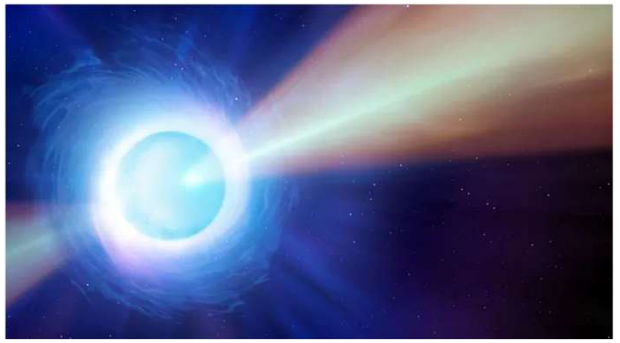
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

PRESTO Jerksearch GPU porting- updates and developement status

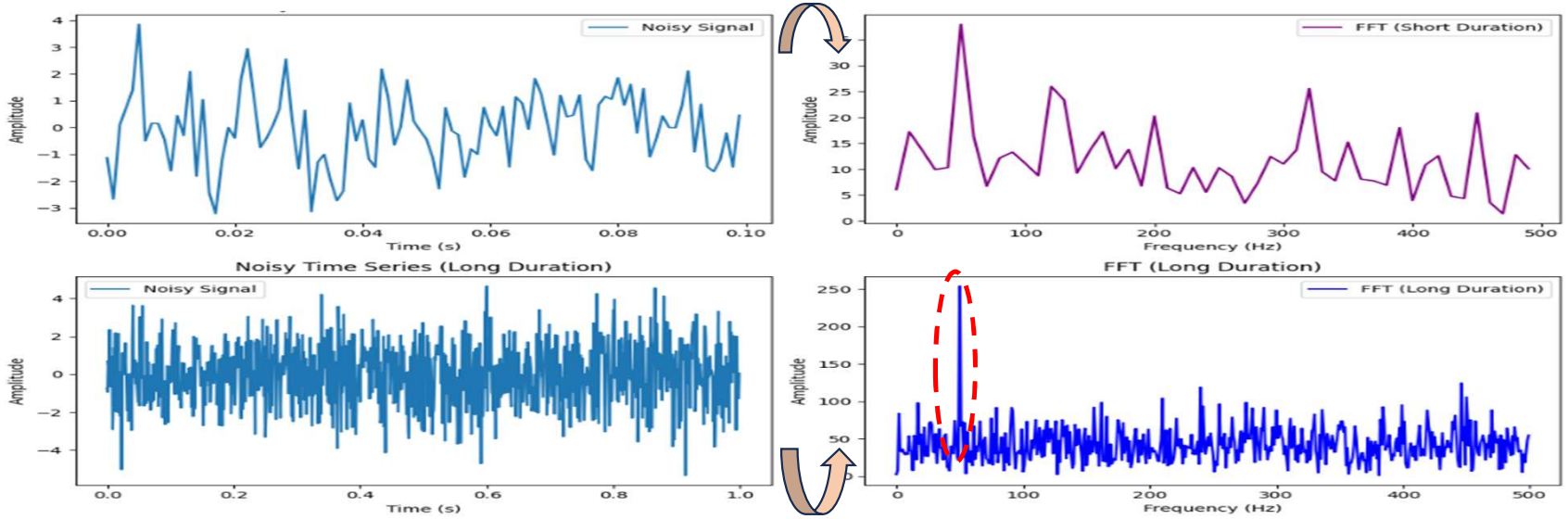
- By Rouhin Nag (PhD student , University of Cagliari and OA Cagliari)
Supervisor: Dr. Andrea Possenti

Spoke 3 III Technical Workshop, Perugia 26-29 Maggio, 2025

Pulsars

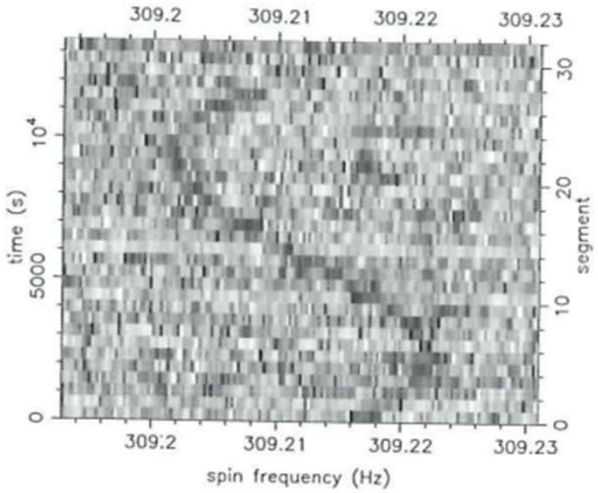


- Fast Rotating, highly magnetized neutron stars
- Cosmic "*Lighthouses*": Radio Emission from magnetic poles
- Misaligned magnetic and spin axis
- One pulse per rotation along line of sight towards either beam
- The time taken for one rotation i.e the interval between two pulses is called the spin period.



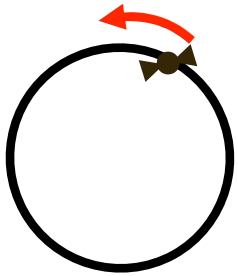
Binary systems

If the pulsar is in a binary the rotational period will change slightly during the course of the observation.

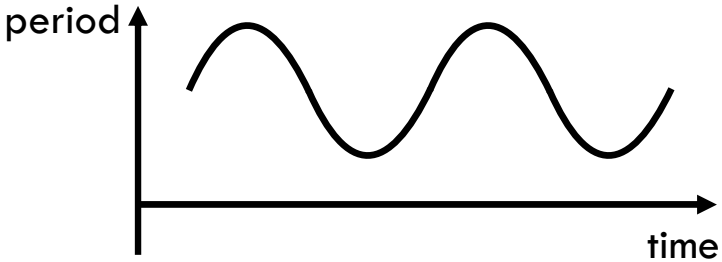


Equation for Doppler shift:

$$P'(t) = P_0(1 + v(t)/c)$$



Evolution of the period in a circular orbit:



This Binary orbital motion and shift in observed frequency of the pulsar due to Doppler effect are handled by **Acceleration searches and Jerksearches**.

Scientific Rationale

- The Acceleration and Jerk searches are our only weapons to effectively look for Binary pulsars, especially binary systems in tight orbits.
- These are computationally heavy and highly iterative processes that are ideal for being handled by GPU's
- Acceleration searches had been ported into GPU's in 2018 and showed runtime improvements of an order of x10 or even more from the CPU only version.
- Therefore my aim is to extend this work and port the Jerksearch into GPU's as well.

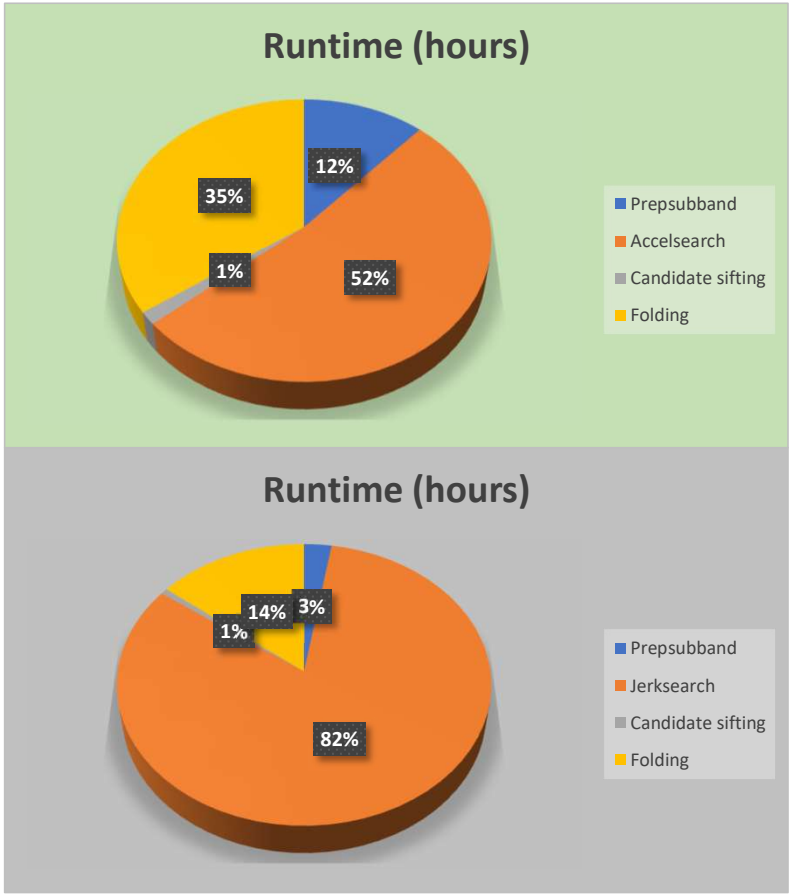
Technical Objectives, Methodologies and Solutions

- **Objectives**: To port the existing CPU only Jerksearch pulsar search routine into GPU.
- **Methodologies**:
  - Test the existing CPU and GPU implementations of existing Acceleration search code and the existing CPU implementation of Jerksearch code to obtain data of Runtime, memory usage, sensitivity etc and understand the working of the codes on available HPC infrastructure.
  - Explore the two main methods and available softwares of performing these acceleration and Jerksearch routines to figure out the most optimal approach in various aspects.
  - To check whether any improvements can be made not just to the speed but also the sensitivity of the Jerksearch code.

Runtimes of various parts of the PRESTO based search Pipeline

- The runtime logs of various routines of the PRESTO* pulsar search pipeline (Pulsar Miner v2.0**) on a 2 hour Meerkat dataset.

- The graphs show the runtimes for the four major routines of the cpu pipeline version of **acceleration search** (top figure) and then **Jerksearch** (bottom figure).



*https://github.com/scottransom/presto.git
**https://github.com/alex88ridolfi/PULSAR_MINER.git

Runtime Benchmark tests (PRESTO 2018).

Computing resource used

- 14 computing nodes at our OAC computing cluster.
- 8 CPU nodes with 64 AMD Milan 7513 (16 core) each, 512 GB DDR4 3200 RAM ,960GB SSD and 4 x 1.92 TB SSD
- 6 GPU nodes with the same configuration and the addition of 2 GPU's (NVIDIA A40 48GB RAM DDR6 Pcle) per node

Computing time requirements

- An acceleration search run over **one MeerKAT S-band Globular Cluster** observation took me ~60 hours with **256 CPUs and 8 GPUs**. Within this, acceleration search alone takes **50-60%** of the computing time.
- With Jerk search enabled, these searches can take up to a week to finish.

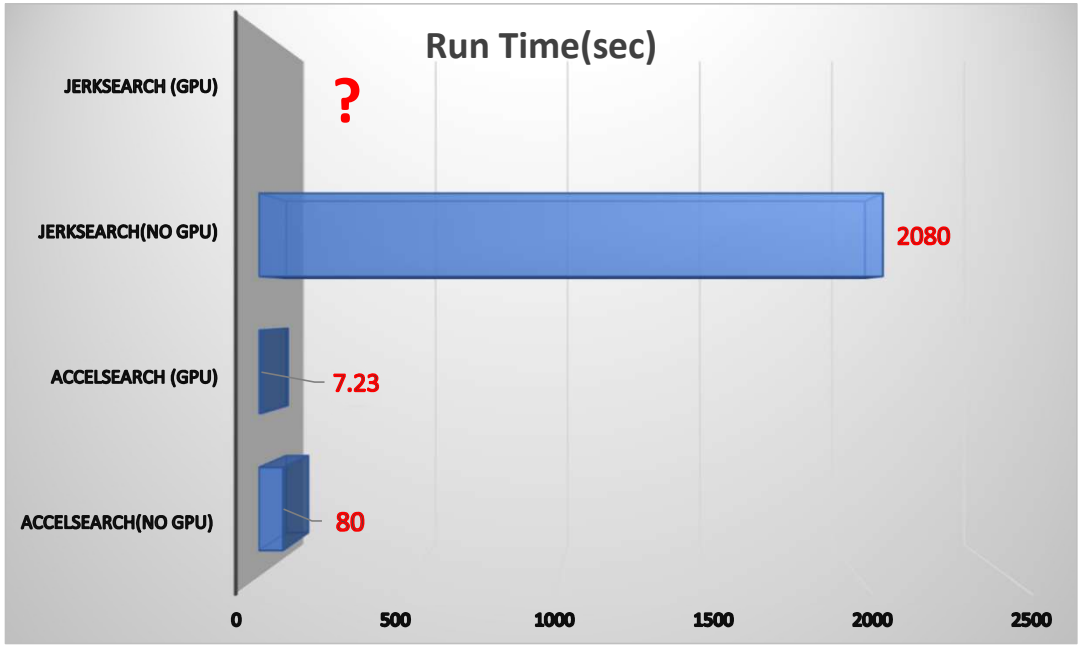


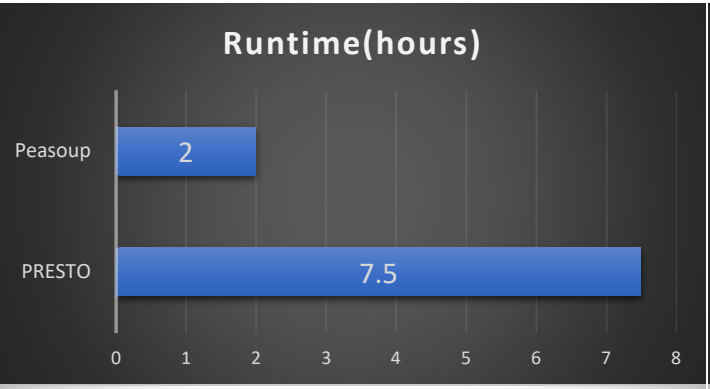
Fig (above): A runtime comparison of Acceleration search and Jerk search for the same 15 minute time series.

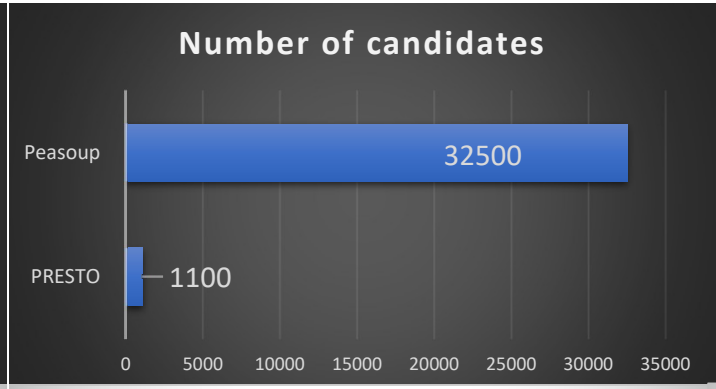
Is GPU porting the only thing we can do?

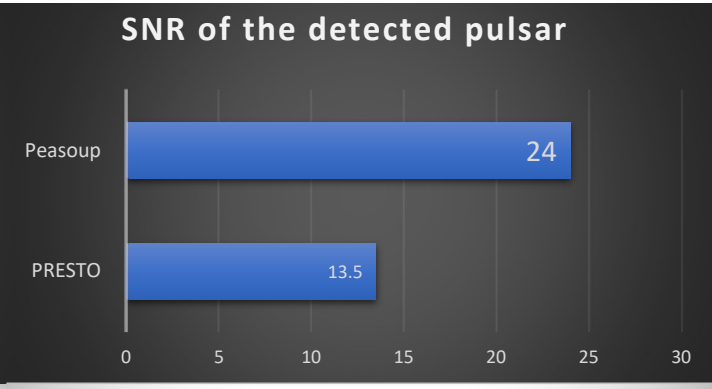
- From the previous graphs It is clear that GPU porting can definitely help us in bring down the time required to perform Jerksearches. But its not the only thing to be explored.
- I started exploring another similar pulsar searching software called **Peasoup*** which also performs acceleration and Jerksearches but has a slightly different approach than **PRESTO**.
- While PRESTO performs these search algorithms in the **frequency domain**, **PEASOUP** approaches it from the **time domain**.
- Also, apart from just the problem of reducing **runtime**, I wanted to approach the problem from the context of **sensitivity** as well.

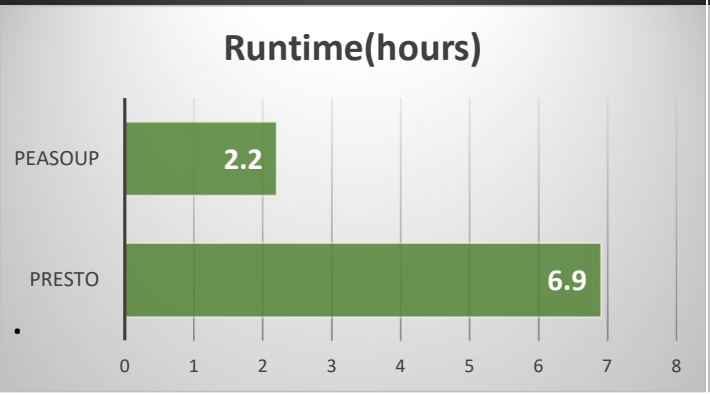
*https://github.com/vishnubk/peasoup.git

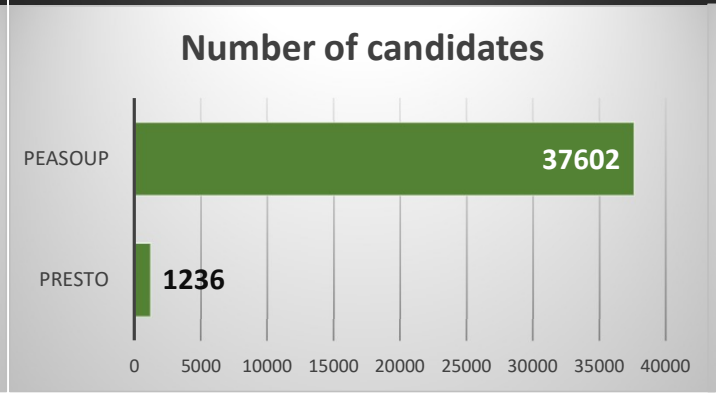
PRESTO vs Peasoup: Figure above contains comparison plots of **acceleration search** runs for **GPU versions** of PRESTO vs. Peasoup for two different datasets (grey and black). Both the observations were 1 hour long and had only 1 pulsar in them

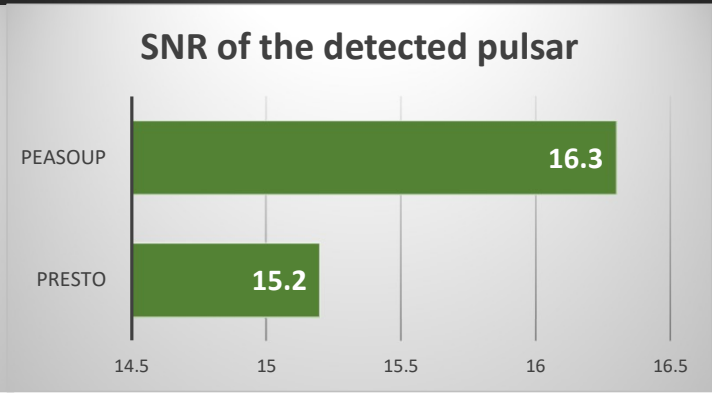












Plans to enhance the Jerksearch code.

- Acceleration and Jerk searches are essentially brute force methods. So, there is still a chance of missing pulsars which are not very bright or systems with extreme orbital parameters that are in an unfortunate orbital phase during the observation.

- We can go 1 step further in enhancing the sensitivity of our pulsar searches with a technique called the template bank method.

- We use the Template bank to model the characteristic features of a pulsar binary system like masses of the pulsar and the companion, the spin period of a pulsar, the eccentricity of the binary orbit, the orbital velocity etc and helps us create searches with higher sensitivity.

Work flow

Step 1

- Device a **new Template bank model** to minimize the number of dimensions of my parameter space while keeping sensitivity. Existing models had 3D parameter space. My model has 2D space.
- Test the **parameter space coverage of the new model** vs the existing model(s)

Step 2

- **Write the template bank Generation code and supporting codes that make it work and incorporate it into the search pipeline**
- Build a **new GPU based Time Domain Resampling code** that uses my template bank code to perform periodicity searches.

Step 3

- Test the code with **artificial data** from a pulsar injector software.

Step 4

- Test the code with real data and roll out the software for public use.

Key Results.

Generation of the new parameter space model, 4D to 3D space conversion

$$\phi_2 = 2\pi f\,[t - \left(\frac{a}{2c}\right)t^2 - \left(\frac{j}{6c}\right)t^3]$$

f = spin frequency
t = time
a = acceleration
j = jerk

$$\phi_1 = 2\pi f\,(t + \tau \sin(\Omega t + \psi)) + \phi_0'$$

f = spin frequency
t= time
Ω= orbital angular velocity
τ= projected orbital radius
ψ= initial orbital phase

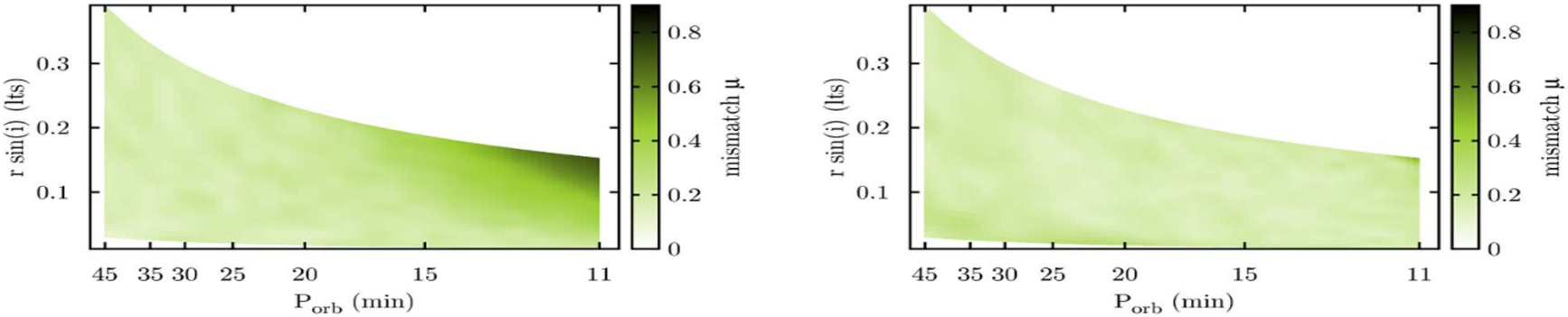


Fig (above) shows parameter space sensitivity models for 90% coverage of parameter space. My new phase model (left) vs the most robust phase model (right) that takes into account all possible parameters.

Performance of my new GPU based, time domain resampler for Jerksearch.

- With a 100,000,000 point data series with 8bits per sample and an NVIDIA A40 48GB RAM DDR6 Pcle,
- One instance of a Jerk search run comprises of resampling time (~0.2ms) + Fourier domain search time (~ 5-10 sec). So, **one instance of Jerk search is in the ballpark of ~ 5-10 sec**.

- Scaling this to a full scale run on HPC infrastructure is not estimated as of now but we will report figures as soon as my pipeline is fully complete and I can run benchmark tests.

- **In the Past month, a GPU version of PRESTO Jerksearch (dubbed PRESTOzl) has been developed by a collaboration between a Chinese research team and a private software company. (https://github.com/zhejianglab/PrestoZL.git)**

PrestoZl vs My code (tentative name: PULSEJET)

- The runtime of one single instance of Jerksearch in my code is significantly shorter than PRESTOzl. (~ x 17)

- Disclaimer: The number of jerksearch iterations in my code, per run, will be significantly higher than PRESTOzl, so, a full runtime and sensitivity comparison can only be achieved after my pipeline is complete.

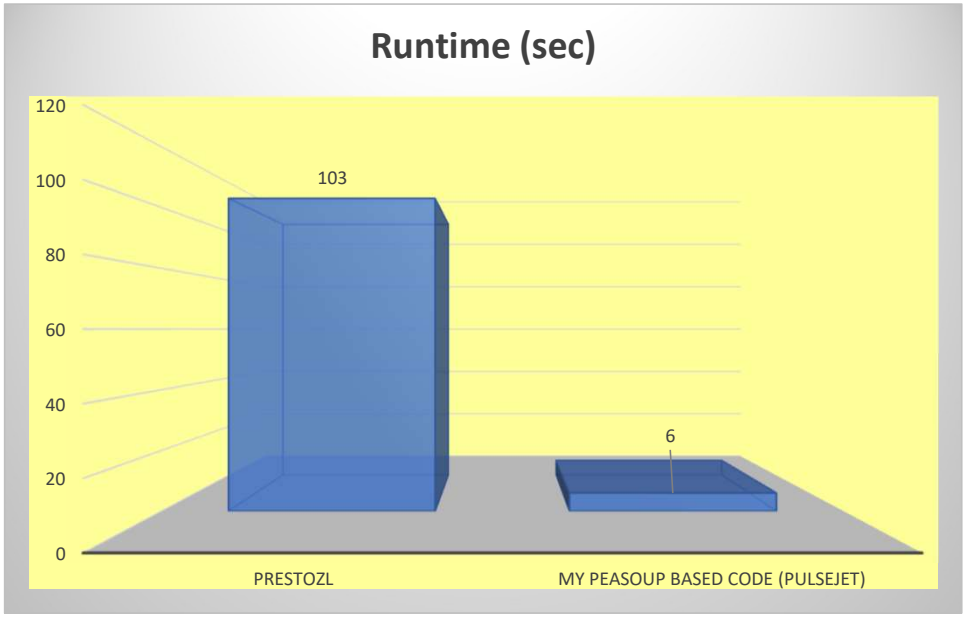


Fig (above) shows runtime comparison for a single instance of Jerksearch between PRESTOzl and my code for a 100,000,000 point timeseries on a NVIDIA A40.

Future work.

Complete the work on my pipeline by September 2025*

Test my code against the new PRESTOzl version over various parameters

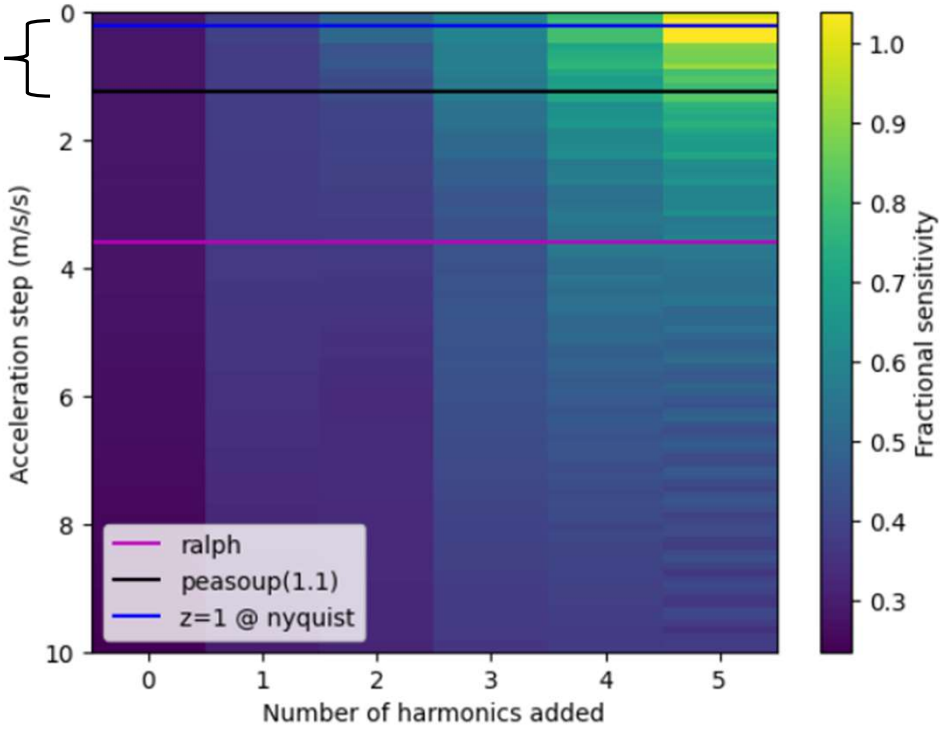
Roll out the code for public use and work on publications

- My code is being developed as a part of the ERC funded project COMPACT, with our collaborators at the Max Planck institute for Radio Astronomy, Bonn.
- You can find the beta version my code on Github: https://github.com/erc-compact/peasoupjerk.git. In line with our **KPI for August 2025.**

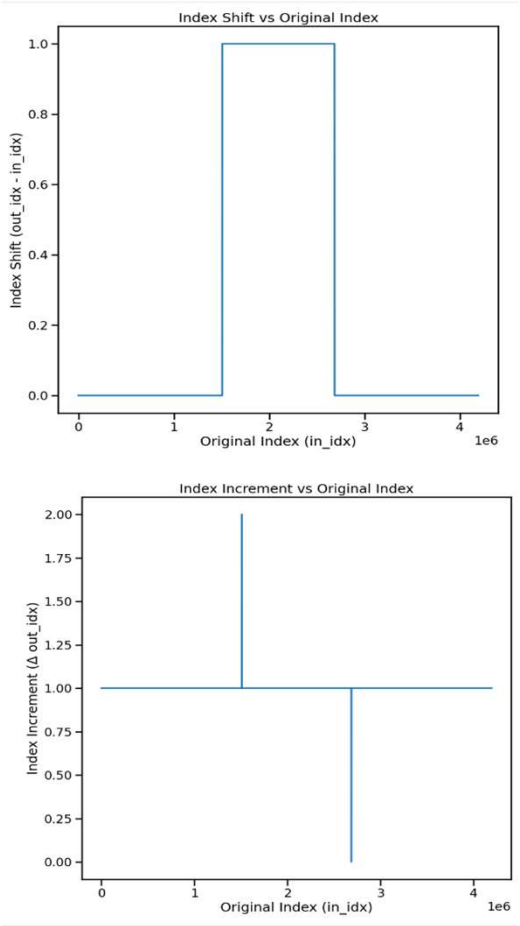
Thanks for listening

Search Sensitivity of different methods

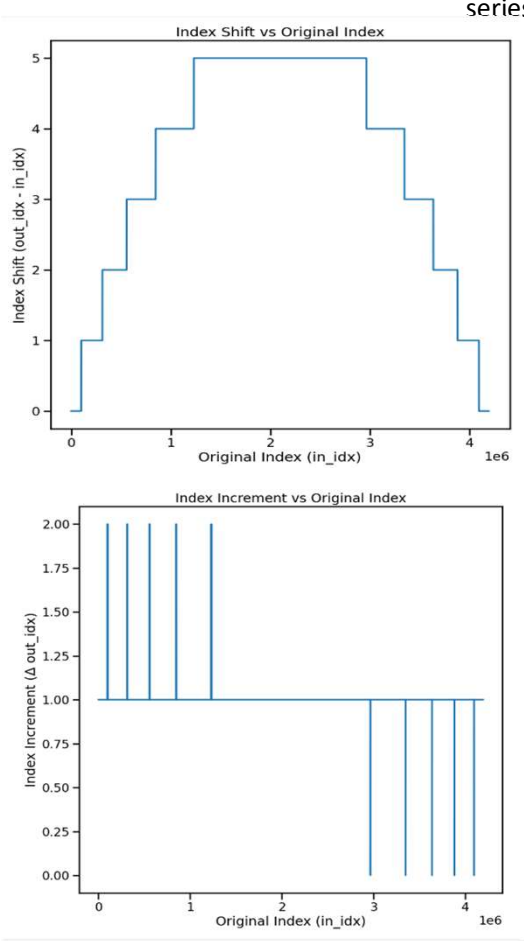
Theoretical sensitivity of Template bank based Jerk searches.



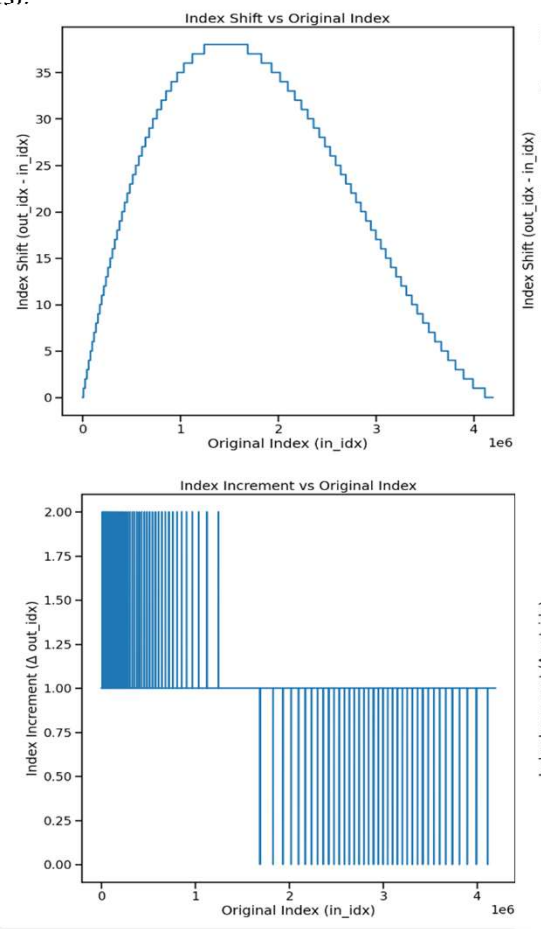
Computational complexity of Jerk searches: The graphs below show how many times a time series needs to be resampled to take into account the effect of the given acceleration and jerk values for a specific pulsar given the starting point of reference in the observation is at the centre of the x axis (Indicates number of samples in the time series).



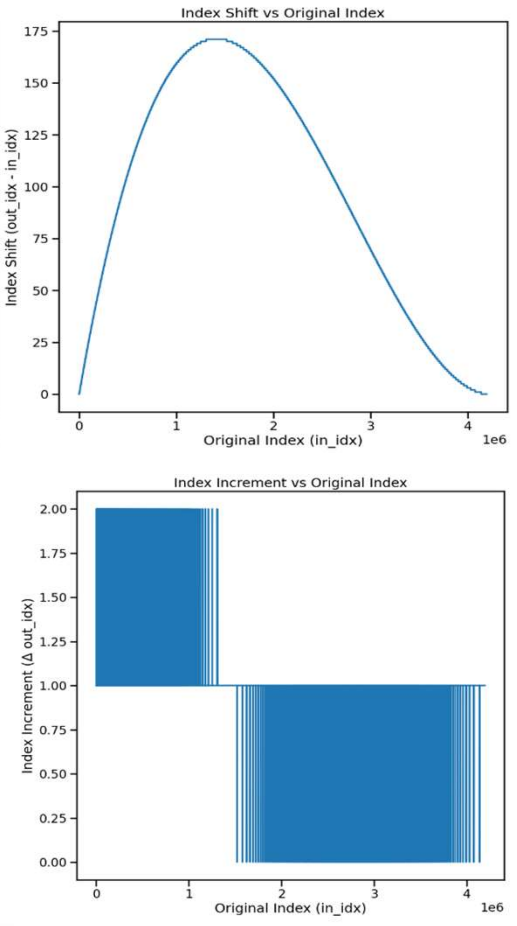
a= $-1m/s^2$
j= $0m/s^2$



a= $-10m/s^2$
j= $0m/s^2$



a= $-10m/s^2$
j= $1m/s^3$



a= $-10m/s^2$
j= $5m/s^3$