GW-PINNs: Progress, Challenges, and Considerations

Arvind Chandrasekar

1 Introduction

I've modelled the orbital frequency (ω) and phase (ϕ) evolution of a supermassive black hole binary (SMBHB) system, withn integrated analytical post-Newtonian (PN) forms to provide physically consistent predictions over long temporal scales. All quantities are handled in SI units, and time in normalised in the PINN and expanded during output.

1.1 Assumptions & Values Chosen:

The current PINNs (for ω and ϕ) assume equal-mass ($M = 10^9 M_\odot$), spin-aligned (i.e. the non-precession case, with $\chi_A = 0$ and $\chi_S = 0.5$ (given $\chi_1 = \chi_2$ SMBHs in the binary system. Naturally, the symmetric mass ratio $\eta = 0.25$ given the equal mass assumption, and my ν is set to be 1, as per Dr. Subhajit's instructions.

1.2 Physical Model

There are 2 primary ODEs that govern my neural network (NN) and "inform" it of the physics of the problem. They are relativistic, given their 1PN η, 1.5PN χ_A & χ_S, and 2PN correction components (as included in equation 1).

$$\begin{aligned}\frac{d\omega}{dt} = \left(\frac{96}{5}\right) \omega^2 \, tN_\omega \Bigg[1 + \left(-\frac{743}{336} - \frac{11}{4}\eta\right) tm_\omega^{2/3} \, \nu \\ + Q_{15} \, tm_\omega \, \nu^{3/2} \\ + \left(\frac{34103}{18144} + \frac{13661}{2016}\eta + \frac{59}{18}\eta^2\right) tm_\omega^{4/3} \, \nu^2 \Bigg]\end{aligned} \tag{1}$$

with tN_ω and tm_ω being PN scaling factors, and Q_{15} encoding spin-aligned 1.5PN corrections. The orbital phase is then calculated from the below equality:

$$\frac{d\phi}{dt} = \omega(t), \tag{2}$$

2 PINN Architecture

2.1 Features

- **Hidden layers:** 4 - deeper networks observed to be better suited for this problem, with a 4-layered NN converging better its 3-layered counterpart)
- **Neurons per layer:** 64 - standard, medium width
- **Activation function:** tanh - smooth and differentiable, required for gradients in residual loss calculations
- **Output:** torch.exp(self.net(t)) - ensures reasonable positive values for both ω and ϕ
- **Epochs:** 4000 - balances convergence with runtime
- **Scaling:** $t \in [0, 1]$ - time normalised to improve numerical stability during training, given large timescale
- **Optimiser:** Adam optimiser - standard, used to combat stiff gradients
- **Gradient Clipping:** torch.nn.utils.clip - prevents exploding gradients
- **Double:** float64 - used for numerical accuracy
- **Collocation Points:** 10,000 - maintains accuracy with computational runtime in mind
- **Sampling:** random sampling - standard, unbiased
- **Consistent Randomness:** seed used for PyTorch and numpy to ensure reproducibility and "consistent randomness" across both libraries, for features including training/collocation points

Two distinct PINNs are implemented, the latter dependent on the first.

- **Orbital Frequency Network (ω-PINN)**: The network enforces $\omega = 0$ at $t = 0$, and is "informed" on Eq. 1. It also comprises a data loss term dependent on the MSE of its difference with respect to the analytical solution for ω.
- **Orbital Phase Network (ϕ-PINN)**: The network is trained using the predicted ω from the first PINN as a physical constraint, and enforces a similar $\phi = 0$ at $t = 0$.

2.2 Loss Function Design

The total loss is a weighted combination of three components in the (ω-PINN) and 2 components in the (ϕ-PINN):

1. **Initial/Boundary Condition Loss (BC):** Enforces $\omega(t_0) = \omega_a = 3.03 \times 10^{-9}$ rad/s and $\phi(t_0) = 0$ rad, ensuring the network begins with physically correct boundary conditions. Weight w_1 controls its contribution in both NNs: the condition is very strictly enforced (to compensate for the fact that there exists only one boundary condition), and so is weighted at $w_1 = 9 \times 10^{10}$ in (ω-PINN) and $w_1 = 1 \times 10^5$ in (ϕ-PINN).

2. **Residual Loss:** Forms the basis of the physics that governs then network, and encodes the respective ODE residuals for ω and ϕ. For ω:

$$\mathcal{L}_{\text{residual}} = \text{MSE}\left(\frac{d\omega_{\text{pred}}}{dt} - \text{RHS}(\omega_{\text{pred}})\right), \tag{3}$$

   and for ϕ:

$$\mathcal{L}_{\text{residual}} = \text{MSE}\left(\frac{d\phi_{\text{pred}}}{dt} - \omega_{\text{PINN}}\right). \tag{4}$$

   where MSE refers to the Mean-Squared Error. Weight w_2 is used for these physics constraints in each PINN, and is initialised at $w_2 = 7$ and $w_2 = 1$ in (ω-PINN) and (ϕ-PINN), respectively. It is important to note that in the ω-PINN, the residual is evaluated using the network's own prediction, ω_{pred}, also used in the ϕ-PINN residual to ensure consistency of ϕ_{pred} with the previously learned orbital frequency. Residuals are consistent in physical units to avoid scaling issues that were seen to cause flat, unphysical $\phi(t)$ evolution.

3. **Data Loss (only used in ω-PINN):** Encourages agreement with analytical PN solution. The corresponding weight w_3 being set to nearly zero ($w_3 = 10^{-14}$) does not significantly alter results, confirming that the BC and residual loss alone suffice to constrain the network - as also mentioned by Dr. Subhajit.

The total loss of the NNs are then calculated by the following equations:

$$\mathcal{L}_\omega = \lambda_1 \mathcal{L}_{\text{BC}} + \lambda_2 \mathcal{L}_{\text{residual}} + \lambda_3 \mathcal{L}_{\text{data}}, \tag{5}$$

$$\mathcal{L}_\phi = \lambda_1 \mathcal{L}_{\text{BC}} + \lambda_2 \mathcal{L}_{\text{residual}}, \tag{6}$$

where each numberedλ corresponds to the respective loss weights.

3 Training and Results

Over 10 years of evolution (in seconds),the ω-PINN converges to a physically reasonable shape (in accordance with the analytical plot), although the y-scale is noticeably off as seen in comparison between the below figures.

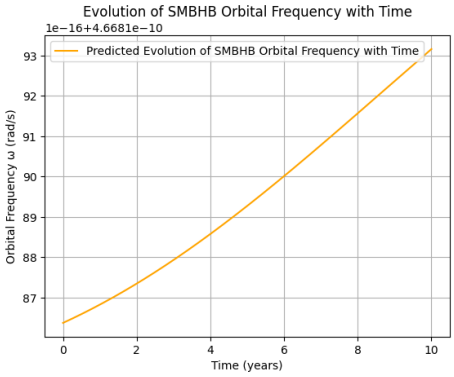

Figure 1: Evolution of Orbital Frequency Over 10 Years in ω-PINN

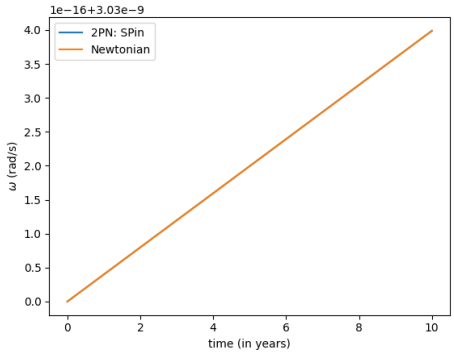

Figure 2: Analytic Evolution of Orbital Frequency Over 10 Years

The ϕ-PINN then essentially integrates this frequency (as per the relation in Equation 2) to produce the plot seen in Figure 3.

Loss evolution plots help confirm convergence and stability of the training process. High initial loss due to random initial weights rapidly decreases boundary conditions and physics residuals are enforced. We see this in Figures 4 and 5, which represent the loss evolution for ω-PINN and ϕ-PINN, respectively. We note the relatively rapid convergence of total loss to 0 (despite hiccups in ϕ-PINN's plot in Figure 5), likely due to the high magnitude of w_1 imposed on the boundary conditions of both NNs.

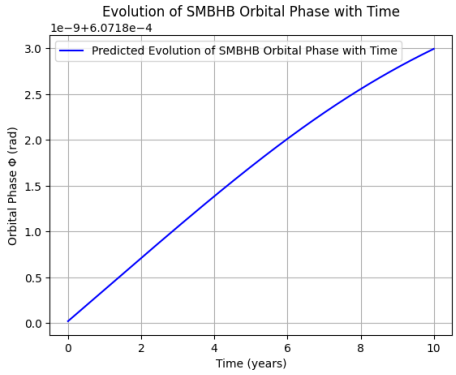

Figure 3: Evolution of Orbital Phase Over 10 Years in ϕ-PINN

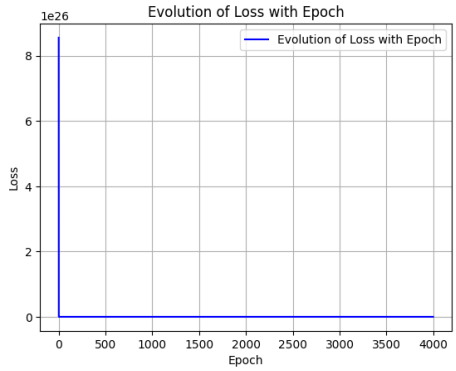

Figure 4: Evolution of Total Loss Over 4000 Epochs in ω-PINN

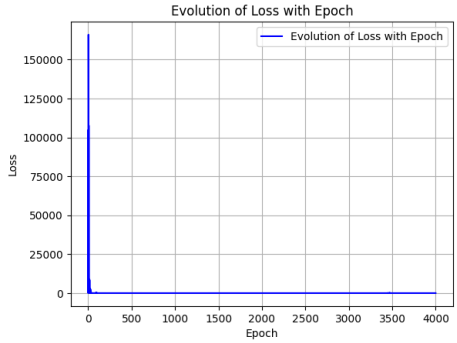

Figure 5: Evolution of Total Loss Over 4000 Epochs in ϕ-PINN

4 Trials, Errors, and Consequences

- I applied a StandardScaler trial approach, normalising time and orbital frequency, as per Dr. Subhajit's advice. However, this scaling reduced the

network's ability to capture extreme variations, so convergence improved slightly but the physical accuracy of the output suffered significantly.

- I also implemented a data loss term first in the ω-PINN, to trial its effectiveness. It proved to derail my final plot significantly, with a MSE loss of the order 10^{28}, which had to be compensated for using a low weight ($w_3 = 10^{-14}$. The same was seen in my ϕ-PINN (for which I used my RK4 model as the quasi-analytical ground truth to extract the data loss from), and though the y-scale was nearly reasonable, the graph shape drastically changed, tapering off towards the end. It is for this reason that both NNs are better without their respective data loss terms.

- The almost unreasonably rapid convergence of the network (in the absence of data loss) is evident in Figure 4 above, with total loss seeming to converge in less than 100 epochs - likely due to the huge BC weight I imposed. The modification of weights proved to improve the reasonability of convergence but compromised the PINN's accuracy, for which reason they've been chosen to remain at their particular values. Likewise, an increase in the number of collocation points during trials (from 10,000 up to values of 15,000) proved to be computationally troublesome and provided no further amelioration to my final plot accuracy either.

- I then proceeded to use data batching (on Dr. Subhajit's recommendation again) in order to improve accuracy and stochasticity in my network. While it a slight improvement to the output of my ω-PINN (the plot of which is shown in Figure 6 below), it drastically altered my ϕ-PINN's accuracy, outputting a plot with negative values in radians. Subsequently, implementing data batching only for my ω-PINN produces a more accurate plot scale for my ϕ-PINN (although still not exact), although the plot slightly tapers off towards the end - as seen in Figure 7.

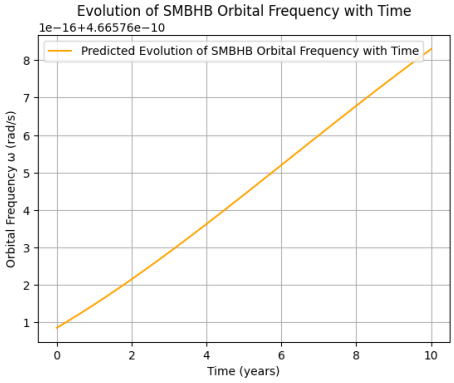

Figure 6: Evoltuion of SMBHB Orbital Frequency with Time with Data Batching in ω-PINN

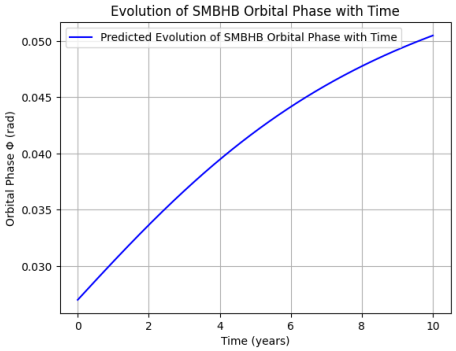

Figure 7: Evoltuion of SMBHB Orbital Phase with Time with Data Batching in ω-PINN

5 Runge-Kutta Implementation and Results

I also implemented a Runge-Kutta model to solve the Equation 1, which produced an identical solution to the analytic form (as per Dr. Subhajit's script), which can be seen below. We integrate the system from $t = 0$ to $t = 10$ years, with initial orbital frequency $\omega_0 = 3.03 \times 10^{-9}$ rad/s, corresponding to a typical SMBHB separation of 0.1 pc.

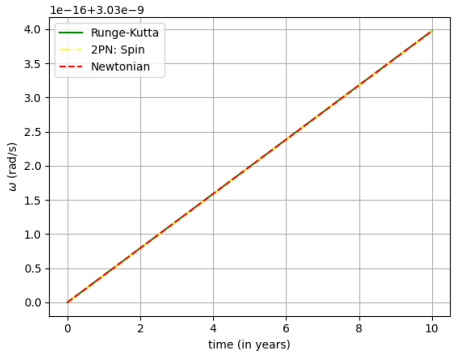

Figure 8: Evoltuion of SMBHB Orbital Frequency with Time in the Runge-Kutta Implementation vs its Analytical Form

I then implemented a second RK4 model to compute the evolution of ϕ with time, using an interpolator with my earlier computed discrete values of ω for weights. The plot is seen below, in Figure 9:

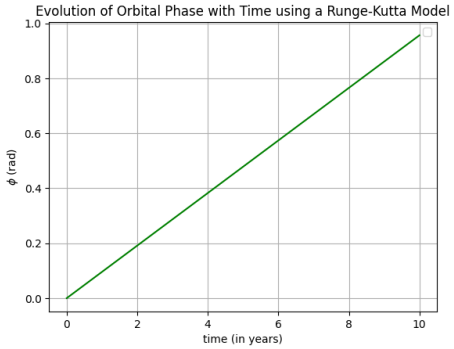

Figure 9: Evolution of SMBHB Orbital Phase with Time in the Runge-Kutta Implementation

6 Extensions and Progress

Given the context of the problem and my current situation, I'm currently spending my time on trying to debug my script to get an exact match to the analytical (and RK4) solutions for both PINNs. On the other hand, I am also willing to begin work on creating a random multi-dimensional array of input parameters, including spin, as per Dr. Subhajit's instructions - after which I'll be able to incorporate the pulsar term. I reckon I'll be able to do this within a week, but the only issue is that my current script isn't yet up to the mark in terms of accuracy. I'd appreciate advice on how to go forward from here.