Usage guide for the pulsar_freq_filter package

Contents

1 Introduction

This package can be used to estimate parameters in the two-component crust-superfluid model of neutron stars [1], [2], [3] by analysing pulsar timing data. It applies a Kalman filter to pulsar angular frequency data to get a probability distribution for the two-component model parameters given the data and then uses the `dynesty` [4] nested sampler through the `bilby` [5] front-end to sample this probability distribution and plot it. This package can work with real data or do simulations. The methodology is based on work done in [2], [3]. It was used to carry out the results in (**newpaper**).

A basic outline of the instructions for the package is as follows

- Download `pulsar_freq_filter` package.
- Install the required python modules listed in `requirements.txt`.
- Download UTMOST pulsar data from https://github.com/Molonglo/TimingDataRelease1/ [6].

To generate the real data results from (**newpaper**),

- run `real_data_programs/make_freq_files/test_script_01.sh` to generate a pulsar rotation frequency time series.
- run `real_data_programs/real_data_run/slurm_test_02.sh` to do parameter estimation with the two-component model on the frequency data.
- run `real_data_programs/real_data_run/slurm_test_01.sh` to do parameter estimation with the one-component model on the frequency data.

To carry out simulations,

- Go to the directory for that simulation (either `freq_programs/test01_R_noise_size_test/`, `freq_programs/test02_R_noise_vary_test/`, `toa_programs/test01_random_toas/` or `toa_programs/test02_cadence_toas/`).
- Run the simulation scripts in that directory eg. `slurm_test_01.sh`.
- In that directory, run `in_out_plots/analyse_modes_ll_noload.sh` and `pp_plots/make_pp_plots.sh` to get summary plots of the results.

If you only want to run the Kalman filter-parameter estimation method on real pulsar data to produce the results seen in the paper then only section 2 of this guide is needed. If you also want to carry out the simulations in the appendices, these are explained in sections 3, 4 and 5. Additional instructions are included in the `note.txt` files contained in each major directory of the package.

2 Real data programs (`real_data_programs` directory)

This section explains how to run the parameter estimation programs on real pulsar data. The raw data used in the paper comes from the UTMOST observing program [6] and is publicly available at https://github.com/Molonglo/TimingDataRelease1/. This data should be downloaded and the parfile and timfile for the pulsar of interest (in our case PSR J1359-6038) should be saved in a directory `pulsar_freq_filter/J1359-6038`.

The raw data comes in the form of TOAs and must be converted into a time series of pulsar rotation frequencies first. Once the TOAs are converted to frequencies the Kalman filter method can be run on them using the programs in the `real_data_run` directory. This will produce corner plots of the posterior distributions for the parameters of the two-component and one-component models.

The basic operation of these programs is as follows

1. `make_freq_files/test_script_01.sh` calls `fit_toas_nooverlap.py` with the necessary inputs. eg. the parfile and timfile.
2. `fit_toas_nooverlap.py` converts the TOAs to frequencies using TEMPO2.
3. Then run `real_data_run/slurm_test_01.sh`, which calls `run_on_real_data_onecomp.py` with the necessary input arguments (`slurm_test_02.sh` will call `run_on_real_data_twocomp.py` and do a similar thing).
4. `run_on_real_data_onecomp.py` then loads the frequency data.
5. A `KalmanLikelihood` object is created from the frequency data (using `models.py` and `sample.py`) with a loglikelihood function, `loglike()`, that can be called (using `kalman.py`).
6. The loglikelihood function is sampled using random choices of parameters with the `dynesty` sampler [4].
7. The sampler results are made into a corner plot.

2.1 `make_freq_files` directory

The programs in the `make_freq_files` directory will generate a time series of pulsar frequencies from real pulsar TOAs by fitting frequencies to TOAs using TEMPO2.

To do this, enter the commands:

```
cd pulsar_freq_filter/real_data_programs/make_freq_files
bash test_script_01.sh
```

The script `test_script_01.sh` calls the program `fit_toas_nooverlap.py` with some preset arguments that can be adjusted. The inputs to `fit_toas_nooverlap.py` are

- `Tfit` - The set of TOAs used in a frequency fit must span a time less than `Tfit`, unless there aren't enough TOAs in that time span. It has units of days. The default is 10 days.
- `Nfit_min` - The minimum number of TOAs used in a frequency fit. The default is 3.
- `parfile` - The file containing the pulsar's parameters.
- `input_timfile` - The file containing the pulsar's TOAs and their errors.
- `out_directory` - The name of the directory the results of the simulations are saved to.

- `tag` - A tag that goes in front of the file names.
- `threshold` - Remove frequency data points with residuals larger than threshold because they are very bad fits. It has units of $\mathrm{rad\ s^{-1}}$. The default is $10^{-7}\mathrm{rad\ s^{-1}}$

The program will create a directory for the output (in this example it is `outdir01_J1359-6038_10_3`). The files produced in the output directory are shown below.

```
(new_env) [noneill@tooarrana1 outdir01_J1359-6038_10_3]$ ls
storetemp2                                  test_10_3_nooverlap_freqs_1e-07cut_resids.png
test_10_3_nooverlap_freqs_1e-07cut.freq     test_10_3_nooverlap_freqs_uncut.freq
test_10_3_nooverlap_freqs_1e-07cut.png      test_10_3_nooverlap_freqs_uncut.png
```

- `storetemp2`
  - Contains the timfiles used for the fitting of every frequency data point.
- `test_10_3_nooverlap_freqs_1e-07cut.freq`
  - Text file containing the sequence of pulsar frequencies, measurement times, and measurement uncertainties. Data points with residuals with magnitude greater than $10^{-7}\mathrm{rad\ s^{-1}}$ have been cut.
- `test_10_3_nooverlap_freqs_1e-07cut.png`
  - Plot of frequency data with residuals greater than $10^{-7}\ \mathrm{rad\ s^{-1}}$ cut.
- `test_10_3_nooverlap_freqs_1e-07cut_resids.png`
  - Plot of frequency data with residuals greater than $10^{-7}\ \mathrm{rad\ s^{-1}}$ cut. This plot only shows the residuals.
- `test_10_3_nooverlap_freqs_uncut.freq`
  - Text file containing the sequence of pulsar frequencies, measurement times, and measurement uncertainties with no outliers cut.
- `test_10_3_nooverlap_freqs_uncut.png`
  - Plot of frequency data with no outliers cut.

The most important plot produced here is `test_10_3_nooverlap_freqs_1e-07cut_resids.png` which should look like this

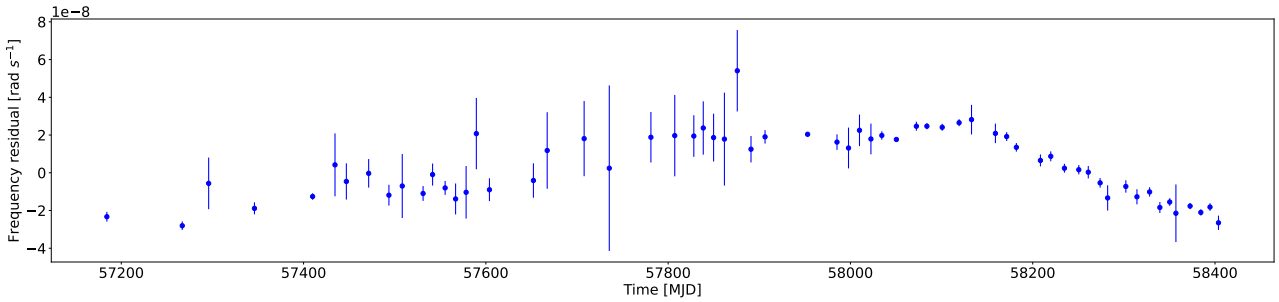


Figure 1: `test_10_3_nooverlap_freqs_1e-07cut_resids.png`. Residuals of frequencies fitted to TOAs from PSR J1359−6038 with outliers greater than `threshold` $= 10^{-7}\ \mathrm{rad\ s^{-1}}$ removed. Figure 1 of the paper.

2.2 real_data_run

Once the frequency data has been made in make_freq_files, the Kalman filter-parameter estimation program can be run on the frequency data. This directory contains programs for parameter estimation using the one-component and two-component models. The script slurm_test_01.sh runs the one-component model and slurm_test_02.sh runs the two-component model.

To do this, run the commands

```
cd pulsar_freq_filter/real_data_programs/real_data_run
mkdir outdir01
sbatch test_script_01.sh
mkdir outdir02
sbatch test_script_02.sh
```

The script slurm_test_01.sh calls run_on_real_data_onecomp_bins.py with some preset arguments which can be adjusted. (Note that a slurm script is not essential, it is just for convenience. An ordinary bash script would do). The inputs to run_on_real_data_onecomp_bins.py are

- freqfile - The file containing the fitted pulsar frequency time series.
- Nwalks - The number of random walks the dynesty sampler uses to sample the posterior. The default is 100.
- Npoints - The number of live points the dynesty sampler uses to sample the posterior. The default is 10000.
- resume_run - If the program is interrupted during sampling then the progress is saved. Adding the resume_run flag in the call of run_on_real_data_onecomp_bins.py and running it again will reload the saved progress files and continue the simulation. Otherwise the program is restarted from scratch.
- out_directory - The name of the directory the results of the simulations are saved to. The default is outdir01.
- tag - A tag that goes in front of the file names. The default is test.

The results of this test with the one-component model will be in the directory outdir01, which contains the directory outdir01_J1359-6038_10_3, the contents of which are listed below

```
(new_env) [noneill@tooarrana1 outdir01_J1359-6038_10_3]$ ls
test_tempo_corner_50bins.png  test_tempo_result.json
test_tempo_dynesty.pickle     test_tempo_resume.pickle
```

The main result is the plot test_tempo_corner_50bins.png which looks like this

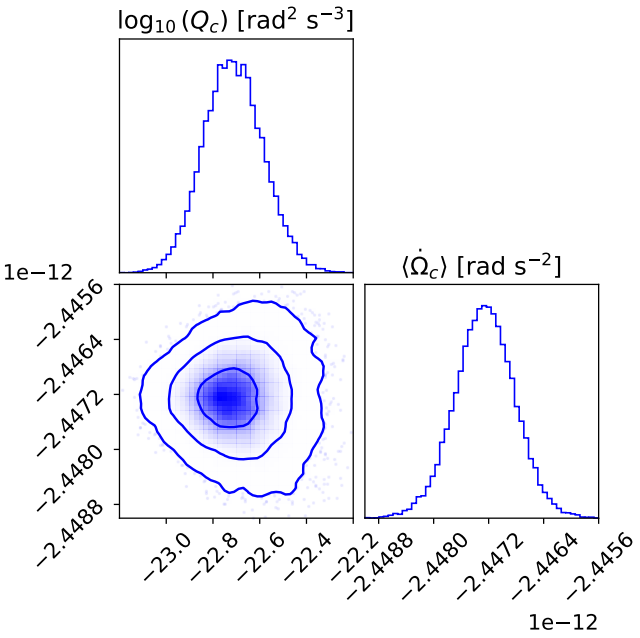


Figure 2: `test_tempo_corner_50bins.png`. Posterior distribution for the one-component model for PSR J1359−6038. Figure H1 of the paper.

The script `slurm_test_02.sh` calls `run_on_real_data_twocomp_bins.py` with the same preset input parameters as the one-component model program (except `out_directory` which is `outdir02` by default). The outputs for this will appear in the directory `outdir02`, which contains the directory `outdir02_J1359-6038_10_3`, the contents of which are listed below

```
(new_env) [noneill@tooarrana1 outdir02_J1359-6038_10_3]$ ls
hist_tempo_4params_filled_test.png  test_tempo_dynesty.pickle   test_tempo_result.json
test_tempo_corner_50bins.png        test_tempo_param_hists.png  test_tempo_resume.pickle
```

The main outputs of interest are the plots `hist_tempo_4params_filled_test.png` and `test_tempo_corner_50bins.png` which should look like this:

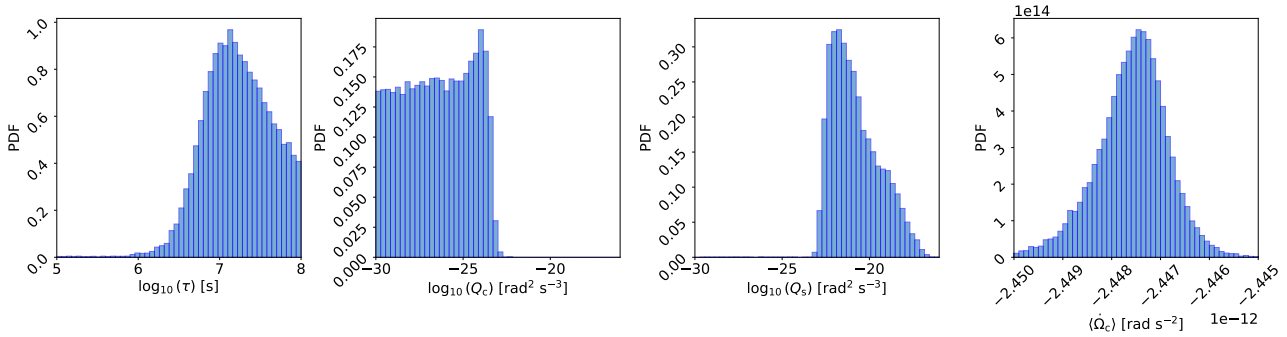


Figure 3: `hist_tempo_4params_filled_test.png`. Posterior distributions for the two-component model for PSR J1359−6038. Figure 2 of the paper.

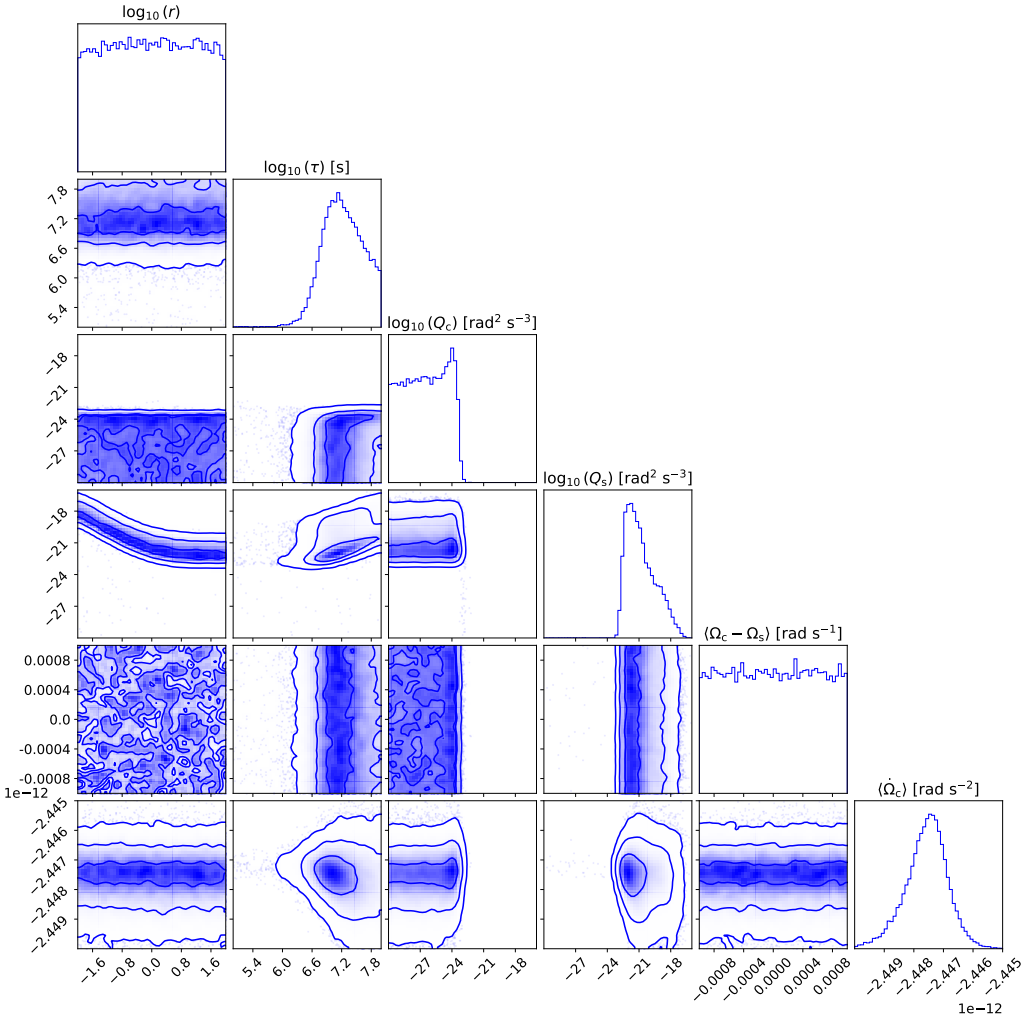


Figure 4: `test_tempo_corner_50bins.png`. Posterior distribution for the two-component model for PSR J1359−6038. Figure G1 of the paper.

3 Pulsar frequency simulation programs (freq_programs directory)

Simulations on pulsar rotation frequencies can be done in this directory. The programs in this directory will simulate pulsar frequency data with a random choice of the two-component model parameters and then run the parameter estimation algorithm on the data to see how well the simulation parameters match the recovered parameters.

The basic operation of these programs is as follows

1. In test01_R_noise_size_test, slurm_test_01.sh calls ns_sim_random_vary_EM_only.py with the necessary input arguments (In test02_R_noise_vary_test, slurm_test_01.sh calls ns_sim_random_vary_EM_only_EFAC.py, which does a similar thing).
2. ns_sim_random_vary_EM_only.py randomly chooses model parameters for the simulated pulsar.
3. The two_component_fake_data function (from fake_data.py) is then used to simulate pulsar frequencies.
4. A KalmanLikelihood object is created from the frequency data (using models.py and sample.py) with a log-likelihood function, loglike(), that can be evaluated for any choice of model parameters (using kalman.py).
5. The loglikelihood function is sampled using random choices of parameters with the dynesty sampler [4].
6. The sampler results are made into a corner plot.
7. Summary plots are generated using in_out_plots and pp_plots.

3.1 Simulate recovery with different levels of measurement noise (freq_programs/test01_R_noise_size_test directory)

This section tests the parameter estimation method's abilities as is done in appendices D1 and D2 of the paper. The four slurm scripts already in this directory (slurm_test_01.sh, slurm_test_02.sh, slurm_test_03.sh and slurm_test_04.sh) will run all the simulations seen in those appendices. These programs simulate pulsar frequency data with a random choice of two-component model parameters and then apply the Kalman filter parameter estimation method to try to recover the original parameters. Each of the four slurm scripts simulates data with a different level of measurement noise to see how it affects the quality of parameter recovery.

To produce the results from the paper, run the commands

```
cd pulsar_freq_filter/freq_programs/test01_R_noise_size_test
mkdir outdir01
sbatch slurm_test_01.sh
```

The script slurm_test_01.sh calls programs/ns_sim_random_vary_EM_only.py with some preset arguments which can be adjusted. The inputs to ns_sim_random_vary_EM_only.py are

- Nobs - The number of frequency measurements simulated for the pulsar. The default is 1000.
- Tdays - The time span the frequency measurements are spread over. It has units of days. The default is 1000 days.
- R_in - The measurement error variance used to simulate the pulsar frequency measurements. It has units of $\mathrm{rad}^2\mathrm{s}^{-2}$.
- R_out - The measurement error variance the Kalman filter is told is present in the data. It has units of $\mathrm{rad}^2\mathrm{s}^{-2}$.
- Nwalks - The number of random walks the dynesty sampler uses to sample the posterior. The default is 10.
- Npoints - The number of live points the dynesty sampler uses to sample the posterior. The default is 100.
- resume_run - If the simulation is interrupted then the progress is saved. Adding the resume_run flag in the call of ns_sim_random_vary_EM_only.py and running it again will reload the saved progress files and continue the simulation. Otherwise the simulation is restarted from scratch.
- out_directory - The name of the directory the results of the simulations are saved to. The default is outdir01.
- tag - A tag that goes in front of the file names. The default is test.

The allowed runtime of the slurm script, the memory allocated to each simulation and the number of simulations (the default is 200) can also be adjusted in the `slurm_test_01.sh` script.

The outputs appear in the directory `outdir01`. Each of the 200 simulations has its own output directory `outdir01_i` inside `outdir01`. The contents of the output folder look like this

```
(new_env) [noneill@tooarrana1 outdir01_1]$ ls
dataplot.png                  test_1_noEFAC_result.json      test_1_simulation_parameters.json
test_1_corner_noEFAC.png      test_1_noEFAC_resume.pickle
test_1_noEFAC_dynesty.pickle  test_1_param_hists_noEFAC.png
```

The `in_out_plots` and `pp_plots` directories (See section 5) can be used to get an overview of all 200 simulations in a test. For example, after running `slurm_test_01.sh`, output directories `in_out_plots/outdir01` and `pp_plots/outdir01` can be produced containing plots of input parameters against output parameters and PP-plots respectively. `in_out_plots/outdir01` contains the following output

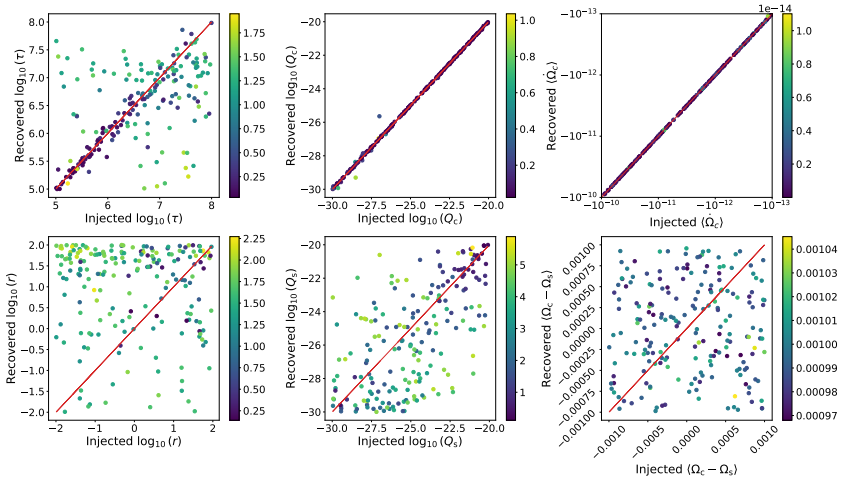
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_noEFAC.png  in_out_modes_ll_tau_Qc_colour_noEFAC.png
```

and `pp_plots/outdir01` contains the following output

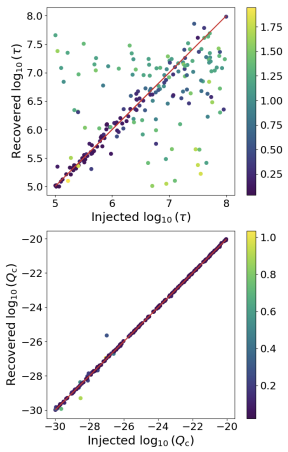
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
test_pp_plot_noEFAC.png
```

Examples are shown below

Figure 5



(a) `outdir01/in_out_modes_ll_colour_noEFAC.png`. Figure D1 in the paper.



(b) Result in `outdir01`. `in_out_modes_ll_tau_Qc_colour_noEFAC.png`

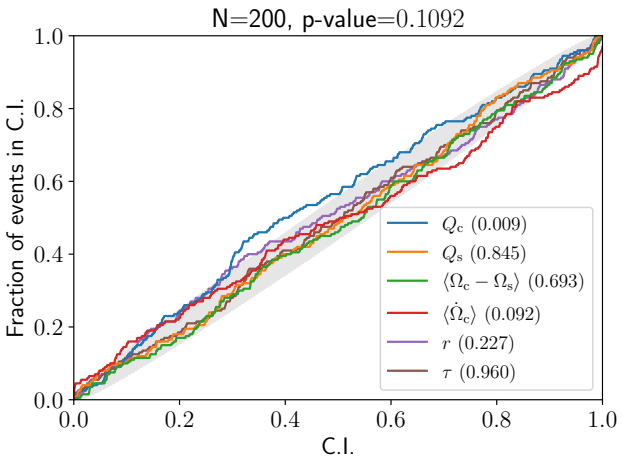
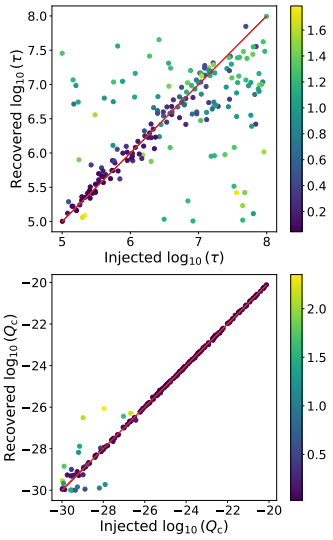
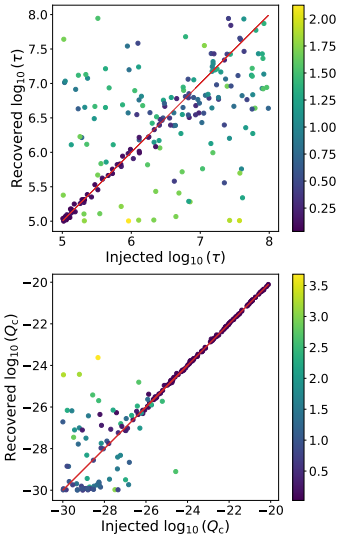


Figure 6: `outdir01/test_pp_plot_noEFAC.png`. Figure D2 in the paper.

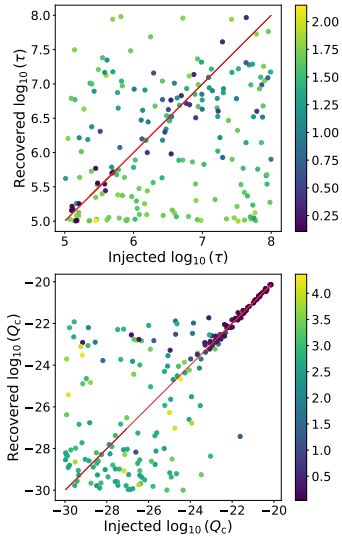
Figure 7



(a) Column 1 of figure D3 in the paper. Result in `outdir02.in_out_modes_tau_Qc_colour_noEFAC.png`.



(b) Column 2 of figure D3 in the paper. Result in `outdir03.in_out_modes_tau_Qc_colour_noEFAC.png`.



(c) Column 3 of figure D3 in the paper. Result in `outdir04.in_out_modes_tau_Qc_colour_noEFAC.png`.

3.2 Simulate recovery correcting for incorrect measurement noise (freq_programs/test02_R_noise_vary_test directory)

This section tests the parameter estimation method's abilities as is done in appendix D3 of the paper. The two slurm scripts already in this directory (slurm_test_01.sh and slurm_test_02.sh) will run all the simulations seen in that appendix. These programs simulate pulsar frequency data with a measurement error variance R_in and then feed an incorrect measurement error variance R_out into the Kalman filter for parameter estimation. The parameter estimation algorithm attempts to fix this by varying the measurement error in the Kalman filter using an EFAC parameter to try to correct for the incorrect R_out value. The parameter estimation algorithm is also run without the EFAC parameter as a control test. This produces two posteriors per simulation.

The script slurm_test_01.sh calls programs/ns_sim_random_vary_EM_only_EFAC.py with some preset arguments which can be adjusted. The inputs to ns_sim_random_vary_EM_only_EFAC.py are

- Nobs - The number of frequency measurements simulated for the pulsar. The default is 1000.
- Tdays - The time span the frequency measurements are spread over. It has units of days. The default is 1000 days.
- R_in - The measurement error variance used to simulate the pulsar frequency measurements. It has units of $\mathrm{rad}^2\mathrm{s}^{-2}$.
- R_out - The measurement error variance the Kalman filter is told is present in the data. It has units of $\mathrm{rad}^2\mathrm{s}^{-2}$.
- Nwalks - The number of random walks the dynesty sampler uses to sample the posterior. The default is 10.
- Npoints - The number of live points the dynesty sampler uses to sample the posterior. The default is 100.
- resume_run - If the simulation is interrupted then the progress is saved. Adding the resume_run flag in the call of ns_sim_random_vary_EM_only_EFAC.py and running it again will reload the saved progress files and continue the simulation. Otherwise the simulation is restarted from scratch.
- out_directory - The name of the directory the results of the simulations are saved to. The default is outdir01.
- tag - A tag that goes in front of the file names. The default is test.

The allowed runtime of the slurm script, the memory allocated to each simulation and the number of simulations can also be adjusted in the slurm_test_01.sh script.

The outputs appear in the directory outdir01. Each of the 200 simulations has its result in an output directory outdir01_i inside outdir01. The outputs for each simulation are

```
(new_env) [noneill@tooarrana1 outdir01_1]$ ls
dataplot.png                test_1_EFAC_result.json       test_1_noEFAC_resume.pickle
test_1_corner_EFAC.png      test_1_EFAC_resume.pickle     test_1_param_hists_EFAC.png
test_1_corner_noEFAC.png    test_1_noEFAC_dynesty.pickle  test_1_param_hists_noEFAC.png
test_1_EFAC_dynesty.pickle  test_1_noEFAC_result.json     test_1_simulation_parameters.json
```

The in_out_plots and pp_plots directories (See section 5) can be used to get an overview of all 200 simulations in a test. For example, after running slurm_test_01.sh, output directories in_out_plots/outdir01 and pp_plots/outdir01 can be produced containing plots of input parameters against output parameters and PP-plots respectively. in_out_plots/outdir01 contains the following output

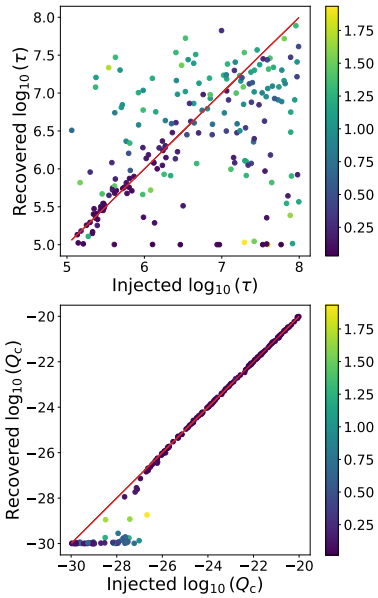
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_EFAC.png     in_out_modes_ll_tau_Qc_colour_EFAC.png
in_out_modes_ll_colour_noEFAC.png   in_out_modes_ll_tau_Qc_colour_noEFAC.png
```

and pp_plots/outdir01 contains the following output

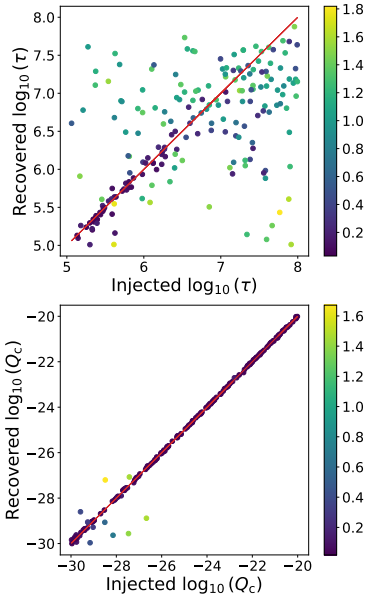
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
test_pp_plot_EFAC.png  test_pp_plot_noEFAC.png
```

The main results of interest are the results that appear in the paper which are shown below.

Figure 8

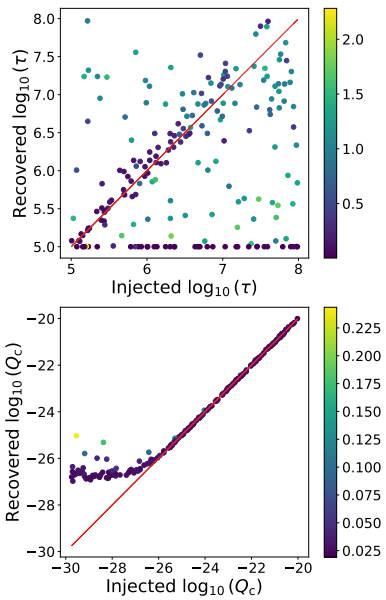


(a) `outdir01/in_out_modes_tau_Qc_colour_noEFAC.png`. Column 1 of Figure D4 of paper.

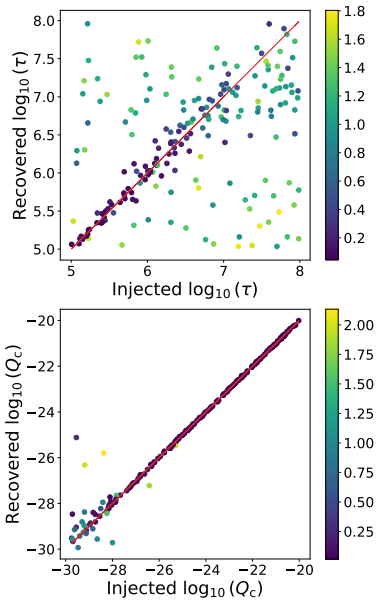


(b) `outdir01/in_out_modes_tau_Qc_colour_EFAC.png`. Column 2 of Figure D4 of paper.

Figure 9



(a) `outdir02/in_out_modes_tau_Qc_colour_noEFAC.png`. Column 1 of Figure D5 of paper.



(b) `outdir02/in_out_modes_tau_Qc_colour_EFAC.png`. Column 2 of Figure D5 of paper.

4 Pulsar TOA simulation programs (toa_programs directory)

Simulations on pulsar TOAs can be done in this directory. test01_random_toas contains the programs to run the parameter estimation method on simulated TOAs with random spacing and test02_cadence_toas contains the programs to run the parameter estimation method on simulated TOAs with the spacing of real pulsar data.

The basic operation of these programs is as follows

1. In test01_random_toas, slurm_test_01.sh calls ns_sim_random_vary_EFAC.py with the necessary input arguments (In test02_cadence_toas, slurm_test_01.sh calls ns_sim_cadence_vary_EFAC.py which does a similar thing).
2. ns_sim_random_vary_EFAC.py randomly chooses model parameters for the simulated pulsar.
3. The simulate_toas_random.py function is then used to simulate pulsar TOAs and frequencies.
4. Frequencies are fitted to the TOAs using the fit_toas_nooverlap and fit_toas_overlap functions.
5. A KalmanLikelihood object is created from the frequency data (using models.py and sample.py) with a log-likelihood function, loglike(), that can be evaluated for any choice of model parameters (using kalman.py).
6. The loglikelihood function is sampled using random choices of parameters with the dynesty sampler [4].
7. The sampler results are made into a corner plot.
8. Summary plots are generated using in_out_plots and pp_plots.

4.1 Programs for simulating TOAs with random spacing (toa_programs/test01_random_toas directory)

This section tests the parameter estimation method's ability to recover parameters on frequencies that have been fitted to TOAs using TEMPO2 as is done in section F of the paper. The programs in test01_random_toas/programs will simulate TOAs and frequencies for a pulsar using a random choice of two-component model parameters and then fit frequencies to the TOAs using TEMPO2. Three sets of frequency data are produced: frequencies fitted using overlapping sets of TOAs (not used in the paper), frequencies fitted using non-overlapping sets of TOAs and true frequencies generated while simulating the TOAs. Then the parameter estimation method is run on these three data sets twice, once with EFAC and EQUAD parameters and once without. So six sets of results are produced for each call of ns_sim_random_vary_EFAC.py.

The script slurm_test_01.sh runs these simulations by calling ns_sim_random_vary_EFAC.py with some preset arguments that can be adjusted. The inputs to ns_sim_random_vary_EFAC.py are

- Nobs - The number of TOAs simulated for the pulsar. The default is 1000.
- Tdays - The time span the TOA measurements are spread over. It has units of days. The default is 1000 days.
- T_error_in - The standard deviation of the measurement error to simulate TOAs with. It has units of seconds. The default is 10^{-16}s.
- Tfit - The set of TOAs used in a frequency fit must span a time less than Tfit, unless there aren't enough TOAs in that time span. It has units of days.
- Nfit_min - The minimum number of TOAs used in a frequency fit. The default is 3.
- Nwalks - The number of random walks the dynesty sampler uses to sample the posterior. The default is 10.
- Npoints - The number of live points the dynesty sampler uses to sample the posterior. The default is 100.
- resume_run - If the simulation is interrupted then the progress is saved. Adding the resume_run flag in the call of ns_sim_random_vary_EFAC.py and running it again will reload the saved progress files and continue the simulation. Otherwise the simulation is restarted from scratch.
- out_directory - The name of the directory the results of the simulations are saved to. The default is outdir01.
- tag - A tag that goes in front of the file names. The default is test.

The allowed runtime of the slurm script, the memory allocated to each simulation and the number of simulations (the default is 200) can also be adjusted in the `slurm_test_01.sh` script.

The outputs appear in the directory `outdir01`. Each of the 200 simulations has its own output directory `outdir01_i` inside `outdir01`. The contents of the output folder look like this

```
(new_env) [noneill@tooarrana1 outdir01_1]$ ls
dataplots.png                              test_1_overlap_control_resume.pickle
storetemp                                  test_1_overlap_corner.png
storetemp2                                 test_1_overlap_dynesty.pickle
test_1_freqs_and_toas_true.png             test_1_overlap.freq
test_1_freqs_nooverlap_uncut.png           test_1_overlap_result.json
test_1_freqs_overlap_uncut.png             test_1_overlap_resume.pickle
test_1_freqs_residuals_comparison.png      test_1_overlap_uncut.freq
test_1_nooverlap_control_corner.png        test_1.par
test_1_nooverlap_control_dynesty.pickle    test_1_simulation_parameters.json
test_1_nooverlap_control_result.json       test_1_true_control_corner.png
test_1_nooverlap_control_resume.pickle     test_1_true_control_dynesty.pickle
test_1_nooverlap_corner.png                test_1_true_control_result.json
test_1_nooverlap_dynesty.pickle            test_1_true_control_resume.pickle
test_1_nooverlap.freq                      test_1_true_corner.png
test_1_nooverlap_result.json               test_1_true_dynesty.pickle
test_1_nooverlap_resume.pickle             test_1_true.freq
test_1_nooverlap_uncut.freq                test_1_true_result.json
test_1_overlap_control_corner.png          test_1_true_resume.pickle
test_1_overlap_control_dynesty.pickle      test_1_true.tim
test_1_overlap_control_result.json
```

The `in_out_plots` and `pp_plots` directories (See section 5) can be used to get an overview of all 200 simulations in a test. After running `slurm_test_01.sh`, output directories `in_out_plots/outdir01` and `pp_plots/outdir01` can be produced containing plots of input parameters against output parameters and PP-plots respectively. `in_out_plots/outdir01` contains the following output

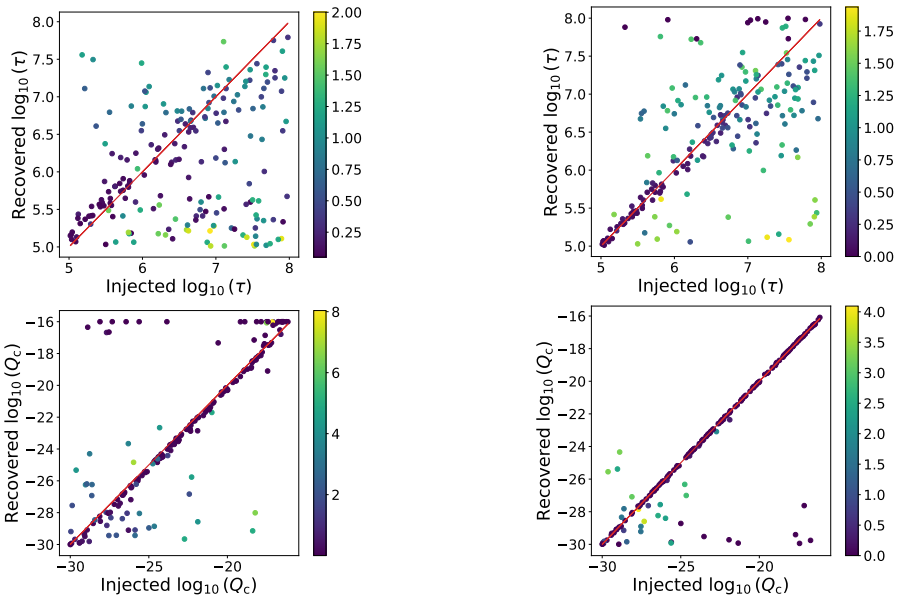
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_nooverlap_control.png  in_out_modes_ll_tau_Qc_colour_nooverlap_control.png
in_out_modes_ll_colour_nooverlap.png          in_out_modes_ll_tau_Qc_colour_nooverlap.png
in_out_modes_ll_colour_overlap_control.png    in_out_modes_ll_tau_Qc_colour_overlap_control.png
in_out_modes_ll_colour_overlap.png            in_out_modes_ll_tau_Qc_colour_overlap.png
in_out_modes_ll_colour_true_control.png       in_out_modes_ll_tau_Qc_colour_true_control.png
in_out_modes_ll_colour_true.png               in_out_modes_ll_tau_Qc_colour_true.png
```

and `pp_plots/outdir01` contains the following output

```
(new_env) [noneill@tooarrana1 outdir01]$ ls
test_pp_plot_nooverlap_control.png  test_pp_plot_overlap_control.png  test_pp_plot_true_control.png
test_pp_plot_nooverlap.png          test_pp_plot_overlap.png          test_pp_plot_true.png
```

The main outputs for this simulation are the plots of τ and Q_c that appeared in the paper which should look like this

Figure 10



(a) `in_out_modes_tau_Qc_colour_nooverlap.png`. Column 1 of Figure F1 of the paper.

(b) `in_out_modes_tau_Qc_colour_true.png`. Column 2 of Figure F1 of the paper.

4.2 Programs for simulating TOAs with the spacing of a real pulsar (`toa_programs/test02_cadence_toas` directory)

This section tests the parameter estimation method's ability to recover parameters on frequencies that have been fitted to TOAs using TEMPO2 as is done in section F of the paper. The programs in `test02_cadence_toas/programs` will simulate TOAs and frequencies at approximately the same times as for a real pulsar using the TOA measurement errors and the frequency and frequency derivatives provided with the real data and using a random choice of the other two-component model parameters. The programs then fit frequencies to the simulated TOAs using TEMPO2. Three sets of frequency data are produced: frequencies fitted using overlapping sets of TOAs, frequencies fitted using non-overlapping sets of TOAs and true frequencies generated during the simulation. Then the parameter estimation method is run on these three data sets twice, once with EFAC and EQUAD parameters and once without. So six sets of results are produced for each call of `ns_sim_cadence_vary_EFAC.py`.

`slurm_test_01.sh` runs these simulations by calling `ns_sim_cadence_vary_EFAC.py` with some preset parameters that can be adjusted. The inputs to `ns_sim_cadence_vary_EFAC.py` are

- `parfile` - The file containing the pulsar's parameters.
- `timfile` - The file containing the pulsar's TOAs and their errors.
- `Tfit` - The set of TOAs used in a frequency fit must span a time less than `Tfit`, unless there aren't enough TOAs in that time span. It has units of days. The default is 10 days.
- `Nfit_min` - The minimum number of TOAs used in a frequency fit. The default is 3.
- `threshold` - Remove frequency data points with residuals larger than threshold because they are very bad fits. It has units of $\mathrm{rad\ s^{-1}}$.
- `Nwalks` - The number of random walks the `dynesty` sampler uses to sample the posterior. The default is 10.
- `Npoints` - The number of live points the `dynesty` sampler uses to sample the posterior. The default is 100.
- `resume_run` - If the simulation is interrupted then the progress is saved. Adding the `resume_run` flag in the call of `ns_sim_cadence_vary_EFAC.py` and running it again will reload the saved progress files and continue the simulation. Otherwise the simulation is restarted from scratch.
- `out_directory` - The name of the directory the results of the simulations are saved to. The default is `outdir01`.
- `tag` - A tag that goes in front of the file names. The default is `test`.

The allowed runtime of the slurm script, the memory allocated to each simulation and the number of simulations (the default is 200) can also be adjusted in the `slurm_test_01.sh` script.

The outputs appear in the directory `outdir01`. Each of the 200 simulations has its own output directory `outdir01_i` inside `outdir01`. The contents of the output folder look like this

```
(new_env) [noneill@tooarrana1 outdir01_1]$ ls
dataplots.png                           test_1_overlap_control_resume.pickle
storetemp                               test_1_overlap_corner.png
storetemp2                              test_1_overlap_dynesty.pickle
test_1_freqs_and_toas_true.png          test_1_overlap.freq
test_1_freqs_nooverlap_uncut.png        test_1_overlap_result.json
test_1_freqs_overlap_uncut.png          test_1_overlap_resume.pickle
test_1_freqs_residuals_comparison.png   test_1_overlap_uncut.freq
test_1_nooverlap_control_corner.png     test_1.par
test_1_nooverlap_control_dynesty.pickle test_1_simulation_parameters.json
test_1_nooverlap_control_result.json    test_1_toa_cadence_adjustments_true.png
test_1_nooverlap_control_resume.pickle  test_1_true_control_corner.png
test_1_nooverlap_corner.png             test_1_true_control_dynesty.pickle
test_1_nooverlap_dynesty.pickle         test_1_true_control_result.json
test_1_nooverlap.freq                   test_1_true_control_resume.pickle
test_1_nooverlap_result.json            test_1_true_corner.png
test_1_nooverlap_resume.pickle          test_1_true_dynesty.pickle
test_1_nooverlap_uncut.freq             test_1_true.freq
test_1_overlap_control_corner.png       test_1_true_result.json
test_1_overlap_control_dynesty.pickle   test_1_true_resume.pickle
test_1_overlap_control_result.json      test_1_true.tim
```

The in_out_plots and pp_plots directories (see section 5) can be used to get an overview of all 200 simulations in a test. After running slurm_test_01.sh, output directories in_out_plots/outdir01 and pp_plots/outdir01 can be produced containing plots of input parameters against output parameters and PP-plots respectively. in_out_plots/outdir01 contains the following output

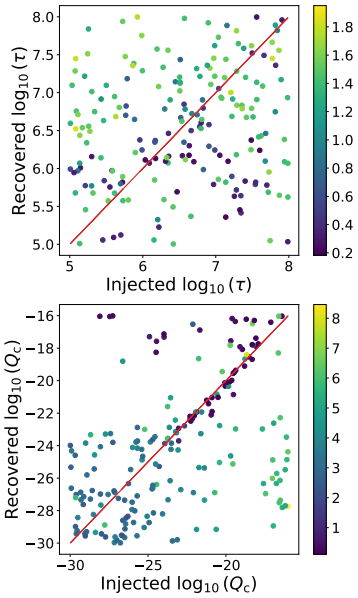
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_nooverlap_control.png  in_out_modes_ll_tau_Qc_colour_nooverlap_control.png
in_out_modes_ll_colour_nooverlap.png          in_out_modes_ll_tau_Qc_colour_nooverlap.png
in_out_modes_ll_colour_overlap_control.png    in_out_modes_ll_tau_Qc_colour_overlap_control.png
in_out_modes_ll_colour_overlap.png            in_out_modes_ll_tau_Qc_colour_overlap.png
in_out_modes_ll_colour_true_control.png       in_out_modes_ll_tau_Qc_colour_true_control.png
in_out_modes_ll_colour_true.png               in_out_modes_ll_tau_Qc_colour_true.png
```

and pp_plots/outdir01 contains the following output

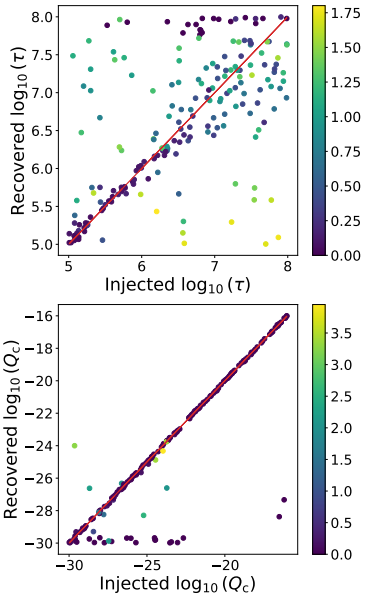
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
test_pp_plot_nooverlap_control.png  test_pp_plot_overlap_control.png  test_pp_plot_true_control.png
test_pp_plot_nooverlap.png          test_pp_plot_overlap.png          test_pp_plot_true.png
```

The main results of interest are those seen in the paper which should look like this

Figure 11



(a) in_out_modes_tau_Qc_colour_nooverlap.png. Column 1 of Figure F2 of the paper.



(b) in_out_modes_tau_Qc_colour_true.png. Column 2 of Figure F2 of the paper.

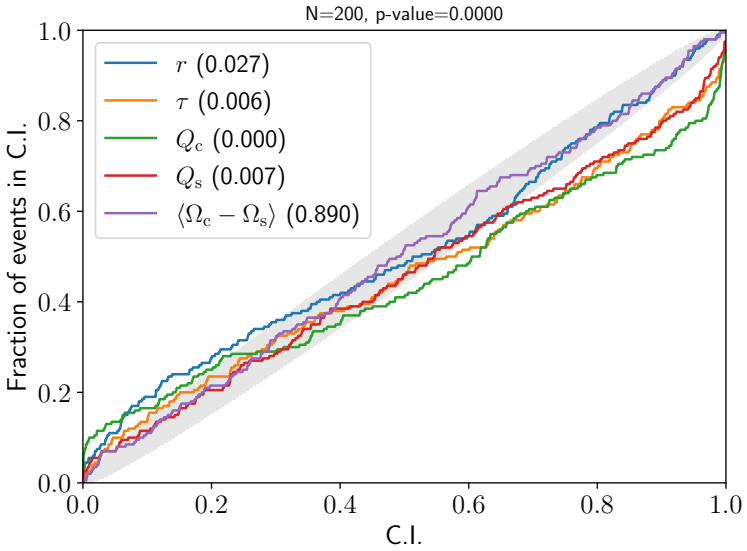


Figure 12: `test_pp_plot_nooverlap.png`. Figure F3 of the paper (the formatting of this figure is slightly different in the paper but the curves are the same).

5 `in_out_plots` and `pp_plots`

Each test simulates `Nsims` (200 by default) pulsars and saves the results in an output directory. The programs in the `in_out_plots` and `pp_plots` directories can analyse the results of all these pulsars together as is seen in the paper. `in_out_plots` produces the plots showing the injected parameters plotted against the recovered parameters and the `pp_plots` directory produces a PP-plot for each set of simulations.

5.1 `in_out_plots`

The programs in `in_out_plots` plot the injected parameters against the recovered parameters as seen in the appendices of the paper. To make the plots of injected parameters against recovered parameters, follow the commands

```
cd pulsar_freq_filter/{simulation_directory}/in_out_plots
bash analyse_modes_ll_noload.sh
```

({`simulation_directory`} is the directory where the simulations were carried out (either `freq_programs/test01_R_noise_size_test/`, `freq_programs/test02_R_noise_vary_test/`, `toa_programs/test01_random_toas/` or `toa_programs/test02_cadence_toas/`)).

The script `analyse_modes_ll_noload.sh` calls `analyse_modes_ll_noload.py` with some preset arguments that can be adjusted. The inputs to `analyse_modes_ll_noload.py` are

- `output_directory` - The name of the directory that the simulation results were stored in. The default is `outdir01`.
- `tag` - The tag that went in front of the simulation file names. The default is `test`.
- `savestr` - This tag says what kind of simulation was run. Examples include `_EFAC`, `_noEFAC`, `_nooverlap`, `_nooverlap_control`, `_overlap`, `_overlap_control`, `_true`, `_true_control`.
- `Nsims` - The number of simulations that were done in the output directory. The default is 200.

`analyse_modes_ll_noload.py` is called once for each type of result produced by the simulation program, with a corresponding `savestr` argument determined by the filenames of the results in the simulation output directory. For example, in Section 3.2, the simulation program `ns_sim_random_vary_EM_only_EFAC.py` runs the parameter estimation algorithm twice, once with an EFAC parameter and once without one, so `analyse_modes_ll_noload.py` is run twice: once with `savestr='_noEFAC'` and once with `savestr='_EFAC'`. So `analyse_modes_ll_noload.sh` produces the following output

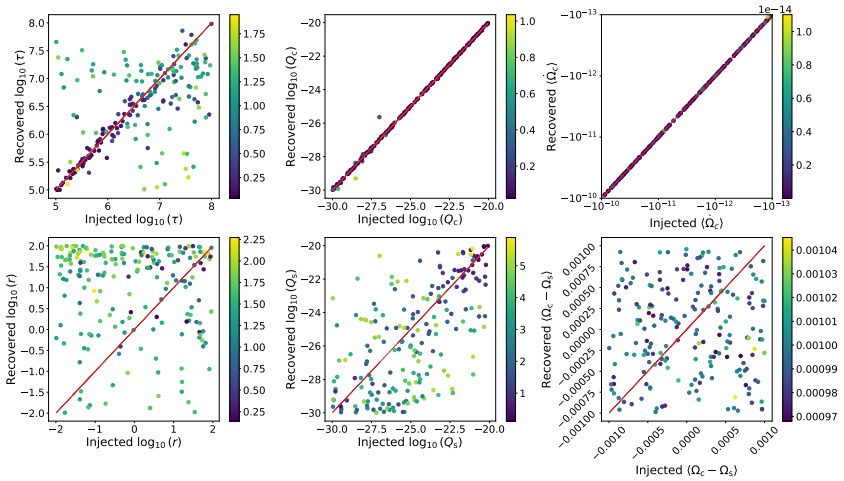
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_EFAC.png     in_out_modes_ll_tau_Qc_colour_EFAC.png
in_out_modes_ll_colour_noEFAC.png   in_out_modes_ll_tau_Qc_colour_noEFAC.png
```

whereas in section 4.1 the simulation program `ns_sim_random_vary_EFAC.py` is more complicated and runs the parameter estimation algorithm six times so the output directory should contain the following output

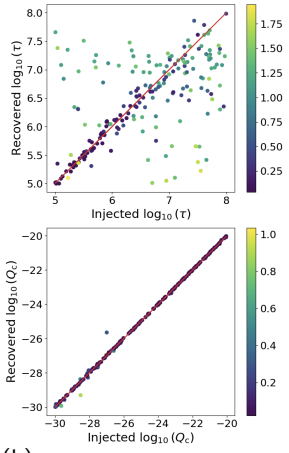
```
(new_env) [noneill@tooarrana1 outdir01]$ ls
in_out_modes_ll_colour_nooverlap_control.png  in_out_modes_ll_tau_Qc_colour_nooverlap_control.png
in_out_modes_ll_colour_nooverlap.png          in_out_modes_ll_tau_Qc_colour_nooverlap.png
in_out_modes_ll_colour_overlap_control.png    in_out_modes_ll_tau_Qc_colour_overlap_control.png
in_out_modes_ll_colour_overlap.png            in_out_modes_ll_tau_Qc_colour_overlap.png
in_out_modes_ll_colour_true_control.png       in_out_modes_ll_tau_Qc_colour_true_control.png
in_out_modes_ll_colour_true.png               in_out_modes_ll_tau_Qc_colour_true.png
```

The important outputs are `in_out_modes_ll_01_colour_{savestr}.png`, which plots all the two-component model parameters that are being estimated and `in_out_modes_ll_01_tau_Qc_colour_{savestr}.png` which just plots the τ and Q_c results. An example from section 3.1 of each type of plot is shown below.

Figure 13



(a) `in_out_modes_tau_Qc_colour_noEFAC.png`



(b) `in_out_modes_tau_Qc_colour_noEFAC.png`

More examples are shown in the earlier sections.

5.2 `pp_plots`

The programs in the `pp_plots` directory produce a PP-plot for the two-component model parameters from the 200 simulations done in a test.

To make the PP-plots, follow the commands

```
cd pulsar_freq_filter/{simulation_directory}/pp_plots
bash make_pp_plots.sh
```

({`simulation_directory`} is the directory where the simulations were carried out (either `freq_programs/test01_R_noise_size_test/`, `freq_programs/test02_R_noise_vary_test/`, `toa_programs/test01_random_toas/` or `toa_programs/test02_cadence_toas/`)).
The script `make_pp_plots.sh` calls `make_pp_plots.py` with some preset arguments that can be adjusted. The inputs to `make_pp_plots.py` are

- `output_directory` - The name of the directory that the simulation results were stored in. The default is `outdir01`.

- `tag` - The tag that went in front of the simulation file names. The default is `test`.
- `savestr` - This tag says what kind of simulation was run. Examples include `_EFAC`, `_noEFAC`, `_nooverlap`, `_nooverlap_control`, `_overlap`, `_overlap_control`, `_true`, `_true_control`.
- `Nsims` - The number of simulations that were done in the output directory. The default is 200.

This should make a directory `outdir01` containing one PP-plot for each result type. For example, the output for section 3.1 is as follows

```
(new_env) [noneill@tooarrana1 outdir01]$ ls
test_pp_plot_noEFAC.png
```

An example PP-plot from section 3.1 is shown below

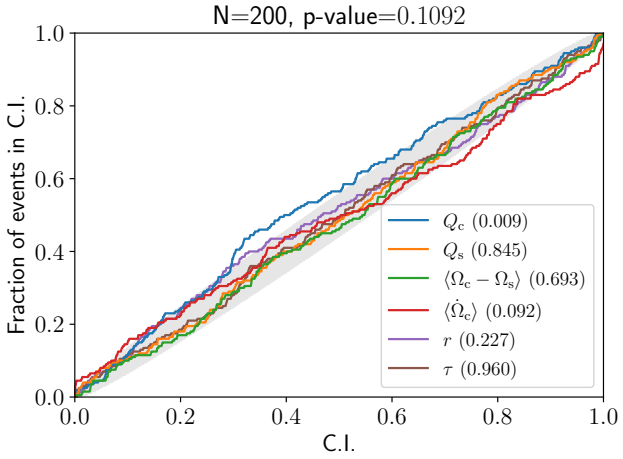


Figure 14: `test_pp_plot_noEFAC.png`

More examples of PP-plots are seen in the earlier sections.

References

[1] Baym. G. et al. *Spin Up in Neutron Stars : The Future of the Vela Pulsar*, Nature, Volume 224, pp. 872-874 (1969).

[2] Meyers, P. M. et al. *Parameter estimation of a two-component neutron star model with spin wandering*, Monthly Notices of the Royal Astronomical Society, Volume 502, Issue 3, pp.3113-3127 (2021).

[3] Meyers, P. M. et al. *Rapid parameter estimation of a two-component neutron star model with spin wandering using a Kalman filter*, Monthly Notices of the Royal Astronomical Society, Volume 506, Issue 3, pp.3349-3363 (2021).

[4] Speagle. J. S. *DYNESTY: a dynamic nested sampling package for estimating Bayesian posteriors and evidences*, Monthly Notices of the Royal Astronomical Society, Volume 493, Issue 3, pp.3132-3158 (2020).

[5] Ashton. G. et al. *BILBY: A User-friendly Bayesian Inference Library for Gravitational-wave Astronomy*, The Astrophysical Journal Supplement Series, Volume 241, Issue 2, p.27 (2019).

[6] Lower, M. E. et al. *The UTMOST pulsar timing programme - II. Timing noise across the pulsar population*, Monthly Notices of the Royal Astronomical Society, Volume 494, Issue 1, pp.228-245 (2020).