Using the HMM glitch detector

Liam Dunn, Andrew Melatos, Sofia Suvorova, Bill Moran, Rob Evans

March 15, 2022

Contents

1 Introduction

This document outlines the use of `glitch_hmm`, a hidden Markov Model (HMM)-based pulsar glitch detection code which is described in detail by Melatos et al. (2020). The purpose of `glitch_hmm` is to provide a quantitative framework for the detection of pulsar glitches based on time of arrival (ToA) data. The package consists of three components:

- A core MATLAB component which contains the HMM implementation (`matlab_code_logs/` in the repository).
- A Python library which takes a `.par` and `.tim` file and constructs an HMM to be passed to the MATLAB code (`pulsar_hmm/` in the repository).
- A Python wrapper which provides a `.ini` file-based interface to the above two components (`hmm_wrapper/` in the repository).

The main system requirements are the following:

- Python 3
- A recent version of MATLAB (tested with 2018b+), with Image Processing Toolbox installed
- TEMPO2
- LIBSTEMPO

2 Method overview

This section provides an overview of the methodology and algorithms used in `glitch_hmm`. A comprehensive treatment is given by Melatos et al. (2020).

2.1 Hidden Markov models in general

Hidden Markov models (HMMs) provide a framework for analysing the evolution of a system whose underlying dynamics are *Markovian* (memoryless) and which is observed indirectly (the true state of the system is *hidden*). The system is taken to be discrete in time, with snapshots taken at times in the set $\{t_1, t_2, \ldots, t_{N_T}\}$. The hidden state of the system is denoted $q(t_n)$ and can take on N_Q distinct values. The evolution of the hidden state of the system is determined by a transition matrix

$$A_{q_j q_i} = \Pr\left[q(t_{n+1}) = q_j \mid q(t_n) = q_i\right]. \tag{1}$$

An observation of the system yields a result $o(t_n)$. Note that the result of an observation is not necessarily drawn from a discrete set. The probability of an observation $o(t_n)$ given that the system is in the state $q(t_n) = q_i$ is referred to as the emission probability, and is given by

$$L_{o(t_n) q_i} = \Pr[o(t_n) \mid q(t_n) = q_i]. \tag{2}$$

A realisation of an HMM yields a sequence of hidden states $Q_{1:N_T} = \left\{q(t_1), q(t_2), \ldots, q(t_{N_T})\right\}$ and a sequence of observations $O_{1:N_T} = \left\{o(t_1), o(t_2), \ldots, o(t_{N_T})\right\}$. The total probability of a particular $Q_{1:N_T}$ given the observations $O_{1:N_T}$ can be written as

$$\Pr\left(Q_{1:N_T} \mid O_{1:N_T}\right) = \Pi_{q(t_1)} L_{o(t_1) q(t_1)} \prod_{n=2}^{N_T} A_{q(t_n) q(t_{n-1})} L_{o(t_n) q(t_n)}, \tag{3}$$

where $\Pi_{q(t_1)}$ is the probability that the state is initalised in the state $q(t_1)$.

2.2 Timing pulsars with hidden Markov models

To apply the HMM framework described in the Section 2.1 to pulsar timing, we first identify the set of hidden states $\{q_i\}$ with a discrete grid of $(f, \dot{f})$ pairs which encode the instantaneous spin frequency and its first time derivative. The grid is uniformly spaced in both directions, with η_f and $\eta_{\dot{f}}$ denoting the space between adjacent grid points in the f and $\dot{f}$ directions respectively.

We also stipulate that measurements of the system occur at each ToA measurement, and the observational datum at each timestep is the time elapsed since the last ToA measurement, i.e. $o(t_n) = t_n - t_{n-1} \equiv x_n$. The values of f and $\dot{f}$ are combined with fixed secular values $f_{\rm LS}$ and $\dot{f}_{\rm LS}$ measured at a reference epoch T_0 to give the full phase evolution

$$\Phi(t_n; f, f_{\rm LS}, \dot{f}, \dot{f}_{\rm LS}) = \left[f + f_{\rm LS} + \dot{f}_{\rm LS}(t_n - T_0)\right] x_n - \frac{1}{2}\left(\dot{f} + \dot{f}_{\rm LS}\right) x_n^2. \tag{4}$$

The minus sign in the second term arises because we use a backwards Taylor expansion. A secular second frequency derivative $\ddot{f}_{\rm LS}$ can also be included.

The emission probability $L_{x_n q_i}$ is calculated based on the fact that the number of cycles elapsed during each ToA gap should be an integer. This is implemented mathematically using a von Mises distribution, the precise mathematical form is given in equation 3 of Dunn et al. (2022). The two main contributions to the variance of $\Phi(t_n; \ldots)$ are the measurement uncertainty on the ToAs, and the phase uncertainty due to the discrete gridding in f and $\dot{f}$.

The evolution of the pulsar's hidden state during the gap between consecutive ToAs in the absence of a glitch is not assumed to be deterministic: we allow for some wandering in f and $\dot{f}$ due to timing noise. The default model assumes that he timing noise is driven by a white-noise torque derivative. An explicit expression for $A_{q_j q_i}$ using this model is given by equations 10–13 and B7–B11 of Melatos et al. (2020), which also includes further discussion of this timing noise model in Section 3.4. We also have the option to construct a model with a glitch included during a particular ToA gap. If the model includes a glitch during a particular ToA gap, the transition matrix for that gap is modified so that the pulsar transitions with equal probability from the state $(f, \dot{f})$ to all available states $(f', \dot{f}')$ satisfying $f' > f + \dot{f} x_n$.

2.3 Computations

In this section we briefly describe the most common computations carried out using the HMM pulsar timing framework. A general overview and description of common algorithms using HMMs can be found in Rabiner (1989), and further detailed descriptions of the algorithms used in this package can be found in Appendix A of Melatos et al. (2020).

Broadly speaking, there are two kinds of questions that `glitch_hmm` can be used to tackle: are there any glitches in a stretch of pulsar timing data, and what is the evolution of f and $\dot{f}$ over that stretch? These two questions are tackled through two closely related algorithms, the forward and forward-backward algorithms respectively.

2.3.1 Glitch detection

The question of glitch detection is framed as a question of Bayesian model selection. Let M_0 denote the model which includes no glitches, and $M_1(k)$ denote the model which includes a glitch during the kth ToA gap. For a given HMM M, the forward algorithm computes the model evidence $\Pr\left[O_{1:N_T} \mid M\right]$ [see

Appendix A.1 of Melatos et al. (2020)]. Then we can compute the Bayes factors

$$K_1(k) = \frac{\Pr\left[O_{1:N_T} \mid M_1(k)\right]}{\Pr\left[O_{1:N_T} \mid M_0\right]}. \tag{5}$$

Glitch candidates occur when $K_1(k)$ exceeds a predetermined threshold K_{th}, by default set at $10^{1/2}$ to achieve a false alarm rate of approximately 1% (Melatos et al., 2020). A greedy procedure may be used to account for the possibility of multiple glitches in a stretch of data, as described in Section 4.2 of Melatos et al. (2020).

2.3.2 Hidden state evolution

The evolution of f and $\dot{f}$ is analysed through the hidden state posterior distribution $\gamma_{q_i}(t_n) = \Pr\left[q(t_n) = q_i \mid O_{1:N_T}\right]$, computed using the forward-backward algorithm [see Appendix A2 of Melatos et al. (2020)]. The full posterior distribution is available as an analysis output, but it is typically more convenient to consider the posterior distributions on f and $\dot{f}$ alone (marginalising over the other), denoted by $\gamma_f(t_n)$ and $\gamma_{\dot{f}}(t_n)$ respectively. A point estimate of the evolution of f, $\hat{f}(t_n)$, can then be obtained by simply taking the maximum of $\gamma_f(t_n)$ at each timestep. An analogous computation provides $\hat{\dot{f}}(t_n)$.

3 Choosing parameters

Putting `glitch_hmm` into practice requires the user to choose a number of parameters on a per-analysis basis. This section describes some considerations and rules of thumb to help guide the user.

3.1 Domain of interest

The domain of interest (DOI) refers to the set of allowed f and $\dot{f}$ states in the HMM. Recall that the hidden states specify deviations away from a secular trend (see Section 2.2), so that $f = \dot{f} = 0$ indicates that the pulsar is precisely following the phase evolution specified by f_{LS}, $\dot{f}_{\text{LS}}$, and T_0. The DOI is always set up as a uniform grid, and the boundaries and grid spacing are both important parameters.

Both the f and $\dot{f}$ intervals covered by the DOI (denoted $[f_-, f_+]$ and $[\dot{f}_-, \dot{f}_+]$ respectively) must be chosen to accommodate both wandering due to timing noise and any possible glitches. Providing a universal prescription for choosing the DOI boundaries is not feasible, but Dunn et al. (2022) found that for a broad range of pulsars a frequency range of $[f_-, f_+] = [-3 \times 10^{-7}\,\text{Hz}, 3 \times 10^{-7}\,\text{Hz}]$ and a frequency derivative range of $[0.1\dot{f}_{\text{LS}}, -0.1\dot{f}_{\text{LS}}]$ was appropriate. In the case where a glitch is in fact present in the data it may be necessary to increase f_+ to accomodate the frequency jump.

The DOI is discretised with uniform spacing in f and $\dot{f}$, with spacings denoted η_f and $\eta_{\dot{f}}$ respectively. The choice of $\eta_{\dot{f}}$ is connected to the level of timing noise in the pulsar – see Section 3.2. At this stage a reliable way of measuring the timing noise and turning this into a value for $\eta_{\dot{f}}$ is not at hand, and a more ad hoc method is unfortunately needed. For example, Dunn et al. (2022) chose

$$\eta_{\dot{f}} = \frac{\dot{f}_+ - \dot{f}_-}{11}, \tag{6}$$

in combination with the prescription for $\dot{f}_{+,-}$ noted in the previous paragraph. Hence the DOI was always discretised into 11 $\dot{f}$ bins, with the bin width increasing as the magnitude of $\dot{f}_{\text{LS}}$ increased. A rule of thumb for choosing η_f depends in turn on the value of $\eta_{\dot{f}}$: it is a good idea for η_f to be chosen so that the typical change in frequency due to a non-zero $\dot{f}$ is larger than η_f, i.e.

$$\eta_f \approx 2\eta_{\dot{f}} \langle x_n \rangle \tag{7}$$

where $\langle x_n \rangle$ is the mean ToA gap in the dataset. This assumes that $\eta_{\dot{f}}$ is the smallest possible non-zero value of $\dot{f}$ in the DOI, which will typically be the case.

3.2 Timing noise

By default, `glitch_hmm` assumes that timing noise in the pulsar is driven by a white noise term in the second frequency derivative, viz.

$$\frac{\mathrm{d}^2 f}{\mathrm{d}t^2} = \xi(t), \quad \langle \xi(t)\xi(t') \rangle = \sigma^2 \delta(t - t'). \tag{8}$$

This model is not intended to be a universal choice, but it has been tested on both synthetic data which includes timing noise of a manifestly different form (white noise term in frequency derivative, see Melatos et al. 2020) and real data sets from a variety of pulsars (Lower et al., 2021; Dunn et al., 2022) (though note in the former reference it was found that modifying the timing noise model to include white noise in the frequency derivative instead was appropriate for a few pulsars).

The magnitude of the timing noise included in the HMM is controlled by the σ parameter in (8), which has units of $\mathrm{Hz\,s^{-3/2}}$. By default, its value depends on the value of $\eta_{\dot{f}}$ (see 3.1) as

$$\sigma = \eta_{\dot{f}} \langle x_n \rangle^{-1/2} \tag{9}$$

where $\langle x_n \rangle$ is the mean ToA gap in the dataset. The rationale behind this choice is described in 6.1 of Melatos et al. (2020). This value can be overridden if desired – see Listing 1.

4 Using `hmm_wrapper`

For the most part, analyses should be run through the scripts in `hmm_wrapper/`, principally `run_hmm.py`. These scripts depend on the `pulsar_hmm` package included in this repository — you can install that package by running `pip install .` inside the `pulsar_hmm/` folder.

`run_hmm.py` takes three command-line inputs: `--par`, which specifies the `.par` file to be used, `--tim`, which specifies the `.tim` file to be used, and `--ini`, which specifies the `.ini` file to be used. The `.par` and `.tim` files should be TEMPO2-style files. An example `.ini` file is shown in Listing 1 below.

```
# This section specifies the "domain of interest" (DOI), the allowed range of frequency and frequency
# derivative, and the spacing between grid points for each.
[doi]
# Minimum allowed frequency in Hz
freq_min = -3e-7
# Maximum allowed frequency in Hz
freq_max = 3e-6
# Spacing between frequency grid points in Hz
dfreq = 1e-8
# Minimum allowed frequency derivative in  Hz/s
fdot_min = -1e-15
```

```
# Maximum allowed frequency derivative in  Hz/s
fdot_max = 1e-15
# Spacing between frequency derivative grid points in Hz/s
dfdot = 1e-16

# This section contains any parameters related to the inclusion of timing noise in the HMM
[tn]
# Override default value of sigma specified in equation 8 (commented out for the purposes of the demo)
#sigma = 1e-20

# This section specifies any restrictions on the ToAs to be used
[toas]
# ToAs before mjd_min will not be used
mjd_min = 58514
# ToAs after mjd_max will not be used
mjd_max = 58700
# ToAs separated by less than min_toa_gap days will be discarded
min_toa_gap = 1

# This section specifies where to find the relevant MATLAB code. Paths must be absolute.
[matlab]
# The path to the core MATLAB implementation (matlab_code_logs/ in the repository)
matlab_path = /fred/oz022/ldunn_glitch/glitch_hmm/matlab_code_logs_stable
# The path to the appropriate MATLAB wrapper script (usually hmm_wrapper/do_matlab_analysis.m)
matlab_wrapper = /fred/oz022/ldunn_glitch/glitch_hmm/hmm_wrapper/do_matlab_analysis.m

# This section specifies the location of the analysis output. Paths must be absolute.
[out]
# All output files will begin with this prefix
out_prefix = /fred/oz022/ldunn_glitch/glitch_hmm/hmm_wrapper/demo/results/J1452-6036_
```

Listing 1: Example `.ini` file

There is an example analysis waiting to be run in `hmm_wrapper/demo/`. It uses data from PSR J1452 – 6036 released by the UTMOST collaboration in their first public data release (Lower et al., 2020), the entirety of which can be found here. To run this demo, first edit `demo.ini` in that directory so that the variables `matlab_path`, `matlab_wrapper`, `working_prefix`, and `out_prefix` are appropriately set (noting that all paths must be absolute, not relative). Also, make sure the directories referred to in `working_prefix` and `out_prefix` exist. Then, from the `hmm_wrapper/demo/` directory, run the analysis:

```
$ python ../run_hmm.py --par J1452-6036.par --tim J1452-6036.tim --ini demo.ini
```

When the analysis has finished, several files will appear in the directory specified by `out_prefix`. The next section discusses these analysis results in detail.

5 Output

The analysis run by `run_hmm.py` (see Section 4) produces quite a few output files, which are described in this section. In the following, we will refer to output files by their name *excluding* `out_prefix`.

5.1 `res.dat`

This file contains the essential details of any glitch candidates which were found in the analysis. Each row corresponds to a glitch candidate, in descending order of significance. The first column gives the index of the ToA gap containing the glitch candidate, and the second column gives the (natural) log Bayes factor of the glitch candidate.

5.2 `{f, fdot}_path.dat`

These files contain the maximum *a posteriori* values of f and $\dot{f}$ during each ToA gap, calculated using the model which includes all glitch candidates above the threshold. Plots of these paths are also produced as `{f, fdot}_path.pdf`.

5.3 `{f, fdot}_posterior.dat`

These files contain the natural logarithms of the marginalised f and $\dot{f}$ posteriors during each ToA gap. Each row corresponds to a frequency (derivative) bin, in ascending order. Plots of these posteriors are also produced as `{f, fdot}_posterior.pdf`.

5.4 `bfs.dat`

This file contains the log Bayes factors computed in the course of the model selection procedure described in Section 4 of Melatos et al. (2020). The procedure is a greedy one, so each iteration corresponds to calculation of new Bayes factors comparing models with all of the glitches of the most-preferred model from the previous iteration, plus one new glitch. Plots showing the Bayes factors at each iteration are also produced, named `bfs_i.pdf` where `i` is the iteration.

5.5 `analysis.mat`

Thie file contains a MATLAB workspace with the results of the analysis. This workspace contains all of the above data products, plus a few others which are not dumped by default. The following is a list of important variables found in that workspace, along with a brief description:

- `max_BFs` — contains the maximum log Bayes factors obtained in each iteration of the greedy glitch detection algorithm.
- `all_BFs` – contains all log Bayes factors calculated in each iteration of the greedy glitch detection algorithm.
- `fdots` — contains the frequency derivatives in the domain of interest (DOI).
- `freqs` — contains the frequencies in the DOI.
- `full_posterior` — contains the unmarginalised posterior on frequency and frequency derivative for the model selected by the glitch detection procedure.
- `fdot_posterior` — contains `full_posterior` after marginalistaion over frequency.
- `freq_posterior` — contains `full_posterior` after marginalistaion over frequency derivative.
- `glitches` — contains the indexes of the ToA gaps during which a glitch was detected.
- `kappa_per_toa` — contains κ for each ToA gap, defined as the reciprocal of the squared phase uncertainty across each ToA gap (measured in radians).

- `path` — contains the maximum *a posteriori* path in frequency derivative (first column) and frequency (second column), calculated using the model returned from the glitch detection algorithm.
- `residuals` — contains the phase residuals of `path`, measured in cycles.

5.6 Plots

In addition to the ASCII and `.mat` files described in the rest of this section, `glitch_hmm` also generates a number of plots in `.pdf` format. In this subsection we describe those plots and show example plots generated from the PSR J1452−6036 analysis in `hmm_wrapper/demo/`. We emphasise that these plots are generated for convenience, and are not necessarily intended to be publication-quality – all the data from which they are derived are available in the files described in the rest of this section.

5.6.1 `bfs_i.pdf`

These plots show the log Bayes factors generated by successive iterations of the greedy model selection procedure used to detect glitches (see Section 2.3.1). `bfs_0.pdf` shows the first iteration, `bfs_1.pdf` the second, and so on. The plots generated for the PSR J1452−6036 example analysis are shown in Fig. 1.

5.6.2 `{f, fdot}_path.pdf`

These plots show the sequence of most likely hidden f and $\dot{f}$ states at each timestep (see Section 2.3.2). The plots generated for the PSR J1452−6036 example analysis are shown in Fig. 2.

5.6.3 `{f, fdot}_posterior.pdf`

These plots show heatmaps of the marginalised f and $\dot{f}$ posterior probability densities, denoted γ_f and $\gamma_{\dot{f}}$ respectively (see Section 2.3.2). We plot the heatmap of $\ln[\gamma_f]$ rather than γ_f directly because γ_f is frequently very strongly peaked, and heatmaps of γ_f thus do not display well.

References

Dunn L., et al., 2022, MNRAS,

Lower M. E., et al., 2020, MNRAS, 494, 228

Lower M. E., et al., 2021, MNRAS, 508, 3251

Melatos A., Dunn L. M., Suvorova S., Moran W., Evans R. J., 2020, ApJ, 896, 78

Rabiner L. R., 1989, Proceedings of the IEEE, 77, 257

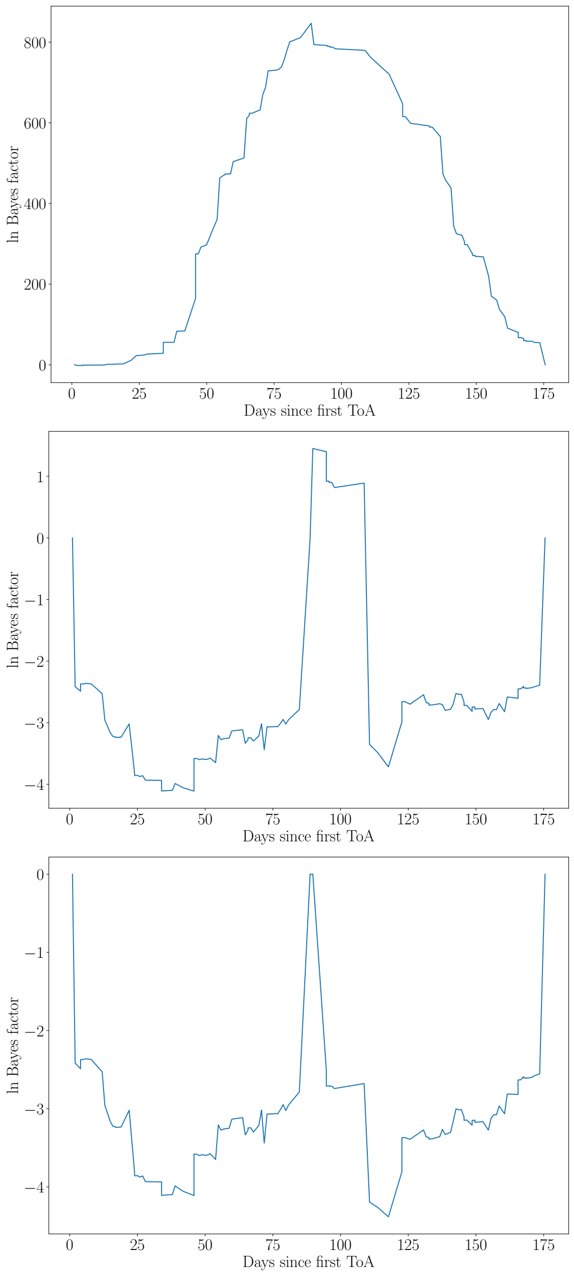

Figure 1: Example `bfs_i.pdf` plots from the demo PSR J1452−6036 analysis. `bfs_0.pdf` is shown in the top panel, `bfs_1.pdf` in the middle, and `bfs_2.pdf` (the terminating step) in the bottom panel.

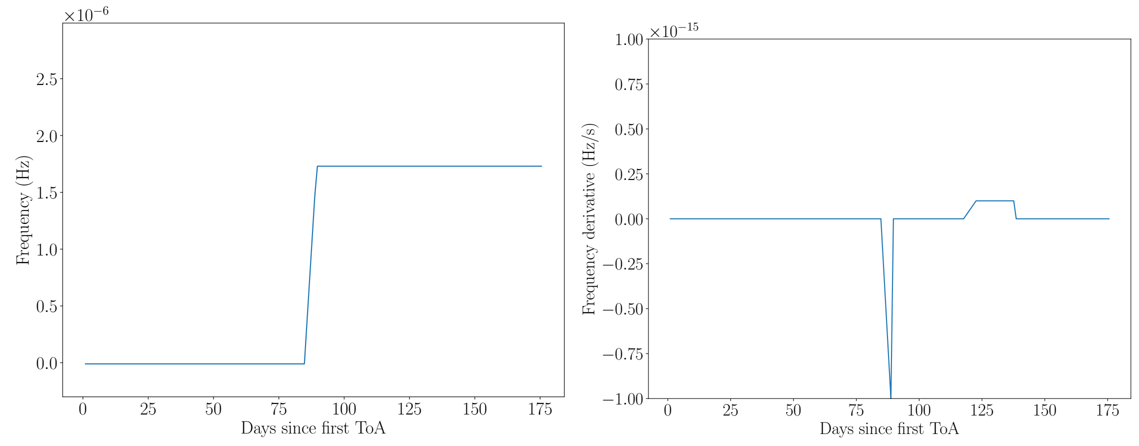

Figure 2: Pointwise most likely paths $\hat{f}(t_n)$ *(left)* and $\hat{\dot{f}}(t_n)$ *(right)* from the demo PSR J1452−6036 analysis.

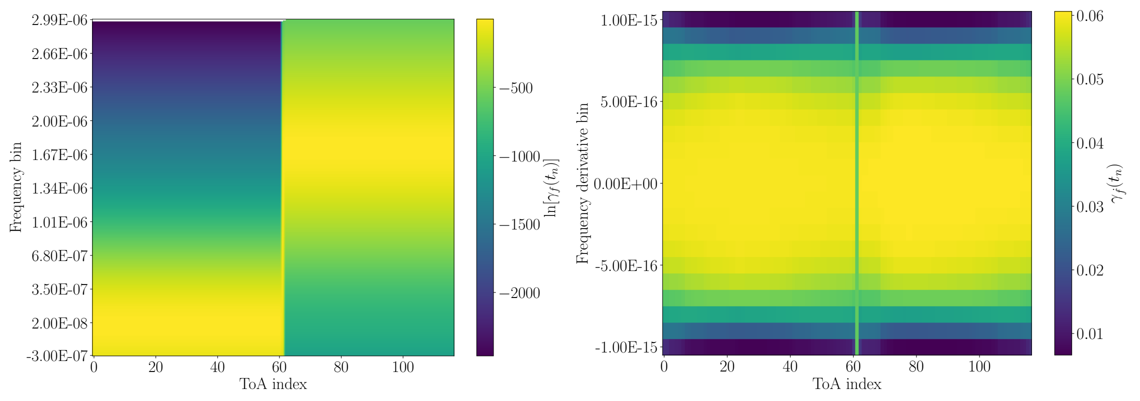

Figure 3: Heatmaps of $\ln[\gamma_f(t_n)]$ *(left)* and $\gamma_{\dot{f}}(t_n)$ *(right)* from the demo PSR J1452−6036 analysis.