PPTA pipeline processing

This document describes the data flow for PPTA data. Observations are taken using the Parkes telescope. Issues that arise during the observation are logged on our dashboard: (currently available from http://www.atnf.csiro.au/people/ghobbs/ppta_log2/dashboard.html). The dashboard provides a searchable means to determine what happened during a particular observation. Older records were kept on paper - these paper logs are also available from the dashboard.

For each pulsar a pulsed-calibrator is first observed, the pulsar is then observed and, for some pulsars, a calibrator source is observed again after the pulsar observation. At some point during the observing session observations of Hydra A are made at all observing bands. These observations are subsequently used for flux calibration.

Typical observation durations are 64 minutes, but observers commonly stop observing a pulsar if it is weak and return to observe it again later in the session.

Observing sessions usually last for ~3 days allowing all PPTA pulsars to be observed with the 20cm multibeam receiver and also with the 10/50cm dual-band receiver. Observations in the 20cm band are recorded using two coherent dedispersion instruments (APSR and CASPSR) and two incoherent filterbanks (PDFB3 and PDFB4). In the 50cm band we use APSR, CASPSR and PDFB3. In the 10cm band we use PDFB4. Note that these observing strategy leads to multiple data files corresponding to the same signal.

The data are automatically transferred to RAID disks at Marsfield. The data are then stored on the RAIDs and also included in the Parkes pulsar data archive (data.csiro.au).

Pipeline processing

The pipeline processing consists of a large number of tcsh scripts. These can be run by any PPTA member. It would be possible to run them automatically, but currently we do not do this. The processing scripts access a mySQL database that records information about the observations and the processing steps.

Note that PSR J0437-4715 currently is processed in a slightly different manner to all the other PPTA pulsars. The PSR J0437-4715 is described at the end of this section.

Emails are sent to george.hobbs@csiro.au as each step of the pipeline completes and various log files are written to disk.

Before the pipeline:

The pipeline relies on averaged fluxcal files being available. These are produced independently of the pipeline. Note that this implies that the most recent fluxcal files are not used when processing the most recent data.

Pipeline: step 0

- The scripts automatically check if the psrchive packages are available and whether there is sufficient disk space for the processing
- Any new averaged flux calibrator files are identified and stored in the mySQL database.
- A script searches for new PCM files that are not already in the database. If any exist then they are added to the database.

Pipeline: step 1 (run for each pulsar separately)

- A script searches for new observations of the specified pulsar. The meta-data for any new observations are recorded in the mySQL database. The meta-data are obtained using "vap" and we record backend, beconfig, bw, freq, nchan, nbin, nsub, npol, tsub, mjd, site, projid, hdrver, name, ra, dec and rcvr
- The same script searches for new pulsed-calibrator observations and they are also stored in the database (with the same meta-data).
- The script then obtains all non-processed observation files from the database for a given pulsar.
- A link to the file is placed in a temporary directory on the local disk for the computer running the script
- A new dispersion measure value is obtained from a "default ephemeris" for that pulsar and "pam" is used to update the observing ephemeris and DM:
  `pam -d $dm -E $psr.par -e newEph $fname`
- The observing frequency for the observation is obtained using "vap -c freq". A band is then chosen:
  20cm: f > 1000 MHz && f < 1600 MHz

10cm: f > 2800 MHz && f < 3500 MHz
50cm: f > 500 MHz && f < 800 MHz

For 50cm observations:

- If APSR observation and MJD > 55039 then zap using:
  paz -F “716 724” -F “736 744”
- If CASPSR observation then zap using:
  paz -F “600 705” -F “758 900”
  (note that this will be updated soon based on recent work by R. Manchester)
- For observations that are not CASPSR various commands are used to remove the digital TV transmissions:

```
if (mjd < 52852 || mjd > 55039)
 paz -E5 -r -R $nchan10
else if (mjd < 53310)
 paz -r -R $nchan10 -E5 -F “652 659”
else if (mjd < 54282)
 paz -r -R $nchan10 -E5 -F “652 659”
-F “666 673”
else if (mjd < 54313)
 paz -r -R $nchan10 -E5 -F “652 659”
-F “666 673” -F “680 687”
else
 paz -r -R $nchan10 -E5 -F “652 673”
-F “680 694”
```

- psrsh is used to t-scrunch the file to 8-subintegrations (producing a .t8z file for each observation)

For 10cm observations:

- psrsh is used to zap 5% of the band edges and time-scrunch to 8 subintegrations (producing a .t8z file for each observation)

For 20cm observations:

psrsh is used to:

- define the median zapping to nchan/8
- zap using a median zapper and 5% of the band edges
- time scrunch to 8 subintegrations (producing a .t8z file for each observation)

Continuation of step 1 for all observing bands

- Code is run (fix_ppta_data) to fix problems with file headers:
  - files beginning with 'a' and have backend = 'WBCORR': change backend to 'PDFB1'
  - DFB1 and DFB2 files whose frontend = 'MULTI' and bw = '256': change bw to '-256'; reverse the frequency channels
  - DFB1 files before 53954 (MJD): correct STT_OFFS
  - 'MUL_1' => 'MULTI'
  - CPSR2 files: ensure SRC_NAME starts with 'J'
  - CPSR2 files: change frontend to '50CM', '10CM', 'NH-OH', or 'nMULTI'

- any related information in the mySQL database is updated based on the output of the `fix_ppta_data` program.
- the `dlyfix` script is run that enters an instrumental delay into the PSRFITS file to align all the different backends. These delays are separately determined.
- The database is updated to create an "intermediate" file (the .t8z file) and link it to the original observation. A flag is set to TRUE to state that the original observation has been processed. A flag is set for the intermediate file to say that that file has not been processed.
- Various book-keeping steps are carried out to update the database if any of the steps in this pipeline failed.

Pipeline: step 2 (run for each pulsar separately)

The second stage of the pipeline is to produce zapped, time scrunched calibrator files

- The mySQL database is queried to determine which calibrator files have not been processed for the specified pulsar

The following steps are carried out for each file:

- the "cleanCalFile" script is run. This is used to determine whether the first or last subintegration in the calibrator file is corrupt. If so, that subintegration is deleted.
- The observing band is determined for the file (using the same criteria as for the pulsar observations in step 1 of the pipeline)
- The same zap commands are used as in step 1 of the pipeline
- The calibrator file is fully time scrunched and a .zTcf file is produced.
- The `fix_ppta_data` program is run on the output files.
- The database is updated to create an intermediate file (.zTcf) and a processed flag set for the original calibrator file.

Pipeline: step 3 (run for each pulsar separately)

- For the specified pulsar, the mySQL database is queried to determine which .t8z files need processing. The following commands are run for each of these files.
- Pulsed-cal files are obtained from the mySQL database that were observed within 2 days of the pulsar observation. All these files are copied into the working directory.
- All available averaged flux calibrator files are copied into the working directory

- A “pac” database file is produced:
  `pac -w -u avfluxcal -u zTcf`
- Polarisation and flux calibration is carried out:
  `pac -d database.txt -u avfluxcal -u zTcf *.t8z`
- The script reads the output from pac to determine which calibrator was actually used for each file and whether the polarisation and/or flux calibration was successful. This information is stored in the database linked to the original file.
- The resulting files are processed using pam to produce 32 frequency channels:
  `pam -e czt8f32 --setnchn 32 ...`
  and also to form fully time, frequency and polarisation scrunched files:
  `pam -e czFTp -FTp ...`
- These resulting files are stored in the mySQL database and linked to the original file.
- The resulting files are copied into specified directories based on the pulsar’s name and the observing band.
- Various book-keeping checks are made to ensure that these final files are not corrupt.

Pipeline: step 4

This part of the pipeline obtains ToAs for all the newly processed files and stores the output in the database

- A list of all the newly made fully frequency and time scrunched files are obtained. The following is run on each of these files:
- The observing band is determined (in the same manner as in step 1 of the pipeline)
- The backend instrumentation is determined using vap

A suitable standard template is chosen as follows:

For 20cm and 10cm observations:
- The script searches for an analytic template in the 20cm or 10cm band for the specific backend.
- If that doesn’t exist then the PDFB4 analytic template is used.
- If that doesn’t exist then the PDFB2 template is used

For 50cm observations:
- The script searches for an analytic template in the 50cm band for the specified backend
- If it doesn’t exist then it uses the APSR template
- If that doesn’t exit then it uses the CPSR2 template

After selection of the standard template:
- “pat” is used to obtain the pulse arrival time:

pat -s $template -C “rcvr,backend” -f tempo2 $file

- The “pat” output is appended with a “-f” flag that contains the receiver and the backend used.
- The output is recorded in the database and the database updated with a statement on which template was used for the given observation.

Pipeline: step 5

This part of the pipeline returns to the .t8z files and processes them using pcm. This part of the processing currently only applies to the 20cm observing band.

- The script queries the database to obtain all 20cm observations that have not been “pcm processed”.
- All the calibrator files, fluxcal files and pcm files are copied into the working directory.
- A “pac” database is produced:
  pac -w -u avfluxcal -u zTcf -u avpcm
- “pac” is then run to carry out the pcm calibration:
  pac -d database.txt -ST -e t8zC -u avfluxcal -u zTcf -u avpcm *.t8z
- Checks are carried out to determine which files were successfully calibrated.
- The database is updated to indicate that this file has been processed using PCM and to indicate which cal file, flux cal file and which pcm file was used in the processing.
- The resulting calibrated file is first frequency scrunched to 32 frequency channels to produce t8zCf32 files
- The files are then fully time and frequency (but not polarisation) scrunched to give zCFT files.
- These files are recorded in the database

Pipeline: step 6

In this stage ToAs are obtained from the pcm calibrated files

- The script queries the database to determine which files have been “pcm-processed”, but need ToA determination
- A template is chosen. If available a polarisation scrunched pcm template is used for the given backend. If a pcm template is not available then the PDFB2 Stokes I template is used instead (CPSR2 for 50cm observations).
- “pat” is used to obtain the ToAs and the result stored in the database along with which template was used.

Pipeline processing for PSR J0437-4715

We currently form the invariant interval for PSR J0437-4715.

- The calibrator files are processed as before
- The “best” ephemeris and DM is installed into the observation file
- As before, the zapping depends upon the observing band
- .t8z files are produced by scrunching to 8 subintegrations
- The “fix_ppta_data” and “dlyfix” scripts are run
- pac is run for polarisation and flux calibration
- The file is scrunched to 32 frequency channels.
- The invariant interval is formed and an intermediate file (Iczt8f32) stored on disk
- The file is fully scrunched in time, frequency and polarisation and a file is stored on disk (IczFTp)
- ToAs are formed as before, but using an analytic template based on the invariant interval.

Ways to access the data

The database stores the output of the pipeline processing. There are various tcsh scripts that are used to reprocess various data files and also to access the ToAs from the database (we also have a web-based interface to the ToAs available from our PPTA dashboard).

Scripts include:

- `runall`: runs all the pipeline processing steps for all PPTA pulsars
- `flagobs`: allows the user to describe individual files that should be flagged. The script allows the user to define different flag types.
- `generalTOAs`: allows the user to obtain TOAs from the t8f32 files, but allows the user to define the amount of frequency/time scrunching
- `getTOAs`: general script for obtaining the ToAs from the database. It accounts for flag entries in the database that have been added using the “flagobs” script. It allows the user to select “normal” ToAs or those obtained used “pcm”
- `new_template`: reprocesses all the “pat” commands when a new template becomes available
- `new_calibration`: this script is used if the calibration procedures have changed and new files need to be produced from the .t8z files.
- `plinfo`: provides information on a given file (e.g., whether it was calibrated and which template was used when forming ToAs)

- `reprocess_sel`: reprocess specific observations of a given pulsar (defined by e.g., backend, frequency, MJD range etc.)

Scripts currently not used

In the past we have had scripts to:
- include DM variability before frequency scrunching files
- obtain ToAs using MTM
- run pat on DM-corrected data files

What's good and bad about the pipeline?

Good
- Written in tcsh which most people understand and can modify.
- Simple step-by-step processing that can easily be followed
- All important information stored in a single database and is properly linked to the original and intermediate files
- Can be run on multiple processors simultaneously
- Has been working well for >3 years without major modification
- Quite modular => easy to add in new calibration steps etc.
- Very straight-forward for the user to access ToAs

Bad
- Written in tcsh which means that it is really hard to tell where conditional statements end, etc.
- If the scripts are updated then it is not easy to know from the database which version of the code was used to process a given file
- No version control being used - i.e., updates are not tracked
- Makes use of lots of different programs and scripts
- Does not make the most of psrsh
- A lot of code is repeated for the J0437-4715 processing
- Reprocessing all the data is slow and could take many weeks.
- Do not have careful checking to ensure that we do not run out of disk space during the processing
- Do not have careful checking to test whether the codes being run (e.g., psrchive, fix_ppta_data) have been updated since the last time the pipeline was run
- Currently not running automatically - requires human intervention to start the scripts.
- The database is not automatically backed up

- The database needs manual intervention if the directory structure for the raw data files change.
- We do not store “goodness-of-fit” measures when forming the pulse arrival times.
- The “getTOAs” script does not yet produce ToAs in the “IPTA”-format.