

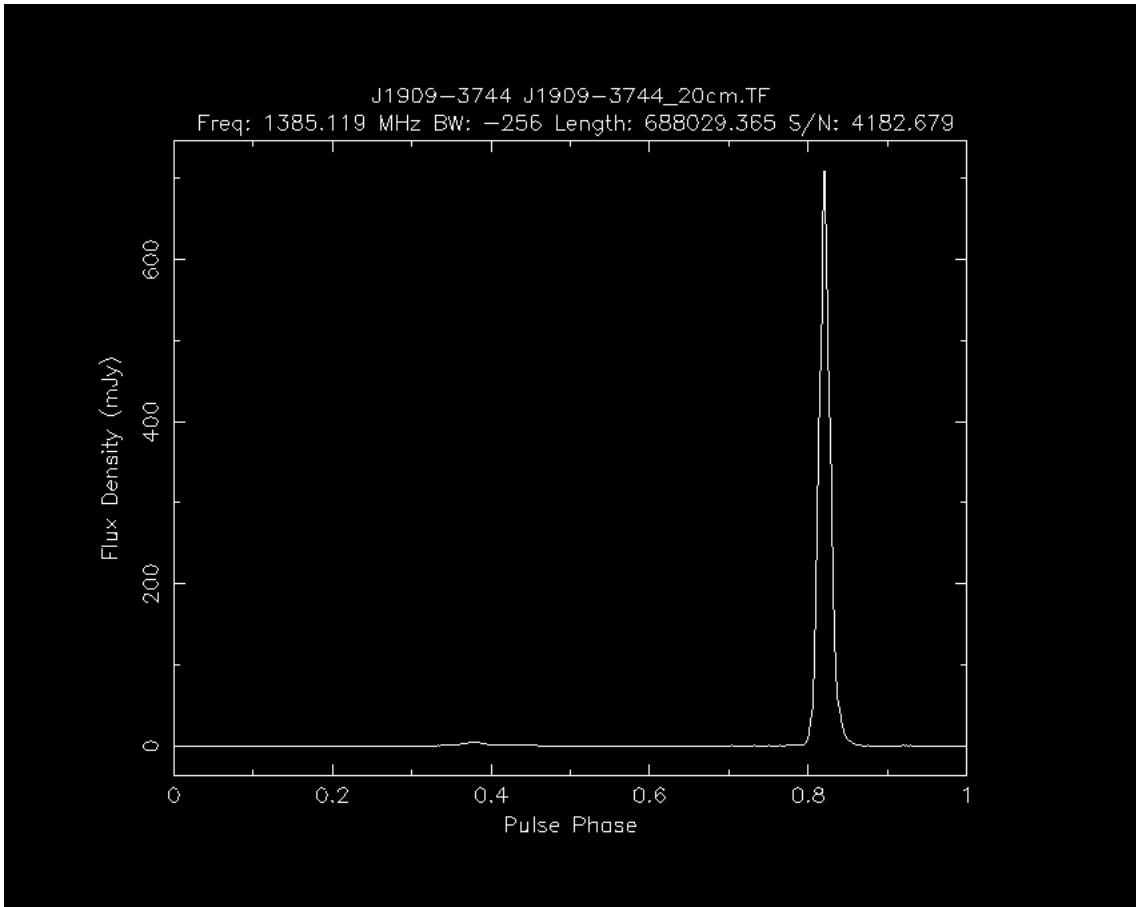
PSRCHIVE: Forming the best pulse profiles, ever

Ryan Shannon & Aditya Parthasarathy



Motivation

- We want to do the best tests of physics possible
- Need the best data sets
- Need the best instrumentation
- Need the best telescopes
- Q: How would you define “best pulse profiles, ever”?



Telescopes

- Pulsars are faint
- Large aperture = large gain
- Historical: Large aperture single-dish telescopes
 - (Exceptions) Westerbork
 - Don't need angular resolution
- Now: Interferometers
 - MeerKAT/LOFAR/VLA/SKA
- Good site (least RFI)
- Works at observing frequencies of choice



Receivers

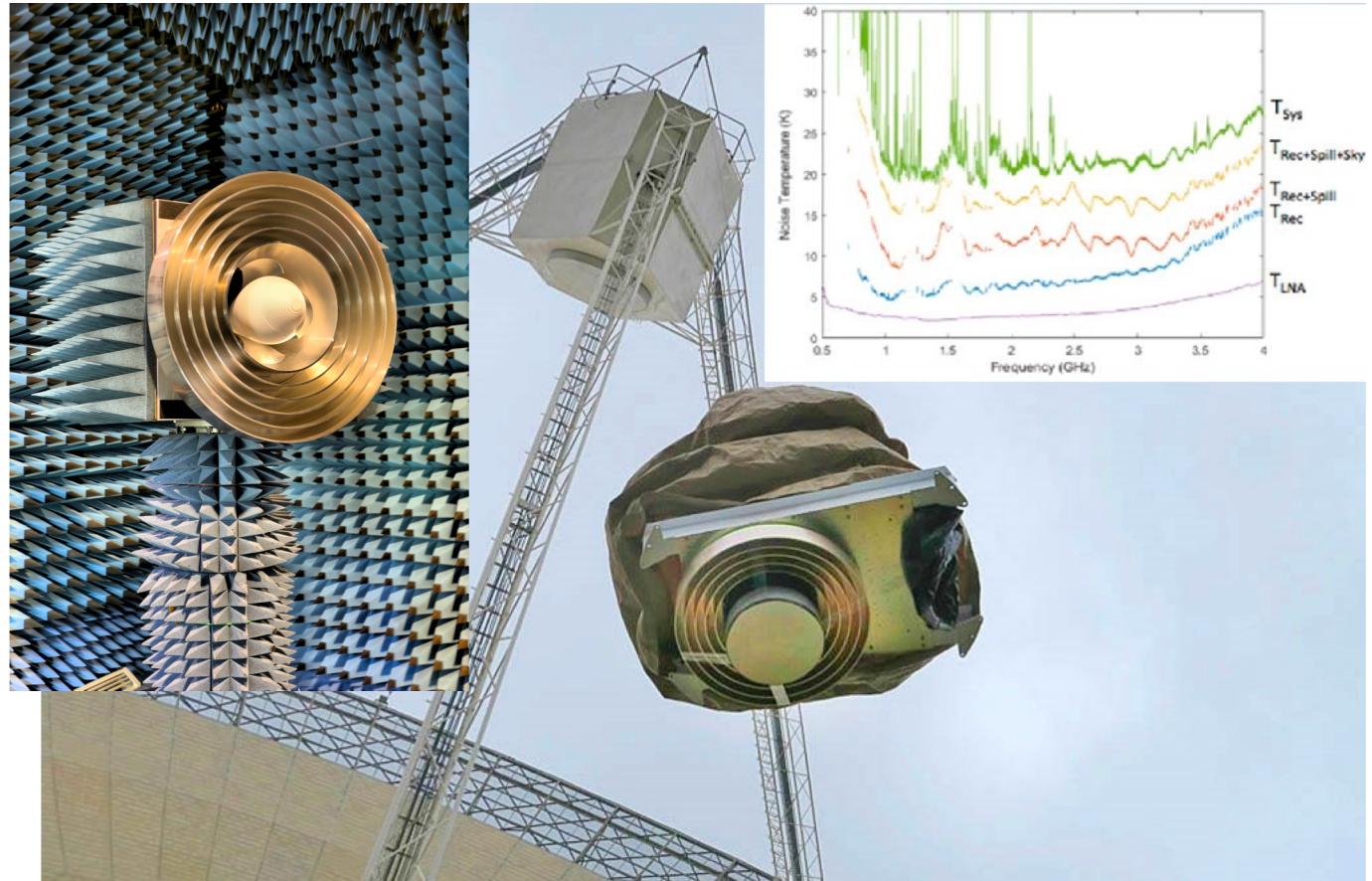
- Frontend
- **Observing frequency**
 - Good: Pulsars are strong at low frequency
 - Bad: Higher sky noise at low frequency (T_{sky})
 - Propagation effects become worse (see James's talk on ISM)
 - Precision timing band 800 MHz - 3.5 GHz (40 cm - 10 cm)
- **System temperature**
 - The lower the better
 - Can't build expensive systems for large arrays

GBT PF1 feed



Receivers

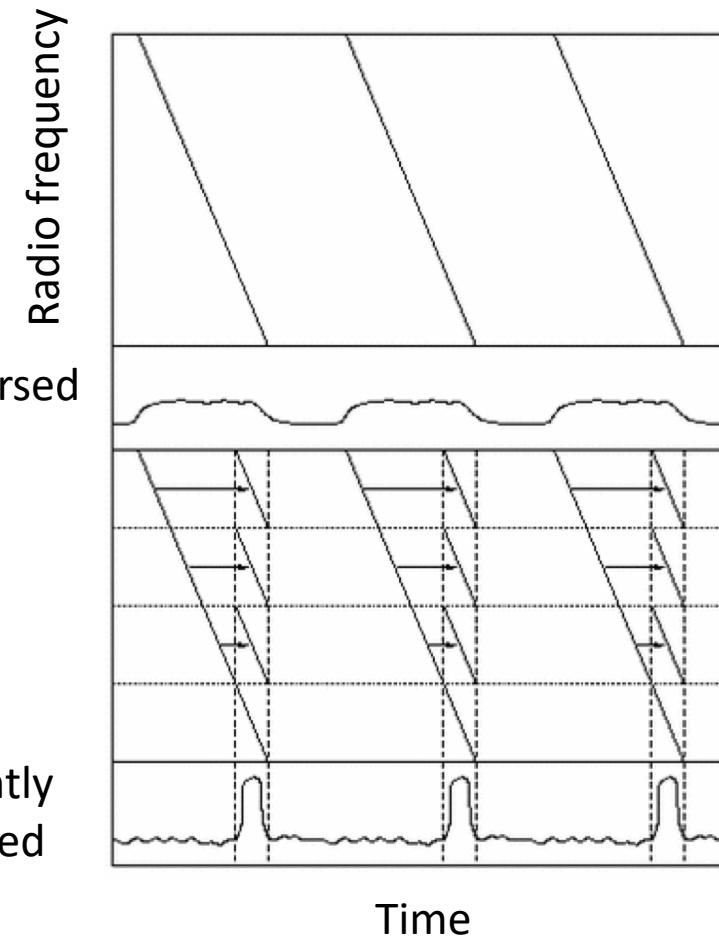
- **Bandwidth**
 - Previous: 10% fractional bandwidth
 - Current: 50-100% fractional bandwidth
 - Imminent: 600% fractional bandwidth
 - Issue: see all the RFI
- **Polarization**
 - Detect both polarization bases of the field
 - (Circular or linearly polarized feeds)
- Output: $V_x(t) V_y(t)$ or $V_L(t) V_R(t)$ or
- $V_A(t) V_B(t)$ for generic basis



Parkes ultrawide band system:
Should work from 700 MHz -4.2 GHz

Backends

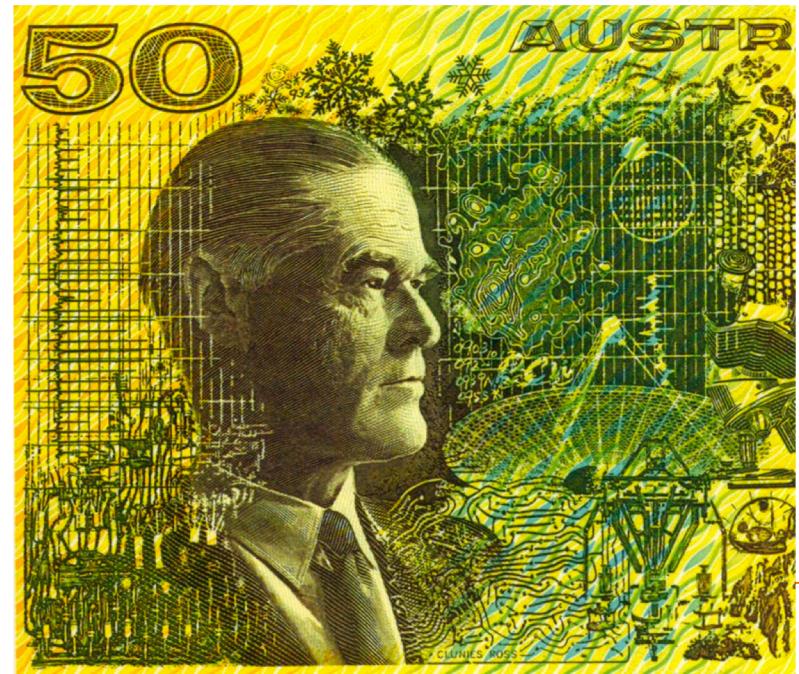
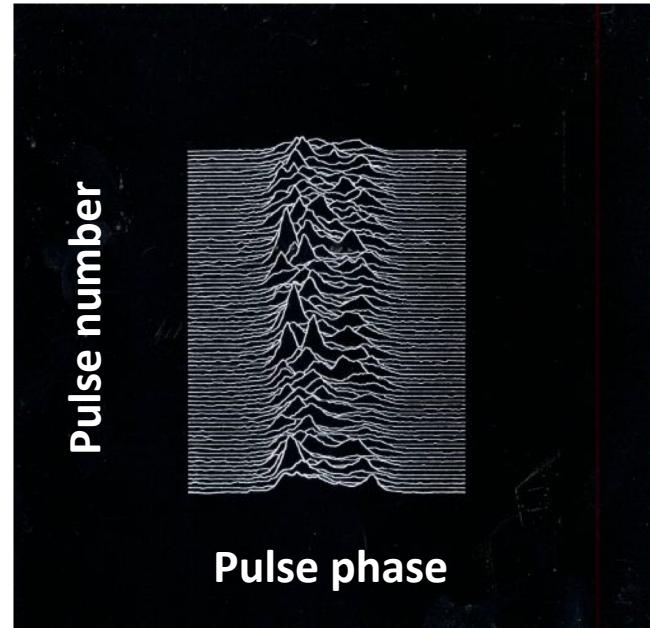
- Digitize voltage time series
- Channelize into subbands/channels Non-dispersed
 - Isolate narrow band RFI
 - Optional: Filter voltages with *coherent dedispersion* to account for in-channel dispersion sweep
- “Detect” channelized data
 - Form coherency parameters
 - AA, BB, A*B, AB*
 - Stokes I = AA + BB if receiver is perfect



From Lorimer & Kramer

Data types

- **Voltages ("Baseband")**
 - Highest data rate
 - No loss in information
 - Decide post-facto how to analyse data/
don't have sufficient compute to process in
real-time
- **Search mode:**
 - Output intensity measurements at high
time resolution (64 microsec)
 - Use for pulsar searching, but also transient
searches (FRBs) and single pulse analysis
- **Commonly use `dspsr` to analyse these
data set types**



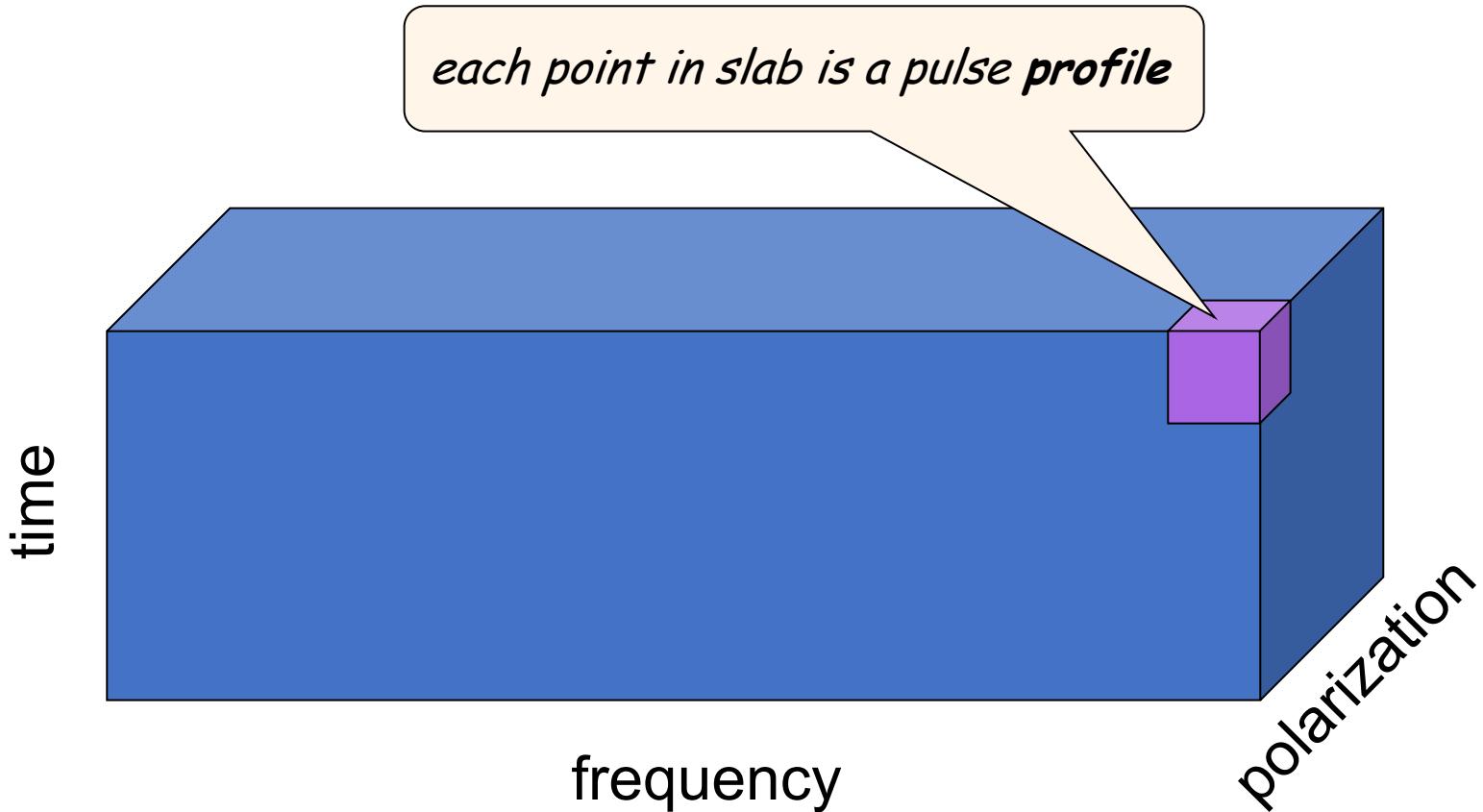
Data types

- **Fold mode:**

- Fold intensities at period of known pulsar
- Average together for a fixed amount of time (8-30sec) “subintegrations” (i)
- Can get higher pulse phase resolution than “search mode” (microsec)
- Much lower data rates

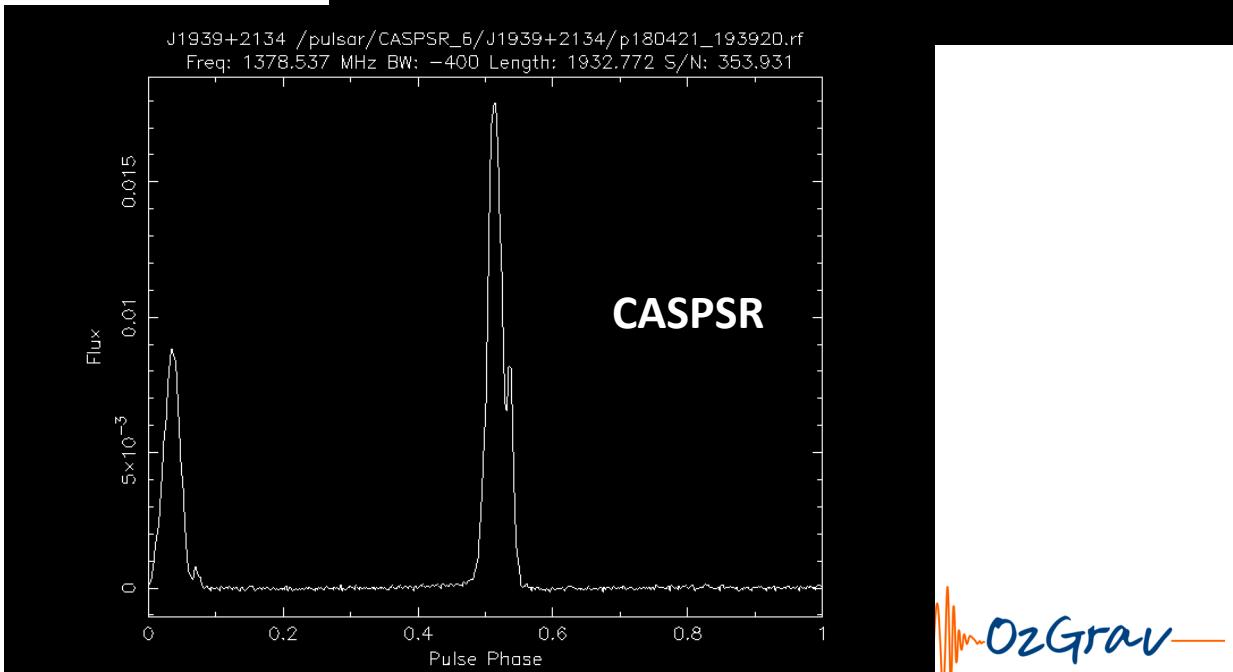
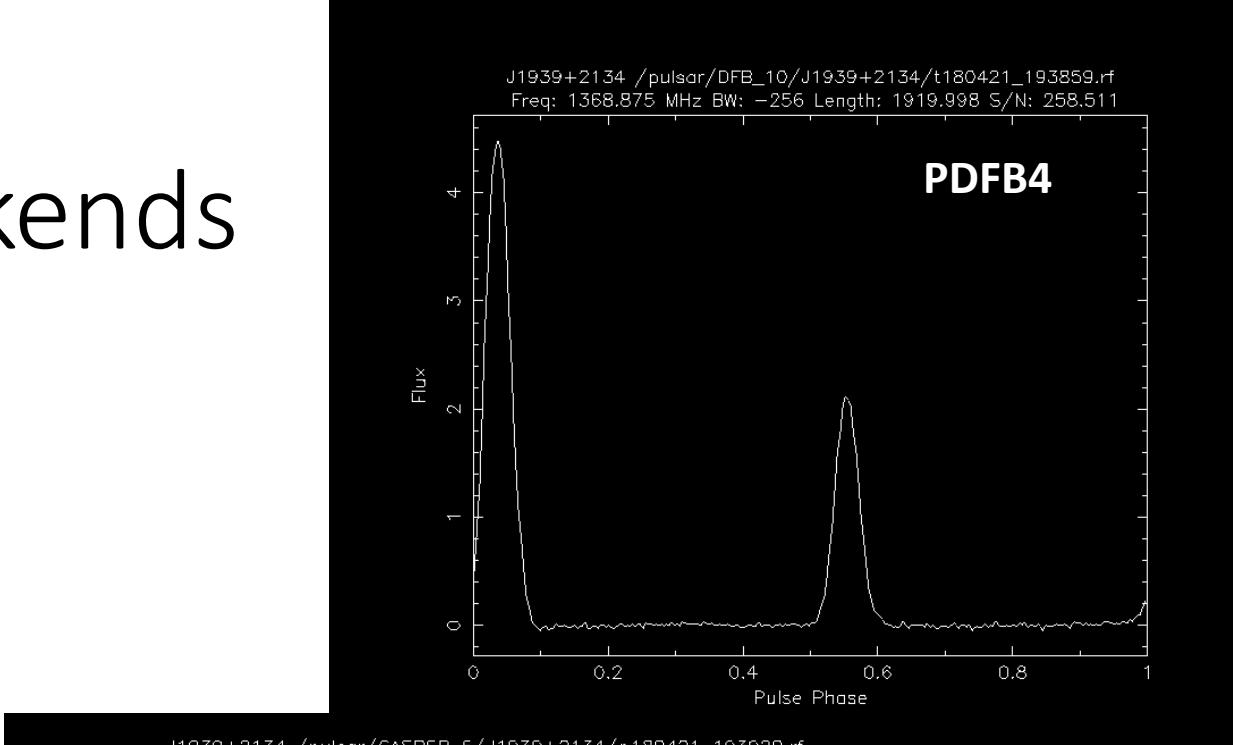
- Data cubes for coherency parameters:
 - $AA(\phi, f, i)$, $BB(\phi, f, i)$,
 - $A^*B_i(\phi, f, i)$, $AB^*_i(\phi, f, i)$

psrchive is used to process these data cubes



Examples of pulsar backends

- Examples of timing backends:
 - GUPPI/PUPPI/CASPSR – coherent dedispersion
 - Now typically implemented on GPU cluster
 - PDFB4 – incoherent dedispersion
 - FPGA
 - DM smearing is not an issue if “DM/P” is low

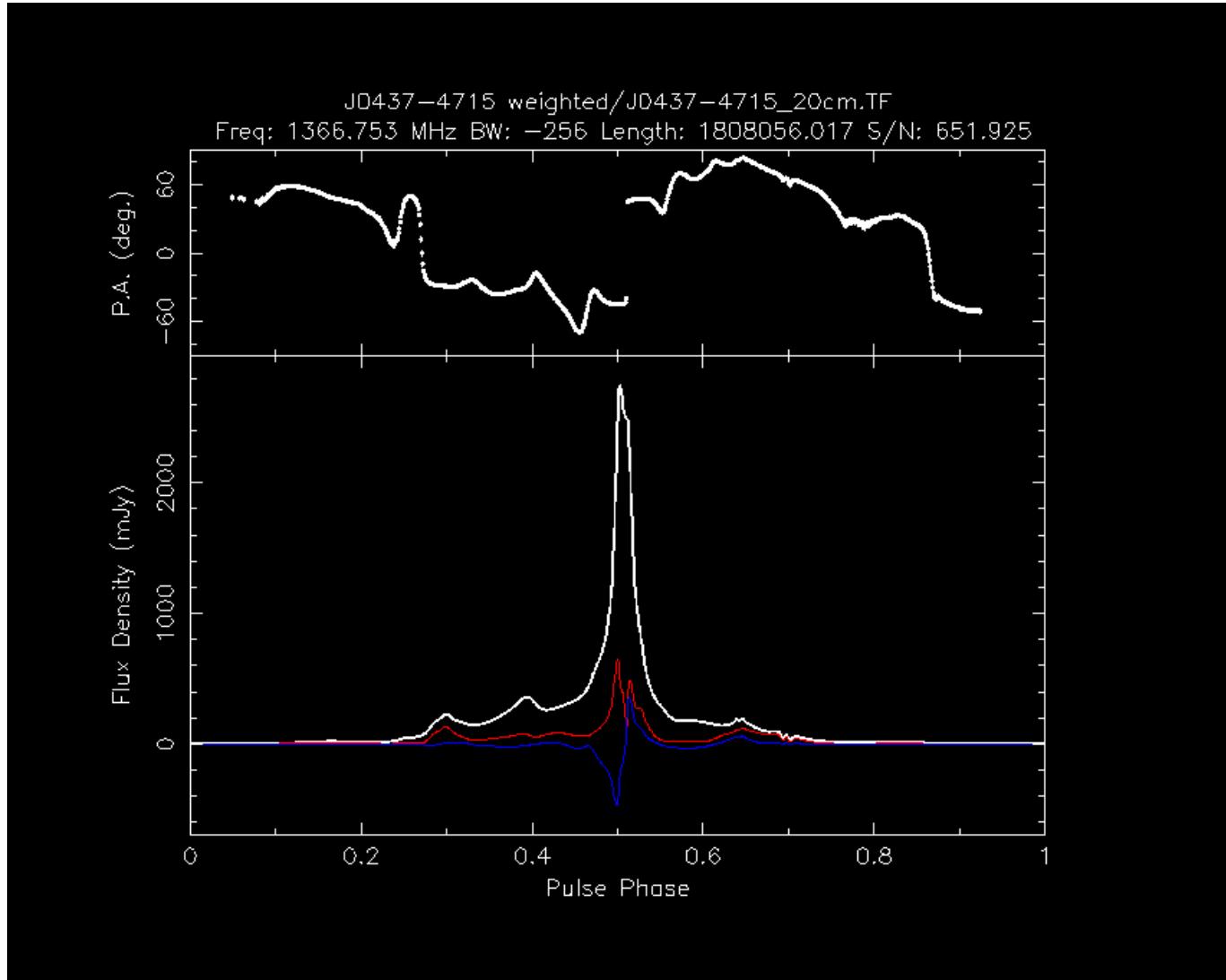




psrchive plot of pulse profile

Linear polarization position angle

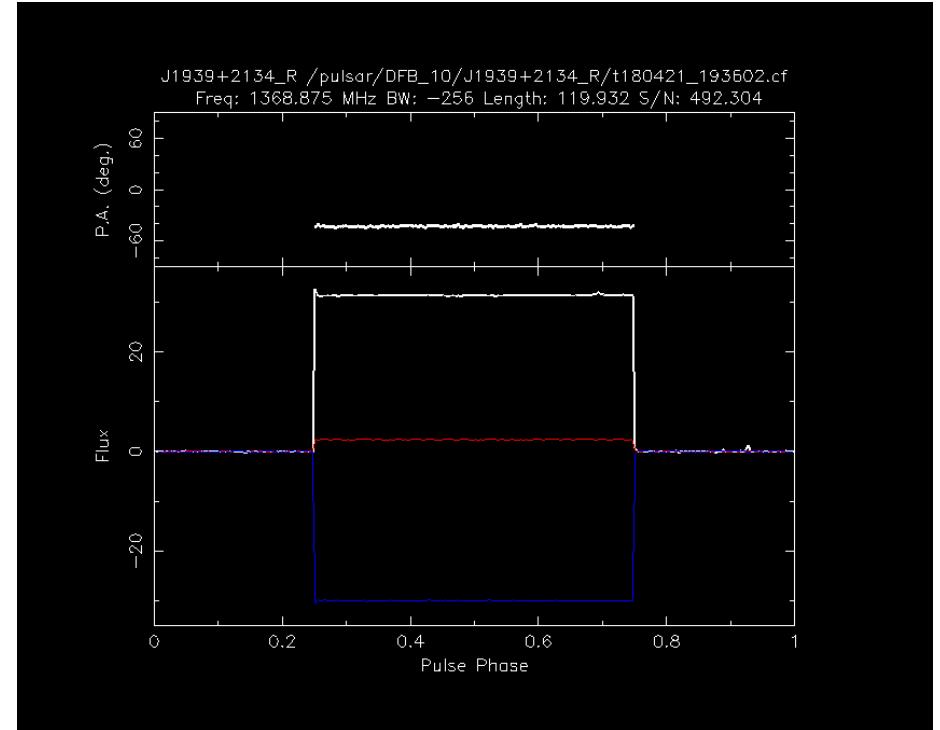
White: total intensity
Red: linear polarization
Blue: circular polarisation



Dai et al. (2015)

Polarization and polarization calibration strategies

- Pulsars are polarized
- Pulsar timing is typically conducted on stokes-I (total intensity)
- Mis-calibration can distort the shape of pulse profile and bias pulse arrival times
- Types of mis-calibration that will distort pulse shape
 - Gain variations in X and Y polarization
 - Inject noise calibration signal
 - Instrumental leakage:
 - Track bright source and solve for terms (PCM)
 - Self calibration: use pulsar with known polarization and solve for cross coupling terms (METM)



Profile of calibrator signal

Typical observing strategies for precision timing

Example for PPTA:

For each pulsar

- Observe a noise calibration signal (90 sec)
- Observe pulsar (3840 sec)
- (Observe noise calibration signal again, 90 sec)

Other calibration observations:

- Flux calibration observations occasionally (once per observing run)
- Occasional PCM observing runs (once every 6 months)
 - Observe PSR J0437-4715 from rise to set

Recipe for forming calibrated pulse profiles

- Excise radio frequency interference
- Excise band edges (important if there is scattered power)
- Calibrate the polarization
- Update ephemeris
- Average in frequency and time to manageable resolutions
- Time the pulse profiles

Error recognition

- Compare profile to template or previous observations
 - Does it look the same?
 - What do profile residuals look like?
 - Does polarization match what would be expected?
 - Does pulsar have S/N as expected
 - Develop heuristics/statistics to identify bad data in large data sets

Arrival-time analysis:

- Are TOAs outlying?
- Are there jumps?

PSRCHIVE is:

- **not** a single monolithic program
- a suite of programs
 - integrated with the UNIX environment
- a C++ development library
 - python bindings also available
- a mature work in progress

PSRCHIVE is:

- Open Source
- widely used
 - Africa, Australia, Canada, China, Germany, Netherlands, United Kingdom, United States, ...
- relatively well documented
 - <http://psrchive.sourceforge.net>

PSRCHIVE is:

- powerful!
 - sophisticated calibration
 - matrix template matching
 - advanced rotation measure estimation
 - unique rotating vector model fitting
 - digitization distortion corrections
 - custom virtual memory management
 - etc.

Why use  PSRCHIVE ?



PSRCHIVE can:

- read/write many folded data formats:
 - PSRFITS, EPN, PRESTO, ASP, WAPP ...
- perform many common tasks:
 - correct dispersion and Faraday rotation
 - calibrate instrumental polarization
 - excise corrupted data (e.g. RFI)
 - calculate arrival times
 - produce various publication quality plots

PSRCHIVE cannot:

- search for new pulsars:
 - sigproc, presto, etc. do this
(used to refine S/N of survey candidates)
- reduce/fold time series data:
 - dspsr, sigproc, presto, etc. do this
(dspsr uses psrchive)

A screenshot of a web browser window titled "PSRCHIVE". The address bar shows the URL "http://psrrchive.sourceforge.net/". The page content includes the PSRCHIVE logo, a search bar with "Search PSRCHIVE:" and "Find" button, and a "powered by FreeFind" link. The main content area features a section titled "The PSRCHIVE Project" with a description of the library's purpose and a blue arrow pointing to the "Install" link. To the left, there is a sidebar with links for Home, Install, Use, Develop, Support, and News.



PSRCHIVE

Search PSRCHIVE:

Find

powered by [FreeFind](#)



Home



Install



Use



Develop



Support



News

Credits

hosted by

sourceforge

The PSRCHIVE Project

PSRCHIVE is an Open Source C++ development library for the analysis of pulsar astronomical data. It implements an extensive range of algorithms for use in pulsar timing, scintillation studies, polarimetric calibration, single-pulse RFI mitigation, etc. These tools are utilized by a powerful suite of user-end programs that come with the library. The software is described in [Hotan, van Straten & Manchester \(2004\)](#).

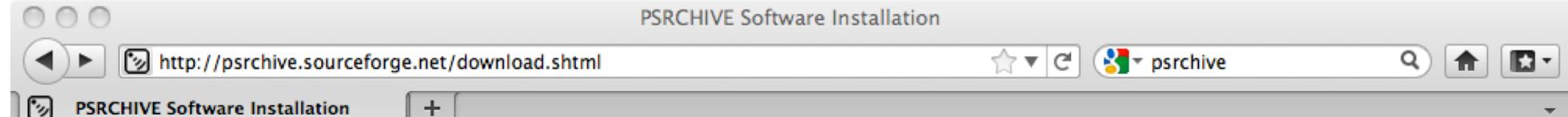
Portability

PSRCHIVE was designed to increase the portability of both algorithms and data. The software is installed and compiled using the standard GNU configure and make system. It is also able to read astronomical data in a number of different file formats, including:

- [PSRFITS](#), a standard data storage format developed at the Australia Telescope National Facility;
- [EPN](#), the file format of the European Pulsar Network;
- Timer, used primarily at the Parkes Observatory; and
- [PuMa](#), an instrument at the Westerbork Synthesis Radio Telescope.

Copyright © 2006-2010 Willem van Straten
Licensed under the Academic Free License version 2.1





Home



Install



Use



Develop



Support



News

Credits

hosted by

sourceforge

PSRCHIVE Software Installation

[Complete Installation Instructions](#)



The latest stable version of PSRCHIVE can be downloaded as a single file. Alternatively, the development version of the code can be checked out of the Git repository. The installation instructions are slightly different in the two cases.

Stable Release

Download: PSRCHIVE version 13.4 [psrrchive-13.4.tar.gz](#) (2.1 MB) was released on 20 August 2010.

Install: Please refer to the [stable release installation instructions](#).

For a list of previous stable releases and the most significant changes between versions, please see the [change log](#).

Development Branch

Download: The latest version of PSRCHIVE is available via the [Git repository](#) from SourceForge.

Install: Please refer to the [development branch installation instructions](#).

PSRCHIVE Manuals

http://psrrchive.sourceforge.net/manuals/ psrrchive

PSRCHIVE Manuals +

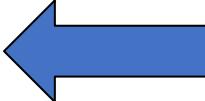
PSRCHIVE

Search PSRCHIVE: Find
powered by FreeFind

PSRCHIVE Applications

The following table provides links to the user manuals of the various pulsar data processing applications that are distributed with the PSRCHIVE package.

A step-by-step [User's Guide](#) is also under development.

Use 

Core Applications

psredit	query or change metadata
psrstat	query attributes and statistics
psradd	combine data in various ways
psrsh	command language interpreter
psrplot	produce customized, publication quality plots

Text-based interfaces

vap	output tables of parameters and derived values
pdv	view some basic data as text

Graphical interfaces

pav	produce a wider variety of plots
psrgui	interactive plot interface
pazi	interactive plotter and zapper

General data processing

pam	command line general purpose data reduction
psrconv	convert from one file format to another

Credits

hosted by **SOURCEFORGE**

OzGrav

PSRCHIVE Core Applications

- standard command line options
 - remember once, use often
- powerful command language
 - full functionality in every program
- evaluation of mathematical expressions
 - variety of statistical tools

Use PSRCHIVE to ...

- get to know your data
 - query and edit: `psredit`
 - evaluate: `psrstat`
 - plot: `psrplot`
- modify your data
 - command: `psrsh`
- combine your data
 - integrate: `psradd`

Tutorial

- Goal: learn command line functionality of psrchive
 - Understand data sets/files
 - Visualise data sets
 - Basic calibration procedures
 - Make TOAs
-
- This is not what psrchive is limited to.