
                    %CHECK IF EPOCH FILE FOR THIS CFG
                       IALREADY EXISTS
                    fNameToCreate=[metadatapath, XX.site,'_
                       ',XX.SR,XX.ch,'_EPOCH',zeroPadStr(XX.
                       epch,2),'.txt'];
                    if (exist(fNameToCreate) &
                       overwriteAsYouDownload==0)
                        display(['File ',fNameToCreate,'
                           already exists, click ''OVERWRITE
                           '' if you want to regenerate the
                           file'])
                    else
                        fName=getFullAsciiHeader(XX);
                    end
                    %now parse the .txt file into fields,
                       and store it as
                    %txt2matIndex(fName)
                    %PKD.LT2.matDatenumStart.matDatenumEnd.
                       mat
                else
                    mssg='EPOCH is not even a day long...
                       need to deal with this case'
                    warning(mssg);
                    display(['Short EPOCH violation for: ',
                       siteID,' ',meta.SR,ch])
                end
            end
        end
```

```
        end
    end



end

function [SR HZ] = getSR(srIndex)
    global ULFEMpath
    if srIndex==1
        SR='ALL';
        HZ='ALL';
    else
        load([SYSpath,'/sr_list'])
        SR=sr_list.letters(srIndex-1);
        HZ=sr_list.numbers(srIndex-1);
    end
end
```

```
function addpath_recurse(strStartDir, caStrsIgnoreDirs,
   strXorIntAddpathMode, blnRemDirs, blnDebug)
%ADDPATH_RECURSE  Adds (or removes) the specified directory and
    its subfolders
% addpath_recurse(strStartDir, caStrsIgnoreDirs,
   strXorIntAddpathMode, blnRemDirs, blnDebug)
%
% By default, all hidden directories (preceded by '.'),
   overloaded method directories
% (preceded by '@'), and directories named 'private' or 'CVS'
   are ignored.
%
% Input Variables
% ===============
% strStartDir::
%   Starting directory full path name.  All subdirectories (
   except ignore list) will be added to the path.
%   By default, uses current directory.
% caStrsIgnoreDirs::
%   Cell array of strings specifying directories to ignore.
%   Will also ignore all subdirectories beneath these
   directories.
%   By default, empty list. i.e. {''}.
% strXorIntAddpathMode::
%   Addpath mode, either 0/1, or 'begin','end'.
%   By default, prepends.
% blnRemDirs::
%   Boolean, when true will run function "in reverse", and
%   recursively removes directories from starting path.
%   By default, false.
% blnDebug::
%   Boolean, when true prints debug info.
%   By default, false.
%
```

```
28 % Output Variables
29 % ================
30 % None. If blnDebug is specified, diagnostic information will
      print to the screen.
31 %
32 % Example(s)
33 % ==========
34 % (1) addpath_recurse();                                        %
      Take all defaults.
35 % (2) addpath_recurse('strStartDir');                          %
      Start at 'strStartDir', take other defaults. i.e. Do addpath
      ().
36 % (3) addpath_recurse('strStartDir', '', 0, true);            %
      Start at 'strStartDir', and undo example (2). i.e. Do rmpath
      ().
37 % (4) addpath_recurse('strStartDir', '', 'end', false, true); %
      Do example (2) again, append to path, and display debug info.
38 % (5) addpath_recurse('strStartDir', '', 1, true, true);      %
      Undo example (4), and display debug info.
39
40 %
41 % See Also
42 % ========
43 % addpath()
44 %
45 % Developers
46 % ===========
47 % Init  Name             Contact
48 % ----  ---------------
      ---------------------------------------------
49 % AK    Anthony Kendall  anthony [dot] kendall [at] gmail [dot]
       com
50 % JMcD  Joe Mc Donnell
51 %
```

```
% Modifications
% =============
% Version   Date      Who    What
% --------  --------  -----
   ------------------------------------------------------------------------

% 00.00.00  20080808  AK     First created.
%           20090410  JMcD   Redo input argument processing/
   checking.
%                             Only do processing/checking once.
   Do recursion in separate function.
%           20090411  JMcD   Add debugging mode to display run
   info.
%           20090414  AK     Modified variable names, small
   edits to code, ignoring CSV by default
%           20091104  AK     Modified an optimization for Mac
   compatibility,
%                             recursive calls to just build the
   string for
%                             addpath/rmpath rather than call it
    each time

%
   ------------------------------------------------------------------------

%Error messages.
strErrStartDirNoExist = 'Start directory does not exist ???';
strErrIgnoreDirsType = 'Ignore directories must be a string or
   cell array. See HELP ???';
strErrIllAddpathMode = 'Illegal value for addpath() mode.  See
   HELP ???';
strErrIllRevRecurseRemType = 'Illegal value for reverse recurse
    remove, must be a logical/boolean.  See HELP ??';

```

```
72 strErrWrongNumArg = 'Wrong number of input arguments.  See HELP
       ???';
73 strAddpathErrMessage = strErrIllAddpathMode;
74
75 %Set input args defaults and/or check them.
76 intNumInArgs = nargin();
77 assert(intNumInArgs <= 5, strErrWrongNumArg);
78
79 if intNumInArgs < 1
80  strStartDir = pwd();
81 end
82
83 if intNumInArgs < 2
84   caStrsIgnoreDirs = {''};
85 end
86
87 if intNumInArgs >= 2 && ischar(caStrsIgnoreDirs)
88   caStrsIgnoreDirs = { caStrsIgnoreDirs };
89 end
90
91 if intNumInArgs < 3 || (intNumInArgs >= 3 && isempty(
       strXorIntAddpathMode))
92   strXorIntAddpathMode = 0;
93 end
94
95 if intNumInArgs >= 3 && ischar(strXorIntAddpathMode)  %Use 0/1
       internally.
96   strAddpathErrMessage = sprintf('Input arg addpath() mode "%s"
         ???\n%s', strXorIntAddpathMode, strErrIllAddpathMode);
97   assert(any(strcmpi(strXorIntAddpathMode, {'begin', 'end'})),
         strAddpathErrMessage);
98   strXorIntAddpathMode = strcmpi(strXorIntAddpathMode, 'end');
       %When 'end' 0 sets prepend, otherwise 1 sets append.
99 end
```

```
100
101 if intNumInArgs < 4
102   blnRemDirs = false;
103 end
104
105 if intNumInArgs < 5
106   blnDebug = false;
107 end
108
109 if size(caStrsIgnoreDirs, 1) > 1
110   caStrsIgnoreDirs = caStrsIgnoreDirs'; %Transpose from column
        to row vector, in theory.
111 end
112
113 %Check input args OK, before we do the thing.
114 strErrStartDirNoExist = sprintf('Input arg start directory "%s"
      ???\n%s', strStartDir, strErrStartDirNoExist);
115 assert(exist(strStartDir, 'dir') > 0, strErrStartDirNoExist);
116 assert(iscell(caStrsIgnoreDirs), strErrIgnoreDirsType);
117 assert(strXorIntAddpathMode == 0 || strXorIntAddpathMode == 1,
     strAddpathErrMessage);
118 assert(islogical(blnRemDirs), strErrIllRevRecurseRemType);
119 assert(islogical(blnDebug), 'Debug must be logical/boolean.
     See HELP.');
120
121 if blnDebug
122   intPrintWidth = 34;
123   rvAddpathModes = {'prepend', 'append'};
124   strAddpathMode = char(rvAddpathModes{ fix(
        strXorIntAddpathMode) + 1});
125   strRevRecurseDirModes = { 'false', 'true' };
126   strRevRecurseDirs = char(strRevRecurseDirModes{ fix(
        blnRemDirs) + 1 });
127   strIgnoreDirs = '';
```

```
128   for intD = 1 : length(caStrsIgnoreDirs)
129     if ~isempty(strIgnoreDirs)
130       strIgnoreDirs = sprintf('%s, ', strIgnoreDirs);
131     end
132     strIgnoreDirs = sprintf('%s%s', strIgnoreDirs, char(
          caStrsIgnoreDirs{intD}));
133   end
134   strTestModeResults = sprintf('... Debug mode, start recurse
        addpath arguments ...');
135   strTestModeResults = sprintf('%s\n%*s: "%s"',
        strTestModeResults, intPrintWidth, 'Start directory',
        strStartDir);
136   strTestModeResults = sprintf('%s\n%*s: "%s"',
        strTestModeResults, intPrintWidth, 'Ignore directories',
        strIgnoreDirs);
137   strTestModeResults = sprintf('%s\n%*s: "%s"',
        strTestModeResults, intPrintWidth, 'addpath() mode',
        strAddpathMode);
138   strTestModeResults = sprintf('%s\n%*s: "%s"',
        strTestModeResults, intPrintWidth, 'Reverse recurse remove
        directories', strRevRecurseDirs);
139   disp(strTestModeResults);
140 end
141
142 %Don't print the MATLAB warning if remove path string is not
      found
143 if blnRemDirs, warning('off', 'MATLAB:rmpath:DirNotFound'); end
144
145 %Build the list of directories
146 caAddRemDirs = {};
147 [caAddRemDirs] = addpath_recursively(caAddRemDirs, strStartDir,
      caStrsIgnoreDirs, strXorIntAddpathMode, blnRemDirs,blnDebug)
      ;
148
```

```
149 %Remove or add the directory from the search path
150 if blnRemDirs
151   if blnDebug, fprintf('"%s", removing from search path ...',
        strStartDir); end
152   rmpath(caAddRemDirs{:})
153 else
154   if blnDebug, fprintf('"%s", adding to search path ...',
        strStartDir); end
155   addpath(caAddRemDirs{:}, strXorIntAddpathMode);
156 end
157
158 %Restore the warning state for rmpath
159 if blnRemDirs, warning('on', 'MATLAB:rmpath:DirNotFound'); end
160
161 end % function addpath_recurse
162 %
      ------------------------------------------------------------------------
163
164 %
      ------------------------------------------------------------------------
165 function [caAddRemDirs] = addpath_recursively(caAddRemDirs,
      strStartDir, caStrsIgnoreDirs, strXorIntAddpathMode,
      blnRemDirs, blnDebug)
166 %Note:Don't need to check input arguments, because caller
      already has.
167
168 %Add this directory to the add/remove path list
169 caAddRemDirs = [caAddRemDirs,strStartDir];
170
171 strFileSep = filesep();
172 %Get list of directories beneath the specified directory, this
      two-step process is faster.
```

```
173 if ispc
174     saSubDirs = dir(sprintf('%s%s%s', strStartDir, strFileSep,
          '*.'));
175 else
176     saSubDirs = dir(strStartDir);
177 end
178 saSubDirs = saSubDirs([saSubDirs.isdir]); %Handles files
      without extensions that otherwise pass through previous
      filter
179
180 %Loop through the directory list and recursively call this
      function.
181 for intDirIndex = 1 : length(saSubDirs)
182   strThisDirName = saSubDirs(intDirIndex).name;
183   blnIgnoreDir = any(strcmpi(strThisDirName, { 'private', 'CVS'
        , '.', '..', caStrsIgnoreDirs{:} }));
184   blnDirBegins = any(strncmp(strThisDirName, {'@', '.'}, 1));
185   if ~(blnIgnoreDir || blnDirBegins)
186     strThisStartDir = sprintf('%s%s%s', strStartDir, strFileSep
          , strThisDirName);
187     if blnDebug, fprintf('"%s", recursing ...', strThisStartDir
          ); end
188     [caAddRemDirs] = addpath_recursively(caAddRemDirs,
          strThisStartDir, caStrsIgnoreDirs, strXorIntAddpathMode,
          blnRemDirs, blnDebug);
189   end
190 end % for each directory.
191
192 end % function addpath_recursively
193 %
      ------------------------------------------------------------------------

194 %__END__
```

```
1
2  function handles=siteChConfigCreateFcns(hObject, eventdata,
       handles)
3  %{
4  create site-channel configuration UI
5  %}
6
7      handles=allSitesListbox_CreateFcn(hObject, eventdata,
         handles);
8      handles=selectedSitesListbox_CreateFcn(hObject, eventdata,
         handles);
9      handles=siteChListbox_CreateFcn(hObject, eventdata,handles)
         ;
10 end
11 %%
12 function handles=allSitesListbox_CreateFcn(hObject, eventdata,
       handles)
13 %{
14 @type homeFrame: integer
15 @var homeFrame: lets the program know in which frame to place
       the listbox
16 @type edgeWidth: float
17 @var edgeWidth: the offset of the listbox from the edge of the
       frame.
18 default to 10%
19 @type listWidth: float
20 @var listWidth: the width of the list as a fraction of the
       Frame.
21 @type listHeight: float
22 @var listHeigth: the height of the list as a fraction of the
       Frame.
23 %}
24     homeFrame=1;
25     edgeWidth=0.1*handles.framePosns(homeFrame,3);
```

```
26      listWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
27      listHeight=handles.framePosns(homeFrame,4)-handles.topEdge*
           edgeWidth;
28
29      handles.allListbox=uicontrol( ...
30          'Parent',handles.ArryMgrFig,...
31          'Style', 'listbox',...
32          'units','normalized',...
33          'Position',[handles.framePosns(homeFrame,1)+edgeWidth,
              handles.horzLine+edgeWidth,listWidth,listHeight],...
34          'BackgroundColor',[1 1 0],...
35          'ToolTipString','Select the sites which you want to
              work with',...
36          'Tag', 'all_Listbox',...
37          'String',handles.SITES,...
38          'max',3);
39
40  end
41  %%
42  function handles=selectedSitesListbox_CreateFcn(hObject,
       eventdata, handles)
43  %{
44
45  Creates a listbox which cotains the selected sites. Lives in
       frame 3 on the
46  far right.
47  @type homeFrame: integer
48  @var homeFrame: lets the program know in which frame to place
       the listbox
49  @type edgeWidth: float
50  @var edgeWidth: the offset of the listbox from the edge of the
       frame.
51  default to 10%
52  @type listWidth: float
```

```
53 @var listWidth: the width of the list as a fraction of the
      Frame.
54 @type listHeight: float
55 @var listHeigth: the height of the list as a fraction of the
      Frame.
56 @type xPosn: float
57 @var xPosn: the pinned-corner position of the listbox within
      the frame
58 @type yPosn: float
59 @var yPosn: the pinned-corner position of the listbox within
      the frame
60
61 %}
62     homeFrame=3;
63     edgeWidth=0.1*handles.framePosns(homeFrame,3);
64     listWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
65     listHeight=handles.framePosns(homeFrame,4)-handles.topEdge*
          edgeWidth;
66     xPosn = handles.framePosns(homeFrame,1)+edgeWidth;
67     yPosn = handles.horzLine+edgeWidth
68     handles.selectedListbox=uicontrol(...
69         'Parent',handles.ArryMgrFig,...
70         'Style', 'listbox',...
71         'units','normalized',...
72         'Position',[xPosn, yPosn,listWidth,listHeight],...
73         'max',3,...
74         'tag', 'Selected_Sites_Listbox',...
75         'String',handles.selSites,...
76         'BackgroundColor',[1 1 0]);
77     if numel(handles.selSites)
78         set(handles.selectedListbox,'Value',1);
79     end
80 end
81
```

```
%%
function handles=siteChListbox_CreateFcn(hObject, eventdata,
   handles)
%{
Creates listbox at bottom of GUI to show the user the aggregate
    selected
sites and channels
@type homeFrame: integer
@var homeFrame: lets the program know in which frame to place
   the listbox
@type edgeWidth: float
@var edgeWidth: the offset of the listbox from the edge of the
   frame.
default to 10%
@type listWidth: float
@var listWidth: the width of the list as a fraction of the
   Frame.
@type listHeight: float
@var listHeigth: the height of the list as a fraction of the
   Frame.
@type xPosn: float
@var xPosn: the pinned-corner position of the listbox within
   the frame
@type yPosn: float
@var yPosn: the pinned-corner position of the listbox within
   the frame
%}
    homeFrame=4;
    edgeWidth=0.05*handles.framePosns(homeFrame,4);
    listWidth=handles.framePosns(homeFrame,3)-2*edgeWidth;
    listHeight=handles.framePosns(homeFrame,4)-2*edgeWidth;
    xPosn = handles.framePosns(homeFrame,1)+edgeWidth;
    yPosn = edgeWidth;

```

```
108     handles.siteChListbox=uicontrol( ...
109         'Parent',handles.ArryMgrFig,...
110         'Style', 'listbox',...
111         'units','normalized',...
112         'Position',[xPosn yPosn listWidth listHeight],...
113         'BackgroundColor',[1 1 0],...
114         'Tag', 'siteChListbox',...
115         'String','',...
116         'Enable','off');
117 end
```

```
function iDedEpoch=associateEpochNumber(yStr,dStr,sta,srCH)
%usage: iDedEpoch=associateEpochNumber(yStr,dStr,sta,srCH)
global EPOCHSpath METADATApath metadatapath
%display(['getting site E scale factor'])
%%ID EPOCH
sta=strrep(sta,' ','');
fiq=[EPOCHSpath,sta,'_',srCH,'_EPOCHS.mat'];
if exist(fiq)
    load(fiq);
else
    warnmssg=['File in question: ',fiq,' DNE'];
    warning(warnmssg);%(['File in question: ',fiq,' DNE'])
    display(['Try Updating Metadata'])
end

%%
%Identify the Epoch associated with the year/day of the data
year=str2num(yStr);
dyr=str2num(dStr);
dayteNumOfDyr=datenum(year,1,1)+dyr-1;
startBefore=1;
iDedEpoch=0;
i=1;
%starting at the first EPOCH, loop over EPOCH startTime until
%identify the first epoch which starts after the day of
   interest.
%Then subtract 1 from to get correct epoch.
while startBefore
    if startT{i}.datenum>dayteNumOfDyr
        startBefore=0;
        iDedEpoch=i-1;
    else
        i=i+1;
    end
```

```
end
if iDedEpoch==0
    warnmssg=['EPOCH 0 was identified: so no Epoch startTime
       was found to be after ',yStr,' ',dStr];
    warning(warnmssg)
    display(['Using EPOCH 1 in this case'])
    iDedEpoch=1;
end
%%
%B058F04
%unixCmd=['grep Sensitivity ',metadatapath,sta,'_',srCH,'_EPOCH
   ',zeroPadStr(iDedEpoch,2),'.txt | grep B | gawk -F: ''{print
   $2}'''];
%[s w]=unix(unixCmd);
%sf=str2num(w);
%class(w);
%get converison factor
%caution this worked from 02-05, no gaurantees on modern data
   ... need a
%thorugh metadata treatment system
```

```
function epochMeta=associateEpochNumber2(yStr,dStr,sta,CH)
%usage: epochMeta=associateEpochNumber2(yStr,dStr,sta,srCH)
%This function replaces associateEpochNumber() which was based on the
%automated method of bookkeeping metaData.  This function does better than
%associate an epoch number, it will also load most of the pertinent
%metaData as well.
global EPOCHSpath METADATApath metadatapath

sta=strrep(sta,' ','');

fiq=[metadatapath,'/SPREADSHEETS/',sta,'_',CH,'.mat'];
if exist(fiq)
    load(fiq);
else
    warnmssg=['File in question: ',fiq,' DNE'];
    warning(warnmssg);%(['File in question: ',fiq,' DNE'])
    display(['Try Updating Metadata']);
end

%%
%Identify the Epoch associated with the year/day of the data
ydhm=[yStr,dStr,'0000'];
numYDHM=str2num(ydhm);
nEpochs=numel(metaData);
startBefore=1;
iDedEpoch=0;
i=1;
epochStart=zeros(1,nEpochs);
t0str=[metaData{1}.yyyy metaData{1}.dyr metaData{1}.hh metaData{1}.mm];
t0=str2num(t0str);
```

```
31 if numYDHM<t0
32     warnMssg=['The day for which you are trying to process data
           corresponds to a day before the recorded inception of
          the instrument!'];
33     warning(warnMssg)
34 else
35     %we have a time of interest.  Identify the latest epoch
          which starts
36     %BEFORE this time of interest.
37     for i=1:nEpochs
38         clear epochStr
39         epochStr=[metaData{i}.yyyy metaData{i}.dyr metaData{i}.
              hh metaData{i}.mm];
40         epochStart(i)=str2num(epochStr);
41     end
42     priorEpochs=find(epochStart<numYDHM);
43     iDedEpoch=priorEpochs(end);
44 end
45 epochMeta=metaData{iDedEpoch};
```

```
    function beep_vec(soundfile,hz)
    eval(['load  ' soundfile]);
    beepvec = y;
    if nargin == 2
      sound(y,hz)
    else
      sound(y)
    end
return
```

```
function FCRAcat=catFCRA(FCRA)
%usage:FCRAcat=catFCRA(FCRA)
%catFCRA stands for "concatenate FCRA"
%Input: data Structure FCRA havinng one element for each
   section (day, 2hr, etc...)
%OutPut: Data Structure covering all decimation levels and FCs,
%concatenated over all sections, so for example, if the input
   FCRA has 2
%segments, such that the first decimation level has 449 FCs in
   a time
%series, The output will have 898 FCs in the first decmiation
   level.

nSegs=numel(FCRA);
FCRAcat=FCRA{1};
for iSeg=2:nSegs
    for iDec=1:numel(FCRAcat) %loop over decimation levels
        for iBand=1:numel(FCRAcat{iDec})
            FCRAcat{iDec}{iBand}=cat(3,FCRAcat{iDec}{iBand},
               FCRA{iSeg}{iDec}{iBand});
        end
    end
end
```

```
function numArray=chr2num_relDB(SITES,CHANNELS,chrArray)
% a list of all Sites and Channels, together with an array of
%characters specifying selected channels, and generates a UIC
   array: an Nx2
%where the first row is a SiteUID and the second row is a
   channel UID.
%it is the opposite of num2chr_relDB
%chrArray
%chrArray{1}
SITES;
CHANNELS;
UIC=siteChArray(SITES,CHANNELS);
UIC(:,1)=abs(UIC(:,1));%+2- 14mar %sites
UIC(:,2)=-abs(UIC(:,2));    %channels
usedSites=[];
nChMax=numel(chrArray);
for iSC=1:nChMax
    siteCode=chrArray{iSC}.locn;
    iSite=find(strcmp(siteCode,SITES)==1);
    usedSites=[usedSites iSite];
    if numel(iSite)==1
        siteSubArrayIndex=find(abs(UIC(:,1))==iSite);%+2- 14mar
        subArray=UIC(siteSubArrayIndex,:);
        chCode=chrArray{iSC}.chn;
        iChn=find(strcmp(chCode,CHANNELS{iSite})==1);
        if numel(iChn)==1
            subArray(iChn,2)=abs(subArray(iChn,2));
            UIC(siteSubArrayIndex,:)=subArray;
        end
    end
end
%usedSites=unique(usedSites)
%for us=usedSites
%    siteSubArrayIndex=find(abs(UIC(:,1))==iSite);
```

```
33 %    UIC(siteSubArrayIndex,1)=-abs(UIC(siteSubArrayIndex,1));
34 %end
35
36
37 numArray=UIC;
```

```
function []=cleanScratch()
global scrpath
display(['Cleaning Local Scratch Directory'])
%may need to toggle between rm and /bin/rm because typhoon sets
    'rm' to 'rm -i'
cmd=['/bin/rm ',scrpath,'*'];
unix(cmd);
```

```
%set up directory structure:
setPaths;
dataTypes={'TS','FC','SDM','SS'};
SLaSH=loadSLASH;
load([SYSpath,'sr_list'])
d=date;
ks=ksplit(d,'-');
yearNow=str2num(ks{3});
for isr=1:length(sr_list.letters)
    for yyyy=1995:yearNow
        yStr=zeroPadStr(yyyy,4);
        newSYYYYDir=[DATApath,sr_list.letters(isr),yStr];
        if isdir(newSYYYYDir)==0
            cmd=['mkdir ''',newSYYYYDir,''''];
            eval(cmd)
        %else
        %    display(['DIR: ',newSYYYYDir,' must already exist
           '])
        end
        for dT=1:numel(dataTypes)
            newDir=[newSYYYYDir,SLaSH,dataTypes{dT}];
            if isdir(newDir)==0
                cmd=['mkdir ',newDir];
                eval(cmd)
            %else
            %    display(['DIR: ',newDir,' must already exist
               '])
            end
        end
    end
end
metaDataDir=[ULFEMpath,'metaData'];
if isdir(metaDataDir)==0
    cmd=['mkdir ',metaDataDir];
```

```
    eval(cmd)
end
epochsDir=[metaDataDir,SLaSH,'EPOCHS'];
if isdir(epochsDir)==0
    cmd=['mkdir ',epochsDir];
    eval(cmd)
end

plotScrDir=[ULFEMpath,SLaSH,'plotscr'];
if isdir(plotScrDir)==0
    cmd=['mkdir ',plotScrDir];
    eval(cmd)
end

CoilDir=[SYSpath,SLaSH,'COILS'];
if isdir(CoilDir)==0
    cmd=['mkdir ',CoilDir];
    eval(cmd)
end

scrDir=[SCRpath];
if isdir(scrDir)==0
    cmd=['mkdir ',scrDir];
    eval(cmd)
end

figDir=[ULFEMpath,SLaSH,'FIGURES'];
if isdir(figDir)==0
    cmd=['mkdir ',figDir];
    eval(cmd)
end
```

```
function [] = dayPush(hObject, eventdata, handles)

    %clear axes:
    axes(handles.tsax); cla;
    axes(handles.specax); cla;

    %get info from GUI - GUISTATE
    handles=guidata(handles.SpecAvgFig);
    iDy=get(hObject,'UserData')
    nSmooth=str2num(get(handles.smoothEdit,'string'));
    showAvg=get(handles.showAvgCheckBox,'value');
    legendTxt=['legend('];
    if showAvg
        includeInAvg=getIncludeInAvgVec(handles);
        includeInAvg=includeInAvg/sum(includeInAvg)
        avgSpecNow=(includeInAvg*handles.avgSpec).^(0.5);
        axes(handles.specax);
        loglog(handles.daysOfData{iDy}.spec.frqs,medfilt1(
           avgSpecNow,nSmooth),'r');
        legendTxt=[legendTxt,'''AverageSpec'',''];
        hold on
    end

    %PLOT TIME SERIES
    axes(handles.tsax); cla;
    [sr sps]=getSR(handles.cfg{iDy}.SR);
    dt=1.0/sps;
    t=dt*(0:length(handles.daysOfData{iDy}.data)-1);
    plot(t,handles.daysOfData{iDy}.data);
    xlabel('Time [s]', 'fontsize',14)
    ylabel('Machine Counts')
    legend(['Day ',handles.cfg{iDy}.ddd])

    %Plot spectrum
```

```
34      axes(handles.specax);
35      loglog(handles.daysOfData{iDy}.spec.frqs,medfilt1(abs(
           handles.daysOfData{iDy}.spec.spec),nSmooth));
36      xlabel('Frequency [Hz]','fontsize',14)
37      if handles.cfg{1}.fieldType=='ELECTRIC'
38          ylabel('mV/km/sqrt(Hz)')
39      elseif handles.cfg{1}.fieldType=='MAGNETIC'
40          ylabel('nT/sqrt(Hz)')
41      end
42      handles.cfg{iDy}
43      %legendTxt=[legendTxt,'''DaySpec'')']
44      legendTxt=[legendTxt,'''Day ',handles.cfg{iDy}.ddd,''')']
45      eval(legendTxt)
46      guidata(handles.SpecAvgFig,handles);
47
48      end
49
50
51
52  function includeInAvgVec = getIncludeInAvgVec(handles)
53      %returns a vector specifying which days to include in the
           average
54      %calculation
55      nDays=length(handles.daysOfData);
56      includeInAvgVec=zeros(1,nDays);
57      for iDay = 1:length(handles.daysOfData)
58          includeInAvgVec(iDay)=get(handles.CB{iDay},'Value');
59      end
60
61  end
```

```
function sout = ddewhite(s)
%DDEWHITE Double dewhite. Strip both leading and trailing whitespace.
%
%   DDEWHITE(S) removes leading and trailing white space and any null
%   characters from the string S.  A null character is one that has an absolute
%   value of 0.
%
%   See also DEWHITE, DEBLANK, DDEBLANK.

%   Author:      Peter J. Acklam
%   Time-stamp:  2003-10-13 11:12:57 +0200
%   E-mail:      pjacklam@online.no
%   URL:         http://home.online.no/~pjacklam

   error(nargchk(1, 1, nargin));
   if ~ischar(s)
      error('Input must be a string (char array).');
   end

   if isempty(s)
      sout = s;
      return;
   end

   [r, c] = find(~isspace(s));
   if size(s, 1) == 1
      sout = s(min(c) : max(c));
   else
      sout = s(:, min(c) : max(c));
   end
```

```
function radians=deg2rad(degrees)
%takes an input number, vector or array ’degrees’ and returns
   its value in
%radians.
radians=degrees*pi/180;
```

```
%frequency spacing considerations
addpath('/home/kappler/TOOLS/MATLAB/')
N=[32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536
   131072 262144 524288 1048576 2097052 4194104];
dt=[1/40 1/20 1/10 1/5 1/2 1 2 4 8 16 32 64 128 256 512 1024
   2048 4096];
fNyq=1./(2*dt);
fSample=1./dt;
df=1./(N'*dt);
figure;
subplot(2,1,1)
pcolor(log2(N), log2(fSample),log2(df))
ttl=['log_{2} \Delta f']
title(ttl,'fontsize', 15)
xlabel('log_{2}(N samples)')
ylabel('log_{2}(fSample)')
colorbar;
%zlabel('delta F')
subplot(2,1,2)
pcolor(log2(N), log2(dt),log2(df))
xlabel('log_{2}(N samples)')
ylabel('log_{2}(\Deltat)');
colorbar;
%zlabel('delta F')
eps811('freqSpacing')
```

```
function [OK]=getAsciiHeader(X)
%USAGE: getAsciiHeader(X)
%FLOW:
%Use of the 'curl' linux command to read a file from NCEDC
   which
%contains information about the various epochs of the
   instrument specified
%by X

%X is a metadata structure;
%X.SR (sampling rate),
%X.ch (channel name),
%X.site (SITE name),
%X.Network  e.g. BK, BP, etc
%X.loaclStagingDir, local directory for temporary metadata
   storage

global ULFEMpath SCRpath SLASH me metaStageDir ulfemToolbox
%metaStageDir
network=X.Network;%'BK';
tenTwentySites={'BRIB','MHDL','JRSC'};
mainWebpage=['http://www.ncedc.org/ftp/pub/doc/',network,'.info
   /',network,'.responses/RESP.',network,'.'];
display(['Retrieving webpage metadata from NCEDC'])
localStagingDir=X.localStagingDir;
cleanScratch();
remoteCmd=['ssh ',me,' python ',ulfemToolbox,'cleanMetaStage.py
   '];
[s w]=unix(remoteCmd);
L1='import os';
%oneortwodots due to inconsitant file storage conventions at
   BSL.  These
%will hopefully be made uniform one day.
oneortwodots='.';
```

```
if length(X.site)==3
    oneortwodots='..';
elseif X.site=='CCRB'
    oneortwodots='..';
end
display(['Curling Webpage for EPOCH info'])
%%ADD TEMPORARY HANDLING for 0.10 to BRIB MHDL JRSC etc.
L2=['cmd=''curl ',mainWebpage,X.site,oneortwodots,X.SR,X.ch,' >
    ',metaStageDir,X.site,'.',X.SR,X.ch,'.txt''''];
for siteHandle=tenTwentySites
        if regexp(X.site,siteHandle{1})
            if regexp(X.ch,'Q')
                app='.10';
            elseif regexp(X.ch,'T')
                app='.20';
            end
            L2=strrep(L2,['RESP.',network,'.',X.site,'.'],['
               RESP.',network,'.',X.site,app,'.']);
        end
end
L3=['os.system(cmd)'];
fid=fopen('cmd.py','w');
fprintf(fid,'%s\n%s\n%s',L1,L2,L3);
fclose(fid);
display(['Finished writting curl script'])
scpCmd=['scp cmd.py ',me,':',metaStageDir];
[s w]=unix(scpCmd);
remoteCmd=['ssh ',me,' python ',metaStageDir,'cmd.py'];
[s w]=unix(remoteCmd);
scpCmd=['scp ',me,':',metaStageDir,X.site,'.',X.SR,X.ch,'.txt '
   ,localStagingDir];
[s w]=unix(scpCmd);
display(['Finished transferring curled RESP page'])

```

```
%%
display(['Checking local staging directiory to confirm delivery
    of metadata epoch file:'])
a=dir(SCRpath);
if numel(a)==3
    display(['File: ',a(3).name,' confirmed located in local
       staging directory'])
else
    display(['There are too few or too many files in the
       staging directory:' ,localStagingDir])
end

display(['Checking File: ',a(3).name,' for useful content'])
four04Cmd=['grep "404 Not Found" ',localStagingDir,a(3).name];
[s w]=unix(four04Cmd);
if s==0
    mssg=['Sampling Rate: ' X.SR,'=',num2str(X.Hz),'Hz Does not
        appear to be an option for ',X.site,' ',X.ch];
    warning(mssg)
    %display(['Sampling Rate: ' X.SR,'=',num2str(X.Hz),'Hz Does
        not appear to be an option for ',X.site,' ',X.ch])
    OK=0;
else
    OK=1;
    %could add a check in here to see if file is very small or
       blank.
end
```

```
function sf=getESiteScaleFactor(yStr,dStr,sta,srCH)
%INPUT AGRUMENTS:
%yStr: A four-character string denoting year, e.g. '2004'
%dStr: A three-character string denoting julian day  e.g. '272'
%sta: A string denoting station  e.g. 'PKD'
%srCH: A three-charcter string denoting sampling rate and
   channel, e.g. 'LT1'


global EPOCHSpath METADATApath metadatapath
display(['getting site E scale factor'])

%%ID EPOCH: EPOCHSpath not currently supported under
   spreadsheet method of metadata handling
sta=strrep(sta,' ','');
fiq=[EPOCHSpath,sta,'_',srCH,'_EPOCHS.mat'];
if exist(fiq)
    load(fiq);
else
    warnmssg=['File in question: ',fiq,' DNE'];
    warning(warnmssg);%(['File in question: ',fiq,' DNE'])
end

%%
%Identify the Epoch associated with the year/day of the data
year=str2num(yStr);
dyr=str2num(dStr);
dayteNumOfDyr=datenum(year,1,1)+dyr-1;
startBefore=1;
iDedEpoch=0;
i=1;
%starting at the first EPOCH, loop over EPOCH startTime until
%identify the first epoch which starts after the day of
   interest.
```

```
%Then subtract 1 from to get correct epoch.
while startBefore
    if startT{i}.datenum>dayteNumOfDyr
        startBefore=0;
        iDedEpoch=i-1;
    else
        i=i+1;
        if i>numel(startT)
            display(['Date may be after the startT'])
            break
        end
    end
end
if iDedEpoch==0
    warnmssg=['EPOCH 0 was identified: so no Epoch startTime
       was found to be after ',yStr,' ',dStr];
    warning(warnmssg)
    display(['Using EPOCH 1 in this case'])
    iDedEpoch=1;
end
%%
%B058F04
unixCmd=['grep Sensitivity ',metadatapath,sta,'_',srCH,'_EPOCH'
   ,zeroPadStr(iDedEpoch,2),'.txt | grep B | gawk -F: ''{print
   $2}'''];
[s w]=unix(unixCmd);
sf=str2num(w);
class(w);
%get converison factor
%caution this worked from 02-05, no gaurantees on modern data
   ... need a
%thorugh metadata treatment system
```

```
function sf=getESiteScaleFactor_ssht(yStr,dStr,sta,srCH)
%INPUT AGRUMENTS:
%yStr: A four-character string denoting year, e.g. '2004'
%dStr: A three-character string denoting julian day  e.g. '272'
%sta: A string denoting station  e.g. 'PKD'
%srCH: A three-charcter string denoting sampling rate and
   channel, e.g. 'LT1'


global EPOCHSpath METADATApath metadatapath
display(['getting site E scale factor'])
%%ID EPOCH: EPOCHSpath not currently supported under
   spreadsheet method of metadata handling
sta=strrep(sta,' ','');
%%
%Identify the Epoch associated with the year/day of the data
epochMeta=associateEpochNumber2(yStr,dStr,sta,srCH(2:end))
vm2c=str2num(epochMeta.cpv)*str2num(epochMeta.gainV)*str2num(
   epochMeta.length);
sf=1/vm2c;
```

```
function [fName]=getFullAsciiHeader(X)
%USGAE: [fName]=getFullAsciiHeader(X);
%X has fields for Site, CH, sampling rate, network.
%gets a full ascii metadata fiel from BSL
global ULFEMpath scrpath METADATApath SCRpath metadatapath me
   metaStageDir ulfemToolbox
%metaStageDir
tenTwentySites={'BRIB','MHDL','JRSC'};
network=X.Network;%'BK';
display(['Retrieving data from NCEDC'])
remoteStagingDir=metaStageDir
%display(['Cleaning Remote Staging Directory']);
remoteCleanCmd=['ssh ',me,' python ',ulfemToolbox,'
   cleanMetaStage.py'];
[s w]=unix(remoteCleanCmd);
cleanScratch();


%get the binary metadata
%bUILD THE REMOTE COMMMAND to get binary metadata
remoteCmd=['ssh ',me,' bsdata -f ',X.yStr,'.',X.dStr,'.',X.hStr
   ,':00 -o ',remoteStagingDir,X.site,'.',network,...
    '.',X.SR,X.ch,'.',X.yStr,'_',X.dStr,'_',X.hStr,'.bin -s ',X
       .durn,' ',X.site,'.',network,'.',X.SR,X.ch]

for siteHandle=tenTwentySites
    if regexp(X.site,siteHandle{1})
        if regexp(X.ch,'Q')
            app='.10';
        elseif regexp(X.ch,'T')
            app='.20';
        end
        remoteCmd=[remoteCmd,app];
    end
```

```
end
display('Executing the bsdata command to get Full Header for
   current EPOCH')

%remoteCmd
[s w]=unix(remoteCmd)
return%
%%Convert binary metadata file to an ascii file
%Build a script to conver the binary metadata to ascii metadata
    which will
%be executed at BSL
L1='import os';
L2=['cmd=''rdseed -f -s ',remoteStagingDir,X.site,'.',network
   ,...
    '.',X.SR,X.ch,'.',X.yStr,'_',X.dStr,'_',X.hStr,'.bin > ',
       remoteStagingDir,X.site,'.',...
    X.SR,X.ch,'_FULL.txt'''];
L3=['os.system(cmd)'];
fid=fopen('cmd.py','w');
fprintf(fid,'%s\n%s\n%s',L1,L2,L3);
fclose(fid);
scpCmd=['scp cmd.py ',me,':',remoteStagingDir];
[s w]=unix(scpCmd);
remoteCmd=['ssh ',me,' python ',remoteStagingDir,'cmd.py'];
[s w]=unix(remoteCmd);

%count the number of ascii files generated by bin2asc
   conversion. Make sure
%there is exactly 1 file
%display('count numascii files, better be 1')
remoteCmd=['ssh ',me,' ls -1 ',remoteStagingDir,'*.txt | wc -l'
   ];
[s w]=unix(remoteCmd);
%at BSL w can have garbage text appended to the beginning.
```

```
%The work around is to pull off the text lines from the top
if (s==0)
    if numel(w)>0
        numFilesInfo=ksplit(w,'\n');
        %remove lines which have to do with ssh garbage text,
           BSL problem)
        numFilesInfoFlagLines=[];
        FlagLines={'ssh_keysign:', 'key_sign failed', 'Warning:
            No xauth data; using fake'};
        for i=1:numel(numFilesInfo)
            for j=1:numel(FlagLines)
                if strmatch(FlagLines{j},numFilesInfo{i})
                    numFilesInfoFlagLines=[
                       numFilesInfoFlagLines i];
                end
            end
        end
        numFilesInfoFlagLines=unique(numFilesInfoFlagLines);
        numFilesInfo(numFilesInfoFlagLines)=[];
        if numel(numFilesInfo)>1
            mssg=['The wc command still returned with excess
               lines from ssh at BSL'];
            warning(mssg)
        else
            w=numFilesInfo{1};
        end
    else
        mssg=['Process of Counting number of cued ascii
           metadata files at BSL has failed somehow'];
        warning(mssg);
    end
end

if str2num(w)==1
```

```
88      %bring the metadata here
89      fName=[X.site,'_',X.SR,X.ch,'_EPOCH',zeroPadStr(X.epch,2),'
           .txt '];
90      scpCmd=['scp ',me,':',remoteStagingDir,X.site,'.',X.SR,X.ch
           ,'_FULL.txt ',scrpath,fName];
91      [s w]=unix(scpCmd);
92  else
93      mssg=['There is ',num2str(w), ' file(s) of header info per
           epoch.  I was expecting only 1.'];
94      warning(mssg);
95  end
96  %finally copy metadata from scratch to metaDataDirectory
97  cpCmd=['cp ',scrpath,fName,' ',metadatapath];
98  [s w]=unix(cpCmd);
99  if s==0
100     display([fName,' has been successfully generated'])
101 end
```

```
function [SR HZ] = getSR(srIndex)
%returns the sampling rate letter and sps
    global SYSpath
    srFile = fullfile(SYSpath, 'sr_list')
    load(srFile)
    if srIndex==1
        SR='ALL';
        HZ='ALL';
    elseif isnumeric(srIndex)
        SR=sr_list.letters(srIndex-1);
        HZ=sr_list.numbers(srIndex-1);
    else
        %probably we have a case where we have input a letter
        SR=srIndex;
        ndx=find(sr_list.letters==srIndex); %find which one is
           L, or B
        HZ=sr_list.numbers(ndx);
    end
end
```

```
function [L startT endT]= idEpochs(X)
%USAGE: idEpochs(X)
%X is a metadata structure;
%X.SR (sampling rate),
%X.ch (channel name),
%X.site (SITE name),
%X.loaclStagingDir, local directory for temporary metadata
   storage
%FLOW:
%Use of the 'grep' command to identify the various EPOCHs of
   the channel
%specified by X.
%each EPOCH has a different metadata file and will require a
   complete metadata download.
display(['Enter idEpochs.m '])
global EPOCHSpath
cmd=['grep -n B052F22 ',X.localStagingDir,X.site,'.',X.SR,X.ch,
   '.txt'];
[s w]=unix(cmd);
L=ksplit(w,'\n');
for nL=1:numel(L)
    tmp=ksplit(L{nL},':  ');
    startT{nL}.textStr=strrep(tmp{2},' ','');
    ydt=ksplit(startT{nL}.textStr,',');
    startT{nL}.year=str2num(ydt{1});
    startT{nL}.ddd=str2num(ydt{2});
    x=datenum(startT{nL}.year,1,1);
    mmdd=datestr(x+startT{nL}.ddd-1,'mm-dd');
    mmdd=ksplit(mmdd,'-');
    startT{nL}.mm=str2num(mmdd{1});
    startT{nL}.dd=str2num(mmdd{2});
    starthms=ksplit(ydt{3},':');
    startT{nL}.h=str2num(starthms{1});
    startT{nL}.m=str2num(starthms{2});
```

```
    startT{nL}.s=str2num(starthms{3});
    startT{nL}.datenum=datenum(startT{nL}.year,startT{nL}.mm,
       startT{nL}.dd,startT{nL}.h,startT{nL}.m,startT{nL}.s);
end
cmd=['grep -n B052F23 ',X.localStagingDir,X.site,'.',X.SR,X.ch,
   '.txt'];
[s w]=unix(cmd);
L=ksplit(w,'\n');
if regexp(L{end},'No Ending Time')
    yr=datestr(now,'yyyy');
    dnn=datenum(now);
    %get ddd by subtracting jan1 from now+1
    ddd=dnn+1-datenum(str2num(yr),1,1);
    L{end}=[':  ',yr,',',num2str(ddd),',00:00:00'];
    display('Last Epoch is valid to present')
end
for nL=1:numel(L)
    tmp=ksplit(L{nL},':  ');
    endT{nL}.textStr=strrep(tmp{2},' ','');
    ydt=ksplit(endT{nL}.textStr,',');
    endT{nL}.year=str2num(ydt{1});
    endT{nL}.ddd=str2num(ydt{2});
    x=datenum(endT{nL}.year,1,1);
    mmdd=datestr(x+endT{nL}.ddd-1,'mm-dd');
    mmdd=ksplit(mmdd,'-');
    endT{nL}.mm=str2num(mmdd{1});
    endT{nL}.dd=str2num(mmdd{2});
    endhms=ksplit(ydt{3},':');
    endT{nL}.h=str2num(endhms{1});
    endT{nL}.m=str2num(endhms{2});
    endT{nL}.s=str2num(endhms{3});
    endT{nL}.datenum=datenum(endT{nL}.year,endT{nL}.mm,endT{nL
       }.dd,endT{nL}.h,endT{nL}.m,endT{nL}.s);
end
```

```
svCmd=['save ',EPOCHSpath,X.site,'_',X.SR,X.ch,'_EPOCHS L
   startT endT'];
eval(svCmd);
%end of idEpochs.m
```

```
function N=idNumExpectedPts(durn,SR)
%{
@type durn: string
@param: durn: the duration of the file
@type SR: string
@param SR: code for sampling rate
Function returns the number of points in expected ts
@note: we will want to change the durn to be a datetime object
   in future.
%}
global SYSpath
load([SYSpath,'sr_list.mat'])
idnum=find(SR==sr_list.letters);
sr=sr_list.numbers(idnum);

%debugging strings
display('Identify Number of points in expected timeSeries')
display(sprintf('durn = %s',durn))
display(sprintf('SR code = %s',SR))
%debugging strings

timeUnit=durn(end);
if regexp(timeUnit,'d')
    ptsPer=86400;
elseif regexp(timeUnit,'H')
    ptsPer=3600;
end
nUnits=str2num(durn(1:end-1));
N=sr*ptsPer*nUnits;
%number of expected points
```

```
function sourceFile=identifySpecAvgFiles(cfgNow)
%Takes as input the cfg data structure from readParseSpecAvgcfg
   .m
%and makes a list of the basic TS files we will need for
   analysis
%OUTPUT FORMAT:
%struct,
%sourceFILE{1}.fName        #file containing the data point
   associated with  t0
%sourceFILE{1}.ddd           #will be day0 for the first file,
   but can change if crossing midnight
%sourceFILE{1}.ndx0         # first point in file to grab
%sourceFILE{1}.ndx1         # last point in file to grab

%sourceFILE{2}.fName        #file containing the data point
   associated with  t0 (_0200_) for example
%sourceFILE{2}.ndx0         # first point in file to grab
%sourceFILE{2}.ndx1         # last point in file to grab
%.
%.
%.
%sourceFILE{N}.fName
%sourceFILE{N}.ndx0         # first point in file to grab
%sourceFILE{N}.ndx1         # last point in file to grab


%FLOW:
%1. Identify file containing the first segment of data. f0
%2. Identify the index of f0 where we start
   minsLeftInCurrentFilereading

%3. Now start iterative loop: Check if last index is in the
   current file
%IF YES... NOTE LAST INDEX; ENTER WHILE LOOP
```

```
%IF NO IDENTIFY NEXT FILE IN CHRONOLOGICAL ORDER
% CHECK IF LAST INDEX IS HERE... LOOPING...

global DATApath
minterval=str2num(cfgNow.minterval);

%% ID 1st FILE
sourceFile{1}.fname=''; %placeholder for first filename
firstFile=[DATApath,cfgNow.SR,cfgNow.yyyy,'/TS/',cfgNow.site,'_
   ',cfgNow.SR,cfgNow.sens,'_',cfgNow.ddd];
if cfgNow.SR=='B'
    %By virtue of 2h file storage, if t0hh is an odd number
       need to subtract 1
    if mod(cfgNow.t0hh,2)==1
        hh0=cfgNow.t0hh-1;
    else
        hh0=cfgNow.t0hh;
    end
    firstFile=[firstFile,'_',zeroPadStr(hh0,2)];
end
sourceFile{1}.fname=[firstFile,'.mat'];     %Now I have
   identified the first file with relevant data.
nFilesSpecified=1;

%% IDENTIFY THE FIRST TIME SAMPLE OF RELEVANCE
%CASES:
%L: 1-day long file
%B: 2hour long file
switch cfgNow.SR
    case 'L'
        display 'its L'
        spf=86400; %samples per file
        spm=60; %samples per minute
        minsPerFile=1440;
```

```
59
60      case 'B'
61          spf=288000;
62          spm=60*40; %samples per minute
63          minsPerFile=120;
64          minutes2t0=0;
65          if mod(cfgNow.t0hh,2)==1
66              minutes2t0=60;
67          end
68  end
69
70  minutes2t0=60*cfgNow.t0hh+cfgNow.t0mm;
71  pctFileToFFWD=minutes2t0/minsPerFile;
72  sample0=round(pctFileToFFWD*spf)+1;
73  sourceFile{1}.ndx0=sample0;
74  %Now have minutes2t0 which is the number of minutes to skip
75  lastIndexInCurrentFile=0; %this is a marker to tell you if you
       need to keep opening more files to cover the "minterval"
76
77  %% CHECK IF LAST INDEX is here?
78  accumulatedMinutes=0;
79  minsLeftInCurrentFile=minsPerFile-minutes2t0;
80  minsNeededFromOtherFiles=minterval-minsLeftInCurrentFile;
81  nFilesNeeded=ceil(minsNeededFromOtherFiles/minsPerFile)+1;
82
83  %if nFilesNeeded is 1, the last index is in the first file.
84  %if its 2 the last index is in the next file
85  %if its greater than 2, all files but the first and last are to
        be fully
86  %loaded.
87  if nFilesNeeded==1
88      sampleEnd=sample0+minterval*spm;     %id last index
89      sourceFile{1}.ndx1=sampleEnd;
90  else
```

```
91      sourceFile{1}.ndx1=spf;
92      yStr=num2str(cfgNow.yyyy);
93      dStr=cfgNow.ddd;
94      hStr=num2str(hh0);
95      accumulatedMinutes=minsLeftInCurrentFile;
96      minsFromFinalFile=rem(minterval-accumulatedMinutes,
           minsPerFile);
97      samplesFromFinalFile=minsFromFinalFile*spm;
98      if minsFromFinalFile==0
99          minsFromFinalFile=minsPerFile; %?needed?
100     end
101 end
102
103
104 while nFilesSpecified<nFilesNeeded
105     [yStr, dStr, hStr] =incrementYDH(yStr,dStr,hStr,cfgNow.SR);
106     nextFile=[dataPath,cfgNow.SR,num2str(yStr),'/TS/',cfgNow.
           site,'_',cfgNow.SR,cfgNow.sens,'_',dStr];
107     if cfgNow.SR=='B'
108         nextFile=[nextFile,'_',zeroPadStr(hStr,2)];
109     end
110     nFilesSpecified=nFilesSpecified+1;
111     sourceFile{nFilesSpecified}.fname=[nextFile,'.mat'];
112     sourceFile{nFilesSpecified}.ndx0=1;
113     if nFilesSpecified==nFilesNeeded
114         %display('the end is near')
115         sourceFile{nFilesSpecified}.ndx1=samplesFromFinalFile;
116     else
117         sourceFile{nFilesSpecified}.ndx1=spf;
118     end
119 end
```

```
function [yyyy, ddd, hh] = incrementYDH(yStr,dStr,hStr,SR)
%{
usage [yStr, dStr, hStr] = function incrementYDH(y,d,h,SR)
@note: handling using datetime needs to be added here;
@rtype : list of strings, y,d,h, padded in standard format

@change: 2013/08/10: Debug off-by-one error in 1-Hz case.
%}
if regexp(class(yStr),'double')
    yStr=num2str(yStr);
end

yyyy=str2num(yStr);
ddd=str2num(dStr);
hh=str2num(hStr);
if regexp(SR,'B')
    display(['40 Hz incrementYDH called'])
    timeInput = datenum(yyyy,1,1)+ddd + datenum(0,0,0,hh,0,0);%
       this is NOW
    nextTime = timeInput + datenum(0,0,0,2,0,0);
    yyyy = datestr(nextTime,'yyyy');
    ddd = floor(nextTime-datenum(['01-Jan-',yyyy]));
    hh = datestr(nextTime,'HH');

elseif regexp(SR,'L')
    %display(['1 Hz data being called'])
    dayNext=datenum(yyyy,1,1)+ddd;
    yyyy=datestr(dayNext,'yyyy');
    ddd=dayNext-datenum(['01-Jan-',yyyy])+1;
end
ddd=zeroPadStr(ddd,3);
hh=zeroPadStr(hh,2);
```

```
sr_list.letters='DBLV';
sr_list.numbers=[500 40.0 1.00 0.10];
svCmd=['save ',syspath,'sr_list sr_list'];
eval(svCmd)
```

```
function l = ksplit(s,expn)

%L=SPLIT(S,D) splits a string S delimited by characters in D.  Meant to

%            work roughly like the PERL split function (but without any

%            regular expression support).  Internally uses STRTOK to do

%            the splitting.  Returns a cell array of strings.

%

%Example:

%    >> split('_', 'this_is___a_/_string/_//')

%    ans =

%        'this'    'is'    'a'    'string'   []

%

%Written by Gerald Dalley (dalleyg@mit.edu), 2004


l = {};
if numel(s)==0
    return
end

```

```
%strip expns from front
firstisexpn=1;
while firstisexpn
    fie=regexp(s(1),expn);
    if fie
        s=s(2:end);
    else
        firstisexpn=0;
    end
end
%strip exps from back
lastisexpn=1;
while lastisexpn
    lie=regexp(s(end),expn);
    if lie
        s=s(1:end-1);
    else
        lastisexpn=0;
    end
end

%so now the string starts and ends without newlines
if length(s)>0
    ndx=regexp(s,expn);
    nChunks=length(ndx)+1;
    strtNdx=[1 ndx+1];
    ndNdx=[ndx-1 length(s)];
    for nC=1:nChunks
        r{nC}=s(strtNdx(nC):ndNdx(nC));
    end
    l=r;
end
```

```
function varargout = lbox2_export(varargin)
% LBOX2_EXPORT Application M-file for lbox2_export.fig
%   LBOX2_EXPORT, by itself, creates a new LBOX2_EXPORT or raises the existing
%   singleton*.
%
%   H = LBOX2_EXPORT returns the handle to a new LBOX2_EXPORT or the handle to
%   the existing singleton*.
%
%   LBOX2_EXPORT('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in LBOX2_EXPORT.M with the given input arguments.
%
%   LBOX2_EXPORT('Property','Value',...) creates a new LBOX2_EXPORT or raises the
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before lbox2_OpeningFunction gets called.  An
%   unrecognized property name or invalid value makes property application
%   stop.  All inputs are passed to lbox2_export_OpeningFcn via varargin.
%
%   *See GUI Options - GUI allows only one instance to run (singleton).
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2000-2006 The MathWorks, Inc.

```

```
% Edit the above text to modify the response to help
   lbox2_export

% Last Modified by GUIDE v2.5 15-May-2010 14:43:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',     gui_Singleton, ...
                   'gui_OpeningFcn',
                     @lbox2_export_OpeningFcn, ...
                   'gui_OutputFcn',     @lbox2_export_OutputFcn
                     , ...
                   'gui_LayoutFcn',  @lbox2_export_LayoutFcn,
                     ...
                   'gui_Callback',      []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin
       {:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before lbox2_export is made visible.
function lbox2_export_OpeningFcn(hObject, eventdata, handles,
   varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to lbox2_export (see VARARGIN)

% Choose default command line output for lbox2_export
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

if nargin == 3,
    initial_dir = pwd;
elseif nargin > 4
    if strcmpi(varargin{1},'dir')
        if exist(varargin{2},'dir')
            initial_dir = varargin{2};
        else
            errordlg('Input argument must be a valid directory'
               ,'Input Argument Error!')
            return
        end
    else
        errordlg('Unrecognized input argument','Input Argument
           Error!');
        return;
    end
end
% Populate the listbox
load_listbox(initial_dir,handles)
% Return figure handle as first output argument

```

```
% UIWAIT makes lbox2_export wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = lbox2_export_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --------------------------------------------------------------------
% Callback for list box - open .fig with guide, otherwise use open
% --------------------------------------------------------------------
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1

```

```
get(handles.figure1,'SelectionType');
if strcmp(get(handles.figure1,'SelectionType'),'open')
    index_selected = get(handles.listbox1,'Value');
    file_list = get(handles.listbox1,'String');
    filename = file_list{index_selected};
    if  handles.is_dir(handles.sorted_index(index_selected))
        cd (filename)
        load_listbox(pwd,handles)
    else
        [path,name,ext] = fileparts(filename);
        switch ext
            case '.fig'
                guide (filename)
            otherwise
                try
                    open(filename)
                catch ex
                    errordlg(...
                      ex.getReport('basic'),'File Type Error','
                         modal')
                end
        end
    end
end
% --------------------------------------------------------------------
% Read the current directory and sort the names
% --------------------------------------------------------------------
function load_listbox(dir_path,handles)
cd (dir_path)
dir_struct = dir(dir_path);
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = sorted_index;
```

```
guidata(handles.figure1,handles)
set(handles.listbox1,'String',handles.file_names,...
    'Value',1)
set(handles.text1,'String',pwd)


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'
       defaultUicontrolBackgroundColor'));
end


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Add the current directory to the path, as the pwd might change thru' the
% gui. Remove the directory from the path when gui is closed
% (See figure1_DeleteFcn)
setappdata(hObject, 'StartPath', pwd);
addpath(pwd);


% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Remove the directory added to the path in the figure1_CreateFcn.
if isappdata(hObject, 'StartPath')
    rmpath(getappdata(hObject, 'StartPath'));
end


% --- Creates and returns a handle to the GUI figure.
function h1 = lbox2_export_LayoutFcn(policy)
% policy - create a new figure or use a singleton. 'new' or 'reuse'.

persistent hsingleton;
if strcmpi(policy, 'reuse') & ishandle(hsingleton)
```

```
193     h1 = hsingleton;
194     return;
195 end
196
197 appdata = [];
198 appdata.GUIDEOptions = struct(...
199     'active_h', [], ...
200     'taginfo', [], ...
201     'override', 1, ...
202     'resize', 'none', ...
203     'accessibility', 'off', ...
204     'mfile', 1, ...
205     'callbacks', 1, ...
206     'singleton', 1, ...
207     'blocking', 0, ...
208     'syscolorfig', 1, ...
209     'lastSavedFile', '/home/kappler/ULFEM/MATLAB/lbox2_export.m
          ', ...
210     'release', 13, ...
211     'lastFilename', '/home/kappler/Desktop/Downloads/
          mathworks_downloads/help/techdoc/creating_guis/examples/
          lbox2.fig');
212 appdata.lastValidTag = 'figure1';
213 appdata.StartPath = '/home/kappler/ULFEM/MATLAB';
214 appdata.GUIDELayoutEditor = [];
215 appdata.initTags = struct(...
216     'handle', [], ...
217     'tag', 'figure1');
218
219 h1 = figure(...
220 'Units','characters',...
221 'Color',[0.701960784313725 0.701960784313725
      0.701960784313725],...
```

```
'Colormap',[0 0 0.5625;0 0 0.625;0 0 0.6875;0 0 0.75;0 0
  0.8125;0 0 0.875;0 0 0.9375;0 0 1;0 0.0625 1;0 0.125 1;0
  0.1875 1;0 0.25 1;0 0.3125 1;0 0.375 1;0 0.4375 1;0 0.5 1;0
  0.5625 1;0 0.625 1;0 0.6875 1;0 0.75 1;0 0.8125 1;0 0.875 1;0
   0.9375 1;0 1 1;0.0625 1 1;0.125 1 0.9375;0.1875 1 0.875;0.25
   1 0.8125;0.3125 1 0.75;0.375 1 0.6875;0.4375 1 0.625;0.5 1
  0.5625;0.5625 1 0.5;0.625 1 0.4375;0.6875 1 0.375;0.75 1
  0.3125;0.8125 1 0.25;0.875 1 0.1875;0.9375 1 0.125;1 1
  0.0625;1 1 0;1 0.9375 0;1 0.875 0;1 0.8125 0;1 0.75 0;1
  0.6875 0;1 0.625 0;1 0.5625 0;1 0.5 0;1 0.4375 0;1 0.375 0;1
  0.3125 0;1 0.25 0;1 0.1875 0;1 0.125 0;1 0.0625 0;1 0
  0;0.9375 0 0;0.875 0 0;0.8125 0 0;0.75 0 0;0.6875 0 0;0.625 0
   0;0.5625 0 0],...
'IntegerHandle','off',...
'InvertHardcopy',get(0,'defaultfigureInvertHardcopy'),...
'MenuBar','none',...
'Name','Directory List',...
'NumberTitle','off',...
'PaperPosition',get(0,'defaultfigurePaperPosition'),...
'Position',[71.8333333333333 54.0714285714286 46
  17.7142857142857],...
'Resize','off',...
'CreateFcn', {@local_CreateFcn, @(hObject,eventdata)
  lbox2_export('figure1_CreateFcn',hObject,eventdata,guidata(
  hObject)), appdata} ,...
'DeleteFcn',@(hObject,eventdata)lbox2_export('figure1_DeleteFcn
  ',hObject,eventdata,guidata(hObject)),...
'HandleVisibility','off',...
'Tag','figure1',...
'UserData',struct(...
    'active_h', 100.005004882812, ...
    'taginfo', struct(...
    'figure', 2, ...
    'listbox', 2, ...
```

```
    'text', 2), ...
    'resize', 'none', ...
    'accessibility', 'off', ...
    'mfile', 1, ...
    'callbacks', 1, ...
    'singleton', 1, ...
    'blocking', 0, ...
    'syscolorfig', 1),...
'Visible','on');

appdata = [];
appdata.lastValidTag = 'listbox1';

h2 = uicontrol(...
'Parent',h1,...
'Units','characters',...
'BackgroundColor',[1 1 1],...
'Callback',@(hObject,eventdata)lbox2_export('listbox1_Callback'
   ,hObject,eventdata,guidata(hObject)),...
'Position',[7 1.76923076923077 31.6 13.2307692307692],...
'String',blanks(0),...
'Style','listbox',...
'Value',1,...
'CreateFcn', {@local_CreateFcn, @(hObject,eventdata)
   lbox2_export('listbox1_CreateFcn',hObject,eventdata,guidata(
   hObject)), appdata} ,...
'Tag','listbox1');

appdata = [];
appdata.lastValidTag = 'text1';

h3 = uicontrol(...
'Parent',h1,...
'Units','characters',...
```

```
'ListboxTop',0,...
'Position',[0.2 15.3076923076923 44.8 2],...
'String','Double Click to Open',...
'Style','text',...
'Tag','text1',...
'CreateFcn', {@local_CreateFcn, blanks(0), appdata} );


hsingleton = h1;


% --- Set application data first then calling the CreateFcn.
function local_CreateFcn(hObject, eventdata, createfcn, appdata
   )

if ~isempty(appdata)
   names = fieldnames(appdata);
   for i=1:length(names)
       name = char(names(i));
       setappdata(hObject, name, getfield(appdata,name));
   end
end

if ~isempty(createfcn)
   if isa(createfcn,'function_handle')
       createfcn(hObject, eventdata);
   else
       eval(createfcn);
   end
end


% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)
```

```
304
305 gui_StateFields =  {'gui_Name'
306     'gui_Singleton'
307     'gui_OpeningFcn'
308     'gui_OutputFcn'
309     'gui_LayoutFcn'
310     'gui_Callback'};
311 gui_Mfile = '';
312 for i=1:length(gui_StateFields)
313     if ~isfield(gui_State, gui_StateFields{i})
314         error('MATLAB:gui_mainfcn:FieldNotFound', 'Could not
            find field %s in the gui_State struct in GUI M-file %
            s', gui_StateFields{i}, gui_Mfile);
315     elseif isequal(gui_StateFields{i}, 'gui_Name')
316         gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
317     end
318 end
319
320 numargin = length(varargin);
321
322 if numargin == 0
323     % LBOX2_EXPORT
324     % create the GUI only if we are not in the process of
        loading it
325     % already
326     gui_Create = true;
327 elseif local_isInvokeActiveXCallback(gui_State, varargin{:})
328     % LBOX2_EXPORT(ACTIVEX,...)
329     vin{1} = gui_State.gui_Name;
330     vin{2} = [get(varargin{1}.Peer, 'Tag'), '_', varargin{end
        }];
331     vin{3} = varargin{1};
332     vin{4} = varargin{end-1};
333     vin{5} = guidata(varargin{1}.Peer);
```

```
334     feval(vin{:});
335     return;
336 elseif local_isInvokeHGCallback(gui_State, varargin{:})
337     % LBOX2_EXPORT('CALLBACK',hObject,eventData,handles,...)
338     gui_Create = false;
339 else
340     % LBOX2_EXPORT(...)
341     % create the GUI and hand varargin to the openingfcn
342     gui_Create = true;
343 end
344 
345 if ~gui_Create
346     % In design time, we need to mark all components possibly
        created in
347     % the coming callback evaluation as non-serializable. This
        way, they
348     % will not be brought into GUIDE and not be saved in the
        figure file
349     % when running/saving the GUI from GUIDE.
350     designEval = false;
351     if (numargin>1 && ishghandle(varargin{2}))
352         fig = varargin{2};
353         while ~isempty(fig) && ~isa(handle(fig),'figure')
354             fig = get(fig,'parent');
355         end
356 
357         designEval = isappdata(0,'CreatingGUIDEFigure') ||
              isprop(fig,'__GUIDEFigure');
358     end
359 
360     if designEval
361         beforeChildren = findall(fig);
362     end
363 
```

```
364     % evaluate the callback now
365     varargin{1} = gui_State.gui_Callback;
366     if nargout
367         [varargout{1:nargout}] = feval(varargin{:});
368     else
369         feval(varargin{:});
370     end
371 
372     % Set serializable of objects created in the above callback
            to off in
373     % design time. Need to check whether figure handle is still
            valid in
374     % case the figure is deleted during the callback
           dispatching.
375     if designEval && ishandle(fig)
376         set(setdiff(findall(fig),beforeChildren), 'Serializable
               ','off');
377     end
378 else
379     if gui_State.gui_Singleton
380         gui_SingletonOpt = 'reuse';
381     else
382         gui_SingletonOpt = 'new';
383     end
384 
385     % Check user passing 'visible' P/V pair first so that its
           value can be
386     % used by oepnfig to prevent flickering
387     gui_Visible = 'auto';
388     gui_VisibleInput = '';
389     for index=1:2:length(varargin)
390         if length(varargin) == index || ~ischar(varargin{index
               })
391             break;
```

```
392         end
393
394         % Recognize 'visible' P/V pair
395         len1 = min(length('visible'),length(varargin{index}));
396         len2 = min(length('off'),length(varargin{index+1}));
397         if ischar(varargin{index+1}) && strncmpi(varargin{index
              },'visible',len1) && len2 > 1
398             if strncmpi(varargin{index+1},'off',len2)
399                 gui_Visible = 'invisible';
400                 gui_VisibleInput = 'off';
401             elseif strncmpi(varargin{index+1},'on',len2)
402                 gui_Visible = 'visible';
403                 gui_VisibleInput = 'on';
404             end
405         end
406     end
407
408     % Open fig file with stored settings.  Note: This executes
          all component
409     % specific CreateFunctions with an empty HANDLES structure.
410
411
412     % Do feval on layout code in m-file if it exists
413     gui_Exported = ~isempty(gui_State.gui_LayoutFcn);
414     % this application data is used to indicate the running
          mode of a GUIDE
415     % GUI to distinguish it from the design mode of the GUI in
          GUIDE. it is
416     % only used by actxproxy at this time.
417     setappdata(0,genvarname(['OpenGuiWhenRunning_', gui_State.
          gui_Name]),1);
418     if gui_Exported
419         gui_hFigure = feval(gui_State.gui_LayoutFcn,
              gui_SingletonOpt);
```

```
420
421         % make figure invisible here so that the visibility of
               figure is
422         % consistent in OpeningFcn in the exported GUI case
423         if isempty(gui_VisibleInput)
424             gui_VisibleInput = get(gui_hFigure,'Visible');
425         end
426         set(gui_hFigure,'Visible','off')
427
428         % openfig (called by local_openfig below) does this for
               guis without
429         % the LayoutFcn. Be sure to do it here so guis show up
               on screen.
430         movegui(gui_hFigure,'onscreen');
431     else
432         gui_hFigure = local_openfig(gui_State.gui_Name,
               gui_SingletonOpt, gui_Visible);
433         % If the figure has InGUIInitialization it was not
               completely created
434         % on the last pass.  Delete this handle and try again.
435         if isappdata(gui_hFigure, 'InGUIInitialization')
436             delete(gui_hFigure);
437             gui_hFigure = local_openfig(gui_State.gui_Name,
                   gui_SingletonOpt, gui_Visible);
438         end
439     end
440     if isappdata(0, genvarname(['OpenGuiWhenRunning_',
           gui_State.gui_Name]))
441         rmappdata(0,genvarname(['OpenGuiWhenRunning_',
               gui_State.gui_Name]));
442     end
443
444     % Set flag to indicate starting GUI initialization
445     setappdata(gui_hFigure,'InGUIInitialization',1);
```

```
446
447     % Fetch GUIDE Application options
448     gui_Options = getappdata(gui_hFigure,'GUIDEOptions');
449     % Singleton setting in the GUI M-file takes priority if
          different
450     gui_Options.singleton = gui_State.gui_Singleton;
451
452     if ~isappdata(gui_hFigure,'GUIOnScreen')
453         % Adjust background color
454         if gui_Options.syscolorfig
455             set(gui_hFigure,'Color', get(0,'
                  DefaultUicontrolBackgroundColor'));
456         end
457
458         % Generate HANDLES structure and store with GUIDATA. If
              there is
459         % user set GUI data already, keep that also.
460         data = guidata(gui_hFigure);
461         handles = guihandles(gui_hFigure);
462         if ~isempty(handles)
463             if isempty(data)
464                 data = handles;
465             else
466                 names = fieldnames(handles);
467                 for k=1:length(names)
468                     data.(char(names(k)))=handles.(char(names(k
                          )));
469                 end
470             end
471         end
472         guidata(gui_hFigure, data);
473     end
474
475     % Apply input P/V pairs other than 'visible'
```

```
476     for index=1:2:length(varargin)
477         if length(varargin) == index || ~ischar(varargin{index
                })
478             break;
479         end
480 
481         len1 = min(length('visible'),length(varargin{index}));
482         if ~strncmpi(varargin{index},'visible',len1)
483             try set(gui_hFigure, varargin{index}, varargin{
                    index+1}), catch break, end
484         end
485     end
486 
487     % If handle visibility is set to 'callback', turn it on
            until finished
488     % with OpeningFcn
489     gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
490     if strcmp(gui_HandleVisibility, 'callback')
491         set(gui_hFigure,'HandleVisibility', 'on');
492     end
493 
494     feval(gui_State.gui_OpeningFcn, gui_hFigure, [], guidata(
            gui_hFigure), varargin{:});
495 
496     if isscalar(gui_hFigure) && ishandle(gui_hFigure)
497         % Handle the default callbacks of predefined toolbar
                tools in this
498         % GUI, if any
499         guidemfile('restoreToolbarToolPredefinedCallback',
                gui_hFigure);
500 
501         % Update handle visibility
502         set(gui_hFigure,'HandleVisibility',
                gui_HandleVisibility);
```

```
503
504         % Call openfig again to pick up the saved visibility or
                apply the
505         % one passed in from the P/V pairs
506         if ~gui_Exported
507             gui_hFigure = local_openfig(gui_State.gui_Name, '
                   reuse',gui_Visible);
508         elseif ~isempty(gui_VisibleInput)
509             set(gui_hFigure,'Visible',gui_VisibleInput);
510         end
511         if strcmpi(get(gui_hFigure, 'Visible'), 'on')
512             figure(gui_hFigure);
513
514             if gui_Options.singleton
515                 setappdata(gui_hFigure,'GUIOnScreen', 1);
516             end
517         end
518
519         % Done with GUI initialization
520         if isappdata(gui_hFigure,'InGUIInitialization')
521             rmappdata(gui_hFigure,'InGUIInitialization');
522         end
523
524         % If handle visibility is set to 'callback', turn it on
                until
525         % finished with OutputFcn
526         gui_HandleVisibility = get(gui_hFigure,'
                HandleVisibility');
527         if strcmp(gui_HandleVisibility, 'callback')
528             set(gui_hFigure,'HandleVisibility', 'on');
529         end
530         gui_Handles = guidata(gui_hFigure);
531     else
532         gui_Handles = [];
```

```
533     end
534
535     if nargout
536         [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn,
                gui_hFigure, [], gui_Handles);
537     else
538         feval(gui_State.gui_OutputFcn, gui_hFigure, [],
              gui_Handles);
539     end
540
541     if isscalar(gui_hFigure) && ishandle(gui_hFigure)
542         set(gui_hFigure,'HandleVisibility',
              gui_HandleVisibility);
543     end
544 end
545
546 function gui_hFigure = local_openfig(name, singleton, visible)
547
548 % openfig with three arguments was new from R13. Try to call
      that first, if
549 % failed, try the old openfig.
550 if nargin('openfig') == 2
551     % OPENFIG did not accept 3rd input argument until R13,
552     % toggle default figure visible to prevent the figure
553     % from showing up too soon.
554     gui_OldDefaultVisible = get(0,'defaultFigureVisible');
555     set(0,'defaultFigureVisible','off');
556     gui_hFigure = openfig(name, singleton);
557     set(0,'defaultFigureVisible',gui_OldDefaultVisible);
558 else
559     gui_hFigure = openfig(name, singleton, visible);
560 end
561
```

```
562 function result = local_isInvokeActiveXCallback(gui_State,
      varargin)
563
564 try
565     result = ispc && iscom(varargin{1}) ...
566              && isequal(varargin{1},gcbo);
567 catch
568     result = false;
569 end
570
571 function result = local_isInvokeHGCallback(gui_State, varargin)
572
573 try
574     fhandle = functions(gui_State.gui_Callback);
575     result = ~isempty(findstr(gui_State.gui_Name,fhandle.file))
          || ...
576              (ischar(varargin{1}) ...
577              && isequal(ishandle(varargin{2}), 1) ...
578              && (~isempty(strfind(varargin{1},[get(varargin{2},
                   'Tag'), '_'])) || ...
579                 ~isempty(strfind(varargin{1}, '_CreateFcn'))) )
                    ;
580 catch
581     result = false;
582 end
```

```
%November 7, 2010; This callback loads and plots arrays of
   Mulitvariate timesereis of FCs.
%SR Sampling Rate
%RA The Array from the 'Select Array' dropdown box which will
   loaded/plotted
%at some point later this plotter should work if you click on
   any .mat file
%right now it is constrained to work with fixed length files.


action = get(gcbo,'Tag');
global ULFEMpath rect0 ARRAYS arrayList DATApath SYSpath SLaSH
   SCRPLOTpath
h0=gcf;
set(h0,'pointer','watch');
SLaSH=loadSLASH();

switch action
    case 'Load'
        clear FCRA
        display(['Loading FC Files'])
        handles=guidata(h0);
        [SR HZ] = getSR(get(handles.SRpopup,'Value'));
        arrayNameIndex=get(handles.ArraySelPopup,'Value');
        RA=ARRAYS{arrayNameIndex};
        YYYYstart=str2num(get(handles.YYstartedit,'String'));
        MMstart=str2num(get(handles.MMstartedit,'String'));
        DDstart=str2num(get(handles.DDstartedit,'String'));
        dayNumStart=datenum(YYYYstart,MMstart,DDstart);
        dyr=dayNumStart-datenum(YYYYstart,1,1)+1;
        yStr=num2str(YYYYstart)
        dStr=zeroPadStr(num2str(dyr),3);
        nSegments=str2num(get(handles.nSegEdit,'String'));

```

```
        if SR=='B'
            HHstart=str2num(get(handles.HRstartedit,'String'))
            durn='2H';
        else
            HHstart=0;  %HHend=00;
            durn='1d';
        end

        hStr=zeroPadStr(HHstart,2);
        clear SITECH
        SITECH=RA.chrArray;
        nCH=numel(SITECH);
        HHlist=HHstart;

        %loop over the number of sections to load (usually days
            of 1Hz)
        for segment=1:nSegments
            segmentSpecifier.yStr=yStr;
            segmentSpecifier.dStr=dStr;
            segmentSpecifier.hStr=hStr;
            segmentSpecifier.SR=SR;
            FCRA{segment}=readFCRA(RA,segmentSpecifier);
            [yStr dStr hStr]=incrementYDH(yStr,dStr,hStr,SR);
        end
        hStr=zeroPadStr(HHstart,2);
        %NOW HAVE FCRA (Arrays of FCs)
        % Concatenate Them
        FCRAcat=catFCRA(FCRA);
        clear FCRA;
        FCRA=FCRAcat;
        clear FCRAcat;
        %handles.FCRA=FCRAcat;
```

```
63          loadedName=[RA.name,'!',yStr,'!',zeroPadStr(dyr,3),'!',
               SR,'!',num2str(nSegments),'nSeg'];
64          if regexp(SR,'B')
65              loadedName=[RA.name,'!',yStr,'!',zeroPadStr(dyr,3),
                   '!',hStr,'!',SR,'!',num2str(nSegments),'nSeg'];
66          end
67          loadedList=get(handles.LOADEDARRAYSlistbox,'String');
68          loadedList{numel(loadedList)+1}=loadedName;
69          set(handles.LOADEDARRAYSlistbox,'string',loadedList);
70          cmd=['save ',SCRPLOTpath,SLaSH,'FC',SLaSH,loadedName,'
               FCRA SITECH']
71          eval(cmd)
72          handles=guidata(h0);
73
74      case 'Plot'
75          %FLOW
76          %LOAD THE FCRA
77          %LOOPING OVER ALL CHANNELS;
78          %separate the channel from FCRA
79          %PLOTSPECTROGRAM of THE CHANNEL
80          %waitforbuttonpress
81          %end
82          display(['Plotting SPECTROGRAMS'])
83          if regexp(SR,'L')
84              load ../sys/bFC_L.mat
85          elseif regexp(SR,'B')
86              load ../sys/bFC_B.mat
87          end
88          handles=guidata(h0);
89          value=get(handles.LOADEDARRAYSlistbox,'Value');
90          strng=get(handles.LOADEDARRAYSlistbox,'String');
91          fNameToPlot=strng{value};
92          %load the data and SITECH
```

```
%data, FCRA have 449 points per day, and 1499 points per 2Hr
%section
cmd=['load ',SCRPLOTpath,SLaSH,'FC',SLaSH,fNameToPlot];
eval(cmd);
splt=regexp(fNameToPlot,'!','split')
SITE=splt{1}; YEAR=splt{2}; DYR=splt{3}; SR=splt{4};

splt=splt{end};
splt=regexp(splt,'nSeg','split');
nSegments=str2num(splt{1}); % geregexp(SR,'B')
%matfName=[matfName,'_',hStr]; %this from the fNametoPlt

%%% 20 January, 2011: FCRA plotting
nCHANNELS=numel(SITECH); % or size(FCRA{1}{1},1);
for iCH=1:nCHANNELS
    chID=[SITECH{iCH}.locn,' ',SR,SITECH{iCH}.chn];
    SITECH{iCH}.chn
    if SITECH{iCH}.chn(1)=='T'
        SIunits='nT';
    elseif SITECH{iCH}.chn(1)=='Q'
        SIunits='mV/km';
    else
        SIunits='???';
    end

    dayRange=[YEAR,' day ',DYR];
    if nSegments>1
        lastDay=str2num(DYR)+nSegments-1;
        dayRange=[dayRange,'--',num2str(lastDay)];
    end

    clear plotFCs plotPERIODS;
```

```
125 nBand=numel(FCRA{1});
126 plotFCs{1}=[];
127 plotPERIODS{1}=[];
128 figure
129 for iBand=1:nBand
130     clear newFCs
131     %concatenate along freq/period axis
132     plotPERIODS{1}=[plotPERIODS{1} bFC{1}{iBand}.
          periods'];
133     newFCs=squeeze(FCRA{1}{iBand}(iCH,:,:))';
134     if size(newFCs,1)==1, newFCs=newFCs'; end %
          squeeze can transpose soemtimes
135     plotFCs{1}=[plotFCs{1} newFCs]
136 end
137
138 iDec=2;
139 nBand=numel(FCRA{iDec});
140 %rather than initiate with empty arrays, initiate
      with a period
141 %corrsponding to the last one plotted so that there
      are no gaps in SPCGRM
142 %For the 1-2 transition of 1Hz, thats band 11...
      note this will
143 %be a hassle to customize for each Sampling Rate!
      :(
144 newFCs=squeeze(FCRA{iDec}{11}(iCH,3,:))';
145 if size(newFCs,1)==1, newFCs=newFCs'; end %squeeze
      can transpose soemtimes
146 plotFCs{iDec}=newFCs;
147 plotPERIODS{iDec}=bFC{iDec}{11}.periods(3);
148 for iBand=12:nBand
149     plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
          iBand}.periods'];
150     newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
```

```
151             if size(newFCs,1)==1, newFCs=newFCs'; end %
                    squeeze can transpose soemtimes
152             plotFCs{iDec}=[plotFCs{iDec} newFCs];
153         end
154
155         iDec=3;
156         plotFCs{iDec}=[];
157         plotPERIODS{iDec}=[];
158         nBand=numel(FCRA{iDec});
159         for iBand=12:nBand
160             plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
                    iBand}.periods'];
161             newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
162             if size(newFCs,1)==1, newFCs=newFCs'; end %
                    squeeze can transpose soemtimes
163             plotFCs{iDec}=[plotFCs{iDec} newFCs];
164         end
165
166         iDec=4;
167         nBand=numel(FCRA{iDec});
168         plotFCs{iDec}=[];%FC{iB}{11}(2:3,:)';
169         plotPERIODS{iDec}=[];%=bFC{iB}{11}.periods(2:3)';
170         for iBand=12:nBand
171             plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
                    iBand}.periods'];
172             newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
173             if size(newFCs,1)==1, newFCs=newFCs'; end %
                    squeeze can transpose soemtimes
174             plotFCs{iDec}=[plotFCs{iDec} newFCs];
175         end
176
177         iDec=5;
178         nBand=numel(FCRA{iDec});
179         newFCs=squeeze(FCRA{iDec}{12}(iCH,2,:))';
```

```
180             if size(newFCs,1)==1, newFCs=newFCs'; end %squeeze
                   can transpose soemtimes
181             plotFCs{iDec}=newFCs;
182             plotPERIODS{iDec}=bFC{iDec}{12}.periods(2)';
183             for iBand=13:nBand
184                 plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
                       iBand}.periods'];
185                 newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
186                 if size(newFCs,1)==1, newFCs=newFCs'; end %
                       squeeze can transpose soemtimes
187                 plotFCs{iDec}=[plotFCs{iDec} newFCs];
188             end
189
190
191             iDec=6;
192             nBand=numel(FCRA{iDec});
193             newFCs=squeeze(FCRA{iDec}{11}(iCH,2,:))';
194             if size(newFCs,1)==1, newFCs=newFCs'; end %squeeze
                   can transpose soemtimes
195             plotFCs{iDec}=newFCs;
196             plotPERIODS{iDec}=bFC{iDec}{11}.periods(2)';
197             for iBand=12:nBand
198                 plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
                       iBand}.periods'];
199                 newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
200                 if size(newFCs,1)==1, newFCs=newFCs'; end %
                       squeeze can transpose soemtimes
201                 plotFCs{iDec}=[plotFCs{iDec} newFCs];
202             end
203
204             iDec=7;
205             nBand=numel(FCRA{iDec});
206             newFCs=squeeze(FCRA{iDec}{11}(iCH,2,:))';
```

```
207             if size(newFCs,1)==1, newFCs=newFCs'; end %squeeze
                   can transpose soemtimes
208             plotFCs{iDec}=newFCs;
209             plotPERIODS{iDec}=bFC{iDec}{11}.periods(2)';
210             for iBand=12:nBand
211                 plotPERIODS{iDec}=[plotPERIODS{iDec} bFC{iDec}{
                       iBand}.periods'];
212                 newFCs=squeeze(FCRA{iDec}{iBand}(iCH,:,:))';
213                 if size(newFCs,1)==1, newFCs=newFCs'; end %
                       squeeze can transpose soemtimes
214                 plotFCs{iDec}=[plotFCs{iDec} newFCs];
215             end
216
217             % iB=7;
218             % nBand=numel(FC{iB});
219             % plotFCs{iB}=FC{iB}{11}(2,:)';
220             % plotPERIODS{iB}=bFC{iB}{11}.periods(2)';
221             % for iBand=12:nBand
222             %     plotPERIODS{iB}=[plotPERIODS{iB} bFC{iB}{
                   iBand}.periods'];
223             %     plotFCs{iB}=[plotFCs{iB} FC{iB}{iBand}'];
224             % end
225             nDec=iDec;
226
227
228             %nB=iB;
229
230             logyscale=1;
231             if logyscale
232                 for iDec=1:nDec
233                     plotPERIODS{iDec}=log10(plotPERIODS{iDec});
234                 end
235             end
236             yL=[plotPERIODS{1}(1) plotPERIODS{nDec}(end)];
```

```
237             widths=[449 224 111 55 27 13 6];
238             t=linspace(0,nSegments*24,nSegments*widths(1));
239             s1=surface(t,plotPERIODS{1},log10(abs(plotFCs{1}'))
                   ,'EdgeColor','none');
240             axis tight;
241             view(0,90);
242             hold on
243             for iDec=2:nDec
244                 t=linspace(0,nSegments*24,nSegments*widths(iDec
                       ));
245                 s2=surface(t,plotPERIODS{iDec},log10(abs(
                       plotFCs{iDec}')),'EdgeColor','none');
246                 axis tight;
247             end
248             axis tight
249             ylim(yL);
250             ylabel(['Log_{10} Period [s]'],'fontsize',16)
251             xlabel('UT Hour','fontsize',16)
252             title([chID, ' ',dayRange],'fontsize',15);
253             cbar=colorbar;
254             set(get(cbar,'Xlabel'),'string',['$$\frac{',SIunits
                   ,'}{sqrt(Hz)}$$'],'interpreter','latex'); %$$\
                   sqrt{}$$ bug in Matlab USGS
255             %unitText=text(cbarPosn(1),0.8*cbarPosn(2),'$$nt \
                   sqrt{Hz}$$','units', 'normalized','interpreter','
                   latex');%,'parent',3);
256
257         end
258 end
259 set(h0,'pointer','arrow')
```

```
%SR Sampling Rate
%RA The Array from the 'Select Array' dropdown box which will
   loaded/plotted
%at some point later this plotter should work if you click on
   any .mat file
%right now it is constrained to wrk with fixed length files.
%@type ii: vector
%@var ii: Array of indices which are not NaN for a particular
   data channel

action = get(gcbo,'Tag');
global ULFEMpath rect0 ARRAYS arrayList DATApath SYSpath SLaSH
   SCRPLOTpath
h0=gcf;
set(h0,'pointer','watch');

switch action
    case 'Load'
        display(['Loading TS'])
        handles=guidata(h0);
        [SR HZ] = getSR(get(handles.SRpopup,'Value'));
        arrayNameIndex=get(handles.ArraySelPopup,'Value');
        RA=ARRAYS{arrayNameIndex};
        YYYYstart=str2num(get(handles.YYstartedit,'String'));
        MMstart=str2num(get(handles.MMstartedit,'String'));
        DDstart=str2num(get(handles.DDstartedit,'String'));
        dayNumStart=datenum(YYYYstart,MMstart,DDstart);
        nSections=str2num(get(handles.nSegEdit,'String'));
        if SR=='B'
            HHMMStartStr = get(handles.HRstartedit,'String');
            HHMMsplit = split(':',HHMMStartStr)
            HHstart=str2num(HHMMsplit{1});
            durn='2H'
        else
```

```
        HHstart=0;  HHend=00;
        durn='1d';
    end
    clear SITECH
    SITECH=RA.chrArray;
    nCH=numel(SITECH);
    OUTVARS=['Y0=',num2str(YYYYstart),';  M0=',num2str(
       MMstart),'; D0=',num2str(DDstart),...
        '; H0=',num2str(HHstart),'; SR=',SR];
    HHlist=HHstart;
    %make dummy NaN array; NAME IT AS
       ARRAY_YYYYstart_dyr_SR,NUMSECTIONS
    ptsPerSeg=idNumExpectedPts(durn,SR);
    nT=nSections*ptsPerSeg;
    yStr=num2str(YYYYstart); yNum=str2num(yStr);
    dyr=dayNumStart-datenum(YYYYstart,1,1)+1;
    dStr=zeroPadStr(num2str(dyr),3);
    hStr=zeroPadStr(HHstart,2);

    arrayName=[RA.name,'_',yStr,'_',dStr,'_',hStr,'_',SR,'_
       ',num2str(nSections)];
    cmd=[arrayName,'=nan(nCH,nT);'];
    eval(cmd);

    %loop over the number of sections to load (usually days
        of 1Hz)
    for section=1:nSections
        sctnNdx=((section-1)*ptsPerSeg)+(1:ptsPerSeg);
        for sc=1:numel(SITECH)
            siteID=SITECH{sc}.locn;

            ch=SITECH{sc}.chn;
            display(['Calling get Ascii: ',yStr,' ',dStr,'
               ',hStr,' ',durn, ' ',siteID,' ',SR,ch]);
```

```
60              fiq=fullfile(DATApath,[SR,yStr],'TS',[siteID,'_
                   ',SR,ch,'_',dStr]);
61              if SR=='B'
62                  fiq=[fiq,'_',hStr];
63                  %@note: July9, 2013: Handling for 40Hz
                       adding
64              end
65              if exist([fiq,'.mat'])
66                  loadCmd=['load ',fiq];
67                  eval(loadCmd);
68              else
69                  display(['File in Question: ',fiq,' does
                       not exist'])
70                  return
71              end
72              data=y.data;
73              cmd=[arrayName,'(sc,sctnNdx)=data;'];
74              eval(cmd);
75
76          end
77          %ESTABLISH yStr, dStr, hStr based on initial y,d,h
               and SR
78          display('Incrementing YDH')
79          [yStr dStr hStr]=incrementYDH(yStr,dStr,hStr,SR);
80      end
81      display(['arrayName ',arrayName])
82      cmd=['y=',arrayName,';'];
83      eval(cmd)
84      loadedName=[strrep(arrayName,'_','!'),'nSeg'];
85      loadedList=get(handles.LOADEDMVTSlistbox,'String');
86      loadedList{numel(loadedList)+1}=loadedName;
87      set(handles.LOADEDMVTSlistbox,'string',loadedList);
88      cmd=['save ',fullfile(SCRPLOTpath,loadedName),' y'];
89      eval(cmd)
```

```
90          handles=guidata(h0);
91
92  %%
93  case 'Plot'
94          display(['Plotting TS'])
95          handles=guidata(h0);
96
97
98  %       %Identify the array to plot (from list of all arrays)
99          TSd1.year=get(handles.YYstartedit,'String');
100         value=get(handles.LOADEDMVTSlistbox,'Value');
101         strng=get(handles.LOADEDMVTSlistbox,'String');
102         fNameToPlot=strng{value};
103         tmp=ksplit(fNameToPlot,'!');
104         arrayName=tmp{1};
105         TSd1.year=tmp{2};
106         TSd1.dyr=tmp{3};
107         TSd1.h0=tmp{4};
108         TSd1.SR=tmp{5};
109
110
111         %Matlab datenum function counts days since (0,0,0),
               Each day has
112         %value 1.0
113         TSd1.timeZero=datenum(str2num(TSd1.year),0,0,str2num(
               TSd1.h0),0,0)+str2num(TSd1.dyr); %used for x-axis
               ticklabels
114         display(['timeZero=',num2str(datevec(TSd1.timeZero))])
115         %if SR=='B'
116         %    HHMMStartStr = get(handles.HRstartedit,'String');
117         %    HHMMsplit = split(':',HHMMStartStr)
118         %    HHstart=str2num(HHMMsplit{1});
119         %    display "HHSTART"
120         %    TSd1.timeZero = TSd1.timeZero + (HHstart/24);
```

```
121         %end
122         %time zero needs to be shifted up for 40Hz data
123 
124         %THIS SHOULD BE DONE WITH A PYTHON DICTIONARY
125         arrayNameIndex=0;
126         for i=1:numel(arrayList)
127             if strmatch(arrayName,arrayList{i})
128                 %display('MATCH')
129                 arrayNameIndex=i;
130                 break
131             end
132         end
133         RA=ARRAYS{arrayNameIndex};
134         %END PYTHON DICT 1-liner
135 
136 %       %load the actual data from temporary SCR file
137         cmd=['load ',fullfile(SCRPLOTpath,fNameToPlot)];
138         eval(cmd)
139         [nCH nT]=size(y);
140 
141 %       %set up the metadata
142 %in particular, get stationName, channel id for each inst in
    time series
143         sta = blanks(nCH*4);
144         sta = reshape(sta,nCH,4)
145         ch_id=sta(:,1:2)
146         for k = 1:nCH
147             k
148             l1 = length(RA.chrArray{k}.locn)
149             sta(k,1:l1) = RA.chrArray{k}.locn
150             ch_id(k,1:2)= RA.chrArray{k}.chn
151         end
152         TSd1.sta=sta;
153         TSd1.ch_id=ch_id;
```

```
154         [SR HZ]=getSR(SR);
155         TSd1.dt=1.0/HZ;
156 %default plotting time units.  tFac is independant of sampling
    rate!
157         plotStart.tUnits='Minutes';
158         plotStart.tFac=60;
159 %       %t0 specifies  a shift to the data; t0Eq0 specifies if
    the shift is applied, 1-NO, 0-yes
160         TSd1.t0=0;
161         plotStart.nT=nT;
162         plotStart.t0Eq0=0;
163         plotStart.nPw=3600; %nPw: num points per window
164         for ich=1:nCH
165             ii=find(isnan(y(ich,:))==0);
166             ybar(ich) = mean(y(ich,ii));
167             ySD(ich) = std(y(ich,ii));
168             if ySD(ich)==0,ySD(ich)=1;end
169             Units{ich}='counts';
170         end
171         MX = ybar+2*ySD; %dim 12
172         MN = ybar-2*ySD; %dim 12
173         TSd1.range  = [MN',MX'];%,scales]; % dim 12x2
174         TSd1.SysTF.PhysU=0;
175         TSd1.nch=nCH;
176         TSd1.npts=nT;
177         plotStart.i0=1;
178         plotStart.pctOverlap=10;
179         plotStart.V=0.01*plotStart.pctOverlap*plotStart.nPw;
180         %plotStart.V=360;
181         plotStart.di=1; %decimation
182         plotStart.T12=[];
183         plotStart.yFac=1;
184         plotStart.centered=1;
185         plotStart.page=1;
```

```
186         fNameSubInfo=ksplit(fNameToPlot,'!');
187         [hTSw,TSw1] = plotTSwin(TSd1,rect0,plotStart);
188         TSd1.Units=Units;
189         TSd1.mVkm=0;
190 
191         set(hTSw,'Name',fNameToPlot);
192         i0 = TSw1.i0; i1 = TSw1.i1; di = TSw1.di;
193         yFac = TSw1.yFac;
194         ddt = TSd1.dt/TSw1.tFac;
195         t0 = (TSd1.t0)*(1-TSw1.t0Eq0)/TSw1.tFac;
196         I = [i0:di:i1];
197         [TSw1] = plotPage(ddt*I+t0,yFac*y(:,I),TSw1,TSd1,1);
198         clear handles
199         handles.y=y;
200         handles.TSd=TSd1;
201         handles.TSw=TSw1;
202         handles.TSc={};
203         setappdata(hTSw,'FigData',handles);
204 end
205 
206 set(h0,'pointer','arrow')
```

```
function SLASH=loadSLASH()

SLASH='/';
if regexp(computer,'PCWIN')
    SLASH='\';
end
```

```
%MkWM makw Window Matrix(TS, L, V, [PL PR])
%this function chops the time series (TS) into a set of windows to be operated
%on separately.  Each window has length L, and the overlap between windows
%is V.  Optional argument [PL PR] is how many NaNs to pad the left and
%right ends of the TS with.  IF left empty or set to 0 no padding is
%applied, if set to 1 we pad with L on the left and $\ref{Appendix ULFReport}$
%on the right
function WM =  mkWM(TS,L,V,varargin)
%TS=mkColVec(TS);

%STEP 1 DETERMINE AMOUNT OF PADDING TO APPLY
if nargin==4  %have specified pads
    %case 1: 0 no padding
    if varargin{1}==0
        warning 'no padding applied; TS must be a multiple of window length'
    elseif varargin{1}==1 & (length(varargin{1})==1) % Case 2: efficient padding
        N=length(TS);
        k=floor(((N+L-1)/(L-V))+1);
        LHS=NaN(L,1);
        RHS=NaN((L+(k-1)*(L-V))-(N+L),1);
        TS=[LHS;TS;RHS];

    else
        LHS=NaN(varargin{1}(1),1);
        RHS=NaN(varargin{1}(2),1);
        TS=[LHS;TS;RHS];
    end;
```

```
28 else
29     %default is no padding (later add just enough NaN so matrix
           is Well
30     %Defined
31     %N=length(TS);
32     %k=floor(((N-1)/(L-V))+1);
33     %RHS=NaN((L+(k-1)*(L-V))-N,1);
34     N=length(TS);
35     k=floor(N/(L-V));
36     RHS=NaN((L+(k-1)*(L-V))-N,1);
37     TS=[TS;RHS];
38 end;
39
40
41 WM=NaN(k,L);
42 for w=1:k
43     WM(w,:)=TS(1+(w-1)*(L-V):L+(w-1)*(L-V));
44     L+(w-1)*(L-V);
45 end;
```

```
function [y_out]=mkshort(y_in);
%This function returns the yLimits of the plots.
%Seems to take array of type double containing endpoints and do
%some rounding off to give axes bounds integer values, and then
%make sure axes limits are increasing.
nd=ndims(y_in);
n=size(y_in);
n1=prod(size(y_in));
%remove NaNs
for k=1:n1
    if isnan(y_in(k))
        if mod(k,2)
            y_in(k)=-1;
        else
            y_in(k)=1;
        end
    end
end

y1=reshape(y_in,1,n1);
%
for k=1:length(y1)
    c=num2str(y1(k));
    minus='';if c(1)=='-',minus='-';c=c(2:end);end
    ie=findstr(c,'e');
    ce=[];
    if isempty(ie)==0,
        ce=c(ie:end);
        c=c(1:ie-1);
    end
    L=length(c);
    idot=findstr(c,'.');
    if isempty(idot),idot=L+1;c=[c,'.'];L=L+1;end
    if idot<=2, % *.****
```

```
        ct=c';ct=ct(idot+1:end);
        c2=str2num(ct);
        i0=find(c2~=0);
        if length(ct)>0
            c1=[c(1:2) ct(1:i0(1))'];
        else
        c1=c(1:2);
        end
    else
        ns=min(idot-1,2); % non-zero digits
        c1=c(1:ns);
        for l=ns+1:idot-1,c1=[c1 '0'];end
    end
    c1=[minus,c1,ce];
    y2(k)=str2num(c1);
end
y_out=reshape(y2,n);

%make sure the returned list is increasing
if nd>1
    if n1>1
        %display(['Diff y_out'])
        d=diff(y_out');
        for i=1:length(d)
            if d(i)==0
                y_out(i,1)=y_out(i,1)-1;
            end
        end
    end
end
return
```

```
function chrArray=num2chr_relDB(numArray, SITES,CHANNELS,
   NETWORKS)
%takes as input a Nx2 array where the first row is a SiteUID
   and the
%second row is a channel UID.
%Converts to a list of Sites and Channels called chrArray

UIC=numArray;
chrArray=[];
ctr=1;
for ndx=1:size(UIC,1)
    if (UIC(ndx,1)>0) & (UIC(ndx,2)>0)
        siteNum=UIC(ndx,1);
        chNum=UIC(ndx,2);
        chrArray{ctr}.locn=SITES{siteNum};
        chrArray{ctr}.chn=CHANNELS{siteNum}{chNum};
        chrArray{ctr}.net=NETWORKS{siteNum};
        ctr=ctr+1;
    end
end
```

```
function [] =pdfPlot(svfName,varargin)
global FIGUREpath
%usage:pdfPlot('filehandle');
%saves gcf as ULFEMpath/FIGURES/svfName.pdf
%usage: save811('x','orient','landscape','figNum','2')
%svfName, e.g. save811('/home/oper/ux060blahblah');
%saves gcf as /home/oper/ux060kjsf.eps
%vararging looks like:
%'figNum','4','orient','portrait','pdf','1'
figureHandle='gcf';
lv=length(varargin);
paperPosn=[0 0 8.5 11];
ornt='landscape';
if lv > 0
    for i=1:lv
        if regexp('figNum',varargin{i})
            varargin{i+1}
            class(varargin{i+1})
            cmd=['figure(',varargin{i+1},');']
            eval(cmd)
        end;
    end
    for i=1:lv
        if regexp('orient',varargin{i})
            ornt=varargin{i+1};
        end
    end
end

    set(gcf, 'PaperPositionMode','manual');
    set(gcf, 'PaperUnits','inches');
    set(gcf, 'PaperPosition',paperPosn);
    orient(ornt);

```

```
35      cmd=['print -dpdf ',fullfile(FIGUREpath,[svfName,'.pdf'])
36      eval(cmd);
```

```
function [TSo] = plotPage(t,data,TSw,TSd,newplt)
%   plotPage  plots a page of data in current window;
%
%   Usage: [TSo] = plotPage(t,data,TSw,TSd,newplt)
%
%    TSw is data structure containing needed parameters,
   handles, etc.
%    data is data to plot , t is time  for x axis, TDw is
   structure containing
%    data array properties, newplt = 1 to initialinze, 0
   thereafter
%the t variable is the x(t) axes on the plot

%plot Page does Two major things: Creates Axes, Plots Data

%Create Axes: if newplt set to 1 a new set of axes are created,
    and if
%CASE 1: newplt: if set to 1 a new set of axes are created
%CASE 2: not a New plot but diff number of channels plotted

display(['Entering plotPage.m'])
plotParams;
E_clr = [.9 0 0 ]; H_clr = [0 .9 0 ];
YL = [TSw.YLch(:,1)-TSw.YLch(:,2) TSw.YLch(:,2)+TSw.YLch(:,1)];
YL=mkshort(YL);
nchplt = sum(TSw.chPlt);
xi=plotpos(1);lxi=plotpos(3);
yi=plotpos(2);
dy=0.005;
lyi = (plotpos(4)-dy*nchplt)/nchplt;
xl = [min(t),max(t)];
fontFac = sum(TSw.chPlt)/10;

%Check if new axes are needed:
```

```
if newplt
    for ic = TSd.nch:-1:1
        if TSw.chPlt(ic)
            TSw.ACh(ic) = axes('Position',[xi yi lxi lyi]);
%           scale buttons for individual channels
            SChPosM = [xi+lxi,yi,.03,lyi/2];psc1=[xi+lxi+0.033,
               yi,.065,lyi/2];
            SChPosP = [xi+lxi,yi+lyi/2,.03,lyi/2];psc2=[xi+lxi
               +0.033,yi+lyi/2,.065,lyi/2];
            yi = yi + lyi + dy;
            TSw.SChM(ic) = uicontrol(gcf,'Style','Pushbutton'
               ,...
             'String',' -  ','FontWeight','bold','Units','
               normalized',...
             'Position',SChPosM,'FontUnits','normalized','
               FontSize',.6*fontFac,...
             'UserData',ic,'Tag','zoomOutCh','BackgroundColor',
               bgc,...
             'ForegroundColor',fgc(2,:),'Callback','plotTScbk')
               ;
%     set min for plotting channel ic
            TSw.mCH(ic) = uicontrol(gcf,'Style','edit','String'
               ,num2str(YL(ic,1)),...
             'Units','normalized','Position',psc1,'Tag','
               set_mCH',...
             'BackgroundColor',bgc2,'FontSize',8,...
             'FontWeight','bold','ForegroundColor',fgc(2,:),...
             'UserData',ic,'Callback','plotTScbk');

            TSw.SChP(ic) = uicontrol(gcf,'Style','Pushbutton'
               ,...
             'String',' +  ','FontWeight','bold','Units','
               normalized',...
             'FontUnits','normalized','FontSize',.6*fontFac,...
```

```
54              'Position',SChPosP,'UserData',ic,'Tag','zoomInCh'
                   ,...
55              'BackgroundColor',bgc,'ForegroundColor',fgc(2,:),'
                   Callback','plotTScbk');
56
57  %     set max for plotting channel ic
58              TSw.MCH(ic) = uicontrol(gcf,'Style','edit','String'
                   ,num2str(YL(ic,2)),...
59              'Units','normalized','Position',psc2,'Tag','
                   set_MCH',...
60              'UserData',ic,'BackgroundColor',bgc2,'FontSize'
                   ,8,...
61              'FontWeight','bold','ForegroundColor',fgc(2,:),'
                   Callback','plotTScbk');
62          end
63      end
64  elseif max(TSw.chPlt ~= TSw.chPltOld)
65      %CASE: THERE ARE A DIFFERENT # OF TS
66      display(['New Number of TS displayed'])
67      for ic = TSd.nch:-1:1
68          if TSw.chPltOld(ic)
69              delete(TSw.ACh(ic));
70              delete(TSw.SChM(ic));
71              delete(TSw.SChP(ic));
72              set(TSw.mCH(ic),'visible','off');
73              set(TSw.MCH(ic),'visible','off');
74           end
75           if TSw.chPlt(ic)
76              TSw.ACh(ic) = axes('Position',[xi yi lxi lyi]);
77
78              %           scale buttons for individual channels
79              SChPosM = [xi+lxi,yi,.03,lyi/2];psc1=[xi+lxi+0.033,
                   yi,.065,lyi/2];
```

```
80              SChPosP = [xi+lxi,yi+lyi/2,.03,lyi/2];psc2=[xi+lxi
                   +0.033,yi+lyi/2,.065,lyi/2];
81              yi = yi + lyi + dy;
82              TSw.SChM(ic) = uicontrol(gcf,'Style','Pushbutton','
                   String',' -  ',...
83               'Units','normalized','FontWeight','bold','
                    FontUnits','normalized',...
84               'FontSize',.6*fontFac,'Position',SChPosM,'UserData
                    ',ic,'Tag','zoomOutCh',...
85               'BackgroundColor',bgc,'ForegroundColor',fgc(2,:),'
                    Callback','plotTScbk');
86
87  %     set min for plotting channel ic
88              TSw.mCH(ic) = uicontrol(gcf,'Style','edit','String'
                   ,num2str(YL(ic,1)),...
89               'Units','normalized','Position',psc1,'Tag','
                    set_mCH',...
90               'BackgroundColor',bgc2,'FontSize',8,'FontWeight','
                    bold',...
91               'ForegroundColor',fgc(2,:),'UserData',ic,'visible'
                    ,'on','Callback','plotTScbk');
92
93              TSw.SChP(ic) = uicontrol(gcf,'Style','Pushbutton','
                   String',' +  ',...
94               'Units','normalized','FontWeight','bold','
                    FontUnits','normalized',...
95               'FontSize',.6*fontFac,'Position',SChPosP,'UserData
                    ',ic,'Tag','zoomInCh',...
96               'BackgroundColor',bgc,'ForegroundColor',fgc(2,:),'
                    Callback','plotTScbk');
97
98  %     set max for plotting channel ic
99              TSw.MCH(ic) = uicontrol(gcf,'Style','edit','String'
                   ,num2str(YL(ic,2)),...
```

```
100                 'Units','normalized','Position',psc2,'Tag','
                       set_MCH',...
101                 'UserData',ic,'visible','on','BackgroundColor',
                       bgc2,'FontSize',8,...
102                 'FontWeight','bold','ForegroundColor',fgc(2,:),'
                       Callback','plotTScbk');
103             end
104         end
105     end
106
107
108     nchMax = max(find(TSw.chPlt));
109     if newplt | max( TSw.chPlt ~= TSw.chPltOld )
110         for ic = TSd.nch:-1:1
111             if TSw.chPlt(ic)
112                 axes(TSw.ACh(ic));
113                 dbar = 0.;
114                 if TSw.centered
115                     inan=find(isnan(data(ic,:))==0);
116                     if isempty(inan)==0
117                         dbar = mean(data(ic,inan));
118                     end
119                     %HERE IS WHERE YOU COULD ADD THE DETREND OPTION
120
121                 end
122                 TSw.PCh(ic) = plot(t,data(ic,:)-dbar);
123                 if upper(TSd.ch_id(ic,1)) == 'E'
124                     clr = E_clr;
125                 else
126                     clr = H_clr;
127                 end
128                 set(TSw.ACh(ic),'fontweight','demi','FontUnits','
                       normalized',...
129                     'FontSize',.3*fontFac,'ytick',[]);
```

```
130             set(TSw.PCh(ic),'Color',clr);
131             yl = TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
132             if isnan(yl(1))
133                 yl=[-1 1];end
134             text(.1,.3,TSd.sta(ic,:),'units','normalized','
                   Fontweight','demi', ...
135                 'FontUnits','normalized','Parent',TSw.ACh(ic),'
                      FontSize',.3*fontFac);
136
137             TSw.unitText(ic)=text(0.05,0.1,['',TSd.Units{ic},''
                   ],'units','normalized',...
138                 'FontUnits','normalized','Parent',TSw.ACh(ic),'
                      FontSize',.2*fontFac,'rotation',90);
139             set(TSw.ACh(ic),'ylabel',text('String',TSd.ch_id(ic
                   ,:)));
140             ylH = get(TSw.ACh(ic),'ylabel');
141             set(ylH,'String',[TSd.ch_id(ic,:)],'Fontweight','
                   demi', ...
142                 'FontUnits','normalized','FontSize',.25*fontFac
                      );
143             if (ic ~= nchMax)
144                 set(gca,'XtickLabelMode','manual','XtickLabel',
                      ' ');
145             end
146         end
147     end
148 else
149     display(['Replotting Same number of Channels in Existing
           Window'])
150     for ic = TSd.nch:-1:1
151         if TSw.chPlt(ic)
152             dbar = 0.;
153             if TSw.centered
154                 inan=find(isnan(data(ic,:))==0);
```

```
155             if isempty(inan)==0
156                 dbar = mean(data(ic,inan));
157             end
158         end
159         if TSw.PCh(ic)~=0,
160             set(TSw.PCh(ic),'Xdata',t,'Ydata',data(ic,:)-
                    dbar);
161         else
162             axes(TSw.ACh(ic));
163             TSw.PCh(ic) = plot(t,data(ic,:)-dbar);
164         end
165         set(TSw.ACh(ic),'fontweight','demi',...
166             'FontUnits','normalized','FontSize',.3*fontFac)
                    ;
167         a=TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
168         if isnan(a)
169             a=[-1 1];
170         end
171         yl=a;
172         text(.1,.3,TSd.sta(ic,:),'units','normalized','
                Fontweight','demi', ...
173             'FontUnits','normalized','Parent',TSw.ACh(ic),'
                    FontSize',.3*fontFac);
174         set(TSw.unitText(ic),'Visible','off');
175         TSw.unitText(ic)=text(0.05,0.1,[TSd.Units{ic}],'
                units','normalized',...
176             'FontUnits','normalized','Parent',TSw.ACh(ic),'
                    FontSize',.2*fontFac,'rotation',90);
177         set(TSw.ACh(ic),'ylim',yl);
178         ylH = get(TSw.ACh(ic),'ylabel');
179         set(ylH,'String',TSd.ch_id(ic,:),'Fontweight','demi
                ', ...
180             'FontUnits','normalized','FontSize',.30*fontFac
                    );
```

```
            set(TSw.ACh(ic),'xlim',xl);
            if upper(TSd.ch_id(ic,1)) == 'E'
                clr = E_clr;
                display 'ch_id(1)'
                TSd.ch_id(ic,1)
            else
                clr = H_clr;
            end
            set(TSw.PCh(ic),'Color',clr);
        end
    end
end

%set xlabel on lowermost plot: e.g. 'Minutes', 'seconds', etc.
set(get(TSw.ACh(nchMax),'xlabel'),'String',TSw.tUnits,...
    'FontWeight','demi','FontUnits','normalized',...
    'FontSize',.25*fontFac);

%%UT TIME AXIS STUFF
xTickLabels=get(TSw.ACh(nchMax),'xticklabel'); %XtickLabels to
   convert for UT
set(TSw.ACh(nchMax),'xMinorTick','off');
xTicks=get(TSw.ACh(nchMax),'xtick');
for i=1:length(xTicks)
    if TSw.tUnits=='Minutes'
        ds{i}=datestr(TSd.timeZero+datenum(0,0,0,floor(xTicks(i
           )/60),rem(xTicks(i),60),0));
    elseif TSw.tUnits=='Seconds'
        ds{i}=datestr(TSd.timeZero+datenum(0,0,0,floor(xTicks(i
           )/3600),rem(xTicks(i),3600),0));
    elseif TSw.tUnits=='Hours'
        ds{i}=datestr(TSd.timeZero+datenum(0,0,0,(xTicks(i))
           ,0,0));
    elseif TSw.tUnits=='Days'
```

```
211         ds{i}=datestr(TSd.timeZero+datenum(0,0,(xTicks(i))
               ,0,0,0));
212     end
213 end
214 set(TSw.ACh(nchMax),'xticklabel',ds);%,'FontSize',18)
215 set(TSw.ACh(nchMax),'fontweight','demi','FontUnits','normalized
    ',...
216                 'FontSize',.11*fontFac,'ytick',[]);
217 %%%%%
218
219
220 %%%
221 ich = min(find(TSw.chPlt));
222 xl = get(TSw.ACh(ich),'xlim');
223 get(TSw.ACh(ich),'xticklabel')
224 xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
225 axes(TSw.hMask);
226 plot(TSw.T12',ones(size(TSw.T12')),'LineWidth',6,'Color','k');
227 %set(gca,'xlim',xl,'Xtick',[],'Ytick',[])
228
229 TSw.chPltOld = TSw.chPlt;
230 TSo = TSw;
231 return
```

```
%  colors for control buttons
   fgc = [0.6 0.6 0.7 ; 0 0 0 ];
   bgc = [.9 .9 1.] ;
   bgc2 = [.7,.7,.9];
   ebc=[1 1 1]; % edit box color

%   FRAME FOR PLOTTING CONTROL
    xframe = 0.3;
    yframe = 1.0;
    ixf    = 0.;
    iyf    = 0.;

% TS plots position
    spleft=0.03;spright=0.1;
    plotpos=[ xframe+spleft 0.07 1-xframe-spleft-spright 0.88];
```

```
function FC= plotTS2Spec(TS, epochMeta,CH, prewhit, apodWindow,
    dT, fieldType,fName,nSmooth)

%If E, B, V? figure typoe of TF
global COILpath
PREWHT=0;ordr=5;
SAVE=0;
N=length(TS);
C2V=str2num(epochMeta.cpv)
yV=TS/C2V;%/(sqrt(2));

%correct electric channels for EFSC gains, and normalize by
%1e6/trodelength to get mV/km TS
if regexp(fieldType,'ELECTRIC')
    gainV=str2num(epochMeta.gainV);
    trodeLength=str2num(epochMeta.length);
    yV=(1e6)*yV/(gainV*trodeLength);
end

if regexp(apodWindow,'HAMMING')
    wndw=hamming_ulfem(N);
    %wndw=ones(N,1);
end
%hamm=ones(1,N)';
nanCheck=sum(sum(isnan(yV)));
if nanCheck>0
    display(['There are still nans in this TSD data! DOI=',
       num2str(doi)])
    return
end
if size(yV) ~= size(wndw)
    wndw=wndw';
end
yV=yV.*wndw;
```

```
yV=yV-mean(yV);
save yV yV
if regexp(fieldType,'MAGNETIC')
        %USE regexpi to load coil file
        if regexp(epochMeta.coilID,'unknown')
            display(['Unknown Coil ID, using 9519'])
            coilID='bf4-9519'
        else
            coilID=epochMeta.coilID;
        end
        [COILpath,coilID,'.rsp']
        bf4=textread([COILpath,coilID,'.rsp'],'','headerlines'
           ,6);
end
dt=dT;  %decimation-corrected sampling rate
    df=1/(N*dt);
    freqs=0:df:(N-1)*df;
    %now decimate by appropriate amount of times:
    %now generate an instrument transfer function for each coil
       :
    if regexp(fieldType,'MAGNETIC')
        clear phs
        phs=interp1(bf4(:,1),exp(j*deg2rad(bf4(:,3))),freqs);
        TF=interp1(bf4(:,1),bf4(:,2),freqs).*phs;
        yUnits='nT/sqrt(Hz)'
    else
        TF=ones(1,N);
        yUnits='mV/km/sqrt(Hz)';
    end
    if PREWHT
        %Window and then PW or PW then window?
        for wdw=1:size(WM,2)
            sig=WM(:,wdw)';
            [B,A]=prony(sig,ordr,ordr);
```

```
        end
        WM=WM.*hammtx';
        sig2=conv(sig,(A));
        sig2=sig2(ceil(ordr/2):end-ceil(ordr/2));
        ft2=fft(sig2);
        zpA=[zeros(1,length(ft2)-ordr-1) A];
        fA=fft(zpA);
        spc=ft2./fA;
        sclSpc=sqrt((2*dt)/(1.36*N))*spc/(0.54);
        sclSpc=sclSpc./squeeze(ITF(ch,1,:))';
    else
        %NHC
        %H=fft(yV);
        %H=H./TF';
        %PS=((16*dt)/(3*N))*(H.*conj(H));
        %sclSpc=PS.^(0.5);
        fWM=sqrt((2*dt)/(1.36*N))*fft(yV)/(0.54);
        sclSpc=fWM./TF';
    end
    figure(4);clf(4);
    spc=sclSpc(1:end/2);
    frqs=freqs(1:end/2);
    loglog(frqs,medfilt1_ulfem(abs(spc),nSmooth),'bx')
    %loglog(frqs,(abs(spc)),'bx')
    dataTextPosn=regexp(fName, 'data')
    ttl=strrep(fName(dataTextPosn:end),'_',' ');
    title(ttl,'fontsize',15)
    ylabel(yUnits,'fontsize',13)
    xlabel('Frequency [Hz]','fontsize',13);
    grid('on')
    if regexp(fieldType,'MAGNETIC')
        xlim([1e-3 100]);
        ylim([1e-7 1])
    end
```

```
99 end
```

```
1
2  %TSw metadata: i0, i1 denote the start and end indices of the
       current plot
3  %TSw.chPlt: vector of 0s and 1s denoting whether a channel is
       plotted or not
4  %TSw.di is a decimation level, defined in laod_plotCB, it
       defines the
5  %spacing between timeseries indices
6  %TSw.ACh: a vector of the uids for Axes; get(TSw.ACH(i)) gives
       you
7  %properties of ith axes
8  %TSw.PCh: a vector of the uids for plots; i.e PCh(i)=plot(
       channel i);
9  %TSw.YLch: somethign to do with Ylimits, but uncentered?
10 %TSw.mCH: Lower limit of axes for each plot in edit box
11 %TSw.MCH: Upper limit of axes for each plot in edit box
12 %TSw gets called in plotPage.m, plotTScbk.m, plotTSwin.m,
       rePltCh.m
13
14 display(['Calling plotTScbk.m'])
15 comp=computer;
16 plotParams;
17 xi=plotpos(1);lxi=plotpos(3);yi=plotpos(2);lyi=plotpos(4);%xi =
       0.4; lxi = 0.55; yi = 0.1;lyi = 0.79;
18 sfac = 1.5;
19 action = get(gcbo,'Tag')
20 Ifig = gcf;
21 clear handles
22
23 handles=getappdata(Ifig,'FigData');
24 TSw=handles.TSw;
25 TSc=handles.TSc; %empty vbl
26 TSd=handles.TSd;
27 %get(TSw.MCH(1),'String')
```

```
%set(TSw.mCH(1),'String','CHANGED')
%before doing the action, add a loop here which looks at the
   current values
%in all the edit boxes and checks if the 'user-typed' values
   from the GUI match the
%'stored' values from TSw
%this allows the user to update more than one field at a time
%start loop update values
%end loop update values

y=handles.y;
nch=TSd.nch;
npts = TSd.npts;

i0 = TSw.i0;
di = TSw.di;
i1 = TSw.i1;
t0 = TSd.t0*(1-TSw.t0Eq0)/TSw.tFac;
dt = TSd.dt/TSw.tFac;
nPw = TSw.nPw;
yFac = TSw.yFac;
V=TSw.V;%number of overlappiung points
switch action
    case 'chnumpt'
        display(['case chnumpt'])
        %check pctOverlap
        get(gco,'String')
        %change number of points plotted in window
        if (length(str2num(get(gco,'String'))) == 0)
            set(gco,'String',num2str(nPw));
        else
            nPw = str2num(get(gco,'String'));
            nPw = min(nPw,npts);
            set(gco,'String',num2str(nPw));
```

```
60              TSw.nPw = nPw;
61              TSw.page = 1;
62              TSw.i0 = 1;
63              if (nPw-V)==0
64                  TSw.npage=ceil(npts/nPw);
65              elseif V==0
66                  TSw.npage=ceil(npts/nPw);
67              else
68                  TSw.npage=floor(((npts+nPw-1)/(nPw-V))+1);
69              end
70              i0=1+(TSw.page-1)*(nPw-V)
71              i1 = min(i0+nPw-1,npts);
72              %i1 = min(i0+nPw-1+V,npts);
73              TSw.i1 = i1;
74              xl = dt*[i0 i1]+t0;
75              for ic = nch:-1:1
76                  if TSw.chPlt(ic)
77                      set(TSw.ACh(ic),'xlim',xl);
78                      rePltCh
79                  end
80              end
81              ss = sprintf('page#\n %d/%d',TSw.page,TSw.npage);
82              set(TSw.PAGE,'String',ss);
83              xl = get(TSw.ACh(1),'xlim');
84              xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
85              axes(TSw.hMask);
86              T12 = TSw.T12;
87              plot(T12',ones(size(T12')),'LineWidth',6,'Color','k
                    ');
88              set(gca,'xlim',xl,'Xtick',[],'Ytick',[]);
89          end
90
91      case 'Centered'
92          %  Center or uncenter range on y axis
```

```
93          TSw.centered=1-TSw.centered; centered=TSw.centered;
94          if centered==1,set(gcbo,'checked','on');else set(gcbo,'
               checked','off');end
95          for ic = 1:nch
96              TSw.YLch(ic,1) = (1-TSw.centered) * mean(TSd.range(
                   ic,1:2));
97              if(TSd.SysTF.PhysU)
98                  TSw.YLch(ic,1) = TSw.YLch(ic,1)*TSd.range(ic,3)
                       ;
99              end
100             a=TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
101             yl = mkshort(a);
102             if TSw.chPlt(ic)
103                 set(TSw.mCH(ic),'String',num2str(yl(1)));
104                 set(TSw.MCH(ic),'String',num2str(yl(2)));
105             end
106             if TSw.chPlt(ic)
107                 set(TSw.ACh(ic),'ylim',yl);
108                 rePltCh
109             end
110         end
111
112     case 'replot'
113         %  replot page
114         rePltPage;
115
116     case 'reset'
117         %  reset plotting ranges to default
118         for ic = 1:nch
119             TSw.YLch(ic,1) = (1-TSw.centered) * mean(TSd.range(
                   ic,1:2));
120             TSw.YLch(ic,2) = (TSd.range(ic,2)-TSd.range(ic,1))
                   /2;
121             if(TSd.SysTF.PhysU)
```

```
122                 TSw.YLch(ic,2) = (TSw.YLch(ic,2))*TSd.range(ic
                       ,3);
123             end
124             if TSw.chPlt(ic)
125                 a=TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
126                 yl = mkshort(a);%TSw.YLch(ic,1)+[-1,1]*TSw.YLch
                       (ic,2));
127                 set(TSw.mCH(ic),'String',num2str(yl(1)));
128                 set(TSw.MCH(ic),'String',num2str(yl(2)));
129                 set(TSw.ACh(ic),'ylim',yl);
130             end
131         end
132
133     case 'zoomOut'
134         %  Zoom out for all channels simultaneously
135         %  Increase range by factor sfac
136         for ic = 1:nch
137             TSw.YLch(ic,2) = mkshort(TSw.YLch(ic,2)*sfac);
138             if TSw.chPlt(ic)
139                 yl = TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
140                 if isnan(yl);yl=[-1 1];end
141                 set(TSw.mCH(ic),'String',num2str(yl(1)));
142                 set(TSw.MCH(ic),'String',num2str(yl(2)));
143                 set(TSw.ACh(ic),'ylim',yl);
144             end
145         end
146
147     case 'zoomIn'
148         %  Zoom in for all channels simultaneously
149         %  Decrease plotting range by factor sfac
150         for ic = 1:nch
151             TSw.YLch(ic,2)/sfac;
152             TSw.YLch(ic,2) = mkshort(TSw.YLch(ic,2)/sfac);
153             if TSw.chPlt(ic)
```

```
154                 yl = TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
155                 if isnan(yl);yl=[-1 1];end
156                 set(TSw.mCH(ic),'String',num2str(yl(1)));
157                 set(TSw.MCH(ic),'String',num2str(yl(2)));
158                 set(TSw.ACh(ic),'ylim',yl);
159             end
160         end
161
162     case 'zoomOutCh'
163         %  Zoom out for one channel
164         %  Increase plotting range by factor sfac
165         ic=get(gco,'UserData');
166         TSw.YLch(ic,2) = TSw.YLch(ic,2)*sfac;
167         tmp= TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
168         yl = mkshort(tmp);
169         set(TSw.mCH(ic),'String',num2str(yl(1)));
170         set(TSw.MCH(ic),'String',num2str(yl(2)));
171         set(TSw.ACh(ic),'ylim',yl);
172
173     case 'zoomInCh'
174         %  Zoom out for one channel
175         %  Decreas plotting range by factor sfac
176         ic=get(gco,'UserData');
177         TSw.YLch(ic,2) = TSw.YLch(ic,2)/sfac;
178         tmp=TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2);
179         yl = mkshort(tmp);
180         set(TSw.mCH(ic),'String',num2str(yl(1)));
181         set(TSw.MCH(ic),'String',num2str(yl(2)));
182         set(TSw.ACh(ic),'ylim',yl);
183
184     case 'chonoff'
185         %  Activate/deactivate a data channel for plotting
186         ic=get(gco,'UserData');
187         TSw.chPlt(ic)=1-TSw.chPlt(ic);
```

```
188         set(gco,'ForeGroundColor',fgc(1+TSw.chPlt(ic),:));
189         %TSw.YLch(ic,1) = (1-TSw.centered) * mean(TSd.range(ic
               ,1:2));
190
191         if TSw.chPlt(ic)
192             yl = mkshort(TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2));
193             set(TSw.mCH(ic),'String',num2str(yl(1)));
194             set(TSw.MCH(ic),'String',num2str(yl(2)));
195         end
196         rePltPage
197
198     case 'set_mCH'
199         %  change plotting range lower limit for one channel by
                editing mCH string
200         h_ob = gco;
201         ic=get(h_ob,'UserData');
202         yl = mkshort(TSw{ii}.YLch(ic,1)+[-1,1]*TSw{ii}.YLch(ic
               ,2));
203         if (length(str2num(get(h_ob,'String'))) == 0)
204             set(h_ob,'String',num2str(yl(1)));
205         else
206             newMin = str2num(get(h_ob,'String'));
207             newMin =mkshort(newMin); %round(newMin*10)/10;
208             newMin = min(newMin,-1);
209             if TSw.centered
210                 yl = mkshort([-abs(newMin), abs(newMin)]);
211                 TSw{ii}.YLch(ic,2) = abs(newMin);
212                 set(TSw{ii}.MCH(ic),'String',yl(2));
213                 set(TSw{ii}.mCH(ic),'String',yl(1));
214             else
215                 yl(1) = newMin;
216                 TSw.YLch(ic,1) = mean(yl);
217                 TSw.YLch(ic,2) = (yl(2)-yl(1))/2;
218                 set(TSw.MCH(ic),'String',yl(2));
```

```
219                 set(TSw.mCH(ic),'String',yl(1));
220             end
221             if TSw.chPlt(ic)
222                 set(TSw.ACh(ic),'ylim',yl);
223             end
224         end
225
226     case 'set_MCH'
227         %  change plotting range lower limit for one channel by
                editing mCH string
228         %relevant variables: TSw, ii
229         display(['Variable ii index has length : ',num2str(
               length(ii))])
230
231         h_ob = gco;
232         ic=get(h_ob,'UserData');
233
234         yl = mkshort(TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2));
235         if (length(str2num(get(h_ob,'String'))) == 0)
236             set(h_ob,'String',num2str(yl(2)));
237         else
238             newMax = str2num(get(h_ob,'String'));
239             newMax = mkshort(newMax);%round(newMax*10)/10;
240             newMax=max(newMax,1);
241             if TSw.centered
242                 yl = mkshort([-abs(newMax), abs(newMax)]);
243                 TSw.YLch(ic,2) = abs(newMax);
244                 set(TSw.MCH(ic),'String',yl(2));
245                 set(TSw.mCH(ic),'String',yl(1));
246             else
247                 yl(2) = newMax;
248                 TSw.YLch(ic,1) = mean(yl);
249                 TSw.YLch(ic,2) = (yl(2)-yl(1))/2;
250                 set(TSw.MCH(ic),'String',yl(2));
```

```
                set(TSw.mCH(ic),'String',yl(1));
            end
            if TSw.chPlt(ic)
                set(TSw.ACh(ic),'ylim',yl);
            end
        end

    case 'PrevPage'
        %  plot previous page
        TSw.page=TSw.page-1;
        if (TSw.page <= 0)
            TSw.page = TSw.page + 1;
            %beep_vec('splat');
            beep_vec('gong');
        end
        i0=1+(TSw.page-1)*(nPw-V);
        i1=nPw+(TSw.page-1)*(nPw-V)+V;
        %       %i0 = nPw*(TSw.page-1) + 1;i1 = i0+nPw-1;
        if(i1 > npts)
            i1 = npts;
            %           i0 = i1-nPw+1-V;
        end
        t = [i0:di:i1]*dt+t0;
        TSw = plotPage(t,yFac*y(:,[i0:di:i1]),TSw,TSd,0);
        ss = sprintf('page#\n %d/%d',TSw.page,TSw.npage);
        set(TSw.PAGE,'String',ss);
        TSw.i0 = i0;
        TSw.i1 = i1;
        %UpdateCohPlot = 1;CohCB

    case 'NextPage'
        %  plot next page
        TSw.page = TSw.page + 1;
        if (TSw.page > TSw.npage)
```

```
            TSw.page = TSw.page - 1;
            beep_vec('splat');
        end
        i0=1+(TSw.page-1)*(nPw-V);
        i1=i0+nPw;
        if(i1 > npts)
            i1 = npts;
            i0 = i1-nPw+1-V;
        end
        t = [i0:di:i1]*dt+t0;
        TSw = plotPage(t,yFac*y(:,[i0:di:i1]),TSw,TSd,0);
        ss = sprintf('page#\n %d/%d',TSw.page,TSw.npage);
        set(TSw.PAGE,'String',ss);
        TSw.i0 = i0;
        TSw.i1 = i1;

    case 'gotoPage'
        %  go to page # typed in page indicator
        if (length(str2num(get(gco,'String'))) == 0)
            set(gco,'String',num2str(TSw.page));
        else
            newpage = str2num(get(gco,'String'));
            if (newpage > TSw.npage) | (newpage < 1)
                beep_vec('laughter');
                set(gco,'String',num2str(TSw.page));
            else
                TSw.page = newpage;
                i0 = nPw*(TSw.page-1) + 1;
                i1 = i0+nPw-1;
                if(i1 > npts)
                    i1 = npts;
                    i0 = i1-nPw+1;
                end
                t = [i0:di:i1]*dt+t0;
```

```
319                 TSw = plotPage(t,yFac*y(:,[i0:di:i1]),TSw,TSd
                       ,0);
320                 ss = sprintf('page#\n %d/%d',TSw.page,TSw.npage
                       );
321                 set(TSw.PAGE,'String',ss);
322                 TSw.i0 = i0;
323                 TSw.i1 = i1;
324                 %UpdateCohPlot = 1;CohCB
325             end
326         end
327
328     case 'Quit'
329         %  quit this window
330         delete(Ifig)
331
332     case 't0'
333         h = gcbo;
334         if(get(h,'UserData') == 0)
335             set(h,'checked','on','UserData',1)
336             TSw.t0Eq0 = 1;
337         else
338             set(h,'checked','off','UserData',0)
339             TSw.t0Eq0 = 0;
340         end
341         rePltPage
342
343     case 'tUnits'
344         h = gcbo;
345         set(findobj('Tag','tUnits'),'checked','off');
346         set(h,'checked','on');
347         TSw.tUnits = get(h,'Label');
348         TSw.tFac = get(h,'UserData');
349         rePltPage
350
```

```
351     case 'yUnits'
352         h = gcbo;
353         set(findobj('Tag','yUnits'),'checked','off');
354         set(h,'checked','on');
355         TSw.yFac = 1./get(h,'UserData');
356         for ic = 1:TSd.nch
357             YLch(ic,1) = (1-TSw.centered) * mean(TSd.range(ic
                ,1:2));
358             YLch(ic,2) = (TSd.range(ic,2)-TSd.range(ic,1))/2;
359             if(TSd.SysTF.PhysU)
360                 YLch(ic,2) = YLch(ic,2)*TSd.range(ic,3);
361             end
362         end
363         TSw.YLch = TSw.yFac*YLch;
364         for ic = 1:nch
365             if TSw.chPlt(ic)
366                 yl = mkshort(TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic
                    ,2));
367                 set(TSw.mCH(ic),'String',num2str(yl(1)));
368                 set(TSw.MCH(ic),'String',num2str(yl(2)));
369             end
370         end
371         rePltPage
372 
373     case 'Uniform ylim'
374         hTSw = gcf;
375         %     xy = get(gcf,'CurrentPoint');
376         xy = [200,200];
377         hUy = figure('Position',[xy(1) xy(2) 210,83],'Menubar',
            'None',...
378              'NumberTitle','off');
379         uicontrol('Parent',hUy,'Units','normalized','FontUnits'
            ,'normalized', ...
```

```
380             'BackgroundColor',[0.7 0.7 0.7],'FontSize',0.3787,
                    ...
381             'Position',[0.0428 0.5783 0.9285 0.3734],'Style','
                    frame','Tag','Frame1');
382         uicontrol('Parent',hUy,'Units','normalized','FontUnits'
                ,'normalized', ...
383             'BackgroundColor',[0.7 0.7 0.7],'FontSize',0.5681,
                    ...
384             'FontWeight','demi','Position',[0.0714 0.6144
                    0.8619 0.2650], ...
385             'String','Set Uniform Y-Limits','Style','text','Tag
                    ','StaticText1');
386         uicontrol('Parent',hUy,'Units','normalized','FontUnits'
                ,'normalized', ...
387             'BackgroundColor',[0.7 0.7 0.7],'FontSize',0.1404,
                    ...
388             'Position',[0.04285 0.0843 0.9142 0.4698],'Style','
                    frame','Tag','Frame2');
389         uicontrol('Parent',hUy,'Units','normalized', ...
390             'FontUnits','normalized','BackgroundColor',[1 1 1],
                     ...
391             'Callback','setUnifYlim','FontSize',0.4464, ...
392             'Position',[0.4428 0.1566 0.4476 0.3373],'Style','
                    edit','Tag','Edit Max');
393         uicontrol('Parent',hUy,'Units','normalized','FontUnits'
                ,'normalized',...
394             'BackgroundColor',[0.7 0.7 0.7],'FontSize',0.625,'
                    FontWeight','demi', ...
395             'Position',[0.1190 0.1687 0.3190 0.2289],'String','
                    Max', ...
396             'Style','text','Tag','Max text');
397
398     case 'Mark'
399         figure(gcf);
```

```
400         set(gcf,'Pointer','FullCrossHair')
401         posFig = get(gcf,'Position');
402         k = waitforbuttonpress;
403         p1 = get(gcf,'CurrentPoint');
404         np1(1) = p1(1)/posFig(3);
405         np1(2) = p1(2)/posFig(4);
406         pyi = yi*posFig(4);
407         plyi = lyi*posFig(4);
408         if(np1(1) > xi-.05 & np1(1) < lxi+xi & np1(2)> yi & np1
              (2) < lyi+yi)
409          if(np1(1) < xi)
410              np1(1) = xi;
411          end
412          initialRect = [p1(1) pyi 0 plyi];
413          fixedPoint = [p1(1) pyi];
414          set(gcf,'Pointer','Arrow');
415          finalRect = rbbox(initialRect,fixedPoint);
416          t1 = (np1(1)-xi)/lxi;
417          p2 = finalRect(3)/posFig(3);
418          t2 = p2/lxi+t1;
419          t2 = min(t2,1);
420          t1 =  fix(t1*(i1-i0+1)*TSd.dt) + i0*TSd.dt + TSd.t0
                ;
421          t2 =  fix(t2*(i1-i0+1)*TSd.dt) + i0*TSd.dt + TSd.t0
                ;
422          T12 = addT(t1,t2,TSw.T12);
423          [nSeg,dum] = size(T12);
424          ich = min(find(TSw.chPlt));
425          xl = get(TSw.ACh(1),'xlim');
426          xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
427          axes(TSw.hMask);
428          TSw.T12 = T12;
429          plot(T12',ones(size(T12')),'LineWidth',6,'Color','k
                ');
```

```
            set(gca,'xlim',xl,'Xtick',[],'Ytick',[]);
        else
            set(gcf,'Pointer','Arrow');
        end

    case 'Unmark'
        figure(gcf);
        set(gcf,'Pointer','FullCrossHair');
        posFig = get(gcf,'Position');
        k = waitforbuttonpress;
        p1 = get(gcf,'CurrentPoint');
        np1(1) = p1(1)/posFig(3);
        np1(2) = p1(2)/posFig(4);
        pyi = yi*posFig(4);
        plyi = lyi*posFig(4);
        if(np1(1) > xi-.05 & np1(1) < lxi+xi & np1(2)> yi & np1
           (2) < lyi+yi)
            if(np1(1) < xi)
                np1(1) = xi;
            end
            initialRect = [p1(1) pyi 0 plyi];
            fixedPoint = [p1(1) pyi];
            set(gcf,'Pointer','Arrow');
            finalRect = rbbox(initialRect,fixedPoint);
            t1 = (np1(1)-xi)/lxi;
            p2 = finalRect(3)/posFig(3);
            t2 = p2/lxi+t1;
            t2 = min(t2,1);
            t1 =  fix(t1*(i1-i0+1)*TSd.dt) + i0*TSd.dt + TSd.t0
               ;
            t2 =  fix(t2*(i1-i0+1)*TSd.dt) + i0*TSd.dt + TSd.t0
               ;
            T12 = subT(t1,t2,TSw.T12);
            [nSeg,dum] = size(T12);
```

```
            ich = min(find(TSw.chPlt));
            xl = get(TSw.ACh(1),'xlim');
            xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
            axes(TSw.hMask);
            TSw.T12 = T12;
            if(isempty(T12))
                delete(findobj('Parent',TSw.hMask,'Type','line'
                   ));
            else
                plot(T12',ones(size(T12')),'LineWidth',6,'Color
                   ','k');
            end
            set(gca,'xlim',xl,'Xtick',[],'Ytick',[]);
        else
            set(gcf,'Pointer','Arrow');
        end

    case 'Zoom plot'
        figure(gcf);
        set(gcf,'Pointer','FullCrossHair');
        posFig = get(gcf,'Position');
        k = waitforbuttonpress;
        p1 = get(gcf,'CurrentPoint');
        np1(1) = p1(1)/posFig(3);
        np1(2) = p1(2)/posFig(4);
        pyi = yi*posFig(4);
        plyi = lyi*posFig(4);
        if(np1(1) > xi-.05 & np1(1) < lxi+xi & np1(2)> yi & np1
           (2) < lyi+yi)
            if(np1(1) < xi)
                np1(1) = xi;
            end
            initialRect = [p1(1) pyi 0 plyi];
            fixedPoint = [p1(1) pyi];
```

```
492             set(gcf,'Pointer','Arrow');
493             finalRect = rbbox(initialRect,fixedPoint);
494             t1 = (np1(1)-xi)/lxi;
495             p2 = finalRect(3)/posFig(3);
496             t2 = p2/lxi+t1;
497             t2 = min(t2,1);
498             temp = i1-i0+1
499             i1 =  fix(t2*temp) + i0;
500             i0 =  fix(t1*temp) + i0;
501             t = [i0:i1]*dt+t0;
502             plotPage(t,yFac*y(:,[i0:i1]),TSw,TSd,0);
503             ich = min(find(TSw.chPlt));
504             xl = get(TSw.ACh(1),'xlim');
505             xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
506             axes(TSw.hMask);
507             T12 = TSw.T12;
508             plot(T12',ones(size(T12')),'LineWidth',6,'Color','k
                  ');
509             set(gca,'xlim',xl,'Xtick',[],'Ytick',[]);
510         else
511             set(gcf,'Pointer','Arrow');
512         end
513
514     case 'Clear marks'
515         TSw.T12 = [];
516         delete(findobj('Parent',TSw.hMask,'Type','line'));
517
518
519     case 'McohPlot'
520         ii = FigInd(gcf);
521         CohInd = zeros(TSd{ii}.nch,3);
522         if length(TSc) >= ii
523             if ~isempty(TSc{ii})
524                 h = findobj;
```

```
                if(~isempty(find(TSc{ii}.hFig==h)))
                    delete(TSc{ii}.hFig)
                    TSc{ii} = [];
                end
            end
        end
        NcohBand = 25;
        UpdateCohPlot = 0;
        [hFig,hAx] = CohMenu(TSd{ii}.ch_id',TSd{ii}.sta',ii,
           NcohBand);
        TScor = struct('CohInd',CohInd,'hFig',hFig,'hAx',hAx,'
           NcB',NcohBand);
        TSc{ii} = TScor;


    case 'pctOverlap'
        TSw.pctOverlap=str2num(get(gco,'String'));
        TSw.V=floor(0.01*TSw.nPw*TSw.pctOverlap);
        V=TSw.V;
        %TSw.npage=floor(((TSd.npts+nPw-1)/(nPw-V))+1);

        nPw = TSw.nPw%str2num(get('chnumpt','String'));
        nPw = min(nPw,npts);
        TSw.nPw = nPw;
        TSw.page = 1;
        i0=TSw.i0% = 1;
        i1=i0+nPw;
        if (nPw-V)==0
            TSw.npage=ceil(npts/nPw);
        else
            TSw.npage=floor(((npts+nPw-1)/(nPw-V))+1);
        end
        TSw.i1 = i1;
        xl = dt*[i0 i1]+t0;
```

```
557         for ic = nch:-1:1
558             if TSw.chPlt(ic)
559                 set(TSw.ACh(ic),'xlim',xl);
560                 rePltCh
561             end
562         end
563         ss = sprintf('page#\n %d/%d',TSw.page,TSw.npage);
564         set(TSw.PAGE,'String',ss);
565         xl = get(TSw.ACh(1),'xlim');
566         xl = xl*TSw.tFac+TSw.t0Eq0*TSd.t0;
567         axes(TSw.hMask);
568         T12 = TSw.T12;
569         plot(T12',ones(size(T12')),'LineWidth',6,'Color','k');
570         set(gca,'xlim',xl,'Xtick',[],'Ytick',[]);
571 
572     case 'mVkm'
573         display('mV/km')
574         TSd.mVkm=1-TSd.mVkm; mVkm=TSd.mVkm;
575         if mVkm==1,set(gcbo,'checked','on');else set(gcbo,'
              checked','off');end
576         for ic = nch:-1:1
577             chID=TSd.ch_id(ic,:);
578             %check if Efield
579             if regexp(chID,'Q')
580                 display(['Efield',num2str(ic)])
581                 if mVkm
582                     TSd.Units{ic}='mV/km';
583                 else
584                     TSd.Units{ic}='counts';
585                 end
586                 display('EsiteSclFactor')
587                 sf=getESiteScaleFactor_ssht(TSd.year, TSd.dyr,
                      TSd.sta(ic,:),[TSd.SR,chID])
588                 sf=sf*1e6
```

```
589                 if mVkm
590                     handles.y(ic,:)=handles.y(ic,:)/sf;
591                     TSd.range(ic,:)=TSd.range(ic,:)/sf;
592                 else
593                     handles.y(ic,:)=handles.y(ic,:)*sf;
594                     TSd.range(ic,:)=TSd.range(ic,:)*sf;
595                 end
596                 TSw.YLch(ic,1) = (1-TSw.centered) * mean(TSd.
                       range(ic,1:2));
597                 TSw.YLch(ic,2) =abs(diff(TSd.range(ic,:))/2); %
                       added nov 4th
598                 y=handles.y;
599                 data=y;
600                 a=TSw.YLch(ic,1)+[-1,1]*TSw.YLch(ic,2)
601                 yl = mkshort(a)
602                 set(TSw.mCH(ic),'String',num2str(yl(1)));
603                 set(TSw.MCH(ic),'String',num2str(yl(2)));
604                 if TSw.chPlt(ic)
605                     set(TSw.ACh(ic),'ylim',yl);
606                 end
607             end;
608             %rePltPage
609         end
610         y=handles.y;
611         data=y;
612         rePltPage;
613
614
615 end
616
617 dt=TSd.dt;
618 TSw.i0 = i0;
619 TSw.i1 = i1;
620 handles.TSw=TSw;
```

```
handles.TSd=TSd;
display(['Guidataing the handles again at end of plotTScbk'])
a=gcf;
if a==Ifig
    setappdata(Ifig,'FigData',handles);
end
```

```matlab
function [hTSw,TSw] = plotTSwin(TSd,rect,plotStart)
%   initialize plotting window
%   Usage:  [hTSw,TSw] = plotTSwin(TSd,rect,plotStart)
    hTSw = figure('Interruptible','on','Position',rect,'menubar','none','numbertitle','off');
    plotParams % Colors: bgc,bgc2,fgc Frame: ixf,iyf,xframe,yframe % TS plots: plots
    dtsp=(1-xframe)*0.1;
    zbarpos=[ plotpos(1) 0.97 plotpos(3) 0.02]; % was [0.4 0.97 0.55 0.02]

    % Axis Scaling on upper menu %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    hTaxis = uimenu('Parent',hTSw,'Label','Axis Scaling/Centering');
    uimenu(hTaxis,'Label','Relative to Start     ','checked','off','Tag','t0','Callback','plotTScbk','UserData',0);
    hTunits = uimenu(hTaxis,'Label','Time Units','Tag','Time units');
    uimenu(hTunits,'Label','Seconds','checked','off','Tag','tUnits','callback','plotTScbk','UserData',1);
    uimenu(hTunits,'Label','Minutes','checked','off','Tag','tUnits','callback','plotTScbk','UserData',60);
    uimenu(hTunits,'Label','Hours','checked','off','Tag','tUnits','callback','plotTScbk','UserData',3600);
    uimenu(hTunits,'Label','Days','checked','off','Tag','tUnits','callback','plotTScbk','UserData',86400);
    hYunits = uimenu(hTaxis,'Label','Y axes factor','Tag','Y units');
    uimenu(hYunits,'Label','.001','checked','off','Tag','yUnits','callback','plotTScbk','UserData',.001);
    uimenu(hYunits,'Label','1','checked','off','Tag','yUnits','callback','plotTScbk','UserData',1);
```

```
20      uimenu(hYunits,'Label','10','checked','off','Tag','yUnits',
           'callback','plotTScbk','UserData',10);
21      uimenu(hYunits,'Label','1000','checked','off','Tag','yUnits
           ','callback','plotTScbk','UserData',1000);
22      uimenu(hTaxis,'Label','Set unif. y limit','Tag','Uniform
           ylim','Callback','plotTScbk');
23      uimenu(hTaxis,'label','Centered       ','Tag','Centered','
           checked','on','CallBack','plotTScbk');
24      uimenu(hTaxis,'label','E-field Units SI','Tag','mVkm','
           checked','off','CallBack','plotTScbk');
25  % end of Axis Scaling/Centering
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26  % Mark Data Segments Menu (now on UPPER menu)
27      hMDS= uimenu(hTSw,'Label','Mark Data Segments');
28            uimenu(hMDS,'label','Mark data      ','Tag','Mark','
                 Callback','plotTScbk');
29            uimenu(hMDS,'label','Unmark data    ','Tag','Unmark',
                 'Callback','plotTScbk');
30            uimenu(hMDS,'label','Zoom plot      ','Tag','Zoom
                 plot','Callback','plotTScbk');
31            uimenu(hMDS,'label','Clear marks    ','Tag','Clear
                 marks','Callback','plotTScbk');
32      hMask = axes('Position',zbarpos,'Ytick',[],'Xtick',[]); %
           keep for now% get rid of later
33
34      MCH = zeros(TSd.nch,1);
35      mCH = zeros(TSd.nch,1);
36
37      i0 = plotStart.i0;
38      di = plotStart.di;
39      pctOverlap=plotStart.pctOverlap;
40      V=plotStart.V;
41      tUnits = plotStart.tUnits;
42      tFac = plotStart.tFac;
```

```
    eval(['set(findobj(' '''Label''' ',tUnits),' '''checked''
       ',' '''on''' ')'])
    t0Eq0 = plotStart.t0Eq0;
    if(t0Eq0), set(findobj('Tag','t0'),'checked','on');end
    T12 = plotStart.T12;
    yFac = plotStart.yFac;
    eval(['set(findobj(' '''Label''' ',num2str(yFac)),' '''
       checked''' ',' '''on''' ')'])
    nPw = plotStart.nPw;
    chPlt = ones(TSd.nch,1);
    chPltOld = ones(TSd.nch,1);
    centered = plotStart.centered;
    if centered==0, set(findobj('Tag','Centered'),'checked','
       off');end
    page = plotStart.page;
    npage=floor(((TSd.npts+nPw-1)/(nPw-V))+1);
    %npage = ceil(TSd.npts/nPw);
    nch = TSd.nch;
    % YL is range NOW on right side of plots
    for ic = 1:nch
       YLch(ic,1) = (1-centered) * mean(TSd.range(ic,1:2));
       YLch(ic,2) = (TSd.range(ic,2)-TSd.range(ic,1))/2;
       if(TSd.SysTF.PhysU)
          YLch(ic,2) = YLch(ic,2)*TSd.range(ic,3);
       end
    end
    YL = [YLch(:,2)-YLch(:,1) YLch(:,2)+YLch(:,1)];
    YL = mkshort(YL);
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    ACh = zeros(nch,1);
    PCh = zeros(nch,1);
    SChM = zeros(nch,1);
```

```
    SChP = zeros(nch,1);
%%%%%%%%%%%%%%%%%%% double frame for control panel %%%%%%%%%%%%%%%%%%%%%%%%%%%
    uicontrol(hTSw,'Style','frame','Units','normalized','Position',[ixf iyf xframe yframe],'BackgroundColor',bgc);
    dwin=min(0.02*xframe,0.02*yframe);
    uicontrol(hTSw,'Style','frame','Units','normalized','Position',[ixf+dwin iyf+dwin xframe-2*dwin yframe-2*dwin ],'BackgroundColor',bgc);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% UPPER BOTTONS, 3 fields %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ubn=3;
    ubdwin=max(0.5*dwin,0.002);
    ubwidth = (xframe-2*dwin-(ubn+1)*ubdwin)/ubn;
    ubheight= max(0.05,0.05*yframe);
    ubxpos=ixf+dwin+ubdwin;
    ubypos=iyf+yframe-2*dwin-ubdwin-ubheight;
    din=0.002;
% #Pts frame
    uicontrol(hTSw,'Style','frame','Units','normalized','Position',[ubxpos ubypos 2*ubwidth+ubdwin ubheight],'BackgroundColor',bgc);
% #Pts text
    uicontrol(hTSw,'Style','text','String','# Pts ','Units','normalized','Position',[ubxpos+din ubypos+din ubwidth ubheight-2*din],...
            'HorizontalAlignment','Center','FontUnits','normalized','FontSize',.5,'FontWeight','demi',...
            'BackgroundColor',bgc2,'ForegroundColor',fgc(2,:), 'tooltipstring', 'Control the Number of points
```

```
                displayed in time series window using the edit
                box');
% #Pts edit
    ubxpos=ubxpos+ubwidth+ubdwin;
    NPtWin = uicontrol(hTSw,'Style','edit','String',num2str(nPw
       ),'Tag','chnumpt','Units','normalized',...
             'Position',[ubxpos+din ubypos+din ubwidth-2*din
                ubheight-2*din],'FontUnits','normalized','
                FontSize',.5,...
             'BackgroundColor',ebc,'ForegroundColor',fgc(2,:),'
                Callback','plotTScbk');
% Replot
    ubxpos=ubxpos+ubwidth+ubdwin;
    uicontrol(hTSw,'Style','Pushbutton','Units','normalized','
       Position',[ubxpos ubypos ubwidth ubheight],...
              'Tag','replot','String','Replot','FontUnits','
                 normalized','FontSize',.5,...
              'FontWeight','demi','Callback','plotTScbk','
                 BackgroundColor',bgc2,'ForegroundColor',fgc
                 (2,:));
%overlap
    ubxpos=ixf+dwin+ubdwin;
    ubypos=iyf+yframe-2*dwin-2*ubdwin-2*ubheight;
    uicontrol(hTSw,'Style','text','String','% Overlap','Units',
       'normalized','Position',[ubxpos+din ubypos+din ubwidth
       ubheight-2*din],...
             'HorizontalAlignment','Center','FontUnits','
                normalized','FontSize',.5,'FontWeight','demi'
                ,...
             'BackgroundColor',bgc2,'ForegroundColor',fgc(2,:))
                ;
    ubxpos=ubxpos+ubwidth+ubdwin;
    pctOvlpWin = uicontrol(hTSw,'Style','edit','String',num2str
       (pctOverlap),'Tag','pctOverlap','Units','normalized',...
```

```
110             'Position',[ubxpos+din ubypos+din ubwidth-2*din
                  ubheight-2*din],'FontUnits','normalized','
                  FontSize',.5,...
111             'BackgroundColor',ebc,'ForegroundColor',fgc(2,:),'
                  Callback','plotTScbk');
112 %%%%% "-" Reset "+"
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113     ubxpos=ixf+dwin+ubdwin;
114     ubypos=iyf+yframe-2*dwin-2*ubdwin-3*ubheight;
115 % -
116     uicontrol(hTSw,'Style','Pushbutton','String','-','Units','
          normalized','Position',[ubxpos ubypos ubwidth ubheight],'
          Tag','zoomOut',...
117             'FontUnits','normalized','FontSize',.6,'FontWeight
                  ','bold','BackgroundColor',bgc,'ForegroundColor'
                  ,fgc(2,:),'Callback','plotTScbk');
118 % Reset
119     ubxpos=ubxpos+ubwidth+ubdwin;
120     uicontrol(hTSw,'Style','Pushbutton','String','Reset','Units
          ','normalized','Position',[ubxpos ubypos ubwidth ubheight
          ],'Tag','reset',...
121             'FontUnits','normalized','FontSize',.5,'FontWeight
                  ','demi','BackgroundColor',bgc,'ForegroundColor'
                  ,fgc(2,:),'Callback','plotTScbk');
122 % +
123     ubxpos=ubxpos+ubwidth+ubdwin;
124     uicontrol(hTSw,'Style','Pushbutton','String','+','Units','
          normalized','Position',[ubxpos ubypos ubwidth ubheight],'
          Tag','zoomIn',...
125             'FontUnits','normalized','FontSize',.6,'FontWeight
                  ','bold','BackgroundColor',bgc,'ForegroundColor'
                  ,fgc(2,:),'Callback','plotTScbk');
126 %%%%%%%%%%%%% LOWER buttons in control panel
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%% now frame adjustable %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    lbn=3; % number of lower buttons
    lbdwin=max(0.5*dwin,0.002);
    lbwidth = (xframe-2*dwin-(lbn+1)*lbdwin)/lbn;
    lbheight= max(0.05,0.05*yframe);
    lbxpos=ixf+dwin+lbdwin;
% Prev
    uicontrol(hTSw,'Style','Pushbutton','Units','normalized','Position',[lbxpos iyf+2*dwin lbwidth lbheight],...
              'String','Prev','Tag','PrevPage','FontUnits','normalized','FontSize',.4167,'FontWeight','demi','Callback','plotTScbk',...
              'BackgroundColor',bgc2,'ForegroundColor',fgc(2,:));
% Next
    lbxpos=lbxpos+lbwidth+lbdwin;
    uicontrol(hTSw,'Style','Pushbutton','Units','normalized','Position',[lbxpos iyf+2*dwin lbwidth lbheight],...
              'String','Next','Tag','NextPage','FontUnits','normalized','FontSize',.4167,'FontWeight','demi','Callback','plotTScbk',...
              'BackgroundColor',bgc2,'ForegroundColor',fgc(2,:));
% Quit
    lbxpos=lbxpos+lbwidth+lbdwin;
    uicontrol('Style','Pushbutton','Units','normalized','Position',[lbxpos iyf+2*dwin lbwidth lbheight],'String','Quit','Tag','Quit',...
        'FontUnits','normalized','FontSize',.4167,'FontWeight','demi','Callback','plotTScbk','BackgroundColor',bgc2,'ForegroundColor',fgc(2,:));
% Goto
    lbxpos=ixf+dwin+lbdwin;lbypos=iyf+lbheight+dwin+2*lbdwin;
```

```
    uicontrol(gcf,'Style','text','String',['Go';'to'],'Units','normalized','Position',[lbxpos lbypos lbwidth lbheight],...
          'FontUnits','normalized','FontSize',.4167,'FontWeight','demi','BackgroundColor',bgc,'ForegroundColor',fgc(2,:));
    lbxpos=lbxpos+lbwidth+lbdwin;
    h_gotopage =uicontrol(gcf,'Style','edit','String',num2str(page),'Units','normalized','Position',[lbxpos lbypos lbwidth lbheight],...
          'Tag','gotoPage','BackgroundColor',bgc2,'ForegroundColor',fgc(2,:),'FontUnits','normalized','FontSize',.4167,'FontWeight','demi',...
          'Callback','plotTScbk');
    lbxpos=lbxpos+lbwidth+lbdwin;
    ss = sprintf('page#\n %d/%d',page,npage);
    PAGE = uicontrol(gcf,'Style','text','String',ss,'Units','normalized','Position',[lbxpos lbypos lbwidth lbheight],'FontWeight','demi',...
        'BackgroundColor',bgc,'ForegroundColor',fgc(2,:));
%%%%%%%%%%%%%%%%%%%%% Channel BUTTONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [ncol,nrow,staN]=NcolNrow(TSd.sta,TSd.ch_id);
% space for the buttons in control panel
    cby=yframe-2*(ubheight+ubdwin)-2*(lbheight+lbdwin);
    cbx=xframe-4*dwin;
    cbh=(cby-(max(nrow+1)+1)*ubdwin)/max(nrow+1);
    cbw=cbx/ncol;
    cbh=min(cbh,0.05);
    cbw=min(cbw,lbwidth); % Lana, Aug 2005, was min(cbw,0.05);
% Channel buttons are centered
    xpos=ixf+2*dwin+(cbx-cbw*ncol-ubdwin*(ncol-1))/2;
```

```
    ypos=iyf+yframe-2*(lbheight+lbdwin)-(cby-cbh*max(nrow)-ubdwin*(max(nrow)-1))/2;
%
    for icol = 1:ncol
    ypos=iyf+yframe-2*(lbheight+lbdwin)-(cby-cbh*max(nrow)-ubdwin*(max(nrow)-1))/2;
    uicontrol(hTSw,'Style','Text','String',staN(icol,:),'Units','normalized','Position',[xpos ypos cbw cbh],...
          'BackgroundColor',bgc,'ForegroundColor',fgc(2,:),'FontUnits','normalized','FontSize',.4,'FontWeight','demi');
     chid=[];ic=[];
     for k=1:nch
        if TSd.sta(k,:)==staN(icol,:),chid=[chid;TSd.ch_id(k,:)];ic=[ic,k];end
     end
     for irow=1:nrow(icol)
       ypos=ypos-cbh-ubdwin;
       uicontrol(hTSw,'Style','PushButton','String',chid(irow,:),'Units','normalized','Position',[xpos ypos cbw cbh],...
          'Tag','chonoff','ToolTipString','Channel plot ON/OFF','UserData',ic(irow),'BackgroundColor',bgc,...
          'ForegroundColor',fgc(2,:),'FontUnits','normalized','FontSize',.4,'FontWeight','demi','CallBack','plotTScbk');
     end
     xpos=xpos+ubdwin+cbw;
    end
    set(hMask,'Visible','off')
%    waitforbuttonpress
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
190 %   plotting window structure:
191 %   Structure TSw     (Time series plot description)
192 %    ACh(nch)  axis handles for each channel
193 %    PCh(nch)  line handles  for each channel
194 %    SChM(nch),SChP(nch)  handles  for channel zoom controls
195 %    YLch(nch,2)   [center,range]    defines ylim for each
    channel
196 %    mCH, MCH are handles for max/min windows
197 %    nPw    # of points to be plotted in one frame
198 %    i0        data array index for first point plotted in
    window
199 %    di        data index skip factor  (1 = no skip)
200 %    i1        data array index for last point plotted in
    window
201 %    chPlt, chPltOld     0/1 arrays indicating which channels
202 %                 are active (i.e., to be plotted)
203 %    centered  indicator for centered TS
204 %    PAGE      handle to page number indicator
205 %    page      page number
206 %    npage     number of pages
207 %  NOTE: many of the fields are blank until the first call to
    plotPage
208       TSw = struct('ACh',ACh,'PCh',PCh,'YLch',YLch,'mCH','
          MCH',MCH,...
209             'nPw',nPw,'i0',i0,'i1',nPw+V,'di',di,'chPlt',chPlt
                ,...
210             'chPltOld',chPltOld,'centered',centered,'PAGE',PAGE
                ,...
211             'page',page,'npage',npage,'SChP',SChP,'SChM',SChM
                ,...
212             't0Eq0',t0Eq0,'tUnits',tUnits,'tFac',tFac,'yFac',
                yFac,...
213             'hMask',hMask,'T12',T12,'TSd',TSd,'V',V);
```

```
function handles = populateSpecAvg(hObject, eventdata, handles)
%This funciton need to perform at least 3 main tasks:
%1. Load the cfg file
%2. Load the associated data into a data structure

%% readCfg
cfg=handles.cfg;
for iDay=1:length(cfg)
    sourceFiles{iDay}=identifySpecAvgFiles(cfg{iDay});
    epochMeta{iDay}=associateEpochNumber2(cfg{iDay}.yyyy,cfg{
       iDay}.ddd,cfg{iDay}.site,cfg{iDay}.sens);
end
%sourceFiles{iDay} is a datastructure with the files to be
   loaded for each day.


%% LOAD DATA
%Initialize data structure, 1 entry per day with field .data
   being the concatenation of the sourceFiles
for iDay=1:length(cfg)
    nPts=0;
    for iSourceFile=1:length(sourceFiles{iDay})
        nPts=nPts+(sourceFiles{iDay}{iSourceFile}.ndx1-
           sourceFiles{iDay}{iSourceFile}.ndx0)+1;
    end
    daysOfData{iDay}.data=zeros(1,nPts);
end

%Now populate the dataStructure daysOfData
avgSpec=nan(length(cfg),round(nPts/2));
for iDay=1:length(cfg)
    data=[];
    for iFile=1:length(sourceFiles{iDay})
        fn=sourceFiles{iDay}{iFile}.fname;
```

```
31          load(fn)
32          data=[data y.data(sourceFiles{iDay}{iFile}.ndx0:
               sourceFiles{iDay}{iFile}.ndx1)'];
33      end
34      daysOfData{iDay}.data=data;
35      daysOfData{iDay}.spec=TS2FC_single(data, epochMeta{iDay},
           0, 'hamming', cfg{iDay}.dT, cfg{iDay}.fieldType);
36      avgSpec(iDay,:)=abs(daysOfData{iDay}.spec.spec).^2;
37  end
38  handles.daysOfData=daysOfData;
39  handles.epochMeta=epochMeta;
40  handles.avgSpec=avgSpec;
41  handles.cfg=cfg;
42  guidata(handles.SpecAvgFig,handles);
```

```
%{
Set Up GUI Geometery.

@note: when working with matlab UI positions, distance is from
   left and from bottom.

@type xOffset: float
@var xOffset: Fraction of the arrayManager GUI width to
   allocate for the
create/remove frame - which is Frame#5
@type nFramesL: int
@var nFramesL: The number of "long (wide) frames in the upper
   box above
the master list - corresponds to selectedSites,AllSites listbox
    frames
@type nFramesS: int
@var nFramesS: The number of "short" (narrow) frames in the
   upper box above
the master list - default=1; this is where the add/remove/
   update buttons go
@type: LSratio: float
@var LSratio: the ratio of the center box-width to the two
   outer boxes in
the region above the masterList

%}
clear handles
xOffset=0.2;
nFramesL=2;
nFramesS=1;
nFrames=nFramesS+nFramesL;
LSratio=1.3; %ratio of long to short boxes
FWunit=(1-xOffset)/(LSratio*nFramesL+nFramesS);%solve for
   FrameWidthUnit
```

```
handles.topEdge=6;
handles.botEdge=2;
handles.figWidth=650;
handles.heightMore=100; %height of 2nd level below the six
   primary frames
oldHand.GET_GUI_POSN=[20,550,320,200];
handles.figHeight=oldHand.GET_GUI_POSN(4)+handles.heightMore;
handles.lhBound=oldHand.GET_GUI_POSN(1)+oldHand.GET_GUI_POSN(3)
   ;
handles.SITE_POSN=[handles.lhBound+20,oldHand.GET_GUI_POSN(2)-
   handles.heightMore,handles.figWidth,handles.figHeight];
handles.sideWidth=FWunit;
handles.horzLine=handles.heightMore/handles.figHeight;
frameVec=[LSratio 1 LSratio];

%<set Frame postions>
handles.framePosns=zeros(nFramesS+nFramesL+2,4);
handles.framePosns(1,:)=[xOffset handles.horzLine frameVec(1)*
   handles.sideWidth 1-handles.horzLine];
for frame=2:nFramesL+nFramesS
    xPosn = handles.framePosns(frame-1)+frameVec(frame-1)*
       handles.sideWidth;
    yPosn = handles.horzLine;
    width = frameVec(frame)*handles.sideWidth;
    height = 1-handles.horzLine;
    positionString = num2str([xPosn,yPosn,width,height]);
    cmd=['handles.framePosns(frame,:)=[',positionString,'];'];
    eval(cmd);
end
%<\set Frame postions>

handles.framePosns(4,:)=[xOffset 0 1-xOffset handles.horzLine];
   %masterListBox
handles.framePosns(5,:)=[0 0 xOffset 1]; %Add/Remove Arrays
```

```
handles.ArryMgrFig=figure('MenuBar','none', ...
    'Name', 'Site and Selection Tool', ...
    'NumberTitle','off','Position',handles.SITE_POSN,...
    'units','normalized','tag','ARRAYMGR');

%<instantiate frames>
for iFrame=1:5%nFrames
    frmCols=[0 2 4 6 8]/10;
    handles.Frame{iFrame}=uipanel( ...
        'Parent',handles.ArryMgrFig,...
        'units','normalized',...
        'Position',handles.framePosns(iFrame,:),...
        'BackgroundColor',[frmCols(iFrame)*1 0.1 0.5],...
        'tag',['frame',num2str(iFrame)]);
end
%production: 'BackgroundColor',[0.1 0.1 0.5],...
%debugging: 'BackgroundColor',[frmCols(iFrame)*1 0.1 0.5],...
%<\instantiate  frames>


%%%MAKE MASTER DATA STRUCTS %%%%%%
if verbose
    display(['Reading Master Data Structs'])
end
[handles.SITES handles.NETWORKS handles.CHANNELS]=readSiteNetCh
   ;

%initializeALL_ARRAY
ARRAYS_file = fullfile(SYSpath,'SITES','ARRAYS.mat')
load(ARRAYS_file)

if numel(ARRAYS)<2
    display(['No stored Arrays, Generating ALL_ALL by default'
       ])
```

```
87      initializeALL_ARRAY
88  end
89  handles.ARRAYS=ARRAYS;
90  for rA=1:numel(ARRAYS)
91      handles.ARRAYList{rA}=ARRAYS{rA}.name;
92      display(['ARRAY #',num2str(rA),' ',handles. ARRAYList{rA}])
93  end
94  arrayNow=1;
95  handles.UIC=chr2num_relDB(handles.SITES, handles.CHANNELS,
       ARRAYS{arrayNow}.chrArray);
96  %return
97  for iSite=1:numel(handles.SITES)
98      siteSubArrayIndex=find(abs(handles.UIC(:,1))==iSite);
99      chListIndex=find(handles.UIC(siteSubArrayIndex,2)>0);
100     handles.siteChList{iSite}=handles.CHANNELS{iSite}(
           chListIndex);
101 end
102 handles=siteChRadio_CreateFcn(hObject, eventdata,handles);
103 handles=updateSelected(hObject, eventdata, handles);
104 handles=arrayManageGUICreateFcns(hObject, eventdata, handles);
105 handles=arrayListCreateFcn(hObject, eventdata, handles);
106 set(handles.arrayListbox,'Value',arrayNow);
107 handles=addArrayPB_CreateFcn(hObject, eventdata, handles);
108 handles=rmArrayPB_CreateFcn(hObject, eventdata, handles);
109 handles=updateSelected(hObject, eventdata, handles);
110 handles=selUnselPBs_CreateFcn(hObject, eventdata,handles);
111 handles=updatePB_CreateFcn(hObject, eventdata,handles);
112 handles.allText=uicontrol('units','normalized','position'
       ,[0.1,0.8,0.8,0.2]...
113      ,'style','text','string','ALL','fontsize',12,'Parent',
           handles.Frame{1});
114 handles.selText=uicontrol('units','normalized','position'
       ,[0.1,0.8,0.8,0.2]...
```

```
115     ,'style','text','string','SELECTED SITES','fontsize',12,'
           Parent',handles.Frame{3});
116 set(handles.siteChListbox,'String',handles.siteChTextList);
117 handles=updateSelected(hObject, eventdata, handles);
118 display(['Exiting prepArrayManagerGUI'])
```

```
%Set Up GUI Geometery
%% 1. BUILD PANELS
%Panels 1 and two stack over each other having the same width - p12Width
%Panel 2 is much taller than panel 1, 1=p1h+p2h (heigths of panel 1 and 2)
%Panels 3 and 4 split the remaining space in half.

nPanels=4;
p12Width=0.2;
p1h=0.2;
p2h=1-p1h;
handles.nPanels=nPanels;


handles.SITE_POSN=[100,100,800,600];

handles.framePosns=zeros(nPanels,4);
handles.framePosns(1,:)=[0 1-p1h p12Width p1h];
handles.framePosns(2,:)=[0 0 p12Width 1-p1h];
handles.framePosns(3,:)=[p12Width 0.5 1-p12Width 0.5];
handles.framePosns(4,:)=[p12Width 0 1-p12Width 0.5];

handles.SpecAvgFig=figure('MenuBar','none', ...
    'Name', 'Spectral Averaging Tool', ...
    'NumberTitle','off','Position',handles.SITE_POSN,...
    'units','normalized','tag','SPECAVG');

%set up frames
for f=1:4%nFrames
    frmCols=[0 2 4 6 8]/10;
    handles.Frame{f}=uipanel( ...
        'Parent',handles.SpecAvgFig,...
        'units','normalized',...
```

```
        'Position',handles.framePosns(f,:),...
        'BackgroundColor',[1 1 frmCols(f)],...
        'tag',['frame',num2str(f)]);
end
handles.tsax=axes('Parent',handles.Frame{3});
handles.specax=axes('Parent',handles.Frame{4});


guidata(handles.SpecAvgFig,handles);
handles.cfg=readParseSpecAvgcfg;

%% PANEL 1
%Generate the seven elements inside of Panel 1
%UT
handles.UTText=uicontrol('units','normalized','position'
   ,[0.1,0.8,0.2,0.2],'style','text','string','UT ','fontsize'
   ,12,'Parent',handles.Frame{1});
handles.HHMM=uicontrol('units','normalized','position'
   ,[0.35,0.8,0.5,0.2],'style','text','string',handles.cfg{1}.
   hhmmStr,'fontsize',12,'Parent',handles.Frame{1});
%SMOOTHING
handles.smooText=uicontrol('units','normalized','position'
   ,[0.1,0.6,0.5,0.2],'style','text','string','SMOOTH','fontsize
   ',12,'Parent',handles.Frame{1});
handles.smoothEdit=uicontrol('units','normalized','position'
   ,[0.65,0.6,0.3,0.2],'style','edit','string','1 ','fontsize'
   ,12,'Parent',handles.Frame{1});
%SHOW AVG
handles.showAvgCheckBox=uicontrol('units','normalized','
   position',[0.1,0.4,0.6,0.2],'style','checkbox','string','
   SHOWAVG','fontsize',12,'Parent',handles.Frame{1});
handles.SiteText=uicontrol('units','normalized','position'
   ,[0.1,0.1,0.5,0.2],'style','text','string',handles.cfg{1}.
   siteSRSens,'fontsize',12,'Parent',handles.Frame{1});
```

```
55
56 guidata(handles.SpecAvgFig,handles);
57
58 %% PANEL 2
59 nDay=length(handles.cfg)
60 for iDay = 1:nDay
61     PB{iDay}=uicontrol('units','normalized','position',[0.02,
          0.9*(iDay-1)/nDay+0.01 , 0.7, 0.9/nDay],'style','
          pushbutton','string',['DAY ',handles.cfg{iDay}.ddd],...
62     'fontsize',12,'Parent',handles.Frame{2}, 'UserData',iDay,'
          callback', {@dayPush,handles});
63     CB{iDay}=uicontrol('units','normalized','position',[0.85,
          (0.9*(iDay-1)/nDay)+0.01, 0.13, 0.9/nDay],'style','
          checkbox',...
64     'fontsize',12,'Parent',handles.Frame{2});
65 end
66 handles.PB=PB;
67 handles.CB=CB;
68 guidata(handles.SpecAvgFig,handles);
```

```
if TSw.PCh(ic)~=0,
    if TSw.centered
        %display(['cntered'])
        data=handles.y;
        d1= data(ic,[i0:di:i1]);
        data0=mean(d1(find(isnan(d1)==0)));
        set(TSw.PCh(ic), ...
            'Xdata',[i0:di:i1]*dt+t0,...
            'Ydata',TSw.yFac*(data(ic,[i0:di:i1])-data0));
        %get(TSw.PCh(ic))
    else
        %centered option
        if size(data,1)>size(data,2)
            data=data';
        end
        set(TSw.PCh(ic),...
              'Xdata',[i0:di:i1]*dt+t0,...
              'Ydata',data(ic,[i0:di:i1])*TSw.yFac);
    end
end
```

```
%replot page
%call the TSw from the figure;

%i0 = TSw.nPw*(TSw.page-1) + 1;
display(['Entering ReplotPage'])
i0=1+(TSw.page-1)*(nPw-V);
np = min(TSw.nPw,TSd.npts-i0+1);
i1 = i0+np-1;
t = ([i0:TSw.di:i1]*TSd.dt+TSd.t0*(1-TSw.t0Eq0))/TSw.tFac;
TSw.i0 = i0;
TSw.i1 = i1;
TSw = plotPage(t,TSw.yFac*y(:,[i0:di:i1]),TSw,TSd,0);
```

```
function FCRA = readFCRA(RA, segmentSpecifier)
global SYSpath DATApath SLaSH
%usage: FCRA = readFCRA(RA, segmentSpecifier)
%This function reads in simultaneous FC files from a bank of instruments specified by RA and concatenates time series
%of univariate (single instrument) FCs into multivariate (multiple station) FCs
%Inputs:
%RA: the collection of instruments to load, RA=ARRAYS{i} where integer i
%specifies the specific array, from list of all arrays ARRAYS.mat
%RA.chrArray: struct, list of channel specifiers

%segmentSpecifier
%number of FCs in each decimation level is fixed priori by bFC.mat for a
%particular sampling rate, so we can initiazed the FCRA dataStructure based
%on that, but that assumes the percent overlap is fixed, which I would
%prefer not to Fix.  I suppose, the way to go is to fix L,V, in
%HarmonicsAndDecimationSchemes, and store this in bFC.  Then, when using
%the procTS GUI, set default values to those from bFC.

SITECH=RA.chrArray;
nInst=numel(SITECH);

%Initialize FCArray as NaN
SR=segmentSpecifier.SR;
load([SYSpath,'bFC_',SR,'.mat'])


```

```
for iD=1:numel(bFC)
    for iB=1:numel(bFC{iD})
        nBin=length(bFC{iD}{iB}.FCI);
        nWin=bFC{iD}{iB}.nWin;
        FCRA{iD}{iB}=nan(nInst,nBin,nWin);
    end
end

for iInst=1:nInst
    FCfilename=[DATApath,SR,segmentSpecifier.yStr,SLaSH,'FC',
       SLaSH,SITECH{iInst}.locn,'_',SR,SITECH{iInst}.chn,'_',
       segmentSpecifier.dStr];
    if SR=='B'
        FCfilename=[FCfilename,'_',segmentSpecifier.hStr];
    end
    FCfilename=[FCfilename,'.mat'];
    clear FC
    if exist(FCfilename,'file')
        load(FCfilename)
    else
        mssg=['FC FILE: ',FCfilename,' DNE - Have you Processed
            the TS to FCs?'];
        warning(mssg);
        break
    end
    %Could later include a check that FC file was created with
       same bFC
    %file as is being used to process it now
    for iD=1:numel(bFC)
        for iB=1:numel(bFC{iD})
            FCRA{iD}{iB}(iInst,:,:)=FC{iD}{iB};
        end
    end
end
```

```
function cfg =readParseSpecAvgCfg()
%7/11/2011
%K.N.Kappler
%Program to read-in and parse the data from a specAvg.cgf file
%SpecAvg file is of fixed format:

%<bof>
%MHDL LT1  #Station  SR SENS
%0230 65       #t0 (HHMM)  dt (in minutes)
%2011 1         #YYYY DDD
%2011 2
%2011 3
%2011 4
%.
%.
%.
%2011 N
%<eof>

%OUTPUT: sourceFile data structure - 1 entry for each 'day' of
   data
%The sourceFile structure is a list of files which need to be
   tapped in
%order to load the interval specified by specAvg.cfg.

global MATpath
cfgFile = fullfile(MATpath,'specAvg.cfg'); %maybe keep in sys?
fid=fopen(cfgFile);

%read in line 1, containing SITE SR SENSOR
line1=fgetl(fid);
parts=regexp(line1, ' ', 'split');
site=parts{1};
SR=parts{2}(1);
```

```
sens=parts{2}(2:end);
clear parts line1

%read in line 2 containing t0 and duration in minutes of
   ensemble
line2=fgetl(fid);
parts=regexp(line2, ' ', 'split');
hhmm=parts{1};
t0hh=str2num(hhmm(1:2));
t0mm=str2num(hhmm(3:4));
minterval=parts{2};
clear parts line2

%Read in remaining lines
a=fgetl(fid);
parts=regexp(a, ' ', 'split');
cfg{1}.yyyy=parts{1};
cfg{1}.ddd=parts{2};
cfg{1}.site=site;
cfg{1}.SR=SR;
cfg{1}.sens=sens;
cfg{1}.t0hh=t0hh;
cfg{1}.t0mm=t0mm;
cfg{1}.minterval=minterval;
cfg{1}.hhmmStr=hhmm;
cfg{1}.siteSRSens=[site,' ',SR,sens];

if regexp(cfg{1}.SR,'L')
    cfg{1}.dT=1;
elseif regexp(cfg{1}.SR,'B')
    cfg{1}.dT=1/40;
end
if regexp(cfg{1}.sens(1),'T')
    cfg{1}.fieldType='MAGNETIC';
```

```
elseif regexp(cfg{1}.sens(1),'Q')
    cfg{1}.fieldType='ELECTRIC';
end


ndx=2;
while a~=-1
    a=fgetl(fid);
    if regexp(a,'eof')
        display('reached end of specAvg.cfg file');
        break
    else
        parts=regexp(a, ' ', 'split');
        cfg{ndx}.yyyy=parts{1};
        cfg{ndx}.ddd=parts{2};
        cfg{ndx}.site=site;
        cfg{ndx}.SR=SR;
        cfg{ndx}.sens=sens;
        cfg{ndx}.t0hh=t0hh;
        cfg{ndx}.t0mm=t0mm;
        cfg{ndx}.minterval=minterval;
        cfg{ndx}.dT=cfg{1}.dT;
        cfg{ndx}.fieldType=cfg{1}.fieldType;
        ndx=ndx+1;
    end
end
clear a parts

fclose(fid);
```

```
function [SITES NETWORKS CHANNELS]=readSiteNetCh()
%INPUTS: none, later would like a verbose mode
%OUTPUTS SITES, NETWORKS, CHANNELS, as three separate data
   structures, can be
%splice toghther in to SITES.locn SITES.chlist, SITES.networks
   later
global ULFEMpath verbose SYSpath
display(['Reading in a List of Sites, and their respective
   Networks and Channels'])
siteNetFile = fullfile(SYSpath,'SITES','sitesNetworks.lst')
SitesNets=textread(siteNetFile,'%s');
numSites=numel(SitesNets)/2;
if mod(numel(SitesNets),2)
    display('The number of SITE names and Network specifiers is
        not the same, Check the SITENET file in the sys folder')
else
    odds=1:2:numel(SitesNets);
    evens=2:2:numel(SitesNets);
    for iSite=1:numSites
        SITES{iSite}=SitesNets{odds(iSite)};
        NETWORKS{iSite}=SitesNets{evens(iSite)};
        snsFile = fullfile(SYSpath,'SITES',[SITES{iSite},'.sns'
           ])
        cmd=['cat ',snsFile];
        [s w]=unix(cmd);
        ch=ksplit(w,'\n');
        CHANNELS{iSite}=ch;
    end
end
```

```
%{
Set up directory paths

@note: Oct 30, 2012: Previously I was supporting separate filepath structures,
windows paths (in lower case) and *NIX paths in upper case. Use of the matlab
command fullfile (which will later be replaced with os.path.join() in python)
eliminates the need to cover different file separator types - however; this
remains to be tested under cygwin, so i will leave the cygwin blocks in for now.

%ULFEMpath is the matlab-windows type path, ulfempath is the unix-cygwin type path
%In general I tried to keep UPPERCASEpath variables as Linux, and lowercase
%variables as windows when dealing with pathnames.
%}

global ULFEMpath SYSpath syspath SCRpath SCRPLOTpath scrplotpath scrpath ...
    DATApath datapath verbose SLASH
global rect0 METADATApath metadatapath EPOCHSpath epochspath me
    binStageDir ...
    ascStageDir ulfemToolbox metaStageDir
global COILpath FIGUREpath figurepath PYTHONpath pythonpath MATpath matpath ...
    samplingRateContainerMap

if verbose
    display(['Setting Paths to sys, scr, data, and TOOLBOX'])
end
```

```
25
26 SLaSH=loadSLASH;
27 lowcayshun='SJMATLAB';%AMPERE_debian';
28 switch lowcayshun
29     case 'BSL'
30         ULFEMpath = fullfile('/scr','em1','kappler','ULFEM');
31         TOOLBOX='/data/22/kappler/TOOLS/';
32         username='kappler';
33         binStageDir=['/scr/01/kappler/dotDeeStage/bin/'];
34         ascStageDir=['/scr/01/kappler/dotDeeStage/ascii/'];
35         metaStageDir=['/scr/01/kappler/ulfemstage/'];
36     case 'GUMP' %typhoon@wr.usgs.gov
37         ULFEMpath = fullfile('gp', 'ulfem', 'ULFEM')
38         TOOLBOX='/gp/ulfem/ULFEM/';
39         username='jglen';
40         binStageDir=['/home/u1/jglen/dotDeeStage/bin/'];
41         ascStageDir=['/home/u1/jglen/dotDeeStage/ascii/'];
42         metaStageDir=['/home/u1/jglen/metaStage/'];
43         ulfemToolbox=['/home/u1/jglen/ulfemToolbox/'];
44     case 'SJMATLAB'
45         ULFEMpath = fullfile('/home', 'sjmatlab', 'repo2007', '
              developer',...
46             'kk', 'ULFEM', 'trunk')
47         %ULFEMpath='/home/sjmatlab/repo2007/developer/kk/ULFEM/
              trunk/';
48         MATpath = fullfile(ULFEMpath, 'src', 'MATLAB');
49         TOOLBOX = fullfile(MATpath, 'TOOLS');
50         username='kappler';
51         binStageDir=['/scr/01/kappler/dotDeeStage/bin/'];
52         ascStageDir=['/scr/01/kappler/dotDeeStage/ascii/'];
53         metaStageDir=['/scr/01/kappler/ulfemstage/'];
54         ulfemToolbox=['/home/u2/kappler/ULFEM/ulfemToolbox/'];
55     case 'AMPERE_windows' %AMPERE@dhcp.lbl.gov
56         %LOCAL DIRECTORIES
```

```
        ULFEMpath = fullfile('C:', 'ULFEM');
        TOOLBOX='C:\ULFEM\';
        username='kappler';
        %REMOTE (BSL) DIRECTORIES
        binStageDir=['/scr/01/kappler/dotDeeStage/bin/'];
        ascStageDir=['/scr/01/kappler/dotDeeStage/ascii/'];
        metaStageDir=['/scr/01/kappler/ulfemstage/'];
        ulfemToolbox=['/scr/em1/kappler/ULFEM/ulfemToolbox/'];
end
me=[username,'@crust.geo.berkeley.edu'];
addpath(TOOLBOX);
addpath_recurse(ULFEMpath)
SYSpath = [fullfile(ULFEMpath,'src','sys'),SLaSH];
COILpath=[fullfile(SYSpath,'COILS'),SLaSH];
SITEpath=[fullfile(SYSpath,'SITES'),SLaSH];
SCRpath = [fullfile(ULFEMpath,'scr'),SLaSH];
SCRPLOTpath = [fullfile(ULFEMpath,'plotscr'),SLaSH];
SCRPLOTFCpath = [fullfile(ULFEMpath,'plotscr','FC'),SLaSH];
DATApath = [fullfile(ULFEMpath,'data'),SLaSH];
METADATApath = fullfile(ULFEMpath,'src','metaData');
EPOCHSpath=[fullfile(METADATApath,'EPOCHS'),SLaSH];
FIGUREpath = fullfile(ULFEMpath,'FIGURES');
PYTHONpath = fullfile(ULFEMpath,'src','python')
if strmatch('PCWIN',computer)
    cohlin=find(ULFEMpath==':');
    %ULFEMpath=['/cygdrive/c',strrep(ULFEMpath(cohlin+1:end)
       ,'\','//')]
    syspath=['/cygdrive/c',strrep(SYSpath(cohlin+1:end),'\','//'
       )];
    sitepath=['/cygdrive/c',strrep(SITEpath(cohlin+1:end),'\','
       //')];
    scrpath=['/cygdrive/c',strrep(SCRpath(cohlin+1:end),'\','//'
       )];
```

```
86      scrplotpath=['/cygdrive/c',strrep(SCRPLOTpath(cohlin+1:end)
           ,'\','//')];
87      datapath=['/cygdrive/c',strrep(DATApath(cohlin+1:end),'\','
           //')];
88      metadatapath=['/cygdrive/c',strrep(METADATApath(cohlin+1:
           end),'\','//')];
89      epochspath=['/cygdrive/c',strrep(EPOCHSpath(cohlin+1:end),'
           \','//')];
90      figurepath=['/cygdrive/c',strrep(FIGUREpath(cohlin+1:end),'
           \','//')];
91  else
92      %ulfempath=ULFEMpath;
93      figurepath=FIGUREpath;
94      syspath=SYSpath;
95      sitepath=SITEpath;
96      datapath=DATApath;
97      scrpath=SCRpath;
98      scrplotpath=SCRPLOTpath;
99      scrplotfcpath=SCRPLOTFCpath;
100     metadatapath=METADATApath;
101     epochspath=EPOCHSpath;
102     pythonpath = PYTHONpath
103 end
104
105 listOfDirsToCheckAndBuildIfDNE={syspath,sitepath, datapath,
       scrpath,...
106     metadatapath,epochspath,scrplotfcpath};
107 for pth = listOfDirsToCheckAndBuildIfDNE
108     display(['checking if ', pth{1}, 'exists:'])
109     if exist(pth{1},'dir')
110         %display([pth{1},' exists'])
111     else
112         display([pth{1},' doesnt exist'])
113         mkdirCmd=['mkdir ',pth{1}]
```

```
114         eval(mkdirCmd)
115     end
116 end
117
118 if verbose
119     display(['Paths set Succesfully'])
120 end
121
122
123 %<Set other global variables>
124 samplingRateContainerMap = containers.Map({'D','B','L','V','500
    ','40','1','0.1'},...
125     {'500','40','1','0.1','D','B','L','V'})
126 %<\Set other global variables>
```

```
function UIC = siteChUIC(SITES,CHANNELS)
%intialize selected sites array [siteUID chUID]
uic=1;
iSite=1;
while iSite<numel(SITES)+1
    nCH=numel(CHANNELS{iSite});
    iCh=1;
    while iCh<nCH+1
        UIC(uic,1)=iSite;
        UIC(uic,2)=iCh;
        iCh=iCh+1;
        uic=uic+1;
    end
    iSite=iSite+1;
end

%save sys/SITES/UIC UIC
```

```
function l = split(d,s)

%L=SPLIT(S,D) splits a string S delimited by characters in D.  Meant to

%            work roughly like the PERL split function (but without any

%            regular expression support).  Internally uses STRTOK to do

%            the splitting.  Returns a cell array of strings.

%

%Example:

%    >> split('_', 'this_is___a_/_string/_//')

%    ans =

%        'this'    'is'    'a'    'string'   []

%

%Written by Gerald Dalley (dalleyg@mit.edu), 2004


l = {};

while (length(s) > 0)

    [t,s] = strtok(s,d);
```

```
32
33      l = {l{:}, t};
34
35  end
```

```
function parts = strsplit(splitstr, str, option)
%STRSPLIT Split string into pieces.
%
%   STRSPLIT(SPLITSTR, STR, OPTION) splits the string STR at every occurrence
%   of SPLITSTR and returns the result as a cell array of strings.  By default,
%   SPLITSTR is not included in the output.
%
%   STRSPLIT(SPLITSTR, STR, OPTION) can be used to control how SPLITSTR is
%   included in the output.  If OPTION is 'include', SPLITSTR will be included
%   as a separate string.  If OPTION is 'append', SPLITSTR will be appended to
%   each output string, as if the input string was split at the position right
%   after the occurrence SPLITSTR.  If OPTION is 'omit', SPLITSTR will not be
%   included in the output.

%   Author:      Peter J. Acklam
%   Time-stamp:  2004-09-22 08:48:01 +0200
%   E-mail:      pjacklam@online.no
%   URL:         http://home.online.no/~pjacklam

   nargsin = nargin;
   error(nargchk(2, 3, nargsin));
   if nargsin < 3
      option = 'omit';
   else
      option = lower(option);
   end

```

```
28      splitlen = length(splitstr);
29      parts = {};
30
31      while 1
32
33         k = strfind(str, splitstr);
34         if isempty(k)
35            parts{end+1} = str;
36            break
37         end
38
39         switch option
40            case 'include'
41               parts(end+1:end+2) = {str(1:k(1)-1), splitstr};
42            case 'append'
43               parts{end+1} = str(1 : k(1)+splitlen-1);
44            case 'omit'
45               parts{end+1} = str(1 : k(1)-1);
46            otherwise
47               error(['Invalid option string -- ', option]);
48         end
49
50
51         str = str(k(1)+splitlen : end);
52
53      end
```

```
classdef timeSeries
    properties
        data
        samplingRate
        duration
        digitizer
        instrument
    end


end
```

```
function [] = warnBox(mssg)
%create a window to prompt the user for a name of a new array
posn=[380,550,200,100];
hFig=figure(...       % SOME FIG
        'MenuBar','none', ...
        'Name', 'WARNING', ...
        'NumberTitle','off',...
        'Position',posn,...
        'units','normalized',...
        'tag','warnBox');
    %get(hFig)
    x=0;
    y=0.7;
    hButton = uicontrol('Style','pushbutton','Parent',hFig,...
                        'String','OK','Callback',@closeWarn);
    txt=uicontrol(...
    'units','normalized',...
    'position',[0,0.4,1,0.5],...
    'style','text',...
    'string',mssg,...
    'fontsize',12,...
    'Parent',hFig);

    function closeWarn(source,event)
        close(hFig)
    end
end
```

```
function [YYYY MM DD] = yesterday()

S=date;
N=datenum(S);
ds=datestr(N-1,'yyyy mm dd');
ks=ksplit(ds,' ');
YYYY=ks{1};
MM=ks{2};
DD=ks{3};
```

```
function zps = zeroPadStr(x, fin_length)
%takes as input an integer x and returns a string which is the
   integer
%zero-padded in front to be length "fin_length"
if length(x)>fin_length
    display(['you are not using zeropad str the right way, it
       adds zeros, NOT subtracts them!'])
    zps=0;
    return
end
x_str=num2str(x);
zro='0';
if length(x_str)==fin_length
    zps=x_str;
else
    for i=1:fin_length-length(x_str)
        x_str=[zro,x_str];
    end;
end;
zps=x_str;
```