Introduction to pulsar timing: From search data to phase-connected timing solution

Emilie Parent
parente@physics.mcgill.ca

October 2020

Pulsar timing consists of monitoring pulsar rotations by tracking the times of arrival of the radio pulses emitted by the pulsar. In this type of analysis, we absolutely must account for every single rotation of the pulsar over a time period of at least one year. This unambiguous tracking of the pulsar rotations allows us to determine its observable properties with high precision, and use these pulsars as exquisite "celestial clocks". Pulsar timing notably enables studies of neutron-star interior physics, gravitational theories in the strong-field regime, binary evolution, the interstellar medium, and can even be used as "celestial GPS" for spacecraft navigation. During the last decade, large international efforts have also been conducted to detect low-frequency gravitational waves via pulsar timing arrays.
This document can be used as a guide to obtained a phase-connected timing solution from radio search-mode pulsar data. It is specifically intended for students that being introduced to pulsar timing.
We will only focus on isolated pulsars, and not discuss the treatment of pulsars in binary systems.

To perform the timing analysis described in this document, you will need two pulsar softwares: `presto`[1] for data preparation, and `tempo`[2].

In brief, the steps for obtaining a phase-connected timing solution are:

1. Data preparation. For each data file, do the following:
   (i) Clean the data by producing a RFI mask. (`rfifind`)
   (ii) Make a topocentric time series, dedispersed at the pulsar's DM (`prepdata`)
   (iii) Fold the data at the pulsar's period (`prepfold`)
2. Extract times of arrival (TOAs) from each `prepfold` plots
3. Proceed to phase-connection with `tempo`

Before we look at each step in more details, here are some definitions for terms and concepts that will often be referred to in this guide.

Definitions

tim file: a file containing a pulsar's TOAs. This file is required by `tempo`.

par file: a file containing the pulsar's ephemeris/timing solution (see below). `tempo` uses this par file (combined with the tim file) to carry out the timing analysis. It lists parameters such as the pulsar's name (PSR), position (RAJ and DECJ), spin frequency (F0), spin frequency derivatives (F1, F2, F3, ..), etc. The par file is also where you indicate whether a parameter should be kept fix or be fitted by `tempo`.

Template profile: the reference pulsar pulse profile that has the highest-signal-to-noise, used to extract TOAs from all MJDs (the .pfd.bestprof created by `prepfold` – see below)

[1] https://www.cv.nrao.edu/$\sim$sransom/presto/
[2] http://tempo.sourceforge.net/reference_manual.html

TOA: time at which a pulsar pulse arrives at an observatory. It is expressed in units of days in the Modified Julian Date (MJD) calendar.

Timing solution: model that includes the astrometric, spin and binary (if applies) parameters of a pulsar, which have been precisely determined in a least-square fit analysis of the measured TOAs. Best-fit parameters and/or starting parameters of the timing solutions are listed in a par file. Ephemeris is another term used when referring to a pulsar timing solution.

Phase-connection: counting the number of pulsar rotations between TOAs. A timing solution that is phase-connected is a model that accounts for every pulsar rotation. The model solution can predict with high-precision when a pulse will arrive at an observatory at any future date and time.

Significant measurement: when the value of a measured parameter (i.e., the ones fitted with `tempo`) is at least 2 times larger than it's uncertainty, σ. In other words, when a parameter is more than 2σ away from 0.

Now, let's look more closely at each step of the analysis.

1 Data preparation

For this part of the analysis, we'll be using three `presto` commands: `rfifind`, `prepdata` and `prepfold`.
Note: Text in purple are commands that can be run directly on the terminal, and items that have to be customized to your own pulsar and dataset are in *italic*.
Gather all the data files from your search-mode observations[3], and do the following for each file:

(i) Clean the data by producing RFI mask.
Run `rfifind` on the terminal:
`rfifind -time 1.0 -o` *filename* *filename*`.fits >` *filename*`_rfifind.out`
where *filename* must be replaced by the name of the file before the .fits extension.

The above command will have the `rfifind` output (which includes the masking fraction) to the file *filename*_rfifind.out. It will also produce additional files, including the mask itself (*filename*_rfifind.mask), and a plot showing the mask (*filename*_rfifind.ps)

Look into the *filename*_rfifind.out, and make sure that the masking fraction is reasonable (less than ∼40%). If the masking fraction is too high, you can try re-doing the masks by running the `rfifind` command using a shorter time (e.g., : `rfifind` -time 0.5 . . .)

(ii) Make a topocentric time series, dedispersed at the pulsar's DM.
Note: This step could be skipped and folded data can be obtained directly from the search data (see section iii)), but folding time series rather than search data speeds up the folding process significantly. The downside to folding time series, however, is that all frequency information is lost, and that frequency information is needed to fit for DM in the timing analysis. I recommend producing and folding time series, but for the brightest detections of the pulsars, re-fold directly the search data instead.

To produce a topocentric, dedispersed time series, run `prepdata` as:

`prepdata -nobary -dm` *DM* `-mask` *filename*`_rfifind.mask -o` *filename*`_topo` *filename*`.fits`
where *DM* must be replaced by the pulsar's DM value.

[3]This assumes that the raw data have already been subbanded. Instructions on subbanding raw data are provided in the Additional notes section

This will create the time series containing the data (*filename*_topo.dat) and a file containing information on the produced time series (*filename*_topo.inf). The -nobary flag tells `prepdata` to not apply barycentric correction, i.e., use the observatory time.

Make sure that ALL data files have been dedispersed into time series at the same DM value. You can look at the time series using `exploredat`, another `presto` command:

`exploredat` *filename*`_topo.dat`

(iii) Fold the data at the pulsar's period.
This step may require more than one iteration of `prepfold` in order to produce a `prepfold` plot as good as possible. This is much faster when folding time series, but it can also be done directly on the search data.

First, run `prepfold` on the time series as follows:

`prepfold -par` *path/to/parfile*`.par -nosearch -n` *N* `-fine` *filename*`_topo.dat`
where *parfile*.par is the initial parameter file, and *path/to/parfile*.par is the directory path to the par file (so that `prepfold` can locate the par file). It contains information on the pulsar, including the spin frequency that `prepfold` needs to know to fold the data at the pulsar's period. *N* is the number of profile bins to use when plotting the data, and it has to be a power of 2 (usually 64, 128, 256 or 512).

The -nosearch flag tells `prepfold` to not search over the period-observed period derivative (p-pdot) parameter space. Note: observed period derivatives are only detectable within a few minutes of data for short-period pulsars in binary systems with a short orbital period. The true pulsar's spin period derivative is only detectable if we have ∼1 year or more of timing data. The -fine flag tells `prepfold` to use a finer gridding in the p-pdot plane.

If instead you want to fold the data directly rather than the time series, you need to specify the folding DM (using the "-dm" flag) and the number of frequency subbands to plot (using the "-nsub" flag):
`prepfold -par` *path/to/parfile*`.par -nosearch -n` *N* `-fine -dm` *DM* `-nsub` *Nsub* *filename*`.fits`

`Prepfold` will write the folding information in *filename*_topo_PSRNAME.pfd.bestprof, and will display the `prepfold` plot on the screen (will also write it to the file *filename*_topo_PSRNM.pfd.ps).

Notes:

- Decreasing the value for N increases the signal's strength, but reduces the profile resolution. Therefore, you don't want too few nor too many profile bins.
- If the initial par file does not exist, you can create your own. It must follow the format required by `tempo`, which is provided in the `tempo` manual. The parameters that must be included in the initial par file are: PSR, RAJ, DECJ, F0, DM, PEPOCH, and UNITS. Also note that the values for F0 and PEPOCH must be **barycentric**.

You can tell if the fold is good enough by looking at the p-pdot search-parameter plane (see Figure 1).

If the fold is not good, then you must re-fold allowing `prepfold` to search for a slightly different period. You can do that by running `prepfold` again but removing the -nosearch flag:

`prepfold -par` *path/to/parfile*`.par -n` *N* `-fine` *filename*`_topo.dat -nopdsearch`

The -nopdsearch tells `prepfold` to not search over the period derivative, only over the period.

If the plot looks good, extract the best topocentric period (P_topo) from the .bestprof file and feed that into `prepfold`. This time, we will not use the par file to tell `prepfold` what period to use but we will

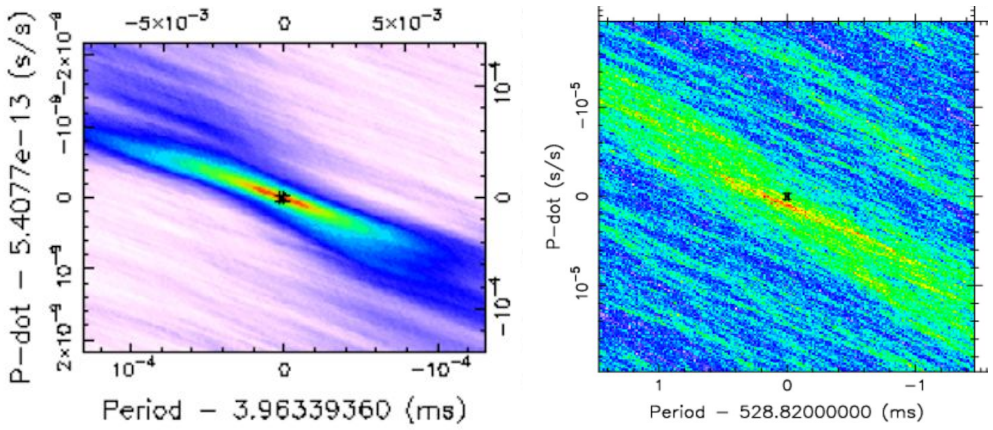

Figure 1: Examples of p-pdot search space produced by `prepfold` (in the bottom right of `prepfold` plots). The color scales with the S/N (or `prepfold` σ) of the pulsar detection (red being optimal). The plot on the left is an example of a good fold. You can see that the cursor (indicating the values used in the current fold) is right in the center of the best p-pdot values. On the other hand, the plot on the right is an example of a bad fold. You can see that the cursor is not aligned with the best p-pdot values.

instead give it the value from the .bestprof file, and won't allow it to search for either a period or period derivative:

`prepfold -n N -fine filename_topo.dat -nosearch -p period`
where *period* is the value of P_topo in the .bestprof file IN SECONDS (the value in the .bestprof file is given in milliseconds). If the plot still does not look good, you must iterate the above two steps until the fold is good.

It is possible that there is bad RFI still in the data. For example, one can see in the following plots that there is RFI from looking at the Time vs Phase panel of the `prepfold` plot shown in Figure 2.

You can tell `prepfold` to ignore the first 20% (or whatever fraction needs to be taken out) with the -start flag:

`prepfold -n N -finefilename_topo.dat -nosearch -p period -start 0.2`

Equivalently, you can do the same if the RFI is at the end of the file with the -end flag. To ignore the last 20% of the sub-integrations, run:

`prepfold -n N -finefilename_topo.dat -nosearch -p period -end 0.8`

RFI masks produced with `rfifind` are not perfect: it is likely that there will still be RFI present in the data. Dataset that are highly contaminated by RFI should be folded from the search data directly, not the time series. While each iterations of `prepfold` on raw data takes longer, it allows you to see the observing frequency versus rotation phase and allows for additional RFI excision. Bad frequency channels can then be removed explicitly when running `prepfold` with the -ignorechan option:

`prepfold -n N -finefilename.fits -nosearch -p period -dm DM -topo -ignorechan chan1`
where *chan1* is the channel to be removed. Note that we also need to provide the pulsar's DM. We also

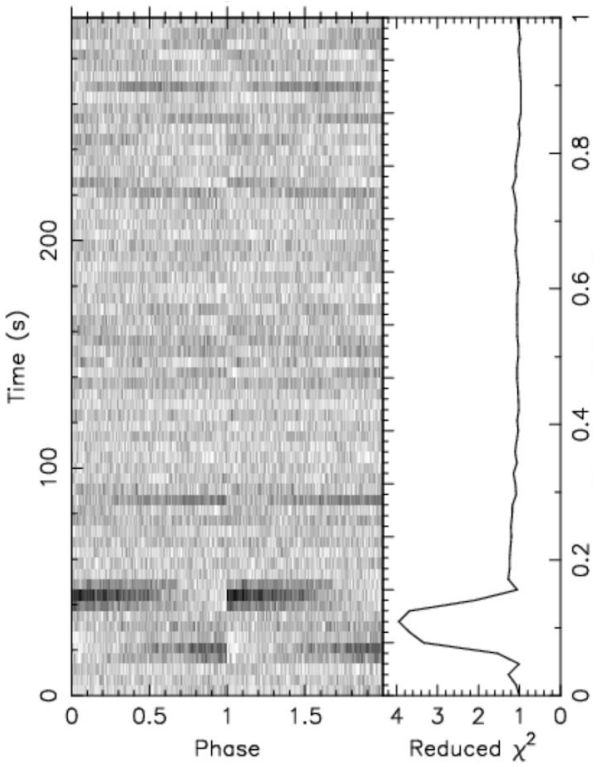

Figure 2: Time vs Phase panel of a `prepfold` plot, where there is no clear pulsar signal but RFI can easily be identified in the first ∼20% (see y label on the right) of the sub-integrations.

need to tell `prepfold` to measure folding parameters in the topocentric frame using the -topo option. It is important for the folds to be topocentric for the timing analysis.

IMPORTANT! the final fold, which we will use to extract TOAs, must always be produced with the -nosearch flag.

Make sure to note which `prepfold` files were generated by the last, optimal fold, if it `prepfold` was called more than once.

Other options can be used with `prepfold` to improve the quality of the fold (e.g., -runavg to eliminate rednoise in the sub-integrations, -npart to change the number of sub-integrations, etc). See the complete list of options by typing `prepfold` on your terminal.

2 Extract TOAs from `prepfold` plots

(i) The first step is to select what `prepfold` plot we will use as a template profile.

TOAs are extracted by cross-correlating a high-signal-to-noise pulsar profile (the template) to the folded profile.

The actual file used as the template is the pfd.bestprof generated by `prepfold` on the MJD that produced the highest `prepfold`-σ detection of the pulsar. In addition to information on the folding parameters, the pfd.bestprof contains the folded profile intensity versus phase bin, which is used as the template profile.

Important! There is only one template for the entire set. Make sure to note what file you used as the template, as you may need it later.

For more details on template matching, see section 8.1.1 of the Handbook of Pulsar Astronomy.

(ii) Extract TOAs from the .pfd files produced by `prepfold`, for all MJD.

You must first determine how many TOAs to extract per `prepfold` folds. Unfortunately, we can't extract one TOA for each pulsar rotation, because as opposed to RRATs, the intensity of individual pulses is usually too faint to be detected (folding is necessary to detect the pulsar signal). Also, the pulse morphology changes from one rotation to next: only the summed profile is stable.

The way TOAs are extracted is by "splitting" the integration into sub-folds (the number of sub-folds will be the number of TOAs). For AO data, we normally extract 1, 2, 4 or 8 TOAs per MJD, depending on the pulsar brightness.

Generally, having more TOAs for a single MJD may help obtaining a phase-connection faster, but the precision on each TOAs reduces.
Equivalently, having less TOAs produces better, more precise TOAs but it may be more difficult to keep track of all pulsar rotations.

Choosing how many TOAs should be extracted from a fold is rather subjective. You only need to make sure that the pulsar signal is visible in each sub-fold. However, if a pulsar shows nulling behavior and is only detectable in, say, the second half of the observation, you can produce multiple TOAs and reject the ones where the pulsar signal is undetectable. In Figure 3, you can see examples of what reasonable numbers of TOAs would be for different pulsar detections.

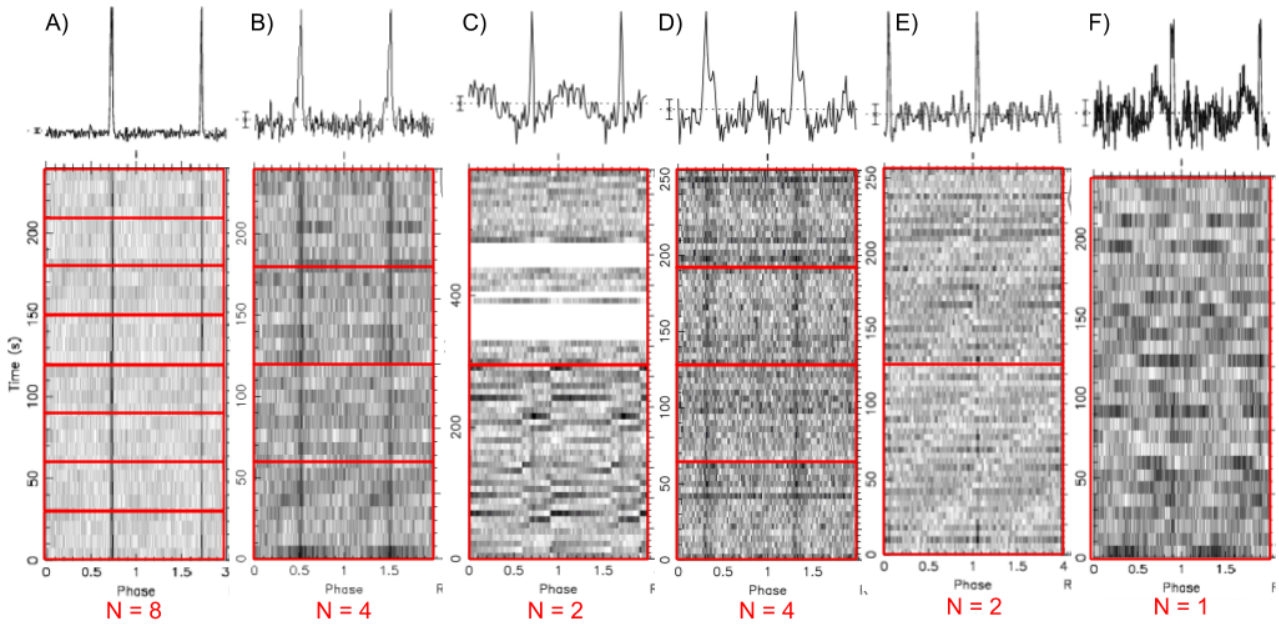

Figure 3: Time vs phase plots of different pulsars. An acceptable choice of number of TOAs to extract, N, is provided for each pulsar. Red boxes on the plots also show how sub-folds are separated for TOA extraction.

Now, for the actual extraction of TOAs, we use the get_TOAs.py program from `presto`:

```
get_TOAs.py -t template.pfd.bestprof -n N filename.pfd
```

where *template*.pfd.bestprof is the template you chose in step i), N is the number of TOAs you want to extract from the fold and *filename*.pfd is the `prepfold` fold file.

This will print the TOAs on the terminal.
You can copy them all into a tim file, which will include all TOAs needed for the timing analysis with `tempo` (must be in chronological order).

The tim file containing the TOAs would look similar to:

```
3 1380.195 58770.9061230461367 1224.87
3 1380.195 58770.9069952466096 1373.30
3 1380.195 58770.9078623543336 1298.81
3 1380.195 58770.9087345854787 864.08

3 1380.195 58798.8180142683254 2567.59
3 1380.195 58798.8188154131565 2405.62
3 1380.195 58798.8196166237001 2269.77
3 1380.195 58798.8204178231125 1765.14

3 1380.195 58815.7701926794426 2653.54

3 1380.195 58816.7985783416085 1555.07

3 1380.195 58829.7522093505333 12911.35
3 1380.195 58829.7539486297999 3445.06
...
```

On each row: "3" is the observatory code for Arecibo, 1380.195 is the frequency of the TOAs in MHz, the MJD date is the actual TOA (e.g., 58770.9061230461367), and the last number is the uncertainty on the TOA, in units of micro-seconds.

(iii) Once all TOAs are in the tim file, we'll need to tell `tempo` that the TOAs between different MJDs are NOT phase-connected. This is done by placing pairs of JUMP statement in the tim file around each MJDs, **except for one set**. This set will be used as a fiducial point.

We'll also add "MODE 1" on the first line of the tim file: this will tell `tempo` to return the reduced chi-square values when doing the phase connection later on.

The tim file should now look like:

```
MODE 1
3 1380.195 58770.9061230461367 1224.87
3 1380.195 58770.9069952466096 1373.30
3 1380.195 58770.9078623543336 1298.81
3 1380.195 58770.9087345854787 864.08

JUMP
3 1380.195 58798.8180142683254 2567.59
3 1380.195 58798.8188154131565 2405.62
3 1380.195 58798.8196166237001 2269.77
3 1380.195 58798.8204178231125 1765.14
JUMP

JUMP
3 1380.195 58815.7701926794426 2653.54
JUMP

JUMP
3 1380.195 58816.7985783416085 1555.07
JUMP

JUMP
3 1380.195 58829.7522093505333 12911.35
3 1380.195 58829.7539486297999 3445.06
JUMP
...
```

When trying to phase-connect (i.e., counting the number of pulsar rotations between different TOAs), `tempo` does not have to count the number of pulsar rotations between TOA sets surrounded by JUMPs. Only TOAs not separated by JUMPs are phase-connected.

For instance, all TOAs in MJD 58798 are phase-connected, but the TOA 58815.7701926794426 is not phase-connected with TOAs in 58798.

To extract TOAs at different frequencies from folded search data, see the **DM fitting** section of the **Additional notes** at the end of this guide.

3 Phase-connection with `tempo`

The program used to obtain a phase-connected timing solution is `tempo`. A user guide to `tempo` is provided here.

First, we will discuss the concept of producing a phase-connected timing solution, then we will dive into the technical details.

Concept Achieving phase-connection with `tempo` is an iterative process. Initially, when TOAs from different MJDs are NOT phase-connected (i.e., they are surrounded by pairs of JUMPs in the tim file) only one pulsar parameter is fit for: the fundamental spin frequency, F0. This is the first pulsar parameter for which we can obtain a significant measurement.

You'll want to remove JUMPs between two MJD sets, and run `tempo` to fit the TOAs to update the timing solution (whose parameters are in the par file), where now two MJDs sets are phase-connected.

If the updated timing solution looks reasonable, i.e. : 1) the reduced-chi-square is smaller than 2 or 3, 2) there seems to be no pattern in the residuals, in other words the residuals "look flat" and 3) the parameters in the `tempo` have not changed too much, then can move on and remove another pair of JUMPs. This time, you provide the updated timing solution to `tempo` to fit again the TOAs.

At some point during this iterative process, you'll reach a point where `tempo` does not return a good timing solution, and the residuals are all over the place. This means either it has miscounted pulsar rotations somewhere, or that it is time to fit for more parameters.

For slow pulsars that are isolated, the only fitted parameters are typically F0, F1, RAJ and DECJ (see `tempo` user guide for a definition of all parameters). If the pulsar is bright and the timespan of the timing data is long (2+ years), then it may be possible to fit for F2, PMRA and PMDEC (depending on the pulsar's distance).

When allowing `tempo` to fit for a new parameter, you have to make sure that it returns a best-fit value that is reasonable for that parameter. For instance, if you attempt to fit for RAJ and `tempo` returns a solution where the new RAJ has changed by half an hour angle, then the solution is unreasonable. You have to discard this solution, and go back to the previous one and try something else. You also need to ensure that the measurement is significant. For example, if F1 is -2.13D-16 and its uncertainty (also called sigma) is 1.53D-16, then the significance of the measurement on F1 is less than 2 sigma. This means that the measurement is not significant, i.e., F1 is consistent with being zero at a 2-sigma level. Fitting for this parameter is therefore not recommended yet: you'll need to phase-connect more TOAs to be able to detect the effect of F1.

When `tempo` produces an unreasonable solution (either fitted parameters are bogus, or the reduced-chi-square is very large) there are a few things you can do. Look for patterns, or missed/extra rotations in the residuals, and then write those explicitly in the tim file where the missed/extra rotations occur. This can happen a lot at the beginning of the phase-connection process if the initial par file contained parameters that are not accurate. You are fitting a parameter that you should not be fitting: the timing solution is not precise enough to measure that parameter yet.

This is done until all MJDs are phase-connected (i.e., no more JUMPs in the tim file).

Technique:
The initial par file should look somewhat similar to the following:

```
PSR J0623+15
RAJ 06:23:00.4524
DECJ 15:35:21.5952
PEPOCH 57294.398657
F0 1.294734938 1
DM 92.5
UNITS TDB
```

You may have some extra lines, and it's okay. It is just important that you have the ones listed above. Again, you can refer to the tempo manual to get a full description of the parameters.

Note the 1 at the end of the line for F0. Having a 1 after a parameter tells tempo to actually fit for that parameter. Other parameters without a 1 are kept fixed during the tempo fit.

Every time you want to fit TOAs to update timing solution (which gets written in a new par file), you run the tempo program as follows:

`tempo -f` *`parfile`*`.par` *`timfile`*`.tim`

This will print a few lines on your terminal, and the last two lines include the pre-fit and post-fit timing residuals, the chi-square (Chisqr) value of the fit as well as the number of degrees of freedom (nfree), and importantly the reduced chi-squared:

```
...
Weighted RMS residual:  pre-fit 6643.277 us.  Predicted post-fit 1021.570 us.
Chisqr/nfree:  148.98/ 180 = 0.827670989 pre/post:  6.50 Wmax:  7.1
```

IMPORTANT! the updated timing solution will be written in ¡PSR¿.par, where ¡PSR¿ is the pulsar name provided in the par file given to tempo (on the line that starts with PSR in the par file). If your initial `tempo` is called ¡PSR¿.par, then it will get OVERWRITTEN by tempo with the new timing solution, and your initial `tempo` will be lost. You don't want that to happen, since you won't be able to take a step back if your last tempo run produces a bad fit. You can remediate that by using a slightly different PSR name in the initial `tempo`, for example:

```
PSR J0623+15_2
```

You will then be able to compare the new timing solution (which will be written in a file called J0623+15_2.par) to the initial `tempo`. Once you have confirmed that the fit is reasonable, you can change the PSR name back to the original name in the initial `tempo`, and let tempo overwrite/update that `tempo`.

For what comes next, I'll use J0623+15 as an example. I have called my initial `tempo` J0623+15.par, and my tim file J0623+15.tim.

Step by step procedure:

(a) Make a backup copy of the initial `tempo`, just in case you make a mistake at some point and need to restart from the beginning. Call the backup copy something different, like initial_parfile.par, to make sure it never gets overwritten by tempo.

(b) Change the PSR name in J0623+15.par, and allow tempo to fit for F0 only:

```
PSR J0623+15_2
RAJ 06:23:00.4524
```

```
DECJ 15:35:21.5952
PEPOCH 57294.398657
F0 1.294734938 1
DM 92.5
UNITS TDB
```

(c) Run tempo:

```
tempo -f J0623+15.par J0623+15.tim
```

(d) Is the fit reasonable?:

- Check that the resulting reduced chi-squared is reasonable (< 2 3)
- Look at the new best-fit parameters (only F0 in this case) in J0623+15_2.par, and make sure that it's reasonable. This will be important when fitting for RAJ, DEC or F1. Note that there will be many more lines in the J0623+15_2.par, and that's normal.
- Look at the residuals with a residual plotter:
  `pyplotres.py`
  pyplotres.py is a `presto` program that reads the best-fit residuals from the files produced by the last `tempo` run (no need to provide any input on the command line, as long as you are in the same folder `tempo` was last run in). A plot similar to Figure 4 will be displayed.

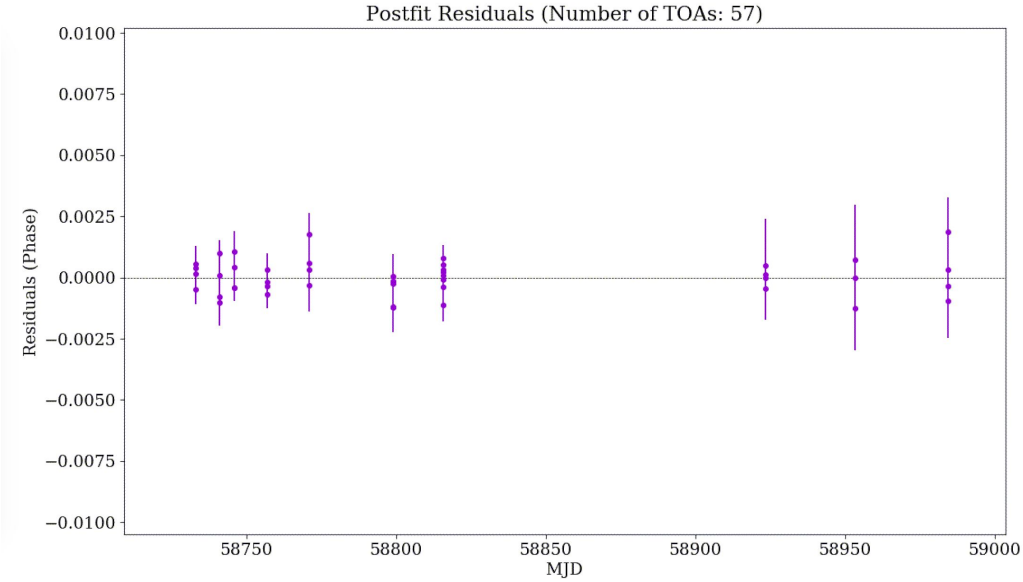

Figure 4: Fitting residuals, i.e., difference between TOAs and times of arrival predicted by the timing model generated by the last `tempo` iteration. Each point is a TOA residual in units of rotational phase, and the bars are the residual uncertainties. The distribution of TOAs is "flat", indicating that there are no noticeable unmodeled effects.

Make sure that all residuals are within 2-3 error bars of being zero. You also want to make sure that the residuals are roughly within +/- 0.02 in units of rotational phase.
If some TOAs that are WAY OFF, but the rest are mostly consistent with zero and no trend is present in the residual distribution, they are probably RFI. This can also explain a large

reduced chi-square returned by `tempo`. For example, you could see something like shown in Figure 5.

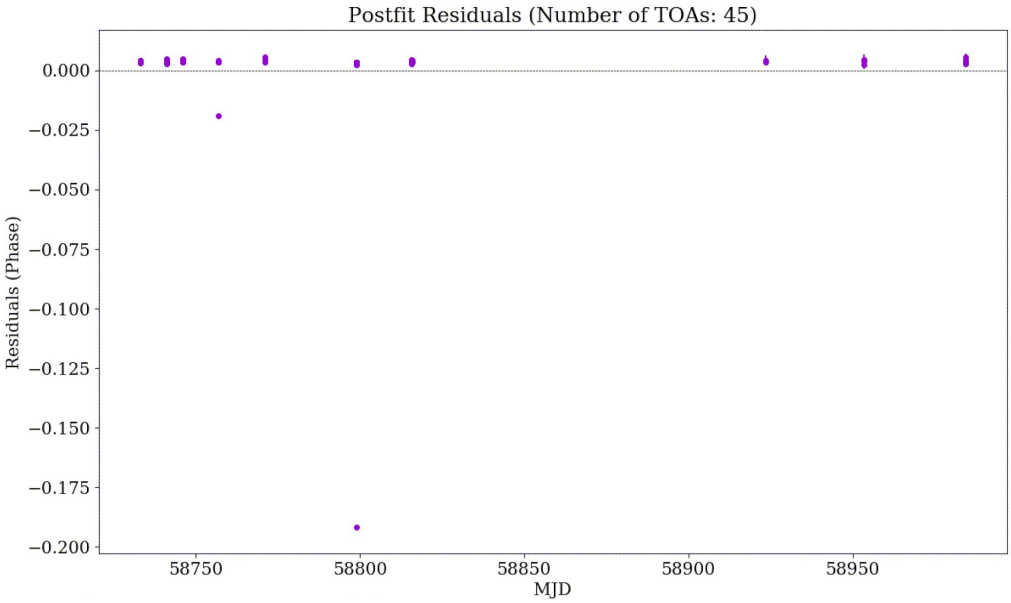

Figure 5: Here, two TOAs have residuals that are much larger than the rest. These TOAs were generated from sub-folds corrupted by RFI, and the template cross-correlations resulted in unusable TOAs.

You can zoom in the plot to determine which TOA those are (e.g., 2nd TOAs in a set of 4 on MJD 58798). You can also hit "tab" twice on your keyboard, and it changes the x-axis to the TOA number. You can also click on the TOA, and information on that TOA will be displayed on the terminal. Go in the tim file, and comment out the bad TOAs by writing "C " at the start of the line:

```
JUMP
3 1380.195 58798.8180142683254 2567.59
C 3 1380.195 58798.8188154131565 2405.62
3 1380.195 58798.8196166237001 2269.77
3 1380.195 58798.8204178231125 1765.14
JUMP
```

Running `tempo` again should produce a better fit.
Something else that could happen is that `tempo` missed/added an extra pulsar rotation somewhere (which would also produce a large reduced chi-square). In the residuals plot, this would look similar to Figure 6.
Notice the phase difference is 1 between the first TOAs and the last fews in the last MJD.
You can see that `tempo` missed a rotation between the first and the second TOAs of this MJD set.
You can tell `tempo` to add one pulsar rotation between these two MJDs by adding PHASE +1 in the tim file :

```
JUMP
3 1380.000 58984.4013391833593 1333.125 0.00000
PHASE +1
3 1380.000 58984.4019116839415 832.818 0.00000
3 1380.000 58984.4025046477005 1237.980 0.00000
3 1380.000 58984.4030669032990 1043.986 0.00000
JUMP
```

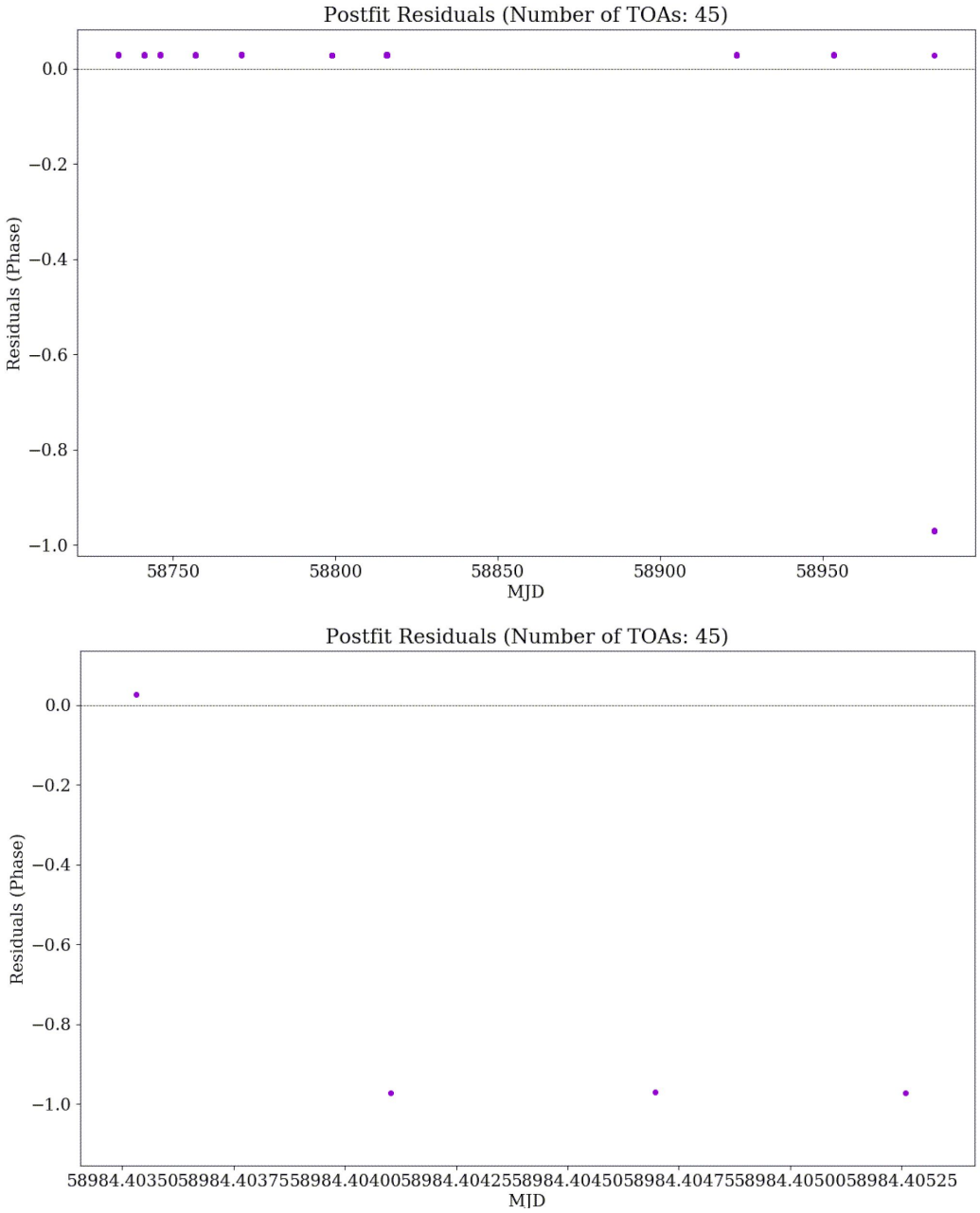

Figure 6: TOA residuals where one pulsar rotation was unaccounted for in the fitted timing model.

If `tempo` instead added an extra rotation, you can write "PHASE -1" in the tim file. The next time you run `tempo`, you'll see that the TOAs are now aligned and the reduced chi-square has reduced.

(e) Once you are satisfied with the new fit, change the PSR name in J0623+15.par:

```
PSR J0623+15
```

And run `tempo` again to update J0623+15.par with the new timing solution:

```
tempo -f J0623+15.par J0623+15.tim
```

It is always a good idea to create a backup copy of the latest J0623+15.par after each iteration so that you can go back to the previous par file without having to start again from the beginning in case something goes wrong (but don't call that copy J0623+15.par, J0623+15_2.par or initial_parfile.par).

IMPORTANT NOTE: if you had added a PHASE +/- 1 statement in your tim file, you need to remove those statements once the par file has been updated. This new solution now accounts for that missed/added pulsar rotation, so you don't want it to add/subtract a pulsar rotation again in the next `tempo` iteration.

(f) Once J0623+15.par has been updated, change PSR again to prepare for the next `tempo` iteration:

```
PSR J0623+15_2
```

(g) Phase-connect the next most closely-spaced sets of TOAs in J0623+15.tim by commenting out JUMP statements between these two sets:

```
MODE 1
3 1380.195 58770.9061230461367 1224.87
3 1380.195 58770.9069952466096 1373.30
3 1380.195 58770.9078623543336 1298.81
3 1380.195 58770.9087345854787 864.08

JUMP
3 1380.195 58798.8180142683254 2567.59
3 1380.195 58798.8188154131565 2405.62
3 1380.195 58798.8196166237001 2269.77
3 1380.195 58798.8204178231125 1765.14
JUMP

JUMP
3 1380.195 58815.7701926794426 2653.54
C JUMP

C JUMP
3 1380.195 58816.7985783416085 1555.07
JUMP

JUMP
3 1380.195 58829.7522093505333 12911.35
3 1380.195 58829.7539486297999 3445.06
JUMP
...
```

(h) Run `tempo` again:

```
tempo -f J0623+15.par J0623+15.tim
```

(i) Check that fit is good. If it is, then change PSR in J0623+15.par to J0623+15 again, and run `tempo` to update J0623+15.par.

(j) Repeat these steps until the reduced chi-squared is becoming large (say the reduced chi-square has more than doubled) and the residuals start being all over the place, without having apparent missed/extra pulsar rotations.

Once you have reached that point, it means that it's time to fit for one additional parameter. You never introduce more than one new parameter to your fit at the time, but you continue fitting for whatever parameter you were fitting for before. For example, you will always fit for F0 even when you introduce new parameters to the fit.

The first few parameters you can fit for are:
RAJ : usually after a few months worth of phase-connected TOAs

DECJ : after close to a year of phase-connected TOAs)
F1 : after 1 year of phase-connected TOAs (depending on the age of the pulsar, may be detectable much faster for young pulsars). If the TOAs are not very precise (e.g., faint pulsar), then it may take longer to be able to fit for and measure significantly F1. On the other hand, if the pulsar is bright and young (therefore has a large period derivative), you may be able to measure F1 after a few months. However, this is rather rare. NOTE: F1 should be NEGATIVE (spin frequency decreases/spin period of pulsars increases with time)
F2 : sometimes detectable after 2 years or more of data, depending on TOA quality and pulsar age
PMRA/PMDEC : after 2+ years, but will probably not be detectable for distant pulsars.

(k) No more JUMPs to remove in the tim file ? You're done!

Additional notes

DM fitting

To fit for DM, `tempo` needs TOAs extracted at different observing frequencies. This cannot be done when extracting TOAs from time series, as the pulsar signal has been summed into one effective frequency. So we need to re-fold the data with `prepfold` but this time, folding directly the raw data rather than the time series. This does not have to be done on all datafiles in order to obtain a good DM fit. I also recommend doing a DM fit after a phase connection has been achieved so that spin and astrometric parameters are measured accurately first.

Step by step procedure:

(i) Identify the ∼2-4 observations where the pulsar is brightest based on your previous folds.

(ii) Fold the search data with `prepfold`, using the same number of profile bins you had been using:
`prepfold -par` *`path/to/parfile.par`* `-n` *`N`* `-fine` *`filename`*`.fits -nosearch -nsub` *`Nsub`*
where Nsub is the number of frequency subbands to produce when folding, and it has to be equal or smaller than the original number of subbands in the fits files. Using 64 subbands is typically a good number to use.

When looking at the `prepfold` plot, if you notice RFI in some frequency subbands, you can zap these channels explicitly by refolding using the "ignorechan" option:
`prepfold -par` *`path/to/parfile.par`* `-n` *`N`* `-fine` *`filename`*`.fits -nosearch -nsub Nsub -ignorechan` *`0:10`*
In the above command, frequency channels from 0 to 10 will be zapped. Note that `prepfold` refers to the original number of frequency subband when determining which channels it has been told to zap. So if your fits file has 256 frequency subbands and you want `prepfold` to show 64 subbands but zap the first and last subband, then you have to run `prepfold` with "-ignorechan 0:4,251:255".

(iii) Extract TOAs with get_TOAs.py as:
`get_TOAs.py -n 1 -s` *`N`* `-t` *`template`*`.pfd.bestprof file.pfd` Here, N is the number of TOAs per extract out of the full frequency bandwidth. You can choose the number of TOAs to extract using the same logic as before, but this time looking at the frequency versus time plot rather than the time versus frequency plot.

(iv) In your tim file, replace the usual per-subintegration TOAs of the epochs you just re-folded by the per-subband TOAs you just extracted. If your timing solution was already phase-connected, no need to add any JUMPs. The solution should be accurate enough that it will converge to the correct solution right away.

(v) In your par file, allow for DM to be fitted.

(vi) Run `tempo` as usual, and look at the best-fit value for the DM. If it's reasonably close to the original DM (within ∼10 units), then you are done!

Combining TOAs from different observatories

When extracting TOAs from follow-up data collected by a specific telescope, we use a template pulse profile that is specifically suitable for that telescope. Therefore, if a pulsar is followed-up by three different telescopes, there will be three different template pulse profiles used for TOA extraction.
Reasons for having telescope-specific templates include the different central observing frequency, bandwidth and bandpass (morphology of pulsar profiles vary with frequency), the observatory clock, terrestrial position, etc. All these effects are taken into account when extracting TOAs with typical pulsar timing software.
Also, when observing a source and recording the data, there are short delays between the times when the signal from a source hits the antenna, and when the data are digitized and recorded into a fits file. The exact duration of these delays is telescope-specific.
For these reasons, we can't have a fully-phase-connected timing solution when TOAs are extracted from data collected by more than one telescope. There are, however, advantages to following up a source with multiple facilities. Notably, it helps in constraining the source's position.
For example, consider a situation where you have two sets of TOAs: one set from the Arecibo Observatory (AO), and one set from the Jodrell Bank Observatory (JBO).

Step by step procedure:

(i) The first step to obtaining a timing solution from different observatory depends on the number of AO datafiles:

- If there are many AO datafiles:
  Start the phase-connection as usual without the JBO TOAs in the tim file. Once all AO TOAs are phase-connected (no more JUMPs), you can put in the JBO TOAs in the tim file.
- If there are only a few AO datafiles:
  Extract only per-subband TOAs for the AO data (for DM fitting, see above). For those pulsars, JBO normally has a phase-connected timing solution already. Therefore, no need to include JUMPs between AO TOAs: we'll assume that the JBO timing solution is accurate enough to produce the correct number of pulsar rotations in the AO set.

(ii) In the tim file, group TOAs together based on their observatory. Add one JUMP statement between the JBO TOAs and the AO TOAs to separate them.

The JBO solution is usually close enough to a AO (phase-connected) timing solution, so you *do not* need to add JUMPs between individual JBO TOAS. Also, you *do not* need to use the same TOA listing format for the two sets (see the different format in the `tempo` guided). The presto program get_TOAs.py normally prints TOAs in the Princeton format, while PALFA/JBO TOAs are usually provided by our collaborators in some other `tempo2` format. Your tim file should look similar to:

```
MODE 1
3 1643.563 57838.5402884845164 17.468 0.00000
3 1475.742 57838.5402884985002 12.211 0.00000
3 1279.444 57838.5402884697188 14.753 0.00000
3 1145.348 57838.5402884951390 23.186 0.00000


3 1643.563 57924.3053888176149 15.730 0.00000
3 1475.271 57924.3053888339855 12.700 0.00000
3 1279.444 57924.3053888032349 8.744 0.00000
3 1145.348 57924.3053888283218 22.198 0.00000


3 1643.563 57983.1461895392036 10.275 0.00000
3 1475.742 57983.1461895530232 6.865 0.00000
3 1279.477 57983.1461895238527 7.006 0.00000
3 1145.348 57983.1461895489670 16.278 0.00000


3 1672.746 58071.8914091466034 14.832 0.00000
3 1475.742 58071.8914091875807 13.111 0.00000
3 1282.722 58071.8914091381831 9.950 0.00000
3 1145.348 58071.8914091827040 27.899 0.00000


JUMP
1 2010+305 1520.000 57956.201860892928 0.000000 30.3 170722 8 0.000000
2 2010+305 1520.000 57983.216133793365 0.000000 33.2 170818 8 0.000000
3 2010+305 1520.000 58071.412114102342 0.000000 58.3 171114 8 0.000000
4 2010+305 1520.000 58115.320801086084 0.000000 19.8 171228 8 0.000000
5 2010+305 1520.000 58125.889652435573 0.000000 37.8 180107 8 0.000000
```

For JBO TOA format, each line lists the following fields: 1- TOA number, 2- pulsar name, 3- central observing frequency, 4- TOA, 5- DM correction (0 if already dispersed, which is always our case), 6- TOA uncertainty, 7- other correction on TOA decimal point (not important), 8- observatory code (8 for JBO), 9- extra phase rotation (if any).

(iii) Finalize the timing solution by running `tempo`. You may be able to fit more parameters now, especially if the second set extended the temporal coverage of the timing solution.

Subbanding and retrieving raw data from Arecibo

1. Log into puppimaster (ssh first to remote, then gpu@puppimaster, password = puppi_gpu)
2. Go to /data/puppi/P2789/subbanded
3. If a sub-directory does not already exist for your source, create one. Go to that directory.
4. List the datafile basename (full path) of the observation(s) you want to retrieve and subbanded in a file called basenames.txt. There should be only one basename per epoch. Make sure that you only list the basename (no "_0001.fits") in basenames.txt.
5. Always in the pulsar sub-directory, subband the data by running:
   `./subband_set.sh` *DM Nsub*
   DM is the pulsar's DM, and has to be an integer. *Nsub* is the number of subbands of the final (subbanded) datafile, and must be a power of 2. For data highly corrupted by RFI, I recommend producing more subbands (e.g. 256).
6. Retrieve the data with `rsync` from puppimaster to your machine.