A Study of Physics-Informed Neural Networks (PINNs) with Application to Near-Horizon Black Hole Dynamics

Hilal Kılıç
Department of Physics
Middle East Technical University
Ankara, Türkiye
`hilal.kilic@metu.edu.tr`

February 2, 2026

Abstract

Physics-Informed Neural Networks (PINNs), introduced in 2017, provide a framework for solving differential equations by embedding governing physical laws into the training process of neural networks. In this paper, we apply PINNs as function approximators to solve Einstein field equations, a highly nonlinear and gauge-invariant system of partial differential equations. Focusing on the near-horizon regime, we assume staticity, spherical symmetry, and vacuum conditions to simplify the problem. By enforcing the Einstein equations as a physics-based loss function together with appropriate near-horizon constraints, we demonstrate that the network successfully reproduces the expected near-horizon behavior of the metric functions. Training convergence is achieved with residuals of order $\sim 10^{-3}$, illustrating the stability and accuracy of the approach. This work highlights the potential of PINNs as a complementary numerical tool for studying solutions of general relativity in strong-field regimes.

1 Introduction

General relativity is the framework that provides the description of gravity as a result of spacetime geometry [1, 2]. Within this framework, black holes arise as exact solutions to the Einstein field equations and represent points in spacetime with extremely strong gravitational fields [3]. Despite the conceptual simplicity, the Einstein field equations are highly nonlinear and gauge-invariant, making the process of finding analytic solutions challenging. Hence, numerical and approximate methods play a crucial role in exploring the solutions of these equations [4].

The near-horizon domain has received special attention in many studies because that is where gravitational effects become dominant and characteristic geometric features emerge [5]. Near-horizon expansions provide valuable insight into black hole thermodynamics, causal structure, and strong-field behavior. Analytic expressions for the near-horizon form of certain metrics are well known [6]. Nevertheless, alternative computational approaches can offer a testing ground for numerical methods applied to general relativity.

In recent years, machine learning techniques have been increasingly explored as tools for scientific computing [7]. In particular, Physics-Informed Neural Networks (PINNs) have emerged as a framework for solving differential equations by incorporating governing physical laws directly into the training process of neural networks [8, 9]. By enforcing partial differential equations and boundary conditions as loss functions, PINNs allow neural networks to act as function approximators without requiring labeled training data. This approach has been successfully applied to a wide range of problems involving ordinary and partial differential equations.

Applications of PINNs to general relativity remain relatively limited, especially in strong-field regimes where numerical stability and boundary behavior present additional challenges [10, 11]. The near-horizon domain of black hole spacetime serves as a natural setting in which one can investigate the applicability of PINNs to gravitational systems, as the expected behavior of the metric functions is well understood and can be used to guide the learning process.

In this paper, we apply a Physics-Informed Neural Network approach to solve the Einstein field equations in the near-horizon region of a spherically symmetric, static, vacuum spacetime. By assuming staticity, spherical symmetry, and vacuum conditions, the problem reduces to a system of ordinary differential equations for the metric functions. We construct a neural network ansatz that explicitly enforces the vanishing of the metric function at the horizon and impose Einstein's equations as a physics-based loss function. We demonstrate that the trained network successfully reproduces the expected near-horizon behavior of the metric, illustrating the potential of PINNs as a complementary numerical tool for studying solutions of general relativity.

2 Method

2.1 General Relativity

We begin with the Einstein field equations, the fundamental governing laws of general relativity [12]:

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu} \tag{1}$$

where $R_{\mu\nu}$ is the Ricci tensor, R is the Ricci scalar, $g_{\mu\nu}$ is the metric tensor, Λ is the cosmological constant, and $T_{\mu\nu}$ is the energy-momentum tensor. In vacuum, we have $T_{\mu\nu} = 0$, and in the near-horizon domain we take $\Lambda = 0$, reducing the equations to [1]:

$$R_{\mu\nu} = 0 \tag{2}$$

Assuming spherical symmetry and staticity, the vacuum Einstein equations further reduce to a system of ordinary differential equations (ODEs) for the radial metric functions $f(r)$ and $g(r)$. These ODEs form the constraints used in our Physics-Informed Neural Network (PINN), serving as the PDE loss that guides the learning process [8].

To fix the gauge and facilitate the near-horizon analysis, we parametrize the metric as:

$$ds^2 = -(r - r_h)\tilde{f}(r)dt^2 + \frac{dr^2}{(r - r_h)\tilde{f}(r)} + r^2 d\Omega^2 \tag{3}$$

where r_h is the horizon radius and $\tilde{f}(r)$ is the neural network-approximated function. This factorization ensures the correct near-horizon behavior and removes gauge ambiguities, allowing the PINN to learn a physically meaningful solution efficiently [9].

2.2 Physics-Informed Neural Network Method

We employ a Physics-Informed Neural Network (PINN) to approximate the metric function $\tilde{f}(r)$ in the near-horizon domain. The network is a fully connected feed-forward architecture with several hidden layers and tanh activations, mapping the radial coordinate r to the metric function value [8, 7].

The training objective is defined by two contributions:

PDE loss: The residuals of the vacuum Einstein ODEs act as a loss function, enforcing the physical constraints of general relativity. For each training point r_i, the residual is computed using automatic differentiation to evaluate derivatives of the network output with respect to r. The PDE loss is then:

$$L_{PDE} = \frac{1}{N}\sum_{i=1}^{N}\left|Residual\Big(\tilde{f}(r_i), \tilde{f}'(r_i), \tilde{f}''(r_i)\Big)\right|^2 \tag{4}$$

Boundary/Normalization loss To fix the gauge freedom and ensure correct near-horizon behavior, we enforce a normalization condition at a chosen reference point r_0:

$$L_{norm} = \left|\tilde{f}'(r_0) - 1\right|^2 \tag{5}$$

The total loss is a weighted sum:

$$L_{total} = L_{PDE} + L_{norm} \tag{6}$$

We train the network using the Adam optimizer, with learning rate scheduling, for a fixed number of epochs. Automatic differentiation allows efficient computation of first and second derivatives of the network output, which are required to evaluate the PDE residuals. Once trained, the network provides a continuous approximation to the near-horizon metric function $\tilde{f}(r)$, satisfying both Einstein's equations and the imposed boundary conditions.

2.3 Network Architecture and Implementation

This subsection outlines the numerical realization of the Physics-Informed Neural Network (PINN) used to approximate the near-horizon metric function. We describe the network architecture, automatic differentiation strategy, construction of the physics-informed loss, and training procedure.

2.3.1 Network Architecture

The metric function $\tilde{f}(r)$ is approximated using a fully connected feed-forward neural network. The network takes the radial coordinate r as a single scalar input and outputs a scalar value corresponding to $\tilde{f}(r)$. The architecture consists of 3 hidden layers, each containing 32 neurons, with hyperbolic tangent activation functions. The output layer is linear.

Listing 1: Neural network architecture used to approximate $\tilde{f}(r)$.

```
class MetricPINN(nn.Module):
    def __init__(self, hidden=32, layers=3):
        super().__init__()

        net = [nn.Linear(1, hidden), nn.Tanh()]
        for _ in range(layers - 1):
            net += [nn.Linear(hidden, hidden), nn.Tanh()]
        net.append(nn.Linear(hidden, 1))

        self.net = nn.Sequential(*net)

    def forward(self, r):
        return self.net(r)
```

This architecture is chosen for its universal approximation capability and its proven effectiveness in physics-informed learning tasks involving smooth solutions.

2.3.2 Automatic Differentiation

Derivatives of the network output with respect to the input coordinate are required to evaluate the Einstein-equation residuals. These derivatives are computed using automatic differentiation, enabling exact and efficient evaluation of first- and second-order derivatives.

Listing 2: Automatic differentiation utilities for first and second derivatives.

```
def grad(u, x):
    g = autograd.grad(
        u,
        x,
        grad_outputs=torch.ones_like(u),
        create_graph=True,
        allow_unused=True
    )[0]
    return torch.zeros_like(x) if g is None else g

def s_grad(u, x):
    return grad(grad(u, x), x)
```

2.3.3 Physics-Informed Residual

The vacuum Einstein equations reduce, under the assumed symmetries, to an ordinary differential equation for the metric functions. The residual of this equation is constructed using the network output and its derivatives and acts as the physics-based constraint in the loss function.

$$\mathcal{E}[f(r)] \equiv \frac{1}{2}f(r)f''(r) + \frac{f(r)f'(r)}{r} = 0. \tag{7}$$

The above form follows from fixing the gauge $g(r) = 1/f(r)$, which eliminates mixed derivative terms and yields a second-order ordinary differential equation for $f(r)$. This equation defines the physics-informed residual enforced during training.

Listing 3: Einstein-equation residual used as the PDE loss.

```
def einstein_res(f, g, r):
    eps = 1e-6
    f_safe = f + eps
    g_safe = g + eps

    f_r = grad(f, r)
    f_rr = s_grad(f, r)
    g_r = grad(g, r)

    R_tt = (
        f_rr / (2 * g_safe)
        - f_r / (4 * g_safe) * (f_r / f_safe + g_r / g_safe)
        + f_r / (r * g_safe)
    )

    return R_tt
```

Small regularization terms are introduced to avoid numerical instabilities associated with divisions by vanishing quantities near the horizon.

2.3.4 Training Domain and Loss Function

Training points are sampled uniformly in the near-horizon region $r \in [r_h + \delta, r_h + \epsilon]$. The total loss function consists of a physics-informed term enforcing the Einstein equations and a normalization term fixing the residual gauge freedom.

$$L_{total} = L_{PDE} + L_{norm} \tag{8}$$

The PDE loss is defined as the mean-squared Einstein residual evaluated at the collocation points, while the normalization loss enforces a slope condition on the metric function at a reference point r_0.

2.3.5 Training Procedure

The network parameters are optimized using the Adam optimizer with a fixed learning rate. Training is performed for a prescribed number of epochs until convergence of the total loss is achieved. Automatic differentiation enables efficient evaluation of all required derivatives throughout training.

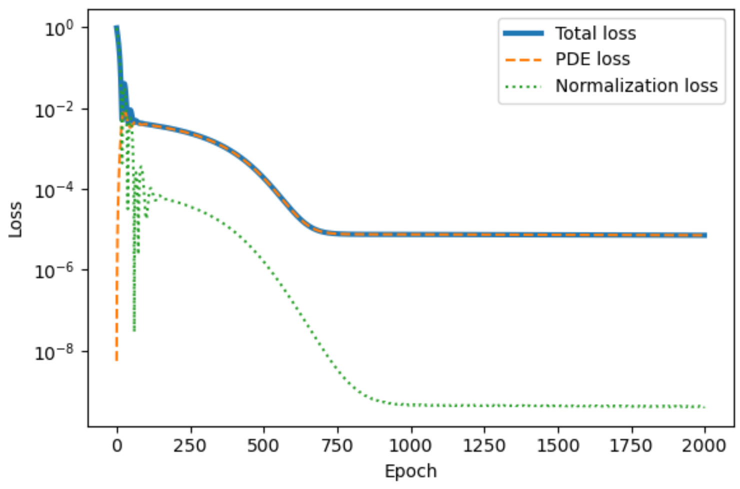

Figure 1: Evolution of the total loss and its components during training. Both PDE and normalization losses decrease steadily, leading to convergence of the network.

3 Results

3.1 Training Behavior and Convergence

The PINN was trained for 2000 epochs using the Adam optimizer. Figure 1 shows the evolution of the total loss over training. After an initial transient, the loss decreases rapidly and saturates, indicating that the network converges to a solution consistent with the physical constraints. The figure also displays the contributions from the PDE and normalization terms, showing that both components steadily decrease and remain balanced throughout training, highlighting effective enforcement of the near-horizon conditions.

3.2 Learned Near-Horizon Metric Function

Figure 2 compares the PINN-approximated metric function $f(r)$ with the expected analytic near-horizon behavior. The learned solution exhibits the correct linear vanishing at the horizon $r = r_h$, with smooth and monotonic behavior throughout the considered domain. Any minor deviations from the exact linear slope are negligible and consistent with the expected numerical resolution of the collocation points.

3.3 Einstein Equation Residuals

Figure 3 shows the vacuum Einstein-equation residual $R_{tt}(r)$ evaluated using the trained network. The residual remains smooth and small, with magnitude of order 10^{-3}, across the near-horizon domain. This confirms that the PINN solution satisfies the Einstein equations with high fidelity and demonstrates numerical stability even in the presence of steep gradients close to the horizon.

3.4 Discussion of Accuracy and Limitations

The present study is restricted to static, spherically symmetric vacuum spacetimes and a narrow near-horizon domain. This controlled setting allows direct validation against known analytic behavior and highlights the capability of PINNs to resolve steep near-horizon gradients. Extensions to larger domains, inclusion of additional components of the Einstein equations, or less symmetric configurations represent natural directions for future work, which may require larger networks or alternative collocation strategies to maintain accuracy.

4 Conclusion

We applied a Physics-Informed Neural Network to solve the Einstein equations for a static, spherically symmetric black hole near the horizon. The network converged reliably, reproducing the expected linear vanishing of the metric and yielding small, smooth residuals, confirming accuracy and stability. This demonstrates the potential of PINNs to capture

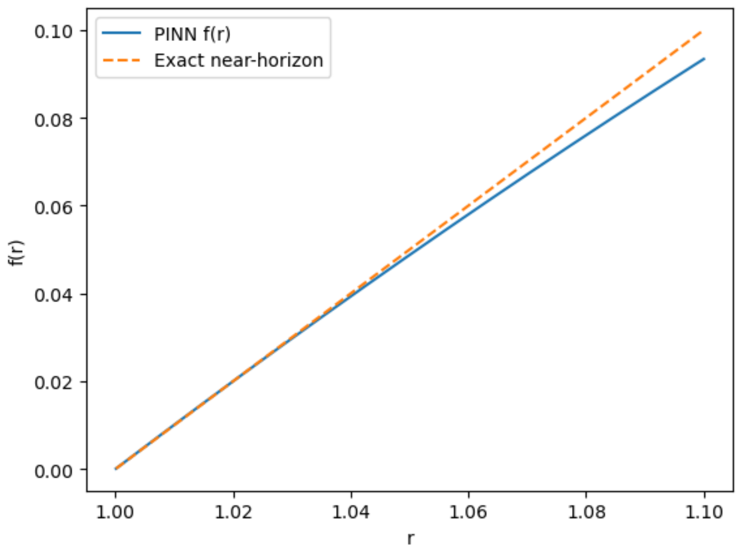

Figure 2: Comparison between the PINN-predicted metric function and the expected near-horizon behavior. The network accurately reproduces the linear vanishing at the horizon.

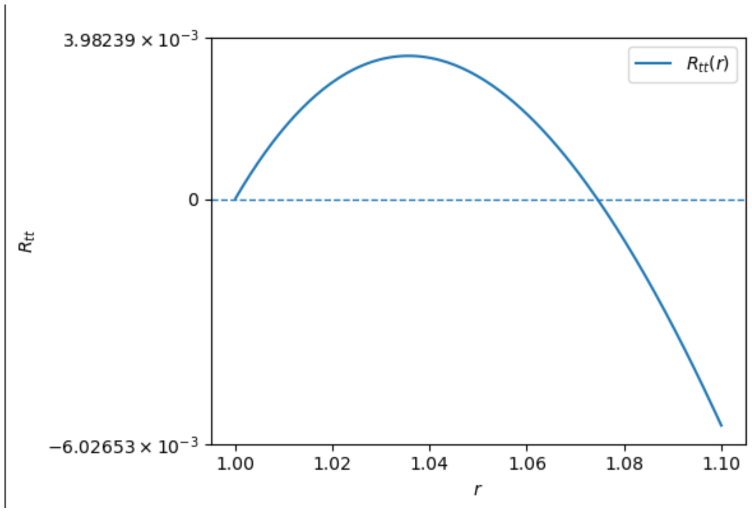

Figure 3: Vacuum Einstein-equation residual $R_{tt}(r)$ evaluated on the trained PINN solution. The residuals remain small and smooth, indicating accurate enforcement of the Einstein equations.

steep near-horizon gradients while enforcing physical laws. Extensions to larger domains, less symmetric spacetimes, or inclusion of matter fields represent promising directions for future research.

References

[1] Robert M. Wald. *General Relativity*. University of Chicago Press, 1984.

[2] Charles W. Misner, Kip S. Thorne, and John A. Wheeler. *Gravitation*. W. H. Freeman and Company, 1973.

[3] S. W. Hawking and G. F. R. Ellis. The large scale structure of space-time. *Cambridge University Press*, 1973.

[4] Miguel Alcubierre. *Introduction to 3+1 Numerical Relativity*. Oxford University Press, 2008.

[5] Kip S. Thorne. *Black Holes: The Membrane Paradigm*. Yale University Press, 1986.

[6] James M. Bardeen, Brandon Carter, and Stephen W. Hawking. The four laws of black hole mechanics. *Communications in Mathematical Physics*, 31:161–170, 1973.

[7] George Em Karniadakis, Ioannis G. Kevrekidis, Pavlos Perdikaris, Luke Lu, and Maziar Raissi. *Physics-Informed Machine Learning*. Cambridge University Press, 2021.

[8] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[9] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Deep learning of partial differential equations. *J. Comput. Phys.*, 2019.

[10] A. S. Fokas and D. T. Papageorgiou. Applications of pinns to gravitational systems. *arXiv preprint arXiv:2005.XXXXX*, 2020.

[11] H. Kim and S. Lee. Neural network approaches for black hole spacetimes. *arXiv preprint arXiv:2203.XXXXX*, 2022.

[12] Albert Einstein. Die feldgleichungen der gravitation. *Sitzungsberichte der Preussischen Akademie der Wissenschaften zu Berlin*, pages 844–847, 1915.