Master's Thesis

PTA Parameter Inference with Neural Networks

Submitted by

Jan-Christopher Knetsch

June 24, 2025

First examiner
Prof. Dr. Kai Schmitz

Second examiner
Dr. Tomáš Ježo

Universität Münster
Department of Physics
Institute of Theoretical Physics

Contents

Acknowledgements

I would like to thank my supervisor, Kai Schmitz, for his guidance and the constructive feedback he gave in the preparation of this thesis.
I am particularly grateful to David Shih for providing the code base and foundational work on which this thesis is built.
I also thank Theo Heimel for the continuous and advice and discussions on all machine learning-related aspects.
My thanks go to Tilman Plehn for the opportunity to visit his research group in Heidelberg, as well as for his valuable input to this thesis.
Further, I would like to thank Stephen Taylor for his valuable input on pulsar timing during the course of this work.
I also thank Tomáš Ježo for taking on the role of second supervisor for this project.
Finally, I thank Katrin Greve and Richard von Eckardstein for their helpful comments and proofreading assistance.

1 Introduction

The direct detection of gravitational waves in 2015 by the Laser Interferometer Gravitational-Wave Observatory (LIGO) marked a pivotal moment in modern physics. Resulting in the award of a Nobel Prize to Barry C. Barish, Kip S. Thorne and Rainer Weiss in 2017 [Nobel17] it marked the opening of a new observational window into the universe. Predicted by Einstein's theory of general relativity in 1916 and published in 1918 [Ein18], gravitational waves, carrying information about some of the most energetic phenomena in the cosmos, allow us to deepen our understanding of previously enigmatic events such as mergers of black holes and neutron stars [Jia24]. Besides signals of astrophysical origin there is hope to find signals of cosmological origin. Their detection could allow the discovery of physics beyond the standard model.

Today the field of gravitational wave experiments is vast, encompassing high precision interferometer detectors such as LIGO [LIG], Virgo [Vir] and KAGRA [KAG]. Since 2023, pulsar timing arrays, a new type of gravitational wave experiment, have managed to establish itself on the scientific stage. With statistic significances of $(2 - 4.6)\,\sigma$ five pulsar timing collaborations, the chinese CPTA [Xu+23], the european EPTA [Ant+23] the indian InPTA [Sri+23], the US-american NANOGrav [Aga+23] and the australian PPTA [Rea+23] collaboration simultaneously published their finding of a Hellings-Downs-correlated stochastic gravitational wave background. While terrestrial and space-based interferometers are sensitive to gravitational waves in the frequency range of $(10^{-4} - 10^{4})\,\mathrm{Hz}$, PTAs are sensitive to lower frequencies of $(10^{-9} - 10^{-6})\,\mathrm{Hz}$. It is in this range that some possible extensions of the standard model such as cosmic inflation, phase transitions in the early universe and cosmic strings predict gravitational wave signals [Tay21]. This way PTAs may allow to observationally probe and constrain beyond the standard model theories where conventional collider experiments still lack the necessary collision energies. In order for PTA observations to allow the preference of one theory over another with relative certainty large amounts of data are required. For this reason all collaborations continuously extend both the number of individually observed pulsars as well as the length of the observation period. Additionally there are efforts being made to unify the datasets gathered by the individual collaborations in the international pulsar timing array IPTA [Per+19].

While the addition of new data to the datasets allows for more precise constraints, it simultaneously increases the computational cost of analyses. In the future this may prove limiting to the scope of analyses that may feasibly be undertaken. Currently a full Bayesian posterior analysis of the NANOGrav 15 year dataset takes time on the order of weeks. For this reason there is a continued effort being made to find new ways of accelerating the analysis [LTH23]. In particular the investigation of the application of machine learning techniques to PTA analyses has recently begun.

Machine learning and in particular neural networks have managed to attract a lot of attention in recent years. With the development of the transformer architecture by Anish Vaswani e.a. in 2023 [Vas+23] and the following success of large language models, machine learning indicates potential to significantly change modern society. At the same time it has left a deep mark on modern physics, the impact of which has led to the award of Nobel Prize to John J. Hopfield and Geoffrey Hinton in 2024 [Nobel24]. At the core neural networks are highly expressive and flexible functions dependant on a large amount of parameters (here $\mathcal{O}(10^6)$), that are optimized on a specific problem commonly using modern high performance computing infrastructure. Similarly to many other fields, the gravitational wave community has started to make attempts to use machine learning techniques e.g. in event detection [GH17; Gab+18] and for the analysis of deterministic signals [Dax+23]. Recent works, pioneered by [Shi+23] and more recently by

[LL25] have also begun using neural networks in the analysis of data from stochastic orgins and in particular pulsar timing arrays. The main aim being to accelerate the often computationally expensive Bayesian inference process.

The basis for this work is provided by Shih et al. [Shi+23] where they used a normalizing flow generative neural network [RM16] in combination with an LSTM encoder network [SSB14] to perform simulation-based parameter inference [PM18] on simulated data based on the NANOGrav 12.5 yr dataset [Aga+23] as a proof of concept. They explore the performance of the network on a constant power law (CPL) model, which corresponds to the noise spectrum predicted for an SGWB of astrophysical origin [Phi01]. They managed to accurately recover the posterior distribution on simulated timing series and conclude that machine learning has great potential to supplement conventional Markov Chain Monte Carlo methods in Bayesian parameter inference.

In this work we aim to continue their work both on the machine learning and on the noise modeling side. We explore the performance of a transformer encoder neural network [Vas+23] as surrogate for the LSTM encoder and find that it manages to produce more accurate posterior distributions. We also explore the usage of diffusion networks [HJA20], another class of generative neural networks in PTA parameter inference, where we do not find any improvement over normalizing flows. On the noise modeling side we explore different SGWB noise models from the CPL. We first explore higher order polynomial models [Aga+25] that while not directly physically motivated can be used as effective and flexible descriptions of the measured data. In particular the encoder networks trained off such a flexible model might in the future be used as a general preprocessing method for any type of noise model, thereby avoiding the need to retrain an expensive encoder network for every noise model of interest. Moreover, we apply neural parameter inference to a more complex noise model predicted by phase transitions in the early universe [Afz+23] and find that neural parameter inference succeeds in recovering induced parameter values from simulated data, further increasing its attractiveness as a potential tool in general PTA data analyses.

2 Bayesian Inference

This section is based on [Bol07] chapters 8 and 9. The question we are trying to answer in this work will be the following: Given a mathematical model dependent on previously unknown parameters, that makes observable predictions, what is the parameter configuration that "best describes" our observations? Secondly, given multiple models, which model makes "more accurate" predictions for our observations? There are generally two schools of thought in statistics that differ mainly in their understanding and mathematical description of ignorance: The *frequentist* paradigm and the *Bayesian* paradigm. Throughout this work we will usually use the Bayesian statistical framework but both will be briefly explained in the following.

Independent of statistical paradigm, the mathematical model is a key concept that will be used throughout this work. A mathematical model M has the general functional form

$$y = F_M(x, u, \theta) + n, \tag{1}$$

(see [RHB02], ch. 27). Here y is a measurable quantity for which the model makes predictions within an experimental or observational setup. Since the arguments of the functional relation F_M may themselves be random variables and the accuracy of the observation is in practice always limited by the measuring accuracy, y is treated as a random variable. x describes the known properties of the setup and it will analogously be treated as a random variable. u represents additional possible unknown properties of the setup. u is also treated as a random variable and in practical applications it will have to be averaged over. Lastly, θ describes free parameters of the model. It is in the treatment of model parameters where the main difference between the Bayesian- and the frequentist statistical framework lies. Whereas the frequentist approach is to treat model parameters as unknown, but fixed, quantities, the Bayesian approach also treats them as random variables. This is done by assigning probabilities to them based on subjective prior belief. This prior belief may be shaped by experiments in the past. In principle, there is no wrong choice of distribution as long as it truly corresponds to the prior belief. Mathematically, we will denote the prior distribution of the parameters as $p_0(\theta)$.

Having defined all input quantities of our model we can now use the algebraic properties of random variables to calculate several probability distributions for the model. For an overview of algebraic properties of random variables and their probability distributions see appendix A.1. The most relevant distributions will be outlined in the following:

- The conditional probability distribution of the observable quantity y conditioned upon the parameter θ

$$p(y|\theta) = \int\int dx\, du\, p(y|x, u, \theta)\, p_{x,u}(x, u), \tag{2}$$

  where $p(y|x, u, \theta)$ is obtained from the functional form of the model. Here the parameter is taken to be a fixed quantity and the probability density of y is given relative to that fixed value. All other input quantities are averaged over to remove the dependence on them. $p(y|\theta)$ is normalized such that it is a probability density in y, i.e.,

$$\int dy\, p(y|\theta) = 1, \quad \forall\, \theta.$$

  Conversely $p(y|\theta)$ is not a probability distribution in θ, but it is commonly considered as a function of θ for a fixed value of y. In the latter case $p(y|\theta)$ is also called the likelihood

function. Since it does not treat θ as a random variable, it is used both in frequentist and Bayesian statistics.

- In the Bayesian framework, one can assign a probability density to y which is given as

$$p(y) = \int d\theta \, p(y|\theta) \, p_0(\theta). \tag{3}$$

There necessarily is an implied dependence on the choice of functional form for the model and the choice of prior distribution for the parameter. In practical applications it allows to compare different functional forms based on measured data. $p(y)$ is commonly called the (Bayesian) evidence.

- Lastly, in the Bayesian framework, one can use the theorem of Bayes for probability densities

$$p(x|y) = \frac{p(y|x) \, p(x)}{p(y)}$$

and use it to give an expression for the conditional probability distribution for the parameter θ conditioned upon a fixed y under the model as

$$p(\theta|y) = \frac{p(y|\theta) \, p_0(\theta)}{p(y)}. \tag{4}$$

This represents a subjective rational belief about the probability of parameter values, given the assumed functional form and prior distribution, after observing the measured data. In the limit of very large sample sizes, the posterior distribution is dominated by the likelihood, and therefore becomes independent of the choice of prior (cf.[Bol07] chapter 8).

One common goal of statistical analyses is to infer the numerical value of a model parameter from measured data. This involves finding the value of the parameter under which the model "best" describes the measured data. However, the term "best" is not well defined in this case due to the random nature of the sampled data. To address this, different estimators $\hat{\theta}$ for the parameter value are employed that can be computed from the data. Common Bayesian estimates are the posterior mean ,eq. (A9), and the posterior median, eq. (A10). Particularly in frequentist statistics, one also uses the maximum likelihood estimator, which does not require a prior distribution. One can see that in the limit of large datasets, maximum likelihood and maximum posterior value are identical. Once an estimator is chosen, it is also necessary to quantify the reliability of the estimate. The posterior mean square (PMS) of an estimate $\hat{\theta}$, defined as

$$\mathrm{PMS}(\hat{\theta}) = \int (\theta - \hat{\theta})^2 \, p(\theta|y) \, d\theta \tag{5}$$

is a common measure. It measures the average squared deviation from the true value, assuming the validity of the posterior distribution. It can be shown that the posterior mean estimator minimizes the PMS. In this case, the PMS equals the variance of the posterior distribution. In some cases, it may not be necessary to find a specific estimator for the parameter values. Instead, one might be interested in identifying a region of the parameter space that contains the true value with high probability. Such regions can be found by using the posterior distribution

to compute credible regions (or credible intervals in the univariate case). An r-credible region I_r is a set in parameter space that has a known posterior probability of r

$$\int_{I_r} p(\theta|y)\, d\theta = r. \tag{6}$$

The choice of region is not unique. The most common choice is the smallest region in terms of its euclidean volume. This corresponds to finding the smallest set with equal posterior values at its boundary and a probability of r. This choice of credible region is the one we will adopt in this work.

Another important statistical objective is the comparison of different models, that is, finding the model that best describes the measured data. There are frequentist and Bayesian criteria that allow to accept or to reject a given model. In frequentist statistics, this is done using *hypothesis testing* (see [Bol07] chapter 9.5). The Bayesian paradigm allows to directly compare two models M_1 and M_2 by computing the quotient of the Bayesian evidences from eq. (3)

$$\mathrm{BF}(M_1, M_2) = \frac{p_{M_1}(y)}{p_{M_2}(y)}. \tag{7}$$

One should note that the evidence is highly dependant on the chosen prior distribution, which can reduce its usefulness in the case of uninformed prior distributions.

3 Gravitational Waves

This chapter is based on chapter 1 of [Mag07]. Gravitational waves (GWs) are a direct prediction of the theory of general relativity [Ein18]. In this chapter we will demonstrate how their mathematical description can be derived starting from the Einstein field equations of general relativity.

3.1 The Linearized Metric

In oder to solve the Einstein equations, we use as our ansatz a linearized theory around a flat spacetime with a small perturbation in which the metric is given as

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}$$

with $\eta_{\mu\nu}$ being the Minkowski metric in $(-,+,+,+)$-convention and $h_{\mu\nu}$ a small perturbation with $h_{\mu\nu} \ll 1$ as well as $\mathcal{O}(\partial_\lambda h_{\mu\nu}) = \mathcal{O}(h_{\mu\nu})$ and $\mathcal{O}(\partial_\lambda \partial_\sigma h_{\mu\nu}) = \mathcal{O}(h_{\mu\nu})$. Since the numerical values of a tensor depend on the choice of coordinates, this assumption implies that there exists a coordinate system in which these conditions hold in sufficiently large regions.

The Einstein field equations are given as

$$G_{\mu\nu} \coloneqq R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} = 8\pi T_{\mu\nu}$$

where $G_{\mu\nu}$ is the Einstein tensor, $R_{\mu\nu}$ the Ricci tensor, R the Ricci scalar and $T_{\mu\nu}$ the stress-energy tensor. Everything is written in natural units, i.e. $G = c = \hbar = 1$. Inserting the linearized metric ansatz we obtain

$$8\pi T_{\mu\nu} = \frac{1}{2}(\partial_\mu \partial^\alpha \bar{h}_{\alpha\nu} + \partial_\nu \partial^\alpha \bar{h}_{\alpha\mu} - \Box \bar{h}_{\mu\nu} - \eta_{\mu\nu} \partial^\alpha \partial^\eta \bar{h}_{\alpha\beta}). \tag{8}$$

Here we have introduced the trace-reversed perturbation $\bar{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu} h$, to write the equation in a more compact form and dropped all terms of $\mathcal{O}(h_{\mu\nu} h_{\alpha\beta})$ and higher since we are working in the linearized theory. It follows directly that $\mathcal{O}(\bar{h}_{\mu\nu}) = \mathcal{O}(h_{\mu\nu})$ and which implies that our ansatz can only be valid in physical cases where the stress-energy tensor also has $\mathcal{O}(h_{\mu\nu})$.

The conditions imposed so far do not uniquely fix our choice of coordinate system. We can therefore use the remaining freedom in the choice of coordinates to bring the equations into a simpler form. To do this we consider coordinate transformations of the form

$$x^\mu \to x'^\mu = x^\mu + \xi^\mu(x) \tag{9}$$

for which we demand that $\mathcal{O}(\partial_\lambda \xi^\mu) = \mathcal{O}(h_{\mu\nu})$. The transformed metric can then be calculated using

$$g'_{\mu\nu}(x') = \frac{\partial x^\rho}{\partial x'^\mu} \frac{\partial x^\sigma}{\partial x'^\nu} \, g_{\rho\sigma}(x).$$

To do this we first compute

$$\begin{aligned}\frac{\partial x^\mu}{\partial x'^\nu} &= \frac{\partial\left[x'^\mu - \xi^\mu(x' - \xi(x))\right]}{\partial x'^\nu} \\ &= \frac{\partial\left[x'^\mu - \xi^\mu(x') + \frac{\partial\xi^\mu}{\partial x^\lambda}(x')\xi^\lambda(x)\right]}{\partial x'^\nu} \\ &= \frac{\partial\left[x'^\mu - \xi^\mu(x')\right]}{\partial x'^\nu} + \mathcal{O}(|\partial_\lambda\xi^\mu|^2) \\ &= \eta^\mu_\nu - \frac{\partial\xi^\mu(x')}{\partial x'^\nu} + \mathcal{O}(|\partial_\lambda\xi^\mu|^2).\end{aligned}$$

This allows to further compute

$$\begin{aligned}\frac{\partial\xi^\mu(x')}{\partial x'^\nu} &= \frac{\partial\left[\xi^\mu(x) + \frac{\partial\xi^\mu}{\partial x^\lambda}(x')\xi^\lambda(x)\right]}{\partial x^\lambda}\frac{\partial x^\lambda}{\partial x'^\nu} \\ &= \frac{\partial\xi^\mu(x)}{\partial x^\lambda}\left[\eta^\lambda_\nu - \frac{\partial\xi^\lambda(x')}{\partial x'^\nu}\right] + \mathcal{O}(|\partial_\lambda\xi^\rho|^2) \\ &= \frac{\partial\xi^\mu(x)}{\partial x^\nu} + \mathcal{O}(|\partial_\lambda\xi^\rho|^2).\end{aligned}$$

Remembering that $\mathcal{O}(\partial_\lambda\xi^\mu) = \mathcal{O}(h_{\mu\nu})$ the transformed metric is then given as

$$\begin{aligned}g'_{\mu\nu}(x') &= \left[\eta^\rho_\mu - \frac{\partial\xi^\rho(x')}{\partial x'^\mu}\right]\left[\eta^\sigma_\nu - \frac{\partial\xi^\sigma(x')}{\partial x'^\nu}\right](\eta_{\rho\sigma} + h_{\rho\sigma}) + \mathcal{O}(|h|^2) \\ &= \eta_{\mu\nu} + h_{\mu\nu}(x) - \frac{\partial\xi_\nu(x')}{\partial x'^\mu} - \frac{\partial\xi_\mu(x')}{\partial x'^\nu} + \mathcal{O}(|h|^2) \\ &= \eta_{\mu\nu} + h_{\mu\nu}(x) - \frac{\partial\xi_\nu(x)}{\partial x^\mu} - \frac{\partial\xi_\mu(x)}{\partial x^\nu} + \mathcal{O}(|h|^2) \\ &=: \eta_{\mu\nu} + h'_{\mu\nu}(x').\end{aligned}$$

We note that demanded condition of smallness for $h_{\mu\nu}$ remains preserved under such transformations, and, similarly, the trace reversed metric transforms as

$$\bar{h}'_{\mu\nu}(x') = \bar{h}_{\mu\nu}(x) - \partial_\nu\xi_\mu(x) - \partial_\mu\xi_\nu(x) + \eta_{\mu\nu}\partial_\lambda\xi^\lambda(x),$$

which implies

$$\begin{aligned}\left(\partial^\nu\bar{h}_{\mu\nu}(x)\right)' = \left(\frac{\partial x_\rho}{\partial x'_\nu}\right)\partial^\rho\bar{h}'_{\mu\nu}(x') &= \left[\eta^\nu_\rho - \partial^\nu\xi_\rho(x)\right]\partial^\rho\bar{h}'_{\mu\nu}(x') + \mathcal{O}(|h|^2) \\ &= \partial^\nu\left[\bar{h}_{\mu\nu}(x) - \partial_\nu\xi_\mu(x) - \partial_\mu\xi_\nu(x) + \eta_{\mu\nu}\partial_\lambda\xi^\lambda(x)\right] + \mathcal{O}(|h|^2) \\ &= \partial^\nu\bar{h}_{\mu\nu}(x) - \Box\xi_\mu(x) + \mathcal{O}(|h|^2).\end{aligned}$$

We now set $f_\mu(x) \coloneqq \partial^\nu\bar{h}_{\mu\nu}(x)$ and since the d'Alembert operator is invertible we can choose ξ_μ such that $\Box\xi_\mu(x) = f_\mu(x)$ which by definition fulfills the demanded condition of smallness on ξ_μ. In the new coordinate frame it follows

$$\left(\partial^\nu\bar{h}_{\mu\nu}(x)\right)' = 0. \tag{10}$$

The stress-energy tensor transforms as

$$\begin{aligned} T'_{\mu\nu} &= \frac{\partial x^\rho}{\partial x'^\mu}\frac{\partial x^\sigma}{\partial x'^\nu}T_{\rho\sigma} = \left[\eta^\rho_\mu - \frac{\partial \xi^\rho(x')}{\partial x'^\mu}\right]\left[\eta^\sigma_\nu - \frac{\partial \xi^\sigma(x')}{\partial x'^\nu}\right]T_{\rho\sigma} + \mathcal{O}(|h|^2) \\ &= T_{\mu\nu} + \mathcal{O}(|h|^2) \end{aligned} \tag{11}$$

since $\mathcal{O}(T_{\mu\nu}) = \mathcal{O}(h)$.

If we reinsert eqs. (10) and (11) into eq. (8), we obtain the wave equation form of the linearized Einstein equations

$$\Box\bar{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu}. \tag{12}$$

We consider the case of gravitational waves propagating in vacuum outside of gravitational sources. In this case the equations simply become

$$\Box\bar{h}_{\mu\nu} = 0. \tag{13}$$

One should note here that, in practice, the assumption $T_{\mu\nu} = 0$ usually does not hold. Depending on the observation, numerous gravitational sources such as the mass of the earth or other stellar and planetary bodies are present. These sources introduce additional correction terms that must be accounted for when trying to make precise, observable predictions. The necessary corrections in the case of Pulsar Timing Arrays will be briefly discussed in section 4.4.

3.2 Transverse Traceless Gauge

With the previously introduced coordinate transformation we have not yet fully exhausted the gauge freedom inherent in the linearized Einstein equations. We can introduce a second coordinate transformation of the form

$$x^\mu \to x'^\mu = x^\mu + \zeta^\mu(x) \tag{14}$$

where we demand that $\zeta(x)$ fulfills the same conditions of smallness as $\xi(x)$ and additionally that $\Box\zeta_\mu(x) = 0$. All tensorial quantities we considered previously transform in the same way as for $\xi(x)$ and thus

$$\begin{aligned} \Box\bar{h'}_{\mu\nu}(x) = \Box\left[\bar{h}_{\mu\nu}(x) - \zeta_{\mu\nu}(x)\right] = \Box\bar{h}_{\mu\nu}(x), \quad \text{where} \\ \zeta_{\mu\nu}(x) \coloneqq \partial_\nu\zeta_\mu(x) - \partial_\mu\zeta_\nu(x) + \eta_{\mu\nu}\partial_\lambda\zeta^\lambda(x) \end{aligned}$$

since $\Box$ commutes with ∂_μ. Furthermore we see that the trace of the transformed metric is given as

$$\bar{h'}^\mu_\mu = \bar{h}^\mu_\mu - \partial^\mu\zeta_\mu(x) + \partial_\mu\zeta^\mu(x) - \eta^\mu_\mu\partial_\lambda\zeta^\lambda(x) = \bar{h} - 4(\partial_0\zeta^0(x) + \partial_i\zeta^i(x)).$$

We then choose ζ^0 such that

$$\partial_0\zeta^0(x) = \frac{\bar{h}}{4} - \partial_i\zeta^i(x), \tag{15}$$

which implies $\bar{h}' = h' = 0$ and therefore $\bar{h}_{\mu\nu} = h_{\mu\nu}$.

We can impose further conditions via the remaining components by choosing ζ_μ such that

$$\partial^0 \zeta^i = \partial^i \zeta^0 - h^{0i}, \tag{16}$$

which leads to

$$h'^{0i} = h^{0i} - \partial^i \zeta^0 + \partial^0 \zeta^i - \eta^{0\,i} \partial_\lambda \zeta^\lambda = 0.$$

For notational simplicity we now drop the prime on the the transformed quantities. Then the previously demanded condition in eq. (10) for $h_{0\nu}$ becomes

$$0 = \partial^\nu h_{0\nu} = \partial^0 h_{00} + \partial^i h_{0i} = \partial^0 h_{00}.$$

We see that h_{00} is constant in time in leading order and therefore does not correspond to the gravitational wave part of the metric but to a static Newtonian potential. If we assume that we are far enough away from such potentials we can assume $h_{00} = 0$.

In total we now have, via the two coordinate transformations, found a coordinate system in which the following conditions on the metric hold

$$h^{0\mu} = 0, \quad h^i_i = 0, \quad \text{and } \partial^j h_{ij} = 0. \tag{17}$$

This coordinate system is called transverse traceless gauge(TT gauge) and the metric in this coordinate system is denoted as h^{TT}_{ij}. It is not possible to impose any further conditions via additional coordinate transformations without loosing the properties we have demanded so far except a remaining reduced Poincaré symmetry that only includes boost transformations that do not violate the smallness conditions imposed on h^{TT}_{ij}.

In TT gauge the Einstein equations have the form

$$\left[-\partial_0 \partial^0 + \partial_k \partial^k\right] h^{TT}_{ij} = 0. \tag{18}$$

Its solutions are linear combinations of the plane wave solutions

$$h^{TT}_{k,ij}(\mathbf{x}) = \mathcal{A}_{ij}(\mathbf{k}) e^{i\mathbf{kx}}$$

where $\mathbf{k} = (|\vec{k}|, \vec{k}) =: (\omega/c, \vec{k})$ and the real part of the solution is taken at the end. We can reinsert the appropriate natural constants and it follows from eq. (18) that gravitational waves propagate at the speed of light

$$\left[-\frac{1}{c^2}\partial_0 \partial^0 + \partial_k \partial^k\right] h^{TT}_{ij} = 0. \tag{19}$$

From the third property in eq. (17) we get

$$0 = \partial^j h^{TT}_{k,ij} = \partial^j e_{ij}(\mathbf{k}) e^{ikx} = ik^j e_{ij}(\mathbf{k}) e^{ikx}$$
$$\Rightarrow 0 = e_{ij}(\mathbf{k}) k^j.$$

This means that the non-zero components of $h^{TT}_{k,ij}$ lie in the plane transverse to $\mathbf{k}$, so in the case of a wave propagating in x^3 direction-h^{TT}_{ij} in leading order has the form

$$h^{TT}_{ij} = \begin{pmatrix} h_+ & h_x & 0 \\ h_x & -h_+ & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ij} \cos\left[\omega(t - z/c)\right] \tag{20}$$

where h_+ and h_x represent the two remaining degrees of freedom after imposing that h^{TT}_{ij} be traceless and symmetric. For a general real-valued solution, we get the linear superposition of plain waves

$$h^{TT}_{ij}(\mathbf{x}) = \int \frac{d^3k}{(2\pi)^3} (\mathcal{A}_{ij}(\mathbf{k}) e^{i\mathbf{kx}} + \mathcal{A}^*_{ij}(\mathbf{k}) e^{-i\mathbf{kx}}), \tag{21}$$

where $\mathbf{k}$ is of the form $\mathbf{k} = (\omega/c, \vec{k})$ with $|\vec{k}| = \omega/c = 2\pi f/c$. In addition we denote the unit vector in the direction of $\vec{k}$ as $\vec{n} \coloneqq \vec{k}/|\vec{k}|$. We know that the dimension of the subspace of matrices fulfilling all of our gauge conditions is 2. Hence if we can find two linearly independent matrices for every wave vector $\mathbf{k}$ that both fulfill the gauge conditions we have found a complete basis for the solutions. For this we chose an orthonormal basis $(\vec{n}, \vec{u}, \vec{v})$ with respect to the standard scalar product where again $\vec{n} = \vec{k}/|\vec{k}|$ as above. Then we set

$$e^+(\vec{n}) \coloneqq \vec{u} \cdot \vec{u}^T - \vec{v} \cdot \vec{v}^T, \qquad e^\times(\vec{n}) \coloneqq \vec{u} \cdot \vec{v}^T + \vec{v} \cdot \vec{u}^T. \tag{22}$$

It is clear from the definition that both are traceless and transverse with respect to $\vec{n}$. Additionally the definition implies that they are normalized following

$$e^A_{ij}(\vec{n}) e^{A',ij}(\vec{n}) = 2\delta_{A,A'}. \tag{23}$$

It follows that we can write the polarization of every mode of h^{TT}_{ij} as a linear combination of $e^+(\vec{n})$ and $e^\times(\vec{n})$, where $\tilde{h}_A(f, \vec{n})$ denotes the occupation of the mode with wave vector $\mathbf{k}$

$$\mathcal{A}_{ij}(\mathbf{k}) = \mathcal{A}_{ij}(f, \vec{n}) = \sum_{A=+,\times} \frac{c^3}{f^2} \tilde{h}_A(f, \vec{n})\, e^A_{ij}(\vec{n}), \tag{24}$$

The choice of the factor $\frac{c^3}{f^2}$ is a convention. We can now insert the expression for the plane wave solution into the general solution eq. (21) which results in

$$h^{TT}_{ij}(\mathbf{x}) = \sum_{A=+,\times} \int_0^\infty df \int_{|\vec{n}|=1} \frac{d^2\vec{n}}{(2\pi)^3}\, e^A_{ij}(\vec{n}) \left(\tilde{h}_A(f, \vec{n})\, e^{-2\pi i f(t - \vec{n}\vec{x}/c)} + \tilde{h}_A(f, \vec{n})^*\, e^{2\pi i f(t - \vec{n}\vec{x}/c)} \right) \tag{25}$$

$$= \sum_{A=+,\times} \int_{-\infty}^\infty df \int_{|\vec{n}|=1} \frac{d^2\vec{n}}{(2\pi)^3}\, e^A_{ij}(\vec{n})\, \tilde{h}_A(f, \vec{n})\, e^{-2\pi i f(t - \vec{n}\vec{x}/c)} \tag{26}$$

where we have defined $\tilde{h}_A(-f, \vec{n}) \coloneqq \tilde{h}_A(f, \vec{n})^*$ for a more compact notation. This gives the form of a general solution for the linearized metric in the TT-coordinate frame in the absence of a source term.

4 Pulsar Timing Arrays

The introduction to Pulsars and PTAs is based on chapters 2 and 3 of [Tay21]. Since the first theoretical prediction of gravitational waves (GWs) from general relativity in the early 1900s, multiple experimental attempts have been made to detect them. Beginning with resonant-mass detectors designed by Joseph Weber in 1969/1970 [Web69], the experimental frameworks have advanced significantly since their early beginnings. Today, the most prominent instruments for the observation of GWs are large ground-based interferometers such as LIGO [LIG], Virgo [Vir] and KAGRA [KAG]. In the future, these efforts will be complemented by the first space-based interferometer LISA [LISA], scheduled to launch in 2035. In addition to interferometry, gravitational waves can be studied through Pulsar Timing Arrays (PTAs) and by analyzing their imprint on the cosmic microwave background (CMB). The main distinction between these methods is the frequency range of GWs to which they are sensitive. While terrestrial and space-born interferometers are sensitive to a frequency range of $10^{-4}\,\mathrm{Hz}$ to $10^{4}\,\mathrm{Hz}$, the imprint on the CMB is sensitive to ultra low frequency GWs at $10^{-18}\,\mathrm{Hz}$ to $10^{-16}\,\mathrm{Hz}$. In between those ranges lies the sensitivity band of PTAs at $10^{-9}\,\mathrm{Hz}$ to $10^{-6}\,\mathrm{Hz}$ (see [Mag18] Chapter 23). This work focuses on analysis methods related to PTA data which is why the concept behind PTAs will be introduced in the following section.

4.1 Pulsars

The central object of observation in a PTA is the name giving pulsar. Pulsars are a special class of neutron star, which themselves are remnants of massive stars that have undergone supernova explosions. During the supernova, the star sheds its outer layers, leaving behind a dense core that collapses in the process to a diameter of $\sim 10 - 15\,\mathrm{km}$. Since the neutron stars are in general significantly more compact than their progenitor star, they tend to have very short rotational periods and strong magnetic fields resulting in a rotational and a magnetic axis that are generally not aligned. This leads to a precession of the magnetic axis induced by the neutron stars' rotation. Due to their strong magnetic fields, neutron stars often emit a beam of magnetic dipole radiation along their magnetic axis. The rotation of the star causes the beam to sweep through space, leading to an effect similar to the rotating light cone of a lighthouse. The cone may, depending on its orientation and width, hit the earth once per rotation. Observed from earth, this then appears like a rapidly pulsating source of electromagnetic radiation, which is why the neutron star is called a pulsar in this case.

The rotation periods of pulsars and their derivatives span a wide range, from $10^{-3} - 10^{1}\,[\mathrm{s}]$, with corresponding derivatives of $10^{-22} - 10^{-9}\,[\mathrm{s/s}]$ (cf. fig. 1). To use the pulsars as a means for gravitational wave detection, one has to detect very small deviations in the arrival times of the received signals. In order to perform high-precision measurements over long periods, one requires pulsars with high long-timescale rotational stability. Such stability is commonly found in a class of pulsars called *millisecond pulsars*, named for their extremely short rotational periods in the millisecond range. In fig. 1, millisecond pulsars are colored in green, while slower-spinning commonly less stable pulsars are shown in orange.

Once a sufficiently stable pulsar has been chosen, the task to measure its change in frequency remains a demanding challenge. For an in-detail explanation of how to measure the times-of-arrival (TOAs) of the emitted pulses, see [BKK22] chapter 4. In the following section, we will derive the predicted deviation in the arrival time of a photon traveling from a pulsar to earth, induced by gravitational waves.

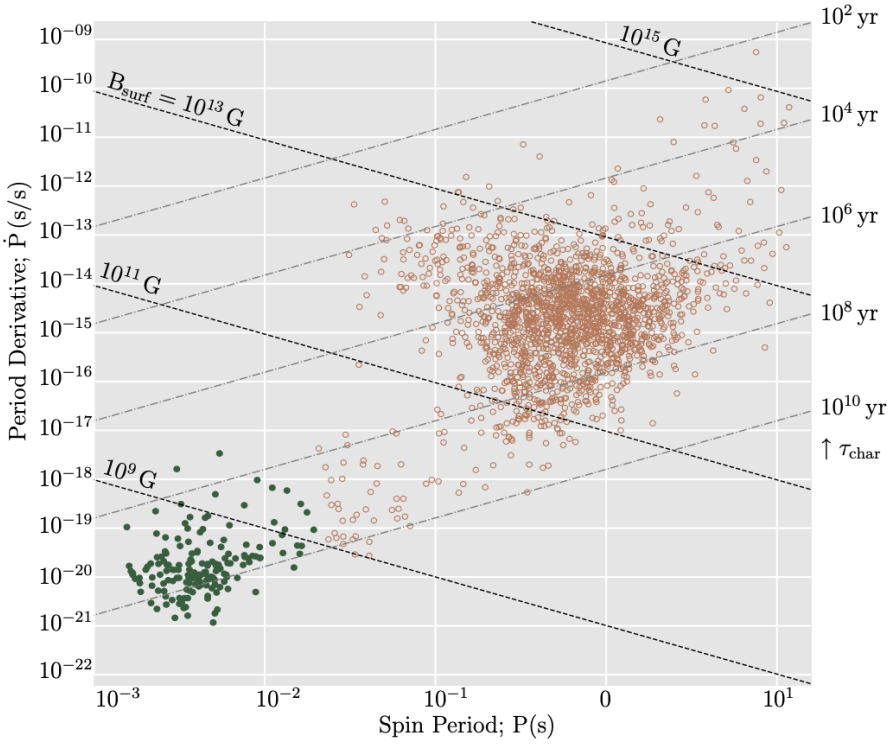


Figure 1: A diagram of the known pulsar population in terms of their rotational period P(s) and their derivatives $\dot{P}$(s). Millisecond pulsars are marked with filled green circles while canonical pulsars are marked with open circles. The dashed lines mark the estimated surface magnetic field strengths and the dot-dashed lines show the characteristic age of the pulsars. (Figure taken from [Tay21] based on the ATNF pulsar catalogue [Man+05]).

4.2 Timing Response to GWs

This section is based on chapter 23 of [Mag18]. To compute the effect of a gravitational wave on the arrival time of a photon traveling through spacetime, we use the residual reduced Poincaré symmetry to fix our coordinate system. Specifically, we choose a coordinate system in which the observer is at rest at $\vec{x}_{obs}(t) = 0$ and the emission event of the photon occurs at $\mathbf{x}_{em} = (t_{em}, d_a, 0, 0)$. We work in TT gauge (see section 3.2), which corresponds to describing the impact of the gravitational wave on a photon as it would be perceived by a freely falling observer. For most experimental setups this does not hold since they find themselves in an accelerated reference frame, for example due to earths gravitational field. To compare theory and experiment requires to transform both results into the same frame of reference (see [Mag07] chapter 1.3.3). This is done as part of the timing model which will be further discussed in section 4.4.

To calculate the timing effect of GWs on a photon, we work with the condition for a photon traveling with the speed of light, emitted by a pulsar P_a, where $\mathbf{c}(t) = (t, x(t), 0, 0)$ describes the trajectory of the photon parameterized with respect to coordinate time with initial conditions $\mathbf{c}(t_{obs}) = (t_{obs}, 0, 0, 0)$ and $\mathbf{c}(t_{em}) = (t_{em}, d_a, 0, 0)$. This leads to the following condition:

$$0 \stackrel{!}{=} -g_{\mu\nu}\frac{\partial c^\mu}{\partial t}\frac{\partial c^\nu}{\partial t} = \left(\frac{\partial c^0}{\partial t}\right)^2 - \left[1 + h^{TT}_{xx}\right]\left(\frac{\partial c^x}{\partial t}\right)^2 = 1 - \left[1 + h^{TT}_{xx}\right]\left(\frac{\partial x}{\partial t}\right)^2 \tag{27}$$

$$\Rightarrow \frac{\partial x}{\partial t} = -\frac{1}{\sqrt{1 + h^{TT}_{xx}}} = -\left[1 - \frac{1}{2}h^{TT}_{xx}\right] + \mathcal{O}(h^2). \tag{28}$$

Neglecting quadratic- and higher orders in h and integrating both sides we obtain

$$x(t_{obs}) - x(t_{em}) = -\int_{t_{em}}^{t_{obs}} \left[1 - \frac{1}{2}h_{xx}^{TT}\right] dt = t_{em} - t_{obs} + \frac{1}{2}\int_{t_{em}}^{t_{obs}} h_{xx}^{TT}\, dt \tag{29}$$

$$\Rightarrow d_a = t_{obs} - t_{em} - \frac{1}{2}\int_{t_{em}}^{t_{obs}} h_{xx}^{TT}\, dt. \tag{30}$$

In the integral bounds we can replace t_{obs} by its first-order approximation $t_{obs} = t_{em} + d_a$. The proof why this is justified is shown in appendix A.6. We also generalize the result to photons emitted by pulsars P_a at arbitrary positions $\vec{x}_a = d_a\vec{n_a}$ where $\vec{n}_a$ denotes the unitary vector pointing in direction of P_a. We therefore perform a spatial rotation of the coordinate frame in the $\vec{e}_x$-$\vec{n}_a$-plane, such that $\vec{e}_x \to \vec{n}_a$. The resulting photon trajectory in the new coordinate frame is given as

$$(t, t_{em} + d_a - t, 0, 0) = \mathbf{c}(t) \to \mathbf{c}'(t) = (t, (t_{em} + d_a - t)\vec{n}_a). \tag{31}$$

Since $h_{\mu\nu}^{TT}$ is a tensorial quantity, we know that the full contraction $h_{\mu\nu}^{TT}\frac{\partial c^\mu}{\partial t}\frac{\partial c^\nu}{\partial t}$ is invariant of coordinate choice, hence:

$$h_{xx}^{TT} = n_a^i n_a^j (h^{TT})'_{ij}. \tag{32}$$

This leads to

$$t_{obs} = t_{em} + d_a + \frac{n_a^i n_a^j}{2}\int_{t_{em}}^{t_{em}+d_a} (h^{TT})'_{ij}[t, (t_{em} + d_a - t)\vec{n}_a]\, dt. \tag{33}$$

The prime on $(h^{TT})'_{ij}$ will now be omitted but it should be clear that we are working in the rotated coordinate frame. Considering a second photon emitted at a later time $t'_{em} = t_{em} + T$ and replacing t_{em} with t'_{em}, the time t'_{obs} at which the second pulse will be observed is given as

$$t'_{obs} = t_{em} + T + d_a + \frac{n_a^i n_a^j}{2}\int_{t_{em}+T}^{t_{em}+T+d_a} h_{ij}^{TT}[t, (t_{em} + T + d_a - t)\vec{n}_a]\, dt \tag{34}$$

$$= t_{em} + T + d_a + \frac{n_a^i n_a^j}{2}\int_{t_{em}}^{t_{em}+d_a} h_{ij}^{TT}[t + T, (t_{em} + d_a - t)\vec{n}_a]\, dt, \tag{35}$$

where the second equality comes from substituting $t - T \to t$. Considering the difference in arrival time, by subtracting eq. (33) from eq. (35), we get

$$t'_{obs} - t_{obs} = T + \Delta T \tag{36}$$

with

$$\Delta T := \frac{n_a^i n_a^j}{2}\int_{t_{em}}^{t_{em}+d_a} h_{ij}^{TT}[t + T, \vec{x}(t)] - h_{ij}^{TT}[t, \vec{x}(t)]\, dt \tag{37}$$

and $\vec{x}(t) = (t_{em} + d_a - t)\vec{n}_a$. We can use the fundamental theorem of calculus to introduce a variable s and write this expression as

$$\Delta T = \frac{n_a^i n_a^j}{2} \int_{t_{em}}^{t_{em}+d_a} \int_0^T \frac{\partial}{\partial s} h_{ij}^{TT}[t+s, \vec{x}(t)]\, ds\, dt. \tag{38}$$

Here the partial derivative with respect to s can be replaced by a derivative with respect to t keeping the spatial argument of h_{ij}^{TT} fixed

$$\Delta T = \frac{n_a^i n_a^j}{2} \int_{t_{em}}^{t_{em}+d_a} \int_0^T \frac{\partial}{\partial t} h_{ij}^{TT}[t+s, \vec{x}]\Big|_{\vec{x}=\vec{x}(t)}\, ds\, dt \tag{39}$$

$$= \frac{n_a^i n_a^j}{2} \int_0^T \int_{t_{em}}^{t_{em}+d_a} \frac{\partial}{\partial t} h_{ij}^{TT}[t+s, \vec{x}]\Big|_{\vec{x}=\vec{x}(t)}\, dt\, ds. \tag{40}$$

By substituting $t' = t + s$ and thus replacing the integration over t one gets

$$\Delta T = \frac{n_a^i n_a^j}{2} \int_0^T \int_{t_{em}+s}^{t_{em}+d_a+s} \frac{\partial}{\partial t'} h_{ij}^{TT}[t', \vec{x}]\Big|_{\vec{x}=\vec{x}(t'-s)}\, dt'\, ds. \tag{41}$$

As seen before in eq. (26), the gravitational waves in TT-gauge are linear combinations of plane waves with frequency ω_{gw} propagating in direction $\vec{n}$

$$h_{k,ij}^{TT}(\mathbf{x}) = e_{ij}(\vec{n}) \cos[\omega_{\mathrm{gw}}(t - \vec{n}\vec{x})], \tag{42}$$

where the Period of the waves is given in eq. (42) as $T_{\mathrm{gw}} = 2\pi/\omega_{\mathrm{gw}}$. Inserting the expression for h_{ij}^{TT} into eq. (41) and performing derivative and integration leads to

$$\Delta T = \frac{n_a^i n_a^j e_{ij}(\vec{n})}{2(1+\vec{n}\vec{n}_a)} \int_0^T \{\cos[\omega_{\mathrm{gw}}(t_{em}+d_a+s)] - \cos[\omega_{\mathrm{gw}}(t_{em}+s-\vec{n}\vec{n}_a d_a)]\}\, ds \tag{43}$$

$$= \frac{n_a^i n_a^j}{2(1+\vec{n}\vec{n}_a)} \int_0^T \left\{h_{k,ij}^{TT}[t_{em}+d_a+s, \vec{0}] - h_{k,ij}^{TT}[t_{em}+s, d_a\vec{n}_a]\right\} ds. \tag{44}$$

The analogous calculation can be performed for h_{ij}^{TT} as a superposition of plane waves. Since the delay ΔT is linear in the gravitational wave modes, an analogous result to the single-mode case is obtained by exchanging $h_{k,ij}^{TT} \to h_{ij}^{TT}$. In the following we will therefore work with h_{ij}^{TT} as a general solution for the metric perturbation.

Choosing the time coordinate such that $t_{em} = -d_a$, the result is

$$\Delta T = \frac{n_a^i n_a^j}{2(1+\vec{n}\vec{n}_a)} \int_0^T \left\{h_{ij}^{TT}[s, \vec{0}] - h_{ij}^{TT}[s-d_a, d_a\vec{n}_a]\right\} ds. \tag{45}$$

So far, we have computed the resulting delay observed between the arrival times of two pulses emitted by the pulsar at times $t_{em} = -d_a$ and $t'_{em} = T - d_a$, that is, the delay as a function of the emission times. We now want to calculate the delay depending on the observed arrival times. We assume that the first pulse is measured at $t_{obs} = 0$. If we observe a second pulse at $t'_{obs} = t$, using eq. (36), the cumulated delay at time t will be given as a function of the corresponding emission times as

$$\Delta T(t'_{em} - t_{em}) = \Delta T(t - \Delta T(T)) = \Delta T(t) - \left(\frac{\partial}{\partial t}\Delta T\right)\Delta T + \mathcal{O}(\Delta T^2). \tag{46}$$

It follows from eq. (44) that $\mathcal{O}(\Delta T) = \mathcal{O}(h_{ij}^{TT})$. Since both h_{ij}^{TT} and $\frac{\partial}{\partial t} h_{ij}^{TT}$ are demanded to be small, we can neglect all orders but the zeroth one. This allows us to define the timing residual of the a-th pulsar at observation time t as

$$R(t) \coloneqq \frac{n_a^i n_a^j}{2(1+\vec{n}\vec{n}_a)} \int_0^t \left\{ h_{ij}^{TT}[t', \vec{0}] - h_{ij}^{TT}[t' - d_a, d_a \vec{n}_a] \right\} dt' =: \int_0^t z(t')\, dt', \tag{47}$$

and we call z the instantaneous shift. The timing residual R is the key quantity measured in PTA observations.

4.3 Stochastic GW Backgrounds

This section is based on chapter 7.8.1 of [Mag07] and chapter 23.3 of [Mag18]. When studying PTAs, many of the possible GW-sources of interest produce a stochastic gravitational wave background (SGWB), that is a superposition of waves of all frequencies arriving from all directions. In this case the Fourier coefficients $\tilde{h}_A(f, \vec{n})$ of the GW modes become randomly distributed quantities and the resulting metric perturbation $h_{ij}^{TT}(t, \vec{x})$ has to be treated as a random process. As shown in the previous chapter, the timing residuals depend only on the metric perturbation at the positions of the pulsar and the observer. We are therefore only interested in fixed spatial locations and will omit the explicit dependence of h_{ij}^{TT} on $\vec{x}$ in the following.

We model the SGWB as a random process described in Fourier space and make the assumptions that it is stationary, Gaussian, isotropic and unpolarized. One should note that when working with complex solutions for the metric perturbation the Fourier-coefficients are themselves complex quantities. For a more in detail discussion of complex random variables and why these assumptions uniquely characterize the process, see appendix A.4. From here, the superscript TT will be omitted. The properties demanded for the random process and their consequences will be discussed in the following.

- The assumption that the distribution is stationary implies that the correlation function $\langle h_{ij}(t), h_{ij}(t') \rangle$ only depends on $|t - t'|$ and not the absolute values of t and t'. It follows that $\langle \tilde{h}_A^*(f), \tilde{h}_{A'}(f') \rangle \sim \delta(f - f')$ (see appendix A.4). This assumption is justified for a background of cosmological origin, that evolves on the timescales of the age of the universe. In contrast, the observation time of any possible experiment will be of the order of years, making the evolution of the background during the observational time span negligible. Additionally this also implies $\langle h_{ij}(t) \rangle \equiv \text{const}$, which can be absorbed in the cosmological constant. We can therefore set $\langle h_{ij}(t) \rangle = 0$.
- The assumption that the process is Gaussian implies that, for any finite set of discrete times $(t_1, \ldots, t_N)$, the resulting random vector $(h_{ij}(t_1), \ldots, h_{ij}(t_N))$ is distributed following a multivariate Gaussian distribution (see appendix A.2). Given the assumption of zero mean, this distribution is fully described by its covariance matrix. It follows that for any set of frequencies $(f_1, \ldots, f_M)$ the resulting random vector of Fourier modes $(\tilde{h}_{A_1}(f_1), \ldots, \tilde{h}_{A_M}(f_M)$ is also distributed following a multivariate Gaussian distribution. This assumption is justified by the central limit theorem which states that the sum of a large number of random variables is distributed following a Gaussian distribution in the limit of an infinite sum,W independent of the distributions of the individual random variables. Since an SGWB from cosmological origins is assumed to arise from a vast number of individual sources, using the central limit theorem is justified in this instance.
- The assumption that the random process is isotropic implies that the distribution of $\tilde{h}_A(f, \vec{n})$ is independent of $\vec{n}$, which implies $\langle \tilde{h}_A^*(f, \vec{n}), \tilde{h}_{A'}(f, \vec{n}') \rangle \sim \delta^2(\vec{n}, \vec{n}') = \delta(\phi -$

$\phi')\,\delta(\cos\theta - \cos\theta')$, where (ϕ, θ) and (ϕ', θ') denote the polar angles of $\vec{n}$ and $\vec{n}'$ respectively. This is supported by the observed isotropy of the CMB, indicating that the early universe was highly isotropic. It is therefore reasonable to assume that an SGWB would be isotropic at first order as well. Nevertheless, it will be potentially interesting to study anisotropies of the SGWB in the future.

- The assumption that the random process is unpolarized implies that distribution of $\tilde{h}_A(f, \vec{n})$ is independent of A which implies that the two polarizations are statistically independent $\langle \tilde{h}^*_A(f, \vec{n}), \tilde{h}_{A'}(f, \vec{n}')\rangle \sim \delta_{A,A'}$. This assumption is justified since none of the sources usually considered for an SGWB has a mechanism to produce a preferred polarization. It is therefore unexpected to find correlations between different polarizations of the gravitational waves.

It follows from these assumption that the correlations between the Fourier modes of h_{ij} are fully described by their spectral distribution (see appendix A.4). Consequently, the spectral density of the stochastic background $S_h(f)$ is defined which allows to give an expressing for the correlation of the Fourier modes as

$$\langle \tilde{h}^*_A(f, \vec{n}), \tilde{h}_{A'}(f', \vec{n'})\rangle = \delta(f - f')\,\frac{\delta^2(\vec{n} - \vec{n}')}{4\pi}\,\delta_{A,A'}\,\frac{1}{2}S_h(f). \tag{48}$$

The factors $\frac{1}{4\pi}$ and $\frac{1}{2}$ are conventional normalization constants.

The question then becomes: What effect does a stochastic gravitational wave background, as defined above, have on pulsar times of arrival? To answer this, we again consider the response of a photon to an SGWB. By inserting h_{ij} from eq. (26) into the instantaneous shift z in eq. (47), we obtain

$$z_a(t) = \sum_{A=+,\times} \int_{-\infty}^{\infty} df \int_{|\vec{n}|=1} d\vec{n}\,\tilde{h}_A(f, \vec{n})\,F^A_a(\vec{n})\,e^{-2\pi i f t}\left[1 - e^{2\pi i f d_a(1+\vec{n}\vec{n_a})}\right], \tag{49}$$

where we use

$$F^A_a(\vec{n}) \coloneqq \frac{n^i_a n^j_a e_{ij}(\vec{n})}{2(1 + \vec{n}\vec{n}_a)}. \tag{50}$$

Considering the case of two simultaneously observed pulsars P_a and P_b, we consider the correlations between the two pulsars signals $\langle z_a(t), z_b(t)\rangle$ at the same observation time, where we average over the random variable $\tilde{h}_A(f, \vec{n})$. Using the correlation in Fourier space given in eq. (48) one obtains

$$\langle z_a(t), z_b(t)\rangle = \frac{1}{2}\int_{-\infty}^{\infty} df\,S_h(f) \int_{|\vec{n}|=1} \frac{d^2\vec{n}}{4\pi}\mathcal{K}_{ab}(f, \vec{n}) \sum_{A=+,\times} F^A_a(\vec{n})\,F^A_b(\vec{n}), \tag{51}$$

where

$$\mathcal{K}_{ab}(f, \vec{n}) \coloneqq \left[1 - e^{-2\pi i f d_a(1+\vec{n}\vec{n}_a)}\right]\left[1 - e^{2\pi i f d_b(1+\vec{n}\vec{n}_b)}\right]. \tag{52}$$

Here we have used the fact that z_b is a real quantity hence $z_b = z^*_b$. In the following we will make the approximation $\mathcal{K}_{ab} \to 1$ for $a \neq b$. This approximation is justified by the following

consideration: In practice the pulsars find themselves at distances of the order of kpc and the PTA frequency range lies at frequencies of order 10^{-9} Hz to 10^{-6} Hz. The resulting products in the exponent $f\, d_a$ and $f\, d_b$ therefore lead to a rapidly oscillating complex exponential function which only contributes negligibly to the integral. The only case in which this approximation does not hold is the case of $\vec{n} = -\vec{n_{a,b}}$. The resulting error is comparatively small though. For a more detailed discussion of the errors see chapter 23.3 of [Mag18].

Using this approximation we obtain

$$\langle z_a(t), z_b(t) \rangle = \frac{1}{2} \int_{-\infty}^{\infty} df\, S_h(f) \int_{|\vec{n}|=1} \frac{d^2\vec{n}}{4\pi} \sum_{A=+,\times} F_a^A(\vec{n})\, F_b^A(\vec{n}). \tag{53}$$

In order to evaluate this integral, an explicit computation of the polarization tensors from eq. (22),

$$e^+(\vec{n}) \coloneqq \vec{u} \cdot \vec{u}^T - \vec{v} \cdot \vec{v}^T, \qquad e^\times(\vec{n}) \coloneqq \vec{u} \cdot \vec{v}^T + \vec{v} \cdot \vec{u}^T, \tag{54}$$

is necessary. This necessitates an explicit choice of basis vectors $\vec{u}$ and $\vec{v}$. We can express $\vec{n}$ using polar and azimuthal angle (ϕ, θ) as

$$\vec{n} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \tag{55}$$

A convenient choice is

$$\vec{u} \coloneqq (\sin\phi, -\cos\phi, 0), \quad \vec{v} \coloneqq (\cos\theta\cos\phi, \cos\theta\sin\phi, -\sin\theta). \tag{56}$$

The resulting integral, written as a function of the angular separation of the pulsars P_a and P_b: $\theta_{ab} = \cos^{-1}(\vec{n}_a \cdot \vec{n}_b)$, can then be solved

$$C(\theta_{ab}) \coloneqq \int_{|\vec{n}|=1} \frac{d^2\vec{n}}{4\pi} \sum_{A=+,\times} F_a^A(\vec{n})\, F_b^A(\vec{n}) \tag{57}$$

$$= x_{ab} \log x_{ab} - \frac{1}{6} x_{ab} + \frac{1}{3} \quad \text{with} \tag{58}$$

$$x_{ab} \coloneqq \frac{1}{2}(1 - \cos\theta_{ab}). \tag{59}$$

For a more detailed derivation see appendix C of [Anh+09]. In the case of two identical pulsars $a = b$ one obtains

$$\mathcal{K}_{ab}(f, \vec{n}) = 2 + \text{fast-oscillating terms} \tag{60}$$

where we can neglect the oscillating terms since the large value of d_a leads to rapid oscillations in f which means that it can be neglected as a contribution to $\langle z_a(t), z_b(t) \rangle$ as shown in [Cor+25]. This results in an additional factor of two in the case of two identical pulsars. The full solution is then given as

$$C(\theta_{ab}) = (1 + \delta_{ab}) \left[x_{ab} \log x_{ab} - \frac{1}{6} x_{ab} + \frac{1}{3} \right]. \tag{61}$$

As a result we obtain for the correlation of the instantaneous shifts

$$\langle z_a(t), z_b(t)\rangle = C(\theta_{ab}) \int_0^\infty df\, S_h(f). \tag{62}$$

Using this result, we can further compute the correlator of the timing residuals. Inserting eq. (49) into eq. (47) and solving the time integral one obtains

$$R_a(t) = \sum_{A=+,\times} \int_{-\infty}^{\infty} df \int_{|\vec{n}|=1} d\vec{n}\, \tilde{h}_A(f,\vec{n})\, F_a^A(\vec{n}) \left[1 - e^{2\pi i f d_a (1+\vec{n}\cdot\vec{n_a})}\right] \frac{\left[e^{-2\pi i f t} - 1\right]}{-2\pi i f} \tag{63}$$

$$= \sum_{A=+,\times} \int_{-\infty}^{\infty} df \int_{|\vec{n}|=1} d\vec{n}\, \frac{i\tilde{h}_A(f,\vec{n})}{2\pi f}\, F_a^A(\vec{n})\, e^{-2\pi i f t} \left[1 - e^{2\pi i f d_a (1+\vec{n}\cdot\vec{n_a})}\right] + c(h). \tag{64}$$

We observe that the timing residuals can be split into a time dependent part and a part constant in time, defined as $c(h)$. In practice, when trying to measure the timing residuals, it is not possible to determine the value of $c(h)$, as it is degenerate with the constant offset in the timing model used to derive the timing residuals from the observed pulse arrival times (see [Tay14], Chapter 4.3). We can therefore neglect $c(h)$ in the following. By complex conjugating the real quantity $R_a(t)$, the equal-time correlator $r_{ab}(t) \coloneqq \langle R_a(t), R_b(t)\rangle$ is computed analogously to case of $z_a(t)$

$$r_{ab}(t) = C(\theta_{ab}) \int_0^\infty df\, \frac{S_h(f)}{(2\pi f)^2}. \tag{65}$$

Following convention, we rescale $C(\theta_{ab})$ to $\zeta(\theta_{ab}) \coloneqq \frac{3}{2} C(\theta_{ab})$, where $\zeta(\theta_{ab})$ is called the *overlap reduction function* (ORF)

$$\zeta(\theta_{ab}) = (1+\delta_{ab}) \left[\frac{3}{2} x_{ab} \log x_{ab} - \frac{1}{4} x_{ab} + \frac{1}{2}\right]. \tag{66}$$

It should be noted that in the case of two identical pulsars $a = b$ it follows $\theta_{ab} = 0$, hence $x_{ab} = 0$ and in the limit

$$\lim_{x_{ab}\to 0} \zeta(x_{ab}) = 1. \tag{67}$$

Using the overlap reduction function, we obtain for the correlation of the timing residuals

$$r_{ab}(t) = \zeta(\theta_{ab}) \int_0^\infty df\, \frac{2}{3} \frac{S_h(f)}{(2\pi f)^2} =: \zeta(\theta_{ab}) \int_0^\infty df\, P_g(f). \tag{68}$$

We have thus found an expression for the same time correlation of the timing residuals for two pulsars P_a and P_b, which is fully determined by the overlap reduction function of their angular separation and the *residual power spectral density* $P_g(f)$. This allows to give an expression for the correlations between the Fourier modes of the timing residuals

$$\langle \hat{R}_a(f), \hat{R}_b(f')\rangle = \zeta(\theta_{ab})\, \delta(f - f')\, P_g(f) = \zeta(\theta_{ab})\, \delta(f - f')\, \frac{S_h(f)}{6\pi^2 f^2}. \tag{69}$$

In the gravitational wave literature, there are several interdependent quantities used to describe an SGWB (see [Aga+25]). $S_h(f)$ is also called the *strain power spectrum*. One also commonly uses the GW *strain amplitude*

$$h_c(f) \coloneqq \sqrt{2fS_h(f)} \tag{70}$$

which is the relevant quantity for experimental detection (see [Mag07] chapter 7). The GW *energy density spectrum*

$$\Omega_{GW}(f) = \frac{4\pi^2}{3H_0^2} f^3 S_h(f) \tag{71}$$

is given in units of the critical energy density ρ_c of the present universe and describes the energy density ρ_{GW} of the SGWB via

$$\frac{\rho_{GW}}{\rho_c} = \int_0^\infty d(\log f)\, \Omega_{GW}(f) \tag{72}$$

(see [Mag07] chapter 7).
To additionally include the inter pulsar spatial correlations into the spectrum, one also uses the *residual cross-power spectrum*

$$S_{ab}^{GW}(f) = \zeta(\theta_{ab})\, P_g(f). \tag{73}$$

4.4 Timing Model

This section is based on [HL12]. In pulsar timing experiments, the directly measurable observables are the times-of-arrival (TOAs) of the electromagnetic pulses emitted by the pulsars. For a mathematical description of the TOAs, must consider multiple contributions that contribute to them. There are deterministic contributions, such as those linked to the coordinate system in which the theoretical calculations were performed. In TT-gauge we use the reference frame of a freely falling observer which corresponds to the reference frame of the solar system barycenter (SSB). The detector (here, a system of radio telescopes) used to measure the TOAs, however, resides in a frame that is relatively accelerated with respect to the SSB (see chapter 1.3.3 of [Mag07]). This makes it necessary to perform corrections, grouped under the name Einstein-delay. Furthermore, there are corrections necessary due to intrinsic properties of the pulsars themselves, such as their rotational period or the gradual change in their rotational velocity over time. Additionally, stochastic contributions from various sources besides the SGWB have to be taken into account. These include white noise e.g. from the used radio telescope itself or from the accuracy of the clock used to time the pulses. There may also be red noise from pulsar dependent sources such as pulse phase jitter, inhomogeneities in the interstellar medium or pulsar intrinsic red noise. For comprehensive account of all necessary corrections, see [HEM06].

The term *timing model* is used somewhat ambiguously in the literature. Depending on the context it is sometimes used to describe the full TOAs, taking into account both deterministic and stochastic contributions. In other cases the timing model only describes the deterministic contributions to the TOAs. In the following we use "timing model" to refer to the deterministic part and use the term *noise model* for the stochastic contributions.

In practical data analysis, the time is discretized in k points due to the finite amount of observations. The continuous observation time thus becomes $t_{\text{obs}} \to \mathbf{t}_{\text{obs}} = (t_1, \cdots, t_k)$ and the

timing residuals become $R(t) \to \delta\mathbf{t} = (R(t_1), \ldots, R(t_k))$.

Mathematically, the timing model describes the deterministic contribution to the TOAs $\mathbf{t}_{\text{det}}$ as a function of the observed TOAs $\mathbf{t}_{\text{obs}}$, parametrized over the m parameters of the timing model $\boldsymbol{\xi}$. Taking into account the additional stochastic contribution to the TOAS $\mathbf{n}$, the full model becomes

$$\mathbf{t}_{\text{obs}} = \mathbf{t}_{\text{det}}(\boldsymbol{\xi}) + \mathbf{n}. \tag{74}$$

While the timing models used are not unique, they usually include a constant term

$$\mathbf{t}_{\text{det}}(\boldsymbol{\xi}) = \boldsymbol{\xi}_1 + \mathbf{t}'_{\text{det}}(\boldsymbol{\xi}_{2,..m}). \tag{75}$$

This constant contribution prohibits any sensitivity of PTA experiments to a constant offset to the timing residuals induced by an SGWB, as previously discussed in eq. (64).

Even after accounting for all known deterministic corrections, there remains a residual degree of uncertainty, stemming from the fact that the pulsars are located at distances of kilo parsecs. Our knowledge of their behavior is thus incomplete. Therefore, in order to accurately extract the timing residuals from the measurable TOAs, we have to account for further uncertainties in the parameters of the timing model. Using the best estimates for the timing model parameters $\boldsymbol{\xi}_{\text{est}}$, yields the pre-fit timing residuals $\delta\mathbf{t}_{pre}$

$$\delta\mathbf{t}_{pre} \coloneqq \mathbf{t}_{\text{obs}} - \mathbf{t}_{\text{det}}(\boldsymbol{\xi}_{\text{est}}) = \mathbf{t}_{\text{det}}(\boldsymbol{\xi}_{\text{true}}) + \mathbf{n} - \mathbf{t}_{\text{det}}(\boldsymbol{\xi}_{\text{est}}). \tag{76}$$

We assume that the estimates for the parameters are accurate up to a small offset,

$$\boldsymbol{\xi}_{\text{true}} = \boldsymbol{\xi}_{\text{est}} + \delta\boldsymbol{\xi}, \tag{77}$$

with $|\delta\boldsymbol{\xi}| \ll |\delta\mathbf{t}_{pre}|$. This allows the linearization of the timing model

$$\delta\mathbf{t}_{pre} = \frac{\partial \mathbf{t}_{det}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}_{\text{est}})\delta\boldsymbol{\xi} + \mathbf{n} + \mathcal{O}(\delta\boldsymbol{\xi}^2) =: \mathbf{M}\delta\boldsymbol{\xi} + \mathbf{n} + \mathcal{O}(\delta\boldsymbol{\xi}^2). \tag{78}$$

The noise time series produced by the stochastic contributions $\mathbf{n}$ is assumed to be Gaussian with $\langle\mathbf{n}\rangle = \mathbf{0}$ and $\langle n_i n_j\rangle = \Sigma_{ij}$, with $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{white}} + \boldsymbol{\Sigma}_{\text{red}} + \boldsymbol{\Sigma}_{\text{GW}}$ being the invertible covariance matrix of the noise, depending on the noise model. We assume red and white noise contributions to $\mathbf{n}$ and a GW-induced noise component given by the SGWB-model dependent covariance matrix $\boldsymbol{\Sigma}_{\text{GW}}$ (cf. appendix A.2). The full covariance matrix $\boldsymbol{\Sigma}$ is dependent on the noise model parameters $\boldsymbol{\theta}$, grouping the parameters from all noise contributions. Dropping higher orders in $\delta\boldsymbol{\xi}$ we can insert the expression for $\mathbf{n}$ we obtain from eq. (78) into the probability distribution for the noise

$$p(\delta\mathbf{t}_{pre}|\delta\boldsymbol{\xi}, \boldsymbol{\theta}) = \frac{1}{(2\pi)^{k/2}\det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\delta\mathbf{t}_{pre} - \mathbf{M}\delta\boldsymbol{\xi})^T\boldsymbol{\Sigma}^{-1}(\delta\mathbf{t}_{pre} - \mathbf{M}\delta\boldsymbol{\xi})\right).$$

This gives the conditional probability distribution for the pre-fit timing residuals conditioned upon the timing model parameters $\delta\boldsymbol{\xi}$ and the SGWB parameters $\boldsymbol{\theta}$, in other words the likelihood (see eq. (2)) for $\delta\mathbf{t}_{pre}$. Given a prior probability distribution $p_0(\delta\boldsymbol{\xi}, \boldsymbol{\theta})$ for $\delta\boldsymbol{\xi}$ and $\boldsymbol{\theta}$, we can compute the posterior distribution (see eq. (4))

$$p(\delta\boldsymbol{\xi}, \boldsymbol{\theta}|\delta\mathbf{t}_{pre}) = \frac{p_0(\delta\boldsymbol{\xi}, \boldsymbol{\theta})}{p(\delta\mathbf{t}_{pre})\,(2\pi)^{k/2}\,\det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\delta\mathbf{t}_{pre} - \mathbf{M}\delta\boldsymbol{\xi})^T\boldsymbol{\Sigma}^{-1}(\delta\mathbf{t}_{pre} - \mathbf{M}\delta\boldsymbol{\xi})\right). \quad (79)$$

One should note that even though the evidence is dependent on the functional form and prior of the model it no longer depends on the values of the parameters $\delta\boldsymbol{\xi}$ and $\boldsymbol{\theta}$ and can therefore be treated as a constant. We are interested in the noise model parameters $\boldsymbol{\theta}$ and thus need to marginalize over the timing model parameters $\delta\boldsymbol{\xi}$. We make the assumption of an uncorrelated prior that is flat in $\delta\boldsymbol{\xi}$

$$p_0(\delta\boldsymbol{\xi}, \boldsymbol{\theta}) = p_0(\delta\boldsymbol{\xi})\, p_0(\boldsymbol{\theta}) = \kappa\, p_0(\boldsymbol{\theta}), \quad (80)$$

with κ being the normalization constant of the flat prior. In this case the marginalized posterior distribution is given as

$$p(\boldsymbol{\theta}|\delta\mathbf{t}_{pre}) = \frac{\kappa\, p_0(\boldsymbol{\theta})}{p(\delta\mathbf{t}_{pre})\,\left[(2\pi)^{k-m}\,\det(\boldsymbol{\Sigma}(\boldsymbol{\theta}))\,\det\left(\mathbf{M}^T\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{M}\right)\right]^{1/2}} \exp\left(-\frac{1}{2}\delta\mathbf{t}_{pre}^T\,\mathbf{C}(\boldsymbol{\theta})\,\delta\mathbf{t}_{pre}\right), \quad (81)$$

where $\mathbf{C}(\boldsymbol{\theta})$ is given as

$$\mathbf{C}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} - \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{M}(\mathbf{M}^T\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{M})^{-1}\mathbf{M}^T\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} =: \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\mathbb{1} - \mathbf{B}) =: \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{O}. \quad (82)$$

The exact derivation can be found in appendix A of [Haa+09]. In numerical analyses one is often interested in the likelihood, which can also be marginalized over $\delta\boldsymbol{\xi}$ and analogously results in

$$p(\delta\mathbf{t}_{pre}|\boldsymbol{\theta}) = \frac{1}{\left[(2\pi)^{k-m}\,\det(\boldsymbol{\Sigma}(\boldsymbol{\theta}))\,\det\left(\mathbf{M}^T\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{M}\right)\right]^{1/2}} \exp\left(-\frac{1}{2}\delta\mathbf{t}_{pre}^T\,\mathbf{C}(\boldsymbol{\theta})\,\delta\mathbf{t}_{pre}\right). \quad (83)$$

To calculate this function is relatively costly since it involves multiple computations of the determinant and the inverse of large matrices. Herein lies the main computational bottleneck of conventional PTA analyses.

It is possible to find an equivalent expression for likelihood by performing a singular value decomposition of the $(k \times m)$ matrix $\mathbf{M}$

$$\mathbf{M} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}, \quad (84)$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal $(k \times k)$ and $(m \times m)$ matrices and D is a diagonal $(k \times m)$ matrix, meaning $\mathbf{D}$ is of the form

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}' \\ \mathbf{0} \end{pmatrix}, \quad (85)$$

where $\mathbf{D}'$ is a $(m \times m)$ diagonal matrix. The columns of $\mathbf{U}$ give a basis of $\mathbb{R}^k$ where the first m columns are an orthogonal basis of the image space of $\mathbf{M}$ and and the remaining $k-m$ columns give a basis of the complement of the image. We can therefore define the matrices $\mathbf{F}$ and $\mathbf{G}$ as

$$\mathbf{U} =: \begin{pmatrix} \mathbf{F} & \mathbf{G} \end{pmatrix}. \tag{86}$$

Here, $\mathbf{G}$ is a projection matrix that allows to project a given vector $\mathbf{t}$ onto its part lying orthogonal to the image of $\mathbf{M}$. One can think of $\mathbf{G}$ as a function that removes all contributions of the timing model from a time series which is illustrated by the fact that $\mathbf{G}^T \mathbf{M} = \mathbf{0}$ and therefore

$$\mathbf{G}^T \delta\mathbf{t}_{pre} = \mathbf{G}^T \mathbf{n} + \mathbf{G}^T \mathbf{M} \delta\boldsymbol{\xi} = \mathbf{G}^T \mathbf{n}. \tag{87}$$

We see that the probability distribution of the projection $\mathbf{G}^T \delta\mathbf{t}_{pre}$ is fully characterized by the distribution of the stochastic contribution to the TOAs $\mathbf{n}$ and in particular independent of the timing model parameters $\delta\boldsymbol{\xi}$.

Having defined $\mathbf{G}$ it was shown in [HL12] that an equivalent expression for the marginalized likelihood is given as

$$p(\delta\mathbf{t}_{pre}|\boldsymbol{\theta}) = \frac{1}{[(2\pi)^{k-m} \det(\mathbf{G}^T \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G})]^{1/2}} \exp\left(-\frac{1}{2}\delta\mathbf{t}_{pre}^T \mathbf{G} \left(\mathbf{G}^T \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G}\right)^{-1} \mathbf{G}^T \delta\mathbf{t}_{pre}\right). \tag{88}$$

Therefore, by defining

$$\delta\mathbf{t}_G \coloneqq \mathbf{G}^T \delta\mathbf{t}_{pre}, \tag{89}$$

we can give an equivalent expression of the likelihood for the pre-fit timing residuals $\delta\mathbf{t}_{pre}$ using the $(k-m)$ dimensional random process $\delta\mathbf{t}_G$ with covariance matrix $\mathbf{G}^T \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G}$. The computation of $\mathbf{G}$ depends only on the functional form of the timing model at $\boldsymbol{\xi}_{\text{est}}$ and thus only has to be computed once. This is why we will work with the projected timing residuals and their likelihood in the following, given as

$$p(\delta\mathbf{t}_G|\boldsymbol{\theta}) = \frac{1}{[(2\pi)^{k-m} \det(\mathbf{G}^T \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G})]^{1/2}} \exp\left(-\frac{1}{2}\delta\mathbf{t}_G^T \left(\mathbf{G}^T \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G}\right)^{-1} \delta\mathbf{t}_G\right). \tag{90}$$

5 Machine Learning and Statistics

In recent years, the application of machine learning techniques in the field of physics has seen great success, culminating the the award of a Nobel Prize to John J. Hopfield and Geoffreq Hinton in 2024 [Nobel24]. In this work we use generative neural networks to perform parameter inference on data that is modeled after Pulsar Timing Array data. In the following, the general concept behind deep learning algorithms and how they can be used to solve statistical problems will be discussed.

5.1 Deep Learning

The section on the fundamental concepts behind deep learning is based on chapters 1 and 5 of [Acq23]. The core idea in deep learning is to try to approximate a functional relationship $\mathbf{y} = F(\mathbf{x})$, using a conceptually rather simple, but numerically flexible mathematical model $F_\lambda(\mathbf{x})$. In contrast to conventional algorithms, the relationship is not initially fixed but is instead inferred from data. Advances in computational power of modern computers and the possibility to gather large sets of data, allow to generalize the concept of *parameter fitting* to increasingly complex problems. Deep learning problems can be broadly divided into the categories of supervised- and unsupervised learning, although this categorization is neither definitive nor exhaustive. In a supervised learning scenario, the given data D consists of pairs $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ that reflect the functional form of interest

$$F(\mathbf{x}_i) = \mathbf{y}_i. \tag{91}$$

In an unsupervised learning context, the given data does not reflect the functional form, which is why the model is expected to learn a general rule of its own. A given example is, learning the general rule behind a probability distribution from a sampling of that distribution. Both approaches share many basic concepts which will be explained in the following.

Irrespective of whether in supervised- or unsupervised learning, one of the most common kind of deep learning models are neural networks. The basic structure of a neural network consists of multiple layers, each of which having the functional form

$$F(\mathbf{x}|\boldsymbol{\lambda}) = \Phi(\mathbf{W}\mathbf{x} + \mathbf{b}). \tag{92}$$

Here, $\mathbf{W}$ is a $(m \times n)$ matrix, $\mathbf{b}$ is an m dimensional vector and Φ is a, usually nonlinear, function. The components of $\mathbf{W}$ and $\mathbf{b}$ are the free trainable parameters of the model and they will be jointly denote as $\boldsymbol{\lambda}$. Φ can have different functional forms. Popular choices are the rectified linear unit function (ReLu), leaky ReLu function, Sigmoid function and Softplus function

$$\text{ReLU}(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{93}$$

$$\text{LeakyReLu(x)} = \begin{cases} \alpha x & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}, \qquad \text{commonly } \alpha = 10^{-2} \tag{94}$$

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{95}$$

$$\text{Softplus}(x) = \log(1 + e^x). \tag{96}$$

Used within a neural network, they are applied component-wise

$$\Phi(x_1, \dots x_n) \coloneqq (\Phi(x_1), \dots, \Phi(x_n)). \tag{97}$$

Additionally, there are nonlinear functions, directly operating on vectors such as the softmax function

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}. \tag{98}$$

The non linear function Φ is also called an *activation function* in this context. Combinations of linear functions and nonlinear activation functions are also called *fully connected* (FC) layers since every component of the output vector is dependent on any component of the input. Fully connected layers form the core component of deep learning, since networks of arbitrary complexity can be created by putting multiple layers in sequence:

$$F(x|\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_k) = F_1(\,\cdot\,|\boldsymbol{\lambda}_1) \circ \cdots \circ F_k(\,\cdot\,|\boldsymbol{\lambda}_k)(\mathbf{x}). \tag{99}$$

Despite the usefulness of FC layers, there are many other layouts for deep learning algorithms in use. Even though the name neural network comes from the concept of fully connected layers and their resemblance of a biological network of neurons, we will call all deep learning architectures neural networks, irrespective of whether they are fully connected or not.

Having chosen a specific architecture for the neural network, one can start the learning process. To do so, it is necessary to quantify the performance of the NN for a given parameter combination. This is done by choosing a *loss function* $\mathcal{L}_F(\boldsymbol{\lambda}|D)$ that outputs a single value, based on the given data, for the specific choice of network parameters. The index F will be omitted in the following, but it should be clear that the loss function depends on the architecture of F. The task of *training* the network then becomes an optimization problem, where $\boldsymbol{\lambda}_{best}$ has to be found, such that

$$\mathcal{L}(\boldsymbol{\lambda}_{best}|D) = \min_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}|D), \tag{100}$$

i.e. such that $\boldsymbol{\lambda}_{best}$ minimizes the loss function. Using a fixed dataset in the learning process, the loss function only depends on the parameters $\boldsymbol{\lambda}$ and we can therefore denote it as $\mathcal{L}(\boldsymbol{\lambda})$ The optimization of a function in multiple variables is a well known mathematical problem, for which many methods exist. The optimization routines, also referred to as *optimizers*, typically require that the loss function is differentiable. Consequently, the NN must be differentiable too. One should note that for numerical reasons it suffices that all functions are differentiable almost everywhere, therefore, functions such as ReLu (eq. (94)) or LeakyReLu (eq. (95)), which are not differentiable at 0, still meet the requirement. A list of popular methods for optimizing a loss function are given in the following, based on [Rud17]:

- Gradient descent is the most basic optimization method. Starting at an initial value for the parameters $\boldsymbol{\lambda}_i$, the minimum of the loss function $\mathcal{L}(\boldsymbol{\lambda})$ is found by computing the direction of fastest increase, given through the gradient of $\mathcal{L}$ at $\boldsymbol{\lambda}_i$, $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_i)$ which can be obtained, using the chain rule. This computation process is commonly referred to as *backpropagation*. The parameters can be iteratively updated, using

$$\boldsymbol{\lambda}_{i+1} \coloneqq \boldsymbol{\lambda}_i - \alpha \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_i). \tag{101}$$

Here, α denotes the *learning rate*, i.e. the step size of the algorithm. Its value has to be chosen based on the given problem. Moreover, it is common to introduce *learning rate scheduling*, i.e. a learning rate α_i that changes from step to step. A common choice for an adaptive learning rate is *cosine annealing*, where the learning rate at each step is given as

$$\alpha_i \coloneqq \alpha_{min} + \frac{1}{2}(\alpha_{max} - \alpha_{min}) \left(1 + \cos\left(2\pi \frac{i}{N}\right)\right). \tag{102}$$

In this case, the learning rate starts at a maximal rate of $\alpha_0 = \alpha_{max}$ and continuously decreases, until it arrives at $\alpha_N = \alpha_{min}$ in the final iteration. The annealing process can additionally be reset once it reaches the minimum, for multiple cycles (see [LH17]) but we will use the simpler schedule given above.

When the parameters approach a local minimum of the loss function, the gradient approaches $\mathbf{0}$, which is why any local minimum of $\mathcal{L}$ is a fix point of the gradient descent algorithm. Naturally, there can be more than a single minimum of the loss function. Depending on the problem, it may therefore be necessary to restart the optimization process with a new set of initial parameter values. Additionally, evaluating the loss function with respect to the whole training dataset is relatively slow, depending on the size of the dataset.

- An alternative algorithm that tries to improve upon the weaknesses of gradient descent, is called *Minibatch Gradient Descent* (MGD) which belongs to the class of *Stochastic Gradient Descent* (SGD) algorithms. Instead of computing the gradient of the loss function on the entire training dataset at every epoch i, the full dataset D is randomly divided into m equally sized subsets D_{ij} at every epoch, where $D = \bigcup_{j=1}^{m} D_{ij}$. The parameters are iteratively updated, based on the gradient for every subset

$$\boldsymbol{\lambda}_i^{(j+1)} \coloneqq \boldsymbol{\lambda}_i^{(j)} - \alpha \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_i^{(j)} | D_{ij}), \tag{103}$$

$$\boldsymbol{\lambda}_{i+1}^{(0)} \coloneqq \boldsymbol{\lambda}_i^{(m)}. \tag{104}$$

  Using this method is numerically faster and the randomness introduced by subdividing the dataset allows the algorithm to more easily jump from one local minimum of the global loss function to another, making it more probable to find the absolute minimum. It has the disadvantage that convergence to the absolute minimum is more difficult, since there is a tendency to overshoot, as the minimum of the global loss function and minima of the batch loss functions do not necessarily align. This effect can be mitigated, by choosing larger minibatch sizes and by introducing a learning rate schedule that slowly decreases the learning rate (see [Rud17]).

- *Adaptive Moment Estimation* (Adam) is a commonly used modification of MGD and it is the optimizer that we use in this work. It allows adapting the step size to the estimated proximity to the convergence point, which helps to avoid overshooting the minimum. It follows an MGD-algorithm, where the batch updates are performed using the following algorithm:

The *Adam* algorithm. All operations on vectors are performed element-wise, i.e., $\mathbf{v}^2 = (v_1^2, \ldots, v_n^2)$, $\mathbf{v} + \epsilon = (v_1 + \epsilon, \ldots, v_n + \epsilon)$ and $\sqrt{\mathbf{v}} = (\sqrt{v}_1, \ldots, \sqrt{v}_n)$.

Require: α: Possibly adaptive step size.
Require: : β_1, $\beta_2 \in [0, 1)$: Exponential decay rates for the momentum estimates
Require: : $\boldsymbol{\lambda}_0$: Initial parameters
Require: : ϵ: A small value for numerical stability. Commonly chosen as 10^{-8}.
  $\mathbf{m}_0 \leftarrow \mathbf{0}$
  $\mathbf{v}_0 \leftarrow \mathbf{0}$
  $i \leftarrow 0$
  while $\boldsymbol{\lambda}_i$ has not converged **do**
    $i \leftarrow i + 1$
    $\mathbf{g}_i \leftarrow \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_{i-1})$
    $\mathbf{m}_i \leftarrow \beta_1 \mathbf{m}_{i-1} + (1 - \beta_1)\mathbf{g}_i$
    $\mathbf{v}_i \leftarrow \beta_2 \mathbf{v}_{i-1} + (1 - \beta_2)\mathbf{g}_i^2$
    $\mathbf{m}'_i \leftarrow \mathbf{m}_i / (1 - \beta_1^i)$
    $\mathbf{v}'_i \leftarrow \mathbf{v}_i / (1 - \beta_2^i)$
    $\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_{i-1} - \alpha \mathbf{m}'_i / \left(\sqrt{\mathbf{v}'_i + \epsilon}\right)$
  end while
  return $\boldsymbol{\lambda}_i$

Adam allows adapted effective step sizes for every parameter. Furthermore, it keeps an exponentially decaying memory of the gradient $\mathbf{m}_i$ and its square $\mathbf{v}_i$. This way, the effective step size automatically decreases when the decaying means of the gradient and the variance decrease. Adam has been empirically shown to work well in many ML scenarios and we found that it works well for our purposes. For a more detailed description of the algorithm, see [KB17].

5.2 Monte Carlo Inference

In section 2, we discussed that one of the central objects in Bayesian inference is posterior probability distribution eq. (4). Since the prior distribution is specified by choice and the evidence mathematically reduces to a normalization constant, the primary computational challenge in evaluating the posterior distribution is the likelihood function. The two most common kinds of problems that arise in statistical analyses are integration and optimization problems. In particular many statistical estimators, such as the mean eq. (A9) or credible regions eq. (6), require computing the expectation value of a function f with respect to the posterior distribution:

$$\mathbb{E}(f) = \int d\boldsymbol{\theta}\, f(\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathbf{y}). \tag{105}$$

Meanwhile, likelihood or maximum posterior estimators require to find the maximum of $p(\mathbf{y}|\boldsymbol{\theta})$ or $p(\boldsymbol{\theta}|\mathbf{y})$ respectively. We are interested in the likelihood function that arises in PTA analyses from eq. (90),

$$p(\delta\mathbf{t}_G|\boldsymbol{\theta}) = \frac{1}{[(2\pi)^{k-m}\, \det(\mathbf{G}^T\, \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G})\,]^{1/2}} \exp\left(-\frac{1}{2}\delta\mathbf{t}_G^T \left(\mathbf{G}^T\, \boldsymbol{\Sigma}(\boldsymbol{\theta})\, \mathbf{G}\right)^{-1} \delta\mathbf{t}_G\right). \tag{106}$$

Even in the analytically simplest case of a constant or "flat" prior distribution it is not possible to find analytical solutions for the most common statistical estimators. This necessitates to evaluate the relevant integrals numerically. Classical numerical methods such as Riemann integration and Newton-Raphson optimization and their improved versions (see [GRK21], chapters

3 and 7), tend to perform poorly in high dimensions (see chapter 1 of [RC99]).

In contrast, Monte Carlo methods address problems of integration and of optimization using random events. Approximations for expectation values can be computed as the arithmetic mean of N samples $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ drawn from the underlying distribution $p(\boldsymbol{\theta}|\mathbf{y})$ as

$$\mathbb{E}(f) \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{\theta}_i) \tag{107}$$

and optimization problems $\boldsymbol{\theta}_{best}$ can be solved by taking the maximal assumed value on the sample distribution

$$\boldsymbol{\theta}_{best} \approx \arg\max_{\boldsymbol{\theta}_i} f(\boldsymbol{\theta}_i). \tag{108}$$

A third type of problem one often faces is the problem of visualizing a marginalized probability distribution, e.g. the posterior distribution for a subset of the parameters. Here, Monte Carlo methods solve this problem in a straightforward way by introducing a binning on the quantities of interest and displaying the resulting histogram of the sampled distribution.

As can be seen, Monte Carlo algorithms are versatile and can be used in many numerical applications. There are many different algorithms optimized for specific problems and an overview can be found in [BZ20]. In PTA analyses the preferred Monte Carlo algorithm in use is an adapted version of the *Metropolis Hastings* algorithm, which constructs a Markov chain in parameter space $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ that converges to a sampling of the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$. It belongs to the class of Monte Carlo Markov Chain (MCMC) algorithms. The algorithm uses a proposal distribution $q(\boldsymbol{\theta}|\boldsymbol{\theta}')$ based on the previous parameter point in the chain and an acceptance algorithm. Starting at a point $\boldsymbol{\theta}_i$, a candidate for a jump is drawn from the candidate distribution $\tilde{\boldsymbol{\theta}}_{i+1} \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}_i)$. This jump is then accepted or rejected based on the quotient of the respective posteriors

$$\boldsymbol{\theta}_{i+1} = \begin{cases} \tilde{\boldsymbol{\theta}}_{i+1} & \text{with probability } \rho(\tilde{\boldsymbol{\theta}}_{i+1}|\boldsymbol{\theta}_i) \\ \boldsymbol{\theta}_i & \text{with probability } 1 - \rho(\tilde{\boldsymbol{\theta}}_{i+1}|\boldsymbol{\theta}_i) \end{cases} \tag{109}$$

with

$$\rho(\boldsymbol{\theta}|\boldsymbol{\theta}') = \min\left\{ \frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta}')} \frac{p(\mathbf{y}|\boldsymbol{\theta})}{p(\mathbf{y}|\boldsymbol{\theta}')}, 1 \right\}. \tag{110}$$

From the definition of ρ follows that a posterior density increasing jump will always be accepted, while a jump that decreases the posterior is only randomly accepted based on the posterior quotient. Accepting posterior decreasing jumps as well insures that the Markov chain does not converge towards a maximum of the posterior function but instead towards a general sampling of the posterior distribution [Rob16]. There are many different possible choices of candidate distribution, one of the simplest choices being a normal distribution centered around the previous point. The choice of candidate distribution can be further optimized to achieve faster convergence. The standard algorithm used by the NANOGrav collaboration is a *Parallel Tempering Markov Chain Monte-Carlo algorithm*, implemented in [EH17].

A full MCMC posterior computation for a NANOGrav dataset requires sample sizes of a least $\mathcal{O}(10^6)$ and takes computation times of the order of weeks. The computational bottleneck here is

the likelihood function $p(\delta\mathbf{t}_G|\boldsymbol{\theta})$ which has to be evaluated for every proposed jump. Evaluation this function involves the computation of the inverse of a large matrix ($\mathcal{O}(1000 \times 1000)$) as well as its determinant. The aim of this work is to accelerate the sampling process using machine learning augmented *simulation-based inference*.

5.3 Simulation-Based Inference

This section is based on [CBL20]. First proposed by [PM18], the aim of simulation-based inference (SBI) algorithms is to infer statistical parameter estimators from data, without having to solve an often computationally challenging inversion problem like the one seen in eq. (90). By a *simulator* we mean a computer program that simulates data based on a given model. Mathematically it corresponds to a random variable that allows to sample data based on a set of input parameters. A common context is a physical model as described in section 2, that predicts observations $\mathbf{y}$ based on some physical parameters $\boldsymbol{\theta}$. The model used in this work is given in eq. (74) with a deterministic contribution discussed in section 4.4 and various noise models that will be introduced in section 8.1.2.

Using the simulator, one proceeds to perform either frequentist or Bayesian inference (see section 2). There are different algorithms for simulation-based inference, the most direct one being Approximate Bayesian Computation (ABC), where the idea is to produce a sampling of a posterior distribution $p(\boldsymbol{\theta}|\mathbf{y}_{obs})$ based on observational data $\mathbf{y}_{obs}$ by drawing $\boldsymbol{\theta}$ from a prior distribution and accepting or rejecting $\boldsymbol{\theta}$ based on the likeness of the simulation output $\mathbf{y}_{sim}$ for $\boldsymbol{\theta}$ to $\mathbf{y}_{obs}$ (for more detail see [BZB02]).

Advancements in machine learning over the last decade have made neural networks an interesting candidate for density estimation in high dimensions [LBH15]. They allow for a different, often more efficient, way of performing SBI. Having simulated a dataset that consists of pairs of parameters and observations $D = \{\mathbf{y}_i, \boldsymbol{\theta}_i\}$ the aim is to train a neural network to learn the underlying conditional probability distributions. Different approaches either focus on learning the likelihood function $p(\mathbf{y}|\boldsymbol{\theta})$ or the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$. The approach we are going to use in this work is *amortized posterior estimation* (APE) (see [CBL20]). A flowchart of the workflow can be seen in fig. 2.

Beginning with a chosen prior distribution for the parameters $p_0(\boldsymbol{\theta})$, one creates a dataset of parameter values sampled from the prior. The parameter dataset is passed to the simulator which simulates a predicted observation $\mathbf{y}$ for each $\boldsymbol{\theta}$. A neural network is then trained on the dataset of pairs $(\mathbf{y}, \boldsymbol{\theta})$ to learn a conditional posterior distribution $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}|\mathbf{y})$ for every given observation. Mathematically, the network is shown pairs $(\mathbf{y}, \boldsymbol{\theta})$ with relative frequencies corresponding to the probability

$$(\mathbf{y}, \boldsymbol{\theta}) \sim p(\mathbf{y}|\boldsymbol{\theta})\, p_0(\boldsymbol{\theta}) = p(\mathbf{y}, \boldsymbol{\theta}), \tag{111}$$

i.e. the joint probability distribution in the Bayesian paradigm. Depending on the choice of loss function, this allows to learn the Bayesian posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ (see sections 6.1 and 6.2). Once trained, the network can be evaluated on the given real experimental observations $\tilde{\mathbf{y}}$ and an approximation of the posterior distribution is obtained by sampling from the learned distribution $p_{\boldsymbol{\lambda}}(\boldsymbol{\theta}|\tilde{\mathbf{y}})$. It is additionally possible to include iterative active learning by using the posterior distribution obtained in one iteration as a prior distribution for the next iteration. In this work we are going to use the posterior distributions obtained via conventional MCMC methods as a guideline for the chosen prior distribution and will therefore not use additional active learning.

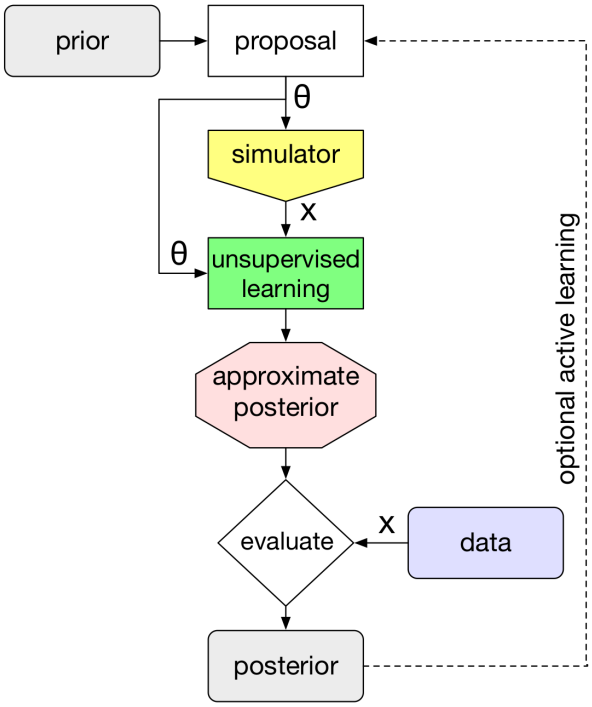


Figure 2: A depiction of the workflow used for amortized posterior estimation. (Figure taken from [CBL20]).

APE offers the advantage that the likelihood function does no longer need to be evaluated directly. Instead the stochastic model is sampled, which may be advantageous, if sampling is comparatively cheap. Additionally, the time required to train, and possibly also fine-tune the neural network, has to be taken into account. To obtain the posterior distribution, the neural network then has to be evaluated on the measured data. Depending on the quality of the learned posterior distribution, it may additionally be necessary to employ reweighting techniques to improve the accuracy of the learned posterior distribution.

5.4 Importance Sampling

This section is based on [Kon92]. The aim of amortized posterior estimation, as explained in the previous section, is to learn the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ of a physical problem using a neural network. The posterior is then often used to compute statistical estimators (see section 2), which commonly corresponds to computing the expectation value of a function g of the form

$$\mathbb{E}_p(g) = \int d\boldsymbol{\theta}\, g(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{y}). \tag{112}$$

Once trained, the network reproduces an approximation $\tilde{p}(\boldsymbol{\theta}|\mathbf{y})$ of the true posterior distribution and it can be used for the computation of the expectation value using

$$\mathbb{E}_p(g) = \int d\boldsymbol{\theta}\, g(\boldsymbol{\theta}) \frac{p(\boldsymbol{\theta}|\mathbf{y})}{\tilde{p}(\boldsymbol{\theta}|\mathbf{y})} \tilde{p}(\boldsymbol{\theta}|\mathbf{y}) = \int d\boldsymbol{\theta}\, \tilde{g}(\boldsymbol{\theta})\, \tilde{p}(\boldsymbol{\theta}|\mathbf{y}), \tag{113}$$

where we defined the new random variable $\tilde{g}$ and introduce the *weight function* w as

$$\tilde{g}(\boldsymbol{\theta}) := g(\boldsymbol{\theta}) \frac{p(\boldsymbol{\theta}|\mathbf{y})}{\tilde{p}(\boldsymbol{\theta}|\mathbf{y})} =: g(\boldsymbol{\theta})\, w(\boldsymbol{\theta}|\mathbf{y}). \tag{114}$$

We note that computing the expectation value $\mathbb{E}_p(g)$ using the true distribution $p(\boldsymbol{\theta}|\mathbf{y})$ is equivalent to computing the expectation value $\mathbb{E}_{\tilde{p}}(\tilde{g})$ using the learned distribution $\tilde{p}(\boldsymbol{\theta}|\mathbf{y})$. In the case of $w(\boldsymbol{\theta}|\mathbf{y}) \approx 1$, we can simply replace the true distribution with the learned distribution without changing the result, which corresponds to the case of a perfectly learned posterior distribution. Having drawn a sample $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m$ from $\tilde{p}(\boldsymbol{\theta}|\mathbf{y})$ with corresponding weights $w(\boldsymbol{\theta}_1|\mathbf{y}), \dots, w(\boldsymbol{\theta}_m|\mathbf{y})$, a corrected estimator for the expectation value of g can be computed by performing a weighted average

$$\mathbb{E}(g) \approx \frac{\sum_{i=1}^{m} g(\boldsymbol{\theta}_i)\, w(\boldsymbol{\theta}_i|\mathbf{y})}{\sum_{i=1}^{m} w(\boldsymbol{\theta}_i|\mathbf{y})}. \tag{115}$$

It is often practically impossible or computationally expensive to compute $w(\boldsymbol{\theta}|\mathbf{y})$ at a large number of points since it requires computing both the true likelihood $p(\mathbf{y}|\boldsymbol{\theta})$ and the true evidence $p(\mathbf{y})$. In particular the computation of the evidence (see eq. (3)) requires evaluating an integral that is often intractable in practice

$$p(\mathbf{y}) = \int d\boldsymbol{\theta}\, p(\mathbf{y}|\boldsymbol{\theta})\, p_0(\boldsymbol{\theta}). \tag{116}$$

For this reason we alternatively consider

$$\tilde{w}(\boldsymbol{\theta}|\mathbf{y}) \coloneqq \frac{p(\mathbf{y}|\boldsymbol{\theta}) p_0(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta}|\mathbf{y})} = p(\mathbf{y})\, w(\boldsymbol{\theta}|\mathbf{y}) \tag{117}$$

which only requires the computation of the likelihood at one point. Here $\tilde{w}(\boldsymbol{\theta}|\mathbf{y}) = p(\mathbf{y})$ corresponds to the case of a perfectly learned distribution and, since $p(\mathbf{y})$ is a constant, it holds that $\mathrm{Var}(w) = \mathrm{Var}(\tilde{w})$. It is further possible to show that the variance of the weights allows a comparison of the efficiency of sampling between p and $\tilde{p}$. For this we consider m independent sample draws $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m$ from $p(\mathbf{y}|\boldsymbol{\theta})$ and $\tilde{\boldsymbol{\theta}}_0, \dots, \tilde{\boldsymbol{\theta}}_m$ from $\tilde{p}(\mathbf{y}|\boldsymbol{\theta})$ respectively. The sample averages

$$\boldsymbol{\mu}_m \coloneqq \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\theta}_i, \quad \tilde{\boldsymbol{\mu}}_m \coloneqq \frac{1}{m} \sum_{i=1}^{m} \tilde{\boldsymbol{\theta}}_i \tag{118}$$

are again random variables with

$$\mathbb{E}(\boldsymbol{\mu}_m) = \mathbb{E}_p(\boldsymbol{\theta}), \quad \mathbb{E}(\tilde{\boldsymbol{\mu}}_m) = \mathbb{E}_{\tilde{p}}(\boldsymbol{\theta}) \quad \text{and} \tag{119}$$

$$\lim_{m\to\infty} \mathrm{Var}(\boldsymbol{\mu}_m) = \lim_{m\to\infty} \mathrm{Var}(\tilde{\boldsymbol{\mu}}_m) = 0. \tag{120}$$

The quotient of the variances gives a measure of the relative sampling efficiency for a fixed amount of samples

$$\epsilon_m = \frac{\mathrm{Var}(\boldsymbol{\mu}_m)}{\mathrm{Var}(\tilde{\boldsymbol{\mu}}_m)} \tag{121}$$

and it can be shown that it is related to the variance of m samples for $\tilde{w}$ using the standardized weights $\bar{w}_m$

$$\bar{w}_m \coloneqq \frac{\tilde{w}}{\sum_{i=1}^{m} \tilde{w}(\boldsymbol{\theta_i})} \tag{122}$$

as

$$\epsilon_m = 1 + \mathrm{Var}(\bar{w}_m). \tag{123}$$

The *weighting efficiency* ϵ and the related *effective sample size* n_{eff} can be shown to equal

$$\epsilon = \frac{(\sum_{i=1}^{m} \bar{w}_m(\boldsymbol{\theta_i}))^2}{m \sum_{i=1}^{m} \bar{w}_m(\boldsymbol{\theta_j})^2} \tag{124}$$

$$n_{\text{eff}} = m\,\epsilon = \frac{(\sum_{i=1}^{m} \bar{w}_m(\boldsymbol{\theta_i}))^2}{\sum_{i=1}^{m} \bar{w}_m(\boldsymbol{\theta_j})^2}, \tag{125}$$

(see [Kon92]), which gives an estimate for the necessary amount of reweighted samples from the learned distribution to have an equal variance to m samples from the true distribution. It can therefore be considered as another metric for evaluating the performance of the trained network. One should note that the standardized weights can equally be used to calculate the weighted average from eq. (115) since the evidence cancels out.

6 Generative Neural Networks

To use amortized posterior estimation, as proposed in the previous section, a neural network architecture, capable of learning the underlying conditional distribution of a given dataset, is necessary. The two candidate architectures we use in this work are *Normalizing Flows* and *Diffusion Networks*, both of which will be introduced in the following sections. Almost all objects we use throughout the introduction of the machine learning architectures will be vector valued. For notational simplicity, we will not write vector objects in bold font in the sections on the machine learning architectures.

6.1 Normalizing Flows

6.1.1 Concept

This section is based on [RM16] where normalizing flows were first proposed. The aim of a normalizing flow (NF) is to use a neural network to define a bijective mapping between two probability spaces, that allows to learn the probability distribution of interest p_t. Additionally, the network is built in such a way that the Jacobian matrix and the inverse are relatively cheap to compute. By $\theta \in \Theta$, the *data space* and its objects are denoted, while the *latent space* and its objects are denoted by $z \in Z$. Therefore, Z and Θ must have equal dimensionality. A NF $F_\lambda : Z \to \Theta$ usually consists of multiple sequential bijective mappings G_{λ_i}

$$F_\lambda(z) = G_{\lambda_{m-1}} \circ \cdots \circ G_{\lambda_0}(z). \tag{126}$$

Here, λ_i denotes the subset of trainable parameters that govern the behavior of the i-th mapping. z_i denote the vectors the individual transformations map between, i.e. $z_{i+1} \coloneqq G_{\lambda_i}(z_i)$, with $z_m \coloneqq \theta$ and $z_0 \coloneqq z$. The Jacobian matrix of the full map is then given by the product

$$DF_\lambda(z) = \prod_{i=1}^{m} DG_{\lambda_i}(z_i). \tag{127}$$

Given a latent probability distribution on Z, $p_l(z)$, $\theta_\lambda \coloneqq F_\lambda(z)$ can be interpreted as a random variable with a given probability distribution of

$$p_\lambda(\theta_\lambda) = p_l\left(F_\lambda^{-1}(\theta_\lambda)\right) \left|\det DF_\lambda^{-1}(\theta_\lambda)\right|, \tag{128}$$

(see eq. (A4)). In the following, the index λ on θ_λ will be omitted, but it should be clear from the context which θ is referred to. Given the invertible nature of the normalizing flow, it can be construct to map from data space to latent space and vice versa depending on the problem at hand. Once a normalizing flow is trained, it allows to quickly sample from the learned distribution on the data space, by sampling z from the latent distribution and mapping z to the data space via $F_\lambda(z)$. In order to learn a good approximation of the true distribution however, a loss function is needed, that measures the similarity between the true- and the learned distribution. There are multiple possible options, but the most common choice is the *Kullback-Leibler divergence* (KL-divergence). By denoting the true distribution as $p_t(\theta)$, it is defined via

$$\mathcal{L}_{KL}(\lambda) = \int d\theta\, p_t(\theta)\, \log \frac{p_t(\theta)}{p_\lambda(\theta)} \tag{129}$$

$$= -\int d\theta\, p_t(\theta) \log p_\lambda(\theta) + \text{const.}, \tag{130}$$

where the term that does not depend on λ was absorbed into a constant. The data- distribution $p_D(\theta)$ can then be used as an approximation of the true distribution. Therefore, omitting the constants irrelevant for training, the KL-divergence becomes

$$\mathcal{L}_{KL}(\lambda) = -\sum_{\theta \in D} \log p_\lambda(\theta) \tag{131}$$

$$= -\sum_{\theta \in D} \log p\left(F_\lambda^{-1}(\theta)\right) \log \left|\det DF_\lambda^{-1}(\theta)\right|, \tag{132}$$

which corresponds to the negative logarithmic likelihood of the data for the model $\theta = F_\lambda(z)$. This is the loss function used in training the normalizing flows for this work. For conceptual simplicity, we have treated the probabilty distribution to be learned as an unconditional distribution. In the context of SBI, as described in the previous section, we are interested in conditional probability distributions. The idea behind unconditional NFs generalizes directly to conditional NFs, which consider an additional *context* y. Therefore, the true distribution becomes a conditional distribution $p(\theta|y)$, whereas the NF $F_\lambda(z|y)$ acquires an additional argument y, while maintaining invertibility in z. Consequently, the learned distribution $p_\lambda(\theta|y)$ also becomes a conditional distribution and the loss from eq. (129) has to be computed by integrating over the Bayesian joint probability distribution. The latent distribution can also be chosen as a conditional distribution to allow for more expressivity. Commonly, this is not necessary to capture the underlying probability density and an unconditional latent distribution $p_l(z)$ suffices. The KL-loss becomes

$$\mathcal{L}_{KL}(\lambda) = -\sum_{(\theta,y) \in D} \log p_\lambda(\theta|y) \tag{133}$$

$$= -\sum_{(\theta,y) \in D} \log p\left(F_\lambda^{-1}(\theta|y)\right) \log \left|\det DF_\lambda^{-1}(\theta|y)\right|. \tag{134}$$

In consequence, the main computational hurdles in evaluating the loss function are the inversion of F_λ and the computation of the determinant $\det DF^{-1}$. This strongly motivates the choice of individual transforms G_{λ_i} used in normalizing flows.

6.1.2 Coupling Blocks

This section is based on [Dur+19]. There are multiple possible architectures, that can be considered to build a NF and throughout this work we use different ones. One possible choice is the method of *coupling blocks*. Consider a component wise invertible transformation $C(z|r)$, dependent on parameters r. By 'component wise' is meant that the j-th component of the output vector only depends on the j-th component of the input vector. We denote the transform acting on one component of z as $C(z_j|r)$ giving the full form of

$$C(z|r) = (C(z_1|r), C(z_2|r)) \dots). \tag{135}$$

For a given d-dimensional input z to a coupling block, z is split, $z = (z^A, z^B)$, where z^A and z^B are $d/2$-dimensional vectors (depending on whether $2|d$, the input can be extended by one dimension with 0). z^A gets passed to a trainable fully connected network Φ, which does not necessarily need to be invertible. Φ is called the *sub-network* of the block. The output $\Phi(z^A)$ is used as the parameter vector for the invertible transformation, while z^B is used as the input. Meanwhile, z^A is kept constant by the transform, giving the full transformation of the block as

$$F(z^A, z^B) = (z^A, C(z^B|\Phi(z^A))). \tag{136}$$

The inverse of this operation is straightforward to compute given that the inverse of C is known

$$F^{-1}(z^A, z^B) = (z^A, C^{-1}(z^B|\Phi(z^A))). \tag{137}$$

The Jacobian matrix of F is of block diagonal form

$$DF(z) = \begin{pmatrix} \mathbb{1} & \mathbf{0} \\ \mathbf{A} & \mathbf{B} \end{pmatrix} \qquad \text{with} \tag{138}$$

$$\mathbf{A} = \frac{\partial C(z^B|\Phi(z^A))}{\partial z^A}(z) \quad \text{and} \tag{139}$$

$$\mathbf{B} = \frac{\partial C(z^B|\Phi(z^A))}{\partial z^B}(z). \tag{140}$$

Since C acts component wise on z^B, $\mathbf{B}$ is diagonal, hence the full Jacobian matrix is of lower triangular form. Therefore, the determinant, given as

$$\det DF(z) = \prod_{j=1}^{d/2} \frac{\partial C(z_j^B|\Phi(z^A))}{\partial z_j^B}(z), \tag{141}$$

only requires $\mathcal{O}(d/2)$ operations and remains comparatively cheap to compute, even for large dimensions. In particular, the matrix $\mathbf{A}$ does not not have to be computed at all. The concept of coupling blocks can also be applied within conditional NFs by concatenating the condition y to z^A, as the input to Φ. The transformation becomes

$$F(z|y) = [z^A, C(z^B|\Phi(z^A, y))], \tag{142}$$

where inverse and determinant of the Jacobian are computed analogously. It is common to use two transforms within one block to transform both z^A and z^B. The output is given as

$$F(z^A, z^B) = (C(z^A|\Phi_1(z^B)), C(z^B|\Phi_2(z^A))). \tag{143}$$

So far the proposed coupling block setup has only limited expressivity since all operations are performed point wise which does not allow for dependencies of the components of z^A and z^B among themselves. This problem is solved by setting multiple blocks in sequence and permuting the components of z between blocks, which leaves the absolute value of Jacobian determinant invariant.

6.1.3 Autoregressive Blocks

This section is based on [Pap+21]. A different way to build a normalizing flow is to use autoregressive blocks, where the *autoregressive* property is used, to ensure that both inversion and computation of the log determinant are tractable. 'Autoregressive' means, that the transformation is of the form

$$F(z) = (C_1(z_1), C_2(z_2|\Phi_2(z_1)), \ldots, C_d(z_d|\Phi_d(z_1, \ldots, z_{d-1}))), \tag{144}$$

where the C_i are invertible transforms in z_i. The Φ_i are trainable, not necessarily invertible, fully connected networks, which are also called *sub-networks* of the block. This ensures that the Jacobian matrix DF is of lower triangular form and its determinant is therefore given as

$$\det DF(z) = \prod_{j=1}^{d} \frac{\partial C_j(z_j|\Phi_j(z_1,\ldots,z_{j-1}))}{\partial z_j}(z). \tag{145}$$

Additionally it allows the inverse F^{-1} to be computed iteratively, as

$$F^{-1}(z) = \left(C_1^{-1}(z_1), C_2^{-1}(z_2|\Phi_2(C_1^{-1}(z_1))), \ldots\right). \tag{146}$$

While within the coupling block architecture both forward and backward directions can be computed in parallel, in the autoregressive framework only the forward direction is parallelizable, while the inverse direction has to be computed in sequence. In exchange, one autoregressive block is generally more expressive than a coupling block, since it allows for more dependencies between input and output components. Analogously to the coupling block architecture, the idea of autoregressive flows can be extended to conditional normalizing flows.

6.1.4 Rational Quadratic Splines

This section based on [Dur+19] and [Hei24]. For the invertible transformation C, used within a coupling layer or autoregressive layer, there are different possible choices. In this work we use *(monotonic) rational quadratic splines* (RQS) as introduced in [Dur+19]. RQS transformations are monotonic maps from an interval $[A, B]$ to itself. Outside of $[A, B]$, the transformation is defined to be the identity. As discussed in the previous sections, the transformation acts on input vectors by transforming them component wise. By rational quadratic spline transformation we mean a piecewise defined function, that consists of rational quadratic functions or splines on each bin. For a spline transformation consisting of K splines, there are $K+1$ boundary points. Setting $A = x_1, \ldots, x_{K+1} = B$ as the boundary points, the i-th spline s_i, defined on the interval $[x_i, x_{i+1}]$, is fully defined by its values y_i, y_{i+1} and derivatives d_i, d_{i+1} at x_i and x_{i+1}. The points (x_i, y_i) are also called the knots of the transformation. The additional requirement that

$$\operatorname{sgn} s_i'(x_i) = \operatorname{sgn} s_i'(x_{i+1}) = \operatorname{sgn} \frac{s_i(x_{i+1}) - s_i(x_i)}{x_{i+1} - x_i} \tag{147}$$

suffices to ensure monotony as shown in [GD82]. Under the given conditions the numerical form of the i -th spline is given as

$$s_i(x) \coloneqq \frac{p_i(x)}{q_i(x)} \coloneqq y_i + \frac{(y_{i+1} - y_i)[\alpha_i \xi_i(x)^2 + d_i \xi_i(x)(1 - \xi_i(x))]}{\alpha_i + [d_{i+1} + d_i - 2\alpha_i]\xi_i(x)(1 - \xi_i(x))}, \quad \text{where} \tag{148}$$

$$\alpha_i \coloneqq \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad \text{and} \tag{149}$$

$$\xi_i(x) \coloneqq \frac{x - x_i}{x_{i+1} - x_i}. \tag{150}$$

Since s_i is monotonous, it is invertible and as a rational quadratic function, it can be analytically inverted by solving for the roots of a second degree polynomial. The sign of the derivative then determines the correct root to choose and the inverse is given as

$$s_i^{-1}(y) = x_k + \frac{2c_i(y)\,(x_{i+1} - x_i)}{-b_(y) - \sqrt{b_i(y)^2 - 4a_i(y)\,c_i(y)}}, \qquad \text{where} \tag{151}$$

$$a_i(y) = (y_{i+1} - y_i)[\alpha_i - d_i] + (y - y_i)[d_{i+1} + d_i - 2\alpha_i], \tag{152}$$

$$b_i(y) = (y_{i+1} - y_i)\,d_i - (y - y_i)[d_{i+1} + d_i - 2\alpha_i] \qquad \text{and} \tag{153}$$

$$c_i(y) = -\alpha_i(y - y_i). \tag{154}$$

The derivative of eq. (148) can also be analytically calculated and results in

$$s_i'(x) = \frac{\alpha_i^2[d_{i+1}\,\xi_i(x)^2 + 2\alpha_i\,\xi_i(x)\,(1 - \xi_i(x)) + d_i\,(1 - \xi_i(x))^2]}{[\alpha_i + (d_{i+1} + d_i - 2\alpha_i)\,\xi_i(x)\,(1 - \xi_i(x))]^2}. \tag{155}$$

This allows the computation of the logarithm of the determinant of the Jacobian matrix from eq. (134) and eq. (141). In total one has $K + 1$ boundary points x_i with $K + 1$ corresponding functional values y_i and $K + 1$ derivatives d_i. The condition that the transformation is bijective on $[A, B]$, gives two constraints on the $x-$ and y-values at $i = 1$ and $i = K + 1$ and the derivatives d_1 and d_{K+1} are set to 1, to ensure that the transformation is differentiable at the transition points to the identity. This leaves a total of $3K - 3$ free parameters.

As discussed in the previous section, the input vector to the RQS-transform z is split into z^A and z^B and z^A is used to parameterize the transform (i.e. the concatenation of z^A and the context in the conditional case, see eq. (142)). z^A is then passed to a fully connected layer with $3K - 1$ output dimension. The output λ can then be split into three vectors $\lambda = (\lambda_x, \lambda_y, \lambda_d)$, where each λ_x and λ_y have dimension K and λ_d has dimension $K - 1$. λ_x and λ_y are passed through a softmax function (eq. (98)) and are multiplied by $2B$. The result is used as the bin widths and heights, giving the position of the knots. One should note that the softmax function ensures that the sum of the components of the output equals one. Therefore one looses one degree of freedom by using it, giving $K - 1$ effective parameters for each. λ_d is passed through a softplus function (eq. (96)) and the result is used as the derivatives at the internal knots.

6.2 Diffusion Models

This section is based on [al23]. A different approach to building a generative neural network, capable of learning a probability distribution from a given sample dataset, are *diffusion models* proposed by [HJA20]. The idea is to continuously deform a latent distribution to resemble the distribution of interest. We denote the data space and its elements as $\theta \in \Theta$ and the latent space and its elements as $z \in Z$. The latent probability distribution is denoted as $p_l(z)$ and the true distribution by $p_t(\theta)$. Mathematically, the idea is to describe the problem in terms of a stochastic differential equation for a random process $x(t)$ of the form

$$\frac{dx(t)}{dt} = v(x(t), t). \tag{156}$$

It can be shown that a differential equation for a random process of this form can equivalently be stated as a differential equation for the probability distribution of the process, or *probability path*, as

$$\frac{\partial p(x,t)}{\partial t} = -\mathrm{div}[p(x,t)\,v(x,t)]. \tag{157}$$

This condition is also called the continuity equation and describes the local conservation of probability. We say in this case that $v(x,t)$ generates the probability path $p(x,t)$. Imposing the boundary conditions

$$p(x,t=0) = p_l(z) \quad \text{and} \tag{158}$$
$$p(x,t=1) = p_t(\theta) \tag{159}$$

ensures, that any probability path $p(x,t)$ that solves the differential equation, describes a smooth transition from the latent distribution to the data distribution. The aim is to teach a neural network to learn the generating velocity field $v(x,t)$. Once we have obtained a network approximation $v_\lambda(x,t)$ of $v(x,t)$, it is possible to sample from p_t by sampling $x_1 \sim p_l$ and computing the sample path for x_1 at $t=1$ given as the solution of

$$\frac{\partial x}{\partial t}(t) = v_\lambda(x(t),t), \quad \text{with initial condition} \tag{160}$$
$$x(0) = x_1. \tag{161}$$

Using modern solvers for ODEs, this can be quickly evaluated. A straightforward objective for learning $v(x,t)$ is the least squares loss

$$\mathcal{L}(\lambda) = \int_0^1 dt \int dx\, p(x,t)\, \|v(x,t) - v_\lambda(x,t)\|^2. \tag{162}$$

There is no unique choice for $v(x,t)$ to model the network after. We will use the method of *conditional flow matching*, introduced by [al23]. Given a sample x_0, the aim is to find a conditional probability path $p(x,t|x_0)$, $t \in [0,1]$, such that

$$p(x,t=0|x_0) = p_l(x) \quad \text{and} \tag{163}$$
$$p(x,t=1|x_0) = \delta(x - x_0). \tag{164}$$

By marginalizing over x_0 using the data distribution, we define the marginal probability path $p_m(x,t)$ as

$$p_m(x,t) = \int dx_0\, p(x,t|x_0) p_t(x_0). \tag{165}$$

The marginal probability path has the properties

$$p_m(x,t=0) = \int dx_0\, p_l(x)\, p_t(x_0) = p_l(x) \quad \text{and} \tag{166}$$
$$p_m(x,t=1) = \int dx_0\, \delta(x - x_0)\, p_t(x_0) = p_t(x), \tag{167}$$

i.e. it fulfills the boundary conditions required. To construct such a conditional probability path, we consider the conditional random process

$$x(t|x_0) = t x_0 + (1-t) r, \quad r \sim p_l(r) = \mathcal{N}[0,\sigma](r), \tag{168}$$

where the choice of a normal distribution centered around zero with standard deviation σ was made for the latent distribution. The corresponding conditional probability path can be computed for a linear combination of random variables (see appendix A.1) and is given as

$$p(x,t|x_0) = \mathcal{N}[tx_0, (1-t)\sigma](x), \tag{169}$$

which by construction fulfills the boundary conditions in eq. (164). The corresponding vector field is then given as

$$v(x,t|x_0) = \frac{x_0 - x}{1-t} \tag{170}$$

which is easy to verify by checking eq. (156) for the proposed random process. This equivalently implies, that $v(x,t|x_0)$ generates the conditional probability path $p(x,t|x_0)$. It can be show that the marginalized vector field is obtained via

$$v_m(x,t) = \int dx_0 \frac{p_t(x_0)\, p(x,t|x_0)}{p_m(x,t)}\, v(x,t|x_0). \tag{171}$$

Consequently, $v_m(x,t)$ generates the marginalized probability path $p_m(x,t)$ and is therefore a suitable velocity field to be learned by the neural network. This way the loss function from eq. (162) becomes

$$\mathcal{L}(\lambda) = \int_0^1 dt \int dx\, p_m(x,t)\, \|v_\lambda(x,t) - v_m(x,t)\|^2 \tag{172}$$

$$= \int_0^1 dt \int dx_0 \int dx\, p_t(x_0)\, p(x,t|x_0)\, \|v_\lambda(x(t|x_0),t) - v(x,t|x_0)\|^2. \tag{173}$$

Using the data distribution, this expression can be numerically evaluated by computing

$$\mathcal{L}(\lambda) = \mathbb{E}_{t\sim u(0,1), x_0\sim p_D(x_0), x\sim p(x,t|x_0)} \|v_\lambda(x(t|x_0),t) - v(x,t|x_0)\|^2, \tag{174}$$

where $u(0,1)$ denotes a uniform distribution on the interval $[0,1]$, the data distribution p_D is used as an approximation for the true distribution p_t. Since learning $v_m(x,t)$ does not require the network to be invertible, a fully connected neural network can be chosen for v_λ.

As in the case of NFs, diffusion models can also be used to learn a conditional probability density. This generalization works completely analogous to the NF-case, by replacing all quantities with conditional versions.

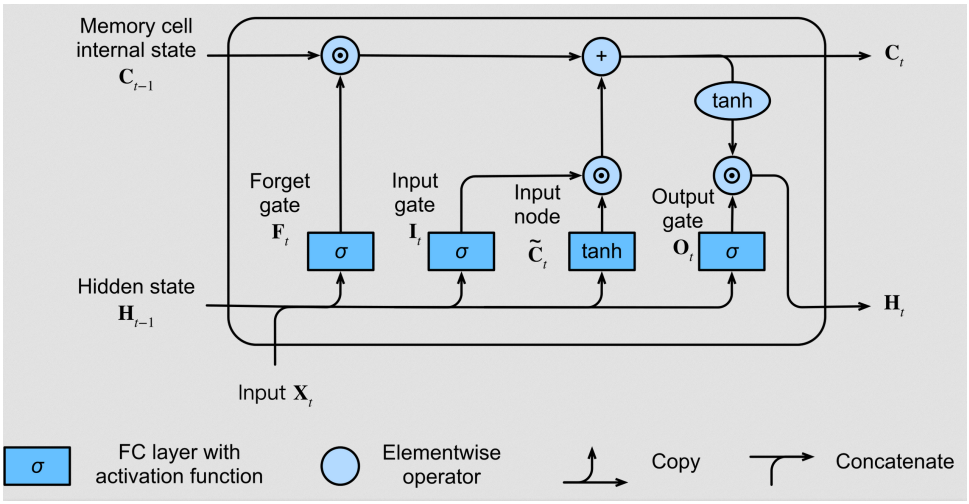


Figure 3: Diagram depicting the structure of an LSTM network. Figure taken from [Zha+23].

7 Encoder Networks

In this work the aim is to learn a conditional posterior density $p(\boldsymbol{\theta}|\delta\mathbf{t}_G)$, with corresponding likelihood given in eq. (90). In the previous section generative neural networks were introduced, capable of this task. The conditions for the learned density are the projected timing residuals $\delta\mathbf{t}_G$ and their dimension is given by the total amount of pulsar observations minus the total degrees of freedom of the timing model. For a full NANOGrav 12.5 yr analysis the number of observations is of $\mathcal{O}(6000)$, and since the amount of timing model parameters is comparatively small, the dimension of $\delta\mathbf{t}_G$ is in the same range.

Large inputs to a neural network increase the amount of parameters and the computational cost required for training and evaluating the network. For this reason, it can be advantageous to exploit symmetries in the data to reduce the dimensionality of the problem. Since those symmetries are in many cases hidden, one way of trying to find them is to train a neural network to "encode" the relevant information contained in a large vector in a smaller representation vector, first proposed in [HS06]. The representation vector is also commonly called the *embedding* vector. The parameters of the encoder network are trained together with the parameters of the generative network.

Several different architectures have been proposed for this task. The two methods we are using in this work, *long short-term memory encoders* and *transformer encoders*, are going to be introduced in the following sections.

7.1 Long Short-Term Memory Encoders

This section is based on [SSB14]. Long short-term memory encoders (LSTMs) are a neural network architecture, designed to take large sequential data and extract and condense the relevant information contained therein. It is a type of recurrent neural network, meaning that it treats its input sequentially, where the output of previous iterations can have an influence on following iterations. The iterations are performed within *memory blocks* or *memory cells*. Fig. 3 shows a diagram of one memory block within the LSTM architecture. In an LSTM network with K memory blocks the input is subdivided into K smaller vectors x_t, of dimension ceil(n/K) (the input vector can be extended with zeros until $K|n$). The output of the network h is computed iteratively, where each iteration computes a segment h_t of h. The network uses a memory state c_t that is updated on each iteration. Within every iteration, the output of

previous iteration h_{t-1} and the input of the current iteration x_t are concatenated and used to compute three gate vectors, f_t, i_t and o_t by using three independent, fully connected, trainable neural networks $\sigma_{f,t}$, $\sigma_{i,t}$, $\sigma_{o,t}$

$$f_t \coloneqq \sigma_{f,t}(h_{t-1}, x_t) \tag{175}$$
$$i_t \coloneqq \sigma_{i,t}(h_{t-1}, x_t) \tag{176}$$
$$o_t \coloneqq \sigma_{o,t}(h_{t-1}, x_t). \tag{177}$$

Additionally, a candidate input vector $\tilde{c}_t$ is computed, using another fully connected neural network $\sigma_{c,t}$, with tanh as final activation function and the new memory state c_t is given via

$$c_t \coloneqq f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{where} \tag{178}$$
$$\tilde{c}_t \coloneqq \sigma_{c,t}(h_{t-1}, x_t). \tag{179}$$

By $\odot$ we denote component wise multiplication of two vectors. Using the new memory state, the new output h_t is given as

$$h_t \coloneqq o_t \odot \tanh(c_t). \tag{180}$$

It is common, to use only one FC layer ($\sigma_i(x) = \Phi_i(W_i x + b_i)$) in all sub networks. $\sigma_{f,t}$, $\sigma_{i,t}$, $\sigma_{o,t}$ use sigmoid functions as activation functions while $\sigma_{c,t}$ uses the tanh. The final output of the network is given as the concatenation of the layer outputs

$$h = (h_1, \cdots, h_K). \tag{181}$$

Additional layers can be added, by using the output $h = (h_1, \cdots, h_K)$ of the previous layer as input sequence to the next layer.

7.2 Transformer Encoders

This section is based on [Vas+23] and [Xio+20]. Transformer encoder networks are a different architecture, made for condensing the information contained in large sequential context vectors into smaller embedded vectors. The architecture was first proposed by Ashish Vaswani e.a. in 2017 [Vas+23] and marked an important step in the development of artificial intelligence and in particular *large language models*. Its core element is the *attention mechanism* that consists of multiple *attention heads*. A single attention head T with K features is a neural network that works as follows:

The input vector x is split into K smaller vectors x_i of dimension $\text{ceil}(n/K)$ (The input vector can be extended with zeros until $K|n$). Three trainable FC neural networks, $\sigma_{Q,i}$, $\sigma_{K,i}$ and $\sigma_{V,i}$, are then applied to each x_i to compute the so called *query-*, *key-* and *value* vectors

$$q_i \coloneqq \sigma_{Q,i}(x_i) \tag{182}$$
$$k_i \coloneqq \sigma_{K,i}(x_i) \tag{183}$$
$$v_i \coloneqq \sigma_{V,i}(x_i). \tag{184}$$

It is common to choose one layer NNs with the identity as activation function for the σ_j i.e. trainable matrices. Query-, key- and value vector are used to compute the *attention matrix* A, given as

$$A_{ij} \coloneqq \text{Softmax}\left(\frac{q_i \cdot k_j}{\sqrt{K}}\right), \tag{185}$$

where the factor $1/\sqrt{K}$ is inserted to prevent the evaluation of the softmax function at large arguments, where the gradients are small. A is used to compute the output vectors of the attention head y_i, as a linear combination of the value vectors given as

$$y_i \coloneqq \sum_j A_{ij}\, v_j, \tag{186}$$

and the final output is given as $y = (y_1, \dots y_K)$.

The aim is to capture the full information contained in sequential data. So far, the proposed architecture is invariant under permutations of the inputs x_i. In order for the network to be able to access the sequential nature of the input, an additional positional embedding is used. The positional embedding consists of a trainable matrix $A_P \in \mathbb{R}^{K \times n/K}$ and maps an index i to the i-th column of A_P, c_i. The full input to the attention head x_i' is given as

$$x_i' = x_i + c_i. \tag{187}$$

In a full network, different attention heads T_j can be used in parallel, where multiple independent attention matrices are computed. This is called a *multi-head attention layer*. The output of all attention heads are concatenated $s = (y_1, \dots y_h)$, assuming h individual attention heads. A trainable matrix $A_O \in \mathbb{R}^{h \dim y_j \times \dim y_j}$ is used to compute the output y as

$$y = A_O s. \tag{188}$$

It is further possible to use multiple multi-attention head layers in sequence, where the output of the previous layer is used as the input of the following layer.

8 Application to PTAs

As discussed in section 5.2 on Monte Carlo Inference, performing a full MCMC analysis of a PTA dataset requires computation time on the order of weeks. The number of SGWB models proposed in the literature is high, as can be seen in [Aga+23; Afz+23]. Additionally, given the aim to perform model comparison and therefore to compute Bayes factors (see eq. (7)) for different models, the result is highly dependent on the chosen model prior distribution. Therefore, several iterative analyses of the PTA data may have to be performed for a single model. This challenge is further intensified by the constantly expanding dataset, as new pulsars are regularly discovered and the duration of existing observations continues to increase. Consequently, our aim is to accelerate the analysis using machine learning and simulation-based inference. The idea to use SBI in PTA analyses searching for an SGWB was first proposed by David Shih et al. in [Shi+23]. As stated in the introduction, the aim of this thesis is to continue their work and the results haven been produced in collaboration with them. The goal is to perform amortized posterior estimation (see section 5.3), which requires the simulation of a training dataset. In the following we will explain in detail the simulation process that is used.

8.1 Data Simulation

8.1.1 Simulation Process

As seen in section 5.3, amortized posterior estimation requires a simulated dataset of the form

$$D = \{(\boldsymbol{\theta}, \delta\mathbf{t}_G)_i\}, \tag{189}$$

where pairs of parameter vectors $\boldsymbol{\theta}$ and projected timing residual series $\delta\mathbf{t}_G$ are produced. The simulated residual series have to match the measured times of arrival $\delta\mathbf{t}_{G,m}$ (see eq. (74)), to allow a later analysis of the measured data. The times of arrival as measured in the NANOGrav 12.5 yr dataset [Arz+20] are taken as reference. Since the aim of this work is to further explore the possibilities of amortized posterior estimation as a concept, we will not use the full 12.5 yr dataset but concentrate on a subset of pulsars. We use the subset proposed in [Shi+23] which were chosen because of their high responsiveness to an SGWB. This allows direct comparison with the results they obtained. The names of the pulsars together with the respective number of TOA-observations and the best-fit values for the red-noise model are depicted in table 1.

Pulsar	Nr. observations	Best-fit $\log_{10} A_r$	Best-fit γ_r
J1909-3744	408	-15.08	1.73
J2317+1439	447	-17.08	3.20
J2043+1711	302	-16.39	2.94
J1600-3053	236	-13.54	0.61
J1918-0642	262	-16.38	2.68
J0613-0200	278	-14.46	2.16
J1944+0907	136	-16.51	3.06
J1744-1134	268	-13.62	2.45
J1910+1256	170	-16.70	3.25
J0030+0451	463	-15.08	4.89

Table 1: Selection of pulsars used for the analysis. Additionally, the number of observations and the best fit parameters for the red-noise model are given. The table is taken from [Shi+23], which takes its values from [Arz+20].

To generate simulated projected residual series $\delta\mathbf{t}_G$ we use eq. (87) which requires the computation of the projection matrix $\mathbf{G}$ and the simulation of a noise time series $\mathbf{n}$. The full timing series

is written as the concatenation of the timing series of individual pulsars $\mathbf{n} = (\mathbf{n}_1, \ldots, \mathbf{n}_N)$, where $N = 10$ denotes the number of individual pulsars used. For each pulsar P_A, the time series consists of a vector of noise values $\mathbf{n}_A = (n_{A,i})_i$, each corresponding to the observation of one pulse.

The projection matrix $\mathbf{G}$ is computed using eq. (84) and eq. (86), using the design matrix $\mathbf{M}$ given in eq. (78) as

$$\mathbf{M} = \frac{\partial \mathbf{t}_{\mathrm{det}}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}_{\mathrm{est}}), \tag{190}$$

and therefore depends on the choice of the timing model. In this work we are going to use the same minimal timing model as in [Shi+23], which includes three parameters $\boldsymbol{\xi} = (\xi_0, \xi_1, \xi_2)$ per pulsar and has the functional form of a polynomial in the observation times

$$t_{\mathrm{det}}(\boldsymbol{\xi}) = t_{\mathrm{obs}}^2\, \xi_2 + t_{\mathrm{obs}}\, \xi_1 + \xi_0 \tag{191}$$

for every pulsar. Using the observed times of arrival for one pulsar with k observations $\mathbf{t}_{\mathrm{obs}} = (\mathbf{t}_{\mathrm{obs,1}}, \ldots \mathbf{t}_{\mathrm{obs,k}})$, the design matrix is given as

$$\mathbf{M} = \begin{pmatrix} (\mathbf{t}_{\mathrm{obs,1}})^2 & (\mathbf{t}_{\mathrm{obs,1}})^1 & 1 \\ \vdots & \vdots & \vdots \\ (\mathbf{t}_{\mathrm{obs,k}})^2 & (\mathbf{t}_{\mathrm{obs,k}})^1 & 1 \end{pmatrix}. \tag{192}$$

This choice of timing model is only a minimal reflection of the physical reality and only serves for a proof of concept. In the future, this could be replaced by the precise timing models used by the PTA collaborations. The same timing model is used for every pulsar, resulting in $3N = 30$ total timing model parameters and a block diagonal design matrix for the full timing series. This allows the computation of the projection matrix $\mathbf{G} = \mathrm{diag}(\mathbf{G_1}, \ldots, \mathbf{G_N})$ for each pulsar individually, which saves computation time. In practice, the computation of $\mathbf{G}$ requires performing a singular value decomposition of $\mathbf{M}$, which can be done efficiently using the numerical framework *NumPy linear algebra* (see [Har+20]). The resulting dimension of the projected timing residuals is given as

$$\dim = \dim \mathbf{t}_{\mathrm{obs}} - \dim \boldsymbol{\xi}, \tag{193}$$

which results, given 2970 total observations of the 10 pulsars in table 1, in a dimension of 2940 for $\delta \mathbf{t}_G$.

The next step is to simulate the noise time series $\mathbf{n}$. As mentioned in section 4.4, we assume a noise model that consists of three contributions

$$\mathbf{n} = \mathbf{n}_{GW} + \mathbf{n}_r + \mathbf{n}_w, \tag{194}$$

where all three contributions are assumed to be Gaussian. Specifically, $\mathbf{n}_{GW}$ describes the noise induced by the SGWB, $\mathbf{n}_r$ a per-pulsar red noise contribution and $\mathbf{n}_w$ a common white noise contribution. All noise contributions are assumed to be centered around $\mathbf{0}$ and are therefore fully described by their covariance functions, i.e., covariance matrices in the discrete case.

Each pulsars individual noise series is assumed to be a stationary, Gaussian random process, that can be equivalently described in Fourier space. Using this property, $\mathbf{n}(t)$ is simulated using

$\mathbf{z}(f) = (z_A(f))_A = (a_A(f), b_A(f))_A$, $f \in \mathbb{R}^+$, with $\mathbf{z}(f)$ being the vector of Fourier transforms for each individual pulsar P_A (see appendix A.4). $\mathbf{z}(f)$ is then fully described by the individual pulsars power spectral densities and the inter pulsar correlations in analogy to eq. (69)

$$\langle z_A^*(f), z_B(f') \rangle = \chi_{AB}\, \delta(f - f')\, P_A(f'). \tag{195}$$

where χ_{AB} gives the inter pulsar correlations and $P_A(f')$ power spectral density of $n_A(t)$. Since only noise contributions are considered where either $\chi_{AB} = \delta_{AB}$ or $P_A(f') = P_B(f')$, this expression is well defined.

For the numerical simulation, frequencies are discretized following

$$f_i = f_L + i\Delta f, \tag{196}$$

where $f_L = \Delta f = 0.01/12.5\,\text{yrs} \approx 0.01/T_{\text{obs}}$ and $N_f = 1000$ frequencies and T_{obs} is the observation time of the NANOGrav 12.5 yr dataset. The corresponding correlations of the discretized random vector are given as

$$\langle a_A(f_i), a_B(f_j) \rangle = \langle b_A(f_i), b_B(f_j) \rangle = \frac{\delta_{ij}}{\Delta f} P_A(f_i)\, \chi_{AB} \quad \text{and} \tag{197}$$

$$\langle a_A(f_i), b_B(f_j) \rangle = 0 \tag{198}$$

(see eq. (A30)). For further discussion of the discretization, see appendix A.5. It follows that, given residual spectral densities $P_A(f)$ and spatial correlations χ_{AB}, the covariance matrix is given as

$$\Sigma_{ijAB} = \frac{\delta_{ij}}{\Delta f} P_A(f_i)\, \chi_{AB}, \tag{199}$$

allowing to sample the random vectors $\mathbf{a} = (a_{A,i})_{A,i} \coloneqq (a_A(f_i))_{A,i}$ and $\mathbf{b} = (b_{A,i})_{A,i} \coloneqq (b_A(f_i))_{A,i}$ from a multivariate Gaussian distribution (see appendix A.2) with mean zero and covariance matrix Σ_{ijAB}, which can be done efficiently using *NumPy random sampling*. Consequently, the simulated i-th residual for pulsar P_A, $n_{A,i}$ can then be calculated using the discretized inverse Fourier transform

$$n_{A,i} = 2 \sum_{j=1}^{N_f} \Delta f (a_{A,j} \cos(2\pi f_j t_i) - b_{A,j} \sin(2\pi f_j t_i)). \tag{200}$$

In order to generate a training dataset for a given noise model, N_{data} parameters $\boldsymbol{\theta}$ are sampled from the chosen prior distribution $p_0(\boldsymbol{\theta})$. For each parameter, the three noise components $\mathbf{n}_{GW}$, $\mathbf{n}_r$ and $\mathbf{n}_w$ are drawn individually based on the noise model and the parameters for each pulsar. The white noise is sampled as described in eq. (204). To sample the red- and GW-noise the (block-)diagonal form of the covariance matrix (see eq. (199)) is used. For the red noise, a diagonal covariance matrix, $\Sigma^r_{ijAB} = \frac{\delta_{ij}}{\Delta f} P^r_A(f_i)\, \delta_{AB}$ is assumed and $a^r_{A,i}$, $b^r_{A,i}$ are sampled, using the noise model given in eq. (208), following

$$a^r_{A,i} = b^r_{A,i} = \sqrt{\frac{P^r_A(f_i)}{\Delta f}}\, \nu, \quad \nu \sim \mathcal{N}(0, 1). \tag{201}$$

This is equivalent to sampling from a multivariate Gaussian distribution with covariance matrix Σ^r_{ijAB} as shown in appendix A.1. Choosing a diagonal covariance matrix corresponds to assuming no inter-pulsar correlations.

The GW-noise covariance matrix has block diagonal form $\Sigma^{GW}_{ijAB} = \frac{\delta_{ij}}{\Delta f} P_g(f_i)\, \zeta_{AB}$ with blocks $\boldsymbol{\Sigma_i} = \frac{P_g(f_i)}{\Delta f}\, \boldsymbol{\zeta}$, where the spatial correlations ζ_{AB} are given by the overlap reduction function from eq. (66). We can therefore sample using

$$a^{GW}_{A,i} = b^{GW}_{A,i} = \sqrt{\frac{P_g(f_i)}{\Delta f}}\, c_{A,i}, \quad \mathbf{c}_i = (c_{1,i}, \ldots, c_{N,i}) \sim \mathcal{N}(0, \boldsymbol{\zeta}). \tag{202}$$

All $\mathbf{c}_i$ are independently sampled from a multivariate Gaussian distribution using the angular correlation matrix $\boldsymbol{\zeta}$ as covariance matrix. This, again, is equivalent to the sampling from a multivariate Gaussian distribution with covariance matrix Σ^{GW}_{ijAB}, but it is computationally less expensive, since N-dimensional vectors $\mathbf{c}_i$ are sampled instead of an NN_f-dimensional vector.

Each noise contribution is then computed using eq. (200) and the final projected residual series is computed by summing the individual noise contributions and projecting with the projection matrix $\mathbf{G}$

$$\delta\mathbf{t}_G = \mathbf{G}^T (\mathbf{n}_w + \mathbf{n}_r + \mathbf{n}_{GW}). \tag{203}$$

In the next section we will discuss the noise models that were used in this work.

8.1.2 Noise Models

The white noise contribution $\mathbf{n}_w$ is assumed to be fully uncorrelated Gaussian noise with constant standard deviation σ_w. Therefore, $n_{A,i}$ can directly be drawn from a univariate Gaussian distribution,

$$n_{A,i} \sim \mathcal{N}(0, \sigma_w), \tag{204}$$

where the choice of $\sigma_w = 100\,\mathrm{ns}$ following [Shi+23] was made. For the red noise contribution we assume temporal correlations for $\mathbf{n}_A$ but no inter pulsar correlations. The Fourier modes are sampled, as described in the previous chapter, from a multivariate Gaussian distribution. The assumption of no correlation between pulsars implies $\chi^r_{AB} = \delta_{AB}$.

The power spectral density P^r_A of the red noise contribution is assumed to follow a *constant power law* (CPL), which corresponds to a first order log-polynomial ansatz. A log-polynomial ansatz makes the assumption of a polynomial in $\log_{10}(f/f_{\mathrm{ref}})$ for the decadic logarithm of the *characteristic strain amplitude* $h_c(f)$ (analogous to eq. (70)),

$$\begin{aligned} \log_{10}(h_c)(x) &= \mathrm{poly}(x), \quad \text{with} \\ x &:= \log_{10}\left(\frac{f}{f_{\mathrm{ref}}}\right). \end{aligned} \tag{205}$$

The reference frequency f_{ref} can be chosen freely. Common choices are $f_{\mathrm{ref}} = 1/T_{\mathrm{obs}}$ and $f_{\mathrm{ref}} = 1/1\,\mathrm{yr}$. In this work, the latter choice was made for ease of numerical implementation. Polynomial models are relatively simple but effective descriptions. In particular, the case of a first order log-polynomial is relevant since it corresponds to the predicted strain amplitude for

a population of inspiraling black hole binaries, as was shown by [Phi01]. The CPL model is given as

$$\text{poly(x)} = \alpha x + \log A, \tag{206}$$

where $\log_{10} A$ and α are free parameters of the model. It is common to replace the parameter α with the parameter $\gamma \coloneqq 3 - 2\alpha$. The characteristic strain amplitude is obtained as

$$h_c(f) = A \left(\frac{f}{f_{\text{ref}}}\right)^{\frac{3-\gamma}{2}}, \tag{207}$$

which, using eq. (70) and eq. (68) results in a residual power spectral density of

$$P(f) \overset{\text{CPL}}{=} \frac{A^2}{12\pi^2 f_{\text{ref}}^3} \left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma}. \tag{208}$$

In modeling the red noise contribution, independent CPL spectra for each pulsar P_A,

$$P_A^r(f) = \frac{(A_A^r)^2}{12\pi^2 f_{\text{ref}}^3} \left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma_A^r}, \tag{209}$$

are assumed, resulting in $20 = 2N$ independent parameters, $\log_{10} A_1^r, \dots \log_{10} A_N^r$ and $\gamma_1^r, \dots \gamma_N^r$, for the red noise.

The GW contribution to the noise is assumed to follow inter pulsar correlations, given by the overlap reduction function in eq. (66)

$$\chi_{AB} = \zeta_{AB} = (1 + \delta_{AB}) \left[\frac{3}{2} x_{AB} \log x_{AB} - \frac{1}{4} x_{AB} + \frac{1}{2}\right] \quad \text{with} \tag{210}$$

$$x_{AB} = \frac{1}{2}(1 - \cos\theta_{AB}). \tag{211}$$

The angular separation of two pulsars θ_{AB} is taken from the NANOGrav 12.5 yr dataset, where the ecliptic longitude λ_A and the ecliptic latitude β_A are stored for each pulsar. The cosine of the angular separation of two pulsars P_A and P_B can be computed, following [Bal13], as

$$\cos(\theta_{AB}) = 1 - 2\left[\sin^2\left(\frac{|\beta_A - \beta_B|}{2}\right) + \cos(\beta_A)\cos(\beta_B)\sin^2\left(\frac{|\lambda_A - \lambda_B|}{2}\right)\right]. \tag{212}$$

The resulting angular correlation matrix is denoted as $\boldsymbol{\zeta} = (\zeta_{AB})_{AB}$.

In this work, different choices for the residual power spectral density $P_g(f)$ are explored. Different log-polynomial ansätze are used, where the first order corresponds to an identical CPL spectrum as in the case of the red noise

$$P_g(f) \overset{\text{CPL}}{=} \frac{A_{GW}^2}{12\pi^2 f_{\text{ref}}^3} \left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma_{GW}}. \tag{213}$$

As discussed before, this fully defines the frequency domain covariance matrix for a CPL model. In this specific case, it has been possible to compute an analytical expression for the time domain covariance matrix C_{ijab} from eq. (90), as was shown in [Haa+09]. It is given as

$$C_{ijab} = \zeta_{ab} \frac{A^2 f_{ref}^{\gamma+2}}{(2\pi)^2 f_L^{\gamma+3}} \Bigg\{ \Gamma(-3-\gamma) \sin\left(\frac{-\pi(\gamma+2)}{2}\right) [f_L \tau_{ijab}]^{\gamma+3} \underbrace{- \sum_{k=0}^{\infty} (-1)^k \frac{[f_L \tau_{ijab}]^{2k}}{(2k)!(2k-3-\gamma)}}_{\text{Cutoff dependent term}} \Bigg\}, \tag{214}$$

where $\tau_{ijab} = 2\pi(t_{ai} - t_{bj})$ and a cutoff f_L has been introduced. Since the simulation also requires a low frequency cutoff for numerical computability, the same frequency will be used in both cases. The time domain covariance matrix for the red noise contribution can be obtained by replacing $\zeta_{ab} \to \delta_{ab}$ and using the respective red noise parameters. It was shown in [Haa+09] that the constant and quadratic contributions to the sum in the cutoff dependent term in eq. (214) are absorbed by the constant and spin-down corrections in the timing model. Even though we do not use a physical timing model that corrects for spin-down, the polynomial timing model used here (see eq. (191)) also accounts for a quadratic term in time that is indistinguishable from the spin-down correction term, used in physical timing models (see [Haa+09]). Furthermore, higher orders in $f_L \tau_{ijab}$ in the cutoff dependent term become very small if $f_L \tau_{ijab} \ll 1$. Since we chose $f_L = 0.01/12.5\,\text{yrs}$, which is comparatively small to the observation time of 12.5 yrs, higher order terms will also be omitted. One should note that the dependency on f_L in the first part of eq. (214) cancels out. Therefore, we follow Shih et al. in omitting the cutoff dependent term from eq. (214) (see Appendix of [Shi+23]). The full time domain covariance matrix can be computed using the projection matrices following eq. (90).

In addition to the CPL model, further higher order polynomials in eq. (205) are explored

$$\text{poly}(x) = -\frac{c_3}{12}x^3 - \frac{c_2}{4}x^2 + \frac{3-c_1}{2}x + \log c_0. \tag{215}$$

In this notation c_1 corresponds to γ from the CPL model and c_0 corresponds to A. The model where c_3 is set to zero is called the *running power law model* (RPL) and is extensively discussed in [Aga+25]. Moreover, we experiment with a polynomial of third order which we will call the *poly3 model*. The resulting residual power spectral densities are given as

$$P_g(f) \overset{\text{RPL}}{=} \frac{c_0^2}{12\pi^2 f_{\text{ref}}^3} \left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma_{\text{RPL}}} \quad \text{with} \tag{216}$$

$$\gamma_{\text{RPL}} \coloneqq \frac{c_2}{2} \log\left(\frac{f}{f_{\text{ref}}}\right) + c_1 \tag{217}$$

and

$$P_g(f) \overset{\text{poly3}}{=} \frac{c_0^2}{12\pi^2 f_{\text{ref}}^3} \left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma_{\text{p3}}} \quad \text{with} \tag{218}$$

$$\gamma_{\text{p3}} \coloneqq \frac{c_3}{6} \log^2\left(\frac{f}{f_{\text{ref}}}\right) + \frac{c_2}{2} \log\left(\frac{f}{f_{\text{ref}}}\right) + c_1. \tag{219}$$

Lastly we also tested a noise model derived from cosmological phase transitions. There are predictions for gravitational waves from collisions of vacuum bubbles and from sound waves generated in a primordial plasma. For a comprehensive overview of gravitational waves from cosmological phase transitions see [Cap+20]. Since this work aims at a proof of concept rather

than an exhaustive exploration of a particular SGWB model, we choose to focus on gravitational waves from bubble collisions. The gravitational wave energy density spectrum (see eq. (71)) for gravitational waves from bubble collisions is given as a *broken power law* (BPL) in [Afz+23] as

$$\Omega_{GW}(f) \overset{\text{BPL}}{=} \mathcal{D}\,\tilde{\Omega}_b \left(\frac{\alpha_*}{1+\alpha_*}\right)^2 (H_* R_*)^2 \mathcal{S}(f/f_b), \tag{220}$$

where

$$\mathcal{D} = \frac{\pi^2}{90}\frac{T_0^4}{M_{pl}^2 H_0^2} g_* \left(\frac{g_{*,s}^{\text{eq}}}{g_{*,s}}\right)^{4/3} \tag{221}$$

$$f_b \simeq 48.4\,\text{nHz}\, g_*^{1/2} \left(\frac{g_{*,s}^{\text{eq}}}{g_{*,s}}\right)^{1/3} \left(\frac{T_*}{1\,\text{GeV}}\right) \frac{f_b^* R_*}{H_* R_*} \tag{222}$$

$$\mathcal{S}(x) = \frac{1}{\mathcal{N}} \frac{(a+b)^c}{(bx^{-a/c} + ax^{b/c})^c} \tag{223}$$

$$\mathcal{N} = \left(\frac{b}{a}\right)^{a/n} \left(\frac{nc}{n}\right)^c \frac{\Gamma(a/n)\Gamma(b/n)}{n\Gamma(c)} \tag{224}$$

$$n = (a+b)/c. \tag{225}$$

Here, T_0 denotes the photon temperature and H_0 the Hubble rate today. The degrees of freedom g_* and $g_{*,s}$ are evaluated at $T = T_*$ based on [SS20] using the implementation in the PTA inference framework *PTArcade* [Mit+23]. $g_{*,s}^{\text{eq}}$ denotes the degrees of freedom of the radiation entropy at the time of matter-radiation equality and is computed the same way. $\tilde{\Omega}_b = 0.0049$ and $f_b^* = 0.58/R_*$ are inferred from simulations (see [JT17]). Lastly $\log_{10}\alpha_*$, $\log_{10} T_*$, $\log_{10} H_* R_*$, a, b and c are going to be treated as free parameters of the model. T_* denotes the percolation temperature, that is the temperature at which $\sim 34\%$ of the universes volume has been converted to the true vacuum and mathematically T_* controls the peak frequency f_b of the spectrum. α_* is defined as the ratio between the vacuum energy and the total energy stored in radiation and it characterizes the strength of the phase transition. $H_* R_*$ denotes the average bubble separation in units of the Hubble radius. Lastly a, b and c are the shape parameters of the spectrum, where a and b give the slope of the spectrum in the limit of low and high frequencies respectively and c controls the width of the peak.

This summarizes all noise models that are used in this work. For a given noise model, the full parameter vector is given as the concatenation of the SGWB- and red noise parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{GW}, \boldsymbol{\theta}_r)$.

8.2 Preprocessing

So far, the process of simulating a training dataset for training the neural network was described. After simulation, the training data $(\delta \mathbf{t}_G, \boldsymbol{\theta})$ is not directly passed to the neural network, but is preprocessed before. The preprocessing of the parameter vectors $\boldsymbol{\theta}$ is directly motivated by the choice of prior distribution. For every parameter θ_i, a uniform (or log-uniform) distribution on an interval $[\theta_{i,\min}, \theta_{i,\max}]$ is assumed. The preprocessed parameter θ_i' are given as

$$\theta_i' \coloneqq \frac{\theta_i - \theta_{i,\min}}{\theta_{i,\max} - \theta_{i,\min}}. \tag{226}$$

This ensures that θ_i' assumes values in $[0, 1]$ which are easier for a neural network to handle when computing the gradients of the loss function (see section 5.1). The parameters can be recovered by applying the inverse transformation

$$\theta_i = (\theta_{i,\max} - \theta_{i,\min})\,\theta'_i + \theta_{i,\min}. \tag{227}$$

Preprocessing the projected timing residuals $\delta\mathbf{t}_G$ is more complicated since their numerical values span a wide range from $\mathcal{O}(10^{-9})$ to $\mathcal{O}(10^{-2})$, depending on the choice of model and prior distribution. We use several different preprocessing schemes:

- The preprocessing scheme proposed by Shih et al. is given as

$$\delta t'_{G,i} = \begin{cases} 10^7\,\delta t_{G,i} & \text{if } |\delta t_{G,i}| < 10^{-4} \\ 0 & \text{else,} \end{cases} \tag{228}$$

  where clipping is used to handle large outliers. The value of 10^7 was chosen based on the inverse order of magnitude of the median value of the dataset generated by them. This allows the network to handle the wide numerical range of the data. The disadvantage is that the preprocessing is not invertible and might lose relevant information from the tails due to the clipping. It will be referred to as the *original preprocessing* method.

- The first alternative is given as follows: We compute the inter quartile range Δ_{IQR} of the data (see eq. (A11)). We then rescale the data and pass it through an arctan and arctanh which gives the preprocessed data as

$$\delta t'_{G,i} = \operatorname{arctanh}\left(\frac{\arctan\left(\frac{2\delta t_{G,i}}{\Delta_{IQR}}\right)}{2\pi}\right). \tag{229}$$

  Rescaling with the IQR makes sure that exactly half of the data lies in $[-1, 1]$. The arctan then serves to squash the data to the interval $[-2\pi, 2\pi]$ and the arctanh smoothens out the tails. This transformation has the advantage that it is invertible and we found it to work well. In the following we will refer to it as the *arc preprocessing* method.

- Lastly we also used a *quantile preprocessing* method using Scikit-learn's [Ped+11] QuantileTransformer (see [Sci24]). The idea here is to fit an invertible nonlinear transformation, such that the transformed data is approximately uniformly distributed. The transformation operates on a quantile basis, i.e., the data is divided into k quantiles and the the i-th quantile is mapped to the interval $[(i-1)/k, i/k]$. We found that using 10 quantiles works well for us. This method produces a highly uniform distribution and the preprocessed data is guaranteed to lie in the $[0, 1]$ interval independent of the raw data.

We worked with all of the above methods to varying degrees. One should note that the choice of preprocessing is no exact science and it involves trial and error testing.

8.3 NN Architectures

We perform experiments using different neural network architectures. The base framework will always be a generative NN (see section 6), supported by an encoder NN (see section 7) and all networks are implemented in the *python* library *PyTorch* [Pas+19]. The optimizer used throughout this work is Adam (see section 5.1) and we always use cosine annealing (see eq. (102)) as learning rate scheduler.

Original NF architecture

In the original work by Shih et al., they used a CPL as noise model (see eq. (213)) for the SGWB, based on which they generated a training dataset of 10^6 samples $(\delta \mathbf{t}_G, \boldsymbol{\theta})$. As generative network they used a conditional normalizing flow. The NF uses 8 autoregressive layers (see section 6.1.3) with sub-networks comprised of two hidden layers with 200 nodes each and ReLU activations (see eq. (94)). The generator network uses RQS transformations (see section 6.1.4) with 8 bins and fixed tail bounds of $A = -1$ and $B = 1$. The latent distribution is chosen to be uniform on $[-1, 1]$.

LSTM architecture

The LSTM encoder network (see section 7.1) used by Shih et al. is comprised of multiple LSTM networks, where each is passed the noise series of one pulsar. Each sub-LSTM network uses one time step per component in the projected pulsars noise series. The dimension of the hidden state for all networks is set to 100 and two layers are used per LSTM. The final hidden- and memory states of both layers are then taken and passed through a FC network comprised of 400 input nodes, 200 and 100 hidden nodes and 50 output nodes. For every layer ReLu activation functions are used. The outputs of all sub-networks are then concatenated and passed through another FC network comprised of $50N_{Pulsar} = 500$ input nodes, 100 and 100 hidden nodes and $2N_{Pulsar} + 2 = 22$ output nodes. Again ReLu activation functions are used.

Shih et al. did provide their code basis to us, which allowed us to use it for generating training data and to train our own network to reproduce their results. We do not modify the hyperparameters of their network architecture which is why the exact parameters are specified.

Modified NF architecture

In this work we have modified the network architecture for generative and encoder network. We use a modified version of the normalizing flow based on coupling blocks (see section 6.1.2). The number of coupling blocks is given as $n_b = \text{ceil}(2 \log_2(n_{par}))$, where n_{par} denotes the number of parameters for the noise model. This choice is motivated by [GIK20], where they prove that it is the minimum amount of coupling block layers required to model all possible correlations. In between coupling blocks, we use permutations as proposed in [GIK20] to ensure all correlations are captured. The sub-networks are again FC networks, with a variable number of hidden layers and hidden states of variable dimension. In this case we use LeakyReLu functions (see eq. (95)) with $\alpha = 10^{-2}$ for the activations. We also use RQS transformations with variable number of splines and fixed tail bounds given as $A = 0$ and $B = 1$ and the latent distribution is chosen to be uniform on $[0, 1]$.

Diffusion Network Architecture

For the diffusion network, a FC network with variable amount of layers and hidden states is used (see section 6.2). ReLu functions are used as activation functions. For the sampling process, an ordinary differential equation needs to be solved. This is done via the PyTorch implementation *torchdiffeq* [Che21], using the *Dormand-Prince method* (see [DP80]) which is a fifth order Runge-Kutta method.

Transformer Encoder Architecture

For the transformer encoder network (see section 7.2)the full noise series is split into a variable number of chunks and each chunk is passed through a trainable matrix with variable output dimension. Additionally, the index of each chunk is mapped to an equally sized vector. The sums of both vectors are taken and concatenated over all chunks. The result is then passed to the transformer encoder with a variable number of attention layers, where each layer is comprised of a variable number of attention heads.

9 Results

9.1 Reproduced Results

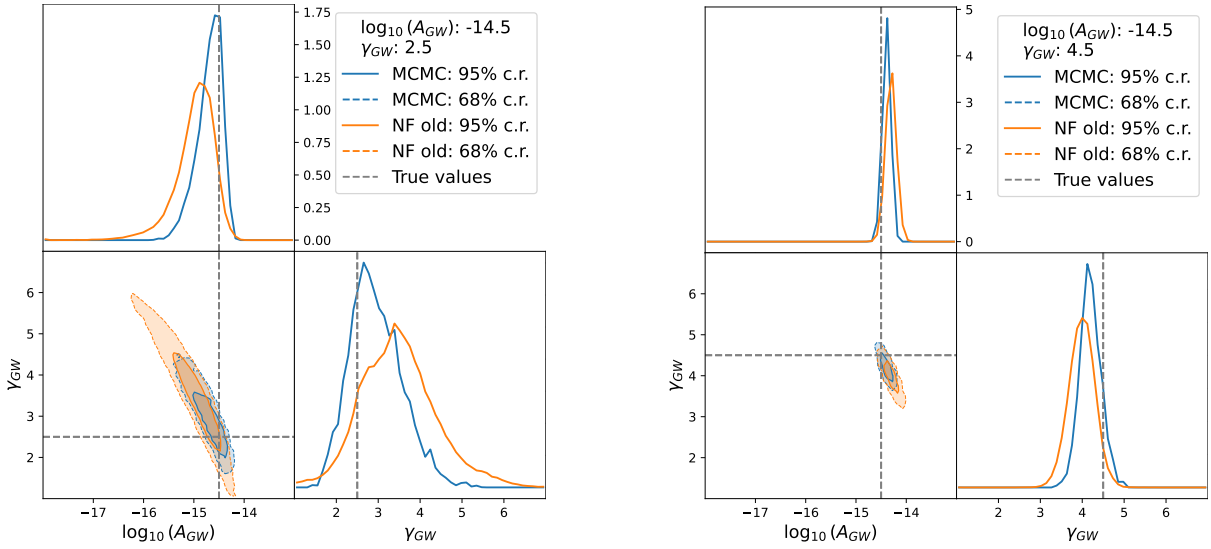
The first step was to reproduce the results obtained by Shih et al. [Shi+23]. Their aim was to generate a training dataset for amortized posterior estimation (see section 5.3), where they use a CPL as SGWB noise model (see eq. (213)) and created 10^6 pairs of noise parameters and noise series. The priors used for the CPL SGBW parameters are uniform on the following intervals: $\log_{10}(A_{GW})$: $[-18, -13]$, γ_{GW}: $[1, 7]$. The prior distributions for all red noise parameters are also chosen as uniform, on the intervals: $\log_{10}(A_a^r)$: $[-19, -13]$ and γ_a^r: $[1, 7]$. As mentioned before, they provided us with the code they used for generating the training data and for training the neural network. Additionally, they provided us with their training dataset and the sample series they used for performance evaluation. To generate the sample series, they used the simulation pipeline to create additional noise series with selected values of the noise parameters, allowing to test how well the trained network manages to reconstruct the injected noise parameters from the noise series. They used four grid-spaced parameter points to assess the network performance in different regions of parameter space. Additionally, the nominal best-fit parameter values from the NANOGrav 12.5 yr analysis were used to generate four testing series. The parameter values used for testing are listed in table 2. To get a comparison for the network generated posterior distributions, they conducted MCMC posterior analyses (section 5.2) on each testing noise series. This was done using the analysis pipeline *Enterprise* [Joh+24] developed by the NANOGrav collaboration. They also provided the resulting MCMC samples to us.

We used their training data and code to retrain a NN with equal architecture and hyperparameters to them i.e. an autoregressive normalizing flow 8.3 with an LSTM embedding network 8.3. The preprocessing follows the preprocessing scheme proposed by Shih et al. in eq. (228). The network is then used to sample a posterior with 10^5 samples, given the contexts of the provided testing noise series. The reproduced posterior distributions and their corresponding MCMC posteriors for the grid spaced noise series are depicted in fig. 4 while the results for the NANOGrav 12.5 yr best valued noise series are given in fig. 5.

Series Nr.	$\log_{10} A_{GW}$	γ_{GW}
1	-14.5	2.5
2	-14.5	4.5
3	-15.5	2.5
4	-15.5	4.5
5	-15	13/3
6	-15	13/3
7	-15	13/3
8	-15	13/3

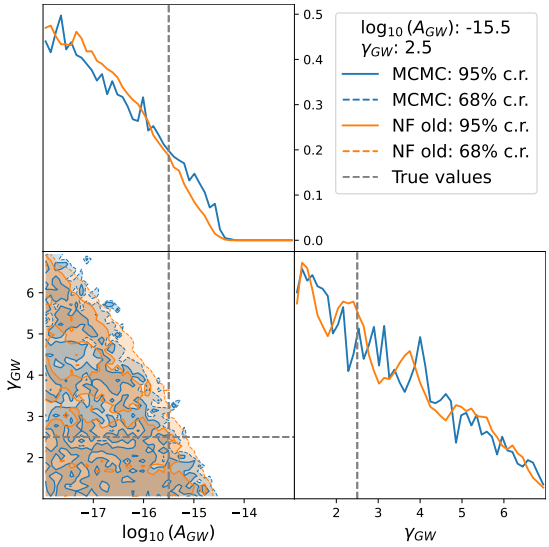
Table 2: The CPL noise parameter values used to create the testing noise series. The first four values are spaced in a grid to explore different regions of the parameter space while the last four values are fixed to the nominal NANOGrav 12.5 yr best-fit values from [Arz+20].

By visually comparing the MCMC posteriors to the Network posteriors and the "true" values, that is, the values used to generate the testing noise series, one notes, that both MCMC and the original network manage to correctly identify the region in which the true parameters lie. In tables A1 to A8 the statical estimators mean, median, maximum posterior, variance and upper and lower bounds of the 68%- and 95%-credible regions are given for each noise series. The quantities are computed based on the marginalized one dimensional distribution for each pa-

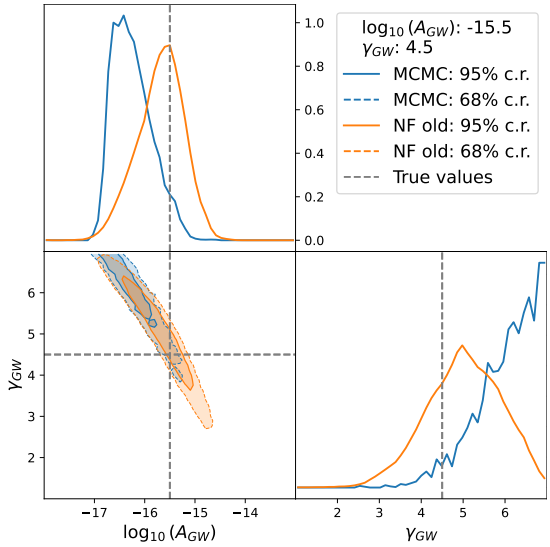


(a) Noise series 1

(b) Noise series 2



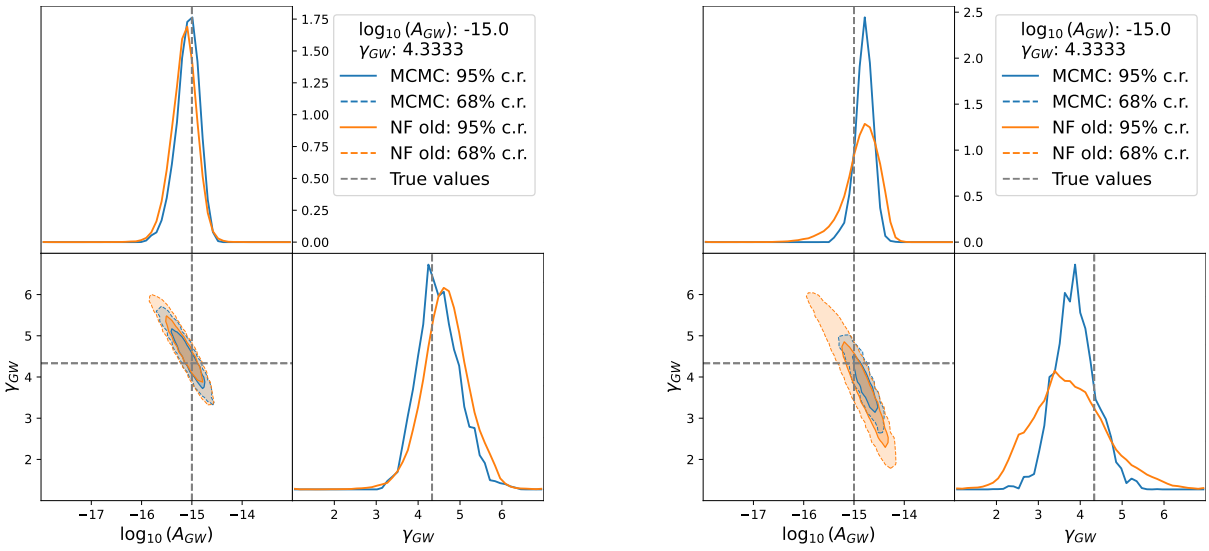
(c) Noise series 3



(d) Noise series 4

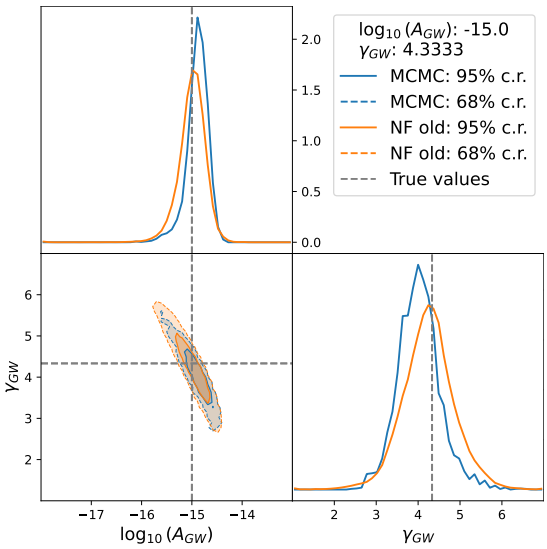
Figure 4: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) with original network architecture (reproduced) for the grid spaced noise series $1-4$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

rameter and the maximum posterior value is computed using gaussian kernel density estimation implemented in *scipy Statistical Functions* [Vir+20] using the default *Scott's rule* for bandwidth selection (see [Sco15] chapter 6). Generally, the point estimators based on the MCMC match the true values more closely and the MCMC based credible regions tend to be more strongly constrained around the true values. The main outliers are the noise series three in fig. 4c and four in fig. 4d. In the case of the third series, the simplest explanation is that the true noise parameters $\log_{10}(A_{GW}) = -15.5$ and $\gamma_{GW} = 2.5$ are overall the smallest. Equation (213) directly implies that the strength of the SGBW signal increases with larger values of $\log_{10}(A_{GW})$, independent of the frequency bin. We further note that the frequency range used to generate the training data is 0.0008/yr to 0.8/yr. With the chosen reference frequency of $f_{\text{ref}} = 1/\text{yr}$ this also leads to an increase in signal strength with larger γ_{GW} for every frequency bin. The

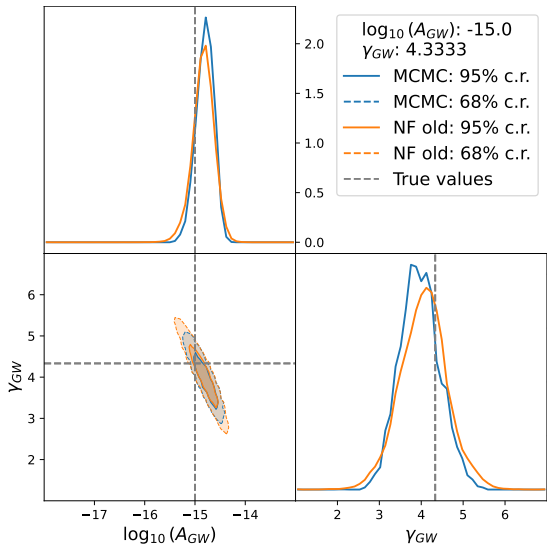


(a) Noise series 5

(b) Noise series 6



(c) Noise series 7



(d) Noise series 8

Figure 5: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) with original network architecture (reproduced) for the nominal best valued noise series $5-8$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

most probable explanation is that the relatively small SGWB signal induced in series three is too small compared to red- and white noise for both the network and the MCMC to identify it. In the case of the fourth testing series, it is more difficult to find an explanation. We observe in fig. 4d that the NN seems to more closely identify the true noise parameters. By comparing the statistical estimators from table A4 between MCMC and NN, we also note that the point estimates based on the the MCMC analysis are generally further away from the true values than those based on the NN. One possible explanation could just be that the testing noise series are randomly sampled using eq. (201). Possibly, noise series four is an improbable outlier that was accidentally correctly identified by the NN but not the MCMC. This observation highlights an important aspect about the testing process we employ here. Since we are only able to cross check the performance of the NN with the MCMC posterior based on a few testing series with

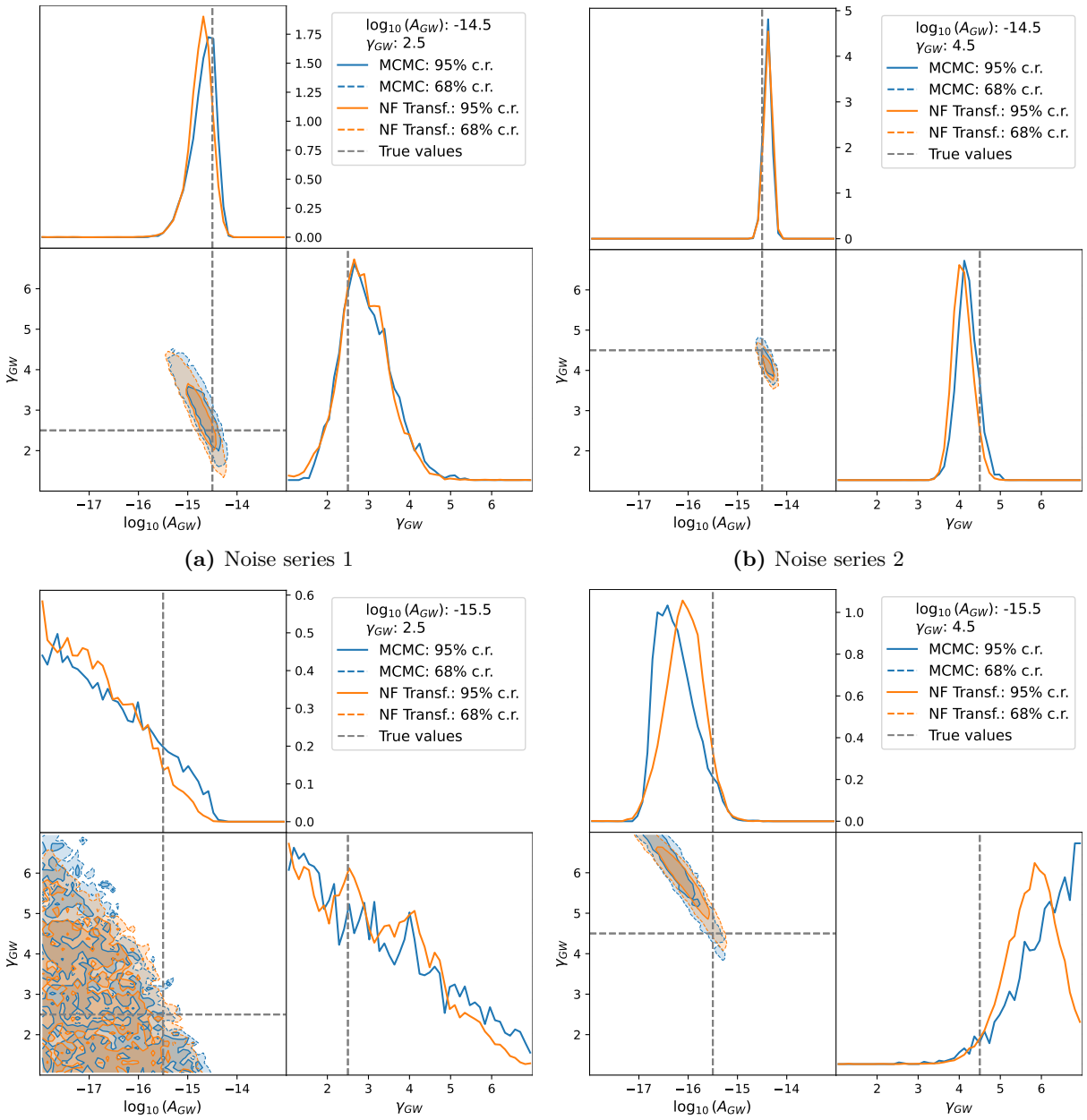
limited statical significance, we can not make strong statements about the general performance of the network.

9.2 Introducing Transformers

In the next step we used a coupling block based normalizing flow as generative network and replace the LSTM encoder with a transformer encoder network. The coupling block NF architecture described in section 8.3 and the transformer embedding network from section 8.3 require to further specify exact values for multiple hyperparameters. As training data, we used the same data provided by Shih et al. to allow comparison of the respective performances. The new network architecture required testing on which configuration for the hyperparameters work well. This was done by fixing all hyperparameters except for one, for which multiple values were tested. We tested 2-4 different configurations per parameter and arrived at the following best performing hyperparameter values for the generator: 3 hidden layers for the sub-networks of dimension 128 and 10 splines for the RQS transformations. For the transformer embedding network we use: 30 chunks in which context noise series is split of dimension 100, 256-dimensional hidden layers for the encoder and 4 attention layers with 8 attention heads per attention layer. Additionally, the batch size for the optimizer was set to 1024 and training was performed over 100 epochs, while we used the arc preprocessing routine from eq. (229). Alternative methods for hyperparameter tuning would have been considered such as a grid search. For a total of 9 parameters (where we count the number and dimension of the chunks as one parameter since they are degenerate) and 3 values per parameter, a grid search would take over 9000 hours, based on the time it took to train the best configuration network which took 4 hours and 50 minutes. This is not feasible, even though some parallelization would haven been possible depending on the availability of computing capacities on the university HPC-system *PALMA*, which we used for training. Another option would be to randomly test hyperparameter values within a certain range and to select the best performing constellation.

Once trained, we again evaluated the network performance based on the posterior distributions, obtained by sampling 10^5 parameter values off the network conditioned upon the testing noise series from table 2. The resulting posterior plots, comparing the posteriors obtained via the NN to the MCMC posteriors can be seen in figs. 6 and 7. The initial visual comparison suggests a better agreement between network posterior and MCMC posterior than in the case of the original network architecture. To further quantify the comparative performance of both network architectures we computed the respective Hellinger distances (see eq. (A12)) to the MCMC posteriors for every testing series. This was done based on a histogram of the marginalized two dimensional distribution for $\log_{10} A_{GW}$ and γ_{GW} on an equidistant 50×50 grid. The resulting distances are listed in table 3. By comparing the HDs for the original and transformer based network architecture, we see that the transformer network manages to outperform the original network in six out of eight cases, where the improvement is greater than 30% in three cases. For series three and eight, the original network performs better, with margins smaller than 10% for series three. For series eight the original network does perform significantly better ($\approx 24\%$). Recalling that series eight seemed to be an outlier decreases the significance of this finding though. Overall it can be said that the introduction of the transformer embedding network has managed to significantly improve the performance of the parameter inference since it performs better in the definite majority of cases.

As mentioned before, since the performance comparison was made based on eight points in a large parameter space its statistical significance is relatively small on a global scale. For this reason we explore an alternative way of evaluating the network performance on a more global level using weights. For this, we sampled 1000 random noise series from the training dataset and



(a) Noise series 1

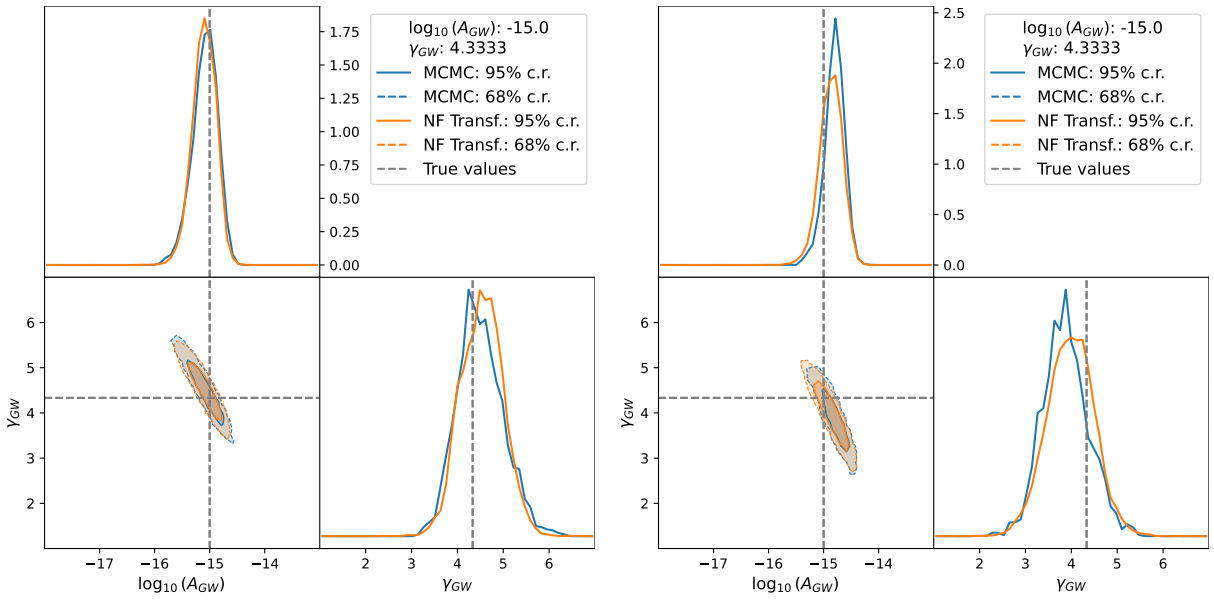
(b) Noise series 2

(c) Noise series 3

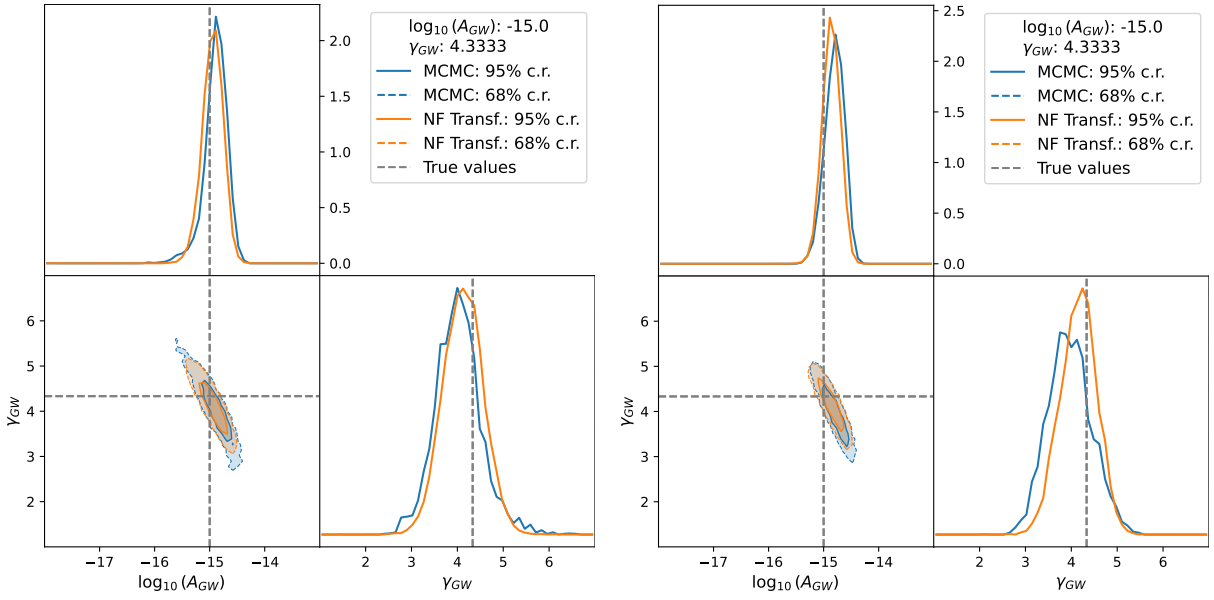
(d) Noise series 4

Figure 6: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) with coupling block NF and transformer embedding for the grid spaced noise series $1-4$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

sampled 100 parameters for each using the trained network. For every sampled parameter we then evaluate the true likelihood $p(\delta\mathbf{t_G}|\boldsymbol{\theta})$ using eq. (90) and eq. (214). We additionally evaluate the trained network at the samples in the normalization direction, thereby obtaining the network estimate of the posterior densities $p_\lambda(\boldsymbol{\theta}|\delta\mathbf{t}_G)$ at these points. Using eq. (122) and eq. (117) we then compute the standardized weights at each point by normalizing the 100 computed weights for every noise series. Histograms of the normalized weights and their logarithms are shown in figs. 8a and 8b, where the weights for all noise series are plotted. From the definition of the standardized weights, we expect to find a narrow distribution around one in the case of a



(a) Noise series 5 **(b)** Noise series 6



(c) Noise series 7 **(d)** Noise series 8

Figure 7: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) with coupling block NF and transformer embedding for the nominal best valued noise series $5-8$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

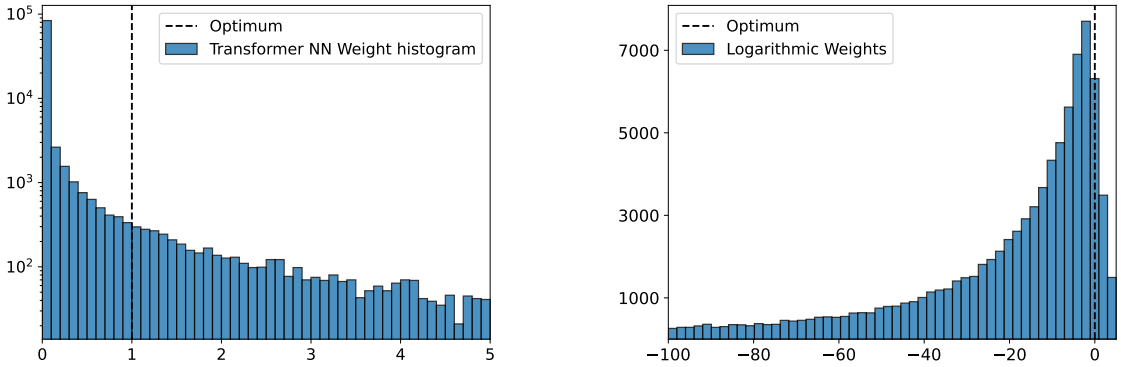
good posterior approximation by the network. figs. 8a and 8b clearly show that the weights do not cluster around one but instead have a strong bias towards very small values. Based on the definition of the weights in eq. (117),

$$\tilde{w}(\delta\mathbf{t}_G|\boldsymbol{\theta}) := \frac{p(\delta\mathbf{t}_G|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p_\lambda(\boldsymbol{\theta}|\delta\mathbf{t}_G)} \tag{230}$$

this corresponds to the trained network overestimating the true posterior distribution, which has to be compensated by the small weights. This has the practical implication, that without

Network	Srs. 1	Srs. 2	Srs. 3	Srs. 4	Srs. 5	Srs. 6	Srs. 7	Srs. 8
Original	0.40	0.35	0.22	0.50	0.18	0.42	0.23	0.16
Transformer NF	0.34	0.23	0.24	0.32	0.14	0.21	0.19	0.21
Transformer CFM	0.46	0.32	0.25	0.32	0.18	0.39	0.23	0.32

Table 3: Hellinger distances of the network produced posterior distributions to the MCMC distribution for every testing noise series. The first row shows the relative distance for the original NF network architecture, the second for the modified transformer based NF architecture and and the third row for the transformer based CFM diffusion model architecture.



(a) Standardized weights

(b) Log-weights

Figure 8: Histograms of the standardized weights (left) and their decadic logarithms (right), computed based on 1000 random noise series from the training dataset, for each of which 100 parameter samples have been drawn from the distribution learned by the transformer NF. The histograms show the weights for all noise series.

performing reweighting, the network will estimate more conservative credible regions. Used as a candidate distribution for importance sampling, small weights are generally preferable to large weights, since in the latter case, interesting regions of the parameter space can stay underexplored. This can lead to either no samples being drawn, which the reweighting can not compensate, or very few samples drawn, where the large weights can cause sharp edges on the distribution. One should also note that the loss function we used for training the neural network is the Kullback-Leibler divergence

$$\mathcal{L}_{KL}(\lambda) = \int d\delta\mathbf{t}_G \, d\boldsymbol{\theta} \, p_t(\delta\mathbf{t}_G|\boldsymbol{\theta}) \, \log \frac{p_t(\delta\mathbf{t}_G|\boldsymbol{\theta})}{p_\lambda(\delta\mathbf{t}_G|\boldsymbol{\theta})}. \tag{231}$$

From the functional form of the loss function, it follows that the training can have a tendency to reward overestimation of the true distribution. This effect is mitigated by the requirement that the NF is invertible. Generally, it can be said that the network does not manage to accurately learn the full posterior distribution. One should note at this point that the posterior distribution is 22-dimensional. Contrary to the full distribution, the posterior plots from figs. 6 and 7 and the Hellinger distances in table 3 indicate that the network does a good job at learning the marginalized two dimensional distributions. This means that the weight histograms in figs. 8a and 8b are only of limited use in evaluating the networks performance at learning the marginalized distributions.

We can furthermore use the posterior samples drawn for the posterior plots in figs. 6 and 7 to compute the weights at these points. This allows to produce a reweighted posterior distribution

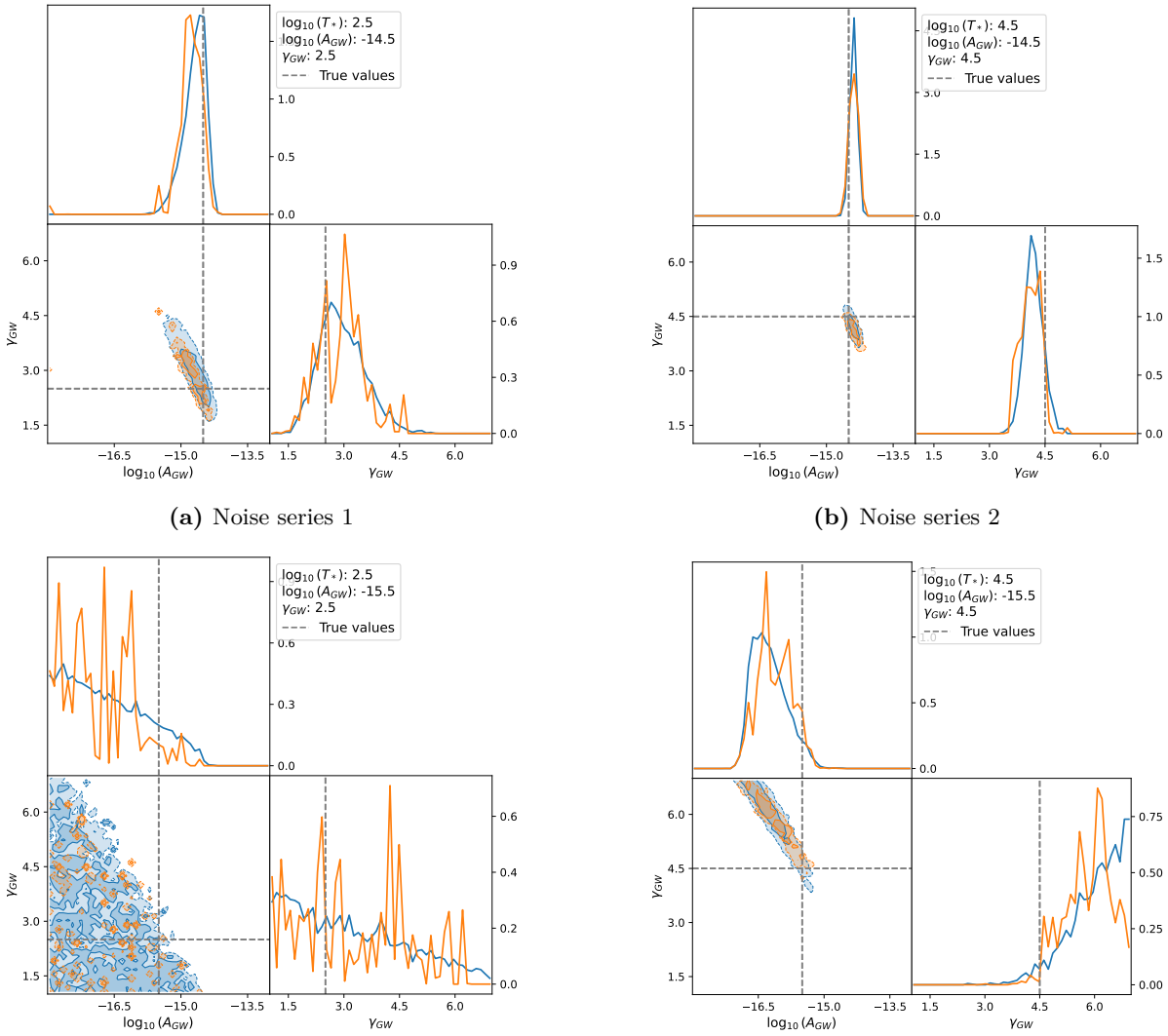
for each testing series, shown in figs. 9 and 10. Even though we discarded the 0.1% biggest outliers, the reweighted posterior distributions tend to be dominated by spikes induced by high weights. Additionally, when computing the effective sample sizes for each testing series using eq. (125), we obtain the values given in table 4.

Network	Srs. 1	Srs. 2	Srs. 3	Srs. 4	Srs. 5	Srs. 6	Srs. 7	Srs. 8
Weighting eff.	0.0033	0.0045	0.0019	0.0124	0.0129	0.0238	0.0042	0.0026
Eff. sample size	48.9	66.5	27.9	184.8	191.8	354.9	62.7	39.4

Table 4: Weighting efficiencies (see eq. (124)) and effective sample sizes (see eq. (125)), using 14900 samples for each testing noise series, computed off a reweighted sampling of the transformer based NF.

The weighting efficiencies are of order 10^{-3} which is of the same order of magnitude to the efficiencies obtained by Shih et al. in [Shi+23]. The low numbers of effective samples also explains the spiked posteriors in figs. 9 and 10. The reweighting could be repeated on a larger sample size of 10^6 to obtain a smooth posterior distribution. Since reweighting is computationally expensive as it requires to evaluate the likelihood, we chose to focus on multiple noise series, with relatively small sample sizes of 14900 instead of focusing on one noise series. For reference, we computed the effective sample size for the MCMC chain produced in the NANOGrav 12.5yr analysis [Arz+20] that are publicly available. With 919500 samples remaining after dropping the first 30000 samples from the burn-in phase, the effective sample size is 18180.2 which gives a sampling efficiency of 0.0191. The effective sample size for the MCMC chain is computed, using the python module *multiESS* based on the method developed in [VFJ19]. One should stress here, that this number is obtained using the chains of the full NANOGrav 12.5 yr analysis including all 45 pulsars and their respective red noise parameters. Additionally, the effective sample size in the case of the MCMC is computed based on the correlation length of the chain, while in the reweighting scenario it is computed based on the variance of the weights and the two quantities are therefore not fully analogous. Nevertheless, they can serve to give an impression of the relative performances of reweighted network and MCMC in estimating expectation values (see chapter 3.3.3 of [RC04]). It is clear that the MCMC outperforms the network in this regard, even more so if one considers the additional cost required to simulate the data and to train the NN. This conclusion becomes more nuanced when reconsidering the inference objective. If the aim is to produce a complete posterior distribution for the full noise model including red noise, MCMC is the preferable option. When focusing only on the SGBW parameters, figs. 7 and 9 and the hellinger distances table 3 suggest that the unweighted NN-learned posterior is a good approximation. It should be noted that the weights are computed based on the 22-dimensional posterior, which as seen, is not accurately learned. Particularly, in scenarios where the requirement is to perform large numbers of inference analyses in limited time, the network might be a suitable alternative to the MCMC.

In conclusion we find from comparing the transformer network produced posteriors with the MCMC posteriors for the testing series based on visual comparison and the Hellinger distances that the transformer network does a good job at learning the two dimensional posterior distribution for the SGWB parameters for the tested noise series and in doing so outperforms the original network architecture. On the other hand we find that the attempt to evaluate the networks performance on a global level based on the weight distribution fails, probably due to the dimensionality of the parameter space. Generally, the network fails to outperform the MCMC based on the respective sampling efficiencies, but might serve as an alternative to MCMC methods in smaller dimensions.



(a) Noise series 1

(b) Noise series 2

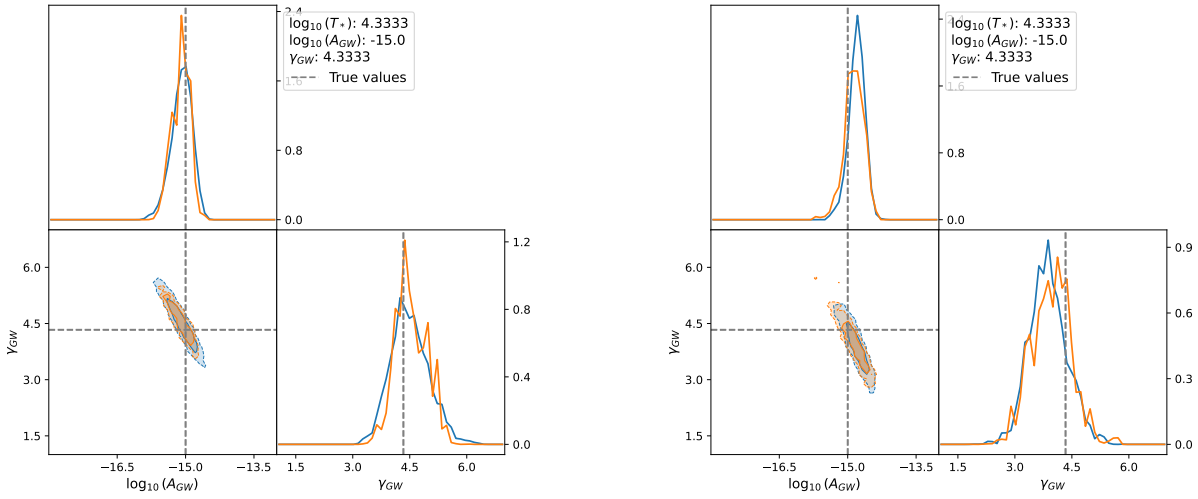
(c) Noise series 3

(d) Noise series 4

Figure 9: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a reweighted NF (orange) with transformer embedding, for the grid spaced noise series $1-4$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

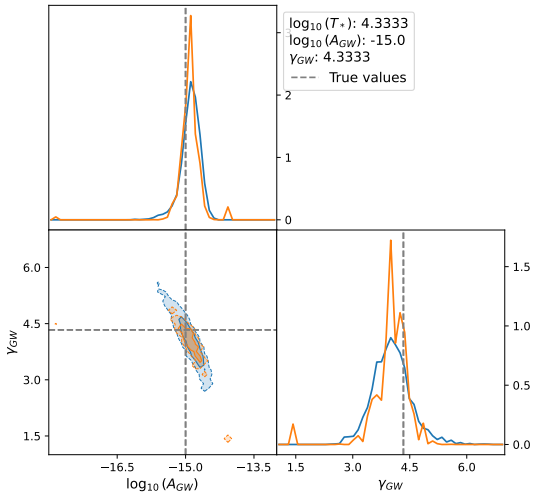
9.3 Introducing Diffusion

The next step was to change the generative network from a coupling block based normalizing flow to a conditional flow matching diffusion network (see section 8.3). Since we found the transformer encoder embedding to outperform the LSTM embedding network, we kept the transformer embedding here. For this network architecture we limited the hyperparameter tuning to the generative network and kept the previously found best configuration for the transformer from the previous section. The hyperparameter for the diffusion network we found to work best are 10 hidden layers for the FC network with 256 nodes each. The batch size for the optimizer was set to 1024 and training was performed over 1000 epochs, while the arc preprocessing routine from eq. (229) was used. The trained network is again evaluated on the testing noise series from table 2 and the resulting posterior distributions for the diffusion network and the MCMC posteriors can be seen in figs. 11 and 12. As can directly be observed by comparing

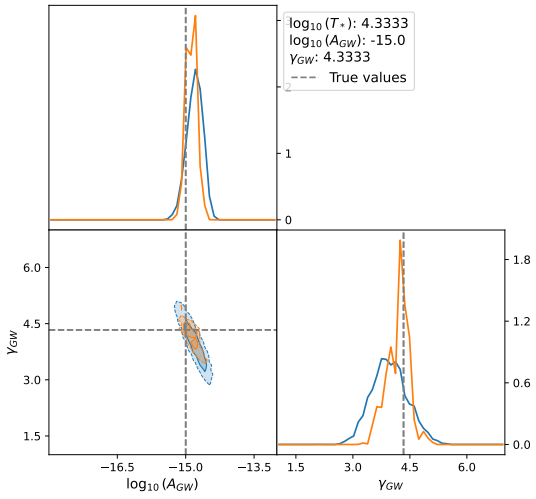


(a) Noise series 5

(b) Noise series 6



(c) Noise series 7

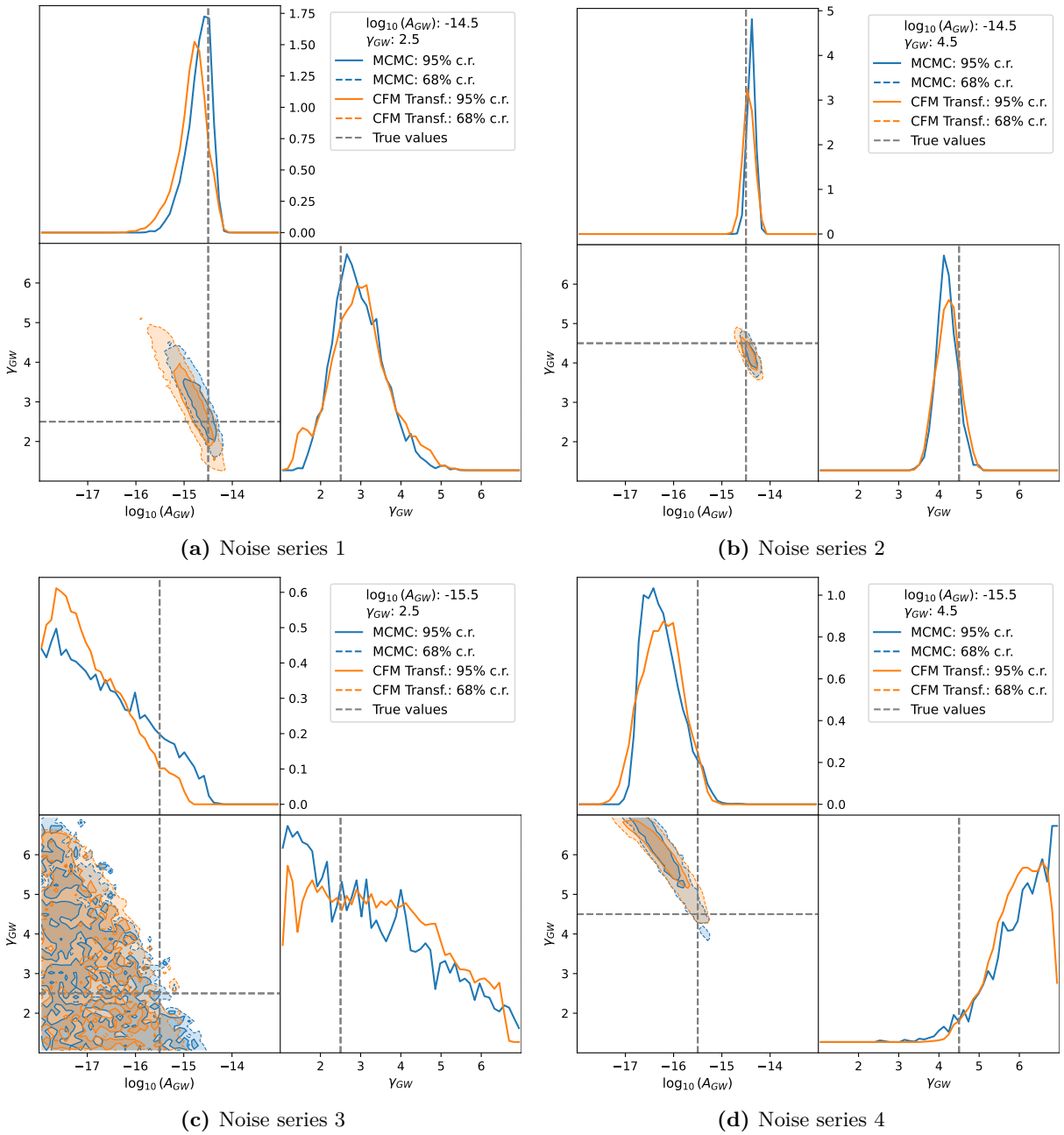


(d) Noise series 8

Figure 10: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a reweighted NF (orange) with transformer embedding, for the nominal best valued noise series $5-8$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

figs. 11 and 12 to figs. 6 and 7, the diffusion network does not manage to learn the posterior distributions as accurately as the normalizing flow. The posteriors are broader for most noise series and in table 3 we can see that the Hellinger distances to the MCMC posteriors are larger for all but noise series four, where the distances are equal. The diffusion network even performs worse than the original network in three cases, while being significantly better in only one case. This result is surprising when considering the differences between normalizing flow and diffusion networks. Since the normalizing flow must be invertible, it is generally expected to be more constrained in learning complex distributions, while the diffusion network is not limited in this way. Therefore, it is generally thought of as being more flexible than a normalizing flow. In the hyperparameter testing process we found the diffusion network to perform worse for all hyperparameter configurations, which is why we attempted the large number of epochs of 1000 shown here (We then use the NN parameters from the best performing epoch, based on the validation loss).

We further made the observation that, while the NF is forced to map the samples drawn from the latent distribution to the support of the prior distribution, this constraint is not present for the diffusion network. This means that the diffusion network can learn posterior distributions that assign non vanishing probability density to regions where the prior would theoretically force them to be zero. This shortcoming of the architecture, and whether it is related to the underperformance, could be further investigated in the future. The underperformance of the diffusion network could also hint at a problem with the implementation we have overlooked.



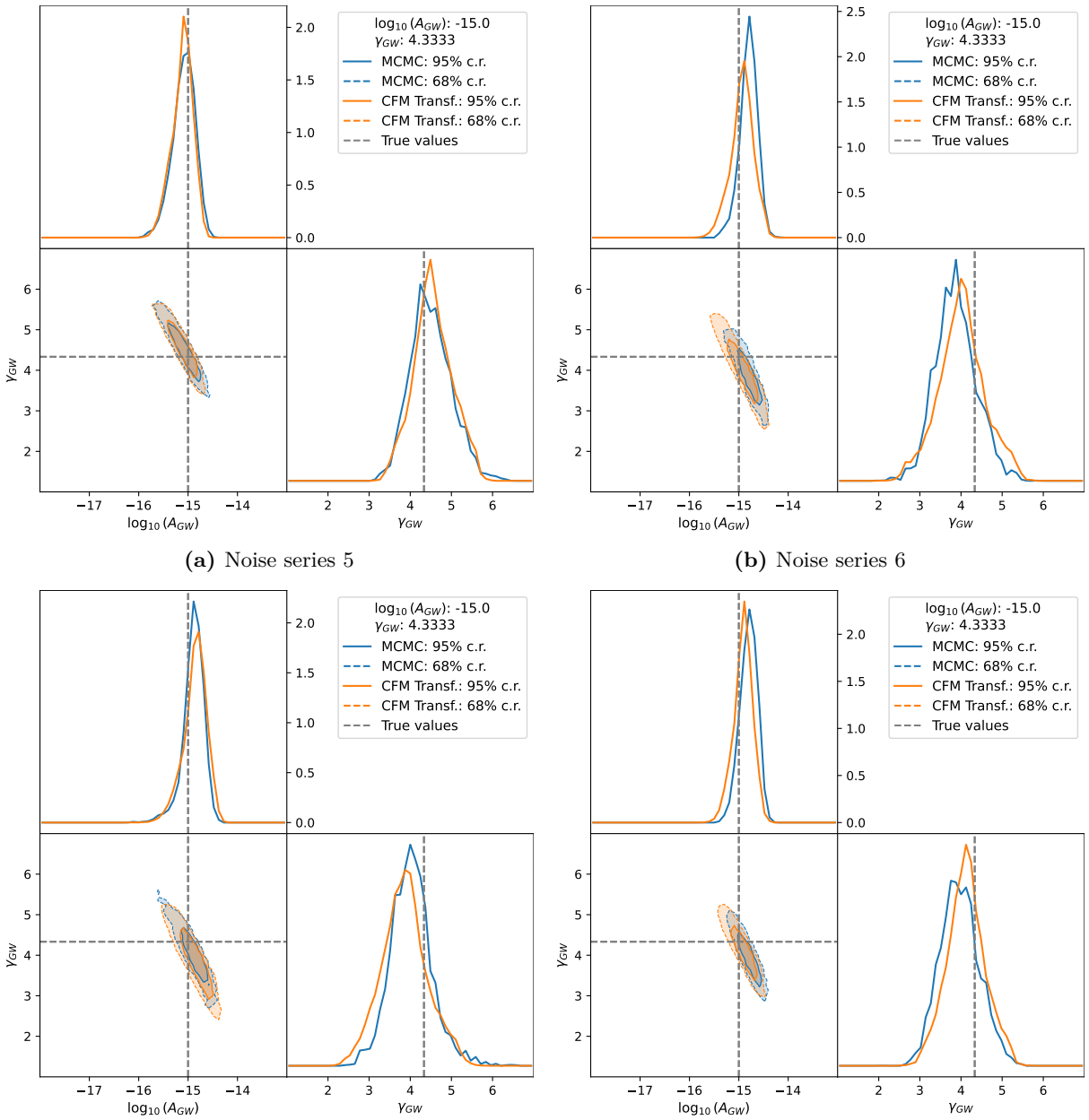
(a) Noise series 1

(b) Noise series 2

(c) Noise series 3

(d) Noise series 4

Figure 11: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) using a conditional flow matching diffusion network and transformer embedding for the grid spaced noise series $1-4$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.



(a) Noise series 5

(b) Noise series 6

(c) Noise series 7

(d) Noise series 8

Figure 12: Cornerplots of the posterior densities for the CPL SGWB noise parameters, produced with MCMC (blue) and a NN (orange) using a conditional flow matching diffusion network and transformer embedding for the nominal best valued noise series $5-8$. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

9.4 Polynomial Models

So far we have only used the CPL model for the SGWB noise and the training data provided by Shih et al. We now investigate higher order polynomial noise models, in particular the running power law model from eq. (216) and the poly3 model from eq. (218). We start out by investigating the poly3 model since mathematically the RPL model is simply the special case of $c_3 = 0$. The simulation of the training data and testing noise series is performed with a modified version of the *micropta* Python module, written by Marat Freytsis in the production

of [Shi+23]. We used the code as a basis and modified it to be able to generate noise series based on the new noise models. The simulation process is described in detail in section 8.1. We also re-simulated a training dataset for the CPL model as a test, and found similar results to the data provided by Shih et al. upon training a neural network on the re-simulated data. When simulating the data for the poly3 model, we ran into the problem that the functional form of the residual power spectral density in the poly3 case,

$$P_g(f) \overset{\text{poly3}}{=} \frac{c_0^2}{12\pi^2 f_{\text{ref}}^3}\left(\frac{f}{f_{\text{ref}}}\right)^{-\gamma_{\text{p3}}} \quad \text{with}$$
$$\gamma_{\text{p3}} \coloneqq \frac{c_3}{6}\log\left(\frac{f}{f_{\text{ref}}}\right)^2 + \frac{c_2}{2}\log\left(\frac{f}{f_{\text{ref}}}\right) + c_1,$$

allows for significantly higher values of $P_g(f)$, since here the dependence on the frequency is additionally present in the exponent. When analyzing the model using MCMC inference, this does not pose a problem since the likelihood vanishes in the limit of large $P_g(f)$ and the region of parameter space is simply ignored. When simulating noise series based on a uniform prior distribution with hyperrectangular support, noise series for every region of the allowed parameter space have to be simulated. This leads to frequent numerical instabilities in the simulation process. We attempt to mitigate this effect by choosing a different prior distribution that excludes the problematic regions. This change in prior distribution should in theory not distort the result since the posterior distribution would have been very close to zero anyway. We choose a threshold value $\epsilon = 1\,\text{s}$ and only include regions of the hyperrectangle in parameter space, where

$$\max_f \sqrt{\frac{P_g(f)}{\Delta f}} < \epsilon. \tag{232}$$

This quantity corresponds to the excess timing delay measured by the NANOGrav collaboration in [Afz+23] and was found to be smaller than 10^{-6} for all investigated frequencies. Since we simulate the training data on a larger frequency range than was measured by the NANOGrav collaboration, we chose a conservative threshold of $\epsilon = 1\,\text{s}$. Additionally, the more flexible quantile transformation preprocessing method, described in section 8.2, is used. For testing purposes, we performed an MCMC analysis of the poly3 model for the NANOGrav 15-year dataset using the PTA analysis framework PTArcade [Mit+23]. The uniform prior distribution range for the noise parameters used are given as: $\log_{10} c_0$: $[-18, -12]$, c_1: $[0, 10]$, c_2: $[-10, 9]$, c_3: $[-10, 15]$. The prior ranges for the red noise parameters for all pulsars remain at: $\log_{10}(A_a^r)$: $[-19, -13]$ and γ_a^r: $[1, 7]$. The resulting MCMC posterior distribution can be seen in fig. 13.

The aim is to use the MCMC posterior for the real NANOGrav data to get a point estimate of the parameter values to use for the simulation of testing noise series. We use the maximum posterior estimate, which could have been computed using KDE approximation. Instead, we used a simpler method where, for every point of the MCMC posterior sample, we compute the number of points with a euclidean distance smaller 1 from the point. We simply take the point with the largest number of neighbors as our estimate. Using the proposed prior distribution, we simulate a training dataset of 10^6 pairs of parameter combinations and noise series, on which a coupling block NF with transformer embedding using the setup from section 9.2 is trained. The trained network is evaluated on four generated testing noise series based on the maximum posterior estimate. The resulting posterior distributions for one noise series can be seen in fig. 14 as an example. One can clearly see that in this case the network fails to learn the underlying distribution and does not manage to reconstruct the induced true parameters. This does not change for different hyperparameter configurations of the network. One should

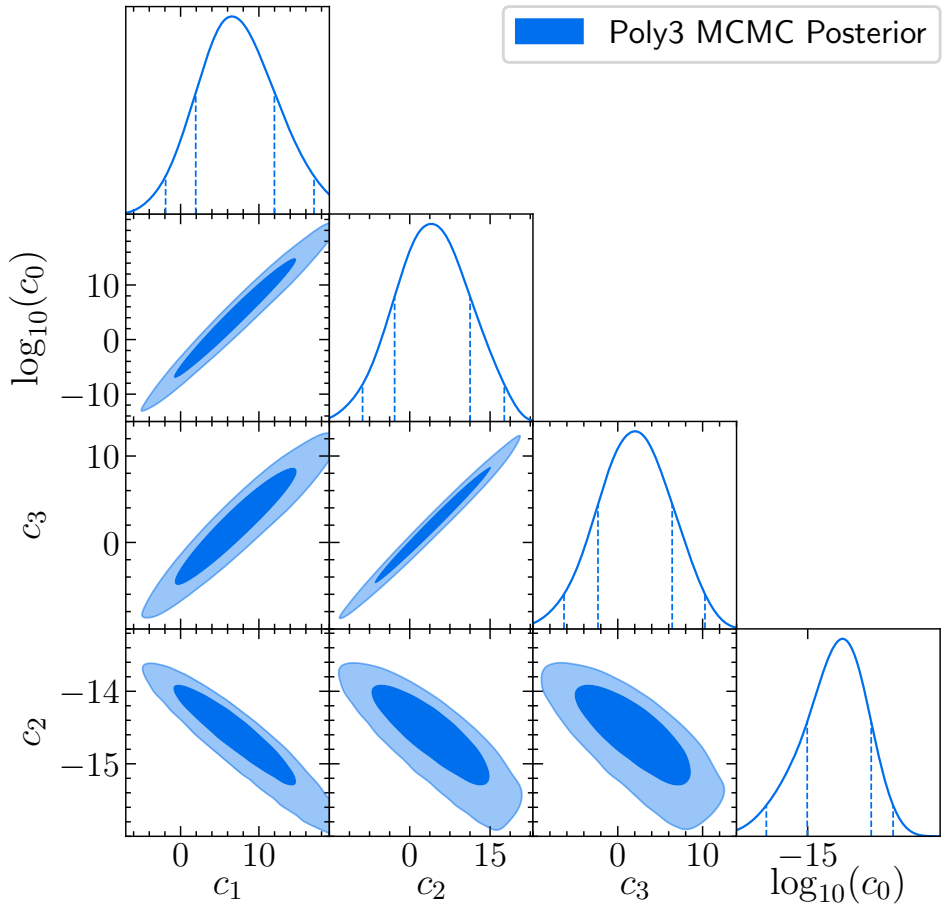


Figure 13: MCMC posterior for a poly 3 SGWB model (see eq. (218)), produced on the NANOGrav 15-year dataset.

note, that the network does manage to learn the given prior distribution. When comparing the produced network posterior distributions with the learned prior, a strong resemblance can be seen. This could indicate that the network does not deal well with the chosen prior and therefore struggles to learn the correct posterior distribution. The possibility of further problems in the implementation cannot be excluded.

We now focus on the running power law model i.e. a second order polynomial model (see eq. (216)). From the MCMC posterior for the poly3 model in fig. 13 it can be clearly seen that the parameters c_3 and c_2 are highly correlated. Their correlation coefficient is 0.988, which strongly suggests that adding the additional parameter c_3 simply introduces a redundancy into the description. The choice of a simpler model also has numerical advantages. Due to the relatively high degree of redundancy in the description for the poly3 model, we chose broad prior intervals to ensure all interesting regions of the parameter space are scanned. Additionally, since we omit the quadratic term in the exponent of $P_g(f)$, the assumed values are more constrained. Based on the previously performed NANOGrav 15 yr dataset analysis of the RPL model in [Aga+25] the chosen uniform prior ranges are: $\log_{10} c_0$: $[-24, -9]$, c_1: $[-5, 20]$, c_2: $[-2, 4]$, where $\log_{10}(A_a^r)$: $[-19, -13]$ and γ_a^r: $[1, 7]$ remain as before. One should note that the numerical difference to the parameter ranges chosen in [Aga+25] stem from the different reference frequencies used. While we use a reference frequency of 1/1 yr, the reference frequency used in the paper is 1/10 yrs. As shown in the same paper there is a simple way of converting a set of parameters $(\log_{10}(c_0), c_1, c_2)$, based on one reference frequency f_{ref}, to the corresponding parameters $(\log_{10}(c_0'), c_1', c_2')$ for

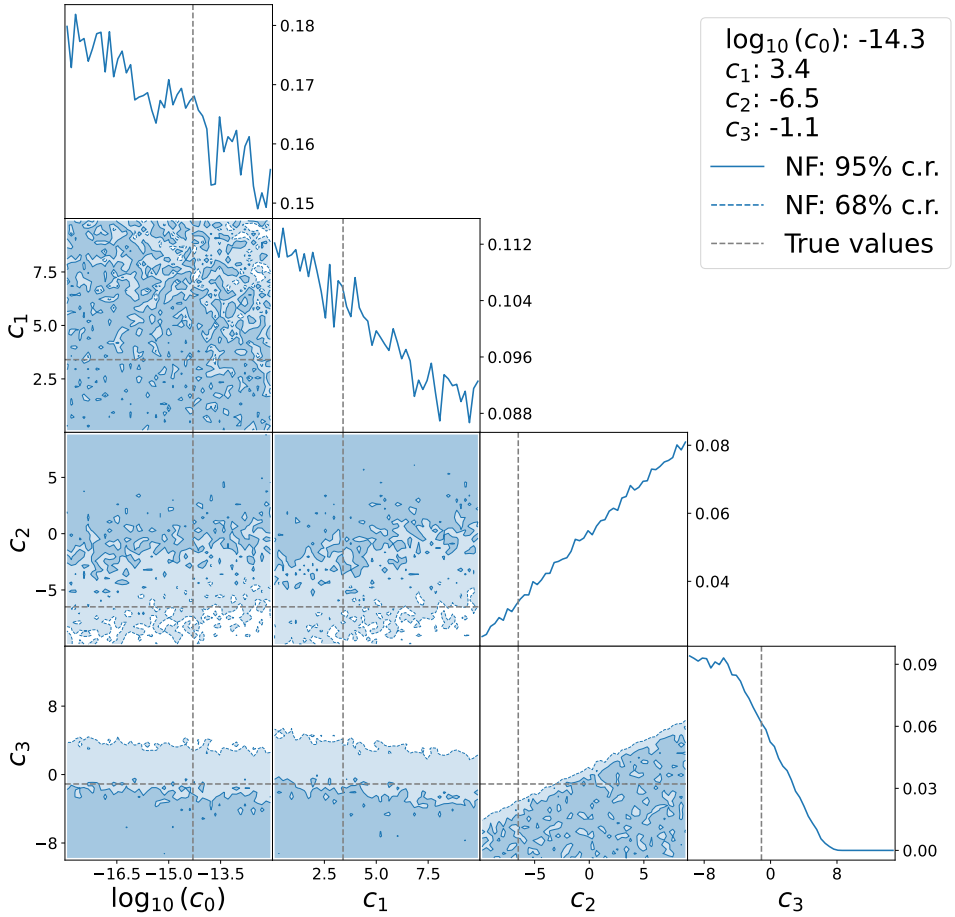


Figure 14: Transformer NF produced posterior distribution for the poly3 noise model. The outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

another reference frequency f'_{ref}

$$\begin{pmatrix} \ln(c'_0) \\ c'_1 \\ c'_2 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{R}{2} & -\frac{3R}{2} \\ 0 & 1 & R \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \ln(c_0) \\ c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} \frac{3R}{2} \\ 0 \\ 0 \end{pmatrix}, \quad \text{where} \tag{233}$$

$$R := \ln(f_{\text{ref}'}/f_{\text{ref}}). \tag{234}$$

One should note that this formula is based on the natural logarithm of c_0 and not the decadic logarithm as before.

Using eq. (233), the three dimensional maximum posterior parameter values produced in [Aga+25] were used to compute the corresponding values for the reference frequency of $1/(1\,\text{yr})$ given as: $\log_{10}(c_0) = -14.36$, $c_1 = 4.45$ and $c_2 = 0.78$. We then simulated a training dataset of 10^6 pairs of parameter configurations and noise series, on which we trained a coupling block NF with transformer embedding, using the setup from section 9.2. Using the best fit parameter values we simulate four testing noise series on which we evaluate the network. The produced posterior distribution is shown in fig. 15 for one of the testing noise series as an example, with similar results for the other cases. In all four cases the NF manages to constrain the posterior distribution around the true parameter values.

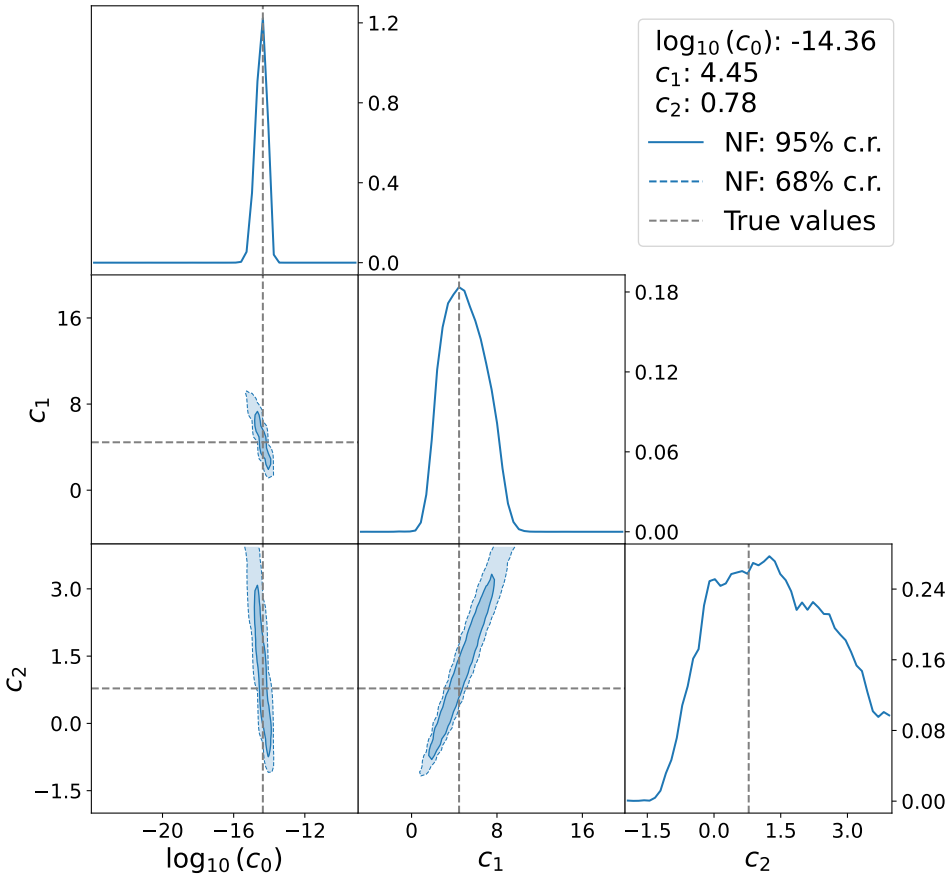


Figure 15: Transformer NF produced posterior distribution for the RPL noise model. The outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

9.5 Broken Power Law Model

The next SGWB model we investigate corresponds to the prediction of GWs from phase transitions in the early universe and numerically follows a broken power law (see eq. (220)). As can be seen from the spectral shape function

$$\mathcal{S}(f) = \frac{1}{\mathcal{N}} \frac{(a+b)^c}{\left(b\left(\frac{f}{f_b}\right)^{-a/c} + a\left(\frac{f}{f_b}\right)^{b/c}\right)^c}, \tag{235}$$

the spectrum is bounded on the frequency domain, since we only consider positive values for a and b. The simulation is thus less vulnerable to numerical instabilities and we can use the arc preprocessing scheme from eq. (229). We again simulate a training dataset of 10^6 pairs of parameter configurations and noise series, based on the uniform prior distributions used in the NANOGrav 15 yr analysis on new physics signals [Afz+23], given as: $\log_{10}(\alpha_*)$: $[-2, 1]$, $\log_{10}(T_*)$: $[-4, 4]$, $\log_{10}(H_*R_*)$: $[-3, 0]$, a: $[1, 3]$, b: $[1, 3]$ and c: $[1, 3]$. The red noise parameter ranges are kept at $\log_{10}(A_a^r)$: $[-19, -13]$ and γ_a^r: $[1, 7]$. We use the maximum posterior values found in [Afz+23]: $\log_{10}(\alpha_*) = 1$, $\log_{10}(T_*) = -0.79$, $\log_{10}(H_*R_*) = 0$, $a = 2.14$, $b = 3$ and $c = 1$ to simulate four testing noise series. A coupling block NF with transformer embedding using 3

hidden layers for the sub-networks of dimension 128 and 8 splines for the RQS transformations for the generator network is then trained. The hyperparameters for the encoder network remain unchanged from the previous sections. The batch size for the optimizer remains at 1024 and the training was performed over 100 epochs. Additionally, a diffusion network, using the same transformer embedding with 12 hidden layers for the FC network with 256 nodes each is trained, using 1024 as batch size for the optimizer, while training is performed over 500 epochs. The posterior plots for the four testing noise series, produced with both trained networks, can be seen in figs. 16 to 19.

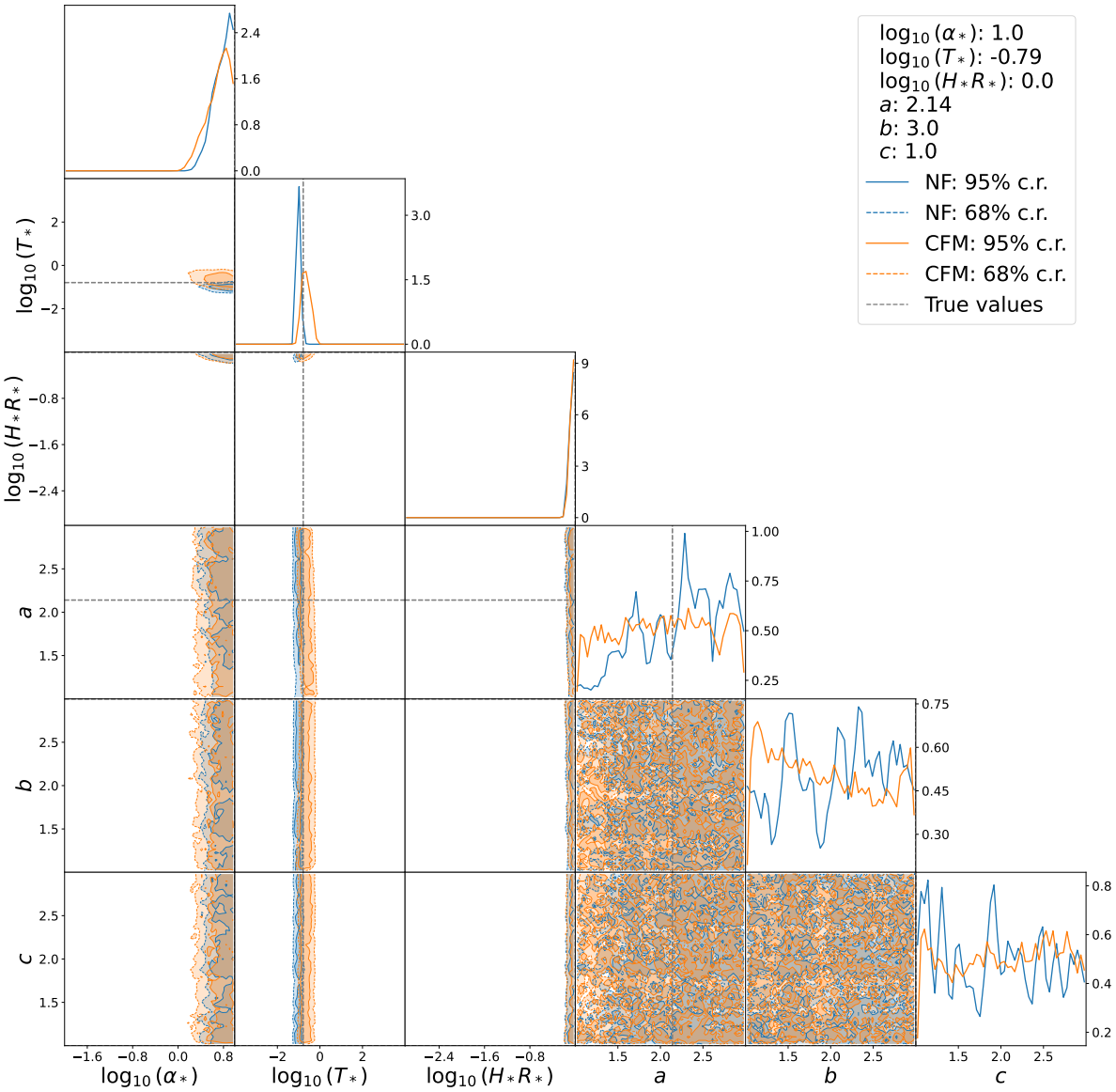


Figure 16: Cornerplots of the posterior densities for the BPL SGWB noise parameters, produced with a NF (blue) and a diffusion model (orange) evaluated for the first noise series. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

Figures 16 to 19 show clearly that both networks manage to constrain the posterior distribution around the true values for the first three parameters. In tables A9 to A12, mean, median and maximum posterior values, based on the one dimensional marginalized distributions, can be

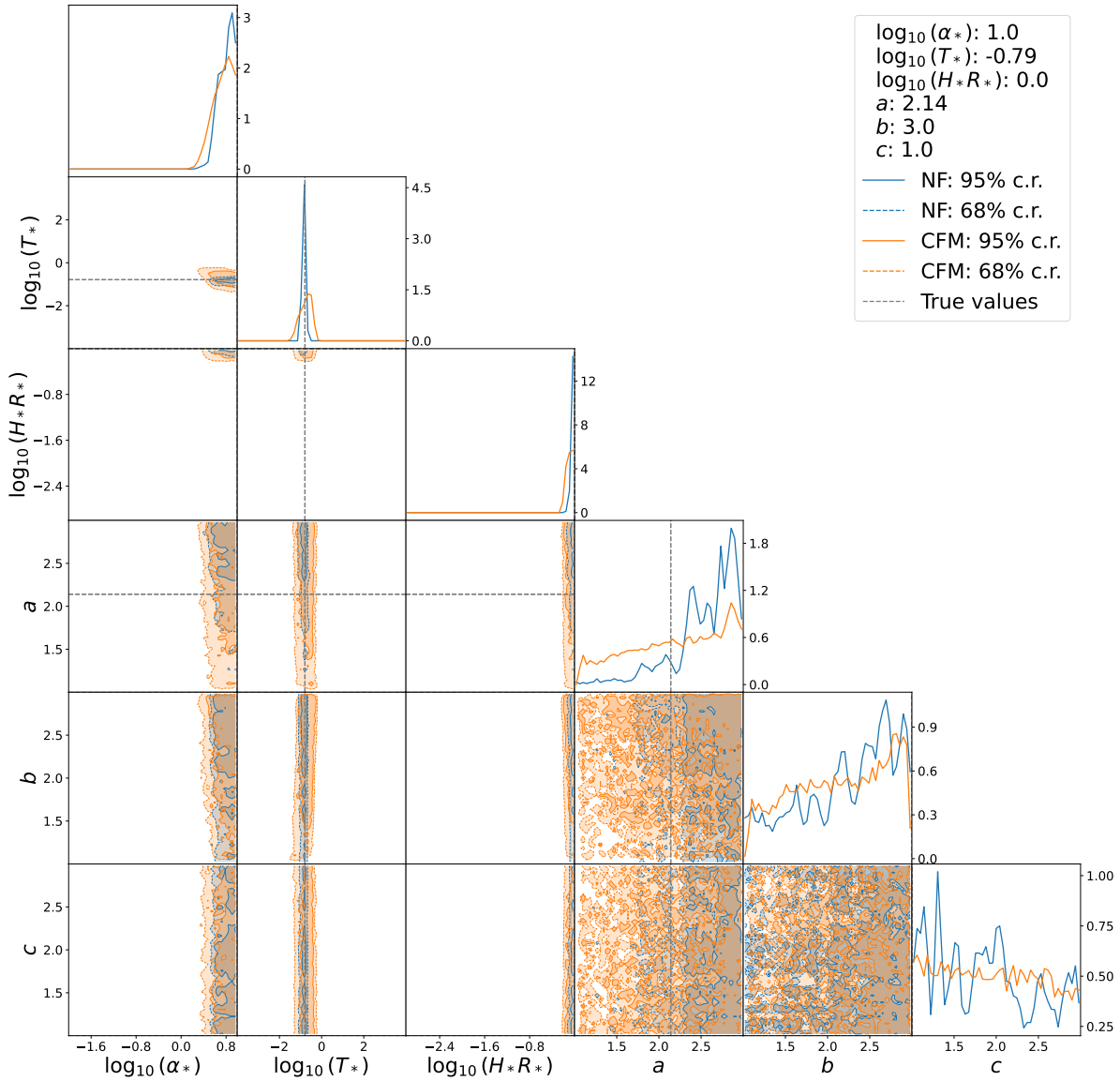


Figure 17: Cornerplots of the posterior densities for the BPL SGWB noise parameters, produced with a NF (blue) and a diffusion model (orange) evaluated for the second noise series. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

found for all posterior distributions and for both networks. We find again that the normalizing flow outperforms the diffusion network, producing statistical estimators that are generally closer to the true parameter values, while predicting credible regions that are smaller around the true values. We also note that the diffusions network sometimes predicts credible region bounds that lie outside of the prior ranges. As mentioned before, this is due to the fact that diffusion networks are not constrained to the prior region and can therefore learn mathematically impossible posterior distributions. This effect is amplified in this scenario, since the true values for $\log_{10}(\alpha_*)$ and $\log_{10}(H_*R_*)$ were chosen exactly at the border of the prior regions, as found by the NANOGrav collaboration. Larger prior regions could be considered, but the ranges were chosen by the NANOGrav collaboration based on the uncertainties of prior physical simulations (see [Afz+23]), therefore changing them would not be physically sensible. In particular, values for $H_*R_* > 1$ would violate the principle of causality.

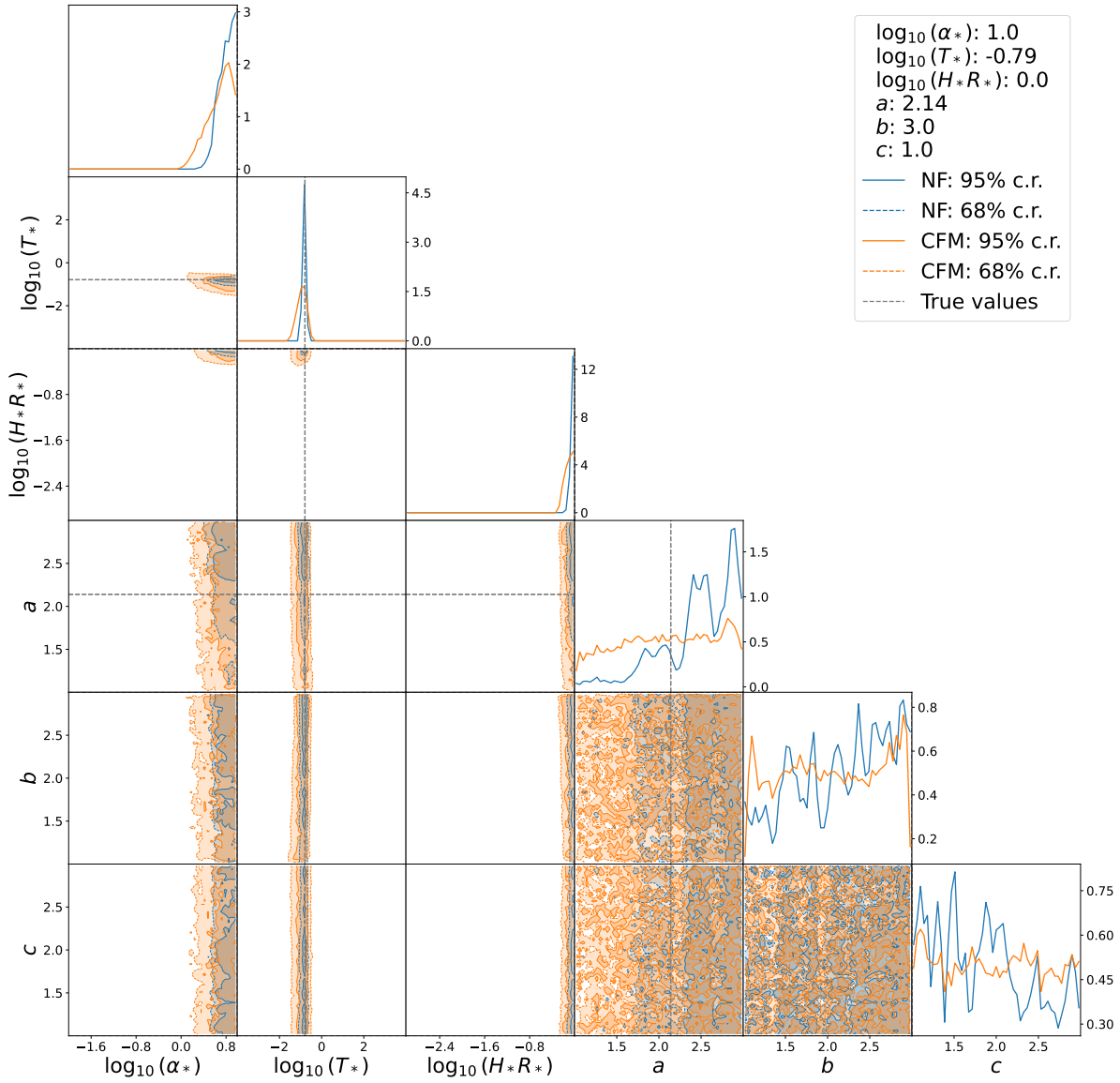


Figure 18: Cornerplots of the posterior densities for the BPL SGWB noise parameters, produced with a NF (blue) and a diffusion model (orange) evaluated for the third noise series. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

In contrast to $\log_{10}(\alpha_*)$, $\log_{10}(T_*)$ and $\log_{10}(H_*R_*)$, both networks do not manage to recover the true shape parameters a, b and c from the noise series. In the case of b and c, this can be explained from the functional form of the model in eq. (235); b governs the slope of the right flank, and c governs the width of the peak of the spectrum on logarithmic axes. The peak frequency, corresponding to the true value $\log_{10}(T_*) = -0.79$, is $\log_{10}(f_b) = -0.38$ $\log_{10}(1/\mathrm{yr})$. The logarithmic frequency range used to simulate the training data, is given as $[-3.10, -0.10]$. Consequently, it is mostly the left flank of the spectrum that is relevant for generating the training data in which b and c are not as relevant. In [Afz+23], a similar insensitivity to the parameters b and c was found for the measured data. The insensitivity to a can potentially be explained with a generally smaller influence of the shape parameters on the spectrum. In particular, when probing the spectrum on the frequency window used for the simulation. We

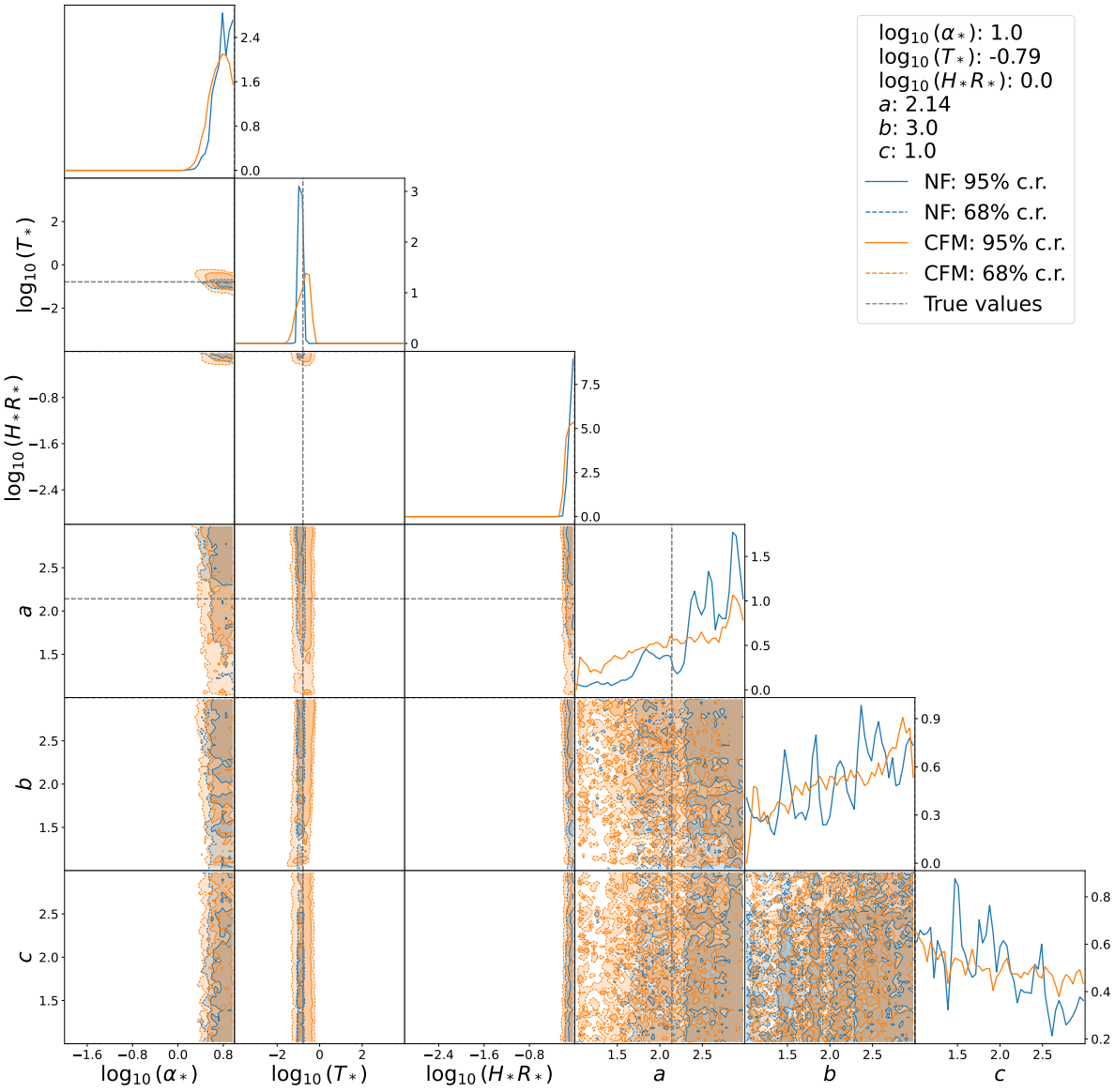


Figure 19: Cornerplots of the posterior densities for the BPL SGWB noise parameters, produced with a NF (blue) and a diffusion model (orange) evaluated for the fourth noise series. For each cornerplot the outer curves depict the marginalized posterior density for one SGWB parameter while the the contour plots show the joint probabilty density via their 68%- and 95%-credible regions. Additionally the parameter values used to generate the noise series are marked in grey.

have simulated additional noise series by shifting $\log_{10}(T_*)$, and thereby the peak frequency, around $\log_{10}(T_*) = 0$ to see whether this allows to recover the parameter a, while keeping all other parameters fixed. The resulting marginalized posterior distributions for a can be seen in fig. 20. We note, that for $\log_{10}(T_*) = 0$ and $\log_{10}(T_*) = 0.2$ a peak around the true value of a forms. In all other cases the posterior does not reflect the true value and rather tends to overestimate a, indicating that, while the network is somewhat sensitive to a for specific values of T_*, the network cannot reliably recover the true value even for different peak frequencies. This again might be due to the dimensionality of the problem.

In conclusion, while the network does not manage to accurately recover the true parameter values in all dimensions, it succeeds at sensibly constraining the posteriors in $\log_{10}(\alpha_*)$, $\log_{10}(T_*)$,

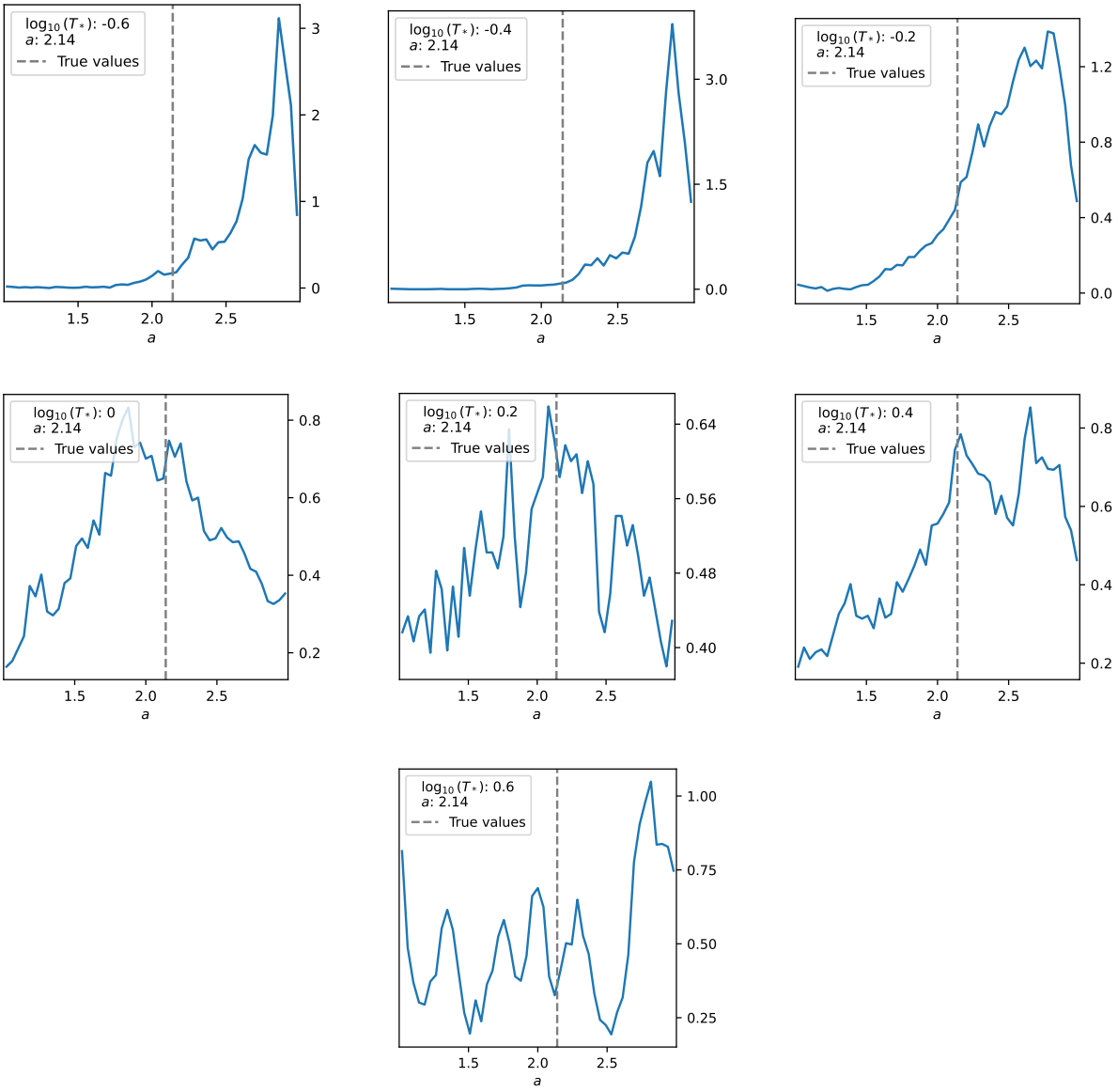


Figure 20: One dimensional posterior distributions for the parameter a, produced with a NF evaluated on testing noise series. All parameters except T_* are kept fixed at $\log_{10}(\alpha_*) = 1$, $\log_{10}(H_* R_*) = 0$, $a = 2.14$, $b = 3$ and $c = 1$, while $\log_{10}(T_*)$ is varied around 0.

and $\log_{10}(H_* R_*)$. This reduced sensitivity to the shape parameters a, b and c is also reflected in the MCMC posteriors produced in [Afz+23]. It should again be noted that, in order to evaluate the performance of the network at accurately estimating the true posterior distribution, it will be necessary to either perform an MCMC analysis of the testing series, or to compute the reweighted posteriors to use as a comparison.

9.6 Inference Time

Trying to make an assessment of whether amortized posterior estimation is a suitable alternative or supplement to conventional MCMC inference, we consider the inference time required. The posterior distribution in fig. 13 was produced for the full NANOGrav 15 yr dataset, using a noise model including 140 parameters. The total time required to produce 20 chains totaling $\approx 1.5 \times 10^6$ samples on the university high performance computing cluster PALMA, was one week. The chains were produced in parallel on 20 Intel Xeon Gold 6140 CPUs. The total time required to perform amortized posterior inference, is determined by multiple tasks. Here, we illustrate computational costs using the example of the full SBI pipeline used to produce the posterior distributions for the BPL model. The first task is simulating the training dataset. This took $\approx$ 3 hours on one Intel Xeon Gold 6140 CPU. It should be mentioned that, there is significant room for acceleration here, since the simulation process is fully parallelizable. Training the neural network took 3 hours and 10 minutes on an Nvidia GeForce RTX 4090 GPU, using PyTorchs internal GPU acceleration. Again, this could be further parallelized on multiple GPUs to accelerate the training. Evaluating the trained model on the testing noise series takes only a few seconds. If the aim is to use the network for importance sampling the additional time required for the computation of the weights has to be taken into account. Since we do not find the network to perform well as a candidate distribution, we do not further take this into account and focus on the case where the unweighted distribution is directly used as the posterior estimate. Lastly, the time required to fine-tune has to be considered for an honest assessment. Here, fine-tuning was mainly performed to find the best-performing setup for the transformer NF in the CPL scenario. From there, we have kept the hyperparameters mostly fixed for the other noise models. Hyperparameter tuning was carried out over the course of several weeks. However, using the method described in section 9.2, in which all parameters are held fixed except for one that is varied, the estimated time required to evaluate at least three values for each hyperparameter would take on the order of 100 hours. The time highly depends on how good the posterior approximation has to be and on how well the relevant range of hyperparameters can be constrained prior to testing. When exploring different noise models, we have found that the hyperparameter configuration found for the CPL model also produces good results for different noise models and that further hyperparameter tuning only slightly affects the quality of the posterior. On the other hand, if additional data is included and the dimensionality of the inference problem thus increases significantly, it may be necessary to modify the network architecture to maintain good posterior results. If hyperparameter testing is unavoidable, the total times required for SBI and MCMC inference lie at a similar order of magnitude. More so, if one considers that the MCMC was performed for a significantly larger noise model than the SBI. In conclusion, in order for SBI to offer an improvement in speed over conventional MCMC, either hyperparameter testing must be reduced to a minimum or other methods have to be found to reduce inference time. One idea to improve speed, that could be further explored uses the flexibility of the RPL model. Since the spectral density of the RPL model can be fitted to large set of other noise models, one could try to use an embedding network trained on an RPL dataset for other models without having to retrain the embedding. This would reduce the complexity of the network and thereby training and hyperparameter testing time significantly, since only the generative model would have to be trained.

10 Conclusion and Outlook

The aim of this work was to explore the possibility of applying simulation-based inference, in particular augmented posterior estimation, to PTA data. Building on the foundation laid by Shih et al. in [Shi+23], we tried to improve the neural network architecture and to explore its applicability in testing noise models different from a CPL. In this work, we have been able to effectively reproduce the CPL results obtained by Shih et al. From there we have introduced a transformer encoder network which has allowed to improve upon the original network architecture proposed by Shih et al. It remains a challenge to find a criterion that allows to globally compare the performance of different networks that is not just based on a relatively small sample of testing noise series. Using importance sampling, we conclude that the transformer NF does not manage to accurately learn the 22-dimensional posterior distribution. Compared to an MCMC produced posterior distribution the network has comparatively low sampling efficiency on the order of 10^{-3}, while the MCMC used on the full NANOGrav 12.5yr has a sampling efficiency of 0.02 which is significantly higher. From this we conclude, that a transformer NF with the architecture used in this work does not pose a good candidate distribution to perform importance sampling with. On the other hand we find, when visually comparing the two dimensional posterior distributions in $\log_{10}(A_{GW})$ and γ_{GW} for both MCMC and the NN, that they align well, which is also reflected in generally small Hellinger distances (> 0.34). Therefore, in cases where one is not interested in the full distribution but only its marginalizations, e.g. when computing estimators such as mean, median or variance, the network approximation is good enough such that importance sampling is not necessary. When introducing a conditional flow matching diffusion network as a replacement for the normalizing flow, we find that it performs worse than the NF in every case. In some cases it is even outperformed by the original network architecture, based on an LSTM encoder network. Since this result is surprising, considering the generally more flexible diffusion network architecture, this might hint at problems in the implementation of the diffusion network. We further explored simulation-based inference on different noise models from the CPL. We first explored the polynomial models of an RPL and a third degree polynomial. While the numerical difficulties, and the resulting compromises on the prior distribution necessary, proved hard to overcome in the poly3 case, the MCMC analysis performed in the process suggest, that using a third degree polynomial mainly introduces redundancy compared to a second degree polynomial. Testing the RPL model was successful. In the RPL case, both NF and diffusion model were able to produce a visually constrained posterior distribution around the true parameter values used for simulating the testing noise series. Similar results were found for a BPL model. While the network was not sensitive to all six parameters it managed to correctly constrain the posterior distribution for the main $\log_{10}(\alpha_*)$, $\log_{10}(T_*)$, and $\log_{10}(H_*R_*)$ parameters. It remained insensitive to b and c while small sensitivity to a could be found for specific choices of peak frequency. The lack of sensitivity to the shape parameters a, b and c is not too surprising, when comparing the result to the MCMC posterior distributions found by the NANOGrav collaboration for the 15 yr dataset, where reduced sensitivity to a and almost no sensitivity to b and c was found. In total, this indicates that it will be possible to successfully use amortized posterior estimation for different noise models from the CPL, a result that was also found by [LL25]. To further evaluate the performance of the neural networks at estimating the posterior distribution for the new models, it will be necessary to perform a direct comparison to MCMC produced, or reweighted posterior distributions as in the case of the CPL. When evaluating potential speed improvements over conventional MCMC methods, the key factor is whether hyperparameter tuning is required. In cases where it is not, simulation-based inference can produce posterior distributions significantly faster. However, if extensive hyperparameter optimization is necessary, the computational advantage of SBI over MCMC vanishes. One interesting direction for reducing the computational cost of SBI, could be to reuse an embedding network trained on an RPL model for different noise models without

retraining. Given the flexibility of the RPL model, this strategy could substantially reduce the required complexity of the network and thereby inference time, while preserving relatively accuracy in the posterior estimates.

A Additional Mathematical Content

A.1 Random Variables

This subsection is based on chapter 1 of [Cow02].

For two independent random variables $x \sim p_x(x)$ and $y \sim p_y(y)$, $z \coloneqq x + y$ is again a random variable and its probability distribution is given as the convolution of p_x and p_y

$$p_z(z) = \int_{-\infty}^{\infty} p_x(x)\, p_y(z - x)\, dx. \tag{A1}$$

If $z \coloneqq x\, y$ denotes the product of two independent random variables, the probability distribution of z is given as

$$p_z(z) = \int_{-\infty}^{\infty} p_x(x)\, p_y(z/x) \frac{1}{|x|}\, dx. \tag{A2}$$

Given a random variable $x \sim p_x(x)$ and a function F, $z \coloneqq F(x)$ is a random variable, the probabilities for which are given by the *pushforward*

$$P(z \in B) = \int_{x \in F^{-1}(B)} p_x(x) dx. \tag{A3}$$

If F is a diffeomorphism, the change of variables rule can be applied and the probability distribution of z is given as

$$p_z(z) = p_x(F^{-1}(z))\, |\det(DF^{-1}(z))|, \tag{A4}$$

where DF^{-1} denotes the Jacobian matrix of the inverse of F. From this follows, that in the case of a Gaussian distributed random variable x (or analogously in the case of a multivariate distribution with diagonal covariance), with $\mu = 0$ and standard deviation σ, the new random variable $z = \frac{x}{\sigma}$ has the probability distribution

$$p_z(z) = \frac{1}{\sqrt{2\pi}} \exp(-z^2/2).$$

Even if F is not diffeomorphic, in many practically relevant cases one can find a finite partition of the sample space of x, $\{\mathcal{D}_i\}$, such that

$$F_i \coloneqq F|_{\mathcal{D}_i} \tag{A5}$$

are diffeomorphisms for all i. The change of variables formula can then be applied to each F_i and the probability distribution of z is given as

$$p_z(z) = \sum_{\{i | z \in F(\mathcal{D}_i)\}} p_x(F_i^{-1}(z))\, |\det(DF_i^{-1}(z))|. \tag{A6}$$

A.2 Multivariate Gaussian Distributions

A multivariate Gaussian distribution for a random vector $\mathbf{x}$ in k dimensions has the form

$$\mathcal{N}[\boldsymbol{\mu}, \boldsymbol{\Sigma}](\mathbf{x}) = \frac{1}{(2\pi)^{k/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

where $\boldsymbol{\mu}$ is the expected value for $\mathbf{x}$, and $\boldsymbol{\Sigma} = \mathrm{cov}(\mathbf{x})$ is the covariance matrix. The covariance matrix has the properties that it is symmetric and positive definite. For two uncorrelated, random vectors $\mathbf{x}$ and $\mathbf{y}$, that are both distributed following a multivariate Gaussian distribution with mean zero, the covariance matrix of $\mathbf{x}+\mathbf{y}$ is given as

$$\mathrm{cov}(\mathbf{x}+\mathbf{y}) = \langle(\mathbf{x}+\mathbf{y})(\mathbf{x}+\mathbf{y})^T\rangle = \langle\mathbf{x}\mathbf{x}^T\rangle + \langle\mathbf{y}\mathbf{y}^T\rangle + \underbrace{\langle\mathbf{x}\mathbf{y}^T\rangle + \langle\mathbf{y}\mathbf{x}^T\rangle}_{=0 \text{ since } \mathbf{x} \text{ and } \mathbf{y} \text{ are uncorrelated}} \tag{A7}$$

$$= \mathrm{cov}(\mathbf{x}) + \mathrm{cov}(\mathbf{y}). \tag{A8}$$

A.3 Measures and Estimators

This section is based on chapter 9.2 of [Bol07]. Random variables are fully characterized by their probability distributions. To interpret them one can consider several measures for their location and spread. In the folowing let $x \sim p_x(x)$ be a random variable.

- One of the most important measures for the location of a distribution is its mean $\bar{x}$ given as

$$\bar{x} = \int x\, p_x(x)\, dx. \tag{A9}$$

  This is also commonly denoted as $\langle x\rangle$ or $\mathbb{E}_{p_x}(x)$.

- For a random variable $x \in (x_{\min}, x_{\max})$, another common measure of location is the median x_{median}, defined via

$$\int_{x_{\min}}^{x_{\text{median}}} p_x(x)\, dx = 0.5. \tag{A10}$$

- If the probability distribution p_x is differentiable one can also consider the modes of the distribution given by the values x' that fulfill

$$\frac{\partial p_x(x)}{\partial x}(x') = 0.$$

  One of the most important modes is the value at which p_x assumes its maximal value, if it exists. If the distribution has only one mode i.e. is unimodal, and symmetric, mean, median and maximum value coincide.

In addition to the location, one often wants to consider the spread of a distribution. Important measures of spread are:

- Using the mean $\bar{x}$ the variance of a distribution is given as

$$\mathrm{Var}(x) = \int |x-\bar{x}|^2 p_x(x)\, dx,$$

and the standard deviation is defined as

$$\sigma(x) = \sqrt{\mathrm{Var}(x)}.$$

Using the notation $\langle\cdot\rangle$ for the mean, the variance can also be denoted as $\langle|x - \bar{x}|^2\rangle$.

- Another important quantity to consider for the spread, is the r^{th} quantile of the distribution x_r, computed via solving

$$r = \int_{x_{\min}}^{x_r} p_x(x)\, dx.$$

Two of the most common quantiles to consider are the quartiles, that is the 0.25^{th} quantile or first quartile and the 0.75^{th} quantile or third quartile. In this instance, one often considers the interquartile range defied via

$$\mathrm{IQR}(x) = x_{0.75} - x_{0.25}. \tag{A11}$$

- Sometimes one is interested in the similarity of two random variables. An important metric for measuring the similarity of two probability distributions $p(x)$ and $q(x)$ is the Hellinger distance [Hel09]. It is defined via

$$\mathrm{HD}(p,q) = \frac{1}{\sqrt{2}} \left(\int dx \left| \sqrt{p(x)} - \sqrt{q(x)} \right|^2 \right)^{1/2}. \tag{A12}$$

A.4 Complex Random Processes

This section is based on chapter 12.4 of [PP02]. By a complex random variable z, we mean the random vector $(a,b) \coloneqq (\mathrm{Re}(z), \mathrm{Im}(z))$, where both a and b are themselves random variables. A complex random process $z(f)$ is therefore characterized by the real random processes $a(f)$ and $b(f)$. We now consider the case of a stationary and Gaussian, real random process $h(t)$, centered around 0 i.e. $\langle h(t)\rangle = 0$, with Fourier coefficients $z(f)$ given as

$$h(t) = \int_{-\infty}^{\infty} df\, z(f)\, e^{2\pi i f t}. \tag{A13}$$

We can show that the stationarity of the process, $\langle h(t), h(t')\rangle = \langle h(t-t'), h(0)\rangle =: C(\Delta t)$, implies $\langle z^*(f), z(f')\rangle \sim \delta(f - f')$. For this we use the inverse Fourier-transform in

$$\langle z^*(f), z(f')\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dt\, dt' e^{ift}\, e^{-if't'} \langle h(t), h(t')\rangle. \tag{A14}$$

Here, the normalization constants of the Fourier transform have been omitted. We now substitute $\tau \coloneqq t' - t$ for t' in the integral, which results in

$$\begin{aligned}
\langle z^*(f), z(f')\rangle &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dt\, d\tau\, e^{ift}\, e^{-if'(t+\tau)} \langle h(t), h(t+\tau)\rangle && \text{(A15)}\\
&= \int_{-\infty}^{\infty} dt\, e^{i(f-f')t} \underbrace{\int_{-\infty}^{\infty} d\tau\, e^{-if'\tau}\, C(\tau)}_{S(f')} && \text{(A16)}\\
&= \delta(f - f')\, S(f'). && \text{(A17)}
\end{aligned}$$

One can show similarly, that $\langle h(t)\rangle = 0$ implies $\langle z(f)\rangle = 0$. Furthermore $\langle h(t), h(t')\rangle = \langle h(t'), h(t)\rangle$ implies $C(\tau) = C(-\tau)$ and it follows that $S(f)$ is a real quantity. $S(f)$ can then be obtained by considering the autocorrelation

$$\langle h(t), h(t)\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} df\, df' \langle z^*(f), z(f')\rangle\, e^{ift}\, e^{-if't} = \int_{-\infty}^{\infty} df'\, S(f'). \tag{A18}$$

To obtain the distribution of the random processes $a(f)$ and $b(f)$, one can use the Gaussianity of $z(f)$, implying that the real processes $a(f)$ and $b(f)$ are fully characterized by their expectation values $\langle a(f)\rangle$ and $\langle b(f)\rangle$ and their covariances $\langle a(f), a(f')\rangle$, $\langle b(f), b(f')\rangle$, $\langle a(f), b(f')\rangle$ and $\langle b(f), a(f')\rangle$. One can see directly, that $\langle z(f)\rangle = 0$ implies $\langle a(f)\rangle = \langle b(f)\rangle = 0$. Additionally, we can use that $h(t)$ is real which gives

$$h(t) = \int_{-\infty}^{\infty} df\, z(f)\, e^{2\pi ift} = \int_{-\infty}^{\infty} df\, z(f)^*\, e^{-2\pi ift} = \int_{-\infty}^{\infty} df\, z(-f)^*\, e^{2\pi ift}. \tag{A19}$$

The uniqueness of the Fourier transform then implies $z(f) = z(-f)^*$, which in turn implies $a(f) = a(-f)$ and $b(f) = -b(-f)$. It follows that

$$\delta(f - f')\, S(f') = \langle z^*(f), z(f')\rangle = \langle a(f), a(f')\rangle + \langle b(f), b(f')\rangle + i\left[\langle a(f), b(f')\rangle - \langle b(f), a(f')\rangle\right] \tag{A20}$$

and therefore

$$\langle a(f), a(f')\rangle + \langle b(f), b(f')\rangle \stackrel{!}{=} \delta(f - f')\, S(f) \qquad \text{and} \tag{A21}$$

$$\langle a(f), b(f')\rangle - \langle b(f), a(f')\rangle \stackrel{!}{=} 0. \tag{A22}$$

Additionally, since $\langle z(f), z(f')\rangle = \langle z^*(-f), z(f')\rangle$, one obtains

$$\delta(f + f')\, S(f') = \langle z(f), z(f')\rangle = \langle a(f), a(f')\rangle - \langle b(f), b(f')\rangle + i\left[\langle a(f), b(f')\rangle + \langle b(f), a(f')\rangle\right] \tag{A23}$$

which in turn implies

$$\langle a(f), a(f')\rangle - \langle b(f), b(f')\rangle \stackrel{!}{=} \delta(f + f')\, S(f) \qquad \text{and} \tag{A24}$$

$$\langle a(f), b(f')\rangle + \langle b(f), a(f')\rangle \stackrel{!}{=} 0. \tag{A25}$$

Equations (A22) and (A25) then imply

$$\langle a(f), a(f')\rangle = \frac{1}{2}(\delta(f - f') + \delta(f + f'))S(f'), \tag{A26}$$

$$\langle b(f), b(f')\rangle = \frac{1}{2}(\delta(f - f') - \delta(f + f'))S(f') \quad \text{and} \tag{A27}$$

$$\langle a(f), b(f')\rangle = \langle b(f), a(f')\rangle = 0. \tag{A28}$$

It is sometimes more convenient to use $z(-f) = z^*(f)$ to express $h(t)$ as

$$h(t) = 2\int_0^{\infty} df\, \left[a(f)\cos(2\pi ft) - b(f)\sin(2\pi ft)\right]. \tag{A29}$$

In this case one only needs to consider $a(f)$ and $b(f)$ on $\mathbb{R}^+$, which simplifies the correlations to

$$\begin{aligned} \langle a(f), a(f')\rangle &= \langle b(f), b(f')\rangle = \delta(f - f')\, S(f') \\ \langle a(f), b(f')\rangle &= \langle b(f), a(f')\rangle = 0. \end{aligned} \tag{A30}$$

A.5 Discretization of Random Processes

We consider a random process $x(t)$. For numerical applications, one often needs to discretize t in a set of points $(t_1, \ldots, t_n)$. Evaluating x in t_i, then naturally creates a random vector $\mathbf{x} = (x_1, \ldots, x_n) = (x(t_1), \ldots, x(t_n))$. Given the covariance function $C(t, t')$ for $x(t)$ one obtains the corresponding covariance matrix for $\mathbf{x}$, C_{ij} as

$$C_{ij} = C(t_i, t_j). \tag{A31}$$

We consider the single sided case of appendix A.4, with the random process given as $z(f) = (a(f), b(f)),\ f \in \mathbb{R}^+$, where we have written the one dimensional complex process as a two dimensional real random process. The process is Gaussian by assumption and the covariance function for the 2 dimensional process is given as

$$C(f, f') = \delta(f - f')\, S(f')\, \mathbb{1}. \tag{A32}$$

Given that f is discretized in n equidistant points as $f_i = f_L + \Delta f i$, the resulting covariance matrix is given as

$$C_{ij} = \frac{\delta_{ij}}{\Delta f}\, S(f_i)\, \mathbb{1}, \tag{A33}$$

i.e. C_{ij} is diagonal, which allows to sample each $a(f_i)$ and $b(f_i)$ from a univariate Gaussian distribution with standard deviation $\sigma = \sqrt{\frac{S(f_i)}{\Delta f}}$. One can verify that $\frac{\delta_{ij}}{\Delta f}$ is the correct discretization of $\delta(f - f')$ by computing the discretized integral against a test function $g(f)$

$$g(f) = \int_{f_L}^{f_{\max}} g(f')\delta(f - f')\, df' \approx \sum_{i=1}^{n-1} \Delta f\, g(f_i) \frac{\delta_{ij}}{\Delta f} = g(f_j), \tag{A34}$$

where we have set $f_j = f$ and $f_{\max} = f_L + (n-1)\Delta f$.

A.6 Taylor Approximation of the Integral Bound

Consider the case of a differential equation of the form:

$$\frac{dx}{dt} = 1 + I \coloneqq 1 + \int_0^{1+I} g(t)\, dt = 1 + \int_0^1 g(t)\, dt + \int_1^{1+I} g(t)\, dt, \tag{A35}$$

where G be an antiderivative of g. From the mean value theorem of calculus, there is a $t' \in [1, 1 + I]$, such that

$$G(1 + I) - G(1) = g(t')\, I = \int_0^{1+I} \underbrace{g(t)g(t')}_{\mathcal{O}(g^2)}\, dt. \tag{A36}$$

It follows

$$\frac{dx}{dt} = 1 + \int_0^1 g(t)\, dt + \mathcal{O}(g^2).$$

B Tables

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-14.50	2.50	-14.50	2.50
Mean	-14.69	2.95	-14.98	3.49
Median	-14.65	2.89	-14.92	3.44
Max post.	-14.54	2.68	-14.85	3.43
Variance	0.06	0.38	0.16	0.8
68% lower b.	-14.85	2.24	-15.2	2.50
68% upper b.	-14.40	3.43	-14.55	4.18
95% lower b.	-15.2	1.78	-15.79	1.83
95% upper b.	-14.28	4.16	-14.3	5.44

Table A1: Statistical estimators for the first testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-14.50	4.50	-14.50	4.50
Mean	-14.39	4.20	-14.30	3.99
Median	-14.39	4.18	-14.30	4.00
Max post.	-14.38	4.12	-14.31	4.05
Variance	0.01	0.06	0.01	0.09
68% lower	-14.46	3.97	-14.41	3.7
68% upper	-14.31	4.45	-14.2	4.29
95% lower	-14.55	3.73	-14.51	3.38
95% upper	-14.24	4.72	-14.1	4.58

Table A2: Statistical estimators for the second testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.50	2.50	-15.50	2.50
Mean	-16.66	2.51	-16.75	3.16
Median	-16.8	2.17	-16.86	2.85
Max Post	-17.64	0.85	-17.32	2.27
Variance	0.82	3.05	0.69	2.33
68% lower	-18	0	-17.99	1.01
68% upper	-16.24	3.26	-16.36	3.83
95% lower	-18	0	-18	1
95% upper	-14.99	5.8	-15.24	5.98

Table A3: Statistical estimators for the third testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.50	4.50	-15.50	4.50
Mean	-16.24	6.04	-15.69	5.01
Median	-16.3	6.18	-15.65	5.04
Max Post	-16.42	6.81	-15.54	4.97
Variance	0.15	0.54	0.21	0.79
68% lower	-16.75	5.79	-16.09	4.18
68% upper	-16.01	7	-15.18	5.99
95% lower	-16.87	4.6	-16.64	3.35
95% upper	-15.45	7	-14.84	6.73

Table A4: Statistical estimators for the fourth testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.00	4.33	-15.00	4.33
Mean	-15.09	4.51	-15.16	4.66
Median	-15.07	4.47	-15.14	4.66
Max Post	-15.03	4.3	-15.12	4.65
Variance	0.05	0.25	0.06	0.31
68% lower	-15.25	3.94	-15.37	4.11
68% upper	-14.82	4.93	-14.89	5.16
95% lower	-15.56	3.61	-15.66	3.63
95% upper	-14.69	5.53	-14.68	5.8

Table A5: Statistical estimators for the fifth testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.00	4.33	-15.00	4.33
Mean	-14.8	3.87	-14.85	3.75
Median	-14.79	3.85	-14.8	3.7
Max Post	-14.76	3.84	-14.76	3.44
Variance	0.03	0.25	0.13	0.78
68% lower	-14.95	3.26	-15.08	2.78
68% upper	-14.62	4.23	-14.46	4.51
95% lower	-15.15	2.96	-15.56	2.15
95% upper	-14.46	4.91	-14.25	5.56

Table A6: Statistical estimators for the sixth testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.00	4.33	-15.00	4.33
Mean	-14.90	4.03	-15.00	4.25
Median	-14.87	4.02	-14.98	4.25
Max Post	-14.86	4.03	-14.94	4.28
Variance	0.05	0.26	0.08	0.39
68% lower	-15.04	3.50	-15.19	3.65
68% upper	-14.68	4.40	-14.72	4.80
95% lower	-15.32	3.04	-15.53	3.06
95% upper	-14.5	5.10	-14.51	5.55

Table A7: Statistical estimators for the seventh testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

Estimator	MCMC: $\log_{10} A_{GW}$	MCMC: γ_{GW}	NN: $\log_{10} A_{GW}$	NN: γ_{GW}
True Values	-15.00	4.33	-15.00	4.33
Mean	-14.80	3.97	-14.83	4.06
Median	-14.79	3.95	-14.83	4.07
Max Post	-14.74	3.89	-14.82	4.10
Variance	0.03	0.22	0.05	0.32
68% lower	-14.96	3.45	-15.01	3.51
68% upper	-14.62	4.39	-14.61	4.59
95% lower	-15.14	3.07	-15.27	2.94
95% upper	-14.49	4.90	-14.43	5.23

Table A8: Statistical estimators for the eighth testing noise series computed using MCMC and the NN. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

	NF			CFM		
Estimator	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$
True Values	1.00	-0.79	0.00	1.00	-0.79	0.00
Mean	0.78	-0.84	-0.03	0.77	-0.73	0.04
Median	0.80	-0.85	-0.02	0.82	-0.71	0.02
MaxPost	0.96	-0.84	-0.33	0.91	-0.54	-0.04
68% lower	0.71	-0.93	-0.03	0.65	-0.94	-0.11
68% upper	1.00	-0.76	0.00	1.06	-0.43	0.12
95% lower	0.48	-1.01	-0.08	0.32	-1.19	-0.14
95% upper	1.00	-0.66	0.00	1.10	-0.34	0.24

Table A9: Statistical estimators for the first testing noise series for the BPL model computed using NF and CFM diffusion network. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

	NF			CFM		
Estimator	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$
True Values	1.00	-0.79	0.00	1.00	-0.79	0.00
Mean	0.82	-0.79	-0.02	0.81	-0.68	0.08
Median	0.85	-0.79	-0.02	0.85	-0.67	0.06
MaxPost	0.94	-0.79	-0.01	0.96	-0.52	0.01
68% lower	0.77	-0.85	-0.03	0.69	-0.86	-0.06
68% upper	1.00	-0.73	0.00	1.08	-0.37	0.15
95% lower	0.53	-0.91	-0.07	0.42	-1.13	-0.10
95% upper	1.00	-0.68	0.00	1.14	-0.30	0.28

Table A10: Statistical estimators for the second testing noise series for the BPL model computed using NF and CFM diffusion network. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

	NF			CFM		
Estimator	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$
True Values	1.00	-0.79	0.00	1.00	-0.79	0.00
Mean	0.83	-0.81	-0.02	0.74	-0.97	-0.02
Median	0.85	-0.81	-0.01	0.79	-0.95	-0.03
MaxPost	0.96	-0.82	-0.01	0.93	-0.88	-0.10
68% lower	0.78	-0.90	-0.02	0.62	-1.19	-0.17
68% upper	1.00	-0.73	0.00	1.08	-0.69	0.06
95% lower	0.58	-0.97	-0.05	0.26	-1.42	-0.20
95% upper	1.00	-0.64	0.00	1.11	-0.58	0.18

Table A11: Statistical estimators for the third testing noise series for the BPL model computed using NF and CFM diffusion network. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

	NF			CFM		
Estimator	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$	$\log_{10}(\alpha_*)$	$\log_{10}(T_*)$	$\log_{10}(H_*R_*)$
True Values	1.00	-0.79	0.00	1.00	-0.79	0.00
Mean	0.78	-0.79	-0.04	0.78	-0.69	0.07
Median	0.80	-0.80	-0.03	0.82	-0.67	0.05
MaxPost	0.71	-0.81	-0.01	0.92	-0.62	-0.01
Variance	0.02	0.01	0.00	0.04	0.05	0.01
68% lower	0.71	-0.88	-0.05	0.68	-0.89	-0.09
68% upper	1.00	-0.71	0.00	1.06	-0.39	0.13
95% lower	0.49	-0.97	-0.10	0.38	-1.12	-0.11
95% upper	1.00	-0.62	0.00	1.11	-0.29	0.27

Table A12: Statistical estimators for the fourth testing noise series for the BPL model computed using NF and CFM diffusion network. All quantities are computed based on the one dimensional marginalized posterior distributions for each parameter. The maximum posterior values haven been computed using kernel density estimation.

References

[Acq23] Viviana Acquaviva. *Machine Learning for Physics and Astronomy*. Princeton University Press, 2023. ISBN: 978-0691206417. URL: https://press.princeton.edu/books/hardcover/9780691203928/machine-learning-for-physics-and-astronomy.

[Afz+23] Adeela Afzal et al. "The NANOGrav 15 yr Data Set: Search for Signals from New Physics". In: *The Astrophysical Journal Letters* 951.1 (June 2023), p. L11. ISSN: 2041-8213. DOI: 10.3847/2041-8213/acdc91. URL: http://dx.doi.org/10.3847/2041-8213/acdc91.

[Aga+23] Gabriella Agazie et al. "The NANOGrav 15 yr Data Set: Evidence for a Gravitational-wave Background". In: *The Astrophysical Journal Letters* 951.1 (June 2023), p. L8. ISSN: 2041-8213. DOI: 10.3847/2041-8213/acdac6. URL: http://dx.doi.org/10.3847/2041-8213/acdac6.

[Aga+25] Gabriella Agazie et al. *The NANOGrav 15 yr Data Set: Running of the Spectral Index*. 2025. arXiv: 2408.10166 [astro-ph.HE]. URL: https://arxiv.org/abs/2408.10166.

[al23] Yaron Lipman et al. *Flow Matching for Generative Modeling*. 2023. arXiv: 2210.02747 [cs.LG]. URL: https://arxiv.org/abs/2210.02747.

[Anh+09] Melissa Anholm et al. "Optimal strategies for gravitational wave stochastic background searches in pulsar timing data". In: *Physical Review D* 79.8 (Apr. 2009). ISSN: 1550-2368. DOI: 10.1103/physrevd.79.084030. URL: http://dx.doi.org/10.1103/PhysRevD.79.084030.

[Ant+23] J. Antoniadis et al. "The second data release from the European Pulsar Timing Array: I. The dataset and timing analysis". In: *Astronomy & Astrophysics* 678 (Oct. 2023), A48. ISSN: 1432-0746. DOI: 10.1051/0004-6361/202346841. URL: https://doi.org/10.1051/0004-6361/202346841.

[Arz+20] Zaven Arzoumanian et al. "The NANOGrav 12.5 yr Data Set: Search for an Isotropic Stochastic Gravitational-wave Background". In: *The Astrophysical Journal Letters* 905.2 (Dec. 2020), p. L34. ISSN: 2041-8213. DOI: 10.3847/2041-8213/abd401. URL: http://dx.doi.org/10.3847/2041-8213/abd401.

[Bal13] Robert S. Ball. *A Treatise on Spherical Astronomy*. eng. Cambridge library collection. Astronomy. Cambridge: Cambridge University Press, 2013. ISBN: 1-139-87729-1.

[BKK22] Cosimo Bambi, Stavros Katsanevas, and Konstantinos D. Kokkotas, eds. *Handbook of Gravitational Wave Astronomy*. Springer, 2022. ISBN: 978-981-16-4305-7, 978-981-16-4306-4, 978-981-15-4702-7. DOI: 10.1007/978-981-15-4702-7.

[Bol07] William M. Bolstad. *Introduction to Bayesian Statistics*. 2nd. Hoboken, NJ: John Wiley & Sons, 2007. ISBN: 978-0-471-72036-5.

[BZ20] Adrian Barbu and Song-Chun Zhu. *Monte Carlo Methods*. eng. 1st ed. 2020. Singapore: Springer Singapore, 2020. ISBN: 9789811329715.

[BZB02] Mark A Beaumont, Wenyang Zhang, and David J Balding. "Approximate Bayesian Computation in Population Genetics". In: *Genetics* 162.4 (Dec. 2002), pp. 2025–2035. ISSN: 1943-2631. DOI: 10.1093/genetics/162.4.2025. eprint: https://academic.oup.com/genetics/article-pdf/162/4/2025/42049447/genetics2025.pdf. URL: https://doi.org/10.1093/genetics/162.4.2025.

[Cap+20] Chiara Caprini et al. “Detecting gravitational waves from cosmological phase transitions with LISA: an update”. In: *Journal of Cosmology and Astroparticle Physics* 2020.03 (Mar. 2020), pp. 024–024. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2020/03/024. URL: http://dx.doi.org/10.1088/1475-7516/2020/03/024.

[CBL20] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* 117.48 (May 2020), pp. 30055–30062. ISSN: 1091-6490. DOI: 10.1073/pnas.1912789117. URL: http://dx.doi.org/10.1073/pnas.1912789117.

[Che21] Ricky T. Q. Chen. *torchdiffeq*. Version 0.2.2. June 2021. URL: https://github.com/rtqichen/torchdiffeq.

[Cor+25] Nina Cordes et al. “On the overlap reduction function of pulsar timing array searches for gravitational waves in modified gravity”. In: *Class. Quant. Grav.* 42.1 (2025), p. 015003. DOI: 10.1088/1361-6382/ad9881. arXiv: 2407.04464 [gr-qc].

[Cow02] Glen Cowan. *Statistical data analysis*. eng. Repr. Oxford science publications. Oxford: Clarendon Press, 2002. ISBN: 9780198501558.

[Dax+23] Maximilian Dax et al. “Neural Importance Sampling for Rapid and Reliable Gravitational-Wave Inference”. In: *Physical Review Letters* 130.17 (Apr. 2023). ISSN: 1079-7114. DOI: 10.1103/physrevlett.130.171403. URL: http://dx.doi.org/10.1103/PhysRevLett.130.171403.

[DP80] J.R. Dormand and P.J. Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19–26. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0771-050X(80)90013-3. URL: https://www.sciencedirect.com/science/article/pii/0771050X80900133.

[Dur+19] Conor Durkan et al. *Neural Spline Flows*. 2019. arXiv: 1906.04032 [stat.ML]. URL: https://arxiv.org/abs/1906.04032.

[EH17] Justin Ellis and Rutger van Haasteren. *jellis18/PTMCMCSampler: Official Release*. Oct. 2017. DOI: 10.5281/zenodo.1037579. URL: https://doi.org/10.5281/zenodo.1037579.

[Ein18] Albert Einstein. “Über Gravitationswellen”. In: *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften* (Jan. 1918), pp. 154–167.

[Gab+18] Hunter Gabbard et al. “Matching Matched Filtering with Deep Networks for Gravitational-Wave Astronomy”. In: *Phys. Rev. Lett.* 120 (14 Apr. 2018), p. 141103. DOI: 10.1103/PhysRevLett.120.141103. URL: https://link.aps.org/doi/10.1103/PhysRevLett.120.141103.

[GD82] J. A. Gregory and R. Delbourgo. “Piecewise Rational Quadratic Interpolation to Monotonic Data”. In: *IMA Journal of Numerical Analysis* 2.2 (Apr. 1982), pp. 123–130. ISSN: 0272-4979. DOI: 10.1093/imanum/2.2.123. eprint: https://academic.oup.com/imajna/article-pdf/2/2/123/2267745/2-2-123.pdf. URL: https://doi.org/10.1093/imanum/2.2.123.

[GH17] Daniel George and E. A. Huerta. *Deep Learning for Real-time Gravitational Wave Detection and Parameter Estimation with LIGO Data*. 2017. arXiv: 1711.07966 [gr-qc]. URL: https://arxiv.org/abs/1711.07966.

[GIK20] Christina Gao, Joshua Isaacson, and Claudius Krause. “<tt>i- flow</tt>: High-dimensional integration and sampling with normalizing flows”. In: *Machine Learning: Science and Technology* 1.4 (Nov. 2020), p. 045023. ISSN: 2632-2153. DOI: 10.1088/2632-2153/abab62. URL: http://dx.doi.org/10.1088/2632-2153/abab62.

[GRK21] Atul Goyal, Madhuchanda Rakshit, and Suchet Kumar. *Numerical methods*. eng. 1st ed. New Delhi, India: New India Publishing Agency NIPA, 2021. ISBN: 9789390512843.

[Haa+09] Rutger van Haasteren et al. "On measuring the gravitational-wave background using Pulsar Timing Arrays". In: *Monthly Notices of the Royal Astronomical Society* 395.2 (May 2009), pp. 1005–1014. ISSN: 1365-2966. DOI: 10.1111/j.1365-2966.2009.14590.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2009.14590.x.

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[Hei24] Theo Heimel. "The Flow of LHC Events - Generative models for LHC simulations and inference". PhD thesis. U. Heidelberg (main), 2024. DOI: 10.11588/heidok.00034758.

[Hel09] E. Hellinger. In: *Journal für die reine und angewandte Mathematik* 1909.136 (1909), pp. 210–271. DOI: doi:10.1515/crll.1909.136.210. URL: https://doi.org/10.1515/crll.1909.136.210.

[HEM06] G. B. Hobbs, R. T. Edwards, and R. N. Manchester. "tempo2, a new pulsar-timing package - I. An overview: tempo2, a new pulsar-timing package - I. Overview". In: *Monthly Notices of the Royal Astronomical Society* 369.2 (May 2006), pp. 655–672. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2006.10302.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2006.10302.x.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[HL12] Rutger van Haasteren and Yuri Levin. "Understanding and analysing time-correlated stochastic signals in pulsar timing". In: *Monthly Notices of the Royal Astronomical Society* 428.2 (Oct. 2012), pp. 1147–1159. ISSN: 0035-8711. DOI: 10.1093/mnras/sts097. URL: http://dx.doi.org/10.1093/mnras/sts097.

[HS06] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[Jia24] Chris Jia. *Gravitational Waves Through Time: Scientific Significance, Detection Techniques, and Recent Breakthroughs*. 2024. arXiv: 2312.16198 [gr-qc]. URL: https://arxiv.org/abs/2312.16198.

[Joh+24] Aaron D. Johnson et al. "NANOGrav 15-year gravitational-wave background methods". In: *Physical Review D* 109.10 (May 2024). ISSN: 2470-0029. DOI: 10.1103/physrevd.109.103012. URL: http://dx.doi.org/10.1103/PhysRevD.109.103012.

[JT17] Ryusuke Jinno and Masahiro Takimoto. "Gravitational waves from bubble collisions: An analytic derivation". In: *Phys. Rev. D* 95 (2 Jan. 2017), p. 024009. DOI: 10.1103/PhysRevD.95.024009. URL: https://link.aps.org/doi/10.1103/PhysRevD.95.024009.

[KAG] Institute for Cosmic Ray Research, University of Tokyo. *KAGRA Kamioka Gravitational Wave Detector*. Accessed: 2025-05-29. URL: https://gwcenter.icrr.u-tokyo.ac.jp/en/.

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[Kon92] Augustine Kong. *A Note on Importance Sampling using Standardized Weights*. Tech. rep. Technical Report 348. Department of Statistics, University of Chicago, 1992. URL: https://d3qi0qp55mx5f5.cloudfront.net/stat/docs/tech-rpts/tr348.pdf.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[LH17] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG]. URL: https://arxiv.org/abs/1608.03983.

[LIG] LIGO Scientific Collaboration. *LIGO Laboratory*. Accessed: 2025-05-29. URL: https://www.ligo.caltech.edu/.

[LISA] European Space Agency. *LISA: Laser Interferometer Space Antenna*. Accessed: 2025-05-29. URL: https://www.lisamission.org/.

[LL25] Junrong Lai and Changhong Li. *Accelerated Bayesian Inference for Pulsar Timing Arrays: Normalizing Flows for Rapid Model Comparison Across Stochastic Gravitational-Wave Background Sources*. 2025. arXiv: 2504.04211 [astro-ph.CO]. URL: https://arxiv.org/abs/2504.04211.

[LTH23] William G. Lamb, Stephen R. Taylor, and Rutger van Haasteren. "Rapid refitting techniques for Bayesian spectral characterization of the gravitational wave background using pulsar timing arrays". In: *Physical Review D* 108.10 (Nov. 2023). ISSN: 2470-0029. DOI: 10.1103/physrevd.108.103019. URL: http://dx.doi.org/10.1103/PhysRevD.108.103019.

[Mag07] Michele Maggiore. *Gravitational Waves. Vol. 1: Theory and Experiments*. Oxford University Press, 2007. ISBN: 978-0-19-171766-6, 978-0-19-852074-0. DOI: 10.1093/acprof:oso/9780198570745.001.0001.

[Mag18] Michele Maggiore. *Gravitational Waves. Vol. 2: Astrophysics and Cosmology*. Oxford University Press, Mar. 2018. ISBN: 978-0-19-857089-9.

[Man+05] R. N. Manchester et al. "The Australia Telescope National Facility Pulsar Catalogue". In: *The Astronomical Journal* 129.4 (Apr. 2005), pp. 1993–2006. DOI: 10.1086/428488. arXiv: astro-ph/0412641 [astro-ph].

[Mit+23] Andrea Mitridate et al. *PTArcade*. 2023. arXiv: 2306.16377 [hep-ph]. URL: https://arxiv.org/abs/2306.16377.

[Nobel17] The Nobel Prize Committee. *The Nobel Prize in Physics 2017: For decisive contributions to the LIGO detector and the observation of gravitational waves*. Accessed: 2025-05-12. 2017. URL: https://www.nobelprize.org/prizes/physics/2017/summary/.

[Nobel24] The Nobel Prize Committee. *The Nobel Prize in Physics 2024: For foundational discoveries and inventions that enable machine learning with artificial neural networks*. Accessed: 2025-05-12. 2024. URL: https://www.nobelprize.org/prizes/physics/2024/summary/.

[Pap+21] George Papamakarios et al. *Normalizing Flows for Probabilistic Modeling and Inference*. 2021. arXiv: 1912.02762 [stat.ML]. URL: https://arxiv.org/abs/1912.02762.

[Pas+19] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703.

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Per+19] B. B. P. Perera et al. "The International Pulsar Timing Array: second data release". In: *Monthly Notices of the Royal Astronomical Society* 490.4 (Oct. 2019), pp. 4666–4687. ISSN: 1365-2966. DOI: 10.1093/mnras/stz2857. URL: http://dx.doi.org/10.1093/mnras/stz2857.

[Phi01] E. S. Phinney. *A Practical Theorem on Gravitational Wave Backgrounds*. 2001. arXiv: astro-ph/0108028 [astro-ph]. URL: https://arxiv.org/abs/astro-ph/0108028.

[PM18] George Papamakarios and Iain Murray. *Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation*. 2018. arXiv: 1605.06376 [stat.ML]. URL: https://arxiv.org/abs/1605.06376.

[PP02] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables, and Stochastic Processes*. 4th ed. Boston: McGraw-Hill, 2002. ISBN: 0071226613.

[RC04] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. 2nd ed. Springer Texts in Statistics. New York: Springer, 2004. ISBN: 978-1-4419-1939-7. DOI: 10.1007/978-1-4757-4145-2.

[RC99] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. eng. 1st ed. 1999. Springer Texts in Statistics. New York, NY: Springer New York, 1999. ISBN: 1-4757-3071-3.

[Rea+23] Daniel J. Reardon et al. "Search for an Isotropic Gravitational-wave Background with the Parkes Pulsar Timing Array". In: *The Astrophysical Journal Letters* 951.1 (June 2023), p. L6. ISSN: 2041-8213. DOI: 10.3847/2041-8213/acdd02. URL: http://dx.doi.org/10.3847/2041-8213/acdd02.

[RHB02] K. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods for Physics and Engineering: A Comprehensive Guide*. 2nd ed. Cambridge University Press, 2002.

[RM16] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML]. URL: https://arxiv.org/abs/1505.05770.

[Rob16] Christian P. Robert. *The Metropolis-Hastings algorithm*. 2016. arXiv: 1504.01896 [stat.CO]. URL: https://arxiv.org/abs/1504.01896.

[Rud17] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG]. URL: https://arxiv.org/abs/1609.04747.

[Sci24] Scikit-learn developers. *QuantileTransformer scikit-learn documentation*. Accessed: 2025-05-05, 3:20 pm. 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html.

[Sco15] David W. Scott. *Multivariate density estimation*. eng. Second edition. Wiley Series in Probability and Statistics. Hoboken, New Jersey: Wiley, 2015 - 2015. ISBN: 9781118575536.

[Shi+23] David Shih et al. *Fast Parameter Inference on Pulsar Timing Arrays with Normalizing Flows*. 2023. arXiv: 2310.12209 [astro-ph.IM]. URL: https://arxiv.org/abs/2310.12209.

[Sri+23] Aman Srivastava et al. "Noise analysis of the Indian Pulsar Timing Array data release I". In: *Physical Review D* 108.2 (July 2023). ISSN: 2470-0029. DOI: 10.1103/physrevd.108.023008. URL: http://dx.doi.org/10.1103/PhysRevD.108.023008.

[SS20] Kenichi Saikawa and Satoshi Shirai. "Precise WIMP dark matter abundance and Standard Model thermodynamics". In: *Journal of Cosmology and Astroparticle Physics* 2020.08 (Aug. 2020), p. 011. DOI: 10.1088/1475-7516/2020/08/011. URL: https://dx.doi.org/10.1088/1475-7516/2020/08/011.

[SSB14] Haim Sak, Andrew Senior, and Françoise Beaufays. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014. arXiv: 1402.1128 [cs.NE]. URL: https://arxiv.org/abs/1402.1128.

[Tay14] Stephen R. Taylor. "Exploring the Cosmos with Gravitational Waves". PhD thesis. Cambridge U., Inst. of Astron., 2014.

[Tay21] Stephen R. Taylor. *The Nanohertz Gravitational Wave Astronomer*. 2021. arXiv: 2105.13270 [astro-ph.HE]. URL: https://arxiv.org/abs/2105.13270.

[Vas+23] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[VFJ19] Dootika Vats, James M Flegal, and Galin L Jones. "Multivariate output analysis for Markov chain Monte Carlo". In: *Biometrika* 106.2 (2019), pp. 321–337. DOI: 10.1093/biomet/asz002.

[Vir] Virgo Collaboration. *Virgo Interferometer*. Accessed: 2025-05-29. URL: https://www.virgo-gw.eu/.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[Web69] J. Weber. "Evidence for discovery of gravitational radiation". In: *Phys. Rev. Lett.* 22 (1969), pp. 1320–1324. DOI: 10.1103/PhysRevLett.22.1320.

[Xio+20] Ruibin Xiong et al. *On Layer Normalization in the Transformer Architecture*. 2020. arXiv: 2002.04745 [cs.LG]. URL: https://arxiv.org/abs/2002.04745.

[Xu+23] Heng Xu et al. "Searching for the Nano-Hertz Stochastic Gravitational Wave Background with the Chinese Pulsar Timing Array Data Release I". In: *Research in Astronomy and Astrophysics* 23.7 (June 2023), p. 075024. ISSN: 1674-4527. DOI: 10.1088/1674-4527/acdfa5. URL: http://dx.doi.org/10.1088/1674-4527/acdfa5.

[Zha+23] Aston Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023.

Declaration of Academic Integrity

I hereby confirm that this thesis, entitled _____

_____ is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited. I am aware that plagiarism is considered an act of deception which can result in sanction in accordance with the examination regulations.

_____
(date, signature of student)

I consent to having my thesis cross-checked with other texts to identify possible similarities and to having it stored in a database for this purpose.

I confirm that I have not submitted the following thesis in part or whole as an examination paper before.

_____
(date, signature of student)