PRESS Data Analysis Steps

Create the UWL calibration database (using METM)

To save on disk space, dspsr must integrate the single-pulse data in frequency before writing the data to disk. Before they can be integrated in frequency, they must be calibrated; therefore, before single-pulse data can be computed, a database of calibrators must be created. The first step before running the pipeline is to set up the environment:

```
cd psrpl
source psrpl_env.csh uwl
```

Split the band into polyphase filter bank sub-bands

The UWL band is divided by analog filters into three sub-bands (high, mid and low), each of which is separately digitized using different thresholds (to deal with different levels of RFI). These three sub-bands are further divided using polyphase filter banks, resulting in 26 sub-bands. The polyphase filter banks also imprint a spectral shape on each sub-band. All of these instrumental spectral variations make it more challenging to search for and excise RFI using outlier detection algorithms. Therefore, it is useful to first split the UWL band up into the 26 sub-bands; e.g.

```
cd psrpl
./psrpl_split_band.csh 0407-658
./psrpl_split_band.csh J0437-4715
```

When these scripts complete, for every sub-folder of `/fred/oz002/timing/uwl` that contains data for the specified source, there should be a corresponding sub-folder of `/fred/oz002/$USER/psrpl_output/uwl/split` that contains the same data split into sub-bands.

Excise Radio Frequency Interference

Inspired by Morello et al. (2019) and the TimeFrequencyZap class written by Paul Demorest, outliers in the folded data are detected using the interquartile range of a bunch of different statistics computed from the pulse profile in each frequency channel and sub-integration. The statistics include things like

1. Range of values in each profile
2. Sum of outlier power after detrending each profile
3. Sum of outlier harmonics after detrending the logarithm of the fluctuation power spectrum
4. (generalized) reduced chi-squared of linear fit between each profile and mean profile

The full list of zap commands used is listed in the Appendix.

```
cd psrpl
source psrpl_env.csh uwl
./psrpl_zap.csh 0407-658
./psrpl_zap.csh J0437-4715
```

When these scripts complete, for every file in `/fred/oz002/$USER/psrpl_output/uwl/split` that contains data for the specified source, there should be a corresponding file in `/fred/oz002/$USER/psrpl_output/uwl/zapped` that contains the RFI excised version of this data.

Challenge: 2021-02-17 RFI Mitigation updates

Challenge: 2022-01-12 IQR invalid when more than 25% of data are corrupted

Challenge: 2022-01-13 Generalized chi squared fails on mode-changing pulsars

Challenge: 2022-07-26 Child processes in merged jobs repeatedly oom-killed

Produce 5-minute sub-integrations and divide into sessions

```
cd psrpl
./psrpl_zapped_integrate.csh 0407-658
./psrpl_zapped_integrate.csh J0437-4715
```

When these scripts complete, for every leaf sub-folder in `/fred/oz002/$USER/psrpl_output/uwl/zapped` that contains the RFI excised data, there should be a corresponding file in `/fred/oz002/$USER/psrpl_output/uwl/integrated` that contains the the 5-minute sub-integrations. Note that cal and fluxcal files are not integrated.

```
./psrpl_sessions.csh 0407-658
./psrpl_sessions.csh J0437-4715
```

When these scripts complete, there will be a folder for each source in `/fred/oz002/$USER/psrpl_output/uwl/sessions`
For each source, there will be three sub-folders, one for each of the three UWL bands (10cm 15cm 25cm) and each band sub-folder will contain a bunch of text files with the `.sesssion` extension.

Create the calibrator databases

The bands (10cm 15cm 25cm) are listed in the sessions folders created in
`/fred/oz002/$USER/psrpl_output/uwl/sessions`

```
cd psrpl
./psrpl_cal_dbase.csh 0407-658
```

After this script finishes, there will be a new database file in
`/fred/oz002/$USER/psrpl_output/uwl/calibrators/0407-658/database.txt`
This file lists all of the input `*.it` files and the `*.fluxcal` solutions derived from them.
The solutions can be inspected in the sub-folders of
`/fred/oz002/$USER/psrpl_output/uwl/integrated/0407-658_O`

To identify outliers, it's probably easiest to do this one sub-band at a time; e.g;

```
psrplot -p calm */768/*.fluxcal
```

At this point the data are still divided into 26 sub-bands. Here's a script to produce a single Postscript file for each sub-band and tar them up for scp

make_fluxcal_plots.csh

Challenge: 2021-02-12 OBSBW and OBSFREQ Bug Fix

Challenge: 2021-02-15 Long double printf "nan" Bug Fix

Fluxcal Plots

Run MEM for each band

```
cd psrpl
source psrpl_env.csh uwl
./psrpl_cal_dbase.csh J0437-4715_R

./psrpl_pcm.csh mem J0437-4715 25cm bri00e19
./psrpl_pcm.csh mem J0437-4715 15cm bri00e19
./psrpl_pcm.csh mem J0437-4715 10cm bri00e19
```

Solutions appear in
`$PSRPL_OUT/calibrators/mem_bri00e19/J0437-4715`

When the MEM jobs finish, check the number of failed fits with

```
cd $PSRPL_OUT/calibrators/mem_bri00e19/J0437-4715
grep -F "invalid fit" */*/pcm.log | awk -F: '{print $1}' | uniq -c | sort -n
```

If there are large numbers of solutions with large numbers of invalid fits, it may indicate

1. a bug in the software
2. undetected RFI
3. undetected steps in instrumental response

Since July 2021, pcm implements some new routines that automatically

1. detect and remove outliers
2. detect and model of steps in instrumental response

Even when the fits are mostly good, sometimes the S/N is not sufficient to accurately model all of the instrumental response parameters. Run the following lines to perform some quality assurance checks and automatically put bad solutions in Trash/ sub-folders. They can be investigated later.

```
./psrpl_pcm_trash.csh mem J0437-4715 25cm bri00e19
./psrpl_pcm_trash.csh mem J0437-4715 15cm bri00e19
./psrpl_pcm_trash.csh mem J0437-4715 10cm bri00e19
```

If you want to see what will get trashed before actually moving the files to Trash/ sub-folders, set the following environment variable before running the above lines

```
setenv PSRPL_TESTING 1
```

If you first run in testing mode and you agree with the results, rather than simply re-running the above commands (which can be time consuming), you can more quickly move what has already been flagged with

```
./psrpl_pcm_move_trash.csh mem J0437-4715 25cm bri00e19
./psrpl_pcm_move_trash.csh mem J0437-4715 15cm bri00e19
./psrpl_pcm_move_trash.csh mem J0437-4715 10cm bri00e19
```

Here's a script to produce a pair of Postscript files for each sub-band and tar them up for scp

make_pcm_plots.csh

Challenge: 2021-02-16 Some of the fits are pretty bad

Challenge: 2021-02-18 Dropped Packets

Challenge; 2021-06-27 Medusa starts before and stops after CAL is fired

Challenge: 2021-07-05 Differential phase changes during session

MEM Plots - Rounds 2 and 3

Create template archives for METM use

Create the templates by running

```
./psrpl_make_templates.csh J0437-4715 bri00e19 >& make_templates.log
```

Challenge: 2021-07-28 Invalid planetary ephemeris

Challenge: 2021-07-31 Bad selection of template

The correct choice of reference template is really important. It is not sufficient to simply choose the solution with the best goodness-of-fit in pcm.fits, for the following reasons

- It may have large uncertainties in certain model parameters that describe the instrumental response; these get "baked" into the template profile, such that all METM solutions also exhibit large variance in the same noisy model parameters
- It may have had large number of channels zapped due to RFI
- It may have low S/N.

Therefore, a rank statistic was designed that takes the following into consideration

- Integration length of total.ar
- Signal-to-noise ratio of total.ar
- Goodness-of-fit of pcm.fits
- Standard deviation of sigma_chi
- Standard deviation of delta_theta
- Number of channels flagged due to RFI

This script performs the following steps for each frequency band:

1. Ranks the MEM solutions (based on both pcm.fits and total.ar)
2. Chooses the total.ar of the best fit as a template, choose.ar
3. Fits all other total.ar to the choose.ar via MTM
4. Adds all of the calibrated MTM output total.calib files to choose.ar, creating a template

The MTM goodness of fit between each of the MEM total.ar and choose.ar is generally very good except at low frequency. The templates appear in sub-folders of

```
$PSRPL_OUTPUT/uwl/templates/J0437-4715
```

Here's a script to produce a single Postscript file with template profiles ordered by frequency

make_template_plots.csh

The 18 Feb 2021 result is quite nice to scroll through page-by-page templates.pdf

Run METM for each band/freq

When happy with the templates, launch the METM jobs

```
cd psrpl
source psrpl_env.csh uwl
./psrpl_pcm.csh metm J0437-4715 25cm bri00e19
```

```
./psrpl_pcm.csh metm J0437-4715 15cm bri00e19
./psrpl_pcm.csh metm J0437-4715 10cm bri00e19
```

All of the jobs finished successfully (except for a corrupted MEM session that was deleted but still logged as a session).

Solutions appear in
`$PSRPL_OUT/calibrators/metm_bri00e19/J0437-4715`

When the METM jobs finish, check the number of failed fits with

```
cd $PSRPL_OUT/calibrators/metm_bri00e19/J0437-4715
grep -F "invalid fit" */*/pcm.log | \
awk -F: '{print $1}' | uniq -c | sort -n > invalid_fits.txt
```

If there are large numbers of solutions with large numbers of invalid fits, it may indicate

1. a bug in the software
2. Undetected RFI
3. Undetected steps in instrumental response

As for MEM, run the following lines to perform some quality assurance checks and automatically put bad solutions in Trash/ sub-folders. They can be investigated later.

```
./psrpl_pcm_trash.csh metm J0437-4715 25cm bri00e19
./psrpl_pcm_trash.csh metm J0437-4715 15cm bri00e19
./psrpl_pcm_trash.csh metm J0437-4715 10cm bri00e19
```

If you want to see what will get trashed before actually moving the files to Trash/ sub-folders, set the following environment variable before running the above lines

```
setenv PSRPL_TESTING 1
```

If you first run in testing mode and you agree with the results, rather than simply re-running the above commands (which can be time consuming), you can more quickly move what has already been flagged with

```
./psrpl_pcm_move_trash.csh metm J0437-4715 25cm bri00e19
./psrpl_pcm_move_trash.csh metm J0437-4715 15cm bri00e19
./psrpl_pcm_move_trash.csh metm J0437-4715 10cm bri00e19
```

Challenge: 2021-12-13 CAL not detected

There appears to be very little variation in the model parameters that describe receptor cross-coupling. Therefore, it might be best to simply set these model parameters to constant values, followed by a statistical test to verify that the experimental data are consistent with these constant values (within the uncertainty). As a function of radio frequency, the constant values could be the weighted median of the estimated values (over all epochs).

Before this can be done, it is necessary to correct the apparent variations in the rotation of the receiver about the line of sight induced by Faraday rotation in the ionosphere. This requires grouping the sub-bands into sessions to be fit, and sesdiv was updated with the -j option to join bands (divide only in time).

Challenge: 2021-12-14 Duplicate sessions

```
cd $PSRPL_OUT/calibrators/metm_bri00e19/J0437-4715
vap -Rqc "NAME FREQ MJD" */*/pcm.fits | sesdiv -j
```

Check that all sessions have max 26 files

```
wc -l *.session | awk '$1>26 {print $0}'
```

A new program, pcmrm, estimates the ionospheric RM contribution and subtracts it from each session

```
foreach ses ( *.session )
  pcmrm -M $ses |& tee -a rms.txt
end
```

The best-fit RMs are listed in rms.txt

```
awk -F= '{print $3}' rms.txt | sort -g | head -1
-1.01684
awk -F= '{print $3}' rms.txt | sort -g | tail -1
0.188259
```

The RM varied by about 1.2 rad m^-2 over the period of UWL observations.

`pcmavg` was updated to compute the weighted median of the model parameters; the weighted median of the RM-corrected pcm solutions can be computed with

compute_weighted_median.csh

The combined results can be plotted with

plot_weighted_median.csh

and are visible at

pcm_rmc_median.pdf

pcm was updated to

- take a known solution as input and copy the model parameters as first guesses
- hold selected model parameters fixed
- accept an ionospheric Faraday RM on the command line and incorporate the transformation in the model

psrpl was updated with a new remetm template that does the following

- locate the weighted median solution for sub-band
- locate the RM-corrected METM solution for this sub-band and epoch
- execute pcm with
  - the weighted median solution as the previous known solution;
  - all cross-coupling parameters (3,4,5,6) held fixed;
  - the noise diode coupled after the OMT; and
  - the best-fit ionospheric RM

```
./psrpl_pcm.csh remetm J0437-4715 25cm bri00e19
./psrpl_pcm.csh remetm J0437-4715 15cm bri00e19
./psrpl_pcm.csh remetm J0437-4715 10cm bri00e19
```

Here's a script to query the goodness of the re-METM fits:

```
#!/bin/csh -f

set gof = '{sum{$pcal:eqn:chisq}/sum{$pcal:eqn:nfree}}'

foreach band ( 768 896 1024 1152 1280 1408 1536 1664 1792 1920 \
               2048 2176 2304 2432 2560 2688 2816 2944 3072 3200 \
               3328 3456 3584 3712 3840 3968 )
  echo $band
  psrstat -c "$gof" */*_$band/pcm.fits >& metm_gof_${band}.txt
  wc -l metm_gof_${band}.txt

end
tar zcvf metm_gof.tgz metm_gof_*.txt
```

The above script can be copied and altered with

```
perl -p -i -e 's|metm|remetm|g' gof.csh
```

And the following script used to compare results

```
#!/bin/csh -f

foreach band ( 768 896 1024 1152 1280 1408 1536 1664 1792 1920 \
               2048 2176 2304 2432 2560 2688 2816 2944 3072 3200 \
               3328 3456 3584 3712 3840 3968 )
  echo $band
  paste metm_gof_${band}.txt remetm_gof_${band}.txt >
both_gof_${band}.txt
  wc -l metm_gof_${band}.txt remetm_gof_${band}.txt
both_gof_${band}.txt
```

```
end
```

Challenge: 2021-12-18 Fewer remetm solutions than metm solutions

In the worst case out of all remetm solutions, holding the frontend parameters fixed increases the reduced chisq by 1.4%

```
cat both*.txt | awk '{print $1,$4/$2}' | sort -gk2 | tail -10
25cm/2020-10-13-1700_1024/pcm.fits 1.00595
25cm/2021-10-10-1500_768/pcm.fits 1.00621
25cm/2021-06-24-2100_896/pcm.fits 1.00643
25cm/2020-10-13-1700_896/pcm.fits 1.00708
25cm/2021-06-24-2100_768/pcm.fits 1.00933
25cm/2020-05-29-0200_768/pcm.fits 1.01013
15cm/2020-12-16-1100_2432/pcm.fits 1.01125
10cm/2020-10-13-1700_2560/pcm.fits 1.01216
25cm/2020-10-13-1700_768/pcm.fits 1.01404
15cm/2019-04-17-0200_2304/pcm.fits 1.01415
```

Challenge: 2021-12-22 Large numbers of trashed solutions at high frequency

Even in the weighted median of the RM-corrected solutions, there remains a lot of scatter at high radio frequencies. The following script smooths each weighted-median solution and plots the difference between the smoothed parameters and the weighted-median parameters

```
#!/bin/csh

foreach rmc ( *.rmc )

  set base=`echo $rmc | awk -F. '{print $1}'`
  echo $rmc $base

  smint -gcv -iqr 1.5 $rmc
  pcmdiff -D ${base}_diff.ps/cps -s ${base}.smint $rmc

end
```

The differences are plotted here.

It's also possible to first combine all of the solutions from each sub-band into one mega-solution before smoothing, as in the following:

```
    37  6:31    mv pcm_med_768.rmc pcm_med_0768.rmc
    38  6:31    mv pcm_med_896.rmc pcm_med_0896.rmc
    39  6:31    psradd -R -o pcm_med_all.rmcR *.rmc
    40  6:31    smint -gcv -iqr 1.5 *.rmcR
```

```
    41   6:31    psrsplit -cal -c 128 pcm_med_all.smint
```

However, the spectral features at low-frequency appear to be discontinuous at band edges and are probably induced by the PFB (i.e. not continuous across the band).

Challenge: 2021-12-30 Smoothing splines have too much structure

2022-01-01: it might be that the response of the system is more significantly/evidently variable at low frequency than at high frequency, and that different strategies should be applied to different parts of the band.

Ideas:

- Use pcm_evolve to look for temporal variations (in each parameter)
- Use box-and-whisker spectra to identify significant outliers (for each parameter)
- Use the AIC to compare models with free parameters and models with fixed parameters

The script to produce a pair of Postscript files for each sub-band and tar them up for scp is identical to the one used for the MEM solutions.

Plots will be placed in pcm_metm

Challenge: 2022-03-02 Weighted median of Stokes vector is over-polarized

Note: 2022-03-17 Persistent spike in delta_chi of 1024 MHz sub-band

Challenge: 2022-03-18 Ionospheric RM baked into noise diode Stokes

Challenge: 2022-07-22 noise diode not handled correctly when steps are detected

Challenge: 2022-07-24 re-definition of bad fit required

Smooth the METM solutions in each band/freq

smint was updated to find the optimal smoothing factor using cross-validation. It can be launched on each of the 26 sub-bands with

```
cd psrpl
source psrpl_env.csh
./psrpl_smint.csh
```

And after this job finishes, the spline fits to the model parameters can be plotted with

```
cd psrpl
source psrpl_env.csh
./psrpl_smint.csh
```

And the database created with

```
cd $PSRPL_UWL/calibrators/metm_bri00e19/J0437-4715
lfs find . -name smint.fits > smint.ls
pac -W smint.ls

cd $PSRPL_UWL/calibrators/0407-658/
lfs find . -name smint.fits > smint.ls
pac -W smint.ls
```

Fold the pulsar

Before producing single-pulse archives, it is necessary to ensure that:

- the DM is accurate;
- the ephemeris is accurate;
- the rotational phase is set up to centre the on-pulse; and
- after calibration, the RM is accurate

This is achieved by first folding the pulsar signal and calibrating the folded pulse profiles, as described in the following subsections:

- Evaluate the folded noise diode (calibrator) observations
- Calibrate the folded pulsar observations
- Verify the DM and ephemeris
- Verify the RM

First set up the environment for PRESS single-pulse data reduction

```
cd psrpl
source psrpl_env.csh press
mkdir logs
```

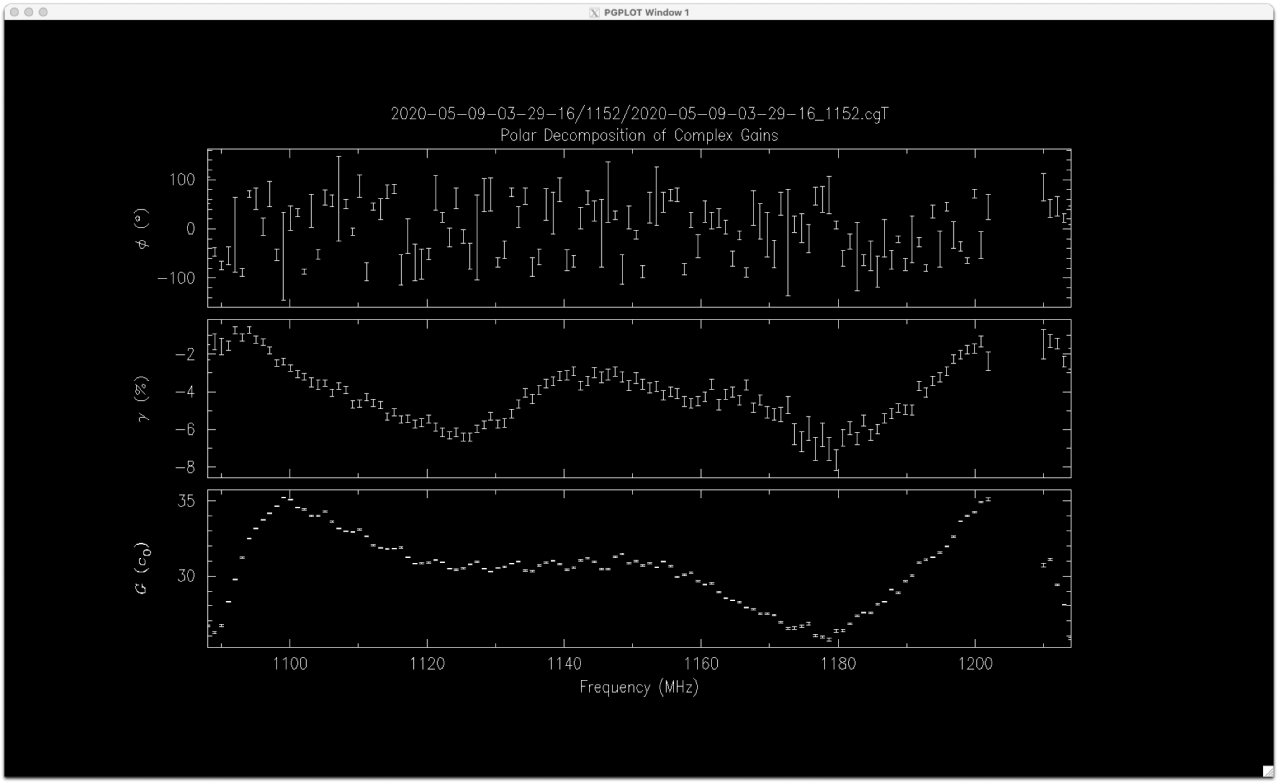
Then launch commands to fold both the pulsar and its calibrator observations

```
./psrpl_fold.csh J1921+2153 >& logs/J1921+2153_fold.log &
tail -f logs/J1921+2153_fold.log
```

Evaluate the folded noise diode (calibrator) observations

```
cd $PSRPL_OUT/folded/J1921+2153_R
psrplot -p calm */*/*.cgT
```

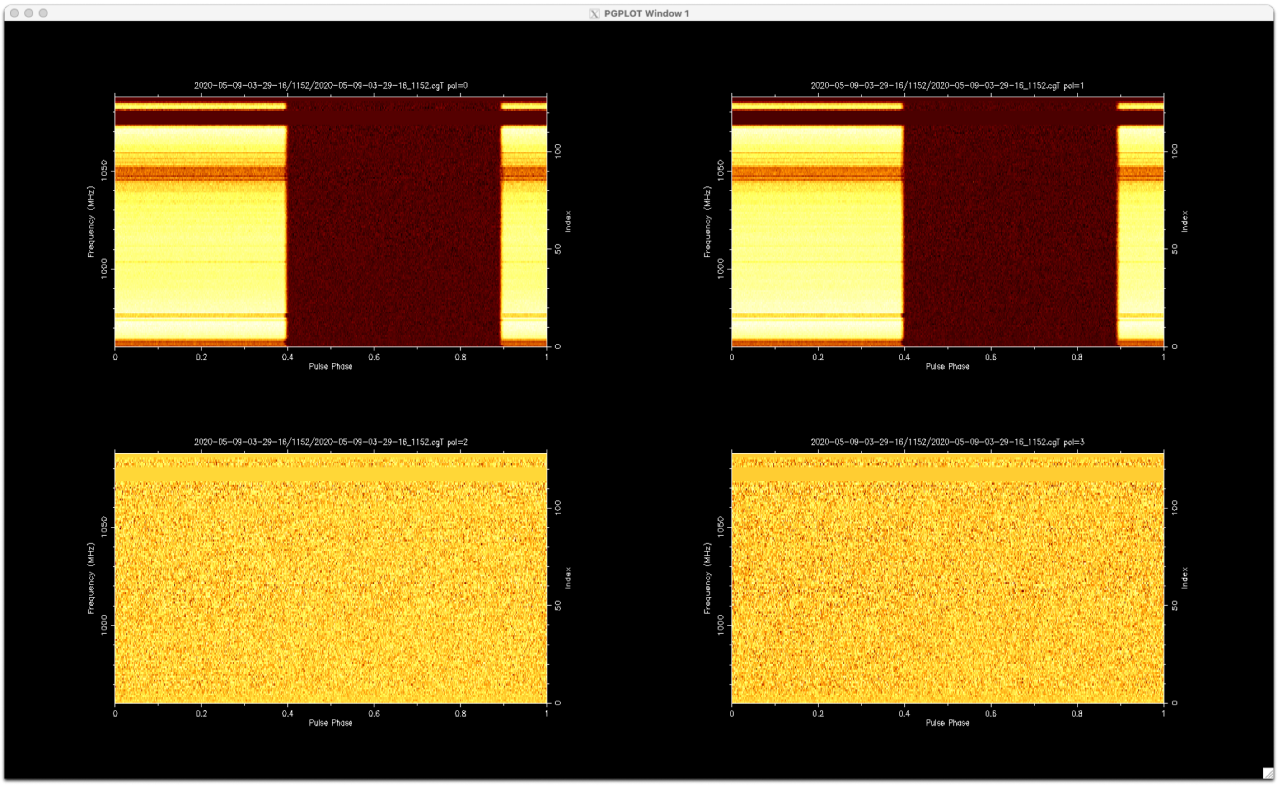
Good data should have well-defined curves, while dropped packet issue creates undefined differential phase; e.g.



The dropped packet issue can also be seen by zero Stokes U and V in the noise diode observations; e.g. in plots of phase vs frequency

```
psrplot -p G -l pol=0- */*/*.cgT -N 2x2
```

(Note that these data are in the Coherence state; i.e. XX, YY, Re[XY], Im[XY])



It's also good to plot the phase-versus-frequency intensity of the folded pulsar observations to ensure that there is no residual/missed RFI.

Calibrate the folded pulsar observations

Set up to use the UWL calibrator solutions in /fred/oz146

```
cd psrpl
source psrpl_env.csh press
```

Then launch the calibration script for the desired pulsar; e.g.

```
./psrpl_pac_folded.csh J1921+2153 launch &
```

One job is launched for each UTC. While the jobs are running, you can monitor progress in the log files; e.g.

```
cd /fred/oz146/psrpl/folded/J1921+2153
tail -f */psrpl_pac_folded.out
```

Calibrated data are output in the UTC sub-directories of the folded archive location; e.g.

```
cd /fred/oz146/psrpl/folded/J1921+2153
ls */*.zCTR
ls */*/*.zCT
```

Challenge: 2022-03-25 Flux calibration solutions full of zeros

Verify the DM and ephemeris

Check if the DM and ephemeris are accurate using pdmp; e.g.

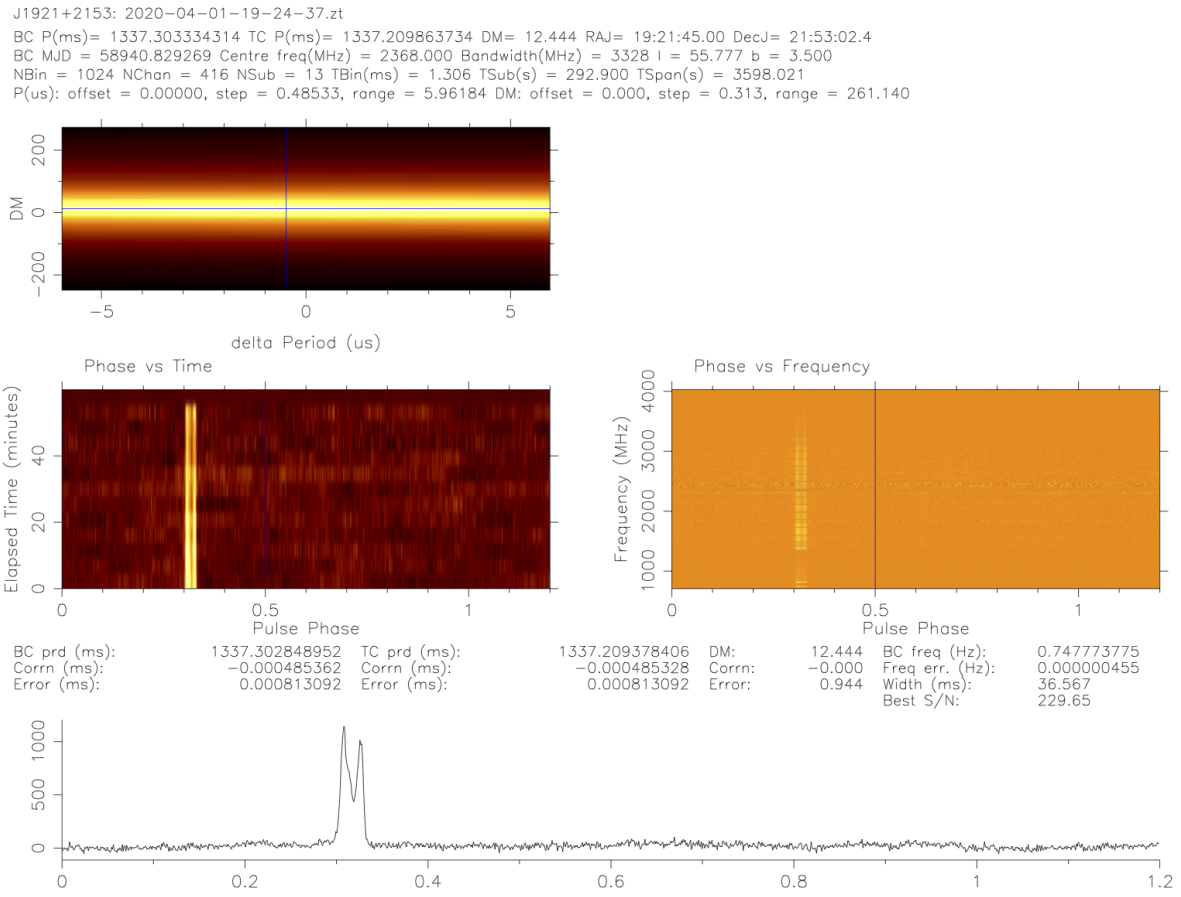
```
pdmp -g 2020-04-01-19-24-37_pdmp.ps/cps 2020-04-01-19-24-37.zt
```

In the text output by pdmp to the terminal, find the following lines:

```
Best BC Period (ms) = 1337.302849  Correction (ms) = -0.0004853619945
Error (ms) = 0.0008130916994
Best TC Period (ms) = 1337.209378  Correction (ms) = -0.00048532807
Error (ms) = 0.0008130916994
Best DM =   12.4440  Correction = -4.226e-12  Error = 0.9442
```

Note that the Correction to the period (-0.5 microseconds) is smaller than the Error (0.8 mus); therefore, within the experimental uncertainty, the ephemeris is accurate enough to track phase for the duration of this observation (1 hour).

The DM correction is also smaller than the error. The output of pdmp (next page) doesn't look bad



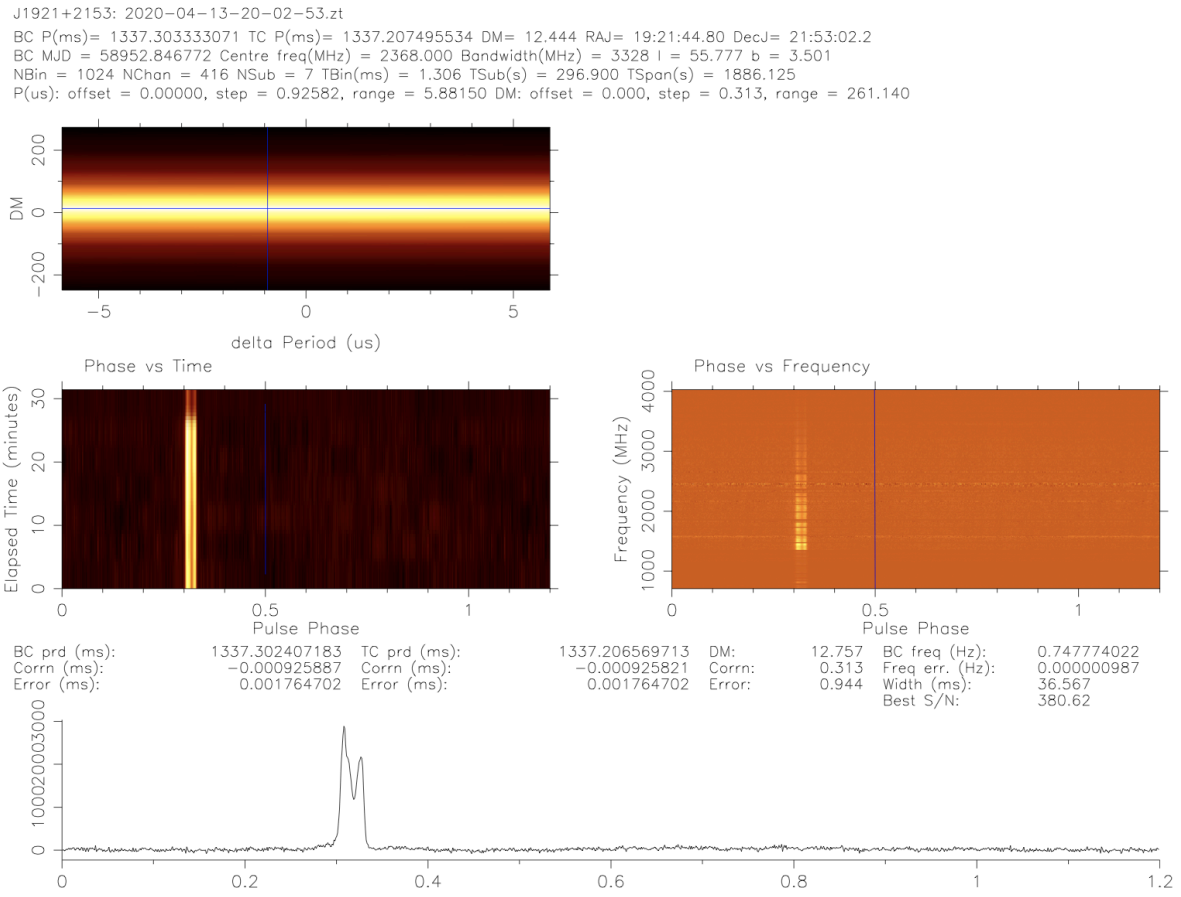
Running the same command on the latter epoch:

```
pdmp -g 2020-04-13-20-02-53.ps/cps 2020-04-13-20-02-53.zt
```

Yields

```
BC Period (ms) = 1337.303333  TC Period (ms) =  1337.207496  DM =
12.4440
Best BC Period (ms) = 1337.302407  Correction (ms) = -0.0009258873877
Error (ms) = 0.001764701992
Best TC Period (ms) = 1337.20657  Correction (ms) = -0.0009258210341
Error (ms) = 0.001764701992
Best DM =   12.7575  Correction = 0.3135  Error = 0.9442
```

Again, all corrections are smaller than the errors. The output of pdmp also looks ok



There is no visible sign of phase drift between these two epochs. This can also be verified using pat; e.g. the latter observation has a better S/N, so choose it as the template and run

```
pam -FT -eFT 2020-04-13-20-02-53.zt
pat -R -jFT -s 2020-04-13-20-02-53.FT 2020-04-01-19-24-37.zt
```

The output

```
2020-04-01-19-24-37.zt 0 0 -0.000190289 4.11697e-05 <-turns::microsec->
58940.829563415472800658 18 -254.456 55.0525
```

indicates a statistically significant shift of -254 +/- 55 microseconds; however, this is only 0.2 times the width of a single phase bin (-0.000190289 * 1024 = -.195); therefore, the shift can most likely be ignored.

Verify the RM

To verify the Faraday rotation measure, it is necessary to first calibrate the data. Produce a calibrator database:

```
cd $PSRPL_OUT/folded/J1921+2153_R/
pac -w -u zzT
```

Verify the RM using rmfit

```
rmfit -r -W 2020-04-13-20-02-53.ztcT
```

Yields

```
rmfit: converged in 3 iterations
       final rotation measure = (-13.2525+-0.363602)
```

This is significantly and worryingly different to the current value in the catalog, RM -16.99 (nsk+15). The brute-force (RM synthesis) algorithm,

```
rmfit -m -32,0,20 -D 2020-04-13-20-02-53.ztcT
```

yields a similar RM

```
Best RM is: -13.1646 +/- 0.24
```

2021 Sept 8 - This requires further investigation and debugging.

- What METM calibrator was selected?
- What flux calibrator was selected?
- Is unflagged RFI a problem?

To process all of the pulsars

ls $PSRPL_DATA/search/*/* -d | awk -F/ '{print $7}' | grep -v -F _R | sort | uniq > psrs.list

Analyse single-pulse profiles

Launch the PRESS single-pulse data reduction pipeline

```
cd psrpl
source psrpl_env.csh press
mkdir -p logs
./psrpl_single.csh J1921+2153 >& logs/J1921+2153_single.log &
```

This can take a while, but you can monitor progress with

```
tail -f logs/J1921+2153_single.log
```

You'll likely see error and recovery messages like the following:

```
ERR: ./J1921+2153_1.err out-of-memory
SRC: ./J1921+2153 JOB: 1
```

```
Relaunching 11 GB job
Submitted batch job 27197371
./J1921+2153_job_1 queued with 11GB of memory
```

The wait script will increase the requested resource allocation and relaunch jobs that run out of time or run out of memory.

Plot and check the outputs of the single-pulse data reduction pipeline

After the single pulses have been produced, it's a good idea to compare the total integrated mean of the single pulses with the total integrated mean of the folded profiles. This is done to see if dspsr is doing the right thing (on-the-fly calibration and RFI mitigation) and to check if RFI mitigation on single-pulse profiles is zapping bright pulses.

Further psrdiff rationale here.

```
cd psrpl
source psrpl_env.csh press
[tcsh] ./psrpl_single_diff.csh J1921+2153
```

The above script runs psrdiff for each UTC and sub-band, the results of which are data files that represent the ***single-pulse total minus the folded total***. These results can be plotted using standard tools; e.g.

```
cd /fred/oz146/psrpl/single/J1921+2153/2020-04-13-20-02-53
psrplot -ps -jF ???/psrdiff.outCcgT ????/psrdiff.outCcgT
psrplot -pfreq+ -l pol=0- ???/psrdiff.outCcgT ????/psrdiff.outCcgT
```

Run psr4th on all of the single pulse *.ar for each band

- **Run psr4th**

For example,

```
cd $PSRPL_OUT/single/J1921+2153/2020-04-01-19-24-37/1024
psr4th -B -j "edit off=normal" -jDF mjd*.cg
```

psr4th always (annoyingly) outputs a file named psr4th.ar ... you'll need to move this file to a unique filename for each .ar file so that files to not get overwritten; I suggest

```
mv psr4th.ar filename.4th
```

I might update psr4th to do this automatically some day. :-)

- **Combine the filename.4th files**

For each of the 26 sub-bands, use psradd -T to combine the filename.4th files produced for each of the separate *.ar files (there should be 3 filename.4th files for each sub-band, 2 in the first epoch and 1 in the second epoch).

Use psradd -R to combine the time-integrated total 26 sub-bands into the three bands that we identified yesterday.

- **Mentally prepare yourself**

Parkes is not Arecibo, and the S/N will be much lower in these data.

- **Plot the phase-resolved eigenvalues**

e.g.

```
psrplot -p4 total_band1.4th -j "fourth debias"
[other options: -c sqrt=0 to look at the variances
-c "x:win=0.ph1:0.ph2" to zoom]
```

Note: if you want to remove the baselines of the eigenvalues to evaluate what is the significance of the relative differences, use:

```
psrplot -p4 psr4th.ar -j "fourth debias" -c "eigbase=1"
```

Note that this takes into account only the spread of the points in the distribution due to the noise, but the distributions on-pulse can additionally spread due to different reasons, e.g., self-noise.

Challenge: 2022-05-25 Correlated Structure in Off-pulse Baseline

Appendix - Zapping Script

```
#!/usr/bin/env psrsh

zap chan 0-7,120-127

# zap based on total intensity
state Stokes
zap tfzap pols=0

# disable smoothing
zap tfzap smooth=0

# use clfd-like outlier detection based on Tukey's fences
zap tfzap mask=iqr

# iterate to cope with large numbers of outliers
zap tfzap iterations=30

zap tfzap report=1

zap tfzap mask:cutmax=10
zap tfzap mask:cutmin=0

# excess modulated power
zap tfzap stat=range
zap tfzap

zap tfzap mask:cutmax=4

# excess periodic structure
zap tfzap stat=sdo
zap tfzap

# excess white noise patches
zap tfzap stat=sho
zap tfzap

# excess structure that varies from subint to subint
zap tfzap stat=chi
zap tfzap mask=set
zap tfzap iterations=1
zap tfzap mask:cutmax=1.7
zap tfzap

zap extend fcutoff=0.8  # if 80% of channels are bad, zap subint
zap extend tcutoff=0.8  # if 80% of subints are bad, zap channel
zap extend report=1
zap extend
```

Appendix - MEM Shortcut Scripts

The following shortcut scripts are not part of the psrpl distribution and were run as follows

```
./complete_J0437-4715.csh >& logs/complete_J0437-4715.log &
./complete_0407-658.csh   >& logs/complete_0407-658.log   &
```

complete_J0437-4715.csh

```
#!/bin/csh -f

source psrpl_env.csh uwl

rm $PSRPL_BOOK/split/J0437-4715*
./psrpl_split_band.csh J0437-4715

rm $PSRPL_BOOK/zapped/J0437-4715*
./psrpl_zap.csh J0437-4715

rm $PSRPL_BOOK/integrated/J0437-4715*
./psrpl_zapped_integrate.csh J0437-4715

./psrpl_sessions.csh J0437-4715
./psrpl_cal_dbase.csh J0437-4715_R

./launch_pcm_mem.csh
```

complete_0407-658.csh

```
#!/bin/csh -f

source psrpl_env.csh uwl

rm $PSRPL_BOOK/split/0407-658*
./psrpl_split_band.csh 0407-658

rm $PSRPL_BOOK/zapped/0407-658*
./psrpl_zap.csh 0407-658

rm $PSRPL_BOOK/integrated/0407-658*
./psrpl_zapped_integrate.csh 0407-658

./psrpl_sessions.csh 0407-658
./psrpl_cal_dbase.csh 0407-658
```

Appendix - link to the Flagged corrupted files

Appendix - link to the files that need to be manually verified (for RFI)