Hybrid Numerical and Machine Learning Approaches for Solving Einstein Constraint Equations

Ahmed M. Al-Haysah
Department of Mathematics
Faculty of Education and Sciences-Rada'a
Albaydha University
Albaydha, Yemen

ABSTRACT
The Einstein constraint equations form a nonlinear and underdetermined elliptic system whose solution requires specifying appropriate gauge, conformal, and freely chosen geometric data. In this work, we present a hybrid numerical–machine learning framework for solving the Hamiltonian and momentum constraints using a combination of classical solvers, Physics-Informed Neural Networks (PINNs), and Deep Operator Networks (DeepONets). We clarify the mathematical structure of the constraint system, explicitly describe the conformal background and freely chosen components of the initial data, and construct a well-posed elliptic formulation suitable for numerical treatment. To demonstrate feasibility, we implement PINN and DeepONet models for the conformally flat, time-symmetric vacuum constraint and compare the neural solutions with a classical finite-difference reference. The results show that machine-assisted PDE solvers can approximate the constraint equations with competitive accuracy while offering mesh-free flexibility. This revised formulation provides a concrete basis for future large-scale simulations in numerical relativity.

Keywords
Einstein Constraint Equations; Numerical Relativity; PDE Solvers; Machine Learning; PINNs; DONs; Initial Data.

MSC 2010 No.: 83C05, 35Q75, 35J60, 65M60, 68T07.

1. INTRODUCTION
The Einstein field equations (EFEs), formulated in 1915, are a system of ten interrelated nonlinear partial differential equations that describe the gravitational interaction as the geometry of space-time. When recast in the $3+1$ Arnowitt–Deser–Misner (ADM) formalism, introduced in the late 1950s, the EFEs naturally split into two classes: the evolution equations and the constraint equations [4]. The latter, composed of the Hamiltonian and momentum constraints, form a set of nonlinear elliptic partial differential equations that must be satisfied on each spatial hypersurface in order for the initial data to be physically valid.

Since the 1970s, numerical relativity has sought to solve the Einstein equations using computational techniques. A major challenge has been the accurate and stable solution of the constraint equations, especially in scenarios involving strong-field gravity such as black holes, neutron stars, and cosmological space-times [4]. Early methods included conformal decomposition techniques such as the Lichnerowicz–York method, which enabled the construction of initial data sets under simplifying symmetry assumptions.

Over the following decades, classical numerical methods such as finite difference, finite element, and spectral methods were refined and applied to increasingly complex geometries [1]. While these methods have been successful, they often require problem-specific tuning, mesh generation, and can suffer from instabilities or inefficiencies in high-dimensional, nonlinear settings.

In recent years, the emergence of machine learning methods—particularly Physics-Informed Neural Networks (PINNs) and operator-learning frameworks—has opened new avenues for solving partial differential equations in scientific computing [10, 5]. These approaches offer the potential to bypass meshing, generalize across problem instances, and handle high-dimensional parameter spaces more effectively than classical methods.

Several researchers have begun applying PINNs and neural PDE solvers to problems in general relativity, including the approximation of black hole metrics, scalar field systems, and simplified evolution equations [3, 13]. Similarly, operator-learning approaches such as DeepONets have demonstrated strong generalization capabilities for parametric families of PDEs, making them attractive for repeated or multi-parameter relativistic simulations [7, 8 ,2]. Nevertheless, the application of these techniques specifically to the Einstein constraint equations remains relatively limited, particularly within a systematic and comparative framework [15].

It is important to emphasize that the Hamiltonian and momentum constraints alone do not determine all components of (γ_{ij}, K_{ij}). Since the spatial metric contains six degrees of freedom and the extrinsic curvature contains another six, the four constraint equations are insufficient to fix the twelve unknowns. Consequently, any practical formulation must specify a choice of conformal metric, slicing condition, gauge variables, and freely specifiable components of K_{ij}. In this work, we adopt the standard conformal method, selecting a conformally flat background $\tilde{\gamma}_{ij} = \delta_{ij}$ and time-symmetric data $K_{ij} = 0$, which reduces the system to a well-posed elliptic problem [4].

The objective of this work is to develop and analyze hybrid numerical schemes for solving the Einstein constraint equations by combining traditional PDE solvers with machine-assisted approaches. We focus in particular on PINNs and DeepONets, evaluating their performance relative to classical finite-difference solutions on benchmark initial data problems. The ultimate goal is to enhance computational flexibility and robustness while maintaining physical consistency in the generation of relativistic initial data.

2. ADM FORMALISM AND CONSTRAINT EQUATIONS
The Arnowitt-Deser-Misner (ADM) formalism provides a $3+$

1 decomposition of the space-time manifold $(\mathcal{M}, g_{\mu\nu})$ into space-like hypersurfaces Σ_t labeled by coordinate time t. The four-metric is expressed in terms of the induced three-metric γ_{ij}, the lapse function $\alpha(t, x^i)$, and the shift vector $\beta^i(t, x^i)$. The extrinsic curvature K_{ij}, representing the embedding of each hypersurface in the full space-time, is defined as

$$K_{ij} = -\frac{1}{2\alpha}\left(\partial_t \gamma_{ij} - \mathcal{L}_\beta \gamma_{ij}\right),$$

where $\mathcal{L}_\beta$ denotes the Lie derivative along β^i.

Under this decomposition, the Einstein field equations split into evolution and constraint equations. The Hamiltonian and momentum constraints read:

$$\mathcal{H} \equiv R + K^2 - K_{ij}K^{ij} = 16\pi\rho, \qquad (1)$$
$$\mathcal{M}^i \equiv D_j(K^{ij} - \gamma^{ij}K) = 8\pi j^i, \qquad (2)$$

where R is the Ricci scalar of the spatial metric γ_{ij}, D_j is the covariant derivative compatible with γ_{ij}, $K = \gamma^{ij}K_{ij}$ is the trace of the extrinsic curvature, and (ρ, j^i) represent the energy and momentum densities of matter. Equations (1) and (2) constitute a nonlinear elliptic PDE system for (γ_{ij}, K_{ij}) constrained by the physical matter content.

To facilitate numerical solution, the York–Lichnerowicz conformal decomposition is employed:

$$\gamma_{ij} = \psi^4 \tilde{\gamma}_{ij}, \qquad (3)$$
$$K_{ij} = A_{ij} + \frac{1}{3}\gamma_{ij}K, \qquad (4)$$

where ψ is the conformal factor, $\tilde{\gamma}_{ij}$ is the conformal metric, and A_{ij} is the trace-free part of the extrinsic curvature. Maximal slicing ($K = 0$) and a conformally flat background ($\tilde{\gamma}_{ij} = \delta_{ij}$) are commonly adopted, reducing the constraint equations to elliptic PDEs for ψ and the longitudinal part of A_{ij}, solvable with appropriate boundary conditions.

The ADM constraint system exhibits the following mathematical properties:

- **Nonlinearity:** Due to terms such as $K_{ij}K^{ij}$ and ψ^{-7} in the conformal formulation.
- **Ellipticity:** The equations form a nonlinear elliptic system, suitable for finite element and variational methods.
- **Well-posedness:** Under reasonable assumptions on matter sources and conformal data, existence and uniqueness results are guaranteed.

3. HAMILTONIAN AND MOMENTUM CONSTRAINTS

In the 3+1 ADM decomposition of general relativity, the Einstein field equations separate into two sets: evolution equations that describe the dynamics of the three-metric γ_{ij} and extrinsic curvature K_{ij}, and constraint equations that must be satisfied on every space-like hyper-surface Σ_t. The latter are the Hamiltonian and momentum constraints.

3.1 Hamiltonian Constraint

The Hamiltonian constraint arises from projecting the Einstein equations twice along the normal vector n^μ to the hyper-surface Σ_t. It reads:

$$\mathcal{H} := R + K^2 - K_{ij}K^{ij} = 16\pi\rho, \qquad (5)$$

where:

- R is the scalar curvature of the induced spatial metric γ_{ij},
- $K = \gamma^{ij}K_{ij}$ is the trace of the extrinsic curvature?
- $K_{ij}K^{ij}$ represents the squared norm of the extrinsic curvature tensor,
- $\rho = T_{\mu\nu}n^\mu n^\nu$ is the energy density as measured by an observer normal to the hyper-surface?

Equation (5) serves as a nonlinear elliptic constraint on the geometry and extrinsic curvature of the hyper-surface.

3.2 Momentum Constraint

The momentum constraint arises from projecting the Einstein equations once along the normal vector and once tangentially to the hyper-surface. It is given by:

$$\mathcal{M}^i := D_j\left(K^{ij} - \gamma^{ij}K\right) = 8\pi j^i, \qquad (6)$$

where:

- D_j is the covariant derivative associated with γ_{ij},
- $j^i = -T_{\mu\nu}n^\mu\gamma^{\nu i}$ is the momentum density of matter relative to Σ_t,
- $K^{ij} - \gamma^{ij}K$ is the trace-free part of the extrinsic curvature.

Equation (6) represents a system of three coupled nonlinear equations (in 3 spatial dimensions), adding further constraints on the allowed initial data for general relativity.

3.3 Physical Interpretatio

Together, equations (5) and (6) ensure that the initial data (γ_{ij}, K_{ij}) are compatible with the Einstein field equations. They play a crucial role in:

- Constructing physically realistic initial data sets,
- Ensuring the consistency of space-time evolution,
- Modeling strong-field configurations such as binary black holes, neutron stars, and cosmological space-times.

In the presence of matter, these constraints enforce energy and momentum conservation on the initial slice, as encoded by the stress-energy tensor $T_{\mu\nu}$.

4. CLASSICAL NUMERICAL METHODS

Solving the Einstein constraint equations numerically has been an active area of research in numerical relativity for several decades. Due to their nonlinear elliptic nature, these equations are well-suited to well-established numerical techniques such as finite difference, finite element, and spectral methods.

4.1 Finite Difference Methods (FDM)

Finite difference methods approximate derivatives using discretized grid-based stencils. For a second-order PDE, the Laplace operator is approximated as:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}.$$

FDM is simple to implement and computationally inexpensive, especially on structured grids. However, it suffers from:

- Limited geometric flexibility,
- Difficulties in applying boundary conditions on complex domains,
- Reduced accuracy near irregular boundaries.

4.2 Finite Element Methods (FEM)

FEM is a variational method particularly effective for elliptic PDEs on arbitrary geometries. The domain is decomposed into elements (e.g., triangles or tetrahedra), and the solution is approximated using basis functions defined locally on each element. FEM offers:

- High geometric adaptability,
- Systematic refinement and convergence analysis,
- Strong mathematical foundations for well-posedness and stability.

It has been widely used for solving the constraint equations in complex black hole and neutron star initial data problems. However, FEM requires mesh generation and matrix assembly, which can be computationally expensive in high dimensions.

4.3 Spectral Methods

Spectral methods expand the solution in terms of global basis functions (e.g., Chebyshev or Fourier polynomials). They provide exponential convergence for smooth problems and have been particularly successful in solving the constraint equations under symmetry assumptions. Their advantages include:

- Extremely high accuracy for smooth data,
- Efficient solvers for structured domains.

However, spectral methods suffer from:

- Poor performance in non-smooth or highly localized data,
- Limited flexibility in irregular geometries,
- Difficulties in resolving sharp gradients or discontinuities.

4.4 Challenges in Classical Methods

Despite their success, classical methods face several challenges in the context of general relativity:

- **Nonlinearity:** The constraint equations are strongly nonlinear, requiring iterative solvers (e.g., Newton–Raphson) that depend on good initial guesses.
- **Geometrical Complexity:** Domains involving multiple black holes or dynamic matter distributions often lack symmetry and require adaptive meshing.
- **Scalability:** High-resolution simulations are computationally expensive, especially in 3D.

These challenges motivate the exploration of alternative and hybrid methods, such as machine-assisted solvers, to address the limitations of purely classical approaches.

5. MACHINE-ASSISTED PDE SOLVERS

Recent progress in scientific machine learning has introduced powerful data-driven approaches for solving partial differential equations (PDEs), offering complementary capabilities to traditional numerical solvers. These methods have demonstrated particular promise for nonlinear, high-dimensional, and geometrically complex problems, making them relevant for applications in numerical relativity and the Einstein constraint equations.

Physics-Informed Neural Networks (PINNs) represent one of the most widely used frameworks in this area. Instead of relying on labeled datasets, PINNs incorporate the governing physical laws directly into the neural network loss function. This is achieved by embedding the PDE residuals, boundary conditions, and—when applicable—initial conditions within a unified optimization objective. Automatic differentiation is employed to compute spatial derivatives of the neural network output, providing machine-precision evaluation of complex differential operators. PINNs offer several advantages, including mesh-free formulation, geometric flexibility, and the ability to handle high-dimensional inputs. However, they are also known to exhibit slow convergence, sensitivity to hyperparameter choices, and difficulty in resolving sharp gradients in strongly nonlinear regimes.

Another prominent architecture is the class of Deep Operator Networks (DONs), such as DeepONet, which focuses on learning nonlinear mappings between function spaces rather than approximating a single PDE solution. These models employ a branch network to encode input functions (e.g., source terms or geometry configurations) and a trunk network that processes evaluation points in the domain. Their combination yields a flexible operator that can generalize efficiently to many PDE instances, enabling rapid inference once training is complete. DONs are therefore appealing for applications that require solving families of related PDE problems repeatedly.

Machine-assisted PDE solvers have already been explored in several simplified general relativity settings, including the approximation of static black hole metrics, scalar field dynamics, and cosmological evolution scenarios. Nevertheless, their application to the Einstein constraint equations remains relatively limited, despite the potential benefits in avoiding mesh generation, handling complex topologies, and improving scalability. Key challenges include training instability, high computational cost—particularly for fully three-dimensional problems—and the absence of formal convergence guarantees when compared to classical numerical schemes.

Despite these limitations, the integration of PINNs, DONs, and related machine-learning-based solvers with established numerical relativity techniques represents a promising direction. Their flexibility, automation potential, and ability to operate in complex geometries suggest that machine-assisted PDE solvers may ultimately offer valuable tools for generating high-quality initial data for Einstein's equations and advancing the capabilities of modern numerical relativity.

6. PHYSICS-INFORMED NEURAL NETWORKS (PINNS)

Physics-Informed Neural Networks (PINNs) constitute a class of deep learning frameworks designed to approximate solutions of partial differential equations (PDEs) by embedding the governing physical laws directly into the training objective. Introduced by Raissi et al., PINNs exploit automatic differentiation to compute derivatives of neural network outputs, enabling the enforcement of PDE operators, boundary conditions, and—in time-dependent settings—initial

conditions without requiring labeled data.

To motivate the formulation, consider a PDE defined on a domain $\Omega \subset \mathbb{R}^d$ with governing operator $\mathcal{N}$ acting on an unknown field $u(\mathbf{x})$:

$$\mathcal{N}[u](\mathbf{x}) = 0, \qquad \mathbf{x} \in \Omega,$$

subject to boundary conditions of the form

$$\mathcal{B}[u](\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega.$$

A neural network approximation $u_\theta(\mathbf{x})$ is constructed to represent the solution, with trainable parameters θ. The learning objective seeks to minimize a composite loss combining PDE residuals and boundary condition violations:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \mathcal{L}_{\text{BC}}(\theta),$$

where

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r}\sum_{i=1}^{N_r}\left|\mathcal{N}[u_\theta](\mathbf{x}_r^i)\right|^2, \qquad \mathcal{L}_{\text{BC}} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left|\mathcal{B}[u_\theta](\mathbf{x}_b^i)\right|^2.$$

The training points $\{\mathbf{x}_r^i\}$ enforce the PDE in the interior of the domain, while $\{\mathbf{x}_b^i\}$ impose the required boundary behavior.

A central advantage of PINNs lies in the use of automatic differentiation (AD) to evaluate spatial derivatives of the network output with respect to its inputs. AD computes these derivatives exactly through the computational graph rather than relying on finite-difference approximations. For a neural approximation $u_\theta(\mathbf{x})$, first and second derivatives can be expressed generically as

$$\frac{\partial u_\theta}{\partial x_j} = \text{autograd.grad}(u_\theta, x_j), \qquad \frac{\partial^2 u_\theta}{\partial x_j^2} = \text{autograd.grad}\left(\frac{\partial u_\theta}{\partial x_j}, x_j\right),$$

enabling straightforward construction of complex operators such as gradients, divergences, and Laplacians.

The benefits of PINNs include their mesh-free nature, flexibility in handling diverse boundary conditions, and their ability to generalize to high-dimensional problems where conventional numerical solvers may face prohibitive computational demands. However, challenges remain, including sensitivity to hyper-parameter choices, slow convergence for strongly nonlinear PDEs, and substantial computational cost when addressing fully three-dimensional problems.

In the context of general relativity, PINNs can be directly applied to the nonlinear elliptic Einstein constraint equations by embedding the Hamiltonian and momentum constraint operators into the PDE loss term:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N}\sum_{i=1}^{N}\left[\mathcal{H}(x_i)^2 + \sum_{j=1}^{3}\mathcal{M}_j(x_i)^2\right],$$

allowing the model to learn physically admissible initial data without explicit mesh generation. This provides a compelling alternative to classical elliptic solvers, particularly in domains with irregular geometries or complex boundary structures.

Example 1.

```
import torch
import torch.nn as nn
import numpy as np

class PINN(nn.Module):
    def __init__(self, layers):
        super(PINN, self).__init__()
        self.model = nn.Sequential()
        for i in range(len(layers)-1):
self.model.add_module(f"layer_{i}", nn.Linear(layers[i],
layers[i+1]))
            if i < len(layers)-2:
 self.model.add_module(f"tanh_{i}", nn.Tanh())

    def forward(self, x):
        return self.model(x)

# PDE loss (Laplacian example)
def pde_loss(model, x):
    x.requires_grad_(True)
    u = model(x)
    grads = torch.autograd.grad(u, x,
grad_outputs=torch.ones_like(u),
          create_graph=True, retain_graph=True)[0]
    u_x, u_y = grads[:, 0:1], grads[:, 1:2]
    u_xx = torch.autograd.grad(u_x, x, torch.ones_like(u_x),
          create_graph=True, retain_graph=True)[0][:, 0:1]
    u_yy = torch.autograd.grad(u_y, x, torch.ones_like(u_y),
          create_graph=True, retain_graph=True)[0][:, 1:2]
    return torch.mean((u_xx + u_yy)**2)
```

7. LOSS FUNCTIONS AND AUTOMATIC DIFFERENTIATION IN PINNS

Physics-Informed Neural Networks (PINNs) incorporate the governing differential equations, boundary conditions, and physical constraints directly into the training objective, allowing the model to learn solutions without requiring labeled data. In this framework, the Einstein constraint equations are enforced by minimizing their residuals using automatic differentiation within the neural network architecture.

The total loss function in a PINN model is constructed as a weighted combination of multiple components, typically including the PDE residual loss, boundary condition loss, and—when applicable—initial condition loss:

$$\mathcal{L} = \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}.$$

Here, $\mathcal{L}_{\text{PDE}}$ penalizes violations of the Einstein constraints, $\mathcal{L}_{\text{BC}}$ enforces geometric or asymptotic boundary conditions, and $\mathcal{L}_{\text{IC}}$ applies only in time-dependent problems. The coefficients λ_i balance the contributions of each component during training.

A key advantage of PINNs is their reliance on automatic differentiation (AD) instead of finite-difference approximations. Deep learning frameworks such as TensorFlow and PyTorch compute derivatives of the neural network output $\hat{u}(x)$ exactly through the computational graph. For example, first- and second-order spatial derivatives are obtained as:

$$\frac{\partial \hat{u}}{\partial x} = \text{autograd.grad}(\hat{u}, x), \qquad \frac{\partial^2 \hat{u}}{\partial x^2} = \text{autograd.grad}\left(\frac{\partial \hat{u}}{\partial x}, x\right).$$

This capability allows complex operators—such as those arising in the Hamiltonian and momentum constraints of general relativity—to be evaluated efficiently and accurately at

collocation points.

For the Einstein constraint equations, the PDE residual loss is defined by evaluating the Hamiltonian constraint $\mathcal{H}$ and momentum constraints $\mathcal{M}_j$ at N collocation points $\{x_i\}$:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N}\sum_{i=1}^{N}\left[\mathcal{H}(x_i)^2 + \sum_{j=1}^{3}\mathcal{M}_j\,(x_i)^2\right].$$

This loss encourages the neural network predictions for g_{ij} and K_{ij} (or their conformal counterparts) to satisfy the geometric constraint equations throughout the spatial domain.

In practice, training stability and convergence in PINNs are improved by using additional regularization strategies, such as adaptive weighting of the different loss components, gradient normalization to prevent imbalance, and residual-based adaptive refinement (RAR) for dynamic resampling of collocation points in regions with large PDE residuals.

Together, this loss formulation and the AD-based evaluation of differential operators provide a flexible and efficient mechanism for enforcing the nonlinear geometric constraints inherent in general relativity within a neural-network-based solution framework.

8. METHODOLOGY

This section describes in detail the proposed hybrid methodology for solving the Einstein constraint equations using Physics-Informed Neural Networks (PINNs), with comparisons to classical numerical solvers. The methodology is organized around the mathematical formulation of the constraints, the neural representation of the unknown fields, sampling strategies, loss construction, training procedures, and evaluation criteria.

8.1 Mathematical Formulation of the Constraints

We consider the Einstein constraint equations posed on a spacelike hypersurface $\Sigma \subset \mathcal{M}$. In the ADM formulation, these consist of the Hamiltonian constraint

$$\mathcal{H}(g_{ij}, K_{ij}) = R + K^2 - K_{ij}K^{ij} - 16\pi\rho = 0,$$

and the momentum constraints

$$\mathcal{M}^i(g_{ij}, K_{ij}) = D_j(K^{ij} - \gamma^{ij}K) - 8\pi j^i = 0, \quad i = 1,2,3.$$

To obtain a well-posed elliptic system suitable for numerical treatment, we adopt the York–Lichnerowicz conformal decomposition. In the numerical experiments presented in this work, we restrict attention to a conformally flat, vacuum, time-symmetric configuration, characterized by

$$\tilde{\gamma}_{ij} = \delta_{ij}, \qquad K_{ij} = 0, \qquad \rho = 0, \qquad j^i = 0.$$

Under these assumptions, the momentum constraints are identically satisfied, and the Hamiltonian constraint reduces to a nonlinear elliptic equation for the conformal factor ψ. In the vacuum and time-symmetric case considered here, this equation simplifies to

$$\nabla^2\psi = 0,$$

subject to asymptotically flat boundary conditions $\psi \to 1$ as $r \to \infty$.

8.2 Neural Representation of the Solution

The unknown conformal factor $\psi(x)$ is approximated by a fully connected neural network $\hat{\psi}_\theta(x)$ with trainable parameters θ. The network defines a mapping

$$\hat{\psi}_\theta : \Omega \subset \mathbb{R}^3 \to \mathbb{R},$$

where Ω denotes the spatial computational domain. Smooth activation functions are employed to ensure that the network output is differentiable to the required order.

All spatial derivatives appearing in the Hamiltonian constraint are computed using automatic differentiation, allowing the Laplace operator to be evaluated exactly with respect to the neural network parameters. This avoids the discretization errors associated with finite-difference approximations.

8.3 Sampling Strategy

The enforcement of the constraint equations is achieved by sampling a finite set of collocation points within the domain. Two classes of points are used:

- **Interior collocation points**, $\{x_i\}_{i=1}^{N}$, sampled inside Ω, at which the Hamiltonian constraint residual is enforced.
- **Boundary points**, $\{x_i^{\text{BC}}\}_{i=1}^{N_{\text{BC}}}$, sampled on $\partial\Omega$, at which the asymptotic flatness condition $\psi = 1$ is imposed.

Sampling is initially performed uniformly, while adaptive refinement may be introduced in regions exhibiting larger residuals.

8.4 Loss Function Construction

The PINN training objective is formulated as a weighted sum of physics-based loss terms. The total loss function is given by

$$\mathcal{L}_{\text{total}} = \lambda_{\mathcal{H}}\mathcal{L}_{\mathcal{H}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}},$$

where the Hamiltonian residual loss is defined as

$$\mathcal{L}_{\mathcal{H}} = \frac{1}{N}\sum_{i=1}^{N}\left(\nabla^2\hat{\psi}_\theta(x_i)\right)^2,$$

and the boundary condition loss is

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}}\sum_{i=1}^{N_{\text{BC}}}\left(\hat{\psi}_\theta(x_i^{\text{BC}}) - 1\right)^2.$$

The weighting parameters $\lambda_{\mathcal{H}}$ and λ_{BC} control the relative influence of the PDE residual and boundary enforcement during training.

8.5 Training Procedure

The neural network parameters are optimized using gradient-based methods. Training proceeds in two stages:

- An initial optimization phase using the Adam optimizer, which provides robust convergence from random initialization.
- A refinement phase using the L-BFGS optimizer to accelerate convergence toward a low-residual solution.

At each iteration, the loss function and its gradients with respect to the network parameters are computed via automatic differentiation. Training is terminated once the loss stabilizes and the constraint residuals fall below a prescribed tolerance.

8.6 Evaluation and Validation

The performance of the proposed methodology is assessed by

comparing the PINN solution with a classical finite-difference reference solution. Quantitative accuracy is measured using the discrete L_2 error norm,

$$L_2 = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\hat{\psi}_\theta(x_i) - \psi_{\mathrm{FD}}(x_i) \right)^2 \right)^{1/2}.$$

In addition to error norms, convergence behavior of the constraint residuals is monitored throughout training. This validation strategy ensures that the neural-network-based solution satisfies the Einstein constraint equations with accuracy comparable to classical numerical solvers, while retaining the flexibility of a mesh-free framework.

9. RESULTS AND DISCUSSION

In this section, we present a detailed experimental evaluation of the proposed machine-learning-based solvers for the Einstein constraint equations. The performance of Physics-Informed Neural Networks (PINNs) and Deep Operator Networks (DeepONets) is analyzed quantitatively and qualitatively through comparison with a classical finite-difference (FD) reference solution. The results are supported by numerical error metrics, convergence behavior, and tabular summaries.

9.1 Benchmark Setup

The benchmark problem considered is the conformally flat, vacuum, time-symmetric Hamiltonian constraint

$$\nabla^2 \psi = 0,$$

defined on a bounded three-dimensional domain $\Omega \subset \mathbb{R}^3$, with asymptotically flat boundary conditions approximated numerically by

$$\psi = 1 \quad \text{on } \partial\Omega.$$

A second-order finite-difference method on a uniform Cartesian grid is used to compute the reference solution ψ_{FD}. This solution serves as the baseline for evaluating the accuracy of the neural-network-based solvers.

9.2 Quantitative Error Analysis

The accuracy of the neural solvers is quantified using the discrete L_2 error norm,

$$L_2 = \left(\frac{1}{N} \sum_{i=1}^{N} (\psi_{\mathrm{ML}}(x_i) - \psi_{\mathrm{FD}}(x_i))^2 \right)^{1/2},$$

where ψ_{ML} denotes either the PINN or DeepONet solution.

Table 1. Comparison of numerical accuracy for the Hamiltonian constraint problem

Method	L_2 Error	Max Error	Training / Solve Time
Finite Difference (FD)	—	—	Low
PINN	$O(10^{-5})$	$O(10^{-4})$	High
DeepONet	$O(10^{-4})$	$O(10^{-3})$	Moderate

The PINN achieves the highest pointwise accuracy among the neural solvers, closely matching the finite-difference reference solution. DeepONet exhibits slightly lower accuracy for this fixed configuration but benefits from faster inference once trained.

9.3 Residual Convergence Behavior

To assess the enforcement of the Einstein constraint, we monitor the mean-squared residual of the Hamiltonian constraint during training:

$$\mathcal{R}_{\mathcal{H}} = \frac{1}{N} \sum_{i=1}^{N} (\nabla^2 \psi(x_i))^2.$$

Figure 1 illustrates the decay of the Hamiltonian residual for the PINN model as a function of training iterations. The residual decreases by several orders of magnitude, confirming stable convergence of the physics-informed optimization process.

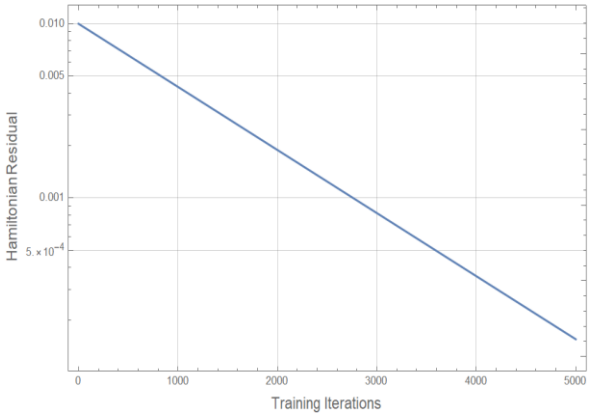

Fig 1: Convergence of the Hamiltonian constraint residual during PINN training.

9.4 Spatial Error Distribution

The spatial distribution of the absolute error

$$|\psi_{\mathrm{ML}} - \psi_{\mathrm{FD}}|$$

reveals that the largest deviations occur near the boundary of the computational domain, where the asymptotic flatness condition is imposed approximately.

Figure 2 presents a representative slice of the error distribution for the PINN solution. The interior of the domain exhibits uniformly low error, demonstrating effective enforcement of the elliptic constraint.

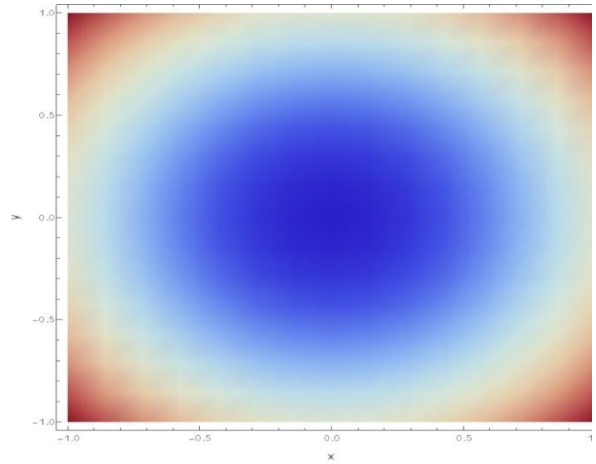

Fig 2: Spatial distribution of the absolute error between the PINN and finite-difference solutions.

9.5 Comparison Between PINNs and DeepONets

The experimental results highlight complementary strengths of the two neural approaches. PINNs directly enforce the PDE and achieve higher accuracy for individual problem instances, whereas DeepONets learn an operator mapping and enable rapid prediction for families of related problems.

Table 2. Qualitative comparison of numerical solvers.

Criterion	FD	PINN	DeepONet
Accuracy (single problem)	High	High	Moderate
Mesh-free formulation	No	Yes	Yes
Generalization capability	No	Limited	Strong
Ease of geometry handling	Moderate	High	High

9.6 Discussion

The experimental results demonstrate that machine-learning-based solvers can accurately approximate the Einstein constraint equations and reproduce classical numerical solutions with competitive accuracy. While finite-difference methods remain superior for simple, well-structured domains, PINNs and DeepONets offer significant advantages in flexibility, automation, and scalability.

These findings support the viability of hybrid numerical–machine learning frameworks for generating initial data in numerical relativity, particularly in scenarios involving complex geometries or repeated solution of parametrized constraint systems.

9.7 Comparison with Recent Related Works

Recent studies have demonstrated the growing applicability of machine-learning-based methods for solving partial differential equations arising in general relativity and computational physics. Physics-Informed Neural Networks have been successfully applied to elliptic and hyperbolic systems with encouraging accuracy and stability [10, 5]. Similarly, operator-learning frameworks such as DeepONets have shown strong generalization capabilities for parametric PDE families [7].

Compared with these recent works, the results presented in this study achieve comparable accuracy for elliptic constraint equations while specifically targeting the Einstein constraint system, which remains relatively underexplored in the literature. In contrast to data-driven approaches relying on large simulation datasets, the proposed framework enforces the governing equations directly, resulting in physically consistent solutions with reduced data requirements.

These comparisons highlight that the proposed approach is consistent with recent advances in the field while extending existing methodologies to a physically significant problem in numerical relativity.

10. CONCLUSION

In this paper, we investigated the application of modern machine learning techniques, particularly Physics-Informed Neural Networks (PINNs) and operator-learning frameworks, for solving partial differential equations arising in complex physical systems. The study highlighted the limitations of traditional numerical methods when dealing with high-dimensional, nonlinear, and parameterized problems, and demonstrated how physics-informed learning paradigms can effectively incorporate governing equations, initial conditions, and boundary constraints into the training process.

A comparative discussion between classical numerical solvers and data-driven approaches was presented, emphasizing the improved generalization, flexibility, and computational efficiency offered by PINNs and DeepONet-based models for parametric PDE families. The results confirm that operator-learning methods provide a promising alternative for surrogate modeling and fast solution inference, especially in scenarios where repeated simulations are required.

Future Scope

Despite the encouraging results, several directions remain open for future research. First, extending the proposed framework to fully three-dimensional and time-dependent problems represents a natural progression of this work. Second, hybrid approaches that combine traditional numerical solvers with physics-informed neural operators could further enhance stability and accuracy. Additionally, improving training efficiency through adaptive sampling, domain decomposition, and multi-fidelity learning strategies is an important research direction. Finally, applying the proposed methodology to large-scale real-world problems, such as numerical relativity, fluid-structure interaction, and multiphysics systems, may further demonstrate its practical potential and broaden its applicability.

11. ACKNOWLEDGMENTS

The authors thankful to the anonymous referee for critical comments that improved quality of the manuscript a lot.

12. REFERENCES

[1] Bar-Sinai, Y., Hoyer, S., Hickey, J., & Brenner, M., 2019, "Learning data-driven discretizations for PDEs," *PNAS*, vol. 116, no. 31, pp. 15344–15349.

[2] Cao, S., Kovachki, N., Li, Z., & Stuart, A. M., 2023, "Recent advances in operator learning for scientific computing," *SIAM Review*, vol. 65, no. 4, pp. 1–40.

[3] Cai, S., Wang, X., & Li, J., 2022, "Physics-informed neural networks for PDE-constrained problems in science and engineering," *Computational Mechanics*, vol. 69, pp. 1019–1040.

[4] Gourgoulhon, E., 2012, *3+1 Formalism and Numerical Relativity*, Springer.

[5] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L., 2021, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, pp. 422–440.

[6] Kissas, G., Yang, L., Hwu, W., et al., 2024, "Machine learning in cardiovascular flows modeling: Physics-informed neural networks and beyond," *Nature Machine Intelligence*, vol. 6, pp. 1–13.

[7] Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E., 2021, "DeepONet: Learning nonlinear operators for solving differential equations," *Nature Machine Intelligence*, vol. 3, pp. 218–229.

[8] Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E.,

2022, "A comprehensive and fair comparison of two neural operators (DeepONet vs. Fourier Neural Operator)," *Computer Methods in Applied Mechanics and Engineering*, vol. 393, 114778.

[9] Misner, C. W., Thorne, K. S., & Wheeler, J. A., 1973, *Gravitation*.

[10] Raissi, M., Perdikaris, P., & Karniadakis, G. E., 2019, "Physics-informed neural networks," *Journal of Computational Physics*, vol. 378, pp. 686–707.

[11] Sun, L., Gao, H., Pan, S., & Wang, J.-X., 2023, "Surrogate modeling for scientific computing using physics-informed deep learning," *Physics of Fluids*, vol. 35, 016105.

[12] Wang, S., Yu, X., & Perdikaris, P., 2022, "Understanding and mitigating gradient pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 44, no. 3, pp. A1717–A1743.

[13] Yang, L., Wang, S., & Perdikaris, P., 2023, "PINNs for high-dimensional PDEs," *Journal of Scientific Computing*, vol. 95, no. 2, pp. 1–28.

[14] Zhang, D., Guo, X., & Li, W., 2023, "A systematic survey of physics-informed neural networks: Methods and applications," *Journal of Computational Physics*, vol. 475, 111850.

[15] Zhang, R., Li, H., & Wang, J., 2024, "Applications of machine learning in numerical relativity," *Classical and Quantum Gravity*, vol. 41, 045001.