Low Frequency Tutorial

1 Introduction

The purpose of this tutorial is to help observers become familiar with observing at LWA1 and how to work with the various data formats. Specifically, this tutorial emphasizes the LWA Software Library (LSL) Python module and the three extensions that are currently available. The data for this tutorial can be found on the LWA Users Computing Facility in the `/data/network/SummerSchool/` directory.

Before beginning this tutorial it is recommended that an additional environment variable be set to the location of the extensions. On the LWA Users Computing Facility this would be:

```
export LEXTS=/usr/local/extensions
```

Many of the exercises below create various intermediate data files and it is best to create a personal or group directory under `/data/local/` to help keep your files separate. Also, in many of the commands given below we use a backslash, '\', to continue long commands on subsequent lines. These commands should be entered on a single line in the terminal without the backslash. Finally, the tutorial assumes a basic knowledge of UNIX command line skills. If you are unfamiliar with this environment, https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners has a good overview of the most important commands.

Additional information about LWA, LWA1, and LSL can be found at the following websites:

- LWA website http://lwa.unm.edu,
- LWA proposal information http://www.phys.unm.edu/~lwa/proposals.html,
- LSL website https://github.com/lwa-project/lsl,
- LWA Memo Series http://www.phys.unm.edu/~lwa/memos/, and
- LWA Data Archive http://lda10g.alliance.unm.edu.
- LWA Docker Images https://github.com/lwa-project/docker_stacks

If you plan to make extensive use of LSL then you may want to make the effort to download and compile LSL and associated software tools (e.g., sessionGUI, TEMPO, PRESTO, psrcat, psrchive, dspsr, etc.) on your computer.

Otherwise a simpler way to get started is to make use of Docker containers. The LWA Docker images are described in the `README.md` on the LWA Docker Images page. For the tutorial the most useful containers are `lwaproject/lsl:pulsar` and `lwaproject/lsl:baseline`. You will want to download the following packages and data:

1. Xpra (for displaying graphics on your computer) – https://xpra.org/trac/wiki/Download
2. Docker – https://www.docker.com/products/docker-desktop
3. LWA data – http://lda10g.alliance.unm.edu/tutorial/
   - B0329+54 – 357 MB
   - CGPs – 23 GB
   - Interferometer – 10 GB
   - Jupiter – 266 GB
   - Meteors – 357 GB
   - Sun – 4.2G
   - UnknownPulsar – 12 GB

You can retrieve data using a web browser or the `wget` command. However, the total is 671 GB so maybe you want to look through the tutorial first and decide what data sets you would like to work on.

A sample Docker/Xpra session might look something like this:

1. Download some LWA data:

```
mkdir tutorial
cd tutorial
wget https://lda10g.alliance.unm.edu/tutorial/B0329+54/056770_000044687
cd ..
```

2. Start Docker:

```
docker pull lwaproject/lsl:pulsar
docker run -it -p 10000:10000 -v /Users/gtaylor/lwa/tutorial:/data lwaproject/lsl:pulsar
```

3. Inside Docker:

```
export DISPLAY=:100
xpra start --bind-tcp=0.0.0.0:10000 --daemon=yes $DISPLAY
```

4. Outside Docker: Set the /path/to/your/Xpra attach tcp://localhost:10000 (on a Mac this happens through a GUI interface in Xpra).
5. Inside Docker: First make a schedule (tutorial §2).

```
cd /data
 ~/session_schedules/sessionGUI.py
```

6. Start working on a known pulsar (section 3.1):

```
~/commissioning/DRX/HDF5/drspecFileCheck.py 056770_000044687
~/commissioning/DRX/HDF5/drspec2hdf.py 056770_000044687
~/commissioning/DRX/HDF5/plotHDF.py 056770_000044687-waterfall.hdf5
...
```

2 Observing with LWA1

Goal: Setup a DRX (beam forming) observation to look for the pulsar B0329+54.

Estimated Time: 10 minutes.

1. Start the LWA observation scheduling GUI that is part of the `SessionSchedules` extension with:

   ```
   $LEXTS/SessionSchedules/sessionGUI.py
   ```

   This will bring up a window that will help in preparing the session definition files (SDFs) used for scheduling observations at LWA1.

2. Define a new set of observations using the `File -> New` menu or by clicking the "New Project" icon ( ). This will open up the observer and project information window (Figure 1). In this window you can define your name, the project[1], the LWA1 mode to be used to collect the data, whether or not to use the data recorder spectrometer, and how the data are be to returned to you. The nomenclature is that a project is made of one or more sessions. Each session uses one of the five LWA1 outputs (four DRX beams and TBW or TBN) and consists of multiple sequential observations. For DRX observations you can choose to observe a static position or you may track an RA/Dec. position, the Sun, or Jupiter. It is a good idea to check the current list of observing constraints posted at http://www.phys.unm.edu/~lwa/astro/currentissues.html to make sure that there are no issues that would interfere with the observations.

   For the observations of B0329+54 you will want to select the DR spectrometer mode. The default values for the number of channels is sufficient for this pulsar but the integration time, the number of FFT windows per integration, is too long to resolve the pulse. You should reduce the number of FFT windows per integration down to a value that is equivalent to about 40 ms. The currently supported modes for the data recorder spectrometer can be found at http://www.phys.unm.edu/~lwa/astro/scheds/spec.html.

3. After defining the observer information, create an observation to look at the pulsar B0329+54. This can be done through the `Observations -> Add -> DRX - RA/Dec` menu or the "New RA/Dec Target" icon (R). This will add a new blank observation to the window. To find the coordinates of B0329+54, enter the source into the target field, select the observation via the check box on the left side of the screen, and choose `Observations -> Resolve Selected`.

4. Define the parameters of the observation (observing time, tuning, bandwidth, etc.). For the DRX mode each pointing has two independent tunings that can be tuned anywhere in the 10 to 88 MHz band of LWA1. To get the list of filter codes, use the `Help->Filter Codes` dialog. For pulsar observations it is good to space the tunings by the bandwidth so that the two tunings can be combined together and jointly analyzed, e.g., if you are using filter #7, space the tunings by 19.6 MHz.

5. Check the visibility of the source using the `Observations -> Session at a Glance` feature. This will plot the elevation of the source throughout the observation.

6. If you have selected raw data to be returned (DRSUs or USB hard drives), check the data volume of your observations using the `Data -> Estimated Data Volume` feature. This will display a window with the raw data volume for each of the observations as well as the session total.

[1] Observer and project IDs are assigned by LWA1 when proposals are accepted. Observer IDs are numeric and project IDs consist of two letters followed by a three digit number, e.g., LF001.

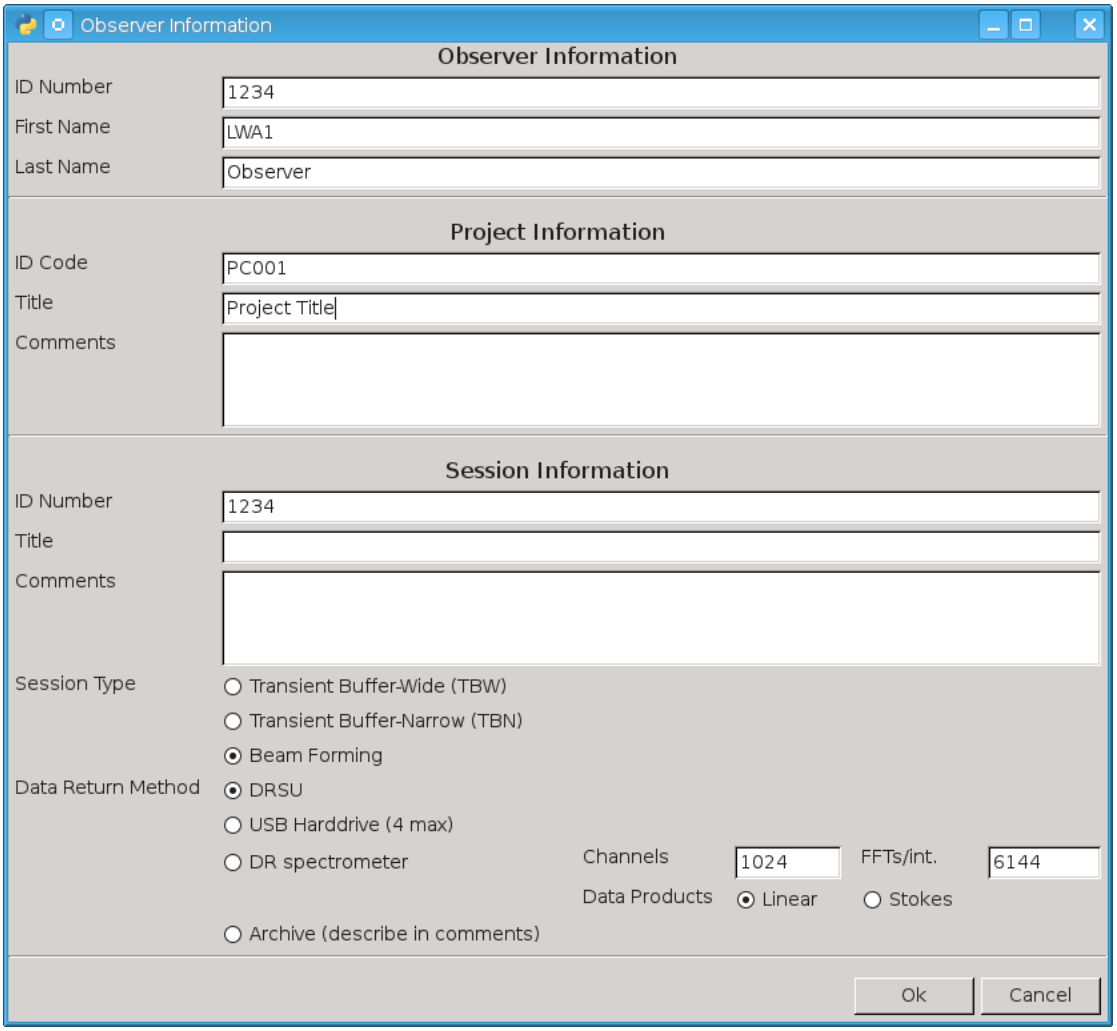

Figure 1: The `sessionGUI.py` observer and project information window. Here you can define your name, the project, the LWA1 mode to be used to collect the data, whether or not to use the data recorder spectrometer, and how the data are be to returned to you.

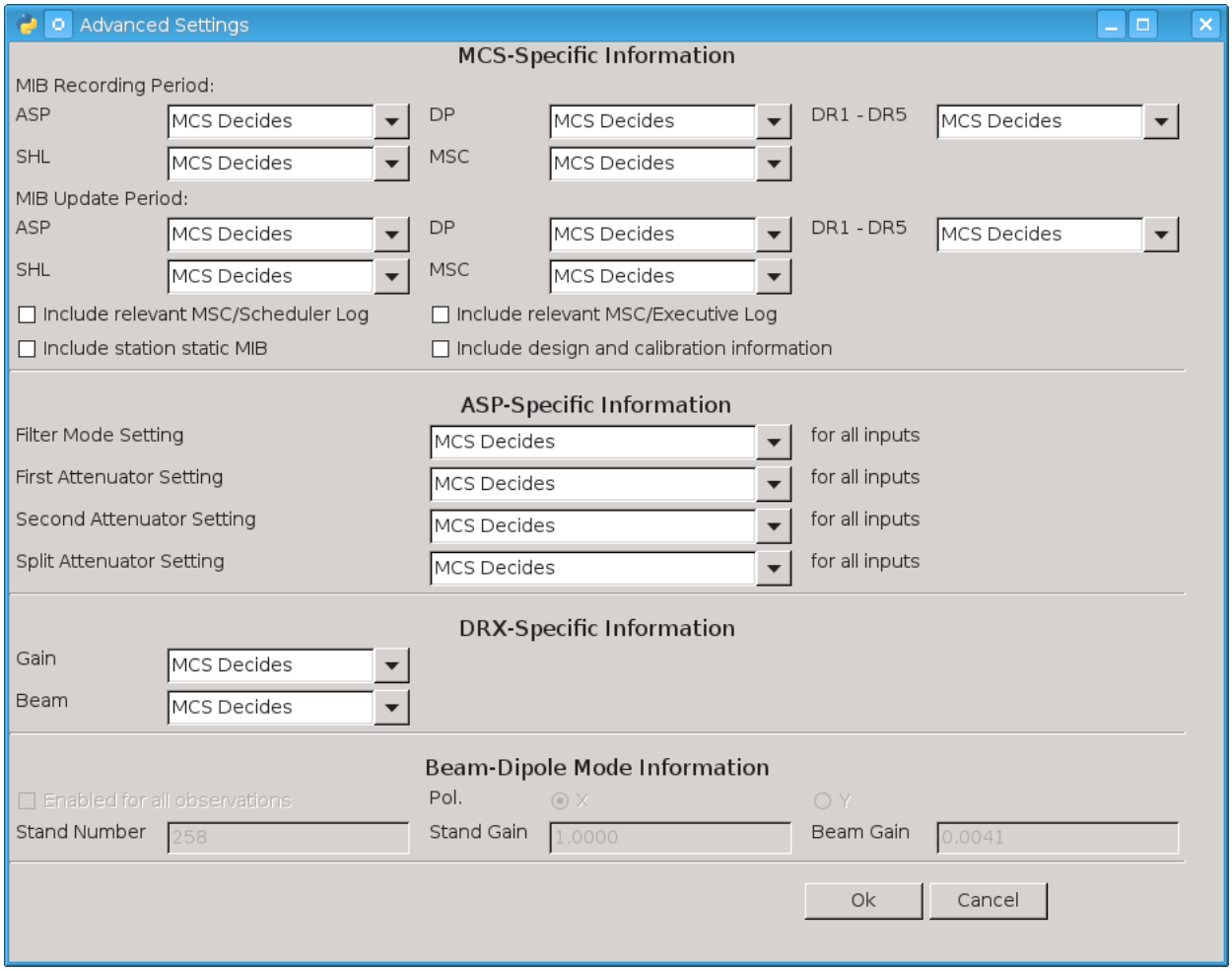

Figure 2: The `sessionGUI.py` advanced observation settings window. Here you can define the gain used for the observations and which beam DRX observations are to use.

7. If you are observing with DRX, make sure that you have assigned a particular beam for the session to use. This is done through the `Observations->Advanced Settings` dialog shown in Figure 2.

8. Once you are happy with the observation setup, save the observation to disk. Before the file is saved it will be validated. If any of the observations have invalid parameters, they will be highlighted in red. The terminal window will also list the invalid parameters. Be sure to correct any problems before you exit `sessionGUI.py`.

9. SDFs are submitted to LWA1 for observation through the SDF validator at `http://fornax.phys.unm.edu/lwa/validator/index.html`. This page accepts multiple SDFs or a tarball containing multiple SDFs and validates each file. Any errors in the SDFs will also be noted as part of the validation process. Valid files are sent to the observing queue to be run by the LWA1 operator.

3 Pulsars at Low Frequencies

3.1 Observing a Known Pulsar in "Spectrometer" Mode

Goal: Use 15 minutes of data from a single LWA1 beam to detect the pulsar B0329+54. In the process get a feel for how to examine beam data.

Estimated Time: 10 minutes.

1. Take a look at the power and clipping levels in the spectrometer data file using scripts in the `Commissioning` extension. You can get an overview of the power levels and clipping via:

   ```
   $LEXTS/Commissioning/DRX/HDF5/drspecFileCheck.py 056770_000044687
   ```

   which will report both quantities. Optimal values of power are between seven and ten. Below seven some dynamic range is lost because the signal strength is not enough to populate many bit levels in the data. Above ten the clipping starts to become severe and the dynamic range is also compromised. However, high values for the power and clipping may be transient, e.g., radio frequency interference (RFI), and do not necessarily mean that the observation is useless.

   If the power or clipping is high for either tuning, convert the data to an HDF5 file and interactively plot the results using:

   ```
   $LEXTS/Commissioning/DRX/HDF5/drspec2hdf.py 056770_000044687
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py 056770_000044687-waterfall.hdf5
   ```

   Inside `plotHDF.py` you can view both tunings and polarizations in a variety of ways and look at how the data evolve in time and frequency.

2. Now convert the file into PSRFITS format in preparation for folding the pulsar using PRESTO.

   ```
   $LEXTS/Pulsar/writePsrfits2FromDRSpec.py -s B0329+54 056770_000044687
   ```

   This will create two files: "drx_56770_B0329+54_b2t1_0001.fits" and "drx_56770_B0329+54_b2t2_0001.fits", one for each tuning. If the tunings are spaced by the sample rate the two PSRFITS files can be combined with `combine_lwa` and jointly analyzed in PRESTO:

   ```
   combine_lwa2 -o drx_56770_B0329+54_b2 drx_56770_B0329+54_b2t2_0001.fits \
        drx_56770_B0329+54_b2t1_0001.fits
   ```

   This will create a new file named "drx_56770_B0329+54_b2_0001.fits" that contains the data from the two input files.

3. Although the LWA1 site is relatively free of RFI it is often a good idea to create a mask to remove any RFI that may be present in the data. To create this mask use the PRESTO `rfifind` utility:

   ```
   rfifind -time 2.0 -o mask_B0329+54_b2 drx_56770_B0329+54_b2_0001.fits
   ```

   The resulting mask includes a postscript file that can be viewed with `evince` or `ghostscript`.

4. Finally, incoherently dedisperse the PSRFITS file and fold the data at the pulsar's period of ≈715 ms using the PRESTO `prepfold` utility:

```
prepfold -psr 0329+54 -mask mask_B0329+54_b2_rfifind.mask -nsub 128 \
     drx_56770_B0329+54_b2_0001.fits
```

After folding, `prepfold` will generate an analysis plot showing the pulse strength over time, the constraints on the period and period rate of change, and the constraints on the dispersion measure (DM).

3.2 Observing a Unknown Pulsar in "Raw Data" Mode

Goal: Use 10 minutes of data from a single LWA1 beam to detect an unknown pulsar.

Estimated Time: 30 minutes.

1. Start by converting the raw DRX data file in `/data/network/SummerSchool/UnknownPulsar/` to PSRFITS via:

   ```
   $LEXTS/Pulsar/writePsrfits2.py 056227_000024985_DRX.dat
   ```

   and then create RFI masks for the two PSRFITS files similar to before.

2. Since the DM of the pulsar is only constrained between 5 and 15 pc cm^{-2}, search with PRESTO by first incoherently dedispersion files at a collection of trial DMs.

   ```
   prepsubband -lodm 5 -dmstep 0.1 -numdms 100 -nsub 128 -mask mask_rfifind.mask  \
        -numout #### -o <output_base_name> psrfitsfilename
   ```

   This script will create a collection of dedispersed files with a base name of ¡output_base_name¿. You can figure out the value used for `-numout` by looking at the number of points that `rfifind` reports when it starts, and then determining the closest power of 2. The next largest power of two can be found with Python through:

   ```
   python -c 'import math; print 2**math.ceil(math.log(#####)/math.log(2))'
   ```

   For each trial DM file search for a pulsar using:

   ```
   accelsearch -zmax 0 -numharm 16 <name_to_search>
   ```

   Since `prepsubband` generates many files it is convenient to process them all at once with the `xargs` command:

   ```
   ls <output_base_name>*.dat | xargs -n1 accelsearch -zmax 0 -numharm 16
   ```

   which will run `accelsearch` on each file found by the `ls` command.

   Sift through the various pulsar candidates found by `accelsearch` to find the approximate DM and period for the pulsar using the PRESTO utility `ACCEL_sift.py`:

   ```
   python $PRESTO/python/ACCEL_sift.py *.cand
   ```

   The pulsar should show up across several closely spaced DMs at the same period.

3. Once the approximate DM and period are known, fold the original data at the pulsar's period. This searches over a small range of DM and period to get the best fit. Note: `prep fold` takes DM values in pc cm^{-3} and periods in seconds.

   ```
   prepfold -dm ##.# -p ##.# psrfitsfilename -nsub 128 \
        -mask mask_rfifind.mask
   ```

   Note the final DM and period.

3.3 Improving the Detection Using Coherent Dedispersion

Goal: Refine the detection of the unknown pulsar in Section 3.2 using coherent dedispersion.

Estimated Time: 30 minutes.

1. Up to this point only incoherent dedispersion has been used for the pulsar analysis. However, the dispersion introduced by the interstellar medium can be removed from the raw data more precisely using coherent dedispersion if the DM is known ahead of time. To apply coherent dedispersion to LWA1 DRX data use the `writePsrfits2D.py` script:

   ```
   $LEXTS/Pulsar/writePsrfits2D.py ##.# 056227_000024985_DRX.dat
   ```

   where the first argument to the script is the DM found above. After the file has been converted, create RFI masks for the two PSRFITS files similar to sections 3.1 and 3.2.

2. Fold the data at the pulsar's period using `prepfold`. This will also search over a small range of DM and period to get the best fit.

   ```
   prepfold -dm ##.# -p ##.# psrfitsfilename -nsub 128 -mask mask_rfifind.mask
   ```

   Note the final DM and period. Is the detection stronger with coherent dedispersion?

3.4 Measuring Faraday Rotation

Goal: Use data from the LWA Pulsar Archive to determine the rotation measure for on PSR B0950+08.

Estimated Time: 30 minutes.

1. The data for this exercise originally come from the LWA Pulsar Archive[2]. This archive contains observations of the 107 pulsars detected using LWA1 and has data that spans several years for some pulsar. The data on PSR B0950+08 comes from January of 2018 and is at a frequency of 64.5 MHz. All of additional data available for this pulsar on this date can be found at http://lda10g.alliance.unm.edu/PulsarArchive/B0950+08/58132/.

2. Since many of the pulsar analysis tools used in this exercise default to writing in the same directory as the data it is convenient to create a local copy of data. Do this by copying the entire `/data/network/SummerSchool/PulsarRM/` directory.

3. Using your local copy of the data, flag obvious radio frequency interference (RFI) using the `median_6.psh` script.

   ```
   ./median_6.psh 58132_B0950+08_64.5MHz.0000.ar
   ```

   This script finds and flags data that are more than 6σ from the median and writes the results to a new file with a `.zz` extension.

4. Next, integrate over all the periods (time-bins) of the pulsar. This can be done using the `PSRCHIVE` command `pam`:

[2] http://lda10g.alliance.unm.edu/PulsarArchive

```
pam -T -e tscrunch 58132_B0950+08_64.5MHz.0000.zz
```

Here, the `-T` flag tells `pam` to integrate over times and the `-e` flag sets the output extension to `.tscrunch`.

5. In order to convert to Stokes parameters we will again make use of the `pam` command, this time with the `-S` flag:

```
pam -S -e tscrunchS 58132_B0950+08_64.5MHz.0000.tscrunch
```

The four Stokes parameters (I, Q, U, and V) can be plotted to view the effects of Faraday Rotation using:

```
./plot_pol 58132_B0950+08_64.5MHz.0000.tscrunchS
```

The Faraday Rotation manifests as frequency dependent banding in the Stokes Q (second plot from left) and U (third plot from left) plots of frequency vs. pulsar phase.

6. Now attempt to fit a rotation measure (RM) to the data. We know the RM of PSR B0950+08 is ≈2, but the ionosphere can change that up to ±1.5, so we will look at RMs from –5 to 5 and test 256 values of RM to get a good fit:

```
rmfit -D -K 58132_B0950+08_64.5MHz.0000_rmfit.eps/CPS -m -5,5,256 \
     58132_B0950+08_64.5MHz.0000.tscrunch
```

This will create a plot of trial RMs versus total polarization and attempt to fit a Gaussian to it. The plot is stored as a `.eps` file that can be viewed with `evince`.

7. Using the best-fit value for the RM, apply it to the data and re-generate the plot of the Stokes parameters:

```
pam -R #.### -e tscrunchRM 58132_B0950+08_64.5MHz.0000.tscrunch
pam -S -e tscrunchRMS 58132_B0950+08_64.5MHz.0000.tscrunchRM
./plot_pol 58132_B0950+08_64.5MHz.0000.tscrunchRMS
```

In the first command of the above "#.###" is the best-fit RM value found with `rmfit`.

8. The best-fit RM value found above includes both the interstellar medium along the line of sight as well as a contribution from the Earth's ionosphere. Since we are interested in understanding the interstellar medium we need to remove to ionosphere component. This is traditionally done using low temporal and spatial resolution global models of the total electron content (TEC) in the ionosphere and then interpolating these to the exact observing time and location.

4 Studying the Transient Universe

4.1 Jovian Bursts

Goal: Search a two hour DRX observation tracking Jupiter to identify Jovian bursts.

Estimated Time: 45 minutes

1. The data being analyzed here are part of a commissioning test observation of Jupiter. Unlike most LWA1 beam formed data, this file contains a single 19.6 MS/s tuning. Since reducing the raw data, even at a reduced spectral and temporal resolution, takes about two hours, a pre-reduced file named "055917_000007389_Jupiter-waterfall-low.hdf5" has been provided in `/data/network/SummerSchool/Jupiter/`. This file has a temporal resolution of five seconds and a spectral resolution of 4.8 kHz. Display this file using:

   ```
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py \
       055917_000007389_Jupiter-waterfall-low.hdf5
   ```

   and look for evidence of bursts during the file. For each burst note the time offset, UTC start time, and duration. The plotting script also allows for the color map, stretch, and scale to interactively adjusted which may help identify events.

2. After you have identified any bursts in the file, produce higher resolution versions of the event using `hdfWaterfall.py`. Since the bursts are highly circularly polarized try processing the file to Stokes parameters:

   ```
   $LEXTS/Commissioning/DRX/HDF5/hdfWaterfall.py -s ##.# -d ##.# -a 0.01 -k \
       055917_000007389_Jupiter.dat
   ```

   where "-s" sets the relative offset from the beginning of the file, "-d" specifies the duration to process, "-a" sets the integration time to 10 ms, and "-k" sets the processing mode to Stokes I, Q, U, and V.

3. Plot the higher resolution files with `plotHDF.py` to get a better view of the burst structure. Depending on the size of the higher resolution file you may need to display a sub-set of the data.

4.2 Solar Bursts

Goal: Search a three hour spectrometer file from the LWA Data Archive for solar bursts.

Estimated Time: 10 minutes

1. The data being analyzed here are part of a program to search for prompt emission from gamma ray bursts (GRB). The data are from Fermi Trigger 405367742[3] which turned out not to be a GRB but, in fact, a solar burst. The data are taken at a temporal resolution of 40 ms which makes displaying the full file difficult. Start by creating a lower resolution version of the HDF5 file in `/data/network/SummerSchool/Sun/` using the `decimateHDF.py` script:

   ```
   $LEXTS/Commissioning/DRX/HDF5/decimateHDF.py -s 5 20 056601_000024625-waterfall.hdf5
   ```

   This will create a new HDF5 file that has one-fifth the spectral resolution and one-twentieth the temporal resolution which will be more suitable for displaying.

2. Examine the file using the `plotHDF.py` script and check both tunings for evidence of bursts.

   ```
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py 056601_000024625-waterfall-decim.hdf5
   ```

[3] http://gcn.gsfc.nasa.gov/other/405367742.fermi

For each burst you find, note the time range of the burst, both in UTC and relative to the start of the observation. Once again, the plotting script also allows for the color map, stretch, and scale to interactively adjusted which may help identify events.

3. For each of the bursts identified, display the burst in the full-resolution data in `plotHDF.py`. The script supports displaying a subset of the data via the "-s/–skip" and "-d/–duration" flags:

   ```
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py -s ## -d ## 056601_000024625-waterfall.hdf5
   ```

4.3 Giant Pulses from the Crab Pulsar

Goal: Search a raw beam forming observation for giant pulses from the Crab pulsar.

Estimated Time: 15 minutes

1. The data being analyzed here are part of a project to characterize the low frequency giant pulsed emission (CGPs) from the Crab pulsar . The data consist of tunings at 60 and 76 MHz with a sample rate of 19.6 MS/s. To begin searching for the CGPs, first convert the first 100 seconds of data in `/data/network/SummerSchool/CGPs/` to an HDF5 file with 16,384 channels and 40 ms integrations:

   ```
   $LEXTS/Commissioning/DRX/HDF5/hdfWaterfall.py -a 0.04 -l 16384 -d 100 056598_000003517
   ```

   The large number of channels will improve the dedispersion by reducing the amount of DM smearing per channel.

2. After the file has been converted, display it using `plotHDF.py` to get an idea of the overall interference environment and to check for any obvious pulses. The CGPs will appear as curves moving from the upper left to the lower right of the waterfall due to dispersion.

   ```
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py 056598_000003517-waterfall.hdf5
   ```

   Masking RFI will help identify pulses. The `calculateSK.py` utility provides a simple way to mask RFI based on spectral kurtosis. To create and apply a basic mask run:

   ```
   $LEXTS/Commissioning/DRX/HDF5/calculateSK.py -g 056598_000003517-waterfall.hdf5
   ```

   and examine the updated file in `plotHDF.py`. You can also manually create and edit the mask stored in an HDF5 file with `plotHDF.py`. To do this use the `Mask` menu. You may want to try different masking parameters via the `Mask -> Adjust Masking Parameters` dialog, particularly try adjusting the number of sections used for spectral kurtosis. You can also manually adjust the mask by clicking on points to be removed with the right mouse button. After creating an acceptable set of masks save them to the HDF5 file via `File -> Save`.

3. Since the Crab pulsar has a DM of ≈56.79 pc cm^{-3} the giant pulses will be more obvious if the data are first incoherently dedispersed and then spectrally averaged. To perform the dedispersion use the `dedisperseHDF.py` utility:

   ```
   $LEXTS/Commissioning/DRX/HDF5/dedisperseHDF.py 56.79 056598_000003517-waterfall.hdf5
   ```

   and then average using `decimateHDF.py`:

```
$LEXTS/Commissioning/DRX/HDF5/decimateHDF.py -s 256 1 \
     056598_000003517-waterfall-DM56.7900.hdf5
```

This will reduce the number of frequency channels to 64.

4. Display the dedispersed and decimated data using `plotHDF.py`:

   ```
   $LEXTS/Commissioning/DRX/HDF5/plotHDF.py 056598_000003517-waterfall-DM56.7900-decim.hdf5
   ```

   The CGPs will appear in the waterfall as horizontal bursts. To more easily identify the CGPs, you may need to turn on bandpassing via the "Bandpass" menu and flag channels that are dominated by RFI.

5 All-Sky Imaging

Goal: Use TBN data to create images of the sky and search for meteor echoes.

Estimated Time: 30 minutes.

1. TBN data consists of a continuous stream of 8-bit I/8-bit Q raw voltage data sampled at up to 100 kS/s from all 520 antennas. The data used here are part of an experiment to study meteor streams using the echoes of analog TV stations at 55.25 MHz (channel 2). Since the signals of interest here are normally considered RFI in astronomical observations it is a good idea to make sure that the data are not clipped. Start by using the `tbnFileCheck.py` to check for saturation in the data in `/data/network/SummerSchool/Meteors/`:

   ```
   $LEXTS/Commissioning/TBN/tbnFileCheck.py 056761_000099453
   ```

   This will display information about the data for Stand #10.

2. Next, examine several spectra from individual stands to make sure that the carrier at 55.25 MHz is detectable using the `tbnSpectra.py` script. Be sure the skip a few minutes into the file to get an idea of what the "typical" spectrum looks like.

   ```
   tbnSpectra.py -s 120 -k 10,173,258 056761_000099453
   ```

   In the above command, the "-s" option sets how many seconds to skip into the file and the "-k" flag sets which stands to keep for the analysis. If this flag is omitted, plots from all 520 antennas will be displayed. The plots created are labeled by the stand number, polarization (in parenthesis), the digital processor digitizer number, and the combined status code (in brackets).

3. In order to form an image the data need to be correlated. LSL provides an FX correlator that can be accessed with the `correlateTBN.py` script. Since correlation of the data is computationally intensive, start by correlating only 1 second of data:

   ```
   correlateTBN.py -o 120 -s1 -t1 -2 -l 32 056761_000099453
   ```

   In the above, the "-o 120" flag offsets two minutes into the file, the "-s" flag sets the number of integrations to generate, and the "-t" flag sets in the integration time in seconds. The "-2" flag specifies the number of polarization products to compute: XX and YY while the "-l 32" flag tells the correlator to compute 32 channels across the 100 kHz bandwidth. The results of the correlation are written out to a FITS IDI file named "056761_000099453.FITS_1".

4. The FITS IDI file generated can be imaged with the `imageIDI.py` script that is included with LSL. This script uses the AIPY w-projection method to deal with the non-zero w terms that arise from the wide field of view at low frequencies:

   ```
   imageIDI.py 056761_000099453.FITS_1
   ```

   The script will generate a zoom-able image of each polarization product and add a coordinate system.

5. One method of looking for meteor echoes is to difference sequential images to remove the relatively constant sky background. To do this use the `correlate.sh` script in `/data/network/SummerSchool/Meteors/` to correlate 11 one-second chunks of data. Be sure to remove the FITS IDI file created in the previous step before running this script.

   ```
   export MEXTS=/data/network/SummerSchool/Meteors
   $MEXTS/correlate.sh 056761_000099453
   ```

   This will create a sequence of files named:

   - 056761_000099453.FITS_SL0
   - 056761_000099453.FITS_SL1
   - 056761_000099453.FITS_SL2
   - ...

6. Difference sequential image pairs using the `diffIDI.py` script in `/data/network/SummerSchool/Meteors`. This is a modified version of `imageIDI.py` that takes two FITS IDI files as arguments and subtracts the second from the first. Use the plots created by the script to identify possible meteor trail echoes.

   ```
   $MEXTS/diffIDI.py 056761_000099453.FITS_SL1\
        056761_000099453.FITS_SL0
   ```

7. Since the `diffIDI.py` script does not have an interactive display it has the option to export the image to a multi-extension FITS image file that can be viewed in `ds9`. To export the images, use the "-f" option:

   ```
   $MEXTS/diffIDI.py -f diff-image.fits 056761_000099453.FITS_SL1 \
        056761_000099453.FITS_SL0
   ```

   and then open the file in `ds9`:

   ```
   ds9 diff-image.fits
   ```

6 Single Baseline Interferometer

Goal: Use data from the LWA correlator to monitor a pulsar

Estimated Time: 45 minutes.

1. We will run AIPS inside a docker container. First we have to retrieve the Docker container. Here is how:

```
docker pull lwaproject/lsl:baseline
docker run -it -p 10000:10000 -v /path/to/data:/data lwaproject/lsl:baseline
```

See the instructions about connecting to the xpra server in §1. Now start up AIPS with:

```
aips
```

When prompted for your user number enter your favorite number between 100 and 1000. A TV display should start up automatically.

2. Visibility data from the LWA correlator is contained in a UVFITS data set. Inside the UVFITS file are cross correlation amplitudes and phases in various combinations of polarizations and frequency channels over the course of the observing run, which in turn consists of multiple scans of sources. For the example dataset considered here, there are two IFs or spectral windows, one centered at 53 MHz and the other at 73 MHz, each with 512 channels across a 19.6 MHz band. The linear polarizations of the LWA beams have been converted to circulars (RR, LL, RL, LR) for ease of processing. This particular observation also used two beams – a calibration beam that looked at calibrators throughout the run, and a source beam that looked at a combination of phase check sources and our target, the strong pulsar B1919+21. The scheduling files were created using the `swarmGUI.py` in LSL which creates interferometry definition files `.idf` files which are conceptually very close to the `.sdf` files described in §2. More details about setting up interferometry observations can also be found on the LWA web pages at http://www.phys.unm.edu/~lwa/singleB_tutorial.pdf.

   Start by loading the data into AIPS using the task FITLD. The calibrator beam is called `B1919CAL.LWASV` and the source beam is called `B1919SRC.LWASV`, both in `/data/network/SummerSchool/Interferometer/`.

```
task 'fitld'
datain '/data/B1919CAL.LWASV
outn 'B1919CAL'; outcl 'LWASV'; ncount 1; outdi 1
go fitld
datain '/data/B1919SRC.LWASV
outn 'B1919SRC'; outcl 'LWASV'; ncount 1; outdi 1
go fitld
```

   This assumes that the variable '/data' was mounted when you started docker.

3. The data loaded into AIPS will have calibration information in CL table 2, bandpass information in BP table 1 and flagging information in FG table 1. If you want to follow the steps below to derive the calibration then you should first delete CL table 2 and BP table 1 for both files using EXTDEST. Remove also SN table 1 from the calibration beam. Keep the FG table as this is automatically generated.

4. To familiarize yourself with the data use LISTR with optyp 'SCAN' on both files and look at what is there. You should see the target source, PSR B1919+21 in the source beam.

5. Perform fringe fitting using FRING on 3C295 in the calibration beam. You just need 1 good minute of data for this.

```
task 'fring'
getn i
aparm 0 0 0 0 0 2 6
dparm 1 5000 0 0.96 0
calsour '3C295',''
solint 1
timer 0 9 5 0 0 9 6 0
go fring
```

where 'i' is the catalog number of the calibrator beam.

6. Use LISTR with optyp 'GAIN' to look at the delays and see that they are reasonable.

7. Next, a little AIPS magic to get amplitudes in the ballpark and create a calibration table with the delays.

```
  task 'sncor'
  getn i
  opcode 'mula'; sncor 10 0
  go sncor; wait sncor
*
  task 'clcal'
  getn i
  opcode ' '; interp '2pt'; bparm 0; snver 1; inver 1;
  gainver 1; gainuse 2; refant 51
  go clcal
```

8. Lets take a look at the amplitudes and phases before and after calibration to see that we have done the right thing.

```
  task 'possm'
  docalib 1; gainuse 1; aparm 0;
  bif 1; eif 1; bchan 1; echan 0
  source '3c295',''
  timer 0 9 5 0 0 9 5 30
  bif 2; eif 2; stokes 'rr'
  plver 0
  go possm; wait possm;
  go tvpl; wait tvpl
```

   Gainuse 1 will apply no calibration, so you can see the effect of the delays. Then gainuse 2 will apply the delay calibration so see what that looks like.

9. Now its time for bandpass calibration using BPASS and then we will look at the results using the weirdly named POSSM task.

```
  task 'bpass'
  getn i
  timer 0
  calsour '3c295',''
  go bpass; wait bpass
*
  task 'possm'
  timer 0
  docalib 1; gainuse 2; aparm 0; aparm(8) 2;
  bif 1; eif 1; bchan 1; echan 0
  source '3c295',''
  plver 0
  go possm; wait possm
  go tvpl;
  wait tvpl
  bif 2; eif 2;
  plver 0
  go tvpl; wait tvpl
```

10. At this point we have finished with the multi-channel calibration (delays and bandpass) so we can now average up the data in frequency. We can also copy the calibration over to the source beam (with catalog entry j) and average that up in frequency as well.

```
  task 'tacop'
  getn i
  geto j
  inext 'bp'; inver 1; outver 1; ncount 1; go tacop
  inext 'cl'; inver 2; outver 2; ncount 1; go tacop
*
  task 'splat'
  getn i
  source ''; bchan 25; echan 485; docalib 1; gainuse 2
  doband 1; bpver 1; bif 1; eif 2
  outn innam
  outcl 'splat'
  outseq 0; aparm 1 0; channel 1; ichansel 0; solint 0
  go splat; wait splat
  getn j; outn innam
  go splat; wait splat
*
```

This should create two new datasets which have been averaged up in frequency.

11. We can now proceed with the amplitude calibration. We will use 3C295 for our flux density calibration which has a known flux density of 120 Jy between 50 and 100 MHz (the spectrum is turning over).

```
  task 'setjy'
  getn (i+2)
  source '3c295',''
  aparm 0; zerosp 120 0; optyp ''
  go setjy; wait setjy
```

12. Now we do some editing of the data trimming off some times at the beginning and ends of scans.

```
  task 'quack'
  source ''
  opcode 'beg'
  reason 'dunno'
  aparm 0 0.1667 0
  go quack; wait quack
  opcode 'endb'
  go quack; wait quack
```

13. The next step is to use CALIB to determine solutions on 3C295. First we solve for the phases on both 3C295 and our phase calibrator CXO J191919 using a short interval of 5 seconds. After that we apply the phase calibration, and then solve for the amplitude solutions on 3C295 alone using a 30 second solution interval. Finally, we apply the amplitude solutions to the whole run and then copy the solutions over to the source beam.

```
  task 'calib'
  doband -1
```

```
  calsour '3c295',''
  docalib 1; gainuse 1
  anten 0; basel 0
  clro
  refant 51; aparm 2 0
  solmode 'p'; cparm 0; snver 1; solint 5/60
  go calib; wait calib
  calsour 'CXO J191919',''
  go calib; wait calib
*
  task 'clcal'
  source ''; calsour ''
  opcode ' '; interp '2pt'; bparm 0; snver 1; inver 1;
  gainver 1; gainuse 2; refant 51
  go clcal; wait clcal
*
  task 'calib'
  doband -1
  calsour '3c295',''
  docalib 1; gainuse 2
  refant 51; aparm 2 0
  solmode 'a&p'; cparm 0; snver 2; solint 0.5
  go calib; wait calib
*
  task 'clcal'
  source ''; calsour ''
  opcode ' '; interp '2pt'; bparm 0; snver 2; inver 2;
  gainver 2; gainuse 3; refant 51
  go clcal; wait clcal
*
  task 'tacop'
  indi 1; outdi 1
  getn (i+2)
  geto (j+2)
  inext 'fg'; inver 1; outver 1; ncount 1; go tacop
  inext 'cl'; inver 2; outver 2; ncount 2; go tacop
```

Calibration is now complete and its time to look at the results. Figure 3 shows the phases on the calibrators 3C295 and CXO J191919. Although there are times when the phases are stable for several minutes, other times they wrap 3 times in 20 minutes or a change of about 40 degrees/minute. This is why a calibration beam is useful, although it still doesn't take into account spatial variations.

14. Use the task UVPLT to apply the calibration and plot up visibility amplitudes with time to see the pulses. Task TVPL will put your plot up on a TV display. Try looking at various combinations of stokes parameters and the two IFs.

```
  task 'uvplt'
  getn (i+2)
  bif 2; eif 2
  source ''
  stokes 'rr'
  timer 0
  xinc 1; bparm 11 1 0; bchan 1; echan 0
```

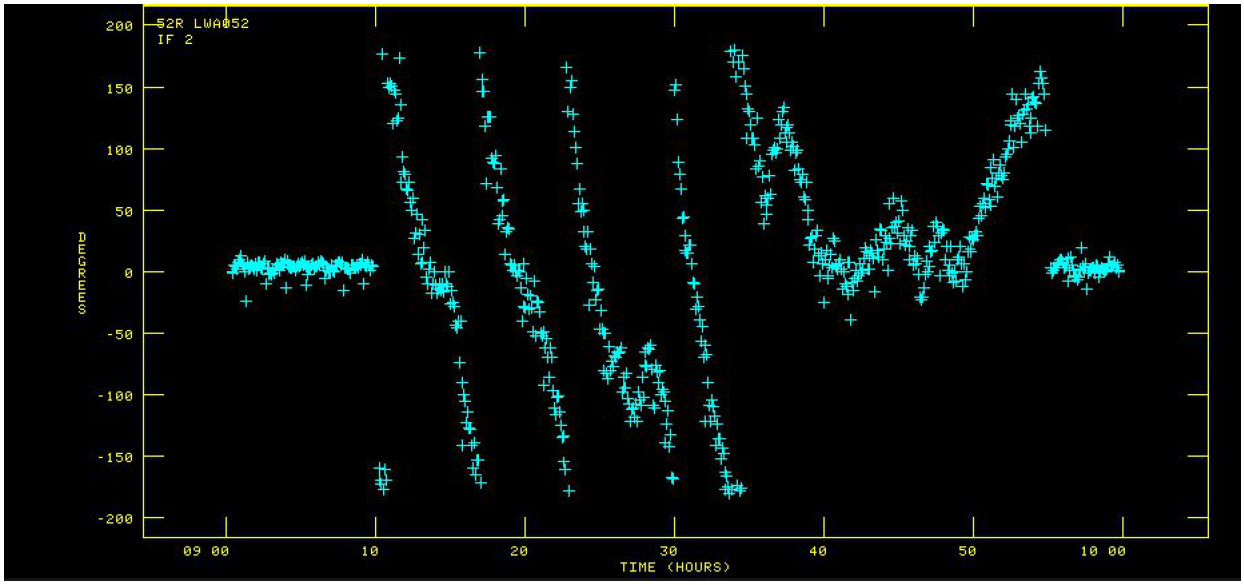

Figure 3: Phase solutions as a function of time derived for the calibrators 3C295 (start and end) and CXO J191919 (middle).

```
go uvplt; wait uvplt;
plver 0; go tvpl; wait tvpl
timer 0 9 25 0 0 9 25 10
factor -1
go uvplt; wait uvplt
plver 0; go tvpl; wait tvpl
```

Done!