COMPETITIVE PHYSICS INFORMED NETWORKS

Qi Zeng, Spencer H. Bryngelson & Florian Schäfer
School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
{qzeng37@,shb@,florian.schaefer@cc.}gatech.edu

ABSTRACT

Physics Informed Neural Networks (PINNs) solve partial differential equations (PDEs) by representing them as neural networks. The original PINN implementation does not provide high accuracy, typically attaining about 10^{-3} L_2 relative error. We formulate and test an adversarial approach called competitive PINNs (CPINNs) to overcome this limitation. CPINNs train a discriminator that is rewarded for predicting PINN mistakes. The discriminator and PINN participate in a zero-sum game with the exact PDE solution as an optimal strategy. This approach avoids the issue of squaring the large condition numbers of PDE discretizations. Numerical experiments show that a CPINN trained with competitive gradient descent can achieve errors two orders of magnitude smaller than that of a PINN trained with Adam or stochastic gradient descent.

1 INTRODUCTION

Physics informed networks. Partial differential equations (PDEs) model physical phenomena like fluid dynamics, heat transfer, electromagnetism, and more. Solving PDEs using numerical methods is expensive. Recently, Physics Informed Neural Networks (PINNs) were developed as a partial solution to this problem. PINNs represent the PDE solution as a neural network trained to minimize the square of the violation of the PDE itself (Raissi et al., 2019).

Training pathologies in PINNs. PINNs can, in principle, be applied to all PDEs, but they exhibit numerous failure modes (Liu et al., 2021; Krishnapriyan et al., 2021). For example, they are often unable to achieve high-accuracy solutions. The first PINN implementations reported relative L_2 errors of about 10^{-3} (Raissi et al., 2019; Liu et al., 2021; Wang et al., 2021), though better accuracy than this is required for engineering and design in many application areas.

The perils of squaring. PINNs use a squared loss function on the residual. For a linear PDE of order s, this is no different than solving an equation of order $2s$, akin to using normal equations in linear algebra. The condition number κ of the resulting problem is thus the *square* of the condition number of the original one. Since solving discretized PDEs is an ill-conditioned problem, this inhibits the convergence of iterative solvers.

Weak formulations. Integration by parts allows the derivation of a *weak form* of a PDE, which for some PDEs can be turned into a minimization formulation that does not square the condition number. This procedure has been successfully applied by E and Yu (2017) to solve PDEs with neural networks (*Deep Ritz*). However, the derivation of such minimization principles is problem-dependent, limiting the generality of the formulation. Deep Ritz also employs penalty methods to enforce boundary values, though these preclude the solution of the minimization problem from being the exact solution of the PDE. Liao and Ming (2019) proposed a partial solution to this problem. The existing work most closely related to ours is the work of Zang et al. (2020), who propose a different game formulation based on the weak form of the PDE.

Competitive PINNs. We propose Competitive Physics Informed Neural Networks (CPINNs) to address the above problems. The CPINN formulation recasts PINN optimization as a minimax game between the PINN and a discriminator network. The discriminator learns to predict mistakes of the PINN and is rewarded for correct predictions, whereas the PINN is penalized. We train both players simultaneously on the resulting zero-sum game to reach a Nash equilibrium that matches the exact solution of the PDE.

Summary of contributions We compare CPINN and PINN solutions to a Poisson problem. The CPINNs are trained using competitive gradient descent (CGD) algorithms (Schäfer and Anandkumar, 2019) or Extragradients (Korpelevich, 1977), and the PINNs are trained using Adam and stochastic gradient descent (SGD). Given the same computational budget, we show that CPINN solutions have *138-times* smaller errors than PINN solutions. In addition, CPINN solves examples where normal PINNs fail to train.

2 Competitive PINN formulation

We formulate CPINNs for a PDE of the general form

$$\mathcal{A}[u] = f, \quad \text{in } \Omega \tag{1}$$

$$u = g, \quad \text{on } \partial\Omega, \tag{2}$$

where $\mathcal{A}[\cdot]$ is a (possibly nonlinear) differential operator and Ω is a domain in $\mathbb{R}^d$ with boundary $\partial\Omega$. To simplify notation, we assume that f, g, and u are real-valued functions on Ω, $\partial\Omega$, and $\Omega \cup \partial\Omega$, respectively. One can extend both PINNs and CPINNs to vector-valued such functions if needed.

2.1 Physics Informed Neural Networks (PINNs)

PINNs approximate the PDE solution u by a neural network $\mathcal{P}$ mapping d-variate inputs to real numbers. The weights are chosen such as to satisfy equation 1 and equation 2 on the points $\boldsymbol{x} \subset \Omega$ and $\overline{\boldsymbol{x}} \subset \partial\Omega$. The loss function used to train $\mathcal{P}$ has the form

$$\mathcal{L}^{\text{PINN}}(\mathcal{P}, \boldsymbol{x}, \overline{\boldsymbol{x}}) = \mathcal{L}^{\text{PINN}}_{\Omega}(\mathcal{P}, \boldsymbol{x}_{\Omega}) + \mathcal{L}^{\text{PINN}}_{\partial\Omega}(\mathcal{P}, \overline{\boldsymbol{x}}), \tag{3}$$

where $\mathcal{L}^{\text{PINN}}_{\partial\Omega}$ measures the violation of the boundary conditions equation 2 and $\mathcal{L}^{\text{PINN}}_{\Omega}$ measures the violation of the PDE of equation 1. They are defined as

$$\mathcal{L}^{\text{PINN}}_{\Omega}(\mathcal{P}, \boldsymbol{x}) = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} \left(\mathcal{A}[\mathcal{P}](x_i)) - f(x_i)\right)^2 \tag{4}$$

$$\mathcal{L}^{\text{PINN}}_{\partial\Omega}(\mathcal{P}, \overline{\boldsymbol{x}}) = \frac{1}{N_{\partial\Omega}} \sum_{i=1}^{N_{\partial\Omega}} \left(\mathcal{P}\left(\overline{x}_i\right) - g\left(\overline{x}_i\right)\right)^2, \tag{5}$$

Where N_{Ω} and $N_{\partial\Omega}$ are the number of points in the sets $\boldsymbol{x}$ (interior) and $\overline{\boldsymbol{x}}$ (boundary), and x_i and $\overline{x}_i$ are the i-th such points in $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}$.

By minimizing the loss in equation 3, PINNs approximate its unique global minimum given by the exact solution u of the PDE. This optimization problem is challenging, resulting in low accuracy or the inability to train at all (Wang et al., 2021; 2022).

2.2 Competitive Physics Informed Neural Networks (CPINNs)

CPINNs introduce a discriminator network $\mathcal{D}$ with input $x \in \mathbb{R}^d$ and output $\mathcal{D}_{\Omega}(x)$ and $\mathcal{D}_{\partial\Omega}(x)$. $\mathcal{P}$ and $\mathcal{D}$ compete in a zero-sum game where $\mathcal{P}$ learns to solve the PDE, and $\mathcal{D}$ learns to predict the mistakes of $\mathcal{P}$. This game is defined as a minimax problem

$$\max_{\mathcal{D}} \min_{\mathcal{P}} \mathcal{L}^{\text{CPINN}}_{\Omega}(\mathcal{P}, \mathcal{D}, \boldsymbol{x}) + \mathcal{L}^{\text{CPINN}}_{\partial\Omega}(\mathcal{P}, \mathcal{D}, \overline{\boldsymbol{x}}), \tag{6}$$

where

$$\mathcal{L}^{\text{CPINN}}_{\Omega}(\mathcal{D}, \mathcal{P}, \boldsymbol{x}) = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} \mathcal{D}_{\Omega}(x_i) \left(\mathcal{A}[\mathcal{P}](x_i) - f(x_i)\right), \tag{7}$$

$$\mathcal{L}^{\text{CPINN}}_{\partial\Omega}(\mathcal{D}, \mathcal{P}, \overline{\boldsymbol{x}}) = \frac{1}{N_{\partial\Omega}} \sum_{i=1}^{N_{\partial\Omega}} \mathcal{D}_{\partial\Omega}(\overline{x}_i) \left(\mathcal{P}\left(\overline{x}_i\right) - g\left(\overline{x}_i\right)\right). \tag{8}$$

Here, $\mathcal{D}_{\Omega}(x_i)$ and $\mathcal{D}_{\partial\Omega}(\overline{x})$ can be interpreted as *bets* by the discriminator that the PINN will over- or under-shoot equation 1 and equation 2. Winning the bet results in a reward for $\mathcal{D}$ and a penalty for $\mathcal{P}$, while a lost bet has the opposite effect.

The Nash equilibrium of this game is $\mathcal{P} \equiv u$ and $\mathcal{D} \equiv 0$. Therefore, one can use iterative algorithms to compute Nash equilibria in zero-sum games to approximately solve the PDE. In this work, we focus on the extragradient method of Korpelevich (1977) and the competitive gradient descent of Schäfer and Anandkumar (2019). Still, CPINNs can be trained with other proposed methods for smooth games optimization (Mescheder et al., 2017; Balduzzi et al., 2018; Gemp and Mahadevan, 2018; Daskalakis et al., 2018; Letcher et al., 2019).

2.3 Avoiding squares of differential operators

Multiagent methods for solving PDEs may seem unorthodox, yet they are motivated by observations in classical numerical analysis. To explore this connection, we consider the particular case of a linear PDE and networks $\mathcal{P}, \mathcal{D}$ with outputs that depend linearly on their weight-vectors $\boldsymbol{\pi}$ and $\boldsymbol{\delta}$ resulting in the parametric form

$$\mathcal{P}(x) = \sum_{i=1}^{\dim(\boldsymbol{\pi})} \pi_i \psi_i(x), \quad \mathcal{D}(x) = \sum_{i=1}^{\dim(\boldsymbol{\delta})} \delta_i \phi_i(x), \tag{9}$$

for basis function sets $\{\psi_i\}_{1 \le i \le \dim(\boldsymbol{\pi})}$ and $\{\phi_i\}_{1 \le i \le \dim(\boldsymbol{\delta})}$. We focus our attention on the PDE constraint in equation 1, which we evaluate at a set $\boldsymbol{x}$ of N_Ω points. Defining $\boldsymbol{A} \in \mathbb{R}^{N_\Omega \times \dim(\boldsymbol{\pi})}$ and $\boldsymbol{f} \in \mathbb{R}^{N_\Omega}$ we have

$$A_{ij} := \mathcal{A}[\psi_j](x_i), \quad f_i := f(x_i) \tag{10}$$

and obtain the discretized PDE

$$\boldsymbol{A}\boldsymbol{\pi} = \boldsymbol{f}. \tag{11}$$

PINNs solve this equation 11 via a least squares problem

$$\min_{\boldsymbol{\pi}} \|\boldsymbol{A}\boldsymbol{\pi} - \boldsymbol{f}\|^2, \tag{12}$$

trading the equality constraint for a minimization problem with solution $\boldsymbol{\pi} = (\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top \boldsymbol{f}$. Since the matrix $(\boldsymbol{A}^\top \boldsymbol{A})$ is symmetric positive-definite, it can be solved with specialized algorithms such as the conjugate gradient method (CG) (Shewchuk, 1994). This approach is beneficial for well-conditioned nonsymmetric matrices, but inappropriate for ill-conditioned $\boldsymbol{A}$ (Axelsson, 1977). This is because $\kappa(\boldsymbol{A}^\top \boldsymbol{A}) = \kappa(\boldsymbol{A})^2$, resulting in slow convergence of iterative solvers. Since differential operators are unbounded, their discretization leads to ill-conditioned linear systems. Krishnapriyan et al. (2021) argue that the ill-conditioning of equation 12 is the reason for the optimization difficulties they observe.

CPINNs turn the discretized PDE in equation 11 into the saddlepoint problem

$$\min_{\boldsymbol{\pi}} \max_{\boldsymbol{\delta}} \boldsymbol{\delta}^\top (\boldsymbol{A}\boldsymbol{\pi} - \boldsymbol{f}). \tag{13}$$

The solution of this problem is the same as that of the system of equations

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{f} \end{bmatrix}, \quad \text{with} \quad \kappa\left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \right) = \kappa(\boldsymbol{A}). \tag{14}$$

By turning equation 11 into the saddle point problem of equation 13 instead of the minimization form of equation 12, CPINNs avoid squaring the condition number.

In light of the above, the decision between PINNs and CPINNs is the nonlinear analog of an old dilemma in numerical linear algebra: should one trade solving a linear system directly for a least-squares problem? This trade allows algorithms like CG at the cost of a squared condition number. However, this is a poor choice for the ill-conditioned systems that arise from discretized PDEs. As we demonstrate in the following numerical experiments, the additional cost of solving the multiagent optimization problem can be worth paying.

3 Results

3.1 When CPINN Trains Better

We consider an example of a two dimensional Poisson equation:

$$\Delta u(x, y) = -2\sin(x)\cos(y), \quad x, y \in [-2, 2] \tag{15}$$

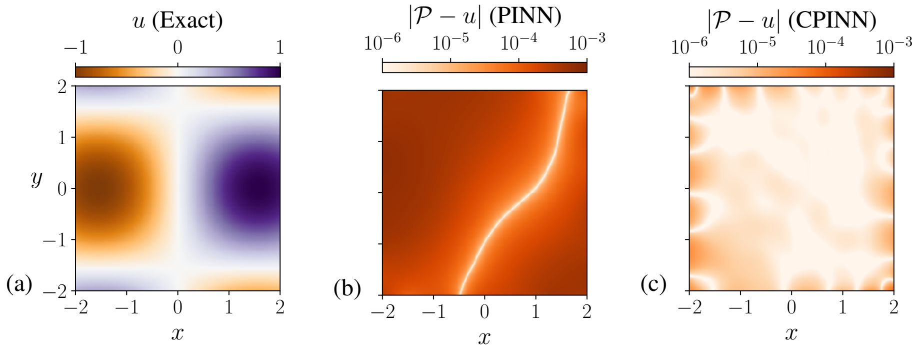

Figure 1: (a) Exact solution u to equation 1 and errors of (b) PINN + Adam and (c) CPINN + ACGD.

with Dirichlet boundary conditions

$$\begin{aligned} u(x,-2) &= \sin(x)\cos(-2), \qquad & u(-2,y) &= \sin(-2)\cos(y), \\ u(x,\ \ 2) &= \sin(x)\cos(\ \ 2), \qquad & u(\ \ 2,y) &= \sin(\ \ 2)\cos(y). \end{aligned}$$

This problem has a manufactured solution

$$u(x,y) = \sin(x)\cos(y). \tag{16}$$

The training set consists of 50 points placed randomly along the edge of the domain to calculate the boundary condition loss and 5×10^3 points placed via the Latin Hypercube strategy to enforce the PDE constraint in equation 15 (Stein, 1987). The PINN implementation has 3 hidden layers with 50 neurons per layer, $\tanh$ activation functions, and losses

$$\mathcal{L}^{\text{PINN}}_{\partial\Omega} = \frac{1}{N_{\partial\Omega}} \sum_{i=1}^{N_{\partial\Omega}} \left(\mathcal{P}(\overline{x}_i, \overline{y}_i) - u(\overline{x}_i, \overline{y}_i)\right)^2,$$

$$\mathcal{L}^{\text{PINN}}_{\Omega} = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} \left(\mathcal{P}_{xx}(x_{\Omega,i}, y_{\Omega,i}) + \mathcal{P}_{yy}(x_{\Omega,i}, y_{\Omega,i}) + 2\sin(x_{\Omega,i})\cos(y_{\Omega,i})\right)^2.$$

PINN optimization is implemented via Adam (Kingma and Ba, 2014) and SGD (Ruder, 2016) algorithms. Adam is parameterized by a learning rate of 10^{-3}, beta values $\beta_1 = 0.99$ and $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$ (all parameters following the usual naming conventions). The SGD implementation uses a learning rate of 10^{-2}. The CPINNs discriminator has 4 hidden layers, each with 50 neurons and ReLU activation functions. ExtraAdam and ExtraSGD (Gidel et al., 2019), CGD, and an Adam-based CGD variant (ACGD) (Schäfer and Anandkumar, 2019; Schäfer et al., 2020b; Zheng, 2020) solve for the Nash equilibrium of the minimax game of equation 6. The implementation of the CGD algorithms used is located at `https://github.com/devzhk/cgds-package`.

Figure 1 (a) shows the exact solution of the PDE and the absolute error of the best models trained using (b) PINN and (c) CPINN. CPINNs achieve lower errors than PINNs by a factor of about 100 throughout the domain, with somewhat larger errors near the domain boundaries.

Table 1: Performance of CPINNs and PINNs on the 2D Poisson problem of equation 15.

	Optimizer	Iterations	L_2 Rel. Error	$\mathcal{L}^{\text{PINN}}$	$\mathcal{L}^{\text{PINN}}_{\Omega}$	$\mathcal{L}^{\text{PINN}}_{\partial\Omega}$
PINN	Adam	5×10^5	6.3×10^{-4}	1.0×10^{-7}	4.3×10^{-8}	5.7×10^{-8}
	SGD	5×10^5	7.5×10^{-3}	6.0×10^{-5}	3.4×10^{-5}	2.5×10^{-5}
CPINN	ACGD	2×10^3	8.7×10^{-6}	4.5×10^{-9}	4.3×10^{-9}	1.1×10^{-10}
	CGD	6×10^4	1.7×10^{-3}	7.1×10^{-5}	6.7×10^{-5}	3.2×10^{-6}
	ExtraAdam	7.5×10^4	2.9×10^{-1}	1.2	1.2	3.9×10^{-4}
	ExtraSGD	7.5×10^4	7.0×10^{-3}	2.0×10^{-4}	1.4×10^{-4}	5.6×10^{-5}

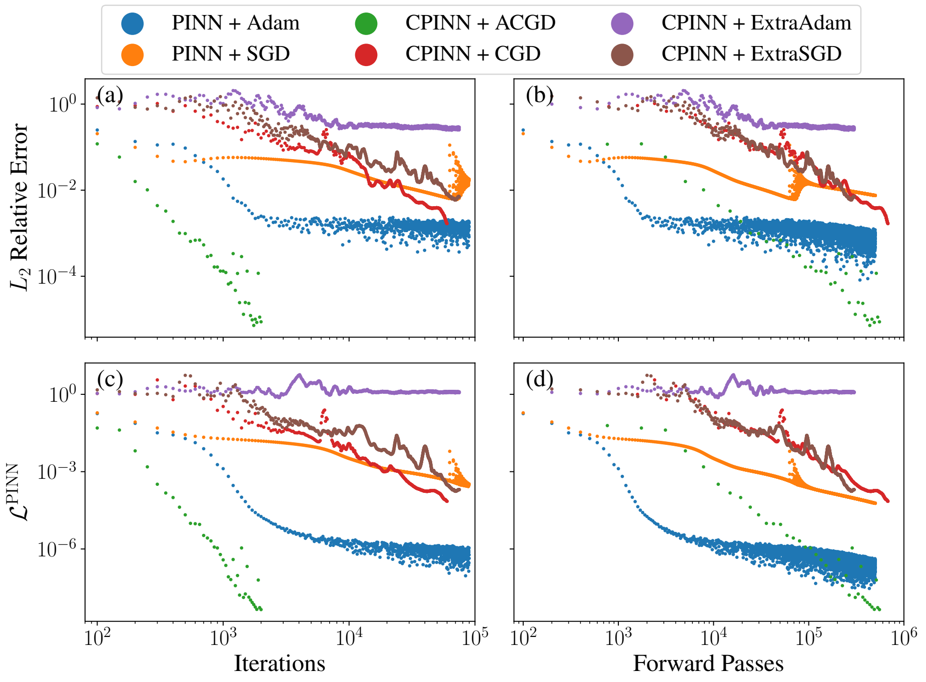

Figure 2: Comparison of PINNs and CPINNs with different optimizers in terms of (a–b) their error and (c–d) their PINN losses. The number of forward passes abscissa of (b) and (d) buttress (a) and (c) to show the errors and losses for comparable computational costs.

Quantitative results are in figure 2 and summarized in table 1. In this example, the CGD algorithm outperforms SGD after roughly 2×10^4 training iterations, and the Adam-based variant of CGD (ACGD) took 2×10^3 iterations to reach an accuracy that is two orders of magnitude higher than that of Adam with 5×10^5 training iterations. Training CPINNs via ExtraSGD also generated better results than PINNs trained with SGD. However, the current ExtraAdam implementation does not give an accurate CPINN result for unknown reasons.

The CGD algorithm entails additional Hessian–vector product computations compared to Adam and SGD optimizers. Figure 2 (b) and (b) account for this additional cost by reporting the number of forward passes through the model. Specifically, Adam and SGD optimizers have 1 forward pass per iteration, ExtraAdam and ExtraSGD have 4, and CGD and ACGD use 4 plus 2 times the number of conjugate gradient iterations used to solve the matrix inverse in CGD.

3.2 When PINNs fail to train

We next consider a case where PINNs fail to train entirely, specific one-dimensional wave equations Wang et al. (2022)

$$u_{tt} - 4u_{xx}(x,t) = 0, \quad (x,t) \in (0,1) \times (0,1) \tag{17}$$

$$u(0,t) = u(1,t) = 0, \quad t \in [0,1] \tag{18}$$

$$u(x,0) = \sin(\pi x) + \frac{1}{2}\sin(4\pi x), \quad x \in [0,1] \tag{19}$$

$$u_t(x,0) = 0, \quad x \in [0,1]. \tag{20}$$

with the solution

$$u(x,t) = \sin(\pi x)\cos(2\pi x) + \frac{1}{2}\sin(4\pi x)\cos(8\pi t), \tag{21}$$

and boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad x \in \delta\Omega. \tag{22}$$

The PINN loss is

$$\mathcal{L}^{\text{PINN}} = \mathcal{L}^{\text{PINN}}_{\partial\Omega} + \mathcal{L}^{\text{PINN}}_{\partial\Omega_t} + \mathcal{L}^{\text{PINN}}_{\Omega}, \tag{23}$$

where

$$\mathcal{L}^{\text{PINN}}_{\partial\Omega} = \frac{1}{N_{\delta\Omega}} \sum_{i=1}^{N_{\delta\Omega}} \left(\mathcal{P}(x_{\delta\Omega}, t_{\delta\Omega})\right)^2 \tag{24}$$

$$\mathcal{L}^{\text{PINN}}_{\partial\Omega_t} = \frac{1}{N_{\delta\Omega_t}} \sum_{i=1}^{N_{\delta\Omega_t}} \left(\mathcal{P}(x_{\delta\Omega_t}, t_{\delta\Omega_t}) - g(x_{\delta\Omega_t}, t_{\delta\Omega_t})\right)^2 \tag{25}$$

$$\mathcal{L}^{\text{PINN}}_{\Omega} = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} \left(\mathcal{P}_{tt}(x_{\Omega}, t_{\Omega}) - 4\mathcal{P}_{xx}(x_{\Omega}, t_{\Omega})\right)^2. \tag{26}$$

$\mathcal{L}^{\text{PINN}}_{\partial\Omega}$ enforces the boundary conditions of equation 18 and equation 19, $\mathcal{L}^{\text{PINN}}_{\partial\Omega_t}$ enforces the boundary derivative constraint in equation 20, and $\mathcal{L}^{\text{PINN}}_{\Omega}$ enforces the PDE constraint in equation 17.

The training set contains 500 points for each loss funcion. We used a PINN with 4 layers with 400 neurons in each layer, trained with Adam with a learning rate of 10^{-3}, beta values $\beta_1 = \beta_2 = 0.99$. The CPINN discriminator has 4 hidden layers, each with 400 neurons. ACGD was used to train CPINN, parameterized by a minimum and maximum learning rate of 5×10^{-4} and 10^{-4}, $\beta = 0.99$ and an iterative solve tolerance of 10^{-8}.

The results are shown in figure 3 and summarized in table 2. Each CPINNs training epoch has higher computational cost than a comparative PINN epoch. However, even under the same computational budget (number of forward passes) CPINNs obtain an L_2 relative error about 3.5 times lower than PINNs do in 10^6 training epochs.

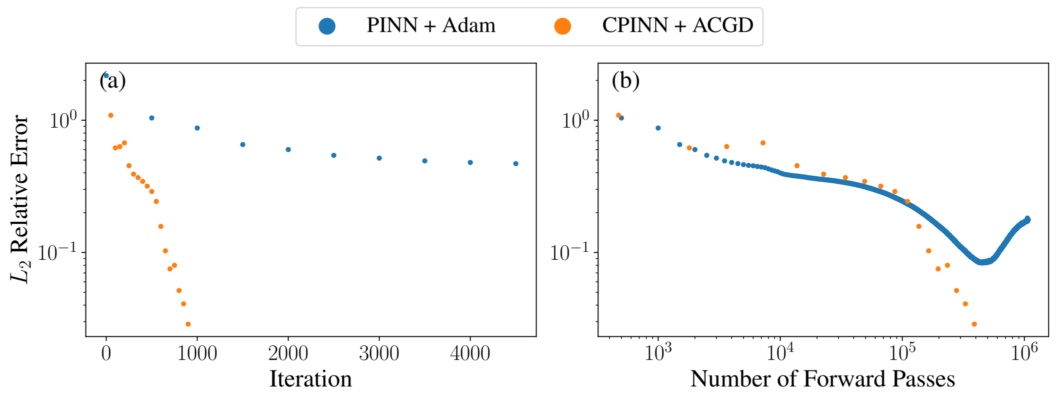

Figure 3: Relative error of CPINNs and PINNs as labeled for equation 17. (a) shows the first 900 training epochs of the CPINN case and the first 5000 training epochs of the PINN one. (b) shows that PINNs do not train for this problem, even with larger computational cost budget.

Table 2: Performance of CPINNs and PINNs on the Wave Problem of equation 17

	Optimizer	Iterations	Forward Passes	L_2 Rel. Error
PINN	Adam	3.8×10^5	3.8×10^5	9.7×10^{-2}
CPINN	ACGD	900	3.8×10^5	2.9×10^{-2}

4 Connections to Existing Work

Saddle point formulations arise in Petrov–Galerkin and mixed finite-element methods, which use function spaces to represent the solution (called trial functions) and measure the violation of the equations (called test functions) (Quarteroni and Valli, 2008; Fortin and Brezzi, 1991). The neural networks introduced here, $\mathcal{P}$ and $\mathcal{D}$, are nonlinear analogs of trial and test spaces. Different from our approach, *Deep Petrov–Galerkin* of Shang et al. (2022) only parameterizes the trial space by a neural network and uses a conventional discretization such as finite elements for the test space. They only train the last layer while keeping all other layers fixed after initialization. Saddle point problems in PDEs can be interpreted geometrically and have a game-theoretic interpretation (Lemaire, 1973). In a complementary game-theoretic approach, Owhadi (2017) cast computation as a game between a numerical analyst and an environment, using ideas from decision theory (Wald, 1945).

Saddle point problems also arise from the introduction of Lagrange multipliers, which are used to cast a constrained optimization problem as an unconstrained saddle point problem (Brezzi, 1974). These discriminators can be viewed as neural-network-parametrized Lagrange multipliers that enforce distributional equality (in GANs) or satisfaction of the PDE (in CPINNs). Our work is thus related to recent efforts that combine CGD with Lagrange multipliers to solve constrained optimization problems arising from reinforcement learning and computer graphics (Bacon et al., 2019; Yu et al., 2021; Soliman et al., 2021). The need for constrained training of neural networks also arises in other applications, resulting in an increasing body of work on this topic (Pathak et al., 2015; Donti et al., 2021; Lokhande et al., 2020; Fioretto et al., 2020).

Following section 1, the *Deep Ritz* method exploits the fact that some PDEs can be cast as minimization problems without squaring the condition number (E and Yu, 2017). In the notation of section 2.3, this expresses equation 11 as

$$\min_{\boldsymbol{\pi}} \frac{\boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{\pi}}{2} - \boldsymbol{\pi}^\top \boldsymbol{f},$$

for a symmetric positive definite matrix $\boldsymbol{A}$. Such a formulation is not always available, and even if it exists, one still must enforce the PDE boundary conditions (Liao and Ming, 2019). Zang et al. (2020) use the weak formulation of the PDE to derive a minimax formulation. In the notation of section 2.3, their approach is similar to using the zero-sum game

$$\min_{\boldsymbol{\pi}} \max_{\boldsymbol{\delta}} \log\left(\left(\boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{\delta}\right)^2\right) - \log\left(\|\boldsymbol{\delta}\|^2\right).$$

Wang et al. (2021) and Xu et al. (2021) instead adapted the weights of the penalty in PINNs during training to improve accuracy. Different from the $\mathcal{D}$ output of CPINNs, these weights are multiplied with the *square* violation of the equality constraint, which is always greater than zero. Therefore, they do not correspond to a meaningful zero-sum game because the optimal discriminator strategy in this case drives all weights to infinity. Alternatively, Krishnapriyan et al. (2021) recommend the use of curriculum learning. They train the initial PINN on a better-conditioned variant of the original problem that slowly transforms into the target problem during training.

5 Discussion

This work introduced CPINNs, an agent-based approach to the neural-network-based solution of PDEs. CPINNs are crafted to avoid the ill-conditioning resulting from traditional PINNs least squares loss functions. As shown in section 3, CPINNs trained with ACGD can solve PINNs to almost single-precision floating-point accuracy and work for a wave equation that traditional PINNs failed to train.

CPINNs can be readily improved by reducing the computational cost of solving the minimax game equilibrium. This work used Schur complementation and CG methods to solve the matrix inverse of CGD. We anticipate that training with different solvers, such as MINRES or GMRES, will accelerate convergence by avoiding the squared condition numbers entirely (Paige and Saunders, 1975; Saad and Schultz, 1986).

We will also attempt to resolve the performance degradation we observed with ExtraAdam. The present experiments used the training points for each iteration, following Raissi et al. (2019). We plan to investigate the effects of batch stochasticity on training accuracy in the future. Finally, we plan to investigate the use of competitive mirror descent (CMD) for partial differential inequalities and contact problems (Schäfer et al., 2020a; Lions, 1972).

Acknowledgments

This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA. We thank Hongkai Zheng for maintaining the CGDs package used to conduct this work.

References

O. Axelsson. Solution of linear systems of equations: Iterative methods. In *Sparse matrix techniques*, pages 1–51. Springer, 1977.

P.-L. Bacon, F. Schäfer, C. Gehring, A. Anandkumar, and E. Brunskill. A Lagrangian method for inverse problems in reinforcement learning. In *Optimization in RL workshop at NeurIPS*, 2019.

D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pages 354–363. PMLR, 2018.

F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Publications mathématiques et informatique de Rennes*, (S4):1–26, 1974.

C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2018.

P. L. Donti, D. Rolnick, and J. Z. Kolter. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021.

W. E and B. Yu. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2017.

F. Fioretto, P. V. Hentenryck, T. W. Mak, C. Tran, F. Baldo, and M. Lombardi. Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 118–135. Springer, 2020.

M. Fortin and F. Brezzi. *Mixed and hybrid finite element methods*, volume 3. New York: Springer-Verlag, 1991.

I. Gemp and S. Mahadevan. Global convergence to the equilibrium of GANs using variational inequalities. *arXiv:1808.01531*, 2018.

G. Gidel, H. Berard, P. Vincent, and S. Lacoste-Julien. A variational inequality perspective on generative adversarial nets. *arXiv:1802.10551*, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

G. M. Korpelevich. Extragradient method for finding saddle points and other problems. *Matekon*, 13(4):35–49, 1977.

A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *arXiv:2109.01050*, 2021.

B. Lemaire. Saddle-point problems in partial differential equations and applications to linear quadratic differential games. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 27(1):105–160, 1973.

A. Letcher, D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. Differentiable game mechanics. *The Journal of Machine Learning Research*, 20(1):3032–3071, 2019.

Y. Liao and P. Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *arXiv:1912.01309*, 2019.

J. Lions. Partial differential inequalities. *Russian Mathematical Surveys*, 27(2):91, 1972.

Z. Liu, Y. Chen, Y. Du, and M. Tegmark. Physics-Augmented Learning: A new paradigm beyond Physics-Informed Learning. *arXiv:2109.13901*, 2021.

V. S. Lokhande, A. K. Akash, S. N. Ravi, and V. Singh. FairALM: Augmented Lagrangian method for training fair models with little regret. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020.

L. Mescheder, S. Nowozin, and A. Geiger. The numerics of gans. *Advances in neural information processing systems*, 30, 2017.

H. Owhadi. Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. *SIAM Review*, 59(1):99–149, 2017.

C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1796–1804, 2015.

A. Quarteroni and A. Valli. *Numerical approximation of partial differential equations*, volume 23. Springer Science & Business Media, 2008.

M. Raissi, P. Perdikaris, and G. E. 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. ISSN 0021-9991.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

F. Schäfer and A. Anandkumar. Competitive gradient descent. In *NeurIPS*, 2019.

F. Schäfer, A. Anandkumar, and H. Owhadi. Competitive Mirror Descent. *arXiv:2006.10179*, 2020a.

F. Schäfer, H. Zheng, and A. Anandkumar. Implicit competitive regularization in GANs. *arXiv:1910.05852*, 2020b.

Y. Shang, F. Wang, and J. Sun. Deep Petrov-Galerkin method for solving partial differential equations. *arXiv preprint arXiv:2201.12995*, 2022.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

Y. Soliman, A. Chern, O. Diamanti, F. Knöppel, U. Pinkall, and P. Schröder. Constrained Willmore surfaces. *ACM Transactions on Graphics (TOG)*, 40(4):1–17, 2021.

M. Stein. Large sample properties of simulations using Latin Hypercube Sampling. *Technometrics*, 29(2):143–151, 1987.

A. Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945.

S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM Journal of Scientific Computing*, 43:A3055–A3081, 2021.

S. Wang, X. Yu, and P. Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Compuational Physics*, 449:110768, 2022.

R. Xu, D. Zhang, M. Rong, and N. Wang. Weak form theory-guided neural network (TgNN-wf) for deep learning of subsurface single-and two-phase flow. *Journal of Computational Physics*, 436: 110318, 2021.

J. Yu, C. Gehring, F. Schäfer, and A. Anandkumar. Robust reinforcement learning: A constrained game-theoretic approach. In *Learning for Dynamics and Control*, pages 1242–1254, 2021.

Y. Zang, G. Bao, X. Ye, and H. Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

H. Zheng. CGDs. https://github.com/devzhk/cgds-package, 2020.