# Using physics-informed neural networks to compute quasinormal modes

Alan S Cornell[*], Anele Ncube[†] and Gerhard Harmsen[‡]

*Department of Physics, University of Johannesburg, P.O. Box 524, Auckland Park 2006, South Africa*

In recent years there has been an increased interest in neural networks, particularly with regard to their ability to approximate partial differential equations. In this regard, research has begun on so-called physics-informed neural networks (PINNs) which incorporate into their loss function the boundary conditions of the functions they are attempting to approximate. In this paper, we investigate the viability of obtaining the quasinormal modes (QNMs) of nonrotating black holes in four-dimensional space-time using PINNs, and we find that it is achievable using a standard approach that is capable of solving eigenvalue problems (dubbed the eigenvalue solver here). In comparison to the QNMs obtained via more established methods (namely, the continued fraction method and the sixth-order Wentzel, Kramer, Brillouin method) the PINN computations share the same degree of accuracy as these counterparts. In other words, our PINN approximations had percentage deviations as low as $(\delta\omega_{\mathrm{Re}}, \delta\omega_{\mathrm{Im}}) = (<0.01\%, <0.01\%)$. In terms of the time taken to compute QNMs to this accuracy, however, the PINN approach falls short, leading to our conclusion that the method is currently not to be recommended when considering overall performance.

## I. INTRODUCTION

In recent years there has been an increased interest in the use of neural networks (NNs) as functional approximators [1–3]. The interest lies in the fact that NNs are versatile as demonstrated in their success in various applications such as natural language processing, image recognition and, more recently, scientific computing [4–6]. In scientific computing, they have been shown to be robust and data-efficient solvers of partial differential equations that govern diverse systems studied in mathematics, science, and engineering [4]. In general, NNs can be trained once and used in a variety of situations that are within the scope of the problem it was trained on. The advantage of applying a trained NN is that it expedites the computation of later solutions whereas, by contrast, more traditional numerical approximating methods would require an inefficient process beginning from scratch each time a solution is derived. Furthermore, NNs are also natively parallelizable, which adds to their higher computational efficiency compared to other numerical approximations.

A new technique has recently been developed to assist in creating NNs that can act as functional approximators, which takes inspiration from boundary type problems where the boundary conditions of the function are used to solve for the underlying function *as is*; namely, physics-informed neural networks (PINNs) [4,7]. In this regard, we are interested in determining if these types of NNs could be used to compute the quasinormal modes (QNMs) of black holes. The QNMs of black holes have been studied for many years and it is well known that they are correlated to the parameters of the black holes that generate them, and as such, they act as a *telltale* sign to probe the properties of black holes [8–10].

Over the years numerous techniques have been used to determine the numerical values of black holes using the radial equations that govern the perturbations of black holes [9]. Some examples are the Wentzel, Kramer, Brillouin (WKB) method, asymptotic iteration method (AIM), and the continued fraction method (CFM) [11–13]. Although all of these approaches have been successful in solving the radial equations of black holes to determine the numerical values of the QNMs, they do have computational limitations [8]. The WKB, in particular, becomes progressively difficult to apply when more accurate results are needed since achieving this requires painstaking derivations of higher-order approximations. In this work, we intend to show that PINNs can potentially supplement the extant techniques as a new alternative method for obtaining the black hole QNMs, with its unique advantages and limitations. Furthermore, we will compare the accuracy of PINNs to the already established methods and test their generalizability when applied to black hole perturbations equations.

Our motivation for using the equations of QNMs to test the usefulness of PINNs is that the equations that govern the QNMs are based on only a few parameters, namely a black hole's physical properties, and their boundary

[*]acornell@uj.ac.za
[†]ncubeanele4@gmail.com
[‡]gerhard.harmsen5@gmail.com

conditions are well defined for the system. As such, the boundary conditions act as a regularization mechanism that sufficiently limits the space of admissible solutions and contributes to the NN's stability [4]. Furthermore, in astrophysical circles, there has been an increased interest in black hole QMNs given the recent landmark detections of gravitational waves at the VIRGO and LIGO detectors [14,15].

The paper is set out in the following manner. In the next section, we describe the equations that govern the QNMs for various space-times. In Sec. III we present the currently accepted methods for determining the QNMs and proceed to touch on the new PINN approach in Sec. IV. Finally, in Secs. V and VI, we discuss the results obtained from applying PINNs and compare them to the QNMs obtained from the canonical methods.

## II. THE RADIAL PERTURBATION EQUATIONS OF BLACK HOLES

In this section, we will derive the equations required to determine the numerical value of QNMs beginning with the simplest space-times and then building up to more complex ones, which will eventually be encoded into the numerical methods. We begin with the Schwarzschild metric:

### A. The asymptotically flat Schwarzschild solution

We consider scalar-type perturbations (and later, electromagnetic, Dirac, and gravitational perturbations) then in order to derive the radial equations required to determine the QNMs we begin by considering the equation of motion, which is given by the Klein-Gordon equation [16,17]

$$\partial_\mu \partial^\mu \Phi + m^2 \Phi = \frac{1}{\sqrt{|g|}} \partial_\mu(\sqrt{|g|} g^{\mu\nu} \partial_\nu \Phi) + m^2 \Phi = 0, \quad (2.1)$$

where $\Phi$ is a scalar field with mass $m$ perturbing the black hole's space-time as given by the metric $g$. In the case of the Schwarzschild black hole, the metric is given as

$$ds^2 = -f dt^2 + \frac{1}{f} dr^2 + r^2(d\theta^2 + \sin^2\theta d\phi^2), \quad (2.2)$$

where $f = 1 - 2M/r$ is the metric function, with $M$ and $r$ representing the mass of the black hole and the radial distance from the center of the black hole, respectively. The last two terms on the right-hand side of this equation represent the metric of a 2-sphere [16]. As $r \to \infty$, we expect to recover a weak-field approximation of the metric wherein the components of the metric tensor can be decomposed into the flat Minkowski metric tensor $\eta_{\mu\nu}$ plus a small perturbation $|h_{\mu\nu}| \ll 1$; that is $g_{\mu\nu} \approx \eta_{\mu\nu} + h_{\mu\nu}$ [16].

Considering the massless form of the Klein-Gordon equation, where $m = 0$ in Eq. (2.1), and plugging in it the metric given in Eq. (2.2) we obtain

$$\frac{1}{\sqrt{|g|}} \partial_\mu(\sqrt{|g|} g^{\mu\nu} \partial_\nu \Phi)$$

$$= -\left(1 - \frac{2M}{r}\right)^{-1} \frac{\partial^2 \Phi}{\partial t^2}$$

$$+ \frac{1}{r^2} \frac{\partial}{\partial r}\left[r^2\left(1 - \frac{2M}{r}\right)\frac{\partial \Phi}{\partial r}\right]$$

$$+ \frac{1}{r^2 \sin\theta} \frac{\partial}{\partial \theta}\left[\sin\theta \frac{\partial \Phi}{\partial \theta}\right] + \frac{1}{r^2 \sin^2\theta} \frac{\partial^2 \Phi}{\partial \phi^2} = 0. \quad (2.3)$$

In this explicit form, we can derive the equation of massless scalar fields in the Schwarzschild background in terms of the radial coordinate $r$ via a separation of variables [18]. By mapping the resulting one-dimensional differential coordinate into an infinite domain given by a tortoise coordinate, $x$, we find [11,19]

$$\frac{d^2\psi}{dx^2} + \{\omega^2 - V(r)\}\psi = 0, \quad (2.4)$$

where

$$V(r) = \left(1 - \frac{2M}{r}\right)\left[\frac{\ell(\ell+1)}{r^2} + \frac{2M}{r^3}\right], \quad (2.5)$$

$$f(r) = \frac{dr}{dx} = \left(1 - \frac{2M}{r}\right)^{-1}. \quad (2.6)$$

Here $n$, $\ell$, and $m$ are the principal, multipole, azimuthal, and numbers, respectively [9]. The tortoise coordinate maps the location of the event horizon of the Schwarzschild black hole from $r = 2M$ (in geometric units) to $x = -\infty$. As such, it maps the space from a semi-infinite domain to an infinite one. Note that Eq. (2.4) is quite similar in form to the one-dimensional time-independent Schrödinger equation, but in this case, $V(r)$ is the effective potential for a which scalar field perturbs an asymptotically flat Schwarzschild metric [13,18]. The QNM frequencies, $\omega$, are complex-valued solutions to Eq. (2.4), which is a non-Hermitian problem, unlike the Schrödinger equation [10]. For asymptotically flat astrophysical black holes, the eigenfunctions, $\psi$, that solve this equation have asymptotic behavior governed by [13]

$$\psi(x) = \begin{cases} e^{-i\omega x}, & x \to -\infty \\ e^{+i\omega x}, & x \to +\infty \end{cases}. \quad (2.7)$$

Transforming from the infinite domain of the tortoise coordinate to the finite domain of a new variable, $\xi = 1 - 2M/r$, it can be shown that Eq. (2.4) takes the form [13]

$$\chi'' = \lambda_0(\xi)\chi' + s_0(\xi)\chi, \quad (2.8)$$

where

$$\lambda_0(\xi) = \frac{4Mi\omega(2\xi^2 - 4\xi + 1) - (1 - 3\xi)(1 - \xi)}{\xi(1 - \xi)^2},$$

(2.9)

$$s_0(\xi) = \frac{16M^2\omega^2(\xi - 2) - 8Mi\omega(1 - \xi) - \ell(\ell + 1) + (1 - s^2)(1 - \xi)}{\xi(1 - \xi)^2},$$

(2.10)

and $\chi(\xi)$ is a complex-valued scale factor. The boundary conditions have been incorporated into Eq. (2.8). The importance of this transformation is that it maps the domain from one that is infinite, i.e., $-\infty < x < +\infty$ to one that is finite, i.e., $0 \leq \xi < 1$. As such, it is now possible to numerically solve the perturbation equation since the domain is now finite and the QNM boundary conditions are implicitly accounted for.

For electromagnetic field perturbations of Schwarzschild black holes, the same Schrödinger-like radial equations are obtained by following the same procedure for deriving the massless scalar fields. However, in this case, the equation of motion considered is the source-free Gauss-Ampère law of Maxwell's equations [16,20],

$$F^{\mu\nu}_{;\nu} = \frac{1}{\sqrt{|g|}} \partial_\nu(\sqrt{|g|}F^{\mu\nu}) = 0,$$

(2.11)

where $F^{\mu\nu}$ is the electromagnetic field tensor. Applying the components of the electromagnetic field tensor $F^{\mu\nu}$, we can determine the radial perturbation equation from Maxwell's equations:

$$\frac{\partial}{\partial t}F^{\mu t} + \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 F^{\mu r}) + \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta F^{\mu\theta}) + \frac{\partial}{\partial\phi}F^{\mu\theta} = 0.$$

(2.12)

We can simplify the equations with indices $\mu = \theta$ and $\mu = \phi$ to obtain the Schrödinger-like perturbation equations. In short, we arrive at

$$-\frac{\partial^2 a_0(t, r)}{\partial t^2} + f^2 \frac{\partial^2 a_0(t, r)}{\partial r^2} - \frac{f\ell(\ell + 1)}{r^2}a_0(t, r) = 0,$$

(2.13)

where $a_0(t, r)$ represents the electromagnetic field perturbations. Thus, if we have $a_0(t, r) = a_0(r)e^{i\omega t}$, converting to tortoise coordinates we retrieve Eq. (2.4), where $\psi(r) = a_0(r)$ and $V(r) = \ell(\ell + 1)f(r)/r^2$ is the effective potential of an asymptotically Schwarzschild black hole perturbed by an electromagnetic field. For gravitational perturbations, the equations have the same form except for the effective potential $V(r)$. References [20,21] outline concisely the steps for arriving at the wave equations for these direct metric perturbations on a Schwarzschild black hole.

### B. The Schwarzschild (anti-)de Sitter solution

We shall also consider asymptotically curved space-times that are solutions to Einstein's equations with a nonzero cosmological constant. The cosmological constant, denoted by $\Lambda$, encodes the curvature of space-time via the relation $\Lambda = \pm 3/a^2$, where $a$ is the cosmological radius [22,23]. The metric in this case is

$$ds^2 = -\left(1 - \frac{r_s}{r} - \frac{\Lambda r^2}{3}\right)dt^2$$
$$+ \left(1 - \frac{r_s}{r} - \frac{\Lambda r^2}{3}\right)^{-1}dr^2 + r^2 d\Omega^2.$$

(2.14)

With this metric as a starting point, the radial perturbation equation derived for a four-dimensional (anti-)de Sitter Schwarzschild black hole is the same form as Eq. (2.4) but, with a more general effective potential given as [13]

$$V(r) = f(r)\left[\frac{\ell(\ell + 1)}{r^2} + (1 - s^2)\left(\frac{2M}{r^3} - \frac{(4 - s^2)\Lambda}{6}\right)\right],$$

(2.15)

where $f(r) = 1 - 2M/r - (\Lambda r^2)/3$ is the metric function for (anti-)de Sitter Schwarzschild space-times and $s = 0, 1/2, 1$ and $2$ denote the spins of scalar, Dirac, electromagnetic, and gravitational fields, respectively.

### C. Near extremal Schwarzschild and Reissner-Nordström-de Sitter solutions

A final case we shall consider are Reissner-Nordström-de Sitter black holes, albeit in the near extremal case. The metric of a Reissner-Nordström-de Sitter black hole is [22]

$$ds^2 = -\left(1 - \frac{r_s}{r} + \frac{r_Q^2}{r^2} - \frac{\Lambda r^2}{3}\right)dt^2$$
$$+ \left(1 - \frac{r_s}{r} + \frac{r_Q^2}{r^2} - \frac{\Lambda r^2}{3}\right)^{-1}dr^2 + r^2 d\Omega^2,$$

(2.16)

where $r_s = 2M$ (the Schwarzschild radius) and $r_Q^2 = Q^2/4\pi\epsilon_0$. Generally, when solving radial perturbation

equations, the nature of the effective potentials preclude applying a direct, analytical approach to deriving exact QNMs [11]. However, in special cases, such as this one involving near extremal Schwarzschild and Reissner-Nordström-de Sitter black holes, the effective potentials can be transformed to yield differential equations with known analytic solutions [9].

To obtain the effective potentials of nonrotating black holes in the near extremal (anti-)de Sitter case, we consider the relevant metric function, $f(r) = 1 - 2M/r - \Lambda r^2/3$. The solutions to $f(r) = 0$ are $r_b$ and $r_c$, which are the black hole's event horizon and the space-time's cosmological radius, respectively (where $r_c > r_b$). For $r_0 = -(r_b + r_c)$, the metric function can be given as [23]

$$f(r) = \frac{1}{a^2 r}(r - r_b)(r_c - r)(r - r_0), \qquad (2.17)$$

where $a^2 = r_b^2 + r_b r_c + r_c^2$ and $2Ma^2 = r_b r_c (r_b + r_c)$. The surface gravity, $\kappa$, associated with the black hole event horizon $r = r_b$ is defined as [23]

$$\kappa = \frac{1}{2}\frac{df}{dr}\bigg|_{r=r_b} = \frac{(r_c - r_b)(r_b - r_0)}{2a^2 r_b}. \qquad (2.18)$$

In the near extremal de Sitter case, the cosmological horizon $r_c$ of the space-time is very close (in the coordinate $r$) to the black hole horizon $r_b$ so that $(r_c - r_b)/r_b \ll 1$, and the following approximations apply [23]:

$$r_0 \sim -2r_b^2; \qquad a^2 \sim 3r_b^2; \qquad M \sim \frac{r_b}{3}; \qquad \kappa \sim \frac{r_c - r_b}{2r_b^2}. \qquad (2.19)$$

Also since the domain of $r$ is within $(r_b, r_c)$ and $r_b \sim r_c$, we find that $r - r_0 \sim r_b - r_0 \sim 3r_0$. In turn the metric function Eq. (2.17) becomes

$$f \sim \frac{(r - r_b)(r_c - r)}{r_b^2}. \qquad (2.20)$$

With this new form of the metric, the relation between the tortoise coordinate and the radial coordinate (2.6) reduces to

$$r = \frac{r_c e^{2\kappa x} + r_b}{1 + e^{2\kappa x}}. \qquad (2.21)$$

Substituting this expression for $r$ into the $f(r)$ equation (2.17), we find the expression for $f(x)$ as [23]

$$f(x) = \frac{(r_c - r_b)^2}{4r_b^2 \cosh^2(\kappa x)}. \qquad (2.22)$$

With this metric function, the effective potential of a near extremal Schwarzschild-de Sitter black hole is an inverted Pöschl-Teller potential [23]:

$$V(x) = \frac{V_0}{\cosh^2(\kappa x)}, \qquad (2.23)$$

where $V_0 = \kappa^2 \ell(\ell + 1)$ for massless scalar and electromagnetic perturbations and $V_0 = \kappa^2(\ell + 2)(\ell - 1)$ for gravitational perturbations. With the effective potential in this form, the perturbation equation (2.4) can now be solved analytically to derive the QNMs of near-extremal Schwarzschild-de Sitter black holes.

For astrophysical near-extremal de Sitter black holes, the asymptotic behavior of the solution is similar to that of an asymptotically flat Schwarzschild black hole equation (2.7); considering that they force the solution near the event horizon (cosmological horizon) not to generate outgoing (incoming) waves.

Considering the boundary conditions for astrophysical black holes, Eq. (2.7), Ref. [24] determined the analytic expressions of the QNM eigenfunctions and eigenfrequencies [22–24] as

$$\psi(x) = [\xi(\xi - 1)]^{i\omega/2\kappa} \cdot {}_2F_1$$
$$\times \left(1 + \beta + i\frac{\omega}{\kappa}, -\beta + i\frac{\omega}{\kappa}; 1 + i\frac{\omega}{\kappa}; \xi\right), \qquad (2.24)$$

$$\frac{\omega}{\kappa} = \sqrt{\left(\ell(\ell + 1) - \frac{1}{4}\right)} - i\left(n + \frac{1}{2}\right), \qquad n = \mathbb{Z}_0^+, \qquad (2.25)$$

where $\xi^{-1} = 1 + \exp(-2\kappa x)$ and $\beta = -1/2 + (1/4 - V_0/\kappa^2)^{1/2}$.

Extending from near extremal Schwarzschild-de Sitter black holes, Ref. [22] showed that an inverted Pöschl-Teller potential can also be used to represent the effective potential of Reissner-Nordström black holes perturbed by scalar fields. This is due to the fact that for any de Sitter black hole in the near extremal limit, the metric function $f(r)$ is given as [22]

$$f(r(x)) = \frac{(r_2 - r_1)\kappa_1}{2\cosh^2 \kappa_1 x} + \mathcal{O}(\delta^3), \qquad (2.26)$$

where $\delta = (r_2 - r_1)/r_1$, $\kappa_1$ is the surface gravity at the horizon, $r_1$ and $r_2$ are two consecutive positive roots of $f(r)$, and $x$ is the tortoise coordinate whose domain lies within $(r_1, r_2)$.

For both Schwarzschild and Reissner-Nordström-de Sitter cases, the terms $r_1$ and $r_2$ are the event and cosmological horizons, respectively, with $r_2 > r_1$. In the near extremal limit where $r_2 \sim r_1$, the metric function for a near-extremal Reissner-Nordström-de Sitter black hole would take the same form as Eq. (2.22). Therefore, when considering the near extremal limit, nonrotating black holes share the same mathematical expression for the metric function, which in turn results in the same expression for the effective potential, Eq. (2.23). From that, we can infer that the corresponding analytic expressions for QNMs of a near extremal Reissner-Nordström-de Sitter black holes are the same as for Schwarzschild-de Sitter black holes as given by Eqs. (2.24)–(2.25).

## III. ESTABLISHED NUMERICAL METHODS FOR DETERMINING QNMs

The perturbation equations of near extremal nonrotating black holes are among a few known cases with exact QNMs as solutions. As we shall use these equations to measure the accuracy of the PINN approach applied in the context of QNMs. More generally, though, the radial perturbations equations of Schwarzschild and Reissner-Nordström black hole perturbations are difficult to solve analytically, though not as challenging as cases involving rotating black holes, which require a more arduous investigation. Therefore, approximation techniques have been employed in the past to determine QNMs. We outline here a few prominent techniques used in the literature on black hole QNMs.

### A. Ferrari and Mashhoon approach

Reference [24] showed the connection between the QNMs of black holes and the bound states of inverted black hole effective potentials. The effective potential, denoted by $U$ in Ref. [24] is parametrized by some constant $p$ and is invariant under the transformations $p \to p' = \Pi(p)$ and $x \to -ix$, as in

$$U(-ix; p') = U(x; p). \quad (3.1)$$

By considering $x \to -ix$, the Schrödinger-like perturbation equation (2.4) transforms to

$$\frac{d^2\phi}{dx^2} + (-\Omega^2 + U)\phi = 0, \quad (3.2)$$

where $\phi(x; p) = \psi(-ix; p')$ and $\Omega(p) = \omega(p')$. The QNM boundary conditions then become

$$\phi \to \exp(\mp\Omega x), \quad \text{as } x \to \pm\infty. \quad (3.3)$$

In this new form, the problem has become a bound state problem with the original black hole effective potential inverted to $-U$. The transformed boundary conditions, Eq. (3.3), now correspond to vanishing states at both infinities as expected for bound state problems. After solving this problem to find $\Omega$ and $\phi$, the corresponding QNMs can then be found using inverse transformations:

$$\omega(p) = \Omega(\Pi^{-1}(p)), \qquad \psi(x; p) = \phi(ix; \Pi^{-1}(p)). \quad (3.4)$$

The values of $\omega$, that are determined from the bound states $\Omega$, are known as proper QNMs. Reference [24] demonstrated this approach using an inverted Pöschl-Teller potential to approximate the effective potential of a Schwarzschild black hole. The former was used because the bound states of a Pöschl-Teller potential are well known and could then provide approximate analytic formulas for the QNMs of the Schwarzschild black hole [24].

### B. WKB Method

The WKB method is a semi-analytic technique that has been used to approximately solve the radial equation of black hole perturbations since 1985, as first proposed by Schutz and Will [25], where they computed the QNMs of an asymptotically flat Schwarzschild black hole. It had already been established as an approximating technique for solving the time-independent Schrödinger equation.

### C. Continued fraction method

In a 1985 paper [12], Leaver put forward the method of continued fractions (previously used to compute the electronic spectra of the hydrogen molecule ion) to compute the QNM spectra of both stationary and rotating black holes. Overall, this approach was found to be very accurate for higher-order $n$ modes, especially after the improvement made by Nollert [26]. It has been used in the context of Schwarzschild, Kerr, and Reissner-Nordström black holes [8,12,27].

### D. Asymptotic iteration method

The AIM is another semi-analytic technique for solving black hole perturbations. In the context of black hole QNMs, this approach was developed by Ref. [13] who made improvements to a more traditional algorithm to make it markedly more efficient. In Ref. [13] the improved AIM was used to compute of QNMs for cases involving (A) dS, Reissner-Nordström and Kerr black holes. In later research, it was used to calculate QNMs of general dimensional and nonsingular Schwarzschild black holes [28,29]. Compared to other extant approximation techniques, the improved AIM was shown to be as accurate as Leaver's CFM [13].

## IV. PHYSICS-INFORMED NEURAL NETWORKS

As briefly recapped above, there are several techniques that already exist for solving radial equations in order to obtain the QNMs of black holes. To supplement them, we now introduce PINNs as an alternative to these methods.

Firstly, we introduce the idea of deep neural networks and how they can act as universal function approximators. We then introduce PINNs and how they can be used to solve ordinary differential equations (ODEs) and partial differential equations (PDEs).

## A. Deep neural networks

Deep neural networks are a system of interconnected computational nodes loosely based on biological neural networks and, mathematically, can be formulated as compositional functions [6,30]. In contrast to shallow neural networks, which are networks with just a single hidden layer, these NNs are composed of two or more hidden layers [3]. In many applications, the latter are favored because they are capable of replicating the complexity of functions and, at the same time, generalize well to unseen data better than shallow models [31].

Of several available types of structures (or architectures) of deep neural networks, the simplest and most common one is the feed-forward neural network (FNN).

*Definition 4.1.*—The FNN is comprised of neurons that hold single numerical values (called activations) combine to form a NN $\mathcal{N}^L(\mathbf{x})$ that is a series of $L$ layers with $N_\ell$ neurons in the $\ell$th layer. There are $L-1$ hidden layers, $N_0$ number of neurons in the input layer ($\ell = 0$) and $N_L$ number of neurons in the output layer ($\ell = L$). The transformations combining the neurons in the $(\ell - 1)$th layer to those in the $\ell$th layer are weight matrices and bias vectors $\mathbf{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $\mathbf{b}^\ell \in \mathbb{R}^{N_\ell}$, respectively. With these transformations, a FNN is generally structured as follows [30]:

input layer: $\mathcal{N}^0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{N_0}$,

hidden layers: $\mathcal{N}^\ell(\mathbf{x}) = \sigma(\mathbf{W}^\ell \mathcal{N}^{\ell-1}(\mathbf{x}) + \mathbf{b}^\ell) \in \mathbb{R}^{N_\ell}$,

for $1 \leq \ell \leq L - 1$,

output layers: $\mathcal{N}^L(\mathbf{x}) = \sigma(\mathbf{W}^\ell \mathcal{N}^{L-1}(\mathbf{x}) + \mathbf{b}^L) \in \mathbb{R}^{N_L}$,

where $\sigma$ denotes nonlinear activation functions that operate on $\mathbf{W}^\ell \mathcal{N}^{\ell-1}(\mathbf{x}) + \mathbf{b}^\ell$ element-wise. Examples of frequently used activation functions are the hyperbolic tangent (tanh) and the logistic sigmoid $1/(1 + e^{-x})$. Given that these are nonlinear functions, this makes values at each of the output nodes nonlinear combinations of the values at the nodes in the hidden and input layers [32].

Key seminal research on NNs, such as Refs. [33–35], has shown that deep neural networks are universal function approximators. That is to say, when NNs have a sufficient number of neurons they can approximate any function and its partial derivatives [30], though in practice this is constrained by the limit in the size of NNs that can be set up before they lead to overfitting. In such cases, the NN model gives the illusion of a good model that captures the underlying pattern in data, while a true test of its accuracy by means of exposing it to an unseen test dataset reveals a fallible model that gives poor predictions and a high generalization error [3,30]. In general, training deep NNs entails minimizing a loss function that measures the deviation of its approximations from the expected solutions. Analogous to linear least squares regression, the loss function is minimized via tuning of the many parameters in the deep neural network (which are the elements of its weight matrices and bias vectors) with the effect of steering their approximations closer to the target functions.

Mathematically, the weights and biases are tuned according to the equations

$$w_{jk}^\ell \rightarrow w_{jk}^\ell - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial w_{jk}^\ell}, \tag{4.1}$$

$$b_j^\ell \rightarrow b_j^\ell - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial b_j^\ell}, \tag{4.2}$$

where $C_x$ is the loss function of the FNN computed for a single training example $x$ that is taken from a mini batch of $m$ training examples, which in turn are taken from a training dataset with $n$ samples. These equations govern stochastic gradient descent optimization, an algorithm that entails randomly selecting different mini batches from the training dataset of $n$ examples until all of them are exhausted (this constitutes one epoch of training). In Eqs. (4.1) and (4.2), $\eta$ is a small, positive parameter known as the learning rate. Ultimately, the Adam optimizer is employed in our investigation of PINNs. It is a standard optimization algorithm that extends from classical methods of stochastic gradient descent [3,36].

## B. Physics-informed neural networks

Inspired by deep neural networks, PINNs follow the same modus operandi as traditional NNs. Similar to traditional NNs, PINNs are trained through gradient-descent optimization, whereby the partial derivatives of the loss function (with respect to the network's weight and biases) are minimized by tuning the weights and biases of the FNN. However, the difference is in the constraints that are embedded within the loss function of the PINNs which enable them to solve PDEs. These constraints are the PDEs themselves (or the governing equations) and their associated initial/boundary conditions [37].

Autodiff is a technique that is used in PINNs to compute the partial derivatives of the NN approximations and thus embed the governing PDEs and associated boundary conditions in the loss function. Given that it facilitates "meshless" numerical computations of derivatives, it endows PINNs with several advantages over traditional numerical discretization approaches for solving PDEs (such as the finite difference and finite element methods) that can be computationally expensive due to complex mesh generation [6,30,38].

For example, Refs. [6,30] demonstrated the advantage of applying NN-aided techniques over using traditional

mesh-based techniques to approximate solutions with steep gradients. The latter give rise to unphysical oscillations when the meshes have low resolution, hence higher resolutions are required to remove these undesirable oscillations, which can be prohibitively expensive and lead to excessive execution times [6]. Remarkably, the same level of accuracy that is achieved by higher resolution meshes (in mesh-based schemes) can be achieved more efficiently in PINNs. In such cases, PINNs could be a viable alternative for solving PDEs.

It is worth noting that derivatives of Padé approximations can be utilized as an alternative to PINNs and autodiff, which is a part of the PINN algorithm. They have indeed been applied in extensions of the WKB method in computing black hole QNMs [28]; therefore, the focus has been to compare their performance (and that of other established approaches in black hole QNMs) with the novel PINNs in this physical context.

The basic structure of PINNs can be divided into two components [30,38]:

 (i) A deep neural network with a particular architecture, such as a FNN. It represents the NN approximation of the PDE's solution [Fig. 1 (left)].

 (ii) A *loss function* that measures the deviation of the FNN solution from the physical constraints of the problem [Fig. 1 (right)]. The NN learns the solution of the PDE through gradient-based optimization, an algorithm that minimizes the loss function through an iterative tuning of the weights and biases in the deep neural network.

In general, to expand on the methodology, PINNs solve PDEs that are parametrized by $\lambda$, satisfied by a dependent variable [39] $u(\mathbf{x})$, and are expressed generally as [30]

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_d \partial x_d}; \lambda\right) = 0 \text{ on } \Omega,$$

(4.3)

where $\mathbf{x} = (x_1, \ldots, x_d)$ defined on a domain $\Omega \subset \mathbb{R}^d$. Along with a given PDE are its boundary conditions:

$$\mathcal{B}(u, \mathbf{x}) = 0 \text{ on } \partial\Omega,$$

(4.4)

where $\mathcal{B}(u, \mathbf{x})$ stands for Dirichlet, Neumann Robin or periodic boundary conditions. Note that both steady-state and dynamic systems can be solved using PINNs; where, for the latter, time $t$ is considered to be special component of $\mathbf{x}$ and $\Omega$ contains the time domain. As such, initial conditions are treated as a type of Dirichlet boundary condition on the spatiotemporal domain [30].

*Remark 4.1.*—It is worth noting the special nature of PINNs compared to traditional NNs, particularly in the case of classification and regression problems. While many FNNs are typically data-driven and highly dependent on labeled datasets, PINNs, however, are suitable within the scant data regime provided the physical laws governing a system are known [38]. In fact, PINNs are unsupervised and learn from based purely on the PDEs and boundary conditions in the case of forward problems and eigenvalue problems. The goal of forward problems is to find the dependent variable $u(\mathbf{x})$ for every $\mathbf{x}$ provided $\lambda$ are known parameters. Eigenvalue problems are more challenging because both $u(\mathbf{x})$ and $\lambda$ are unknown. In the case when PINNs utilize a labeled dataset these are inverse problems, where the goal is to determine $\lambda$ given a dataset (which can be small) of $u(\mathbf{x})$ at given points $\mathbf{x} \subset \Omega$.

### 1. The PINN algorithm for solving PDEs

PINNs follow these steps when solving forward, inverse and eigenvalue problems [30]:

 (1) Build a neural network $\hat{u}(\mathbf{x}; \theta)$ with parameters $\theta$: The neural network $\hat{u}(\mathbf{x}; \theta)$ takes in $\mathbf{x}$ as input and is a surrogate of the function $u(\mathbf{x})$ that satisfies the governing PDE and boundary/initial conditions.



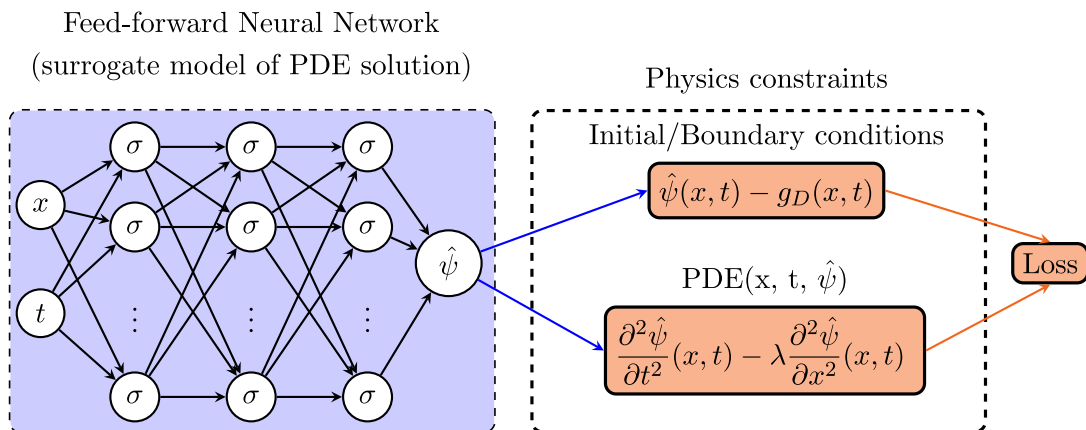FIG. 1. A schematic of a typical PINN. For the sake of illustration, a well-known hyperbolic type PDE (one-dimensional wave equation), $\partial^2\psi/\partial t^2 - \lambda\partial^2\psi/\partial x^2 = 0$, and initial/boundary conditions $\psi = g_D(x, t)$ are embedded in the loss function.

Whereas, $\theta = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \le \ell \le L}$ is a set of all weight matrices and vectors in the neural network [30].

(2) Specify the training dataset $\mathcal{T}$: In the case of forward and eigenvalue problems, we specify a dataset of "unlabeled" randomly distributed points in the domain (also known as residual points). The points within the $\mathcal{T}_f \subset \Omega$ are used to restrict the NN approximation $\hat{u}(\mathbf{x}; \theta)$ to satisfy the physics imposed by the PDE. Similarly, the boundary points of the spatiotemporal domain $\mathcal{T}_b \subset \partial\Omega$ are used to restrict the NN to satisfy the physics represented by the initial/boundary conditions. For inverse problems, since $\lambda$ is missing from the PDE, a "labeled" dataset of $u(\mathbf{x})$, denoted by $\mathcal{T}_o$, is required in addition.

(3) Specify a loss function by adding the weighted Euclidean norm of the PDE, boundary conditions, and other regularization functions: In general, the loss function of PINNs will be given as [30]

$$\mathcal{L}(\theta; \mathcal{T}) = w_f \mathcal{L}_f(\theta; \mathcal{T}_f) + w_b \mathcal{L}_b(\theta; \mathcal{T}_b) + w_r \mathcal{L}_r(\theta; \mathcal{T}_r), \tag{4.5}$$

where $w_f$, $w_b$, $w_r$ are weights and still $\theta = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \le \ell \le L}$, with $\ell$ specifying a hidden layer as defined in Sec. IV A [5]. Additionally $\mathcal{L}_f$ and $\mathcal{L}_b$ are loss terms due to the PDE and initial/boundary conditions, respectively:

$$\mathcal{L}_f(\theta; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, ..., \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, ..., \frac{\partial^2 \hat{u}}{\partial x_d \partial x_d}; \hat{\lambda}\right) \right\|_2^2, \tag{4.6}$$

$$\mathcal{L}_b(\theta; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2, \tag{4.7}$$

where the circumflex in $\hat{u}$ and $\hat{\lambda}$ denotes that these are the NN's approximations of the dependent variable and any unknown PDE paramters of inverse problems. The loss term $\mathcal{L}_r$ represents regularization functions in general. For example, for forward problems this term is left out while for inverse problems it is the error between the NN approximations and a "labeled dataset" of $u(\mathbf{x})$:

$$\mathcal{L}_f(\theta; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_r|} \sum_{\mathbf{x} \in \mathcal{T}_r} \|u(\mathbf{x}) - \hat{u}(\mathbf{x})\|_2^2. \tag{4.8}$$

(4) Train the FNN towards the optimal weights and biases $\theta^*$ by minimizing the loss function $\mathcal{L}(\theta; \mathcal{T})$: The goal of training is to optimize $\theta$, $\hat{u}$ and $\hat{\lambda}$ such that we have

$$\theta^*, \hat{u}^*, \hat{\lambda}^* = \operatorname{argmin}_{\theta, u, \lambda} \mathcal{L}(\theta, \hat{u}, \hat{\lambda}; \mathcal{T}) \tag{4.9}$$

Note that the loss function is highly nonlinear and nonconvex with respect to $\theta$, thus gradient-descent optimizers such as Adam are often used during training. The disadvantage of a nonconvex optimization problems is the difficulty to find unique solutions compared to traditional numerical methods of solving PDEs [30].

*Remark 4.2.*—Two important differences between PINNs and typical NNs are worth noting. Firstly, the former has the approximate function $\hat{u}(\mathbf{x})$ bound by the domain $\Omega$ where the governing PDE is defined. Secondly, PINNs learn from their own predictions (which is to say the governing PDEs and initial/boundary conditions are sufficient to optimize $\hat{u}(\mathbf{x})$, with respect to $\theta$. The "unlabeled" dataset of points randomly selected from $\Omega$ are split into training and validation/test sets, which is not done in some variations of PINNs such as the eigenvalue solvers (see Sec. IV B 3) since the FNN hyperparamters are fixed. In the case of inverse problems, when a dataset of the true values of $u(\mathbf{x})$ is available, part of that dataset can be used for validating the approximate $\hat{u}(\mathbf{x})$ by computing the $L_2$ relative error, one example of a test metric.

In the following, we discuss two examples of Python libraries which have been employed to construct PINNs; namely, DeepXDE [30] and Pytorch [7].

### 2. The DeepXDE package

The DeepXDE package is customized primarily for constructing PINN models. To help elaborate on the DeepXDE package, we consider here a toy problem that was discussed in Ref. [13], which involves the same Schrödinger-like differential equation in Eq. (2.4) but with an inverted symmetric Pöschl-Teller potential $V_{PT}(x)$ [13]:

$$V_{PT}(x) = \frac{1}{2\cosh^2(x)}. \tag{4.10}$$

In the tortoise coordinate $x$, the domain of our problem is infinite, i.e., $x \in (-\infty, +\infty)$, where the QNM boundary conditions are given by Eq. (2.7). Via quasi-exactly solvable theory, Ref. [40] found the exact solutions of Eq. (2.4) with $V = V_{PT}$ to be given as [13]

$$\psi_n(x) = (\cosh(x))^{(i+1)/2}\chi_n(\sinh(x)), \quad (4.11)$$

$$\omega_n = \pm\frac{1}{2} - i\left(n + \frac{1}{2}\right), \quad (4.12)$$

where $\chi_n$ is a polynomial of degree $n$ in $\sinh(x)$ and $n = \mathbb{Z}_0^+$.

As a first step to finding the approximate solutions using PINNs, we need to change to a new coordinate $y = \tanh(x)$, which maps the infinite domain $-\infty < x < +\infty$ to a finite domain of $-1 < y < +1$, so that Eq. (2.4) becomes [13]

$$(1 - y^2)^2 \frac{d^2\psi(y)}{dy^2} - 2y(1 - y^2)\frac{d\psi(y)}{dy}$$
$$+ \left[\omega^2 - \frac{1}{2}(1 - y^2)\right]\psi(y) = 0. \quad (4.13)$$

In this form, numerical implementation of this problem in PINNs becomes possible. We test the feasibility of solving Eq. (4.13) given as an inverse problem using DeepXDE. We specify $\omega$ as an unknown to be tuned while the PINN undergoes training. The total loss function $\mathcal{L}(\theta; \mathcal{T})$ of the PINN, in this case, is a weighted sum of the squared Euclidean ($L^2$-) norm of the physical constraints, similar to Eq. (4.5) [30]:

$$\mathcal{L}(\theta; \mathcal{T}) = w_f\mathcal{L}_f(\theta; \mathcal{T}_f) + w_b\mathcal{L}_b(\theta; \mathcal{T}_b) + w_o\mathcal{L}_o(\theta; \mathcal{T}_o), \quad (4.14)$$

where

$$\mathcal{L}_f(\theta; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|}\sum_{y\in\mathcal{T}_f}\left\|(1 - y^2)^2\hat{\psi}'' - 2y(1 - y^2)\hat{\psi}' + \times\left[\hat{\omega}^2 - \frac{(1 - y^2)}{2}\right]\hat{\psi}\right\|_2^2, \quad (4.15)$$

$$\mathcal{L}_b(\theta; \mathcal{T}_b) = \frac{1}{2}\sum_{y\in\mathcal{T}_b}\|\hat{\psi}(y) - \psi_b(y)\|_2^2, \qquad \mathcal{L}_o(\theta; \mathcal{T}_o) = \frac{1}{|\mathcal{T}_o|}\sum_{y\in\mathcal{T}_o}\|\hat{\psi}(y) - \psi(y)\|_2^2. \quad (4.16)$$

Note that $w_f, w_b, w_o$ are weights that are typically set to one and $\theta$ is as defined as in Sec. IV B 1. $\mathcal{T} = \{y_1, y_2, ..., y_{|\mathcal{T}|}\}$ is a set which consists of all training points randomly selected from our 1D spatial domain ($-1 < y < 1$). The

subset $\mathcal{T}_f$ are points chosen from the domain to train the FNN based on the governing equation (4.13). The subsets $\mathcal{T}_b(= \{-1, 1\}), \mathcal{T}_o$ are the boundary points for training on the boundary conditions, and the dataset of the true values
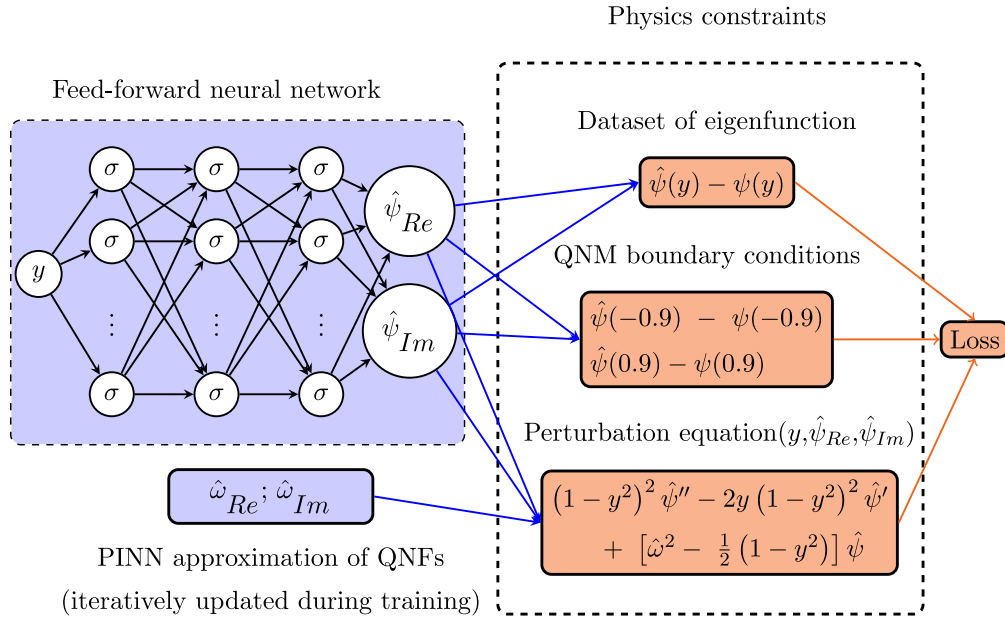


FIG. 2.   The structure of PINN for solving Eq. (4.6).

of the dependent variable (4.16), respectively. Training of this PINN proceeds as outlined in Sec. IV B 1.

Figure 2 illustrates the PINN for solving this problem. The input layer of the FNN consists of one input for coordinate $y$, while the output layer has two output nodes for real and imaginary parts of the approximate solution $\hat{\psi}$.

In building PINNs, the code we used mirrors the two-component structure of PINNs discussed in Sec. IV B. The code is fairly intuitive as it is a high-level representation that closely resembles the mathematical formulation [30]. Beginning with the physics constraints, our ODE is defined using the DeepXDE functions for executing the first- and second-order derivatives via auto diff; that is, `dde.grad.jacobian` and `dde.grad.hessian`, respectively. We define $\omega$ with the function `tf.Variable` and have represented it with $\hat{\omega}_{Re}$ and $\hat{\omega}_{Im}$ in Fig. 2.

To provide Dirichlet boundary conditions and a labeled dataset, as needed to solve our inverse problem, we define both the real and imaginary parts of the known eigenfunction $\psi(y)$ that satisfies Eq. (4.13). Numerically, at the true boundary points, $y = -1$ and $y = 1$, the solution $\psi(y)$ yields a complex infinity. As such, a narrower domain $-0.9 < y < 0.9$ is specified in the definition of the domain of our problem using the function `dde.geometry.Interval(-0.9, 0.9)`. The exact values of $\psi$ at these artificial boundary points are considered to be the Dirichlet boundary conditions. The DeepXDE function for defining these boundary conditions is `dde.DirichletBC`. To create a labeled dataset to train our PINN, we generate 50 equidistant points in the domain $(-0.9, 0.9)$ and their associated exact eigenfunctions using Eq. (4.11). This dataset is the set $\mathcal{T}_o$ in Eq. (4.16).

At this stage, we have defined the physics constraints of the PINN, but for completeness, we set up the deep neural network (our surrogate model). In the code we also define a FNN with one input node, two output nodes and three hidden layers with 20 nodes per layer. In each of the hidden layer nodes, we use the nonlinear activation function "tanh"

considering that it is a smooth, infinitely differentiable function [41]. Generally for PINNs, "smooth" activation functions are preferred over the ReLU-like nonsmooth activation functions since the former have demonstrated significant empirical success [42]. For this reason, the tanh function is chosen here by default; however, it is worth noting that (of late) adjustable, smooth function such as Swish have proven to outperform fixed functions such as tanh in terms of convergence rate and accuracy [41,43]. Swish is defined by $x\cdot$ Sigmoid$(\beta x)$, where $\beta$ is a trainable parameter.

The loss function `dde.Model` combines the FNN with the physical constraints to form a complete PINN. We also add the "callback" function `dde.callback` in the algorithm so as to keep track of the FNN approximations of $\omega$ during training. Finally, our PINN model is compiled and trained. Compilation defines the learning rate and algorithm for optimizing our model. For training the model, we choose 20 000 training epochs wherein the model will be iteratively tuned based on the physics constraints. Figure 3 displays the evolution of the loss function and model accuracy over 20 000 epochs and compares the NN approximation of $\psi(y)$ with the exact function. The PINN algorithm in DeepXDE illustrated here works well for inverse problems where $\psi$ is known at some points in the domain. However, for more general scenarios of black hole QNMs, where both $\omega$ and $\psi$ are unknown, we require an algorithm capable of solving eigenvalue problems.

### 3. The eigenvalue solver

One such algorithm that we have investigated was initiated in Ref. [7] to solve quantum eigenvalue problems using unsupervised NNs (also called, data-free surrogate models). The authors experimented with their "eigenvalue solvers" on well-known equations in quantum mechanics; namely, the time-independent Schrödinger equation with an infinite square well potential and, in another case, a quadratic potential function of a quantum harmonic oscillator. Although their approach is similar to the PINNs, in terms of embedding learning biases in the loss function,
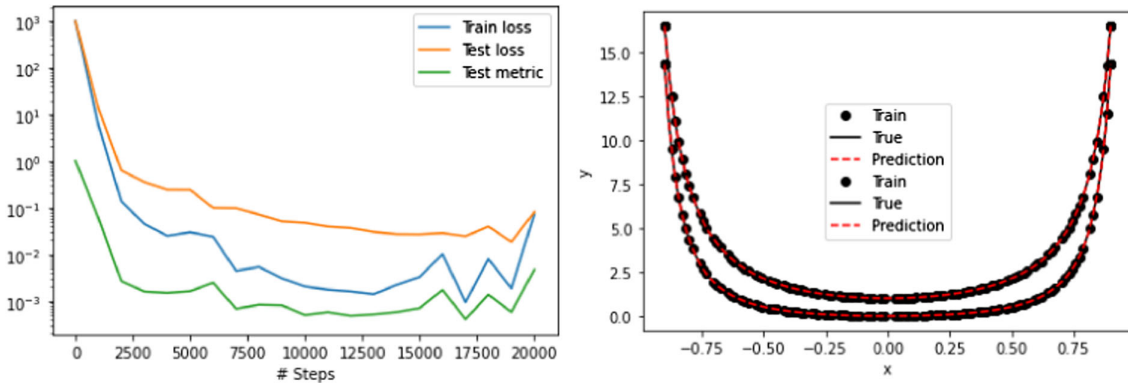


FIG. 3.   Left: plot of the train loss, test loss, and $L_2$ relative error (test metric) over a period of 20 000 epochs. Right: plot of the NN prediction of $\psi(y)$ superimposed with the true function and training points selected from the domain, $-0.9 < y < 0.9$.

there is an additional feature which allows the eigenvalue solver to scan the eigenvalue space in a scheduled manner and progressively find several eigenvalues in a single training.

To help visualize this approach, we consider one well-known bound-state eigenvalue problem [44],

$$-\frac{1}{2}\psi''(x) + V(x)\psi(x) = E\psi(x), \qquad (4.17)$$

where

$$V(x) = -\frac{\lambda(\lambda+1)}{2}\operatorname{sech}^2(x), \qquad (4.18)$$

which is a Pöschl-Teller potential and $\lambda = 1, 2\ldots$. We can now change to a new coordinate $u = \tanh(x)$. As such,

Eq. (4.17) can be written in the form of a Legendre differential equation:

$$[(1-u^2)\psi'(u)]' + \lambda(\lambda+1)\psi(u) + \frac{2E}{1-u^2}\psi(u) = 0, \qquad (4.19)$$

which is solved exactly by associated Legendre functions, i.e., $\psi(x) = P_\lambda^\mu(\tanh(x))$ with $E = -\mu^2/2$ and $\mu = 1, 2, 3\ldots, \lambda$. These are bound states that vanish at the boundaries of the eigenvalue problem, i.e., $\psi(x = \pm\infty) = 0$ or $\psi(u = \pm 1) = 0$.

The eigenvalue solvers in Ref. [7] are built using the PyTorch library. To solve Eq. (4.19) using the eigenvalue solvers, we embed them in the loss function of the NN along with some regularization terms, similar to Eq. (4.5):

$$\mathcal{L}(\theta; \mathcal{T}) = \mathcal{L}_{\text{ODE}}(\theta; \mathcal{T}) + \mathcal{L}_{\text{reg}}(\theta; \mathcal{T}), \qquad (4.20)$$

where

$$\mathcal{L}_{\text{ODE}}(\theta; \mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{u\in\mathcal{T}}\left[((1-u^2)\hat{\psi}'(u))' + \lambda(\lambda+1)\hat{\psi}(u) + \frac{2\hat{E}}{1-u^2}\hat{\psi}(u)\right]^2, \qquad (4.21)$$

$$\mathcal{L}_{\text{reg}}(\theta; \mathcal{T}) = w_f\mathcal{L}_f(\theta; \mathcal{T}) + w_E\mathcal{L}_E(\theta; \mathcal{T}) + w_{\text{drive}}\mathcal{L}_{\text{drive}}(\theta; \mathcal{T}). \qquad (4.22)$$

As defined in Sec. IV B 1, $\mathcal{T}$ is a set of training points randomly selected from the domain $u \in (-1, 1)$. Figure 4 illustrates how the boundary conditions (i.e., a vanishing solution at the boundary points) are enforced using a parametric function $(1 - u)(1 + u)$. Note also the absence

of the observational bias term (the reason our eigenvalue solver is called a data-free model).

In Eq. (4.22) $\mathcal{L}_{\text{reg}}$ is a weighted sum of regularization functions, where the weights $w_f, w_E, w_{\text{drive}}$ are typically set to one [7]. Individually, the regularization functions are
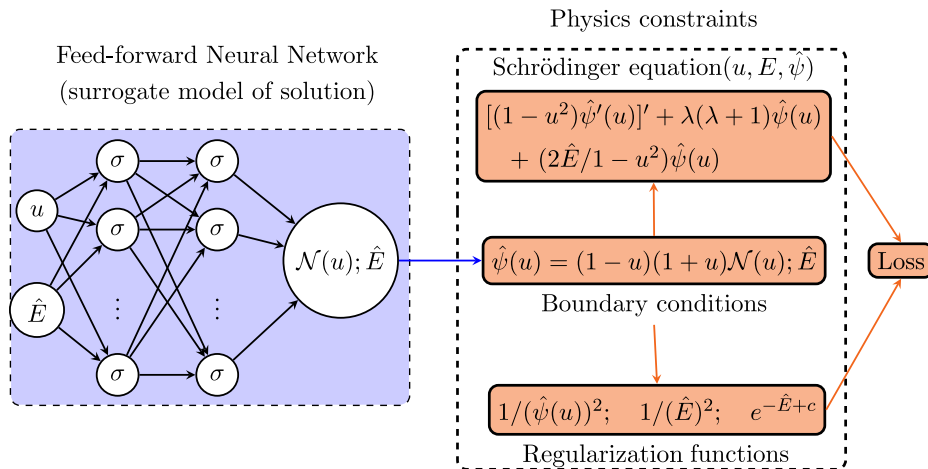


FIG. 4. The structure of the eigenvalue solver. Unlike the PINNs used for inverse problems in DEEPXDE, these FNNs are unsupervised. Instead, the unknown eigenpairs can be determined only from the governing equations and boundary conditions (that are enforced using a parametric function $(1 - u)(1 + u)$, a "hard constraint," to ensure they are satisfied exactly [30]).
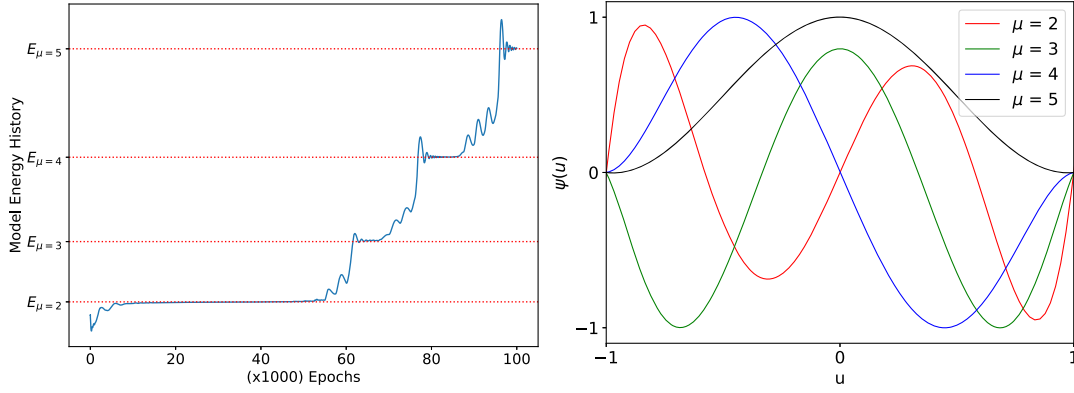
FIG. 5. Plot of the NN approximations of the $\psi$ and $-E$, the Pöschl-Teller bound states. Notice that the plot of $-E$ plateaued whenever the NN found an eigenvalue given by $-E = \mu^2/2$ for $\mu = 1, 2, 3, \ldots$. As expected, the bound states $\psi(u)$ vanish at the boundary points.

$$\mathcal{L}_f = \frac{1}{|\mathcal{T}|} \sum_{u \in \mathcal{T}} \frac{1}{\hat{\psi}^2}, \qquad \mathcal{L}_E = \frac{1}{|\mathcal{T}|} \sum_{u \in \mathcal{T}} \frac{1}{\hat{E}^2},$$

$$\mathcal{L}_{\text{drive}} = \frac{1}{|\mathcal{T}|} \sum_{u \in \mathcal{T}} \exp(-\hat{E} + c), \qquad (4.23)$$

where $\mathcal{L}_f$ and $\mathcal{L}_E$ steer the learning algorithm away from zero as a possible value for the eigenfunction and eigenvalue, respectively. For this purpose, the mathematical form of these loss terms have the PINN approximations ($\hat{\psi}$ and $\hat{E}$) inversely proportional to the loss so that as they approach zero they are penalized by high loss values. The crucial term in these unsupervised NNs is $\mathcal{L}_{\text{drive}}$, which motivates the NN to scan through the space of eigenvalues. This is achieved by adding within the training algorithm a mechanism that increases the constant $c$ in $\mathcal{L}_{\text{drive}}$ at regular intervals, after an arbitrary number of training epochs.

It is important to note that without the $\mathcal{L}_{\text{drive}}$ loss component the PINNs lack the necessary constraint to learn other eigenvalues than the first energy level it initially gravitates towards during training, which is often but not always the ground energy level. In this case, the algorithm has more limitations, similar to the original, baseline PINN loss function [Eq. (4.5)] where forward and inverse approaches can learn, respectively, only one of the eigenfunctions and eigenvalues at a time, and only when at least one of the other eigenpairs is known. As a consequence, a classification approach (with, for example, output nodes of the PINN representing the dependent variable) may not be applicable because the loss function will only have the wherewithal to learn a single eigenstate in each training, regardless of the input data since it is unlabeled and is randomly selected from a domain where one eigenstate cannot be separated spatially from the other solutions.

The key Pytorch functions used in defining our physics constraints include `torch.autograd`, which executes automatic differentiation to find the first and second derivatives in $\mathcal{L}_{\text{ODE}}$ given by Eq. (4.21). With the physics constraints defined, we set the structure of our FNN: 2 input

nodes, 1 output node, and 2 hidden layers with 50 nodes each (see Fig. 4), where our chosen activation function is the trigonometric function, sine. This activation function has been found to greatly accelerate the NN's convergence to eigenstates compared to more common functions, e.g., sigmoid and ReLU [7,45].

Compared to the code in DeepXDE, the eigenvalue solvers provided more flexibility when customizing the training algorithm. The total loss function in our training algorithm was defined according to Eqs. (4.20)–(4.23). To generate `n_train` points from the domain of our example problem $u \in (-1, 1)$, we used the Pytorch function `torch.linspace`. In terms of optimization, the standard Adam optimizer is applied [36].

Ultimately, the training phase follows after all parameters for training the model (such as the number of training epochs) have been defined. In our case, we chose the following parameters: 100 training points, 100 000 training epochs, and a learning rate of $8 \times 10^{-3}$. Figure 5 shows the resulting NN approximations of the eigenvalues and eigenfunctions. Note that the $\mathcal{L}_{\text{drive}}$ is only included in the loss function of this example, for complete demonstration of the method. However, it is not applied in the QNM computations resulting in the PINNs converging on one eigenvalue (as we will see), rather than several (as in the many plateaus of Fig. 5). As seen in the example, the flips between eigenvalues occur arbitrarily, without any method of controlling when they occur. Therefore, this loss term requires further investigation, outside this present work, to make it less random.

## V. RESULTS: QNM COMPUTATIONS WITH THE EIGENVALUE SOLVER

The results from our investigation of the performance of PINNs when applied to the computation of QNMs shall now be presented, where it is important to note that, generally for deep neural networks, there are no set rules for customising them since they are statistical tools with too many parameters to admit any meaningful physical

interpretability. Taking this into account in this work, we have carried out grid-search-like experimentation of the NN hyperparameters to discover the most optimal choices with the best performance. Specifically, we considered a range of values for three hyperparameters, which are the number of training points, number of training epochs, and the number of nodes per layer, keeping the other hyperparameters (e.g., optimizer and activation function) fixed. In this work, we have focused on computing the QNMs of a Schwarzschild black hole in the asymptotically flat and near extremal de Sitter cases. For the former, we considered massless scalar, Dirac, electromagnetic and gravitational field perturbations; while, for the latter, we only considered massless scalar fields where the equations look the same as near extremal Reissner-Nordström-de Sitter black holes. Due to this, the QNMs of near extremal Schwarzschild-de Sitter black holes can be more generally treated as the QNMs of near extremal nonrotating de Sitter black holes.

### A. Scanning hyperparameters

Figure 6 graphs the results we obtained from testing different hyperparameter configurations for computing the QNMs of an asymptotically flat Schwarzschild black hole ($s = 0, \ell = 2, n = 0$). The accuracy of the NN approximations (measured in terms of percentage deviation) and the execution times for training our NNs have been measured as a function of the number of training points, number of training epochs, and number of nodes per layer. The fixed hyperparameters were learning rate of $8 \times 10^{-3}$, 2 hidden layers, and sine as the activation function.

Note that the accuracy values measure the deviation of the NN approximations from Leaver's QNMs, whose precision is up to 4 decimal places [12,18]. As seen in

Fig. 6, the percentage deviations of our computations remain the same across all hyperparameter configurations. But for a few cases, the percentage deviations for the real and imaginary parts of the QNMs hover around about 0.009% and $-0.042\%$, respectively. Both these values correspond to a 4 decimal place precision, making the NN approximations as good as Leaver's CFM. Note that beyond 4 decimal places we cannot reliably determine the accuracy of our NN approximations based on the QNMs given in the literature [12,18].

The red cells given in the right panel of Fig. 6 correspond to cases where the eigenvalue solvers veer from determining the QNMs with a minimum loss, which are the $n = 0$ modes. These are few in comparison to "normal" cases where the eigenvalue solvers converge to a loss minimizing solution. The displayed training times and percentage deviations were obtained by iterating the eigenvalue solver algorithm automatically and scanning through the specified range of hyperparameter combinations. Note that the total loss was set as

$$\mathcal{L}(\theta; \mathcal{T}) = \mathcal{L}_{\text{ODE}}(\theta; \mathcal{T}) + \mathcal{L}_f(\theta; \mathcal{T}), \qquad (5.1)$$

where

$$\mathcal{L}_{\text{ODE}}(\theta; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{\xi \in \mathcal{T}} [\chi'' - \lambda_0(\xi)\chi' - s_0(\xi)\chi]^2, \quad (5.2)$$

$$\mathcal{L}_f(\theta; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{\xi \in \mathcal{T}} \frac{1}{\hat{\chi}^2}. \qquad (5.3)$$

Here $\theta, \mathcal{T}, \hat{\chi}$, and $\hat{\omega}$ have their definitions from Secs. II A and IV B 1. We have considered the ODE given by
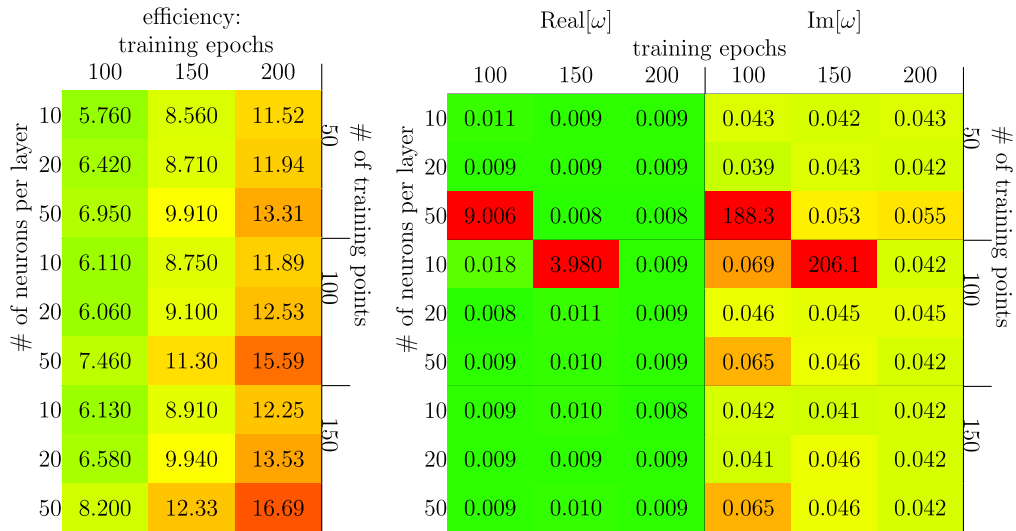


FIG. 6.    The training times, in minutes (right panel) and percentage deviations (left panel) obtained for different hyperparameter choices. To compute the QNMs of asymptotically flat Schwarzschild BHs ($s = 0, \ell = 2, n = 0$), we tested different permutations of the number of training points, number of neurons per layer and number of training epochs ($\times 1000$).

TABLE I.  The eigenvalue solver (eigeNN) approximations of the fundamental mode ($n = 0$, $\ell = 1, \ldots, 3$) QNMs for massless scalar field perturbations of near extremal SdS and RNdS BHs.

| $n$ | $\ell$ | $\omega_{\mathrm{Exact}}$ [24] | $\omega_{\mathrm{eigeNN}}$ (no seed) | $\omega_{\mathrm{eigeNN}}$ (with seed) |
|---|---|---|---|---|
| 0 | 1 | $1.322876 - 0.5i$ | $1.322886 - 0.500004i$ | $1.322894 - 0.500011i$ |
|   |   |   | $(< 0.001\%)(< 0.001\%)$ | $(0.001\%)(0.002\%)$ |
|   | 2 | $2.397916 - 0.5i$ | $2.397917 - 0.500001i$ | $2.397916 - 0.500000i$ |
|   |   |   | $(< 0.001\%)(< 0.001\%)$ | $(< 0.001\%)(< 0.001\%)$ |
|   | 3 | $3.427827 - 0.5i$ | $3.427828 - 0.500001i$ | $3.427828 - 0.500000i$ |
|   |   |   | $(< 0.001\%)(< 0.001\%)$ | $(< 0.001\%)(< 0.001\%)$ |

Ref. [13], which is a transformation of the radial perturbation equation (2.4) to a finite domain of the coordinate $\xi \in (-1, 1)$. In this form of the loss function given by Eq. (5.1), the NN is motivated to converge on the QNMs with the highest amplitude, $|\chi|$, because of the regularization loss term $\mathcal{L}_f$. Incidentally, it turns out that the QNM with the highest $|\chi|$ (for any given multipole number) is the $n = 0$ mode. This is consistent with the fact that for black hole QNMs, the higher overtones are damped faster [14].

Some observations from Fig. 6 are that varying the hyperparameters, as we did, has no significant effect on the accuracy. However, there is an increase in the training time with the number of epochs for a fixed number of training
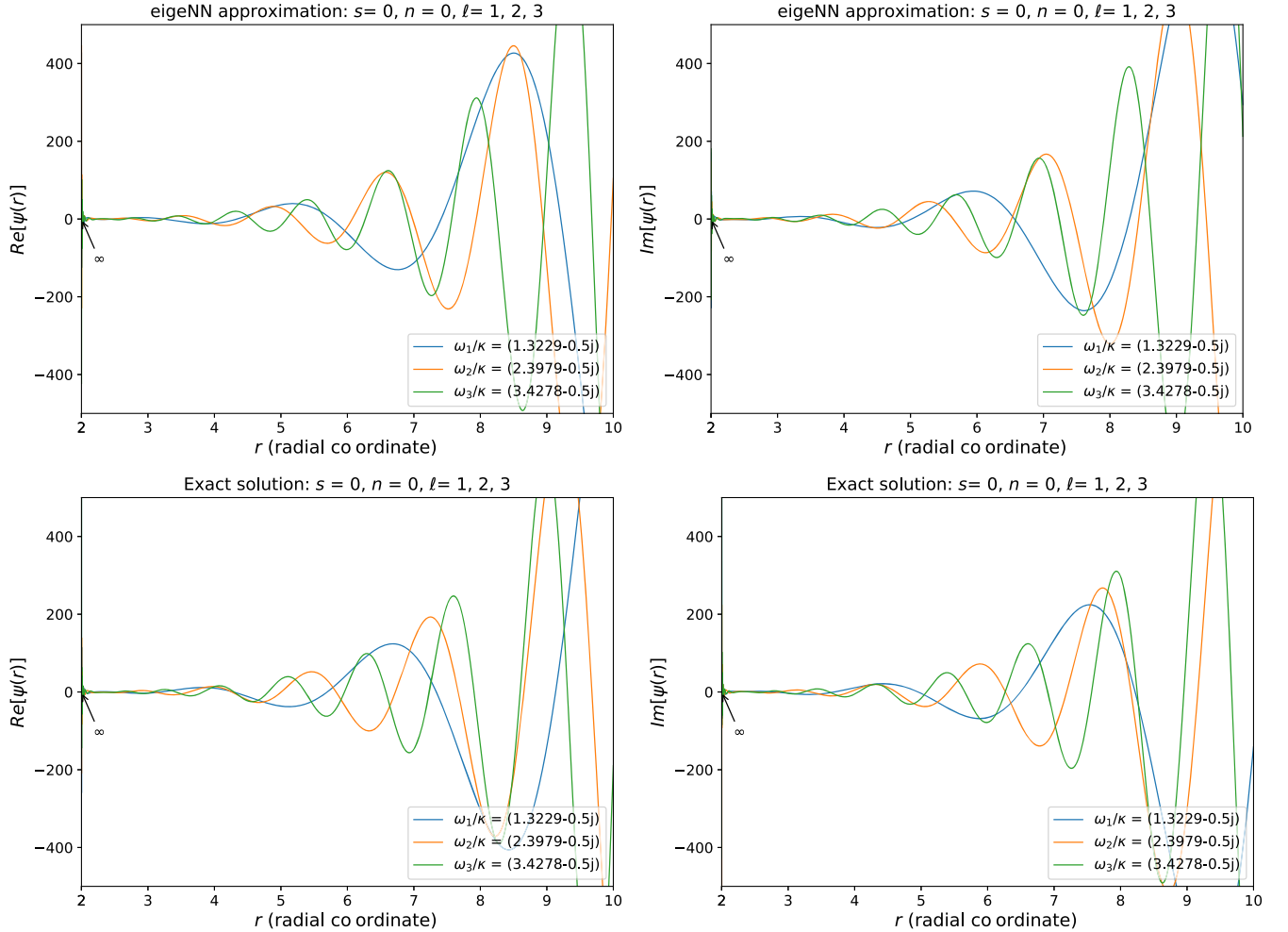


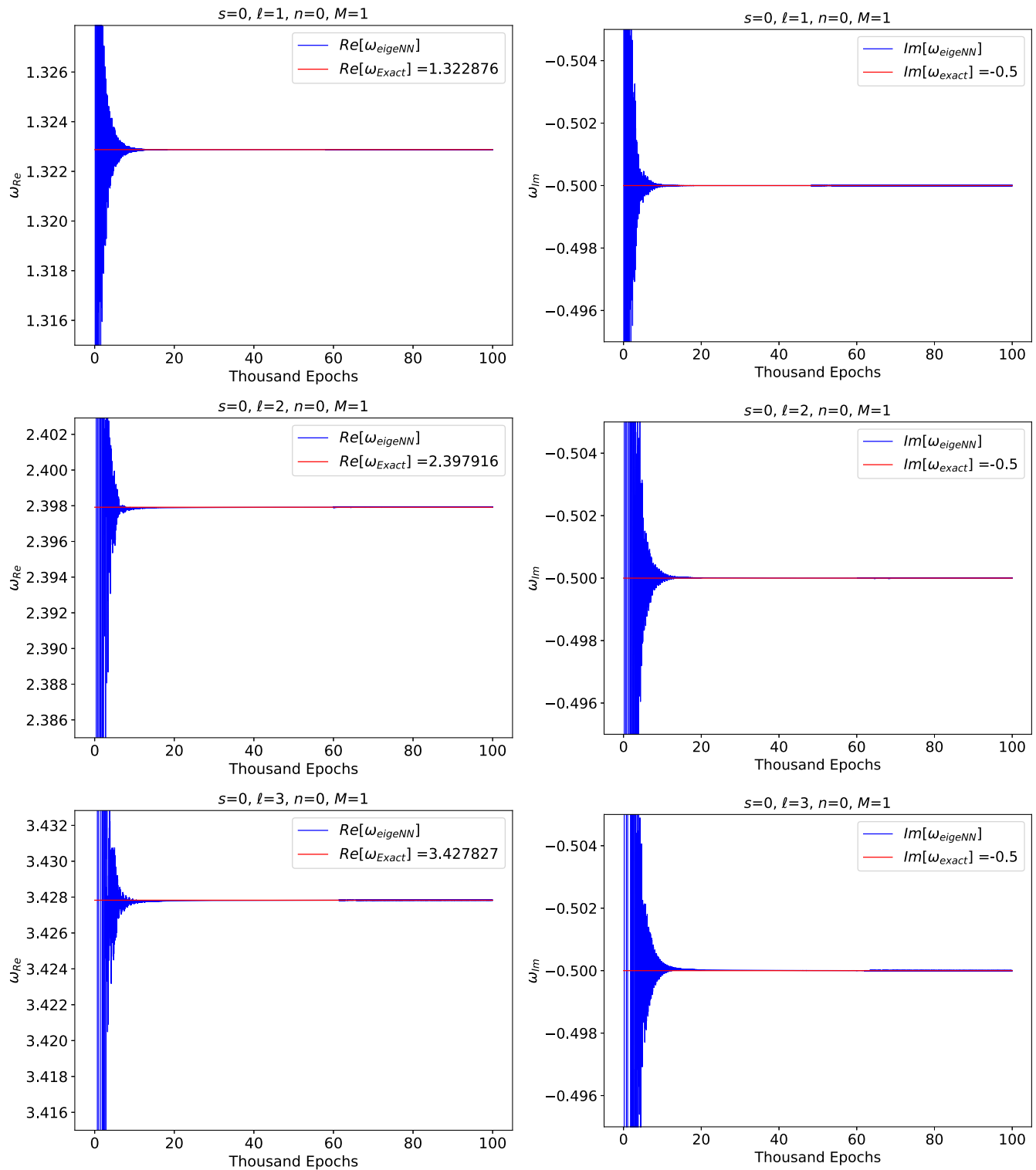FIG. 7.  The exact QNM wave functions [cf. Eq. (2.21)] vs the eigeNN approximations.

FIG. 8. Evolution of the NN approximations of the QNM frequencies during the 100 000 epoch-long training phase. These were generated from our computations using the eigenvalue solver with no seed loss term in the loss function.

points and neurons per layer. Additionally, an increase in the number of neurons per layer also leads to a slight increase in the training time. Therefore, to obtain a favorable trade-off between accuracy and efficiency, one may train for 100 000 epochs instead of 200 000 to achieve the same level of accuracy in less time. This reduction in training time becomes significant when running a large batch of computations.

TABLE II. The eigenvalue solver (eigeNN) approximations of the fundamental mode ($n = 0$) QNMs given up to 4 decimal places for massless scalar field perturbations ($s = 0$) of asymptotically flat Schwarzschild BHs.

| $n$ | $\ell$ | $\omega_{\text{Leaver}}$ [18] | $\omega_{\text{eigeNN}}$ (90 000 epochs) | $\omega_{\text{WKB}}$ [46] (sixth order) |
|---|---|---|---|---|
| 0 | 0 | $0.1105 - 0.1049i$ | $0.1106 - 0.1049i$ | $0.1105 - 0.1008i$ |
|   |   |   | $(0.10\%)(-0.01\%)$ | $(-0.03\%)(-3.89\%)$ |
|   | 1 | $0.2929 - 0.0977i$ | $0.2929 - 0.0977i$ | $0.2929 - 0.0978i$ |
|   |   |   | $(0.02\%)(-0.02\%)$ | $(<0.01\%)(0.06\%)$ |
|   | 2 | $0.4836 - 0.0968i$ | $0.4836 - 0.0968i$ | $0.4836 - 0.0968i$ |
|   |   |   | $(0.01\%)(-0.04\%)$ | $(<0.01\%)(-0.04\%)$ |

TABLE III. The eigeNN approximations of the fundamental mode ($n = 0$) QNM frequencies for Dirac field perturbations ($s = 1/2$).

| $n$ | $\ell$ | $\omega_{\text{Leaver}}$ [18] | $\omega_{\text{eigeNN}}$ (90 000 epochs) | $\omega_{\text{WKB}}$ [46] (sixth order) |
|---|---|---|---|---|
| 0 | 1 | $0.2822 - 0.0967i$ | $0.2822 - 0.0966i$ | $0.2822 - 0.0967i$ |
|   |   |   | $(0.01\%)(-0.09\%)$ | $(<-0.01\%)(-0.02\%)$ |
|   | 2 | $0.4772 - 0.0963i$ | $0.4772 - 0.0963i$ | $0.4772 - 0.0963i$ |
|   |   |   | $(0.01\%)(0.03\%)$ | $(<0.01\%)(0.05\%)$ |
|   | 3 | $0.6708 - 0.0963i$ | $0.6708 - 0.0963i$ | $0.6708 - 0.0963i$ |
|   |   |   | $(<-0.01\%)(-0.02\%)$ | $(<-0.01\%)(-0.02\%)$ |

TABLE IV. The eigeNN approximations of the fundamental mode ($n = 0$) QNM frequencies for electromagnetic field perturbations ($s = 1$).

| $n$ | $\ell$ | $\omega_{\text{Leaver}}$ [18] | $\omega_{\text{eigeNN}}$ (90 000 epochs) | $\omega_{\text{WKB}}$ [46] (sixth order) |
|---|---|---|---|---|
| 0 | 1 | $0.2483 - 0.0925i$ | $0.2483 - 0.0925i$ | $0.2482 - 0.0926i$ |
|   |   |   | $(-0.01\%)\ (0.01\%)$ | $(-0.04\%)\ (0.15\%)$ |
|   | 2 | $0.4576 - 0.0950i$ | $0.4576 - 0.0950i$ | $0.4576 - 0.0950i$ |
|   |   |   | $(<-0.01\%)\ (<0.01\%)$ | $(<-0.01\%)\ (0.01\%)$ |
|   | 3 | $0.6569 - 0.0956i$ | $0.6569 - 0.0956i$ | $0.6569 - 0.0956i$ |
|   |   |   | $(<0.01\%)\ (0.01\%)$ | $(<-0.01\%)\ (0.02\%)$ |

TABLE V. The eigeNN approximations of the fundamental mode ($n = 0$) QNM frequencies for gravitational field perturbations ($s = 2$).

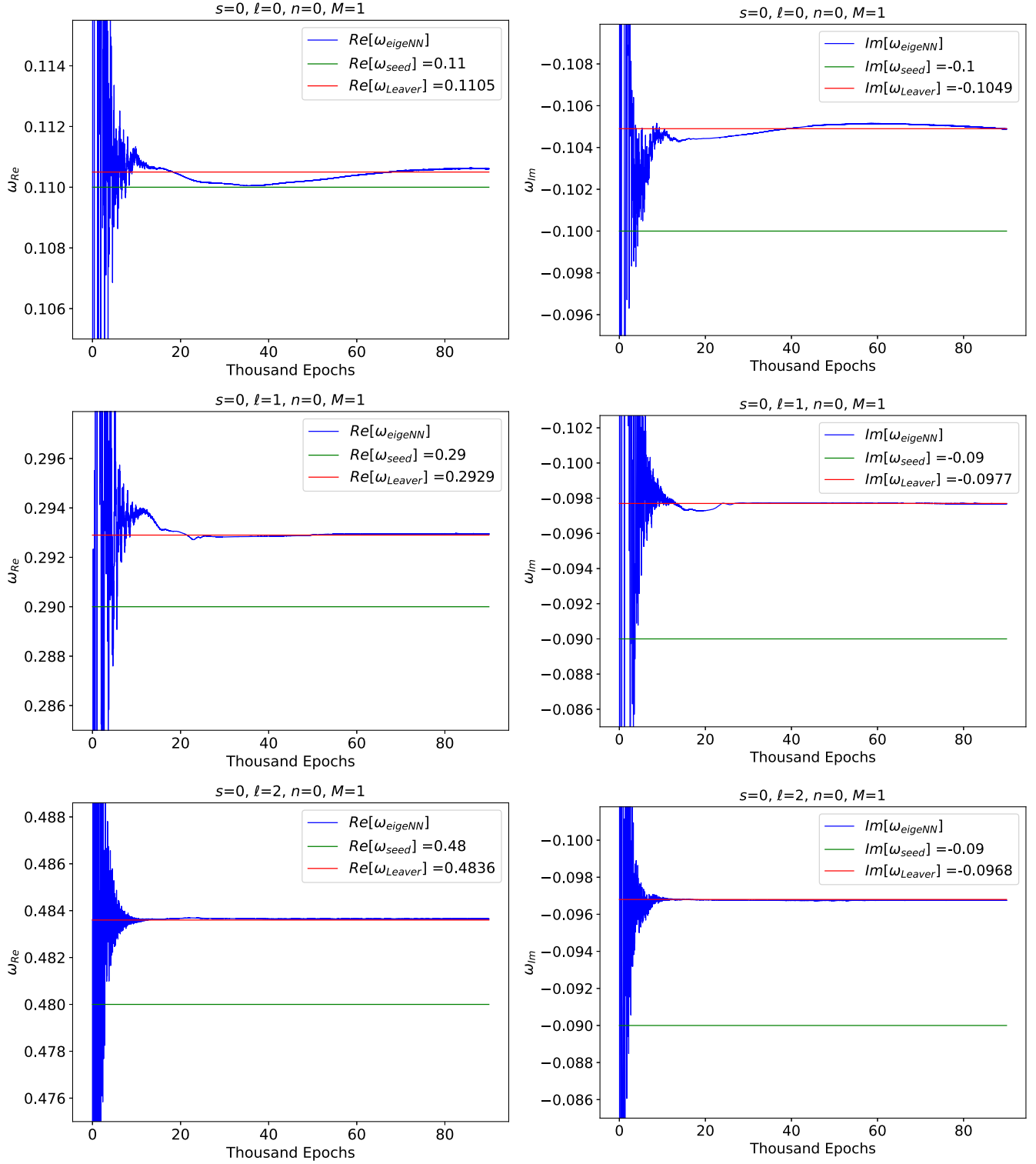| $n$ | $\ell$ | $\omega_{\text{Leaver}}$ [18] | $\omega_{\text{eigeNN}}$ (90 000 epochs) | $\omega_{\text{WKB}}$ [46] (sixth order) |
|---|---|---|---|---|
| 0 | 2 | $0.3737 - 0.0896i$ | $0.3737 - 0.0890i$ | $0.3736 - 0.0889i$ |
|   |   |   | $(0.01\%)\ (-0.05\%)$ | $(-0.02\%)\ (-0.12\%)$ |
|   | 3 | $0.5994 - 0.0927i$ | $0.5994 - 0.0927i$ | $0.5994 - 0.0927i$ |
|   |   |   | $(0.01\%)\ (<0.01\%)$ | $(<0.01\%)\ (<0.01\%)$ |
|   | 4 | $0.8092 - 0.0942i$ | $0.8092 - 0.0942i$ | $0.8092 - 0.0942i$ |
|   |   |   | $(<0.01\%)\ (0.04\%)$ | $(<-0.01\%)\ (-0.03\%)$ |

FIG. 9.    Evolution of the NN approximation of QNMs for an asymptotically flat Schwarzschild BH. The 90 000 epoch-long training phase takes an average time of 12 minutes. By contrast, it takes less than 1 minute to generate the QNMs (listed in Table II) using the sixth order WKB method.
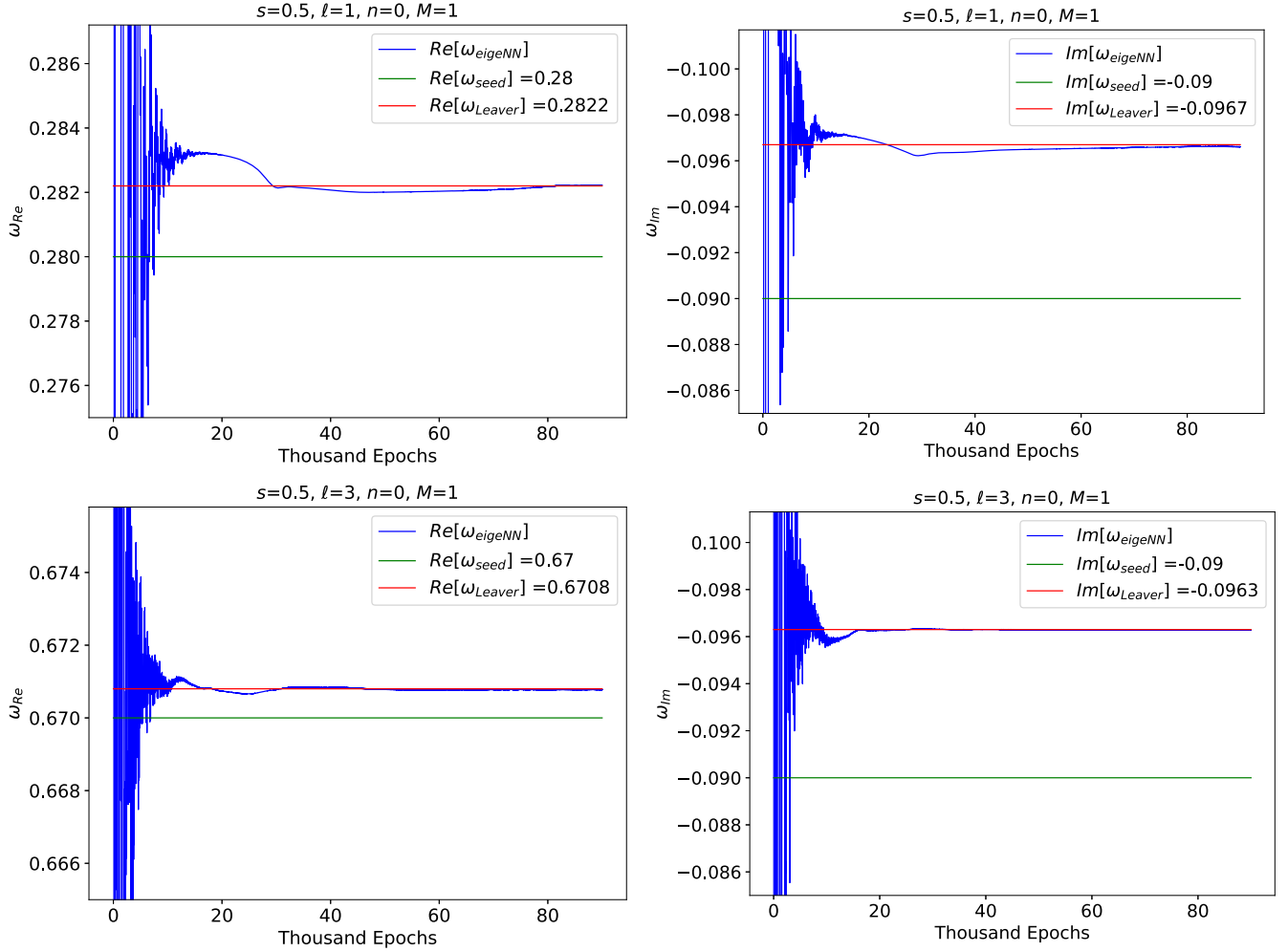
FIG. 10. Evolution of the NN approximation of QNMs over a 90 000 epoch-long training phase.

## B. QNMs of near extremal nonrotating black holes

In our discussion of black hole perturbation equations in Sec. II C, we have seen a special case where the effective potential is given exactly by an inverted Pöschl-Teller potential: namely, the near extremal Schwarzschild and Reissner-Nordström-de Sitter black holes. In these cases, analytic expressions of the QNMs are known and we could reliably test the accuracy of our NN approximations compared to the exact QNMs given as

$$\omega = \sqrt{\left(\ell(\ell+1) - \frac{1}{4}\right)} - i\left(n + \frac{1}{2}\right), \quad n = 0, 1, 2, \dots \quad (5.4)$$

where $\ell$ and $n$ are as defined in Sec. II A.

In Table I, the exact QNMs for $n = 0$ and $\ell = 1, \dots, 3$ are compared with the NN approximations ($\omega_{\text{eigeNN}}$). The latter were obtained by embedding the governing differential equation of near extremal nonrotating de Sitter black holes and extra regularization terms in the loss function. In the last column of Table I are values that were

produced by adding to the loss function a seed value loss term given as

$$\mathcal{L}_{\text{seed}}(\theta, \hat{\omega}; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{\xi \in \mathcal{T}} [\hat{\omega} - \omega_{\text{seed}}]^2. \quad (5.5)$$

The seed value loss term measures the deviation of the NN approximations from specific $n$ and $\ell$ dependent seed values close to the exact QNMs (i.e., accurate up to a certain number of decimal places, e.g., 2 decimal places, in this case). The goal of the seed loss term is to steer the NN towards specific QNMs of the several possible differential equation residual minimizers (or eigenstates) that exist for a chosen multipole number $\ell$.

The plots in Fig. 7 are the NN approximations of the eigenpairs ($\omega, \psi$) associated with Table I, where the first three multipole numbers for the $n = 0$ mode are superimposed. These are the QNM eigenfunctions that obey the asymptotic behavior expected for astrophysical asymptotically de Sitter black holes [9]. As was pointed out previously, this is
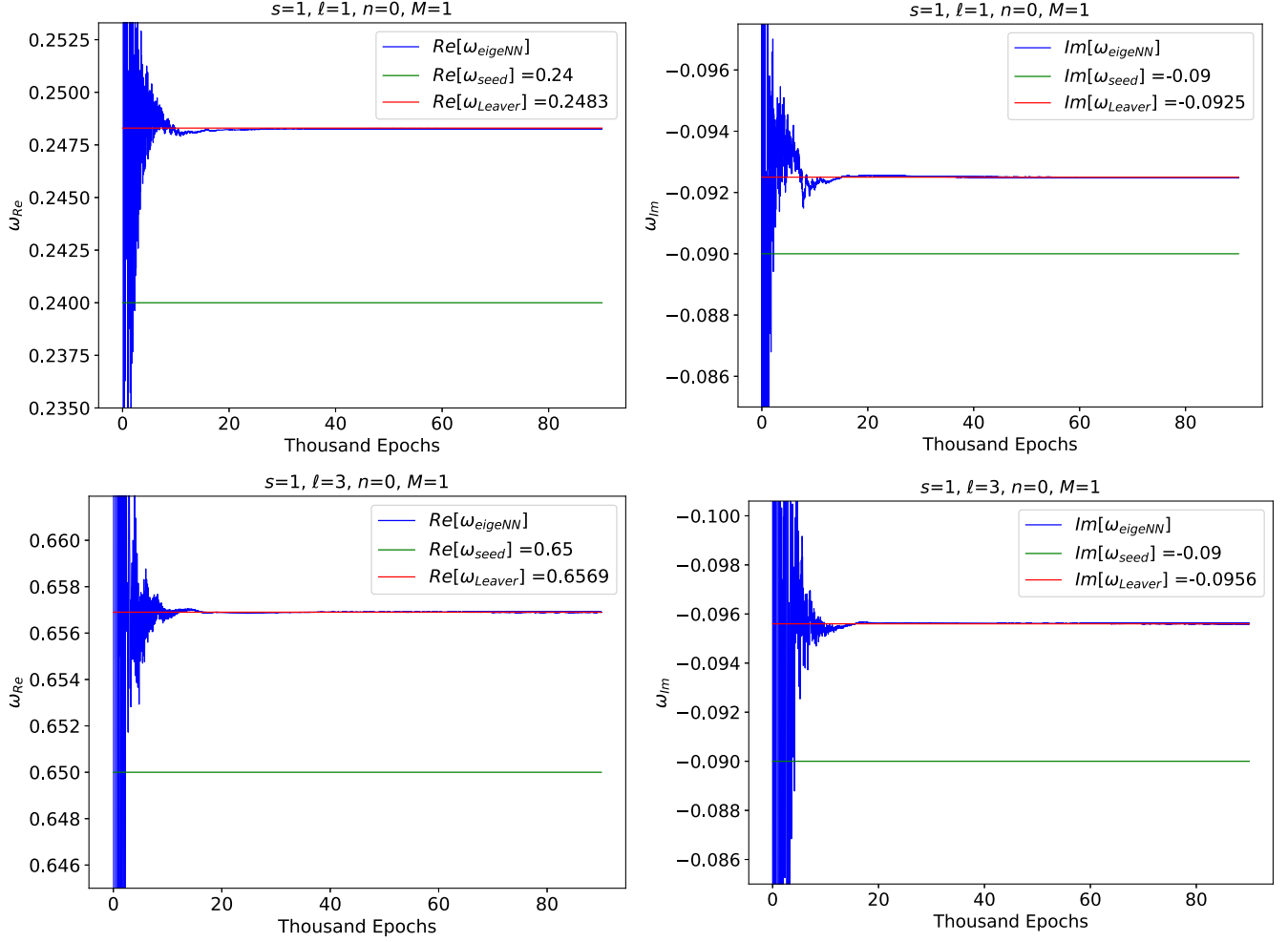
FIG. 11.   Evolution of the NN approximation of QNMs during a 90 000 epoch-long training phase.

$$\psi \sim \text{pure outgoing wave}, \qquad x \rightarrow +\infty. \qquad (5.6)$$

More importantly, Fig. 8 shows the evolution of the real ($\omega_{\text{Re}}$) and imaginary ($\omega_{\text{Im}}$) parts of the NN's approximations of the QNMs as they train for 100 000 epochs. These plots were obtained from our computations without the seed loss term in the loss function.

Figure 8 shows that the convergence of the NN approximation ($\omega_{\text{eigeNN}}$) towards the expected QNMs ($\omega_{\text{exact}}$) given by Eq. (5.4), occurs swiftly after training begins. Table I and Fig. 8 indicate that the NN learns QNMs with similar levels of accuracy for different multipole numbers, regardless of the presence of a seed loss term in the loss function.

### C. QNMs of asymptotically flat Schwarzschild black holes

For this scenario, we have considered perturbations of asymptotically flat Schwarzschild black holes by massless scalar, Dirac, electromagnetic and gravitational fields given by Eqs. (2.4) and (2.5) in the tortoise coordinate.

Tables II–V compare our NN approximations of the QNMs with those given in the literature for the CFM and WKB approaches for solving the perturbation equations.

With regards to the setup of our eigenvalue solvers, the same FNN configuration was used for all our computations. That is, we set up 2 hidden layers, 50 nodes per layer, and sine as the nonlinear activation function. Moreover, we employed the Adam optimizer, set up 90 000 training epochs and used a learning rate of $8 \times 10^{-3}$. Our training data consisted of 100 points randomly selected from the domain $\xi \in [0, 1]$. Note also that the percentage deviation values given inside the parentheses in the tables are

percentage deviation

$$= \frac{|\text{Re/Im}[\omega_{\text{eigeNN/WKB}}]| - |\text{Re/Im}[\omega_{\text{Leaver}}]|}{|\text{Re/Im}[\omega_{\text{Leaver}}]|} \times 100. \quad (5.7)$$

In the plots of Figs. 9–12, the green line represents the seed values of $\omega$ that were embedded in the loss function of our eigenvalue solvers. Note that the NN converges towards the expected QNMs, rather than the seed values, which are

given up to 2 decimal places. The QNM values given in Tables I–V are in geometrical units.

In Figs. 9–12, it is clear that the eigenvalue solvers are able to learn the expected values of $\omega_{\mathrm{Re}}$ and $\omega_{\mathrm{Im}}$ for various perturbation scenarios of an asymptotically flat Schwarzschild black hole. For the fundamental mode, $n = 0$ and various choices of $s$ and $\ell$, the physical constraints provided in the loss function are sufficient to steer the NN toward the exact, nontrivial solutions of our perturbation equations. This is remarkable, considering the conceptual simplicity of the NN optimization algorithm.

As is evidenced by the Tables II–V, the QNMs computed by our eigenvalues solvers are as accurate as the values computed with the CFM and the sixth-order WKB method. Since we have used the QNMs from Leaver [12,18], which are given up to 4 decimal places, as our closest approximation to an exact solution, we only confirm accuracy up to that level. Needless to say, an ideal measure of accuracy of our approximations would be exact QNMs obtained via analytical methods for solving our differential equations. Note that the QNMs obtained using the AIM (which are not listed in the tables but can be found in Ref. [13]) were shown to be as accurate as Leaver's method.

Regarding the time taken by eigenvalue solvers to complete executing computations, there is a significant difference between the duration for training our NNs compared to that of running the other approximation techniques. While it takes around 10 minutes to run each of the 90 000 epoch-long training sessions to solve our perturbation equations, the computation takes much less time. For example, in the case of the WKB and AIM techniques, many QNM values can be computed in less than a minute. Note that this comparison is tentative and reflects just the outcome of the present work, which is a baseline for potential future improvements. For example, the computational speeds of PINNs could be enhanced by tapping into the parallelizable nature of NNs. So far we have focused on fine-tuning the accuracy of NN approximations as that is particularly important for black hole QNMs.

In terms of other performance measures, the scalability of PINNs, with regard to the ability to handle a large number of input dimensions, is one major advantage of
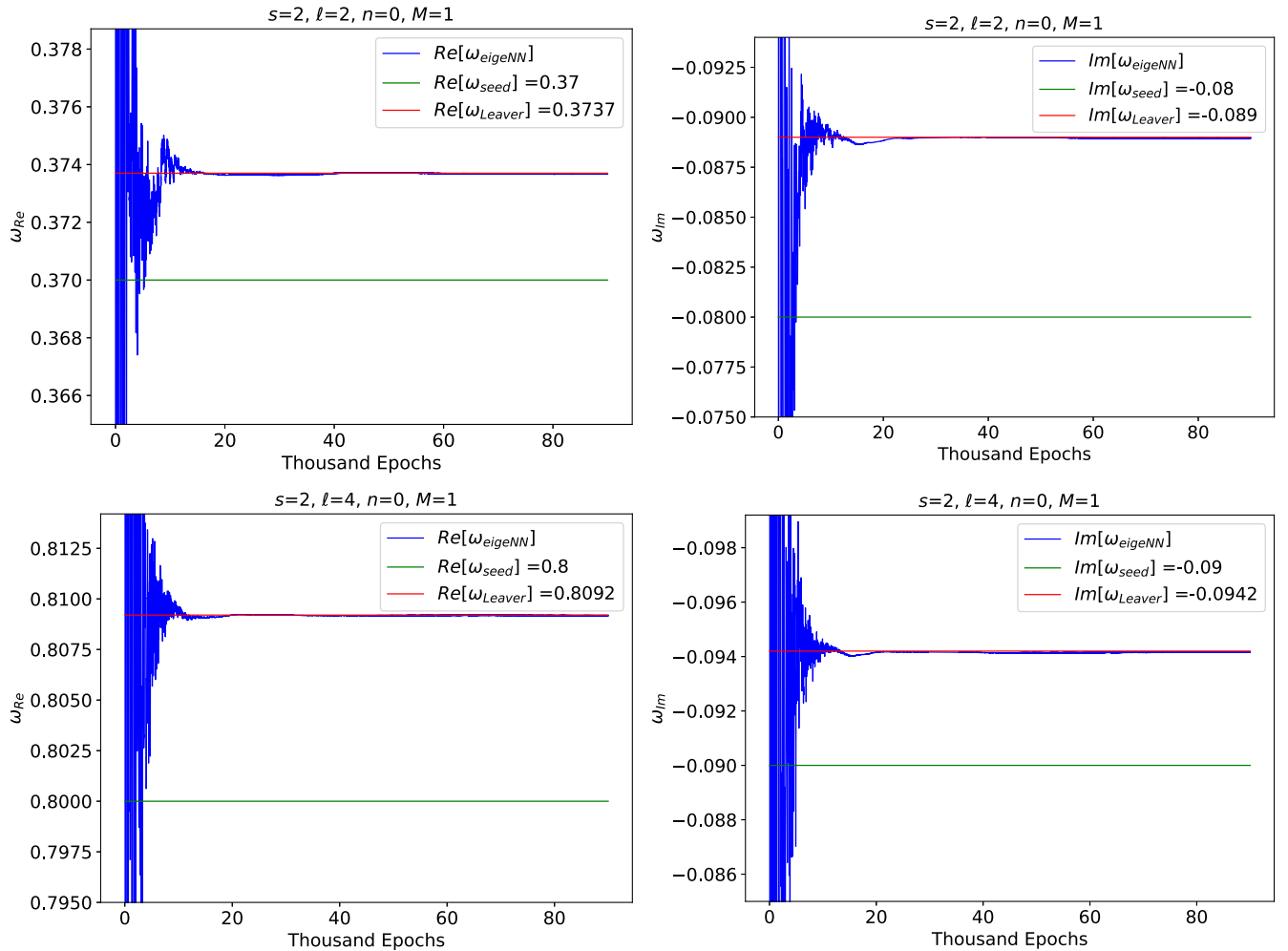


FIG. 12. Evolution of the NN approximation of QNMs during a 90 000 epoch-long training phase.

PINNs over other numerical methods. As pointed out in Refs. [30,47], NNs overcome the curse of dimensionality and, therefore, have the capacity to approximate high-dimensional functions quite efficiently. This attribute justifies the potential future extension of PINNs to compute the QNMs associated with high-dimensional perturbation scenarios, where traditional mesh-based approximation techniques could suffer.

Our results show signs of the expected limitations, listed in Ref. [38], of solving PDEs with NNs that have been observed in various applications of physics-informed machine learning. One is the fact that complicated loss functions (with many terms in the governing equations) lead to highly nonconvex optimization problems. As a result, the training process may not be sufficiently stable and convergence to the global minimum may not be guaranteed [38].

We can see this by contrasting the results obtained from our computations involving the relatively simple differential equation for near extremal nonrotating black holes versus the relatively more complex perturbation equations of asymptotically flat Schwarzschild black holes. For the former, Fig. 8 shows that our NN quickly converges towards the expected solution regardless of the multipole number, even in the absence of a seed loss term to further constrain the eigenvalue solvers. However, for the latter, the NN has more difficulty converging in lower multipole number cases as seen in Fig. 9 where the NN converges faster for $\ell = 2$ when compared to $\ell = 0$. In fact, without the seed loss term to solve asymptotically flat Schwarzschild black holes, the NN fails to converge when we have $\ell = 0, 1, 2$ but does for $\ell > 3$.

In our attempts to solve even more challenging problems, such as the perturbation equations of asymptotically flat Reissner-Nordström and asymptotically (anti-)de Sitter Schwarzschild black holes, the instability appears to be more pronounced as the NNs fail to converge on the expected QNMs for these cases. To alleviate this constraint and broaden the scope of PDEs to be solved we will need to add to our eigenvalue solvers some stronger constraints or features that address instability.

## VI. DISCUSSIONS AND CONCLUSION

In summary, we have explored the possibility of implementing PINNs as a new technique to solve black hole perturbation equations. We considered two variations of PINNs built with the DeepXDE and Pytorch packages in Python. To give some background on the underlying physics, we began by revisiting the perturbation equations for static, spherically symmetric black holes, particularly asymptotically flat and (anti-)de Sitter Schwarzschild black holes whose perturbations are described by one-dimensional Schrödinger-like eigenvalue problems. Our goal was to determine when and how PINNs can be best applied to solve these equations, which are generally difficult to solve analytically and compute the QNMs of black holes.

Since PINNs are extensions of deep neural networks, we outlined NNs in Sec. IV A, in terms of their structure and the mechanisms behind their function approximation abilities. Afterwards, PINNs were described with illustrative examples showing how physics constraints are embedded in the loss function of a NN. These constraints include the governing PDE, its associated boundary conditions, and regularization functions.

Of the two variations of PINNs considered in this work, the eigenvalue solvers were implemented to compute the QNMs of asymptotically flat Schwarzschild and near extremal nonrotating de Sitter black holes. Given that the latter scenario has known exact formulas for the QNM frequencies (given by Refs. [22–24]), we were able to reliably validate the accuracy of our NN approximations. We obtained QNM values with up to 6 digit accuracy and plots showing the evolution of the NN's approximation of the QNMs over a 100 000 epoch training phase. The plots showed that the NN's approximation quickly converged towards the expected solutions, regardless of the multipole number $\ell$ or the existence of a seed loss term in the loss function.

Regarding the more analytically intractable problems, we managed to solve the perturbation equations of asymptotically flat Schwarzschild black holes by embedding the equations themselves, the QNM boundary conditions and a seed loss term into the loss function of the eigenvalue solvers. The computed QNMs have the same level of accuracy as those obtained through Leaver's CFM [12] or Konoplya's sixth order WKB method [46] (at least up to 4 decimal places as given in the literature [18]). However, in terms of efficiency, our eigenvalue solvers take several minutes to train, compared to the few seconds to a minute it takes to generate accurate results using other techniques such as the WKB method. We also found that the efficiency of PINNs could be optimized by setting up the NN using lower values in the range of values of the hyperparameters that we tested, that is, the number of training epochs, number of training points and number of nodes per layer.

To date, we have been able to compute only the fundamental mode frequencies (i.e., $n = 0, \ell \geq 0$) that, as it turns out, are the least damped, longest-lived modes compared to higher overtones with $n > 0$. This is because we have added regularization terms that simultaneously penalize the NN for learning trivial eigenfunctions and encourage it to learn the most energetic QNMs, which happen to occur when $n = 0$ for any given $\ell$. Potential future work would seek a modification of the eigenvalue scanning mechanism similar to that introduced by Ref. [7], which will allow for the computation of higher overtones for our complex-valued QNMs.

Concerning the question of the stability of PINNs as they increase in depth, that is still very much an open problem, in general, within the literature and is in the early stages of investigation. When studying PINNs to mimic the analysis

of numerical discretization techniques, convergence and stability are related to how well the NN learns from the physical laws embodied by the governing PDEs and initial/boundary conditions [41]. It is well known that there is a bias-variance trade-off that comes with choosing the right depth of a neural network, where too deep a neural network may lead to overfitting and inefficiency. The most recent literature in laying out a theoretical framework for PINNs includes Refs. [48,49] that provide formal findings regarding PINNs and their convergence when dealing with linear problems including second-order elliptic, hyperbolic, and parabolic type PDEs. Utilizing these recent developments is important to understand and improve on PINNs in continued use to compute QNMs.

As discussed in Sec. II A, our NNs exhibit signs of instability which we suspect to be a result of the level of complexity in the loss function, which makes for a highly nonconvex optimization process [38]. This is counterintuitive to our initial expectation that PINNs can accurately solve any PDE (regardless of complexity) if they are formulated in a finite domain and their associated boundary conditions are properly set up. This was not the case for our attempts when applying eigenvalue solvers to the Reissner-Nordström case. To overcome this instability in future work, one plausible approach would be to consider the recent work in Ref. [42] that shows that a "self-scalable" activation function leads to PINNs which are less susceptible to spurious stationary points, an obstacle in highly nonconvex loss functions.

A final point to note concerning the limitations of PINNs is their relative inefficiency compared to the extant methods for computing QNMs. Further investigation needs to be done to improve the performance of the eigenvalue solvers as they currently do not surpass the efficiency of established methods such as the WKB method and the AIM. Overall, PINNs have not developed far enough to be applied broadly in the study of black hole perturbations. In conclusion to their seminal work on PINNs, Ref. [4] pointed out that this method should not be viewed as a replacement for classical numerical methods for solving PDEs, but rather as methods that can bring added merits such as implementation simplicity to accelerate the rate of testing new ideas. In a similar vein, the application of PINNs to QNMs brings at least a new angle to study the perturbation equations, even though they are not as efficient as canonical methods.

As is, the PINN approach may only work in computing the fundamental QNMs of not only four-dimensional Schwarzschild black holes, but also general dimensional Schwarzschild black holes (described in Ref. [9]) given the similarity of the differential equations. Despite the present challenges (which are expected for a burgeoning method) this approach to computing QNMs is worth pursuing further as it demonstrates the same level of accuracy as the leading existing methods.

## ACKNOWLEDGMENTS

[1] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Netw. **2**, 359 (1989).

[2] S. Wu and M. J. Er, Dynamic fuzzy neural networks—a novel approach to function approximation, IEEE Trans. Syst. Man. Cybern. B Cybern. **30**, 358 (2000).

[3] M. A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, San Francisco, CA, USA, 2015), Vol. 25.

[4] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. **378**, 686 (2019).

[5] Y. Lu and J. Lu, A universal approximation theorem of deep neural networks for expressing distributions, arXiv:2004.08867.

[6] J. Nordström and O. Alund, Neural network enhanced computations on coarse grids, J. Comput. Phys. **425**, 109821 (2021).

[7] H. Jin, M. Mattheakis, and P. Protopapas, Unsupervized neural networks for quantum eigenvalue problems, arXiv:2010.05075.

[8] K. D. Kokkotas and B. G. Schmidt, Quasinormal modes of stars and black holes, Living Rev. Relativity **2**, 2 (1999).

[9] R. A. Konoplya and A. Zhidenko, Quasinormal modes of black holes: From astrophysics to string theory, Rev. Mod. Phys. **83**, 793 (2011).

[10] E. Berti, V. Cardoso, and A. O. Starinets, Quasinormal modes of black holes and black branes, Classical Quantum Gravity **26**, 163001 (2009).

[11] S. Iyer and C. M. Will, Black-hole normal modes: A WKB approach. I. Foundations and application of a higher-order WKB analysis of potential-barrier scattering, Phys. Rev. D **35**, 3621 (1987).

[12] E. W. Leaver, An analytic representation for the quasinormal modes of Kerr black holes, Proc. R. Soc. A **402**, 285 (1985).

[13] H. T. Cho, A. S. Cornell, J. Doukas, T. R. Huang, and W. Naylor, A new approach to black hole quasinormal modes: A review of the asymptotic iteration method, Adv. Theor. Math. Phys. **2012**, 281705 (2012).

[14] M. Isi, M. Giesler, W. M. Farr, M. A. Scheel, and S. A. Teukolsky, Testing the No-Hair Theorem with GW150914, Phys. Rev. Lett. **123**, 111102 (2019).

[15] A. Ghosh, R. Brito, and A. Buonanno, Constraints on quasinormal mode frequencies with LIGO-Virgo binary black-hole observations, Phys. Rev. D **103**, 124041 (2021).

[16] S. M. Carroll, *Spacetime and Geometry: An Introduction to General Relativity* (Cambridge University Press, 2019).

[17] D. McMahon, *Quantum Field Theory Demystified* (McGraw-Hill Education, New York City, 2008).

[18] S. Iyer, Black hole normal modes: A WKB approach. II. Schwarzschild black holes, Phys. Rev. D **35**, 3632 (1987).

[19] T. Regge and J. A. Wheeler, Stability of a Schwarzschild singularity, Phys. Rev. **108**, 1063 (1957).

[20] S. Dey and S. Chakrabarti, A note on electromagnetic and gravitational perturbations of the Bardeen de Sitter black hole: Quasinormal modes and greybody factors, Eur. Phys. J. C **79**, 504 (2019).

[21] V. Cardoso and J. P. S. Lemos, Quasinormal modes of Schwarzschild anti-de Sitter black holes: Electromagnetic and gravitational perturbations, Phys. Rev. D **64**, 084017 (2001).

[22] C. Molina, Quasinormal modes of D-dimensional spherical black holes with near extreme cosmological constant, Phys. Rev. D **68**, 064007 (2003).

[23] V. Cardoso and J. P. S. Lemos, Quasinormal modes of the near extremal Schwarzschild-de Sitter black hole, Phys. Rev. D **67**, 084020 (2003).

[24] V. Ferrari and B. Mashhoon, New approach to the quasinormal modes of a black hole, Phys. Rev. D **30**, 295 (1984).

[25] B. F. Schutz and C. M. Will, Black hole normal modes: A semianalytic approach, Astrophys. J. **291**, L33 (1985).

[26] H.-P. Nollert, Quasinormal modes of Schwarzschild black holes: The determination of quasinormal frequencies with very large imaginary parts, Phys. Rev. D **47**, 5253 (1993).

[27] E. W. Leaver, Quasinormal modes of Reissner-Nordström black holes, Phys. Rev. D **41**, 2986 (1990).

[28] R. Konoplya, A. Zhidenko, and A. Zinhailo, Higher order WKB formula for quasinormal modes and grey-body factors: Recipes for quick and accurate calculations, Classical Quantum Gravity **36**, 155002 (2019).

[29] R. G. Daghigh, M. D. Green, J. C. Morey, and G. Kunstatter, Scalar perturbations of a single-horizon regular black hole, Phys. Rev. D **102**, 104040 (2020).

[30] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. **63**, 208 (2021).

[31] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, On the number of linear regions of deep neural networks, in *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger (Curran Associates, Inc., 2014), Vol. 27, https://proceedings.neurips.cc/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf.

[32] O. Ålund, G. Iaccarino, and J. Nordström, Learning to differentiate, J. Comput. Phys. **424**, 109873 (2021).

[33] K. Hornik, M. Stinchcombe, and H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. **3**, 551 (1990).

[34] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. **4**, 251 (1991).

[35] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta Numer. **8**, 143 (1999).

[36] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[37] X. Chen, R. Chen, Q. Wan, R. Xu, and J. Liu, An improved data-free surrogate model for solving partial differential equations using deep neural networks, Sci. Rep. **11**, 19507 (2021).

[38] G. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, Nat. Rev. Phys. **3**, 422 (2021).

[39] Dependent variable or eigenfunction is used in this paper rather than solution, to avoid misunderstading PINNs whose goal may not always be to find $u(\mathbf{x})$ as a solution; but, in some cases, a labeled dataset of the predictors $\mathbf{x}$ and associated $u(\mathbf{x})$ may be used to find unknown parameters in a PDE for inverse problems (for example).

[40] H.-T. Cho and C.-L. Ho, (Quasi)-exactly solvable quasinormal modes, J. Phys. A **40**, 1325 (2007).

[41] S. Cuomo, V. S. Di Cola, G. Rozza, M. Raissi, and F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, J. Sci. Comput. **92**, 88 (2022).

[42] R. Gnanasambandam, B. Shen, J. Chung, X. Yue, and Z. Kong, Self-scalable tanh (stan): Faster convergence and better generalization in physics-informed neural networks, 2022.

[43] C. Cheng and G.-T. Zhang, Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems, Water **13**, 4 (2021).

[44] G. Pöschl and E. Teller, Bemerkungen zur quantenmechanik des anharmonischen oszillators, Z. Phys. **83**, 143 (1933).

[45] M. Mattheakis, D. Sondak, A. S. Dogra, and P. Protopapas, Hamiltonian neural networks for solving equations of motion, Phys. Rev. E **105**, 065305 (2022).

[46] R. A. Konoplya, Quasinormal behavior of the *D*-dimensional Schwarzschild black hole and the higher order WKB approach, Phys. Rev. D **68**, 024018 (2003).

[47] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations, arXiv:1809.02362.

[48] Y. Shin, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, Commun. Comput. Phys. **28**, 2042 (2020).

[49] Y. Shin, Z. Zhang, and G. E. Karniadakis, Error estimates of residual minimization using neural networks for linear PDEs, arXiv:2010.08019.