Parkes Pulsar Timing Array

TEMPO2 examples

Authors:
George Hobbs

Updates:

May 24, 2014

Contents

1 Overview 2

2 Accessing data files 3

2.1 The Parkes pulsar data archive . . . 3

2.1.1 Virtual Observatory Access to the Data Archive . . . 3

2.2 Fermi . . . 3

2.3 The first Parkes Pulsar Timing Array data release . . . 3

2.4 The Demorest files . . . 3

2.5 The Kaspi files . . . 3

3 Initial processing of data files 4

4 Forming a template and obtaining arrival times 5

5 Basic pulsar timing procedures 6

6 Detailed data processing routines 7

7 Detailed template modelling 8

8 Detailed timing procedures 9

8.1 Publishing results . . . 9

1 Overview

The purpose of this document is to describe how to do various common tasks with the TEMPO2 software package.

1.1 Arrival time files

The TEMPO2 arrival time file format (which typically has the file extension `.tim`) has the following form:

```
FORMAT 1
directory/filename frequency mjd uncertainty site
```

For instance,

```
FORMAT 1
J1713+0747/10cm/w050611_125148.czFTp 3102.0 53532.55833332803828 0.388 PKS
J1713+0747/10cm/a050628_120144.czFTp 3100.25 53549.523611123047736 0.269 PKS
...
```

The FORMAT 1 statement defines this file to be a TEMPO2 format arrival time file. If a directory and filename is not available then the first column in this file can simply be an identifier for the particular observation. The frequency (in MHz), arrival time (MJD) and uncertainty (μs) can be given to any number of decimal places.

1.2 Obtaining some test data sets

Very few pulsar timing data sets are currently available online. Raw data files are available for the Parkes radio telescope from `http://datanet.csiro.au`, however these observations will need to be processed using, e.g., the PSRCHIVE software package to obtain pulsar arrival times. Parameter files can be obtained using the "Get Ephemeris" option on the ATNF pulsar catalogue: `http://www.atnf.csiro.au/research/pulsar/psrcat`.

A set of challenges in which pulsar data sets have been created that include a gravitational wave signal are available from `http://www.ipta4gw.org/?page_id=89`.

1.3 Useful command line arguments

The following command line arguments are available:

-allInfo	Lists out all information about which clock correction files are being used
-displayVersion	Display detailed CVS version numbers of every file used.
-epoch centre	Updates the pulse and astrometric parameters so that they refer to the centre of the data span
-fit X	Turn on fitting for parameter X (e.g., -fit F0)
-h	Provides some help information and lists the available plugin packages.
-nofit	Turns off all fitting
-reminder	Saves the command line to a file called T2command.input for future reference.

1.4 Using flags

Each arrival time line may have one or more "flags" that provide extra information for the user and the TEMPO2 algorithms. A few pre-defined flags have been set, but the user is free to define as many flags as required. For instance the user may wish to define the backend instrument, the receiver and a processing pipeline identifier:

```
FORMAT 1
J1713+0747/10cm/w050611_125148.czFTp 3102.0 53532.55833332803828 0.388 PKS -b WBCORR -rcvr 10CM -pid 1
J1713+0747/10cm/a050628_120144.czFTp 3100.25 53549.523611123047736 0.269 PKS -b DFB -rcvr 10CM -pid 1
...
```

The flags may be given in any order (and flags do not have to be set for all observations). Flags can be used for numerous purposes including:

Selecting data: command line arguments can be used to filter observations based on their flag values. For instance:

```
$ tempo2 -f mypar.par mytim.tim -pass "-b WBCORR DFB4"
```

will only select observations that have a "-b" flag set to "WBCORR" or to "DFB4".

```
$ tempo2 -f mypar.par mytim.tim -filter "-rcvr 10CM"
```

will reject all observations that have a "-rcvr" flag set to "10CM". (More complex filters are available using the select plugins or through a "select file".)

Dealing with phase jumps between instruments/observatories: It is common to include a phase jump between observations from different instruments or observatories. Sometimes these phase jumps are known and can be fixed. Usually, such jumps are measured as part of the TEMPO2 fit. The timing model can be updated to include

```
JUMP -rcvr 10CM 0 1
JUMP -rcvr MULTI 0 1
```

that will include a fit for the phase jump for data obtained with the 10CM receiver and for a phase jump for the MULTI receiver. These jumps are with respect to any observations that were not obtained using the 10CM and MULTI receivers (note: you cannot place a jump around all observations - the jumps d to be with respect to a set of arrival times),

Highlighting particular observations: The PLK plugin allows particular observations to be highlighted. This can be initiated from the command line:

```
$ tempo2 -gr plk -f mypar.par mytim.tim -colour -rcvr
```

will draw the observations with different "-rcvr" flags with different colours. Within the plugin, the ctr-i and 'H' options enable further highlighting of particular observations.

1.5 The plk plugin

The plk plugin is the most commonly used interface to TEMPO. It is run using:

```
$ tempo2 -gr plk -f psr.par psr.tim
```

This will graphically show the timing residuals and allow the user to turn on and off fitting and plot the residuals versus various parameters such as date, orbital phase, year, etc.

The plk plugin accepts the following command line arguments:

-colour X	Highlight all observations with a flag defined by X in different colours
-grdev X	Set the graphical device to X
-image	Make a plot suitable for inclusion in presentations
-lockx A B	Lock the x scale between A and B
-locky A B	Lock the y scale between A and B
-ms	Plot the timing residuals in units of ms
-nohead	Do not put a header on the plot (for publication quality)
-nophase	Do not display the timing residuals in units of phase
-ns	Plot the timing residuals in units of ns
-publish	Make a publication quality plot
-us	Plot the timing residuals in units of μs
-setup	Use a setup file (see below)
-showchisq	Show the reduced χ^2 value of the fit on the plot

When the plugin is running the user can use the following key-strokes (commonly used key-strokes are highlighted):

Key	**Description**
1	plot pre-fit residuals versus date
2	plot post-fit residuals verus date
3	plot pre-fit residuals versus orbital phase
4	plot post-fit residuals versus orbital phase
5	plot pre-fit residuals serially
6	plot post-fit residuals serially
7	plot pre-fit residuals versus day of year
8	plot post-fit residuals versus day of year
9	plot pre-fit residuals versus frequency
a	plot post-fit residuals verus frequency
!	plot pre-fit residuals versus TOA error
@	plot post-fit residuals versus TOA error
#	plot pre-fit residuals versus user values
$	plot post-fit residuals versus user values
%	plot pre-fit residuals versus frequency
ˆ	plot post-fit residuals versus frequency
&	plot pre-fit residuals versus elevation
	plot post-fit residuals versus elevation
(	plot pre-fit residuals versus rounded MJD
)	plot post-fit residuals versus rounded MJD
b	bin residuals within a certain time bin
B	place periodic marks on the x-axis
c	change the fitting parameters
C	run a unix command with filenames for the highlighted observations
ctrl-c	toggle between period epoch and centre for the reference epoch
d (or right mouse button)	delete the closest observation
D (or middle mouse button)	View profile of closest observation
ctrl-d	Delete all highlighted observations
e	multiply all arrival time uncertainties by a specified amount
ctrl-e	Highlight points more than 3 sigma from the mean
E	toggle plotting error bars
f	end of zoom region
g	change graphics device
G	change gridding on graphics device
h	give help information
H	highlight points with specific flag using symbols
i (or middle mouse button)	Identify the closest observation to the mouse cursor
ctrl-i	highlight points with specific flag using colours
I	indicate individual observations
j	draw a line between points
J	toggle plotting points
ctrl-J	get an output listing of the residuals in “Jodrell” format
l	list all the observations within the zoomed region
L	add label to plot
ctrl-l	add line to plot
m	use the mouse to measure the distance between two points
M	toggle removing the mean from the residuals

Key	Description
ctrl-m	toggle through various menu bars
N	highlight observation with a given filename
o	highlight all observations in the current zoom region
ctrl-o	get an output listing of the residuals in a simple format
p	change parameter values
P	write a new parameter file
q	quit
ctrl-p	toggle plotting versus pulse phase
r	re-load the original .par and .tim files
ctrl-r	select regions and write to regions.dat file
s	Start of zoom region
S	save a new arrival time file
ctrl-s	overwrite the input arrival time file
t	toggle displaying statistical information for the zoomed region
ctrl-t	set text size
u	unzoom
U	unselect highlighted points
v	view profiles for highlighted observations
V	define the user parmeter
w	toggle fitting using weights
ctrl-w	over-write the input parameter file
ctrl-x	define the x-axis
x	Re-do the fit
ctrl-y	define the y-axis
z	zoom on a region defined using the mouse
ctrl-z	get a listing of all the highlighted points
<	In zoom mode - include previous observation
>	In zoom mode - include next observation
+	Add positive phase jump
-	Add negative phase jump
Backspace	Remove all phase jumps
[	toggle plotting x-axis on log scale
]	toggle plotting y-axis on log scale

It is possible to make publication quality plots using the plk plugin. The following line may be sufficient:

```
$ tempo2 -gr plk -f psr.par psr.tim -publish -nohead
```

However, it is possible to control the plotting parameters in more detail using the -setup option. The -setup command line argument should provide a filename that contains the following setup information:

aspect X	Give the aspect ratio for the viewport
background X	Background colour
flag A B C D	Define all observations with a flag identifier of A and value of B to have colour C and point style D
fontsize X	Define the size of the font
fonttype X	PGPLOT font definition
freq A B C D	Define all observations with frequencies between A and B MHz to have colour C and point style D
line X	Line colour
linewidth X	Width of the lines
menu X	Which menu bar to plot
viewport_x0 X	The x coordinate for the start of the x-axis
viewport_x1 X	The x coordinate for the end of the x-axis
viewport_y0 X	The y coordinate for the start of the y-axis
viewport_y1 X	The y coordinate for the end of the y-axis

1.6 The splk and plotMany plugins

Tempo2 is designed to process multiple pulsars simultaneously. On the command line the data sets for each pulsar are simply included using the `-f` command-line argument:

```
$ tempo2 -f psr1.par psr1.tim -f psr2.par psr2.tim -f psr3.par psr3.tim ...
```

By default TEMPO2 can process 23 pulsars. If more pulsars are required then the `-npsr` command line argument must be used:

```
$ tempo2 -npsr 40 -nobs 1000 -f psr1.par psr1.tim -f psr2.par psr2.tim
     -f psr3.par psr3.tim ...
```

It is possible that the amount of available memory will limit the number of pulsars/observations that can be processed. The `-nobs` option can be used to specify the maximum number of observations to process (default = 10,000).

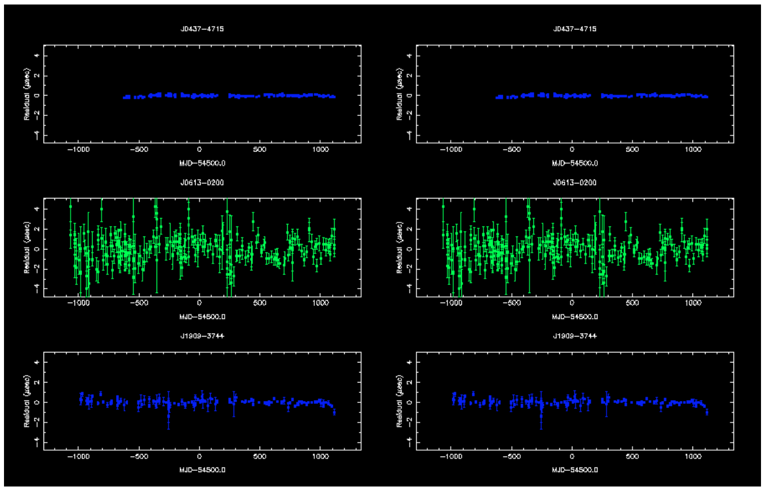
The plk plugin can only be used to plot a single pulsar. In order to compare timing residuals for multiple pulsars it is necessary to use either the splk or the plotMany plugins. Generally the splk plugin is useful for displaying a few pulsars whereas the plotMany plugin should be used for a large number of pulsars.

Using the splk plugin

The following example shows how the timing residuals for three pulsars can be displayed.

```
$ tempo2 -gr splk -f 0437.par 0437.tim -f 0613.par 0613.tim -f 1909.par 1909.tim
```

By default the splk figure shows the pre-fit residuals (left panels) and the post-fit residuals (right panels). Each panel is scaled individually. Pressing ‘x’ places all plots on the same x-axis and pressing ‘y’ places all plots on the same y-axis. ‘g’ allows the user to create a postscript file or other figure, ‘f’ changes the font size and ‘q’ quits the plugin.



Plotting the pre- and post-fit timing residuals for three pulsars using the splk plugin.

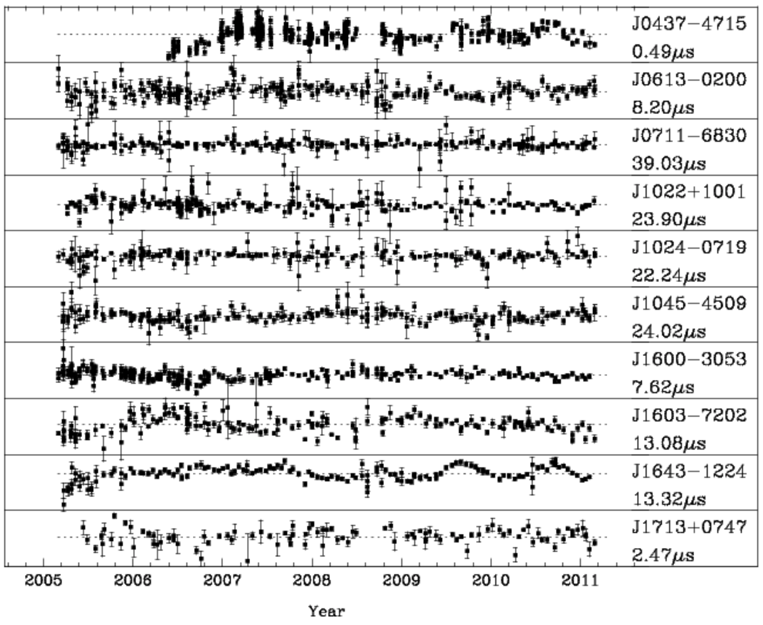
Using the plotMany plugin

The following example shows how the timing residuals for 10 pulsars can be displayed.

```
$ tempo2 -gr plotMany -f 0437.par 0437.tim -f 0613.par 0613.tim -f 0711.par 0711.tim -f 1022.par 1022.tim -f 1024.par 1024.tim -f 1600.par 1600.tim -f 1603.par 1603.tim -f 1643.par 1643.tim -f 1713.par 1713.tim -reverse -centremjd -1
```

If only the required parameter and arrival time files exist in the current directory then they will be automatically selected if no `-f` option is included on the command line. The following options are available:

-centremjd X	Set the central mjd value to X. Use -1 to use years instead of MJD
-fontsize X	Set the font size to X
-g X	Set the graphics device to X
-h	Display help information
-plotus	Scale the values in microseconds
-ptsize X	Set the point size to X
-ptstyle X	Set the point style to X
-reverse	Reverse the ordering of the pulsars in the display



Plotting the pre- and post-fit timing residuals for ten pulsars using the plotMany plugin.

1.7 The general2 plugin

It is often useful to obtain a text output of values of interest relating to each observation. Such values include the timing residuals, observing frequency, clock corrections, Shapiro delay etc. The general2 output plugin provides this functionality.

```
$ tempo2 -output general2 -s "<values>" -f mypar.par mytim.tim
```

The `-s` option allows the user to define some text and which values to display. For instance:

```
$ tempo2 -output general2 -s "Hello: arrival time = {sat}, frequency = {freq},
    Shapiro delay = {shapiro}\n" -f mypar.par mytim.tim
```

will, for each observation, list the site arrival time, frequency and Shapiro delay. Note that all TEMPO2-specific commands are listed within '{' and '}'. The following are available (common parameters are highlighted):

Value	Description
bat	barycentric arrival time (MJD)

Value	Description
bbat	barycentric arrival time corrected for binary motion (MJD)
binphase	binary phase
clock	Correction to TT (s)
clock0	The first clock correction (s)
clock1	The second clock correction (s)
clock2	The third clock correction (s)
clock3	The fourth clock correction (s)
clock4	The fifth clock correction (s)
clkchain	List the chain of clocks being used
del	= 1 if observation is deleted, 0 otherwise
dtCOE	Light travel time from the observing site to the centre of the Earth (s)
earth_ssb	Magnitude of the vector from the Earth to the SSB (lt-s)
earth_ssb1	X-component of the vector from the Earth to the SSB (lt-s)
earth_ssb2	Y-component of the vector from the Earth to the SSB (lt-s)
earth_ssb3	Z-component of the vector from the Earth to the SSB (lt-s)
earth_ssb4	X-component of the velocity vector from the Earth to the SSB (lt-s/s)
earth_ssb5	Y-component of the velocity vector from the Earth to the SSB (lt-s/s)
earth_ssb6	Z-component of the velocity vector from the Earth to the SSB (lt-s/s)
einstein	Einstein delay (s)
elev	Source elevation angle (degrees)
file	File name or observation descriptor
flags	List of flags for this observation
freq	Observing frequency (MHz)
freqCOE	Approximate frequency at centre of Earth (MHz)
freqSSB	Frequency at SSB (MHz)
ipm	Delay caused by the interplanetary medium (s)
ism	Delay caused by the ISM (s)
npulse	Pulse number
posPulsar	Three components of the unit vector from the Earth to the pulsar
posTel	Three components of the position of the observing site with respect to Earth's centre
pre	Pre-fit residual (s)
pre_phase	Pre-fit residual in phase
post	Post-fit residual (s)
post_phase	Post-fit residual in phase
roemer	Solar System Roemer delay (s)
sat	Site arrival time (MJD)
shapiro	Solar Shapiro delay (s)
shapiroJ	Shapiro delay caused by Jupiter
shapiroN	Shapiro delay caused by Neptune
shapiroU	Shapiro delay caused by Uranus
shapiroS	Shapiro delay caused by Saturn
shapiroV	Shapiro delay caused by Venus
siteVel0	X-component of the velocity of the observing site
siteVel1	Y-component of the velocity of the observing site
siteVel2	Z-component of the velocity of the observing site
solarangle	Angle between the line of sight to the pulsar and the Sun (degrees)
sun_earth1	X-component of the vector from the Sun to the Earth (lt-s)
sun_earth2	Y-component of the vector from the Sun to the Earth (lt-s)
sun_earth3	Z-component of the vector from the Sun to the Earth (lt-s)
sun_ssb1	X-component of the vector from the Sun to the SSB (lt-s)
sun_ssb2	Y-component of the vector from the Sun to the SSB (lt-s)
sun_ssb3	Z-component of the vector from the Sun to the SSB (lt-s)
t2tb	Correction to TCB including TT (s)
telSSB	Three components of the vector of the observing site from the SSB (lt-s)
telVel	Three components of the velocity of the observing site wrt to the SSB
tropo	Tropospheric corrections (s)
tt	Correction to Terrestrial Time (s)
tt2tb	Correction from TT to TCB (s)
Ttt	TOA corrected to TT (MJD)
ut1	UT1 correction (s)
velPulsar	Three components of the vector describing the pulsar's velocity
x	Barycentric arrival time minus the period epoch (MJD)
zenith	Zenith angle for observation (degrees)

1.8 Beyond the standard fitting

The standard TEMPO2 fitting algorithm is linear and can only be applied to a single pulsar. In many cases it is necessary to fit for parameters that are specific to a given pulsar (e.g., its pulse frequency) and simultaneously fit for parameters that are global to multiple pulsars (e.g., a gravitational wave signal or errors in the terrestrial time standard). It is often common that the parameters being fit are constrained. For instance, errors in a terrestrial time standard are covariant with errors in the pulsars' pulse frequencies. In such cases it is necessary to use an algorithm that can "constrain" the parameter values.

In order to change the fitting algorithm it is necessary to use a "fitfunc" plugin.TEMPO2 will then use the algorithm in the plugin instead of the standard TEMPO2 fitting algorithm. For instance:

```
$ tempo2 -gr splk -f psr1.par psr1.tim -f psr2.par psr2.tim -global global.par -fitfunc global
```

or

```
$ tempo2 -gr splk -f psr1.par psr1.tim -f psr2.par psr2.tim -global global.par -fitfunc globalDCM
```

In the first example, two pulsar data sets are loaded and fitted using the "global" plugin that allows for global parameters. In the second example, the "globalDCM" plugin is used. This allows for global parameters and takes account of red noise in the timing residuals.

Parameter files may also define which parameters need to be constrained. For instance:

```
CONSTRAIN TEL_DX
```

will ensure that the TEL_DX parameters (see §**??**) do not include a quadratic term that would be covariant with the pulsars' spin parameters.

1.9 Linked parameters

In some cases it is necessary to link two parameters together. For instance, SINI gives the sine of the orbital inclination angle whereas KIN is a parameter representing the inclination angle. If both parameters are to be used then it is necessary to ensure that TEMPO2 treats them as a single parameter. This is done in the parameter file. For instance, including

```
SINI  KIN
```

will link these parameters.

2 Analysing glitches

This section is based on algorithms developed and tested by G. Hobbs, R. Manchester, Y. Meng and S. Buchner. The tempo2 glitch model is based on the algorithm developed for tempo1.

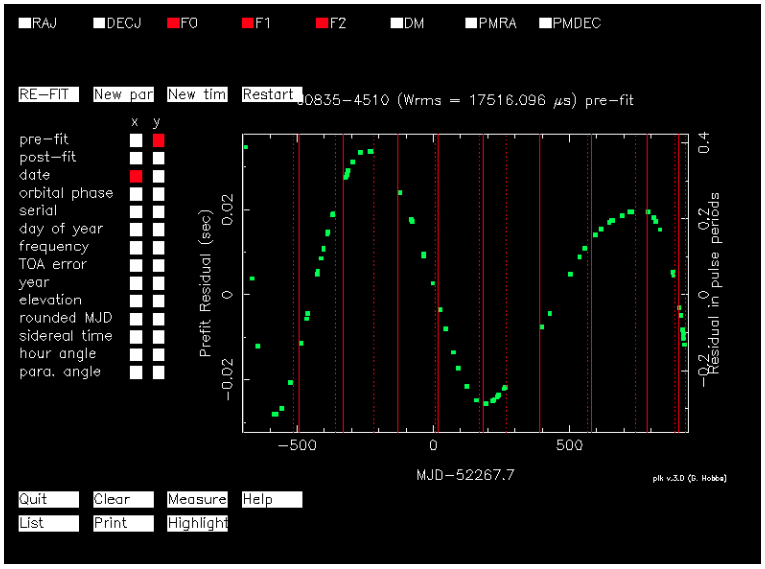
Tempo2 allows a simple glitch to be fitted as part of the timing model. This fit requires an initial estimate of the glitch epoch and any decay terms. The glitch plugin may be used to inspect the data for glitches and to obtain initial parameter estimates. In order to use the glitch plugin it is necessary to define a set of "regions". These regions define time intervals that contain at least two (but usually more) observations. The plugin will subsequently fit for the pulsar's pulse frequency (ν) and, if requested $\dot{\nu}$ for each region. The results are stored on disk (by default in a file called "results.dat") and can be displayed graphically.

The initial stage is to produce the "regions" file. This ascii file contains in column order (1) start MJD of region, (2) end MJD of region, (3) name of parameter file containing a phase-connected solution for the points within this region, (4) arrival time file containing these points. Note that it is common to have multiple parameter files that contain phase-connected solutions for different sections of data. This file can be created by the hand (or using simple scripts) or can be obtained from the plk plugin:

Creating a regions file

```
$ tempo2 -gr plk -f mypar.par mytim.tim
```

Pressing ctrl-r allows the user to enter regions using the mouse cursor. The right-mouse button completes the selection of regions and writes the file ("regions.dat") to disk. The user can then press ctrl-r again to add more regions. Currently the only way to remove a region is to edit the regions.dat file using a text browser. The regions.dat file is always appended to by the tempo2 software. If you wish to start a new set of regions then you should give the current file a different filename.



Creating a regions.dat file for Vela observations from the Parkes radio telescope; see Yu et al. (2012). The solid red vertical lines indicate the start of a given region and the vertical dotted line indicate the end of a particular region (note that regions may overlap each other).

If the data sets are long then creating a regions file may take significant time and effort. We recommend that the file is re-named and backed-up.

Running the glitch plugin

```
$ tempo2 -gr glitch -f mypar.par mytim.tim -t regions.dat
```

The following command line arguments are available:

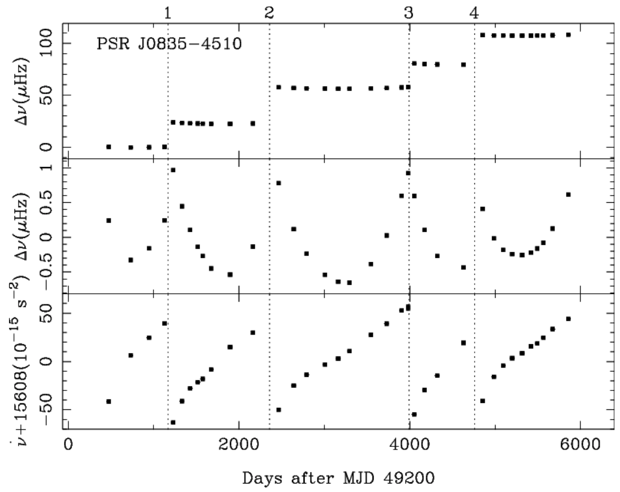
-combine	Combine multiple plots on a single page
-fitf1	Fit for ν and $\dot{\nu}$ (default is just to fit for ν)
-font x	Set font size to X
-foot x	Set the fraction of the page as a footer to X
-g x	Set graphics device to x
-gt x	Set glitch epoch to be MJD x (can use -gt multiple times for multiple glitches)
-h	Lists help information for the plugin
-head x	Sets fraction of the page as a header to be X
-i	Make the plots interactive (allow zooming etc.)
-internaltitle	Put the title for the plot within the figure
-loadResults x	Instead of re-doing the fitting, load the results from the file 'x' for immediate display
-ngltfntsize x	Sets the font size for numbering the glitch events
-numberglitch	Number the glitches
-offset x	Set time offset for x-scale axis
-removeExpected	see text below
-removeF2	see text below
-p x	Make a figure of plot type 'x' - see below (can be used multiple times)
-t x	Give the file name for the regions file (compulsory)
-title x	Set the title for the plot
-yscale a b c	Set the y-range for plot number 'a' to be from 'b' to 'c'

The following plot types are available (used with the -p option):

1	ν as a function of time
2	ν with a gradient removed and information provided on the number of observations within the corresponding region
3	$\dot{\nu}$ as a function of time
4	ν with the pre-glitch gradient removed
5	ν with the mean post-glitch values removed
6	$\dot{\nu}$ with mean removed
7	Post-fit timing residuals
8	ν with mean removed
9	As in plot 2, but without any data information

Yu et al. (2012) use commands such as:

```
$ tempo2 -gr glitch -f preglt.par vela.tim -t vela.regions -gt 50369.3 -gt 51559.3 -gt 53193 -gt 53959.9
-offset 49200 -title "PSR J0835-4510" -yscale 6 -70 70 -fitf1 -p 6 -p 5 -p 4 -combine -font 1.4 -head
0.04 -foot 0.12 -internaltitle -numberglitch -g plot.ps/ps -ngltfntsize 1.4
```



Running the glitch plugin on Vela observations from the Parkes radio telescope; see Yu et al. (2012).

The basic algorithm that the glitch plugin applies is to obtain specific regions from the regions.dat file. The values in the parameter file corresponding to that region are updated to refer to the centre of the region. All fits are turned off except for ν and, if requested using the -fitf1 option, $\dot{\nu}$. The observations within the region are fitted for these parameters and the resulting post-fit values stored. Note that fitting for ν requires at least two observations within the region. Fitting also for $\dot{\nu}$ requires at least three observations. If the -removeExpected parameter is set then the "expected" values of ν and $\dot{\nu}$ from the original parameter file (defined using the -f option on the command line) at the specified time is removed from the post-fit values. If the -removeF2 command line argument is set then the value of $\ddot{\nu}$ from the parameter file currently being used in the regions.dat file is removed from the post-fit values.

If a plot of the post-fit residuals is requested then this is obtained from the parameter and arrival time file that is specified on the command line using the -f option.

The plots can also be run in "interactive" mode by using the '-i' command line argument. After producing the plot, this parameter. Currently the only options are to zoom-in on the x-axis by pressing 'z' and drawing a box around the region of interest using the mouse and 'u' which un-zooms and returns to the default x-axis.

2.1 Including a glitch in the timing model

The glitch plugin makes no assumption about the structure of the residuals during or after a glitch; it simply measures ν and $\dot{\nu}$. TEMPO2 can include a simple glitch as part of the timing model. The phase $\phi(t)$ is modelled as:

$$\phi(t) = \phi_{g,0} + \nu_g(t - t_g) + \frac{1}{2}\dot{\nu_g}(t - t_g)^2 + \frac{1}{6}\ddot{\nu_g}(t - t_g)^3 + \nu_{g,d}\tau_d\left(1 - e^{-(t-t_g)/\tau_d}\right) \tag{1}$$

for an observation time greater than the glitch epoch, t_g (i.e., $t > t_g$) and

$$\phi(t) = 0 \tag{2}$$

otherwise. The terms in this equation and their corresponding parameters in the parameter file are given in the following table:

Symbol	Parameter	Description
$\phi_{g,0}$	GLPH_X	Phase increment at the time of the glitch
ν_g	GLF0_X	Glitch permanent pulse frequency increment (Hz)
$\dot{\nu_g}$	GLF1_X	Glitch permanent frequency derivative increment (s^{-2})
$\ddot{\nu_g}$	GLF2_X	Glitch permanent second frequency derivative increment (s^{-3})
$\nu_{g,d}$	GLF0D_X	Glitch frequency increment as part of decaying term (Hz)
τ_d	GLTD_X	Glitch decay time constant (d)

Note that the τ_d and $\nu_{g,d}$ terms are nonlinear. As the default TEMPO2 fitting routine only applies to linear functions the user requires a "good" initial guess of these parameters for the fit to converge. Usually a small section of data around a specific glitch (typically a few hundred days before and after the glitch) is obtained. Initial estimates of t_g, ν_g and $\dot{\nu_g}$ are obtained from the glitch plugin and included in the parameter file. Fitting is turned on for ν, $\dot{\nu}$, ν_g, $\dot{\nu_g}$ and $\phi_{g,0}$ and the resulting post-fit parameters stored. Choices are subsequently made about the requirement for $\ddot{\nu_g}$, τ_d and $\nu_{g,d}$ terms from the post-fit timing residuals and the plots made using the glitch plugin. If these are required then good initial estimates are obtained from the glitch plugin plots and fitted as part of the timing model. If significant red noise is present then the Cholesky algorithm (described later) can be used to obtain reasonable error estimates for the fitted parameters.

After fitting for the glitch parameters TEMPO2 will use the measurement of $\phi_{g,0}$ to estimate the glitch epoch. This is obtained using an iterative procedure that determines the change in glitch epoch that would have led to $\phi_{g,0} = 0$. This usually has two solutions; both are displayed in the output.

2.2 More details

Details of a large number of glitches that have been processed using TEMPO2 are given in Yu et al. (2012; in preparation).

2.3 Known issues

- Problem with the definition in the original tempo2 paper

Note: some of the work developed to study the gravitational wave memory effect is applicable for glitch studies.

2.4 References

3 Pulsar timing noise

This section is based on algorithms developed and tested mainly by G. Hobbs, W. Coles, X. Deng and R. Shannon. More information is available from Coles et al. (2012) and Deng et al. (submitted).

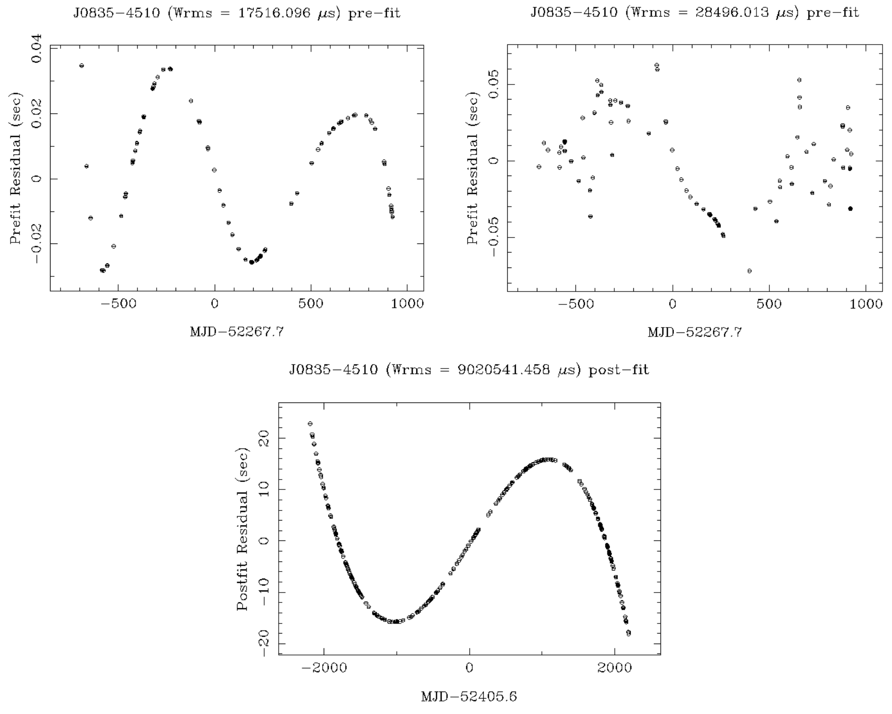
3.1 Pulse numbering and tracking

Pulsar timing noise can often mean that the pulse phase wanders by more than half a pulse period. This leads to a phase wrap and the pulsar timing model is not phase-connected. However, it is often possible to obtain a shorter region of data that does not include phase wraps. The exact pulse numbers for each observation (with respect to the first observation in the file) can then be written out in a new arrival time file by pressing 'n' in the plk plugin.

Example of setting pulse numbers for each observation

```
$ tempo2 -gr plk -f mypar.par mytim.tim
```

In this example we obtain observation of the Vela pulsar from the Parkes telescope and have fitted for ν, $\dot{\nu}$ and $\ddot{\nu}$ to obtain a phase connected timing solution. In the right hand figure we show the resulting timing residuals if we set $\ddot{\nu} = 0$. These residuals are not phase connected.



Timing residuals for the Vela pulsar with (left) and without (right) including $\ddot{\nu}$. In the bottom panel the timing residuals are displayed using knowledge of the exact pulse numbers for each observation (and setting $\ddot{\nu} = 0$. Note that the residuals traverse more than 200x the pulse period.

If we return to the initial parameter file (that contains a $\ddot{\nu}$), run plk and then press 'n' we get a new arrival time file ("withpn.tim") that contains a flag ("-pn") for each observation that gives the corresponding pulse number. To make use of these pulse numbers we need to include TRACK -2 in the parameter file. The resulting post-fit timing residuals (with $\ddot{\nu} = 0$) is shown in the bottom panel of the Figure.

3.2 Spectral estimation of pulsar timing noise

Numerous algorithms within TEMPO2 require a simple model of the power spectrum of the timing noise signal in the timing residuals. Such a power spectrum is intrinsically interesting as it allows searches for e.g., unknown binary companions. Details of the algorithm implemented are given in Coles et al. (2012). The standard method is to use the spectralModel plugin, but as shown below an automatic spectral estimation plugin is also now available.

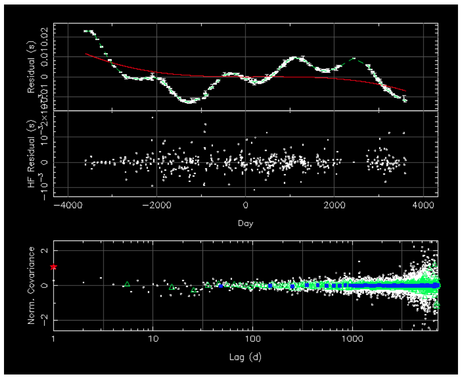
Using the spectralModel plugin

For this example we make use of observations of PSR B1540−06 obtained at the Jodrell Bank Observatory.

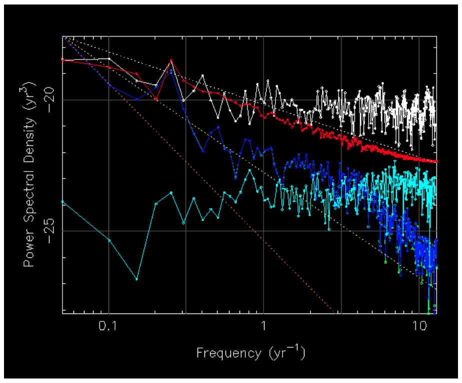
```
$ tempo2 -gr spectralModel -f 1540.par 1540.tim -nofit -fit f0 -fit f1
```

The plugin leads to the following graphical displays:

Step 1: The first screen that appears when the plugin is run is shown below. In the top panel the original post-fit timing residuals are shown in white and an unweighted cubic fit to the residuals shown in red. The cubic fit is used to obtain a smooth model of the low-frequency component of the timing residuals (see algorithm details below). The smooth model (using a default smoothing time of 20 days) is shown as the green dashed line. The middle panel shows, without uncertainties, the differences between the smooth model and the actual timing residuals (known as the "high-frequency residuals"; these are the high-pass filtered residuals). In this case it is clear that the smooth model does a good job of describing the low frequency structures, but there are still some remaining higher-frequency features. The covariance function of these high-frequency residuals are shown in the bottom panel. The user then has the option of changing the smoothing timescale or continuing. For this example the smoothing time was not changed from the default 20 days.

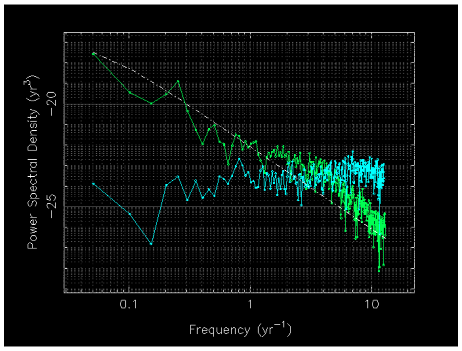


Step 2: The next plot shows power spectral density plots for the original residuals (white), high frequency residuals (light blue), the smooth low-frequency model interpolated onto a 14d regularly sampled grid (red) and the same interpolated data after first (green) and second (dark blue) pre-whitening and subsequent post-darkening. The white, yellow and orange dotted lines represent slopes of -2, -4 and -6 respectively. The user is asked to select the amount of pre-whitening required. In this case we have chosen to use first order prewhitening.

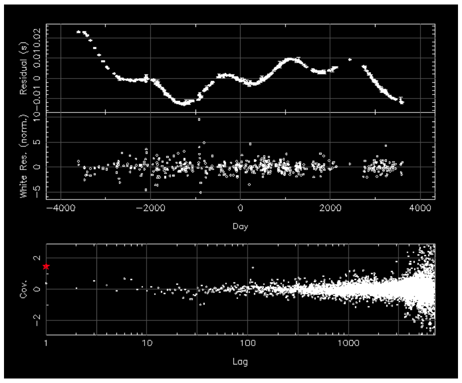


Step 3: A simple model for the low-frequency noise must be entered. This model is defined by

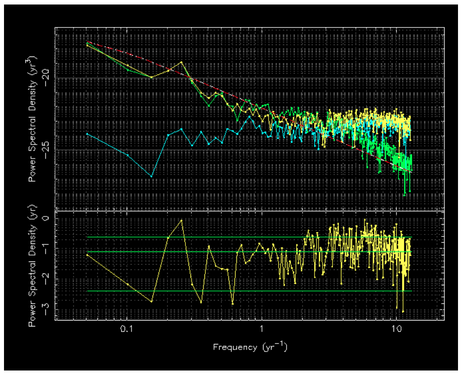
$$P(f) = \frac{A}{\left[1 + \left(\frac{f}{f_c}\right)^{\alpha/2}\right]^2} \tag{3}$$



Step 4: This figure is for display only. The data are whitened using the Cholesky algorithm and the spectral model obtained in Step 3. The top panel shows the original residuals, the central panel the whitened residuals and the bottom panel the covariance of the whitened residuals.



Step 5: The original spectrum of the high-frequency noise is shown in blue, The original spectrum of the red noise in green. A new spectrum of both noise components using the Cholesky algorithm is now overlaid in yellow. The bottom panel shows the spectrum of the whitened residuals. For a successful model of the timing noise this spectrum should be flat and lie within the bounds given using horizontal green lines.



3.2.1 Automatic spectral estimation

This work is still "in progress". Please contact R. Shannon before making use of the following algorithms.

The spectralModel plugin is slow and required manual input. It is sometimes necessary to obtain a reasonable spectral model automatically. R. Shannon has developed a set of routines to do this. These are included in the autoSpectralFit plugin:

```
$ tempo2 -gr autoSpectralFit -f 1540.par 1540.tim
```

3.3 Interpolation and prediction

This work is still "in progress". Please contact X. Deng, W. Coles or G. Hobbs before making use of the following algorithms. A description of this work has been submitted to MNRAS.

The spectral model can be used to interpolate the pulsar timing residuals or to predict future residuals. In order to carry out this interpolation/prediction the interpolate plugin can be used. The result can be viewed using the plotInterp plugin.

Using the interpolate and plotInterp plugins

For this example we attempt to interpolate the PSR B1540−06 timing residuals that were used in the previous example. The first observation is at MJD 46236 and the last at 53443. The parameters of the spectral model can be obtained from the 1540-06.model file and are included using the `-a`, `-fc` and `-alpha` command line arguments.

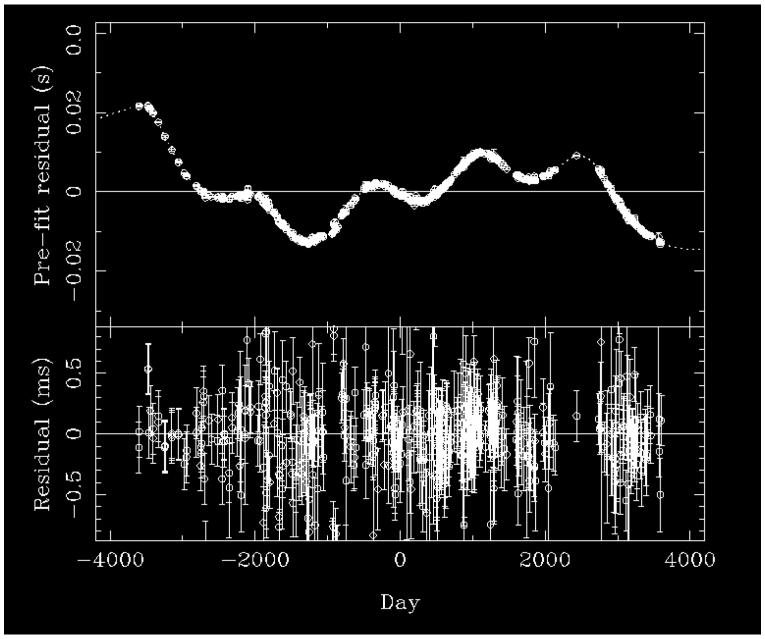
$ tempo2 -gr interpolate -f 1540.par 1540.tim -x1 45830 -x2 54043 -dx 100 -a 2.8015e-18 -fc 0.05059 -alpha 3.921

This produces a new parameter file (the file name is set by the user) that contains a set of IFUNC parameters which model the timing noise (and therefore can be used to whiten the data). The IFUNC gridding starts at `-x1` and finishes at `-x2` in steps of `-dx`. If `-x1` is before the start of the observations then the algorithm will post-predict the timing residuals. Similarly is `-x2` is after the most recent observation then the timing residuals will be predicted.

The interpolation/prediction can be viewed using the plotInterp plugin:

$ tempo2 -gr plotInterp -f interp.par 1540.tim -xextend 600

The result is shown in the Figure below.



The top panel shows the original timing residuals with the interpolated function (dashed line). The lower panel shows the timing residuals whitened using the interpolation function.

The plotInterp plugin accepts the following command line arguments:

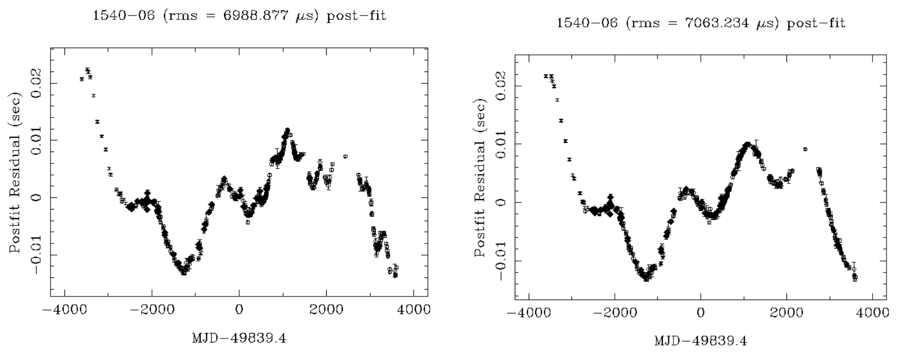
-label X	Label the plot with X
-xextend X	Extend the x-axis by X days before the first observation and after the last observation
-xextendLate X	Extend the x-axis by X days after the last observation
-xextendEarly X	Extend the x-axis by X days before the first observation
-resminy X	Select the minimum y-value for the y-scale of the white residuals
-resmaxy X	Select the maximum y-value for the y-scale of the white residuals
-miny X	Select the minimum y-value for the y-scale for the interpolation panel
-maxy X	Select the maximum y-value for the y-scale for the interpolation panel
-nopt	Do not draw points
-solidLine	Draw a solid line for the interpolated function

3.4 Fitting in the presence of red noise

With the data covariance function it is possible to generalise the TEMPO2 least-squares-fit to whiten both the data and the model during the fitting procedure. The data covariance function is specified using the `-dcf` option.

```
$ tempo2 -gr plk -f 1540.par 1540.tim -dcf covarFunc.dat_1540-06
```

In the left-hand panel of the following figure the PSR B1540−06 data set has been fitted for the pulsar's position and proper motion without using the data covariance function. The resulting post-fit residuals show the effect of an incorrectly determined annual term. In the right-hand panel the data covariance function is included in the fit and the position and proper motion (and their corresponding errors) are correctly determined.



Fitting for the position and proper motion for PSR B1540−06 without accounting for the low-frequency noise (left panel) and using the data covariance function (right panel).

4 Dealing with dispersion measure variations

Many attempts have been made to measure and correct dispersion measure variations within TEMPO2. Initial work was undertaken by X. You and W. Coles and was published in You et al. (2007). The work described below has been developed by M. Keith and W. Coles. This work is still "in progress". Please contact M. Keith before making use of the following algorithms.

Timing residuals will be caused by variations in a pulsar's dispersion measure (DM). Methods to correct for such variations rely on observations having been obtained at multiple observing frequencies. The method described below is from Keith et al. (2012; in preparation) which ensures that no low-frequency, non-DM signal is removed whilst correcting for the DM variations.

This new method requires that the parameter file has a nominal DM and also a set of grid points where offsets to this nominal DM are defined. For instance,

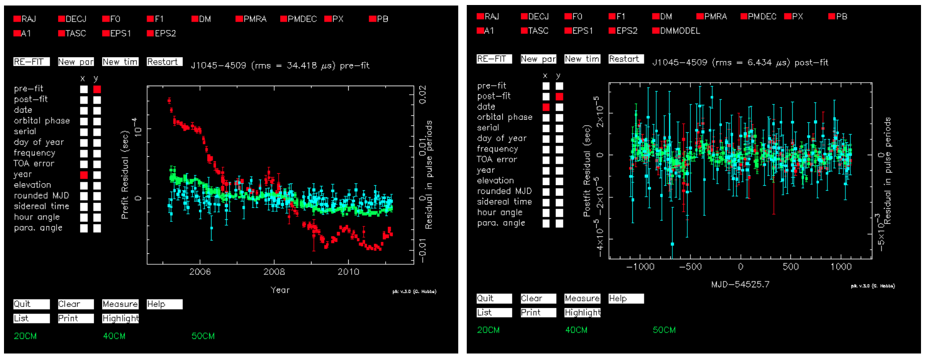
```
DM 58.14375
DMMODEL DM 1
DMOFF 53432 0 0
DMOFF 53502 0 0
DMOFF 53572 0 0
...
CONSTRAIN DMMODEL
```

Here, the first DM parameter is the nominal DM, the `DMMODEL DM 1` links the parameter DMMODEL to DM and the '1' implies that this parameter should be fitted for and the `DMOFF` parameters define the set of grid points (the first value is the MJD of the grid point, the second value is the DM offset and the third value its uncertainty).

In order to deal with DM variations in the presence of red noise, the algorithm fits simultaneously for the DM variations and, with the same grid points, a "common-mode" signal that does not have any observing-frequency-dependence. Currently, that "common-mode" signal is subsequently "thrown away". In order for the "common-mode" signal not to be covariant with the pulsar pulse parameters it is necessary to use a constrained fit; hence the `CONSTRAIN DMMODEL` parameter.

Measuring and removing dispersion measure variations

For this example we make use of the PPTA observations of PSR J1045−4509. These observations are in three bands (10 CM, 20 CM and 40 CM). The 10 and 40 CM bands are usually simultaneous with the 20 CM observations normally taken within a few days. The first observation was taken at MJD 53431 and the most recent at 55619. The timing residuals obtained without DM correction and with a nominal DM of 58.14375 are shown in the left-hand panel of the Figure below.



RAJ DECJ F0 F1 DM PMRA PMDEC PX PB A1 TASC EPS1 EPS2
RE-FIT New par New tim Restart
J1045−4509 (rms = 34.418 μs) pre-fit
J1045−4509 (rms = 6.434 μs) post-fit
pre-fit post-fit date orbital phase serial day of year frequency TOA error year elevation rounded MJD sidereal time hour angle para. angle
Prefit Residual (sec)
Postfit Residual (sec)
Residual in pulse periods
Year
MJD−54525.7
Quit Clear Measure Help List Print Highlight
20CM 40CM 50CM

Timing residuals for the PSR J1045−4509 from the PPTA data set before (left) and after (right) correcting for dispersion measure variations. 10 CM residuals are shown in blue, 20 CM in green and 50 CM in red.

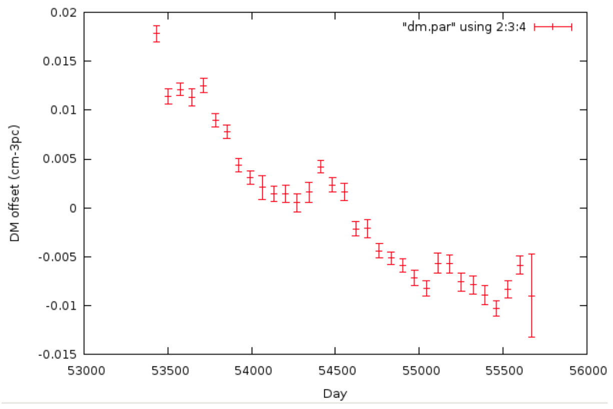
In the parameter file we then add

```
DMMODEL DM 1
DMOFF 53432 0 0
DMOFF 53502 0 0
DMOFF 53572 0 0
...
DMOFF 55672 0 0
CONSTRAIN DMMODEL
```

Then we can simply run TEMPO2 again with this updated par file:

```
$ tempo2 -gr plk -f mypar.par mytim.tim
```

The resulting DM-corrected timing residuals are shown in the right-hand panel of the figure above. There is currently no means within TEMPO2 to plot or analyse the actual DM measurements. However the plk plugin can be used to write out a new parameter file (e.g., dm.par). The DM offsets and their uncertainties can easily be extracted from this file. For instance, the figure below shows the DM offsets plotted in gnuplot.



Dispersion measure variations for PSR J1045−4509.

4.1 Known issues

- Currently up to 500 DM grid points can be used.
- The common-mode signal cannot be extracted from TEMPO2.

5 Gravitational wave signals

This section is based on algorithms developed and tested mainly by G. Hobbs, W. Coles, R. Shannon, W. Jingbo, S. Burke-Spolaor, M. Keith, D. Yardley, F. Jenet and K. J. Lee. More details are available in Hobbs et al. (2010).

TEMPO2 has the ability to simulate the induced timing residuals caused by GW signals and to detect/limit such signals in actual data sets. Here we divide the different tools into those relevant for the gravitational wave background, continuous wave sources, evolving sources and the gravitational wave memory effect.

Note: some of the software to simulate the induced timing residuals caused by GW signals will be superceded in the near future by the new methods to simulate data sets (see Section XX).

5.1 Gravitational wave background

5.1.1 The tempo2 background

The default simulation software within TEMPO2 assumes that the gravitational wave background has a characteristic strain amplitude that follows a power law. The induced timing residuals have the following power spectral density:

$$P(f) = \frac{A^2}{12\pi^2}\left(\frac{f}{f_{1\mathrm{yr}}}\right)^{2\alpha_g-3} \tag{4}$$

where A is the gravitational wave background amplitude and $f_{1\mathrm{yr}} = 1/(1\mathrm{yr})$. The induced timing residuals and corresponding times-of-arrival can be obtained using the GWBKGRD plugin.

Running the GWbkgrd plugin

```
$ tempo2 -gr GWbkgrd -f psr1.par psr1.tim -f psr2.par psr2.tim -gwamp 1e-15 -alpha -0.6666 -ngw
1000 -dist 1.23 -dist 2.34
```

The following command line arguments are available:

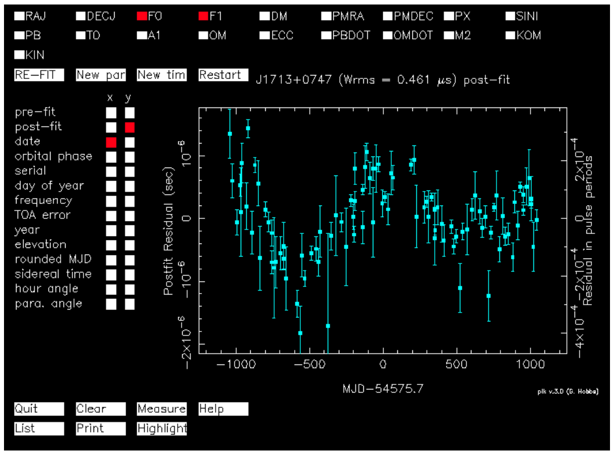
-alpha x	The spectral exponent of the characteristic strain, α_g. Note that the spectral exponent of the power spectrum of the timing residuals will be $2\alpha_g - 3$.
-dist x	Give the distance for each pulsar in kpc (use a -dist option for each pulsar)
-f x y	List the .par and .tim files for each pulsar
-fhi x	Set the highest gravitational wave frequency to simulate (Hz)
-flo x	Set the lowest gravitational wave frequency to simulate (Hz)
-gwamp x	The dimensionless gravitational wave background amplitude
-h	Provide help information
-ngw x	Number of gravitational wave sources to simulate
-plot	See below

The standard output of the plugin is to produce a new file for each pulsar with name "psrname.gwsim.tim" which is a new arrival time file that contains the effect of the gravitational wave background.

The interactive plot mode (use -plot on the command line) allows the user to plot the plot the pulsar positions, gravitational wave source positions, pre- and post-fit timing residuals etc. The following key-presses are defined:

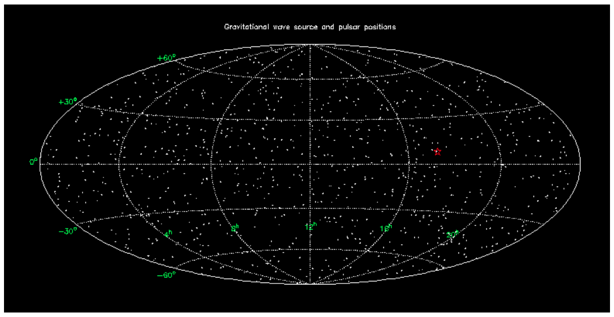
1	plot the position of the pulsars and the individual simulated gravitational wave sources
2	plot the spectrum of the gravitational wave source characteristic strain
3	plot the induced timing residuals caused by the gravitational wave background for selected pulsar as a function of time
4	as in plot '3', but fit and remove a quadratic polynomial.
5	plot the pre-fit timing residuals
6	plot the post-fit timing residuals
p	select pulsar number to display (starting at zero)
q	quit
s	save residuals and site arrival times.

The following figure shows the post-fit timing residuals after simulating a gravitational wave background with $A_g = 10^{-14}$ and $\alpha = -2/3$ for the Parkes Pulsar Timing Array observations of PSR J1713+0747.



Post-fit timing residuals obtained by added a simulated gravitational wave background to PPTA observations of PSR J1713+0747. This figure was obtained using the GWbkgrd plugin to produce a new arrival time file and then viewing the post-fit residuals using the plk plugin.

The corresponding sky position of each gravitational wave source (white dots) and the pulsar (red star) are shown in the next Figure.



The sky position of the pulsar position (red star) and the simulated gravitational wave sources (white dots) obtained using the GWbkgrd plugin.

Note that the GWbkgrd plugin simply adds the effect of a gravitational wave background to the site arrival times given on the command line. These site arrival times can have any sampling. It is therefore easy to produce "realistic" data sets that include the effects of gravitational waves, by first using the simulation software described later (§**??**) and subsequently using GWbkgrd plugin (or simply applying the plugin to an actual data set).

When simulating each gravitational wave source the GWbkgrd plugin first chooses a random sky position (α, δ):

$$\alpha = 2\pi\Psi \tag{5}$$
$$\delta = \cos^{-1}[2(\Psi - 0.5)] \tag{6}$$

where Ψ is a random deviate from 0 to 1 (a different value is chosen for each usage). The angular frequency of the gravitational wave is

$$\omega = 2\pi e^{\log f_{lo} + \Psi \log(f_{hi}/f_{lo})} \tag{7}$$

where f_{lo} and f_{hi} are the frequencies of the lowest and highest gravitational wave frequencies to be simulated (they can be set on the command line or take default values of $f_{lo} = 0.01/T_{\rm span}$ and $f_{hi} = 1/[1d]$).

The gravitational wave amplitudes are set as

$$A_+ = A_g \left(\frac{\omega_g}{2\pi}\right)^{\alpha_g} \sqrt{\frac{2\pi(f_{hi} - f_{lo})}{N_{gw}\omega_g}}\zeta \tag{8}$$

where ζ is a Gaussian random deviate. An identical expression is used to calculate A_x (but a different Gaussian random number chosen).

The gravitational wave phase is set by

$$\phi_g = 2\pi\Psi. \tag{9}$$

5.1.2 Realistic source lists

5.1.3 Limiting the amplitude of the gravitational wave background amplitude

5.1.4 Detecting the gravitational wave background

This work is still “in progress”. Please contact G. Hobbs or W. Coles before making use of the following algorithms. These algorithms are based on those initially developed by D. Yardley and published in Yardley et al. (2011).

A few TEMPO2-based methods now can be used to search for gravitational wave signals. Here we describe the detectGWB plugin that uses a “frequentist” approach to the problem. The plugin obtains a smooth model of the timing noise for each pulsar, obtains the cross power spectrum for a pair of pulsars and determines the covariance between the two pulsar data sets. These can be plotted against the angle between the pulsar pair and the Hellings & Downs (1983) expected functional form can be fitted to the resulting covariances. This plugin may significantly change in the near future as more consideration is given to the algorithms implemented. For now, this plugin must be considered as “work in progress”.

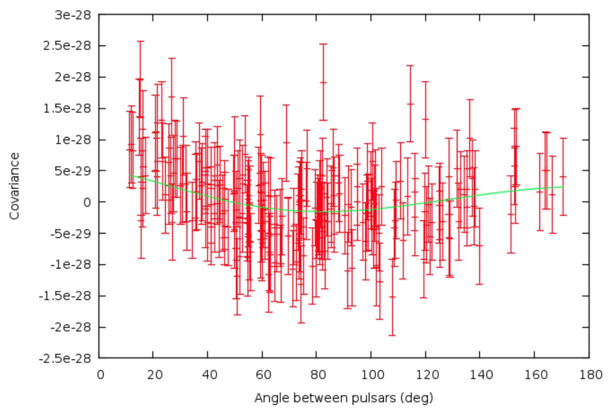
The plugin requires a guessed amplitude for the gravitational wave background (given by -gwamp) and spectral exponent for the power spectrum of the timing residuals (-alpha_res). A Cholesky model is required for each pulsar data set in order to obtain spectra for each pulsar (and subsequently to form the cross-spectra).

Searching for the gravitational wave background using the detectGWB plugin

We first simulate regularly sampled, data sets for the PPTA sample of pulsars using the fake plugin. Each pulsar is sampled every 14 days from MJD 51000 to 52826.25. Arrival time uncertainties are assumed to be 100 ns. We then use the GWbkgrd plugin to add in a gravitational wave background with an amplitude of $A = 10^{-14}$ and exponent of $\alpha = -2/3$. For each pulsar data set we need a Cholesky model of the pulsar timing noise. We do this for each pulsar using the autoSpectralFit plugin. Finally we run the detectGWB plugin as follows:

```
$ tempo2 -gr detectGWB -gwamp 1e-14 -t1 51000 -t2 52700 -dt 100 -f psr1.par psr1.gwsim.tim -f psr2.par psr2.gwsim.tim -f psr3.par psr3.gwsim.tim ...
```

This produces an output file, hdcurve.dat, that we plot in the Figure below.



The Hellings-and-Downs curve recovered from simulated data using the detectGWB plugin. The green line is the expected functional form.

5.2 Continuous wave sources

This work is still "in progress". Please contact G. Hobbs or S. Burke-Spolaor before making use of the following algorithms.

Continuous wave sources are individual sources of gravitational waves that do not change significantly over the observing data span. These sources can be included in pulsar timing model. They are defined by a sky position (α,δ), the amplitude in the + polarisation state, A_+, the amplitude in the x polarisation state, A_x and an angular frequency, ω_g. In the parameter file these can be included as:

```
GW_SINGLE X
GW_POSITION X Y
GW_APLUS X_R X_I
GW_ACROSS X_R X_I
```

where the `GW_SINGLE` parameter gives the angular frequency of the gravitational wave. The position is given in radians. It is also necessary to give:

```
GW_EPOCH X
```

which provides a nominal epoch, t_0, when calculating the gravitational wave signal. It is also necessary to give an estimate for each pulsar's distance. For each pulsar, the following should be included in the parameter file:

```
GW_PSR_DIST X
```

(where the distance is given in kpc). Note, if this is not set then the "pulsar term"-component of the gravitational wave signal will not be calculated.

It is possible to fit for `GW_APLUS` and `GW_ACROSS` as part of the pulsar timing fit, but it is not possible to fit for the angular frequency or position. Note that both `GW_APLUS` and `GW_ACROSS` take two values corresponding to the real and imaginary parts. Fitting for these parameters therefore gives four parameters that can be used to determine the gravitational wave phase.

Simulating the effect of a continuous gravitational wave source

Here we demonstrate the induced timing residuals caused by a continuous gravitational wave source for three pulsars. We first create a global.par file that contains a source that is completely polarised in A_+:

```
GW_SINGLE 5e-8
GW_POSITION 0.4 0.6
GW_APLUS 1e-12 0
GW_ACROSS 0 0
GW_EPOCH 52000
```

For the first pulsar we add into psr1.par:

```
GW_PSR_DIST 0.85
```

and for psr2.par and psr3.par similar pulsar distances. Then we simulate regularly sampled data spanning from MJD 51000 to MJD 54000 with rms white noise of 100 ns.

```
$ tempo2 -gr fake -f psr1.par psr2.par psr3.par -global global.par -nobsd 1 -ndobs 14 -randha y -ha 8 -start 51000 -end 54000 -rms 1e-4
```

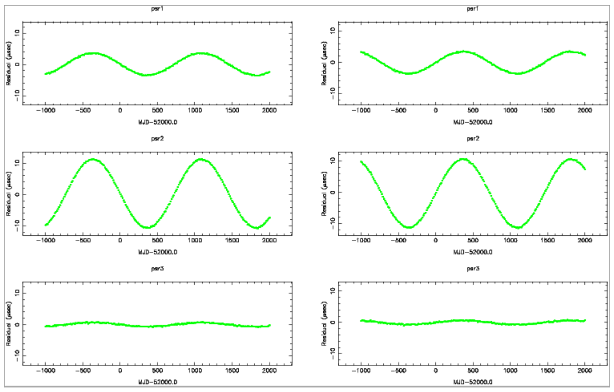
We can then produce a new global_new.par file that contains:

```
GW_SINGLE 5e-8  2
GW_POSITION 0.4 0.6
GW_APLUS 0 0
GW_ACROSS 0 0
GW_EPOCH 52000
```

(Note that the `GW_SINGLE` parameter now includes a '2' to indicate a global fit and that the `GW_APLUS` amplitudes are now set to 0.).

```
$ tempo2 -gr splk -f psr1.par psr1.simulate -f psr2.par psr2.simulate -f psr3.par psr3.simulate -global global_new.par -fitfunc global
```

The resulting plot is shown in the figure below. Note that fitting for the amplitude of the gravitational wave source does not significantly improve the residuals. This is because the pulsar term (i.e., from the pulsar distances) is used within TEMPO2 to simulate the induced timing residuals, but is assumed unknown in the general case and therefore not included during the fitting procedure.



Simulating timing residuals caused by a continuous gravitational wave source for three pulsars. The left panel shows the pre-fit residuals and the right panel after attempting to fit for the source.

5.3 Evolving sources

5.4 The gravitational wave memory effect

This work is still “in progress”. Please contact J. Wang or G. Hobbs before making use of the following algorithms.

The gravitational wave memory (GWM) effect leads to a step change in a pulsar’s pulse period as the gravitational wave passes the Earth. This exhibits as a glitch-event in the timing residuals of all pulsars. The size of the glitch depends upon the pulsar-Earth-gravitational wave source angle. Such events can be simulated within TEMPO2 using the following parameters:

GWM_AMP x	The amplitude of the GWM
GWM_POSITION x y	The right ascension and declination of the GWM source (in radians)
GWM_EPOCH x	The epoch (MJD) of the GWM passing the Earth
GWM_PHI	The GWM polarisation angle

It is posible to fit for the amplitude of the GWM_AMP, but currently it is not possible to fit for the position, epoch or polarisation angle.

Simulating observations that contain the gravitational wave memory effect

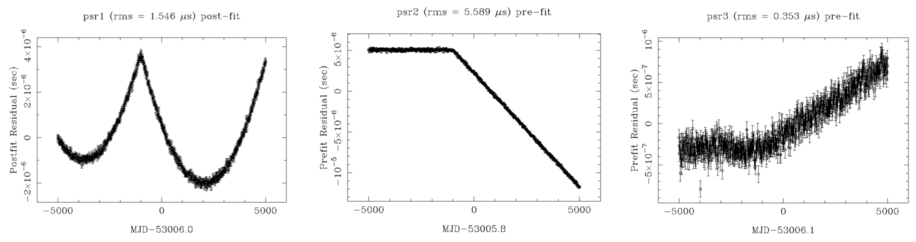
In order to simulate a GWM effect for multiple pulsars we create a global.par file that contains:

```
GWM_AMP 1e-13
GWM_POSITION 0.2 -0.6
GWM_EPOCH 52000
GWM_PHI 0.3
```

we then simulate (using the fake plugin) three pulsar data sets using this global.par file, e.g.,:

```
$ tempo2 -gr fake -f psr1.par -nobsd 1 -ndobs 14 -ha 8 -randha y -start 48000 -end 58000 -rms 1e-4
-global global.par
```

The resulting arrival times can subsequently be processed as normal using, e.g., the plk plugin.



The effect of a gravitational wave memory event on the timing residuals of three pulsars. The timing residuals are shown post-fit (after fitting for ν and $\dot{\nu}$) for the first pulsar and pre-fit for the other two pulsars.

5.5 Known issues

- If more than 20 pulsars are to be simulated simultaneously then the -npsr command-line option needs to be set.

6 Searching for errors in terrestrial time standards

This work is still "in progress". Please contact G. Hobbs before making use of the following algorithms. A paper describing this work will soon be submitted to MNRAS: Hobbs et al. (in preparation).

Errors in a realisation of a terrestrial time standard (TT) will lead to timing residuals that have the same functional form in all pulsars. The realisation of terrestrial time used is defined in each pulsar's parameter file (and for this work must be the same for each pulsar). Using the realisation of TT from the International atomic time:

```
CLK TT(TAI)
```

Using the best terrestrial time standard:

```
CLK TT(BIPM2011)
```

The correlated component of the timing residuals can be obtained and inspected using the clock plugin:

```
$ tempo2 -gr clock -f psr1.par psr1.tim -f psr2.par psr2.tim -f psr3.par psr3.tim
      -global global.par -fitfunc global
```

The plugin works by using the IFUNC parameters (in a global parameter file) to define a set of grid points. Each grid point represents the value of the clock error at that time. For any specified time the clock errors are obtained by linearly interpolating the values at adjacent grid points. It is essential to constrain these clock errors so that they do not contain a quadratic polynomial signal. This is done by adding the following into any pulsar parameter file:

```
CONSTRAIN IFUNC
```

These offsets are then fitted to all pulsars simultaneously as part of the fitting procedure. This is a global fit and hence requires the use of the global fitfunc plugin.

Determining errors in terrestrial time standards using the clock plugin

For this example we simulate observations of four pulsars using the fake plugin. These pulsars are regularly sampled and have arrival time uncertainties of 100 ns. The first observation is at MJD 48000 and the last at 53000. They are simulated using parameter files that contain:

```
CLK TT(BIPM2011)
```

We then change the `CLK TT(BIPM2011)` parameter to `CLK TT(TAI)` in the parameter files. Viewing the residuals (with e.g., the splk plugin) shows the errors in TAI with respect to BIPM12. We now try and recover those differences using the clock plugin.

First, we add `CONSTRAIN IFUNC` into the parameter file for the first pulsar, psr1.par. Second, we create a global.par file that contains:

```
CLK TT(TAI)
SIFUNC 1 2
IFUNC1 48000 0 0
IFUNC2 50100 0 0
IFUNC3 50200 0 0
....
IFUNC52 53100 0 0
```

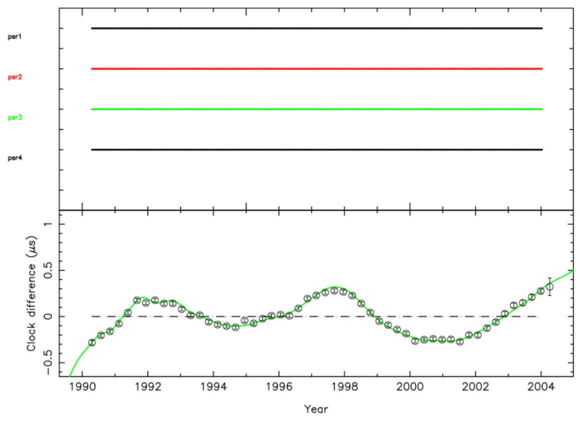
Note that this can be made using

```
$ seq 48000 100 53100 | awk '{print "IFUNC"NR,$1,0,0}' >> global.par
```

The clock plugin can then be used:

$ tempo2 -gr clock -f psr1.par psr1.tim -f psr2.par psr2.tim -f psr3.par psr3.tim -f psr4.par psr4.tim -global global.par -fitfunc global -miny -0.5 -maxy 1 -overlay tai2tt_bipm2011.clk

The output is shown in the Figure below. The top panel shows the sampling for the four pulsars (in this case each pulsar has the same sampling and data spans. The lower panel shows the clock errors measured using the pulsar data (shown as black circles with error bars). The green solid line is the expected clock errors in TT(TAI) with a quadratic polynomial fitted and removed. This curve is included using the `-overlay` option which gives an ascii data file that simply contains the MJD values and the clock error at that time.



The output from the clock plugin running on four simulated data sets.

The following options are available:

-maxy	Maximum y-value for the clock error panel
-miny	Minimum y-value for the clock error panel
-invert	Invert the clock errors given in the overlay file
-overlay	List of clock errors to overlay on the plot

7 Pulsar navigation

This work is still "in progress". Please contact G. Hobbs or D. Xinping before making use of the following algorithms.

The standard usage of TEMPO2 is to start with pulse arrival times at a well-known observatory, to form barycentric arrival times and then compare the barycentric arrival times with a pulsar timing model to obtain timing residuals. The fitting procedure can then be used to improve the timing model. In contrast TEMPO2 can also be used to assume a "perfect" pulsar timing model and reverse to process to obtain the observatory coordinates. If the observations were made by a space probe then this procedure can be used to determine the barycentric coordinates of the space probe.

The algorithms described here are currently only part of the solution. We assume that the space probe can observe multiple pulsars simultaneously and that a good ephemeris already exists for each pulsar. We also assume that the estimated position of the space probe is close to the actual position and so we're only searching for small errors in a position estimate.

The procedure is to include the estimated space-probe position in the arrival time files. The telescope site code must be set to `STL_FBAT` and the actual position (in barycentric coordinates) given using the `-telx`, `-tely` and `-telz` flags. For instance and arrival time file could look like:

```
FORMAT 1
groundStation_obs1 1400.000 51000.12345678 0.1 PKS
groundStation_obs2 1400.000 51010.12345678 0.1 PKS
...
satellite_obs1 3000.000 51030.12345678 1.0 STL_FBAT -telx 600.12345 -tely 300.12345 -telz 123.1234
satellite_obs2 3000.000 51031.12345678 1.0 STL_FBAT -telx 601.12345 -tely 301.12345 -telz 124.1234
...
```

TEMPO2 will automatically search for the `-telx`, `-tely` and `-telz` flags and set the observatory coordinates to this value. In order for the error on X, Y and Z to be determined at least three pulsars need to be observed (giving, psr1.tim, psr2.tim and psr3.tim). As we assume that these observations are simultaneous the `-telx`, `-tely` and `-telz` flags will be the same for the different pulsars.

In order to fit for the errors in the position, ΔX, ΔY and ΔZ, it is necessary to produce a global.par file that contains:

```
STEL_DX 2 2
TEL_DX1 51029.5 0 0
TEL_DX2 51030.5 0 0

STEL_DY 2 2
TEL_DY1 51029.5 0 0
TEL_DY2 51030.5 0 0

STEL_DZ 2 2
TEL_DZ1 51029.5 0 0
TEL_DZ2 51030.5 0 0
```

The grid positions must span a particular observation from the space craft. The positions can then be fitted using:

```
$ tempo2 -gr splk -f psr1.par psr2.tim -f psr2.par psr2.tim -f psr3.par psr3.tim
    -global global.par -fitfunc global
```

The space craft position errors are displayed on the screen and written out to a file called telescopeXYZ.dat for subsequent analysis.

8 Searching for errors in the planetary ephemeris

This section is based on algorithms developed and tested mainly by D. Champion, G. Hobbs, W. Coles, R. Shannon and J. Lopez. More details on §?? are available in Champion et al. (2010).

Tempo2 provides the possibility of simulating (or fitting for) the effect of a small error in the mass of the known planets. This work was described in Champion et al. (2010). Further updates have allowed the determination of offsets in the Earth-Solar System Barycentre vector with respect to a particular ephemeris. Subsequent searches for periodicities in the offsets may lead to the discovery of unknown objects in the Solar System.

8.1 The effect of a small error in the masses of the known planets

Pulse arrival times are measured at a particular observatory that is usually on the Earth. Tempo2 subsequently uses a Solar System ephemeris to determine the pulse arrival time at the Solar System Barycentre (SSB). If the mass of a planet is incorrect in the ephemeris then this correction will not be perfect. As emphased by Champion et al. (2010), we can approximate the effect of a change in the mass of a planet as a relocation of the SSB along the vector from the original SSB to that planet.

A mass change can be included in the parameter file using:

```
DMASSPLANET5 1e-7 0
```

where the '5' refers to the 5th planet (Jupiter). The error on the planet mass is in units of Solar Masses.

The induced timing residuals caused by a mass error will depend upon the ecliptic latitude of the pulsar. It is therefore common to fit for mass errors to multiple pulsar data sets simultaneously. The DMASSPLANET parameter should therefore be placed in a global parameter file (see §??). In order to carry out a global fit the "global" fit function should be used (or, in the presence of red noise, the "globalDCM" fit function).

Simulating the induced residuals caused by a planet mass error

For this example we have chosen pulsar timing models for PSRs J1022+1001, J0437−4715 and J1909−3744. In order to simulate timing residuals with the effect of an error in the mass of Jupiter we create a global parameter file (global_sim.par) that contains

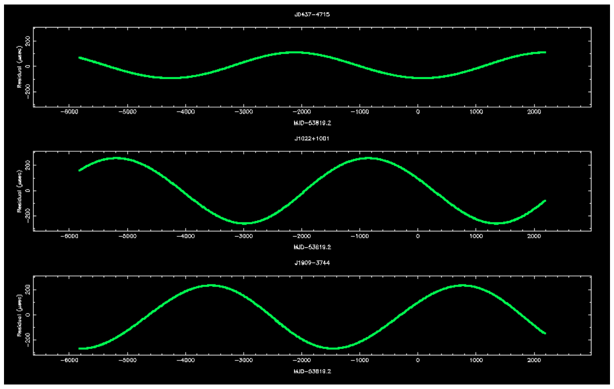
```
DMASSPLANET5 1e-7 0
```

We can then use the fake plugin to simulate arrival times for each pulsar with 14 d sampling, 100 ns uncertainties between MJDs of 48000 and 56000:

```
$ tempo2 -gr fake -f 0437.par 1022.par 1909.par -global global_sim.par -ndobs 14 -nobsd 1 -randha
y -ha 8 -start 48000 -end 56000 -rms 1e-4
```

We can then view the resulting timing residuals (without including the global parameter file) using the splk plugin.

```
$ tempo2 -gr splk -f 0437.par 0437.simulate -f 1022.par 1022.simulate -f 1909.par 1909.simulate
```



Induced timing residuals caused by an error in the estimated mass of the Jovian system of $\Delta M_J = 10^{-7} M_\odot$.

We can then attempt to fit for the planet mass error by creating a global parameter file (global.par) that contains

```
DMASSPLANET 0 2
```

(i.e., no initial estimate of the mass error and '2' to indicate undertaking a global fit.) The following

$ tempo2 -gr splk -f 0437.par 0437.simulate -f 1022.par 1022.simulate -f 1909.par 1909.simulate -global global.par -fitfunc global

produces white post-fit timing residuals and an estimate of the mass error of $(0.99999 \pm 0.00002) \times 10^{-7} M_\odot$, which is consistent with the simulated value.

8.2 Searching for unknown masses

This work is still "in progress". Please contact G. Hobbs or R. Shannon before making use of the following algorithms.

In order to detect an unknown mass in the Solar System it is first necessary to measure the difference between the actual Earth-SSB vector from the vector predicted using a Solar System ephemeris. Attempts can then be made to model those differences as being caused from an unknown mass. Offsets to the Earth-SSB vector can be obtained using the `TEL_DX`, `TEL_DY` and `TEL_DZ` parameters. These parameters take the following form in the parameter files:

```
STEL_DX 0 2
TEL_DX1 48000 0 0
TEL_DX2 48100 0 0
TEL_DX3 48200 0 0
TEL_DX4 48300 0 0
...
TEL_DX10 ....
```

(and similar parameters for `TEL_DY` and `TEL_DZ`). The parameters are set using the `STEL_DX` parameter that defines the number of telescope offsets and whether the parameter is to be fit (2=global fit). The Earth-SSB vector is subsequently modified by the values defined by a grid of points. Each grid position is defined by the MJD values in the second column. The value of the offset to the Earth-SSB vector at a given time is obtained by linearly interpolating the values at adjacent grid points. Column 3 gives the offsets and 4 their corresponding uncertainties.

Note that the `TEL_DX`, `TEL_DY` and `TEL_DZ` parameters can be covariance with the fits for the pulsars' pulse parameters. It is therefore often necessary to constrain the fit (see §**??**) using the following lines in a parameter file:

```
CONSTRAIN TEL_DX
CONSTRAIN TEL_DY
CONSTRAIN TEL_DZ
```

Searching for an unknown mass in the Solar System using the TEL_DX, TEL_DY and TEL_DZ parameters

Here we use the data sets that were created for the previous example. These data sets include an error in the mass of Jupiter of $10^{-7}\,\mathrm{M}_\odot$. Previously we assumed that we knew that the residuals were caused by a mass error in Jupiter. Here we make no assumption about the position, or size, of the mass error.

The data sets span from MJD 48000 to 56001. We choose to sample the Earth-SSB position vector every 50 d. A simple means to obtain the suitable lines to include in the global parameter file is to use the unix command:

$ seq 48000 50 56050 — awk 'print "TEL_DX"NR,$1,0,0' >> global.par

and then do the same for TEL_DY and TEL_DZ. It is then necessary to add the following lines into the global.par file:

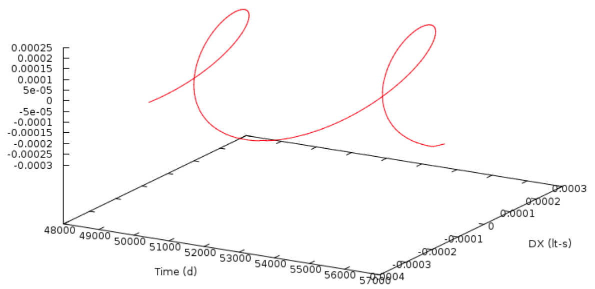
```
STEL_DX 0 2
STEL_DY 0 2
STEL_DZ 0 2
CONSTRAIN TEL_DX
CONSTRAIN TEL_DY
CONSTRAIN TEL_DZ
```

A standard global fit can then be carried out

$ tempo2 -gr splk -f 0437.par 0437.simulate -f 1022.par 1022.simulate -f 1909.par 1909.simulate -global global.par -fitfunc global -fit f0 -fit f1

The output produces white post-fit timing residual and a data file "telescopeXYZ.dat" that gives the ΔX, ΔY and ΔZ values (and uncertainties) for the offset of the Earth-SSB vector from the Solar System ephemeris prediction. Currently there is no simple way to visualise the result within TEMPO2, but the output file can be processed to search for periodicities and, if detected, fit for the orbital parameters of the unknown mass.

In the following figure the periodicity of Jupiter is clearly recovered. By rotating the plane, it is possible to identify the angle with respect to the ecliptic of the "unknown" mass (which, in this case, agrees with the known orbit of Jupiter).



ΔX and ΔY as a function of time obtained by fitting TEL_DX, TEL_DY and TEL_DZ parameters to a simulation that include an incorrect mass of Jupiter of $\Delta M_J = 10^{-7} M_\odot$.

9 Simulating pulsar data sets

This work is still “in progress”. Please contact M. Keith before making use of the following algorithms.

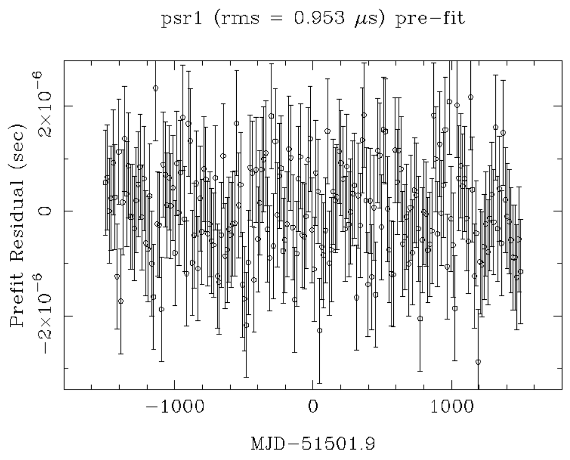
It is common to need to simulate a set of pulsar times-of-arrival. For instance, such data sets can be used to test new algorithms. The standard method for simulating arrival times is to use the fake plugin, but a new suite of software tools are being developed that have significantly more functionality than the fake plugin.

Simulating data sets using the fake plugin

The following line will simulate site arrival times for three pulsars (psr1, psr2 and psr3). The resulting arrival time files are named psr1.simulate, psr2.simulate and psr3.simulate. Each data set has 1 observation (-ndobs) every 14 days (-ndobs). The observation is assumed to have a random hour angle (-randha) that spans 8 hours from transit (-ha). The first observation is at MJD 50000 (-start) and the last at 53000 (-end). Each arrival time has an uncertainty of 1μs (the -rms option is given in milliseconds).

```
$ tempo2 -gr fake -f psr1.par psr2.par psr3.par -ndobs 14 -nobsd 1 -ha 8 -randha y -start 50000 -end 53000 -rms 1e-3
```

By default the fake plugin will simulate observations at the Parkes observatory site at an observing frequency of 1440 MHz. The data are regularly sampled and all points have the same uncertainties.



Simulated data set obtained with the fake plugin.

The fake methodology is relatively limited. A new set of tools are now being developed to extend TEMPO2’s simulation capability. These rely on an initial set of arrival times that have the correct observing frequency, observation times and telescope identifiers. Such a file can be an actual observation file or can be created using the fake plugin.

Initially “idealised” site arrival times are produced using the formIdeal plugin. Such arrival times will give zero residuals with the parameters used in the simulation. Various “correction” files are produced that can include multiple realisations of a given physical phenomenon. The user can then choose to add as many of these correction files as required to idealised site arrival times (using the createRealisation plugin). Plugins that are currently available are:

addGaussian	Adds Gaussian noise given by the arrival time uncertainties
addGWB	Adds a gravitational wave background signal
addDmVar	Adds dispersion measure variations

Simulating realistic data sets

We use with the PPTA observations of PSR J0437−4715 (with parameter file 0437.par and arrival time file 0437.tim). These observations are taken at three observing frequencies, the data are unevenly sampled and different arrival times have different uncertainties.

First we form idealised site arrival times:

```
$ tempo2 -gr formIdeal -f 0437.par 0437.tim
```

This produces a new arrival time file: 0437.tim.sim. We can now simulate one realisation of white, Gaussian noise to add to the arrival times:

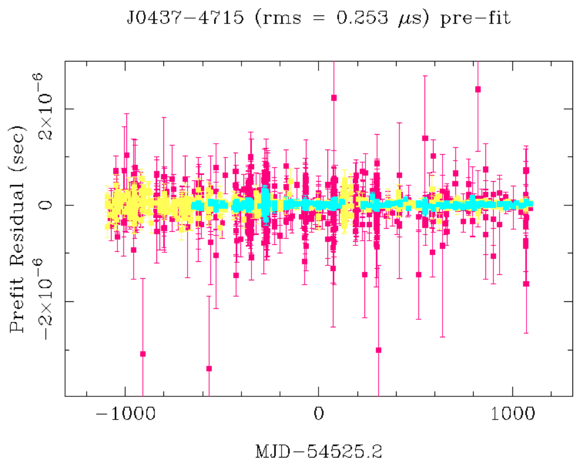
```
$ tempo2 -gr addGaussian -f 0437.par 0437.tim.sim
```

This produces a "correction" file: 0437.tim.sim.addGauss. We can create a set of arrival times with this correction using:

```
$ tempo2 -gr createRealisation -f 0437.tim.sim -corr 0437.tim.sim.addGauss
```

This produces a new arrival time file called 0437.tim.sim.real (note that the parameter file is not used with the createRealisation plugin). This can be plotted as normal using, e.g., plk:

```
$ tempo2 -gr plk -f 0437.par 0437.tim.sim.real
```



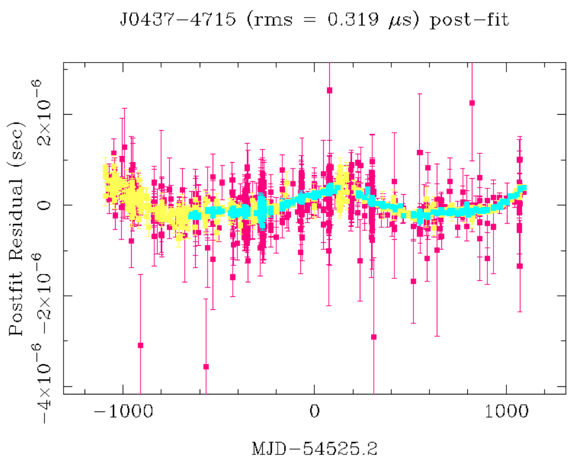
Simulated data set with Gaussian white noise. The different colours correspond to different observing frequencies.

We can then add in the effect of a gravitational wave background signal (using the same definitions as in the GWbkgrd plugin):

```
$ tempo2 -gr addGWB -f 0437.par 0437.tim.sim -dist 1 -gwamp 1e-14 -alpha -0.6666 -ngw 1000
```

This produces 0437.tim.sim.addGWB. We can then form another realisation with the Gaussian noise and the gravitational wave background

```
$ tempo2 -gr createRealisation -f 0437.tim.sim -corr 0437.tim.sim.addGauss -corr 0437.tim.sim.addGWB
```



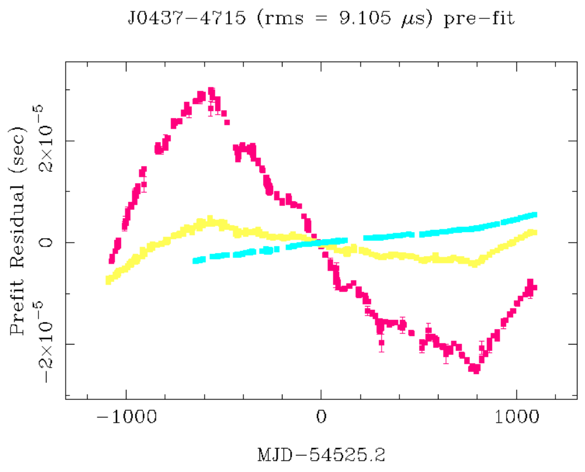
Simulated data set with Gaussian white noise and a gravitational wave background.

It is also possible to simulate dispersion measure variations caused by the interstellar medium:

```
$ tempo2 -gr addDmVar -f 0437.par 0437.tim.sim -tdiff 100
```

Note that the diffractive time scale is given in seconds. This produces 0437.tim.sim.addDmVar which can, if required, be included in the realisation:

```
$ tempo2 -gr createRealisation -f 0437.tim.sim -corr 0437.tim.sim.addGauss -corr 0437.tim.sim.addGWB -corr 0437.tim.sim.addDmVar
```



Simulated data set with Gaussian white noise, a gravitational wave background and dispersion measure variations.

10 Modifying and fixing arrival time files

10.1 The fixData plugin

The TEMPO2 fits usually do not result in a reduced-χ^2 value close to 1. This can be caused by non-white noise or incorrect uncertainties on the pulse times of arrival. It is very common to find that the arrival time uncertainties underestimate the amount of scatter in the data and, in a few cases, they overestimate the amount of scatter. TEMPO2 includes various ways to "fudge" the uncertainties to give values that are more consistent with the observed scatter. However, these factors have no physical meaning within TEMPO2 and therefore should be used with care.

In order to increase all following arrival times uncertainties by a given factor the

```
EFAC X
```

parameter can be used where X is the multiplication factor for all future arrival times (note: this can be placed at any point in the arrival time files and multiple EFACs can be used). In some cases it may be useful to include increase the error bars by some amount:

```
EQUAD X
```

This will add the existing error bar size to X (in μs) in quadrature.

Usually different corrections are required for different instruments and at different telescopes. EFAC and EQUAD apply to all observations that follow the command and so cannot easily be used in this case. Instead each observation line should include a flag (see §**??**) that contains, for instance, the backend instrumentation (e.g., `-be backend1`). Then the `T2EFAC` and `T2EQUAD` commands can be used:

```
T2EFAC -be backend1 2.1
T2EQUAD -be backend2 1.5
```

Determining whether an EFAC or an EQUAD to use, and the value that it should be set to is non trivial. In the simplest case it is often possible to carry out a weighted fit across a small section of data that does not exhibit significant red noise. The EFAC can then be set to $\sqrt{\chi_r^2}$ where χ_r^2 is the reduced-χ^2 value of the fit.

A more detailed analysis can be carried out using the fixData plugin. This plugin has two major modes:

- Mode 1: The data set is split based on a specified flag (usually to select different backend instrument). A structure function is obtained for each data set individually and the high frequency component is compared with the known arrival time uncertainties. This leads to a T2EFAC for each backend that can subsequently be included in the arrival time file.
- Mode 2: The user can input a EFAC, EQUAD or both. Various plot and histograms are shown that allow the user to decide whether the choice of EFAC or EQUAD is sufficient and, if not, choose different values.

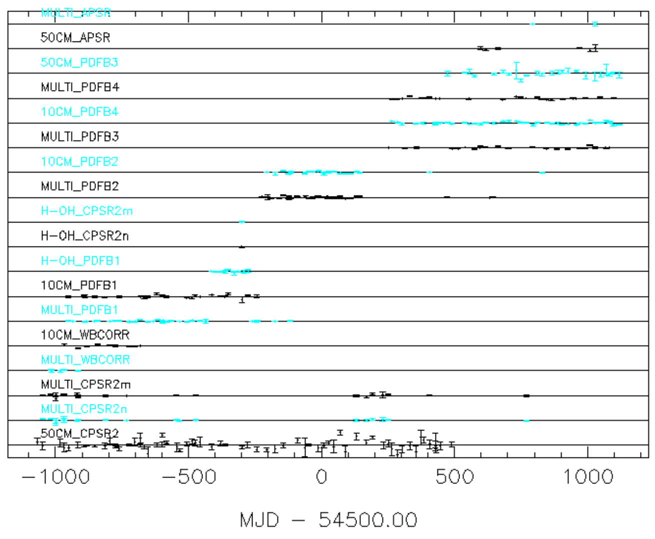
Automatically setting T2EFAC values using the fixData plugin

$ tempo2 -gr fixData -f psr.par psr.tim -flag -f -removeQuad -daygap 30

The -flag option indicates how to divide the data sets (i.e., into different backend instruments). -removeQuad removes a quadratic polynomial from the resulting residuals. The -daygap defines the longest data span to include when calculating the actual white noise level. Note, if this is set too long then red noise may affect the results. If too short then there may not be a sufficient number of observations that can be used to estimate the white noise level. The plugin lists a set of T2EFAC values, such as:

```
T2EFAC -f 50CM_CPSR2 1.30
T2EFAC -f MULTI_CPSR2m 1.16
T2EFAC -f MULTI_WBCORR 1.11
...
```

that can be added to the start of the psr.tim file.

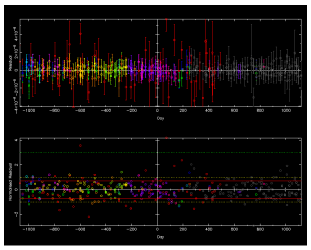
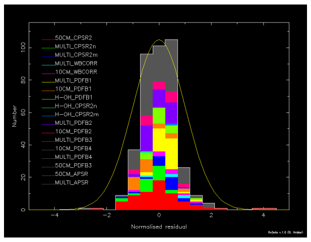
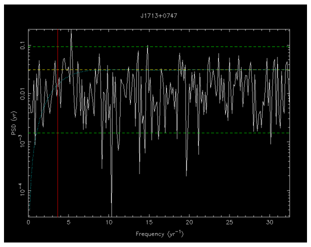


Using the fixData plugin to determine T2EFAC values automatically. Note: the colours here are shown in reverse video (TEMPO2 gives, by default, a black background).

Manually calculating EFAC and EQUAD values using the fixData plugin

$ tempo2 -gr fixData -f psr.par psr.tim -plot 3 -flag -f

This plugin initially shows three figures (examples given below). The top panel of the left-most figure shows the timing residuals where each different backend instrument is shown using a different colour. The bottom panel shows the normalised timing residuals (i.e., the residual divided by its error bar size). Red horizontal lines indicate a normalised residual of 1 and the green dotted lines indicate a normalised residual of 3. The central figure shows a histogram of the normalised residuals with a Gaussian curve overplotted. The right-most figure shows a power spectrum of the normalised timing residuals.

Using the fixData plugin to determine T2EFAC values automatically. Note: the colours here are shown in reverse video (TEMPO2 gives, by default, a black background).

The user can inspect these plots to determine whether an EFAC or an EQUAD (or both) is necessary and then the figures are re-drawn using the new values. When complete the EFAC and/or EQUAD values can be included in the arrival time file.

10.2 Averaging arrival times together

averageData

10.3 Cleaning the arrival time file

cleanTim

10.4 Select files

A “select file” can be used to select specific observations to use within the analysis. A simple file (called, for example, psr.select) may select a particular MJD range:

```
PASS MJD 55660 56035.2
```

The select file is then used by including the -select command line option:

```
$ tempo2 -gr plk -f psr.par psr.tim -select psr.select
```

10.5 Select plugins

11 Dealing with the Solar Wind

This section is based on algorithms developed and tested by X. You and W. Coles. The routines are described in You et al. (2007).

12 Binary pulsars

13 Issues arising when installing tempo2

Index

Arrival time files, 3, 37

Command line arguments, 3

Dispersion measure variations, 20
DMMODEL, 20

efac, 37
equad, 37

Fitting, 10
  Constrained fitting, 10
Flags, 3

Glitches, 11
Gravitational waves, 22
  background, 22
    detection, 24
    upper bound, 24
  continuous waves, 25
  evolving sources, 27
  gravitational wave memory, 27

Installation, 39
Interpolation, 17

Linked parameters, 10

plugins
  addDmVar, 34
  addGaussian, 34
  addGWB, 34
  autoSpectralFit, 17
  average, 39
  cleanTim, 39
  clock, 28
  createRealisation, 34
  detectGWB, 24
  fake, 34, 37, 38
  fixData, 37
  formIdeal, 34
  general2, 8
  glitch, 12
  global, 10
  GWbkgrd, 22
  interpolate, 17
  plk, 4, 15
  plotInterp, 17
  plotMany, 6
  spectralModel, 16
  splk, 6
Public data sets, 3
Pulsar navigation, 30
Pulse numbering, 15

Select plugins, 39
Simulation, 34
Solar System ephemeris, 31
Solar wind, 39
Spectral estimation, 16
STL_FBAT, 30

T2efac, 37
T2equad, 37
Time standards, 28
Timing noise, 15
Tracking, 15