The Astrophysical Journal, 980:165 (16pp), 2025 February 20 https://doi.org/10.3847/1538-4357/adaaec

Bayesian Pulsar Timing and Noise Analysis with `Vela.jl`: An Overview

Abhimanyu Susobhanan
Max-Planck-Institut für Gravitationsphysik (Albert-Einstein-Institut), Leibniz Universität Hannover, Callinstraße 38, 30167 Hannover, Germany; abhimanyu.susobhanan@aei.mpg.de
Received 2024 November 18; revised 2025 January 7; accepted 2025 January 9; published 2025 February 12

Abstract

We present `Vela.jl`, an efficient, modular, easy-to-use Bayesian pulsar timing and noise analysis package written in Julia. `Vela.jl` provides an independent, efficient, and parallelized implementation of the full nonlinear pulsar timing and noise model along with a Python binding named `pyvela`. One-time operations such as data file input, clock corrections, and solar system ephemeris computations are performed by `pyvela` with the help of the `PINT` pulsar timing package. Its reliability is ensured via careful design utilizing Julia's type system, strict version control, and an exhaustive test suite. This paper describes the design and usage of `Vela.jl` focusing on the narrowband paradigm.

Unified Astronomy Thesaurus concepts: Pulsars (1306); Astronomy software (1855); Astronomy data analysis (1858)

1. Introduction

Pulsars are rotating neutron stars whose electromagnetic radiation is received as periodic pulses. Their high rotational stability makes them excellent celestial clocks, and pulsar timing, the technique of tracking a pulsar's rotation by measuring the times of arrival (TOAs) of its pulses, is one of the most precise techniques in astronomy (D. R. Lorimer & M. Kramer 2012). Over the years, pulsar timing has been applied to study a wide array of astrophysical phenomena, such as neutron star equation of state and internal dynamics (e.g., H. T. Cromartie et al. 2020; M. Antonelli et al. 2022), tests of theories of gravity (e.g., G. Voisin et al. 2020), solar wind (e.g., C. Tiburzi et al. 2021), galactic dynamics (e.g., B. B. P. Perera et al. 2019), etc. Recently, pulsar timing array (PTA; R. S. Foster & D. C. Backer 1990) experiments found evidence (e.g., G. Agazie et al. 2024) for a stochastic gravitational-wave background (GWB; R. W. Hellings & G. S. Downs 1983) in the nanohertz frequency range with the help of pulsar timing.

In addition to the pulsar rotation, the measured TOAs are also influenced by several deterministic astrophysical processes such as the orbital motion of the Earth, proper motion of the pulsar, solar wind, interstellar dispersion, the orbital motion of the pulsar, etc. (R. T. Edwards et al. 2006), as well as stochastic processes such as radiometer noise, pulse jitter, rotational irregularities, interstellar medium variability, etc. (G. Agazie et al. 2023a). High-precision pulsar timing requires accurate modeling of all of these effects. A pulsar timing model or pulsar ephemeris provides a mathematical description of the deterministic processes that affect the measured TOAs and is often accompanied by a noise model that describes the stochastic processes therein.

In practice, pulsar timing involves estimating the parameters of a pulsar timing model given a set of TOA measurements, usually in a frequentist setting using a software package such as `tempo2` (R. T. Edwards et al. 2006; G. B. Hobbs et al. 2006) or `PINT` (J. Luo et al. 2021; A. Susobhanan et al. 2024). Noise characterization can be done in a few different ways. In `PINT`, noise parameters can be estimated together with the timing parameters in a frequentist way (A. Susobhanan et al. 2024). `tempo2` provides plugins like `fixData` and `spectralModel` that can estimate some noise parameters by fixing the timing model parameters (G. Hobbs 2014). Bayesian noise characterization can be performed using the `ENTERPRISE` (A. D. Johnson et al. 2024) and `TEMPONEST` (L. Lentati et al. 2014) packages starting from a postfit timing model, and is considered standard practice for high-precision timing experiments like PTAs. `ENTERPRISE` and `TEMPONEST` differ in how they treat the timing model fit; the former analytically marginalizes a linearized approximation of the timing model (R. van Haasteren & Y. Levin 2013), whereas the latter enables inference over the full nonlinear timing and noise model. The `PINT` package also supports Bayesian parameter estimation through `pint.fitter.MCMCFitter` and `pint.bayesian` interfaces although their use on large data sets is hampered by slow performance (J. Luo et al. 2021; A. Susobhanan et al. 2024). Other packages that have been used for Bayesian pulsar timing and/or noise analysis include `libstempo` (S. J. Vigeland & M. Vallisneri 2014), `piccard` (R. van Haasteren 2016), `PAL2` (J. Ellis & R. van Haasteren 2017), and `FortyTwo` (e.g., S. Chen et al. 2021).

In this paper, we present `Vela.jl`,[1,2] a new framework written in Julia (J. Bezanson et al. 2017) for performing Bayesian inference over the full nonlinear timing and noise model. `Vela.jl` provides an efficient and modular implementation of the pulsar timing and noise model independent of other pulsar timing packages,[3] and supports parallelized evaluation of the pulsar timing likelihood function using multithreading. We also provide a Python interface named

[1] Named after the Vela pulsar (J0835–4510; M. I. Large et al. 1968), the brightest radio pulsar. Also, Vēḷa is a word meaning occasion, time, etc. in Malayalam with cognates in several other Indian languages.
[2] The source code is available at https://github.com/abhisrkckl/Vela.jl. The documentation is available at https://abhisrkckl.github.io/Vela.jl/. This paper corresponds to version 0.0.7 (A. Susobhanan 2025).
[3] Note that `Vela.jl` relies on `PINT` to read input files, perform clock corrections, and compute solar system ephemerides. Aside from this, the timing and noise model is implemented independently. See Section 3.5 for more details.

pyvela since Python is more popular among astronomers than Julia. This package is developed with a focus on reliability, performance, and modularity (in that order), employing strict version control (using git) and rigorous testing.

The main objective of Vela.jl is to provide a flexible, robust, modular, well-documented tool for performing Bayesian inference over the full nonlinear pulsar timing and noise model. While this functionality is already available in the TEMPONEST package, Vela.jl differs from TEMPONEST in the following ways. (1) TEMPONEST can only be used with the nested sampling packages MULTINEST (F. Feroz et al. 2009) and PolyChord (W. J. Handley et al. 2015), whereas Vela.jl can be used with any Markov Chain Monte Carlo (MCMC; P. Diaconis 2009) or nested sampler (G. Ashton et al. 2022) available in Julia or Python. (2) TEMPONEST uses tempo2 internally to evaluate the timing model, whereas Vela.jl contains an independent implementation of the timing model. (3) TEMPONEST only supports the narrowband timing paradigm, whereas Vela.jl supports both the narrowband and wideband timing paradigms (the application of Vela.jl on wideband data will be discussed in a separate paper). (4) TEMPONEST is a stand-alone application whereas Vela.jl is a library that can be used programmatically alongside various Julia and Python packages.

This paper is arranged as follows. In Section 2, we provide a brief overview of the pulsar timing and noise analysis. In Section 3, we describe the design and implementation of Vela.jl and pyvela. In Section 4, we provide three examples that showcase the functionalities of Vela.jl. Finally, in Section 5, we summarize our work and discuss the future directions.

2. A Brief Overview of Bayesian Pulsar Timing and Noise Analysis

The fundamental measurable quantity in conventional pulsar timing is the TOA. TOAs are measured by folding the time series pulsar data over a known pulsar period and cross-correlating the resulting integrated pulse profile against a noise-free template (J. H Taylor 1992), usually after splitting the observation into multiple frequency subbands (this is known as narrowband timing). The more recent wideband timing technique involves measuring a single TOA and a dispersion measure (DM) from an observation by matching the frequency-resolved integrated pulse profile against a two-dimensional template known as a portrait (T. T. Pennucci et al. 2014; T. T. Pennucci 2019). Pulsar timing can also be applied directly to integrated pulse profiles (e.g., L. Lentati et al. 2015), although such techniques are unsuitable for longer pulsar timing campaigns due to intractable data volumes. In the case of high-energy observations, where it is difficult to form integrated pulse profiles or TOAs due to low photon counts, photon arrival times can be used directly for pulsar timing (e.g., M. Ajello et al. 2022). In this paper, we focus exclusively on the narrowband timing paradigm.

A narrowband TOA $t_{\rm arr}$ measured at a terrestrial observatory is related to the pulse emission time $t_{\rm em}$ as

$$\begin{aligned} t_{\rm arr} = t_{\rm em} &+ \Delta_{\rm B}(t_{\rm em}) + \Delta_{\rm ltt} + \Delta_{\rm DM}(t_{\rm em}, \nu) \\ &+ \Delta_{\rm scatter}(t_{\rm em}, \nu) + \Delta_{\rm GW}(t_{\rm em}) + \Delta_{\odot}(t_{\rm em}) \\ &+ \Delta_{\rm clock}(t_{\rm em}) + \Delta_{\rm jump} + \mathcal{N}_{\rm R} + \cdots , \end{aligned} \tag{1}$$

where ν is the observing frequency, $\Delta_{\rm B}$ represents the delays caused by the binary motion of the pulsar including Rømer delay, Shapiro delay, and Einstein delay (T. Damour & N. Deruelle 1986), $\Delta_{\rm ltt}$ represents a constant light travel time between the pulsar system barycenter and the solar system barycenter at some fiducial epoch, $\Delta_{\rm DM}$ is the dispersion delay caused by free electrons in the interstellar medium and the solar wind (D. C. Backer & R. W. Hellings 1986), $\Delta_{\rm scatter}$ represents the delay caused by interstellar scattering (D. A. Hemberger & D. R. Stinebring 2008), $\Delta_{\rm GW}$ represents gravitational-wave induced perturbations to the light travel time (F. B. Estabrook & H. D. Wahlquist 1975), $\Delta_{\odot}$ represents the delays caused by the motion of the Earth in the solar system including the Rømer delay, Shapiro delay, and Einstein delay (R. T. Edwards et al. 2006), $\Delta_{\rm clock}$ is a series of corrections that converts the TOA measured against an observatory clock to a timescale defined at the solar system barycenter (G. B. Hobbs et al. 2006), and $\Delta_{\rm jump}$ represents instrumental delays. $\mathcal{N}_{\rm R}$ is a stochastic term arising from the radiometer noise of the telescope (D. R. Lorimer & M. Kramer 2012).

Note that each of the delays Δ_X described above can have both deterministic and stochastic components, and such a separation is arbitrary and model dependent in many cases. The stochastic components corresponding to the delay terms above are orbital variations (due to tidal effects, ablation/accretion from the companion, presence of a third body, etc.; e.g., Z. Arzoumanian et al. 1994), dispersion and scattering variations (due to dynamic and inhomogeneous interstellar medium and variable solar wind; e.g., M. A. Krishnakumar et al. 2015, 2021), the stochastic GWB (e.g., G. Agazie et al. 2023b), solar system ephemeris errors (e.g., M. Vallisneri et al. 2020), clock errors (e.g., C. Tiburzi et al. 2015), etc. Some of these processes, such as stochastic GWB, solar system ephemeris errors, and clock errors, are correlated across multiple pulsars. Such processes usually cannot be distinguished from the rotational irregularities of the pulsar (described below) when only one pulsar is considered, and are therefore not included in single-pulsar analyses.

The emission time $t_{\rm em}$ estimated using Equation (1) can be related to the pulsar rotational phase ϕ using the Equation (G. B. Hobbs et al. 2006)

$$\begin{aligned} \phi = \phi_0 &+ \sum_{n=0}^{N_F} \frac{F_n}{(n+1)!}(t_{\rm em} - t_0)^{n+1} \\ &+ \phi_{\rm glitch}(t_{\rm em}) + \phi_{\rm SN}(t_{\rm em}) \\ &+ \phi_{\rm prof}(t_{\rm em}, \nu) + \mathcal{N}_{\rm jitter} + \cdots , \end{aligned} \tag{2}$$

where ϕ_0 represents an arbitrary initial phase, F_n represent the pulsar frequency and its derivatives, N_F is the number of frequency derivatives, $\phi_{\rm glitch}$ represents phase corrections due to glitches, and $\phi_{\rm SN}$ is a stochastic term representing slow stochastic variations in the pulsar rotation known as spin noise. Here, we have defined ϕ such that one full rotation corresponds to $\phi = 1$. The initial phase ϕ_0 can only be measured modulo an integer number of full rotations, and the constant light travel time $\Delta_{\rm ltt}$ is fully covariant with ϕ_0, and it is therefore excluded from Equation (1) in practice. Effects that alter the shape of the integrated pulse profile, such as frequency-dependent profile evolution (T. H. Hankins & B. J. Rickett 1986), profile shape change events (e.g., J. Singha et al. 2021), etc., thought to be of pulsar magnetospheric origin, can also influence the computed

pulse phase although they are not directly related to pulsar rotation. These are represented together as $\phi_{\rm prof}$ in the above equation. Similarly, random pulse shape variations (pulse jitter; A. Parthasarathy et al. 2021) are represented using the stochastic term $\mathcal{N}_{\rm jitter}$.

Further, if a delay term Δ_X is sufficiently small compared to the pulse period F_0^{-1}, it can be moved into Equation (2) as a phase $\Delta_X F_0$ (e.g., $\Delta_{\rm jump}$), and if a phase $|\phi_X| \ll 1$, it can be moved into Equation (1) as a delay ϕ_X/F_0 (e.g., $\phi_{\rm SN}$). We have arranged Equations (1) and (2) such that Equation (1) contains effects that are external to the pulsar whereas Equation (2) contains pulsar-intrinsic effects. However, their implementation follows historical conventions to be consistent with other pulsar timing packages.

The timing residual r is defined as

$$r = \frac{\phi - \mathfrak{N}[\phi]}{F}, \tag{3}$$

where $\mathfrak{N}[\phi]$ is the integer closest to ϕ, and $F = d\phi/dt_{\rm arr}$ is the topocentric pulse frequency (G. B. Hobbs et al. 2006).

The pulsar timing log-likelihood function $\ln L$ can be written as

$$\ln L = -\frac{1}{2}\boldsymbol{r}^T\boldsymbol{C}^{-1}\boldsymbol{r} - \frac{1}{2}\ln\det[2\pi\boldsymbol{C}], \tag{4}$$

where $\boldsymbol{r}$ is an $N_{\rm toa}$-dimensional column vector containing the timing residuals r_i corresponding to the TOAs t_i, $\boldsymbol{C}$ is the $N_{\rm toa} \times N_{\rm toa}$-dimensional TOA covariance matrix,[4] and $N_{\rm toa}$ is the number of TOAs (L. Lentati et al. 2014). Pulsar timing in a frequentist setting involves maximizing this likelihood over timing (and possibly noise) parameters (G. B. Hobbs et al. 2006; A. Susobhanan et al. 2024).

The covariance matrix $\boldsymbol{C}$ in general can be a dense symmetric matrix, and $\boldsymbol{C}^{-1}$ and $\ln\det\boldsymbol{C}$ can be computationally expensive to evaluate. To mitigate this computational cost, it is customary to use a reduced-rank approximation of $\boldsymbol{C}$ where

$$\boldsymbol{C} = \boldsymbol{N} + \boldsymbol{U}^T\boldsymbol{\Phi}\boldsymbol{U}, \tag{5}$$

such that $\boldsymbol{C}^{-1}$ and $\ln\det\boldsymbol{C}$ can be evaluated relatively inexpensively using the Woodbury lemma and the matrix determinant lemma (R. van Haasteren & M. Vallisneri 2014). Here, $\boldsymbol{N}$ is an $N_{\rm toa} \times N_{\rm toa}$-dimensional diagonal matrix containing scaled TOA measurement variances ς_i^2 given by

$$\varsigma_i^2 = E_f^2(\sigma_i^2 + E_q^2), \tag{6}$$

where σ_i is a TOA measurement uncertainty, E_f is an "error factor" (EFAC), and E_q is an "error added in quadrature" (EQUAD; L. Lentati et al. 2014). Physically, $\boldsymbol{N}$ represents the fully uncorrelated noise present in the TOAs such as radiometer noise and the uncorrelated part of the pulse jitter noise, which depend on the observing system. $\boldsymbol{U}$ is an $N_{\rm toa} \times p$ dimensional rectangular "basis" matrix, and $\boldsymbol{\Phi}$ is a $p \times p$ dimensional diagonal "weight" matrix where $N_{\rm toa}$ is the number of TOAs, and p is the rank of $\boldsymbol{C}$ such that $p \ll N_{\rm toa}$.

The correlated stochastic processes present in the TOAs are often represented as Gaussian processes, which can be written in the form of a delay $\Delta_X = \boldsymbol{U}_X^T\boldsymbol{a}_X$, where $\boldsymbol{U}_X$ is a "basis" matrix, $\boldsymbol{a}_X$ is a vector of amplitudes, which are a priori Gaussian distributed with means $\langle\boldsymbol{a}_X\rangle = 0$ and (co)variances $\langle\boldsymbol{a}_X\boldsymbol{a}_X^T\rangle = \boldsymbol{\Phi}_X$, and $\boldsymbol{\Phi}_X$ is a diagonal "weight" matrix. Such a process can be included in the timing and noise analysis in two ways. One, they can be included as a delay term in Equation (1) along with hyperpriors imposed on the weights $\boldsymbol{\Phi}_X$. Alternatively, if the contribution of $\boldsymbol{\Delta}_X$ to $\boldsymbol{r}$ is approximately linear in $\boldsymbol{a}_X$ (which is valid in most cases), the amplitudes $\boldsymbol{a}_X$ can be analytically marginalized assuming the above-mentioned Gaussian priors, and this moves the contribution of this process into $\boldsymbol{U}^T\boldsymbol{\Phi}\boldsymbol{U}$, leaving only the weights $\boldsymbol{\Phi}_X$ as free parameters (L. Lentati et al. 2014).

For example, the spin noise is usually modeled as a Fourier series such that the elements of $\boldsymbol{U}_{\rm SN}$ are given by (L. Lentati et al. 2014)

$$U_{{\rm SN};jk} = \begin{cases} \sin\left[\frac{\pi(k+1)(t_j - t_0)}{T_{\rm span}}\right] & \text{for odd } k \\ \cos\left[\frac{\pi k(t_j - t_0)}{T_{\rm span}}\right] & \text{for even } k \end{cases}, \tag{7}$$

where t_0 is some fiducial epoch, and $T_{\rm span}$ is the total span of the TOAs. In this case, the amplitude vector $\boldsymbol{a}_{\rm SN}$ contains the corresponding Fourier coefficients, and the weights contained in $\boldsymbol{\Phi}_{\rm SN}$ can be interpreted as power spectral densities. See Appendix D for details on how such processes are implemented in `Vela.jl`. Another example is the ECORR, which represents the component of the pulse jitter noise that is correlated across narrowband TOAs derived from the same observation but uncorrelated otherwise; see Appendix C for further details.

Finally, the log-posterior distribution $\ln P$ of the timing and noise model parameters $\boldsymbol{\alpha}$ given a data set $\mathfrak{D}$ (containing TOAs, TOA uncertainties, observing frequencies, instrumental configuration information, etc.) and a timing and noise model $\mathfrak{M}$ can be written using the Bayes theorem as

$$\begin{aligned} \ln P[\boldsymbol{\alpha}|\mathfrak{D}, \mathfrak{M}] = {} & \ln L[\mathfrak{D}|\boldsymbol{\alpha}, \mathfrak{M}] \\ & + \ln\Pi[\boldsymbol{\alpha}|\boldsymbol{A}, \mathfrak{M}] \\ & + \ln\Pi[\boldsymbol{A}|\mathfrak{M}] - \ln Z[\mathfrak{D}|\mathfrak{M}], \end{aligned} \tag{8}$$

where Π represents the prior distributions, $\boldsymbol{\alpha}$ represents the free parameters appearing in $\ln L$ (i.e., Equations (1)–(4)), $\boldsymbol{A}$ are the hyperparameters that determine the prior distributions of $\boldsymbol{\alpha}$ (like the weights $\boldsymbol{\Phi}_X$), and Z is a normalizing constant known as the Bayesian evidence.

The implementation of the pulsar timing and noise model summarized above (given by Equations (1)–(8)) in `Vela.jl` is detailed in the next section.

3. The Design and Implementation of `Vela.jl`

3.1. Numerical Precision

Pulsar timing is one of the few applications where the required numerical precision exceeds the precision provided by the 64 bit floating point type available in most programming languages (e.g., `double` in C and C++, `float` in Python, `Float64` in Julia). An extended-precision floating point type is required to accurately represent the measured TOA values, the pulse phase, and, in some cases, the pulsar rotational frequency. `Vela.jl` uses the `Double64` type provided by the `DoubleFloats.jl` package (J. Sarnoff 2022), which

[4] The elements of $\mathbf{C}$ are defined as $C_{ij} = \langle t_i t_j\rangle - \langle t_i\rangle\langle t_j\rangle$ where $\langle\cdot\rangle$ represents ensemble average.

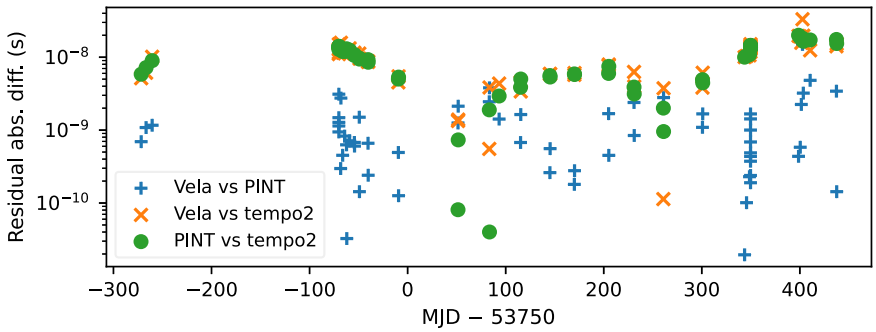

Figure 1. Absolute difference between the timing residuals computed using `Vela.jl`, `PINT`, and `tempo2` for the same timing and noise model given identical model parameters. The data set used herein corresponds to PSR J1748−2021E (P. C. C. Freire et al. 2008) and is distributed as an example along with `PINT`. See Section 4.1 for more details. The three packages agree within ∼10 ns level.

represents an extended-precision number as a sum of two `Float64`s (T. Dekker 1971), for this purpose.[5,6] Note that only the TOA values and the rotational phases are represented using `Double64`, and all other quantities are represented using `Float64`s because the software-implemented `Double64` arithmetic is significantly slower than `Float64` arithmetic, which is usually hardware supported.

The Frequency F_0 is handled manually as a sum of two `Float64` numbers as $F_0 = F_0^{\rm big} + F_0^{\rm small}$, where $F_0^{\rm big} \ll F_0^{\rm small}$ and $F_0^{\rm small}$ are treated as a free parameter where applicable. Handling F_0 this way significantly simplifies the implementation by allowing all model parameters to have the same underlying floating point type.

The timing residuals computed using `Vela.jl` agree with those computed using `PINT` and `tempo2` within the ∼10 ns level given identical timing and noise models. An example comparison is shown in Figure 1. The differences between timing residuals computed using `Vela.jl`, `PINT`, and `tempo2` shown in this figure are slightly larger than, but of the same order of magnitude as, the difference between `PINT` and `tempo2` residuals shown in J. Luo et al. (2021). Further, we find better agreement between `Vela.jl` and `PINT` as compared to either package with `tempo2`. This is expected because `Vela.jl` uses `PINT` for clock corrections and solar system ephemeris computations. We suspect the difference between `Vela.jl` and `PINT` residuals primarily arises due to the difference in the representation of extended-precision numbers.

3.2. Quantities

Dimensional analysis is an important tool for ensuring that the mathematical expressions describing the physical processes being studied are correctly translated into code. It turns out that the pulsar timing formula given in Equations (1)–(3) can be expressed such that all quantities therein have dimensions of the form $[T^d]$ where $d \in \mathbb{Z}$. We give two examples below.

1. The dispersion delay $\Delta_{\rm DM} = \mathcal{K}\mathcal{D}\nu^{-2}$ where $\mathcal{K}$ is known as the dispersion constant, $\mathcal{D}$ is the DM (the electron column density along the line of sight), and ν is the observing frequency in the solar system barycenter (SSB) frame (D. R. Lorimer & M. Kramer 2012). Although $\mathcal{D}$ has dimensions of $[L^{-2}]$, we can replace this parameter with the dispersion slope $\bar{\mathcal{D}} = \mathcal{K}\mathcal{D}$, which has dimensions of $[T^{-1}]$.
2. The binary Shapiro delay appearing in $\Delta_{\rm B}$ for an eccentric orbit can be written as (T. Damour & N. Deruelle 1986)

$$\Delta_{\rm BS} = \frac{-2GM_2}{c^3}\log[1 - e\cos u - \sin\iota(\sin\omega(\cos u - e) + \sqrt{1-e^2}\cos\omega\sin u)],$$

where M_2 is the companion mass, e is the eccentricity, u is the eccentric anomaly, ι is the orbital inclination, and ω is the argument of periapsis. Here, the parameter M_2 has dimensions of $[M]$, but it can be replaced with $\bar{M}_2 = GM_2/c^3$, which has dimensions of $[T]$.

The above observation allows us to represent each quantity q appearing in pulsar timing as a combination of a floating point number x_q and a compile-time constant integer d_q, i.e., $q = x_q\,\mathrm{s}^{d_q}$ where s represents the unit second, and the arithmetic of the q objects follow the usual dimensional analysis rules. Julia's just-in-time compilation allows arithmetic operations using this representation to be executed with almost zero run-time overhead, and this is implemented in the `GeometricUnits.jl` package[7] as the `GQ{d,X<:AbstractFloat}` type ("<:" represents "subtype of"). For example, a TOA value has the `GQ{1,Double64}` type, and the observing frequency has the `GQ{-1,Float64}` type. It also has the benefit of localizing the unit conversion operations to a certain part of the codebase, resulting in easier debugging. It should be noted that time quantities like the TOA values and the various epochs appearing in the timing and noise model are shifted such that the rotational frequency epoch (t_0 in Equation (2)) vanishes.

In comparison, `PINT` uses the `astropy.units` module (T. P. Robitaille et al. 2013) for this purpose, which has a nonnegligible computational overhead, and `tempo2` does not enforce dimensional correctness in this way at all.

[5] The `tempo2` package (and by extension, `TEMPONEST`) uses the `long double` type available in C++ to represent TOAs and other quantities internally. Similarly, `PINT` uses the `numpy.longdouble` type provided by the `numpy` package for this purpose, which in turn is implemented using the C `long double`. Unfortunately, `long double` is not fully defined by the C and C++ standards, and its properties are hardware and compiler dependent. In machines where `long double` does not provide adequate precision, `tempo2` falls back to the `__float128` type available as a compiler extension in the GNU C Compiler (`gcc`). `Double64` avoids this issue since it is defined in terms of the `Float64` type.

[6] Extended precision is not required to represent the linearized timing model such as in `ENTERPRISE` since it is only concerned with small deviations from best-fit values.

[7] Available at https://github.com/abhisrkckl/GeometricUnits.jl/.

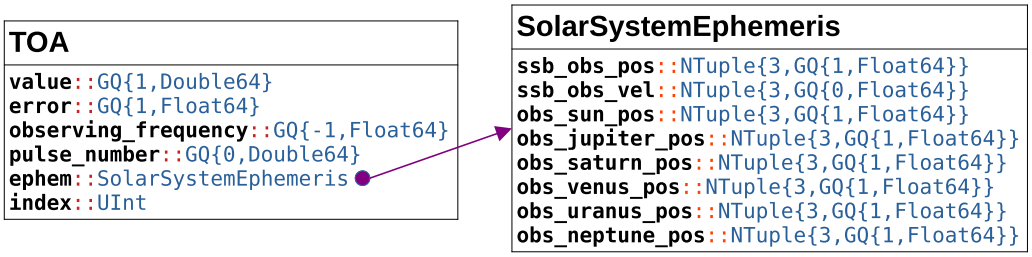

Figure 2. The structure of the `TOA` type. "::" represents "is instance of." `TOA.value` is in the TDB timescale. The `SolarSystemEphemeris` type contains three tuples representing the position and velocity of the observatory with respect to the SSB (`SolarSystemEphemeris.ssb_obs_pos` and `SolarSystemEphemeris.ssb_obs_vel`), and the positions of various solar system objects with respect to the observatory. The clock corrections required for computing `TOA.value` and the solar system ephemerides are evaluated with the help of `PINT`. `UInt` is an unsigned integer.

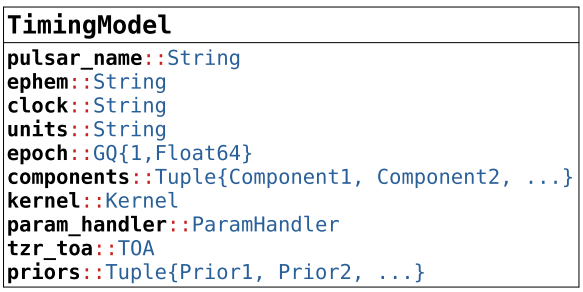

Figure 3. The structure of the `TimingModel` type. "::" represents "is instance of." `pulsar_name` stores the name of the pulsar. `ephem` is the name of the solar system ephemeris model. `clock` is the name of the realization of the TT timescale used in clock corrections (e.g., TT(BIPM2021)); see G. B. Hobbs et al. (2006). `units` is the timescale of `TOA.value` (only TDB is supported). `epoch` is the rotational frequency epoch. The `components` tuple contains `Component` objects representing different terms in the timing and noise model that are uncorrelated across TOAs. `Kernel` is an object representing the computation in Equation (4) including the various contributions to the TOA covariance matrix $\boldsymbol{C}$. `param_handler` contains information about the different parameters appearing in the timing and noise model. `tzr_toa` is a fiducial TOA with respect to which the rotational phase is measured ("tzr" stands for "t-zero"). `priors` is a tuple that contains the prior distributions for the free model parameters.

3.3. TOAs

A TOA value measured against an observatory clock is usually stored alongside the corresponding measurement uncertainty, observing frequency, a telescope code, and information about the observation setup represented as flags. The various delay and phase corrections as well as the TOA covariance matrix appearing in Equations (1)–(4) may in general depend on any subset of this information.

A TOA is represented in `Vela.jl` by the `TOA` type, whose structure is shown in Figure 2. The `TOA.value` element represents the measured TOA value in barycentric dynamic time (TDB; S. Klioner et al. 2009), `TOA.error` represents the measurement error, `TOA.observing_frequency` represents the observing frequency in the observatory frame, `TOA.pulse_number` is the pulse number corresponding to the TOA ($N[\phi]$ in Equation (3)), `TOA.ephem` contains the solar system ephemerides evaluated at the TOA epoch, and `TOA.index` is an ordinal index. The TOAs are usually stored and distributed as `tim` files, and `Vela.jl` uses `PINT` to read these files to create a collection of `TOA` objects. The clock corrections needed for converting the TOA value from the observatory timescale to TDB and the solar system ephemerides are precomputed using `PINT`, which uses `astropy` (T. P. Robitaille et al. 2013) internally; see Section 3.5. A data set contains many TOAs, and this is represented as a `Vector{TOA}` object that stores the `TOA`s contiguously in memory.

This representation of the TOA is based on the following assumptions: (a) a phase-connected timing solution is available, and (b) the collection of TOAs is immutable.[8] The first assumption allows us to precompute `TOA.pulse_number`. The second assumption allows us to convert the TOA flags, which are string key-value pairs that are expensive to store and manipulate, into inexpensive bit masks specific to the timing model. For example, if the timing model contains $N_{\rm jump}$ observing system-dependent jumps ($\Delta_{\rm jump}$), which depend on some TOA flags, the same information can be represented as an $N_{\rm toa} \times N_{\rm jump}$-dimensional bit mask (this is stored within the timing model instead of the TOAs).

3.4. The Timing and Noise Model

We begin this section by writing down a version of Equation (1) that is closer to how it is implemented in practice. Some of the delays are omitted for simplicity.

$$t_{\rm em} + \mathcal{N} = t_{\rm tdb} - \Delta_{\odot}(\bar{t}_{\rm tdb}) - \Delta_{\rm DM}(\bar{t}_{\rm ssb}) - \Delta_{\rm B}(\bar{t}_{\rm dd}) - \cdots , \tag{9}$$

where $\bar{t}_X$ represents a TOA (t_X) along with its uncertainty (σ_X), observing frequency (ν_X), etc. after a correction step X, $\mathcal{N}$ represents the uncorrelated noise present in the data, $t_{\rm tdb} = t_{\rm arr} - \Delta_{\rm clock}(\bar{t}_{\rm arr})$ is the TOA in the TDB timescale, $t_{\rm ssb} = t_{\rm tdb} - \Delta_{\odot}(\bar{t}_{\rm tdb})$ is the barycentered TOA, $\nu_{\rm ssb} = \nu_{\rm tdb}(1 - \delta_{\odot}(\bar{t}_{\rm tdb}))$ is the barycentered observing frequency, $\delta_{\odot}$ is a Doppler factor due to solar system motion, and $t_{\rm dd} = t_{\rm ssb} - \Delta_{\rm DM}(\bar{t}_{\rm ssb})$ is the dedispersed TOA. The above expression makes it clear that the delay corrections must be applied in a specific order: solar system delays, interstellar dispersion delays, pulsar binary delays, etc. *Then*, the different phase terms in Equation (2) are computed.

In `Vela.jl`, the different delay and phase terms in Equations (1) and (2) and the TOA uncertainty corrections in Equation (6) that can be computed independently for each TOA are implemented as subtypes of the `Component` type. The hierarchy and descriptions of various `Component` types available in `Vela.jl` are given in Appendix A. Each `Component` type has an associated `correct_toa()` method, which computes the corresponding delay, phase, Doppler factor, uncertainty correction, etc., as applicable. The

[8] These assumptions are valid for single-pulsar noise and timing analysis, which is performed after the data preparation and some preliminary timing is done. However, they do not apply to timing packages like `tempo2` or `PINT` because their use cases include adding/removing TOAs and editing TOA flags.

```python
from pyvela import SPNTA
import emcee
import numpy as np

spnta = SPNTA(
    parfile="NGC6440E.par",
    timfile="NGC6440E.tim",
    custom_priors="NGC6440E_priors.json",
)

nwalkers = spnta.ndim * 5
p0 = np.array([
    spnta.prior_transform(cube)
    for cube in np.random.rand(nwalkers, spnta.ndim)
])

sampler = emcee.EnsembleSampler(
    nwalkers,
    spnta.ndim,
    spnta.lnpost,
)

sampler.run_mcmc(p0, 6000, progress=True)
samples_raw = sampler.get_chain(flat=True, discard=1000, thin=50)

samples = spnta.rescale_samples(samples_raw)
```

Figure 4. An example code snippet demonstrating the `pyvela` interface with the MCMC sampler `emcee`. User-defined priors are read from a `JSON` file (see Appendix B). The `emcee.EnsembleSampler` object is initialized with samples drawn from the prior distribution with the help of `SPNTA.prior_transform()`. The samples are originally in `Vela.jl`'s internal units. They are converted into their commonly used units using `SPNTA.rescale_samples()`.

```python
from pyvela import SPNTA
import nestle

spnta = SPNTA(
    parfile="NGC6440E.par",
    timfile="NGC6440E.tim",
    custom_priors="NGC6440E_priors.json",
)

results = nestle.sample(
    spnta.lnlike,
    spnta.prior_transform,
    spnta.ndim,
    method="multi",
    npoints=500,
    dlogz=0.001,
    callback=nestle.print_progress,
)

samples_raw = nestle.resample_equal(results.samples, results.weights)
samples = spnta.rescale_samples(samples_raw)

log_evidence = results.logz
```

Figure 5. An example code snippet demonstrating the `pyvela` interface with the nested sampler `nestle`. `method="multi"` invokes the `MultiNest` algorithm (F. Feroz et al. 2009).

corrected TOA $t_{\rm em}$, the phase $\phi(t_{\rm em})$, the scaled TOA uncertainty ς, and the topocentric frequency F are computed by successively applying these methods in the correct order. Finally, the timing residuals are computed using Equation (3).

The likelihood function is computed from the timing residuals with the help of a `Kernel` object, which represents the matrix operations present in Equation (4). Two `Kernel` types are available currently. `WhiteNoiseKernel` represents the case when only uncorrelated (white) noise is present in the TOAs, i.e., the covariance matrix $\boldsymbol{C} = \boldsymbol{N}$ is diagonal. In this case, $\ln L$ can be evaluated with $\mathcal{O}(N_{\rm toa})$ time complexity. `EcorrKernel` represents the case when only time-uncorrelated noise (i.e., white noise and ECORR) is present. It turns out that $\ln L$ can be evaluated with $\mathcal{O}(N_{\rm toa})$ time complexity in the latter case also; see Appendix C for details.

The timing and noise model is represented in `Vela.jl` as the `TimingModel` type, shown in Figure 3. In addition to the `Components` and the `Kernel` described above, a `TimingModel` object also contains a `ParamHandler` object that contains information about the model parameters and an ordered collection of `Prior` objects, which implement the prior distributions appearing in Equation (8); see Appendix B for more details.

`Vela.jl` provides functions that return callable objects, which compute the log-likelihood (`get_lnlike_func()`),

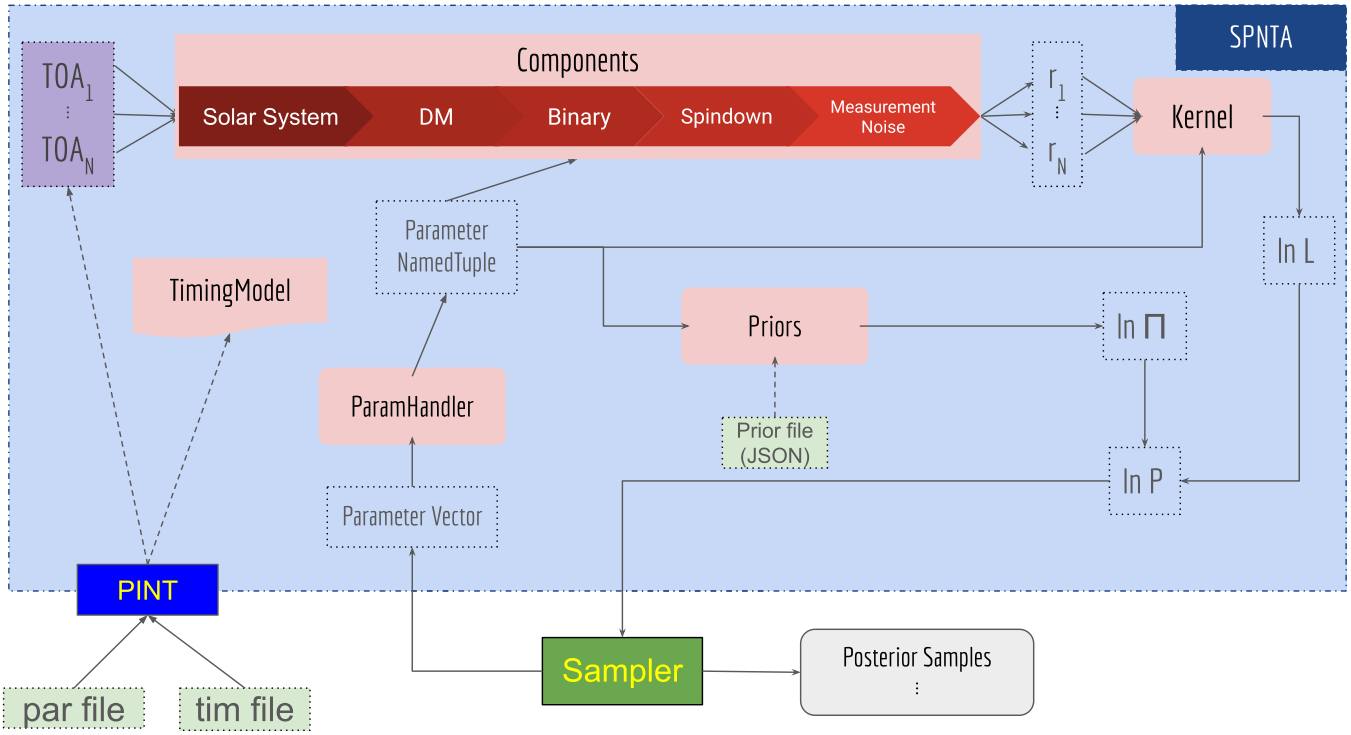

Figure 6. A schematic diagram summarizing `Vela.jl`'s architecture. `Components`, `Kernel`, `Priors`, and `ParamHandler` are parts of the `TimingModel`.

the log-prior (`get_lnprior_func()`), the prior transform (`get_prior_transform_func()`), and the log-posterior (`get_lnpost_func()`). The outputs of these functions can be passed on to any MCMC or nested sampler.

Three parallelization paradigms are provided for the log-likelihood and log-posterior computation. By default, the computation is parallelized across TOAs using multithreading (Equations (4) and (C5) are trivially parallelizable this way). Some ensemble samplers like `emcee` support vectorized execution of the posterior distribution across multiple points in the parameter space, and `Vela.jl` provides the option to do this parallelly using multiple threads. Alternatively, `Vela.jl` also provides serial versions of the same functions for cases where the sampler itself implements parallelization, e.g., using Message Passing Interface (W. Gropp et al. 1996).

3.5. The `pyvela` Interface

We provide a Python interface for `Vela.jl` called `pyvela` developed using `JuliaCall/PythonCall` (C. Rowley 2022) for easy usage. This is useful because pulsar astronomers tend to be more familiar with Python than Julia, and because Python offers a wider choice of sampling packages than Julia.

The single-pulsar noise and timing analysis (SPNTA) class in `pyvela` reads a pair of `par` and `tim` files, performs the clock corrections and solar system ephemeris computations with the help of `PINT`, constructs the prior distributions (see Appendix B), and creates a pair of `TimingModel` and `Vector{TOA}` objects described above. It also provides a convenient interface for evaluating the log-likelihood, log-prior, prior transform, and log-posterior functions. An example code snippet using `pyvela` with the MCMC sampler `emcee` (D. Foreman-Mackey et al. 2013) is shown in Figure 4, and an example using `pyvela` with the nested sampler `nestle` (K. Barbary 2021) is shown in Figure 5.

A schematic diagram that summarizes how `Vela.jl` works is shown in Figure 6.

4. Application to Simulated Data Sets

In this section, we demonstrate the application of `Vela.jl` on two simulated data sets. These are generated using the `pint.simulation` module (A. Susobhanan et al. 2024) based on certain real data sets.

4.1. PSR J1748–2021E (NGC 6440E)

PSR J1748−2021E is an isolated pulsar located in the globular cluster NGC 6440 (P. C. C. Freire et al. 2008). The simulated data set is generated based on a real data set distributed as an example alongside `PINT` (J. Luo et al. 2021) and contains 61 TOAs taken using the Green Bank Telescope from 2005 to 2006 with observing frequencies in the range 1550–1212 MHz.

We convert the TOAs from the observatory timescale to TDB with the help of the BIPM2021 realization of the terrestrial time (TT) and the DE421 solar system ephemeris. The timing and noise model includes solar system delays, interstellar dispersion, spindown, an overall phase offset, and a global EFAC that scales the measured TOA uncertainties. This model has seven free parameters.

We run the Bayesian analysis using `Vela.jl` with `emcee`, which implements the affine-invariant ensemble sampler algorithm (D. Foreman-Mackey et al. 2013; see Figure 4 for the Python script).[9] The prior distribution of the global EFAC is LogNormal[0, 0.25], and that of the overall phase offset is Uniform[−0.5, 0.5]. "Cheat" prior distributions, i.e., uniform distributions centered at the maximum likelihood values obtained

[9] This MCMC run was executed for 6000 steps with 35 walkers on an AMD Ryzen 7 CPU with 16 logical cores using four threads. It took approximately 2 s.

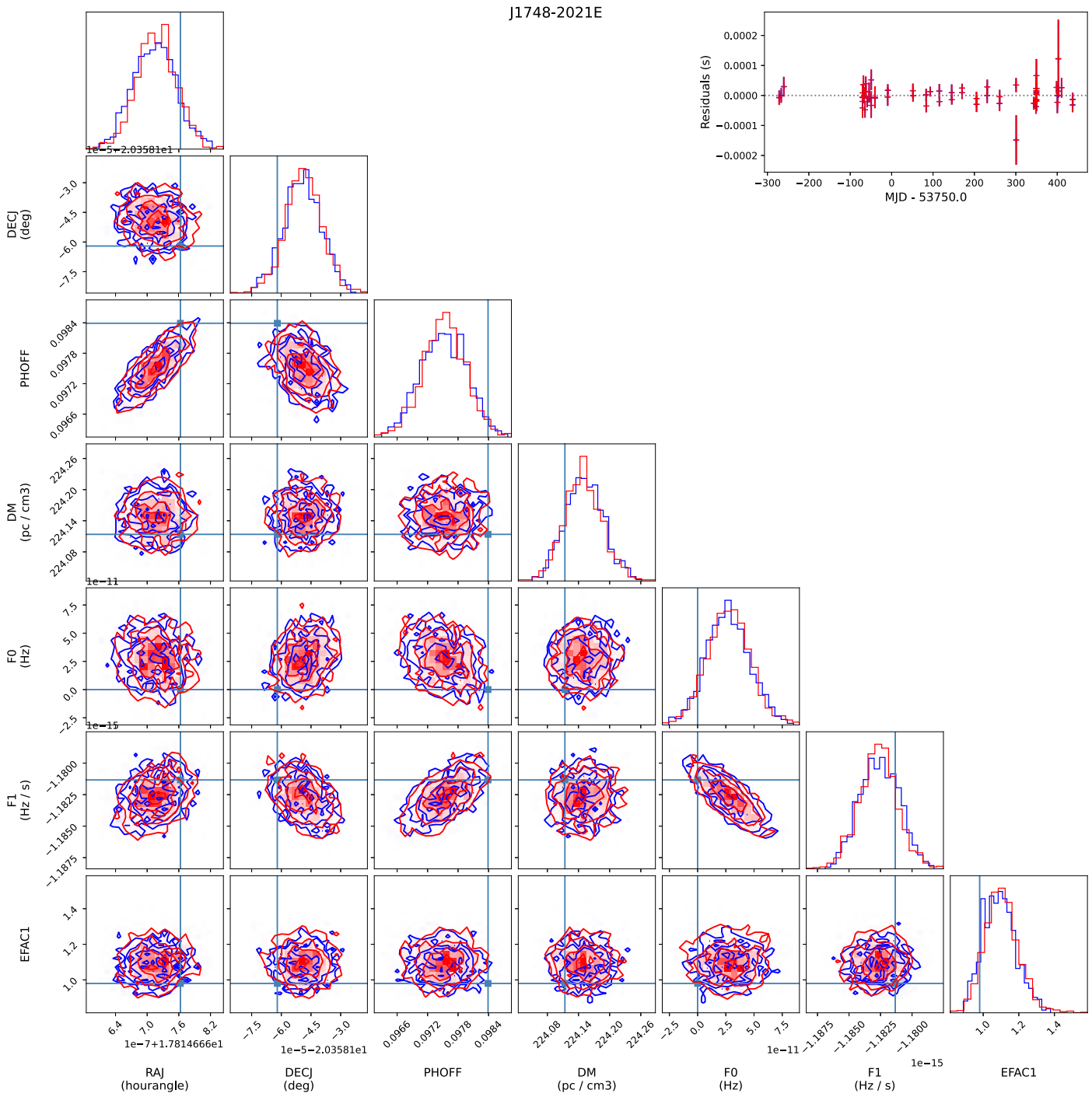

Figure 7. Bayesian timing and noise analysis results for a simulated data set of PSR J1748−2021E. The plotted parameters are source coordinates (RAJ, DECJ), overall phase offset (PHOFF), dispersion measure (DM), rotational phase and its derivative (F0, F1), and a global EFAC (EFAC1). The results obtained using `Vela.jl` are plotted in red, and those obtained using `PINT` are plotted in blue. The inset shows the timing residuals computed using the median values of the posterior samples. A value of 61.4854765543713046 Hz has been subtracted from the rotational frequency F0.

using `PINT` whose widths are 10 times the frequentist uncertainties, are used for all other parameters (see Appendix B). We have checked that increasing the width of the "cheat" priors does not appreciably alter the posterior distribution.

We repeat this analysis using the `pint.bayesian` module (A. Susobhanan et al. 2024) with `emcee` for comparison. This is computationally feasible due to the small size of the data set. The posterior distributions and postfit residuals obtained from this exercise are shown in Figure 7, and show good agreement between the posterior distributions and postfit residuals obtained using `Vela.jl` and `pint.bayesian`. Further, the estimated parameters agree with the injected parameters within 2σ uncertainties.

4.2. PSR J1909−3744

PSR J1909−3744 is a millisecond binary pulsar observed as part of multiple PTA campaigns. The simulated data set used in

Table 1
The Prior Distributions Used for Analyzing the Simulated Data Set of PSR J1909−3744

Parameter	Description	Prior
M2	Companion mass ($M_\odot$)	Uniform distribution around the IPTA DR2 measurement with width 40 times the corresponding uncertainty
SINI	Orbital inclination	$\Pi(\sin\iota) = \frac{\sin\iota}{\sqrt{1-\sin^2\iota}}$ (see Appendix B)
TNDMAMP	Spectral log-amplitude of DM noise	LogUniform[10^{-18}, 10^{-12}]
TNDMGAM	Spectral index of DM noise	Uniform[0.5, 7]
PHOFF	Overall phase offset	Uniform[−0.5, 0.5]
EFAC	Scale factor for TOA uncertainties	LogNormal[0, 0.25]
EQUAD	TOA uncertainty correction added in quadrature (μs)	LogUniform[10^{-3}, 10]

Note. "Cheat" priors are used for timing parameters not mentioned here. IPTA DR2 is the International Pulsar Timing Array Data Release 2 (B. B. P. Perera et al. 2019).

this section is generated based on a subset of the narrowband data of J1909–3744 published as part of the Indian Pulsar Timing Array data release 1 (InPTA DR1; P. Tarafdar et al. 2022). It contains 361 TOAs measured using the Giant Metrewave Radio Telescope during 2020–2021 in two frequency bands (300–500 MHz and 1260–1460 MHz). Notably, we have injected the DM variations based on the epoch-wise DM measurements given in the InPTA DR1.

We convert the TOAs from the observatory timescale to TDB with the help of the BIPM2019 realization of the TT timescale and the DE440 solar system ephemeris. The timing and noise model used to fit the simulated data includes solar system delays (including parallax and proper motion), DM variations modeled as a combination of a Taylor series (up to the second DM derivative), and a Fourier series Gaussian process with 40 harmonics (see Appendix D), the ELL1 model for a nearly circular binary, pulsar spindown, and EFACs and EQUADs applied to the two observing frequency bands. This model has 104 free parameters in total.

The prior distributions of some of the parameters are given in Table 1. The priors for the scaled Fourier amplitudes of the DM noise are unit normal distributions as discussed in Appendix D. "Cheat" priors are used for the other parameters with width 40 times their frequentist uncertainty. We have checked that increasing the width of the "cheat" priors does not appreciably alter the posterior distribution.

The posterior samples were obtained using `emcee`,[10] and the results of this analysis are plotted in Figure 8. We see that some of the estimated parameters do not agree well with their injected values. This may be because the DM model is not adequately modeling some of the short-timescale DM variations (see inset of Figure 8).

5. Summary and Future Directions

We have developed a new package for performing Bayesian single-pulsar timing and noise analysis named `Vela.jl`. This package is written in Julia and provides an efficient and parallelized implementation of the nonlinear timing and noise model along with a Python interface named `pyvela`. It uses `PINT` to read `par` & `tim` files, apply clock corrections to the TOAs, and compute solar system ephemerides but provides an independent implementation of the timing and noise model. Given a set of TOAs and a timing and noise model, `Vela.jl` provides an intuitive interface for computing the log-likelihood, log-prior, prior transform, and log-posterior functions that are compatible with various MCMC and nested samplers available in Julia and Python. The architecture of `Vela.jl` is summarized below.

1. Dimensionful quantities are converted to a form that has dimensions $[T^d]$ and is represented using `GQ{d,X<:AbstractFloat}` types. Values that require extended precision are stored using the `Double64` type.
2. Clock-corrected TOAs and their metadata along with the precomputed solar system ephemerides are stored using the `TOA` type.
3. The timing and noise model is represented using the `TimingModel` type. It contains the following:
   (a) `Components` that represent various astrophysical and instrumental effects that are uncorrelated across TOAs,
   (b) a `Kernel` that represents the log-likelihood computation and correlated noise if any,
   (c) `Priors` that represent the prior distributions of free model parameters,
   (d) a `ParamHandler` that converts the parameter values provided by the sampler into a representation that can be accessed efficiently.
4. The `pyvela` interface provides a Python binding to `Vela.jl`. It contains the `SPNTA` class, which
   (a) reads the `par` and `tim` files, computes clock corrections and solar system ephemerides, and constructs the `TimingModel` and `TOA` objects;
   (b) provides methods that compute the log-likelihood, log-prior, prior transform, and log-posterior that can be passed into samplers.

We demonstrated the usage of `Vela.jl` using two example data sets. In the case of the smaller data set, we showed that the parameter estimation results are consistent with those estimated using `PINT`.

`Vela.jl` was developed to be a more user-friendly and flexible alternative to `TEMPONEST`, and it should complement

[10] Executed for 6000 steps with 520 walkers on an AMD Ryzen 7 CPU with 16 logical cores with 16 threads. It took approximately 4.5 minutes.

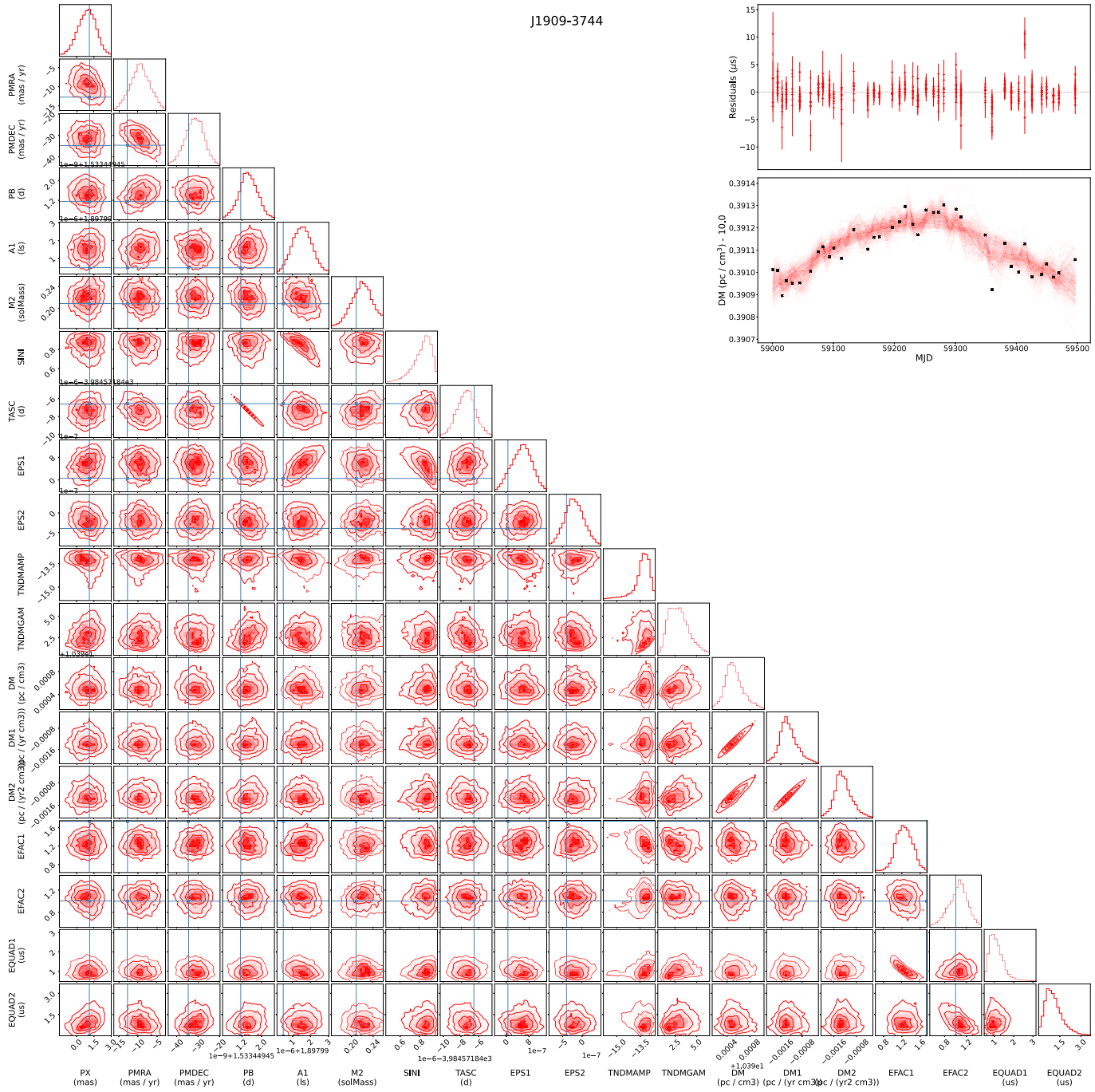

Figure 8. Bayesian timing and noise analysis results for a simulated data set of PSR J1909−3744. The plotted parameters are parallax (PX), proper motion (PMRA, PMDEC), orbital period (PB), projected semimajor axis of the pulsar orbit (A1), companion mass (M2), sine-inclination (SINI), epoch of ascending node (TASC), Laplace–Lagrange parameters (EPS1, EPS2), dispersion measure and its derivatives (DM, DM1, DM2), power-law spectral parameters of the DM noise (TNDMAMP, TNDMGAM), EFACs, and EQUADs. The two EFACs and EQUADs correspond to the 300−500 MHz band and the 1260−1460 MHz band in that order. The source coordinates (RAJ, DECJ), the rotational frequency and its derivative (F0, F1), the overall phase offset (PHOFF), and the Fourier amplitudes of the DM noise are not plotted. The vertical and horizontal lines represent the injected values. The estimated parameters of astrometric and binary parameters are consistent with the injected values within 3σ uncertainties. Note that a Fourier series Gaussian process model was used for the analysis whereas the injected DM values were taken from real epoch-wise measurements. The inset shows postfit residuals in the top panel. The bottom panel of the inset shows the injected DM time series (black points) and 200 random realizations of the DM time series drawn from the posterior distribution.

the capabilities of `ENTERPRISE`, which does not allow the exploration of the full nonlinear timing model. `Vela.jl` will be a useful tool for analyzing the various pulsar timing data sets, especially those of high-precision experiments such as PTAs that are rapidly growing in volume and sensitivity.

The planned future development wishlist for `Vela.jl` is as follows.

1. Develop a vibrant community of users and developers.
2. Implement wideband timing (M. F. Alam et al. 2021; under development).
3. Implement photon domain timing for high-energy pulsar timing (H. J. Pletsch & C. J. Clark 2015), and enable simultaneous analysis of radio TOA data and high-energy photon data.

4. Implement a sampler that is tuned to the pulsar timing and noise analysis. Possible avenues to explore are Gibbs sampling (e.g., N. Laal et al. 2023) and Hamiltonian Monte Carlo (e.g., G. E. Freedman et al. 2023).
5. Implement automatic differentiation using a tool like `Zygote.jl` (M. Innes 2018). This is crucial for implementing Hamiltonian Monte Carlo.
6. Implement a fitter for `PINT` using `Vela.jl` as a backend. This should be useful in cases where `PINT` is too slow, especially for noise estimation (A. Susobhanan et al. 2024).
7. Implement timing model components that are currently unavailable in `Vela.jl`, such as the DDGR binary model (T. Damour & N. Deruelle 1986), advanced solar wind models (J. S. Hazboun et al. 2022; S. C. Susarla et al. 2024), etc.

Acknowledgments

A.S. thanks David Kaplan for valuable suggestions on the manuscript and Tjonnie Li and Rutger van Haasteren for fruitful discussions. A.S. thanks the anonymous referee for valuable comments and suggestions.

Software: `PINT` (J. Luo et al. 2021; A. Susobhanan et al. 2024), `emcee` (D. Foreman-Mackey et al. 2013), `nestle` (K. Barbary 2021), `numpy` (C. R. Harris et al. 2020), `astropy` (T. P. Robitaille et al. 2013), `matplotlib` (J. D. Hunter 2007), `corner` (D. Foreman-Mackey 2016), `DoubleFloats.jl` (J. Sarnoff 2022), `Distributions.jl` (M. Besanon et al. 2021), `JuliaCall/PythonCall.jl` (C. Rowley 2022), `git`.[11]

Data Availability

The data set of PSR J1748−2021E used to generate the simulated data set is distributed with `PINT` as an example data set at https://github.com/nanograv/PINT/. The data set of PSR J1909−3744 used to generate the simulated data set is available as part of the InPTA DR1 at https://github.com/inpta/InPTA.DR1/.

Appendix A
Component Types Available in `Vela.jl`

Figure 9 shows the hierarchy of the various abstract subtypes of `Component`. Concrete `Component` types are listed in Table 2.

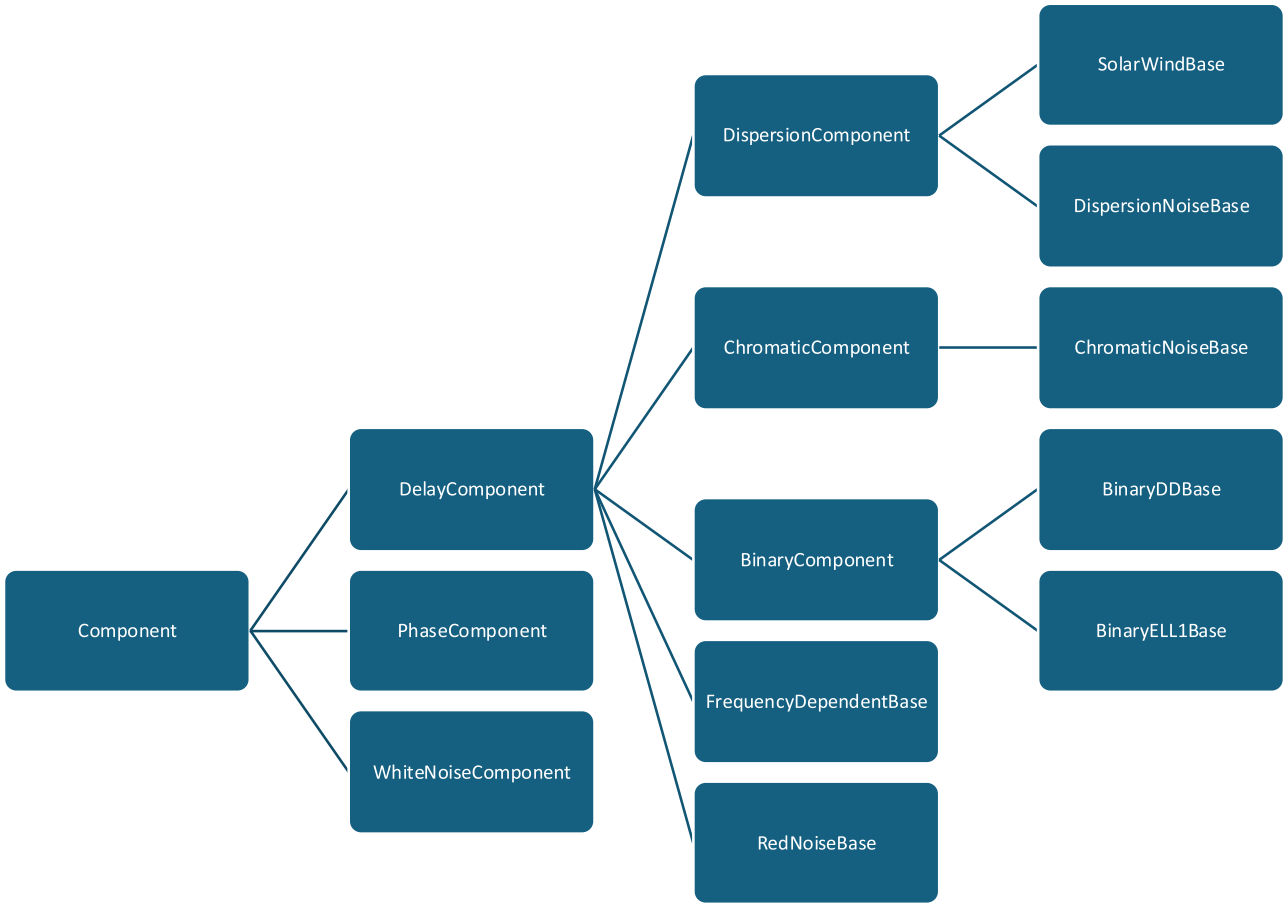

Figure 9. The hierarchy of `Component` base types. See Table 2 for their implemented concrete subtypes.

[11] https://git-scm.com/

Table 2
Timing and Noise Model Components Available in `Vela.jl`

Component and Base Type	`PINT` Equivalent(s)	Description	References

`SolarSystem`
`<: DelayComponent`
	`AstrometryEquatorial`
	`AstrometryEcliptic`
	`SolarSystemShapiro`
		Solar system delays (Rømer, parallax, and Shapiro) and Doppler correction	(R. T. Edwards et al. 2006)

`SolarWindDispersion` `<: SolarWindBase`	`SolarWindDispersion`	Solar wind dispersion assuming spherical symmetry	(R. T. Edwards et al. 2006)
`DispersionTaylor` `<: DispersionComponent`	`DispersionDM`	Taylor series representation of interstellar dispersion	(D. C. Backer & R. W. Hellings 1986)
`DispersionPiecewise` `<: DispersionComponent`	`DispersionDMX`	Piecewise-constant representation of interstellar dispersion	(Z. Arzoumanian et al. 2015)
`DMWaveX` `<: DispersionNoiseBase`	`DMWaveX`	Unconstrained Fourier series representation of interstellar dispersion variations	(A. Susobhanan et al. 2024)
`PowerlawDispersionNoiseGP` `<: DispersionNoiseBase`	`PLDMNoise`	Fourier series Gaussian process representation of interstellar dispersion variations with a power-law spectrum	(L. Lentati et al. 2014)
`DispersionOffset` `<: DispersionComponent`	`FDJumpDM`	System-dependent dispersion measure offset	
`DispersionJump` `<: DispersionComponent`	`DispersionJump`	System-dependent dispersion measure offset without a delay (for wideband data)	(M. F. Alam et al. 2021)
`ChromaticTaylor` `<: ChromaticComponent`	`ChromaticCM`	Taylor series representation of a chromatic delay	
`ChromaticPiecewise` `<: ChromaticComponent`	`ChromaticCMX`	Piecewise-constant representation of a chromatic delay	(D. A. Hemberger & D. R. Stinebring 2008)
`CMWaveX` `<: ChromaticNoiseBase`	`CMWaveX`	Unconstrained Fourier series representation of variable chromatic delay	
`PowerlawChromaticNoiseGP` `<: ChromaticNoiseBase`	`PLChromNoise`	Fourier series Gaussian process representation of variable chromatic delay with a power-law spectrum	(L. Lentati et al. 2014)
`BinaryDD` `<: BinaryDDBase`	`BinaryDD` `BinaryBT`	Parameterized model for eccentric binaries with relativistic effects	(T. Damour & N. Deruelle 1986)
`BinaryDDH` `<: BinaryDDBase`	`BinaryDDH`	Similar to `BinaryDD`, but with an orthometric representation of Shapiro delay suitable for low-inclination systems	(J. M. Weisberg & Y. Huang 2016)
`BinaryDDK` `<: BinaryDDBase`	`BinaryDDK`	Similar to `BinaryDD`, but with Kopeikin corrections due to parallax and proper motion	(S. M. Kopeikin 1995, 1996)
`BinaryDDS` `<: BinaryDDBase`	`BinaryDDS`	Similar to `BinaryDD`, but with an alternative parameterization of Shapiro delay suitable for almost edge-on systems	(R. R. Rafikov & D. Lai 2006)
`BinaryELL1` `<: BinaryELL1Base`	`BinaryELL1`	Parameterized model for almost circular binaries with relativistic effects	(C. Lange et al. 2001)

Table 2
(Continued)

Component and Base Type	PINT Equivalent(s)	Description	References
BinaryELL1H <: BinaryELL1Base	BinaryELL1H	Similar to BinaryELL1, but with an orthometric representation of Shapiro delay	(P. C. C. Freire & N. Wex 2010)
BinaryELL1k <: BinaryELL1Base	BinaryELL1k	Similar to BinaryELL1, but with an exact treatment of advance of periapsis	(A. Susobhanan et al. 2018)
FrequencyDependent <: FrequencyDependentBase	FD	Phenomenological model for apparent delays caused by unmodeled profile evolution	(Z. Arzoumanian et al. 2015)
FrequencyDependentJump <: FrequencyDependentBase	FDJump	Similar to FrequencyDependent, but accounts for experiment-dependent differences in modeling profile evolution	(A. Susobhanan et al. 2024)
WaveX <: RedNoiseBase	WaveX	Apparent delay due to rotational irregularities of the pulsar represented as an unconstrained Fourier series	(A. Susobhanan et al. 2024)
PowerlawRedNoiseGP <: RedNoiseBase	PLRedNoise	Rotational irregularities of the pulsar represented as a Fourier series Gaussian process with power-law spectrum	(L. Lentati et al. 2014)
Spindown <: PhaseComponent	Spindown	Taylor series representation of the pulsar spin-down	(G. B. Hobbs et al. 2006)
Glitch <: PhaseComponent	Glitch	Pulsar glitches	(G. B. Hobbs et al. 2006)
PhaseOffset <: PhaseComponent	PhaseOffset	Overall initial phase	(A. Susobhanan et al. 2024)
PhaseJump <: PhaseComponent	PhaseJump	System-dependent phase offsets	(G. B. Hobbs et al. 2006)
MeasurementNoise <: WhiteNoiseComponent	ScaleToaErrors	Corrections to the measured TOA uncertainties	(L. Lentati et al. 2014)
DispersionMeasurementNoise <: WhiteNoiseComponent	ScaleDmErrors	Corrections to the measured wideband DM uncertainties	(M. F. Alam et al. 2021)

Note. See Section 2 for an overview of the timing and noise model and Section 3.4 for its implementation in Vela.jl. See Figure 9 for the hierarchy of base types. The "<:" symbol represents "subtype of."

Appendix B
Prior Distributions

In principle, the prior for each parameter should be set strictly based on our prior knowledge. Indeed, we may have prior information on some of the parameters from previous timing experiments, very long baseline interferometry campaigns (e.g., A. T. Deller et al. 2016), detection of counterparts in other parts of the electromagnetic spectrum (e.g., D. L. Kaplan et al. 2018), etc. Or priors may be estimated from population statistics using a catalog like psrcat (R. N. Manchester et al. 2005).

However, for many parameters, pulsar timing provides so much signal-to-noise ratio (S/N) that the effect of the prior on the posterior distribution is entirely negligible provided the prior is sufficiently broad. This is the case for parameters like F_0, F_1, source coordinates, etc. even for small timing data sets. In the context of analytic marginalization of linearized timing model parameters, it is customary to assume uninformative infinitely broad Gaussian priors (R. van Haasteren & Y. Levin 2013). On the other hand, given the insensitivity of the posterior distribution on the priors on some of the timing model parameters, "cheat" priors, namely, uniform distributions centered around the frequentist estimate whose width is several times (e.g., 10x) the frequentist uncertainty, have also been used (e.g., L. Lentati et al. 2014). In cases like the amplitudes of a Fourier series Gaussian process (e.g., PowerlawRedNoiseGP), the priors are defined by the model itself.

Care must be taken to ensure that the data provides enough S/N for the parameter for the "cheat" prior distribution to be valid, lest we effectively do a circular analysis (N. Kriegeskorte et al. 2009). A "cheat" prior can also become invalid if the parameter has a physical range, e.g., the sin-inclination of the binary orbit $\sin\iota \in [0, 1]$, and the frequentist measurement is close to the physical upper/lower bound. (See Table 3 for

```
{
    "EFAC": {
        "distribution": "LogNormal",
        "args": [0.0, 0.5]
    },
    "EQUAD": {
        "distribution": "LogUniform",
        "args": [0.01, 2.0]
    }
    "M2": {
        "distribution": "Normal",
        "args": [1.0, 0.02],
        "lower": 0.0
    }
}
```

Figure 10. An example of the user-defined prior distributions represented in the `JSON` format. "`distribution`" should be a distribution available in the `Distributions.jl` package (M. Besanon et al. 2021). "`args`" contains the arguments to construct a distribution object. The "`upper`" and "`lower`" options allow the distribution to be truncated. Only univariate distributions with constant hyperparameters are supported. The arguments and the upper and lower bounds have the same units used in the `par` files where applicable. For example, the prior distribution for M2 above corresponds to a normal distribution with mean 1 $M_\odot$ and standard deviation 0.02 $M_\odot$ truncated to exclude negative values.

Table 3
Physically Motivated Prior Distributions for Different Parameterizations of the Orbital Inclination ι Assuming that $\cos\iota$ Is Uniformly Distributed in [0, 1]

Parameter Name	Binary Models	Definition	Prior Distribution			
			Support	PDF	CDF	Quantile
KIN	BinaryDDK	Orbital inclination (ι)	$[0, \pi/2]$	$\sin\iota$	$1-\cos\iota$	$\arccos(1-q)$
SINI	BinaryDD	$s = \sin\iota$	[0, 1]	$\frac{s}{\sqrt{1-s^2}}$	$1-\sqrt{1-s^2}$	$\sqrt{(2-q)q}$
	BinaryELL1 BinaryELL1k					
STIGMA	BinaryDDH	$\varsigma = \frac{\sin\iota}{1+\cos\iota}$	[0, 1]	$\frac{4\varsigma}{(1+\varsigma^2)^2}$	$\frac{2\varsigma^2}{1+\varsigma^2}$	$\sqrt{\frac{q}{2-q}}$
	BinaryELL1H					
SHAPMAX	BinaryDDS	$S = -\ln(1-\sin\iota)$	$[0, \infty)$	$\frac{1-e^{-S}}{\sqrt{2e^S-1}}$	$1-e^{-S}\sqrt{2e^S-1}$	$\ln\left[\frac{\sqrt{2q-q^2}+1}{(q-1)^2}\right]$

Note. The support, probability density function (PDF), cumulative distribution function (CDF), and quantile function (relevant for computing the prior transform) are listed. See Table 2 for the definitions of different binary models.

physically motivated priors on different parameterizations of the inclination ι.)

In cases where a physically motivated prior can be analytically derived, `Vela.jl` uses those priors. For other parameters, "cheat" priors with a user-defined width are used by default. Crucially, the user may override any of these defaults with the help of univariate distributions defined in `Distributions.jl`. The `pyvela` interface accepts such user-defined univariate prior distributions in the form of a `JSON` file (see Figure 10 for an example). Finally, if the prior distributions described above are inadequate, the user also has the flexibility of defining their own prior distributions outside the `Vela.jl` framework since `Vela.jl` is not closely coupled to any sampler.

Appendix C
Representation of ECORR

In this section, we describe the representation of the ECORR noise used in `Vela.jl` following A. D. Johnson et al. (2024) and A. Susobhanan et al. (2024).[12] When ECORR is the only correlated contribution to $\boldsymbol{C}$ (i.e., the time-correlated noise components, if any, are included as a delay or phase correction rather than in $\boldsymbol{C}$), assuming that there are no overlapping ECORR groups, $\boldsymbol{C}$ can be written as

$$\boldsymbol{C} = \boldsymbol{N} + \sum_{ab} c_a \boldsymbol{v}_{ab} \boldsymbol{v}_{ab}^T, \tag{C1}$$

where c_a^2 is the ECORR weight, the index a represents the different systems for which the ECORRs are assigned, b represents different observing epochs, and $\boldsymbol{N}$ is a diagonal matrix. The $\boldsymbol{v}_{ab}$ are vectors that have 1s for TOAs belonging to the system a and epoch b, and 0s otherwise. It is clear that $\boldsymbol{C}$ is block-diagonal in this case, and each block $\boldsymbol{C}_{ab}$ can be written as

$$\boldsymbol{C}_{ab} = \boldsymbol{N}_{ab} + c_a \boldsymbol{v}_{ab} \boldsymbol{v}_{ab}^T, \tag{C2}$$

where $\boldsymbol{N}_{ab}$ is the portion of $\boldsymbol{N}$ corresponding to the system a and epoch b. The inverse and determinant of $\boldsymbol{C}_{ab}$ can be written as

$$\boldsymbol{C}_{ab}^{-1} = \boldsymbol{N}_{ab}^{-1} - \frac{c_a^2 \boldsymbol{N}_{ab}^{-1} \boldsymbol{v}_{ab} \boldsymbol{v}_{ab}^T \boldsymbol{N}_{ab}^{-1}}{1 + c_a^2 \boldsymbol{v}_{ab}^T \boldsymbol{N}_{ab}^{-1} \boldsymbol{v}_{ab}}, \tag{C3}$$

$$\det \boldsymbol{C}_{ab} = \det \boldsymbol{N}_{ab} \times (1 + c_a^2 \boldsymbol{v}_{ab}^T \boldsymbol{N}_{ab}^{-1} \boldsymbol{v}_{ab}) . \tag{C4}$$

[12] Please note that there are typos in Equations (21)–(23) of A. Susobhanan et al. (2024) where factors of c_a^2 are missing. The expressions below show the correct versions.

Defining an inner product $(\boldsymbol{x}_{ab}|\boldsymbol{y}_{ab}) = \boldsymbol{x}_{ab}^T \boldsymbol{N}_{ab}^{-1} \boldsymbol{y}_{ab}$ and writing the log-likelihood function as a sum $\ln L = \sum_{ab} \ln L_{ab}$, we can write

$$\ln L_{ab} = -\frac{1}{2}\left\{(\boldsymbol{r}_{ab}|\boldsymbol{r}_{ab}) + \ln\det \boldsymbol{N}_{ab} - \frac{c_a^2(\boldsymbol{r}_{ab}|\boldsymbol{v}_{ab})^2}{1 + c_a^2(\boldsymbol{v}_{ab}|\boldsymbol{v}_{ab})} + \ln\left[1 + c_a^2(\boldsymbol{v}_{ab}|\boldsymbol{v}_{ab})\right]\right\}. \tag{C5}$$

Since $\boldsymbol{N}_{ab}$ is diagonal, the inner products appearing in Equation (C5) can be evaluated with linear time complexity with a single pass over the TOAs without the need for any dynamic memory allocations. Further, it is straightforward to see that the evaluation of $\ln L$ can be trivially parallelized over the ECORR groups ab.

Appendix D
Representation of Red Noise Processes

We represent the red noise processes affecting the TOAs as delays represented by truncated Fourier series

$$\Delta_{\mathrm{RN};\alpha}(t) = \left(\frac{\nu_{\mathrm{ref}}}{\nu}\right)^{\alpha} \sum_{j=0}^{N_{\mathrm{harm}}} \left[a_j \cos(2\pi j f_1 (t - t_0)) + b_j \sin(2\pi j f_1 (t - t_0))\right], \tag{D1}$$

where f_1 is a fundamental frequency, α is the chromatic index, $\alpha = 0$ represents spin noise, and $\alpha = 2$ represents dispersion noise. f_1 is usually taken to be the reciprocal of the total observation span of the data set.

We provide two types of representation for red noise. `WaveX` (spin noise), `DMWaveX` (dispersion noise), and `CMWaveX` (variable-α chromatic noise) treat the coefficients a_j and b_j as unconstrained free parameters with uninformative priors. This representation is useful for reconstructing cross–pulsar correlations from single-pulsar noise analysis runs post facto (S. Valtolina & R. van Haasteren 2024). (This will be explored in a future work.)

The Gaussian process models `PowerlawRedNoiseGP`, `PowerlawDispersionNoiseGP`, and `PowerlawChromaticNoiseGP` impose Gaussian prior distributions on the Fourier coefficients such that $\langle a_j \rangle = \langle b_j \rangle = 0$, $\langle a_j a_k \rangle = \langle b_j b_k \rangle = \sigma_j^2 \delta_{jk}$, $\langle a_j b_k \rangle = 0$. Further, the spectral power densities σ_j follow a power-law spectrum

$$\sigma_j^2 = \frac{A^2}{12\pi^2 f_{\mathrm{yr}}^3} f_1 \left(\frac{f_{\mathrm{yr}}}{j f_1}\right)^{\gamma}. \tag{D2}$$

Note that certain physical models of the spin noise introduce a low-frequency turnover in its spectrum (B. Goncharov et al. 2020). While this power spectral model is currently unavailable, it will be included in a future version of `Vela.jl`.

It turns out that the joint prior distribution of a_j or b_j and the power-law parameters A and γ displays Neal's funnel-like geometry (R. M. Neal 2003), which is hard for MCMC samplers to explore. We handle this by treating $\bar{a}_j = a_j/\sigma_j$ and $\bar{b}_j = b_j/\sigma_j$ as free parameters rather than a_j and b_j. It is easy to see that these new parameters are a priori unit-normal distributed. This also has the advantage of simplifying the implementation of prior transform functions by ensuring that the prior distribution of each parameter is independent of the other parameters.

Note that these Gaussian process models have $2N_{\mathrm{harm}} + 2$ number of parameters. This leads to high-dimensional parameter spaces, which can be challenging to sample. On the other hand, treating the time-correlated noise processes as delays has the advantage that the log-likelihood can be evaluated in linear time without memory allocations (see Appendix C). The sampling challenges posed by the large number of Fourier coefficients can be addressed somewhat by employing Gibbs sampling for those parameters, e.g., N. Laal et al. (2023), since it turns out that the conditional distributions for these parameters can be analytically derived. This will be explored in a future work.

ORCID iDs

Abhimanyu Susobhanan https://orcid.org/0000-0002-2820-0931

References

Agazie, G., Anumarlapudi, A., Archibald, A. M., et al. 2023a, ApJL, 951, L10
Agazie, G., Anumarlapudi, A., Archibald, A. M., et al. 2023b, ApJL, 951, L8
Agazie, G., Antoniadis, J., Anumarlapudi, A., et al. 2024, ApJ, 966, 105
Ajello, M., Atwood, W. B., Baldini, L., et al. 2022, Sci, 376, 521
Alam, M. F., Arzoumanian, Z., Baker, P. T., et al. 2021, ApJ, 252, 5
Antonelli, M., Montoli, A., & Pizzochero, P. M. 2022, in Astrophysics in the XXI Century with Compact Stars, ed. C. A. Z. Vasconcellos (Singapore: World Scientific), 219
Arzoumanian, Z., Brazier, A., Burke-Spolaor, S., et al. 2015, ApJ, 813, 65
Arzoumanian, Z., Fruchter, A. S., & Taylor, J. H. 1994, ApJL, 426, L85
Ashton, G., Bernstein, N., Buchner, J., et al. 2022, Nature Reviews Methods Primers, 2, 39
Backer, D. C., & Hellings, R. W. 1986, ARA&A, 24, 537
Barbary, K. 2021, nestle: Nested Sampling Algorithms for Evaluating Bayesian Evidence, http://kylebarbary.com/nestle/
Besanon, M., Papamarkou, T., Anthoff, D., et al. 2021, Journal of Statistical Software, 98, 1
Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. 2017, SIAMR, 59, 65
Chen, S., Caballero, R. N., Guo, Y. J., et al. 2021, MNRAS, 508, 4970
Cromartie, H. T., Fonseca, E., Ransom, S. M., et al. 2020, NatAs, 4, 72
Damour, T., & Deruelle, N. 1986, AIHPA, 44, 263
Dekker, T. 1971, NuMat, 18, 224
Deller, A. T., Vigeland, S. J., Kaplan, D. L., et al. 2016, ApJ, 828, 8
Diaconis, P. 2009, BAMaS, 46, 179
Edwards, R. T., Hobbs, G. B., & Manchester, R. N. 2006, MNRAS, 372, 1549
Ellis, J., & van Haasteren, R. 2017, PAL2 (PTA Algorithm Library), https://github.com/jellis18/PAL2
Estabrook, F. B., & Wahlquist, H. D. 1975, GReGr, 6, 439
Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601
Foreman-Mackey, D. 2016, JOSS, 1, 24
Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306
Foster, R. S., & Backer, D. C. 1990, ApJ, 361, 300
Freedman, G. E., Johnson, A. D., van Haasteren, R., & Vigeland, S. J. 2023, PhRvD, 107, 043013
Freire, P. C. C., Ransom, S. M., Bégin, S., et al. 2008, ApJ, 675, 670
Freire, P. C. C., & Wex, N. 2010, MNRAS, 409, 199
Goncharov, B., Zhu, X.-J., & Thrane, E. 2020, MNRAS, 497, 3264
Gropp, W., Lusk, E., Doss, N., & Skjellum, A. 1996, ParC, 22, 789
Handley, W. J., Hobson, M. P., & Lasenby, A. N. 2015, MNRAS, 450, L61
Hankins, T. H., & Rickett, B. J. 1986, ApJ, 311, 684
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
Hazboun, J. S., Simon, J., Madison, D. R., et al. 2022, ApJ, 929, 39
Hellings, R. W., & Downs, G. S. 1983, ApJL, 265, L39
Hemberger, D. A., & Stinebring, D. R. 2008, ApJL, 674, L37
Hobbs, G. 2014, TEMPO2 examples, https://www.jb.man.ac.uk/~pulsar/Resources/tempo2_examples_ver1.pdf
Hobbs, G. B., Edwards, R. T., & Manchester, R. N. 2006, MNRAS, 369, 655
Hunter, J. D. 2007, CSE, 9, 90
Innes, M. 2018, arXiv:1810.07951
Johnson, A. D., Meyers, P. M., Baker, P. T., et al. 2024, PhRvD, 109, 103012
Kaplan, D. L., Stovall, K., van Kerkwijk, M. H., Fremling, C., & Istrate, A. G. 2018, ApJ, 864, 15
Klioner, S., Capitaine, N., Folkner, W., et al. 2009, in IAU Symp. 261, Relativity in Fundamental Astronomy: Dynamics, Reference Frames, and Data Analysis (Cambridge: Cambridge Univ. Press), 883

Kopeikin, S. M. 1995, ApJL, 439, L5
Kopeikin, S. M. 1996, ApJL, 467, L93
Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S. F., & Baker, C. I. 2009, Nature Neuroscience, 12, 535
Krishnakumar, M. A., Manoharan, P. K., Joshi, B. C., et al. 2021, A&A, 651, A5
Krishnakumar, M. A., Mitra, D., Naidu, A., Joshi, B. C., & Manoharan, P. K. 2015, ApJ, 804, 23
Laal, N., Lamb, W. G., Romano, J. D., et al. 2023, PhRvD, 108, 063008
Lange, C., Camilo, F., Wex, N., et al. 2001, MNRAS, 326, 274
Large, M. I., Vaughan, A. E., & Mills, B. Y. 1968, Natur, 220, 340
Lentati, L., Alexander, P., & Hobson, M. P. 2015, MNRAS, 447, 2159
Lentati, L., Alexander, P., Hobson, M. P., et al. 2014, MNRAS, 437, 3004
Lorimer, D. R., & Kramer, M. 2012, Handbook of Pulsar Astronomy (Cambridge: Cambridge Univ. Press)
Luo, J., Ransom, S., Demorest, P., et al. 2021, ApJ, 911, 45
Manchester, R. N., Hobbs, G. B., Teoh, A., & Hobbs, M. 2005, AJ, 129, 1993
Neal, R. M. 2003, AnSta, 31, 705
Parthasarathy, A., Bailes, M., Shannon, R. M., et al. 2021, MNRAS, 502, 407
Pennucci, T. T. 2019, ApJ, 871, 34
Pennucci, T. T., Demorest, P. B., & Ransom, S. M. 2014, ApJ, 790, 93
Perera, B. B. P., Barr, E. D., Mickaliger, M. B., et al. 2019, MNRAS, 487, 1025
Perera, B. B. P., DeCesar, M. E., Demorest, P. B., et al. 2019, MNRAS, 490, 4666
Pletsch, H. J., & Clark, C. J. 2015, ApJ, 807, 18
Rafikov, R. R., & Lai, D. 2006, PhRvD, 73, 063003
Robitaille, T. P., Tollerud, E. J., Greenfield, P., et al. 2013, A&A, 558, A33
Rowley, C. 2022, PythonCall.jl: Python and Julia in Harmony, https://github.com/JuliaPy/PythonCall.jl
Sarnoff, J., et al. 2022, DoubleFloats, 1.2.2, https://github.com/JuliaMath/DoubleFloats.jl
Singha, J., Surnis, M. P., Joshi, B. C., et al. 2021, MNRAS, 507, L57
Susarla, S. C., Chalumeau, A., Tiburzi, C., et al. 2024, A&A, 692, A18
Susobhanan, A. 2025, Vela.jl: A Bayesian Pulsar Timing and Noise Analysis Package, v0.0.7, Zenodo, doi:10.5281/zenodo.14609404
Susobhanan, A., Gopakumar, A., Joshi, B. C., & Kumar, R. 2018, MNRAS, 480, 5260
Susobhanan, A., Kaplan, D. L., Archibald, A. M., et al. 2024, ApJ, 971, 150
Tarafdar, P., Nobleson, K., Rana, P., et al. 2022, PASA, 39, e053
Taylor, J. H. 1992, RSPTA, 341, 117
Tiburzi, C., Hobbs, G., Kerr, M., et al. 2015, MNRAS, 455, 4339
Tiburzi, C., Shaifullah, G. M., Bassa, C. G., et al. 2021, A&A, 647, A84
Vallisneri, M., Taylor, S. R., Simon, J., et al. 2020, ApJ, 893, 112
Valtolina, S., & van Haasteren, R. 2024, arXiv:2412.11894
van Haasteren, R., 2016 Piccard: Pulsar Timing Data Analysis Package, Astrophysics Source Code Library, ascl:1610.001
van Haasteren, R., & Levin, Y. 2013, MNRAS, 428, 1147
van Haasteren, R., & Vallisneri, M. 2014, MNRAS, 446, 1170
Vigeland, S. J., & Vallisneri, M. 2014, MNRAS, 440, 1446
Voisin, G., Cognard, I., Freire, P. C. C., et al. 2020, A&A, 638, A24
Weisberg, J. M., & Huang, Y. 2016, ApJ, 829, 55