Enforcing Continuous Physical Symmetries in Deep Learning Network for Solving Partial Differential Equations

Zhi-Yong Zhang 1 * Hui Zhang 1 † Li-Sheng Zhang 2 ‡ Lei-Lei Guo 2 §

1 College of Science, Minzu University of China, Beijing 100081, P.R. China
2 College of Science, North China University of Technology, Beijing 100144, P.R. China

Abstract: As a typical application of deep learning, physics-informed neural network (PINN) has been successfully used to find numerical solutions of partial differential equations (PDEs), but how to improve the limited accuracy is still a great challenge for PINN. In this work, we introduce a new method, symmetry-enhanced physics informed neural network (SPINN) where the invariant surface conditions induced by the Lie symmetries of PDEs are embedded into the loss function of PINN, for improving the accuracy of PINN. We test the effectiveness of SPINN via two groups of ten independent numerical experiments for the heat equation, Korteweg-de Vries (KdV) equation and potential Burgers equations respectively, which shows that SPINN performs better than PINN with fewer training points and simpler architecture of neural network. Furthermore, we discuss the computational overhead of SPINN in terms of the relative computational cost to PINN and show that the training time of SPINN has no obvious increases, even less than PINN for some cases.

Keywords: Physics-informed neural network, Invariant surface conditions, Lie symmetry, Partial differential equations

1 Introduction

Partial differential equations (PDEs) describe the complex phenomenon in various fields such as physics, chemistry and biology, thus finding solutions of PDEs is of great interest and a direct and effective way to study the dynamical behaviors of PDEs. The first choice is to use the approaches related with intrinsic properties of PDEs to construct exact solutions. The classical Lie symmetry theory of PDEs is one of the main inspiring sources for various new methods to obtain exact solutions for PDEs and also used to detect the integrability, translational or rotational invariance, and to construct conservation laws of PDEs [1,2]. Specifically, a continuous Lie symmetry of PDEs is a continuous transformation which maps one solution to another solution of the same PDEs. Furthermore, one can construct invariant solutions of PDEs via the invariant surface conditions generated by the Lie symmetries. However, for some PDEs such as the Navier-Stokes equations in fluid mechanics and the Schrödinger equation in quantum mechanics, it is difficult to find explicit exact solutions, thus one resorts to numerical methods to search for numerical solutions of PDEs.

In addition to the traditional sophisticated numerical methods, such as finite element, finite difference and finite volume, the physics-informed neural networks (PINN) attracted more attentions in the data-driven discovery of solutions of PDEs in recent years [3]. The core idea of PINN
is to represent the solutions of PDEs by a neural network where the parameters are trained via gradient descent of the loss function related with the PDEs, initial and boundary conditions. In particular, the technique of automatic differentiation is employed by the deep learning community to deal with the derivatives \[4\]. Up to now, PINN has been widely applied in the field of scientific computing, especially in solving forward and inverse problems of nonlinear PDEs due to the merits of flexibility and gridless nature \[5\]-\[7\]. The physics in PINN is described by the considered PDEs. However, the solutions may fail the physical properties of the equations such as the Lie symmetries and conservation laws. Thus in order to address this gap, the improved versions of PINN spring up. For example, a gradient-enhanced PINN was proposed in solving both forward and inverse problems of PDEs where the gradient information of the PDEs residual is embed into the loss function \[8\]. Lin and Chen introduced a two-stage PINN where the first stage is the PINN and the second stage is to incorporate the conserved quantities into mean squared error loss to train neural networks \[9\]. In \[10\], the authors enforced nonlinear analytic constraints in the architecture of neural network into the loss function to produce the results consistent with the constraints. Zhu et.al constructed the group-equivariant neural networks which respect the spatio-temporal parity symmetries and successfully emulated different types of periodic solutions of nonlinear dynamical lattices \[11\]. More targeted neural networks such as the Bayesian PINN \[12\], discrete PINN framework based on graph convolutional network and variational structure of PDE \[13\], extended physics-informed neural networks (XPINNs) \[14\], parareal physics-informed neural network (PPINN) \[15\] and variational PINN \[16\] were devised to eliminate roadblocks in more complex and realistic applications.

In this paper, we propose a new method, PINN enforced by the continuous Lie symmetry information of PDEs (SPINN for brief), where the invariant surface conditions (ISC) induced by the Lie symmetries are incorporated into the loss function in PINN. If the PDEs together with the initial and boundary conditions admit a Lie symmetry, then the solutions are invariant under the Lie symmetry and also satisfy the ISC, i.e. the ISC are the inherent properties of the PDEs and place new essential constraints on the solutions. Since the standard loss function in PINN is the mean square error of PDE residual, thus adding the ISC to the loss function will definitely increase objective function value during the optimization process and improve the accuracy of the solutions. Our method is fundamentally different from those prior approaches which use parity-symmetry in the framework of neural networks \[11\] or the information of conservation law in the loss function \[9\], and thus the first time to embed the continuous symmetries into the loss functions. Furthermore, we perform two groups of ten independent experiments for the three PDEs respectively and show that SPINN largely outperforms than PINN, even in the worst case of SPINN the results is better than the best one of PINN under the same training data set and initializations. In particular, the symmetries admitted by the PDEs together with the initial and boundary conditions are further studied in \[1\]-\[17\]-\[18\] and thus the proposed SPINN has broad application prospects. It is worthy of saying that the non-classical symmetry, which was introduced by Bluman and Cole to obtain new exact solutions of the linear heat equation, is also effective for generating ISC \[19\].

The remainder of the paper is arranged as follows. In the following section, we first briefly review the main idea of PINN and the framework of Lie symmetry of PDEs, and introduce SPINN in detail. In Section 3, two sets of ten independent experiments are performed for the heat equation, KdV equation and the potential Burgers equation to illustrate the effectiveness of SPINN. The computational cost of SPINN is also discussed, showing that the error accuracy of the numerical solutions by SPINN is greatly improved without obvious increases of the computational cost. We conclude the results in the last section.
2 Main ideas of the methods

In this section, we take the following system of two \( r \)-th order PDEs

\[
\begin{align*}
  f &= u_t + N_u[u,v] = 0, \\
  g &= v_t + N_v[u,v] = 0, \\
  t &\in [0,T], \quad x \in \Omega,
\end{align*}
\]

together with the initial and boundary conditions

\[
\begin{align*}
  I(x,u,v) &= 0; \\
  B(t,u,v) &= 0, \quad \text{on} \ \partial \Omega,
\end{align*}
\]

as an example to introduce the main ideas of the methods, where \( u = u(t,x) \) and \( v = v(t,x) \) are the solutions to be determined, \( \Omega \) denotes a finite interval, \( N_u[u,v] \) and \( N_v[u,v] \) denote the smooth functions of \( u, v \) and their \( x \)-derivatives up to \( r \)-th order.

2.1 PINN for solving PDEs

The main idea of PINN for solving PDEs is to construct a neural network \((\tilde{u}(t,x,\theta), \tilde{v}(t,x,\theta))\) to approximate the exact solution \((u(t,x), v(t,x))\) via the trainable parameters \( \theta \). The usual activation function in the neural network is the hyperbolic tangent (tanh) function while the weights and bias are initialized by the Xavier initialization and the derivatives of the network \((u,v)\) with respect to time \( t \) and space \( x \) are derived by automatic differentiation \[3\]. Meanwhile, PINN method usually utilize the Adam or the L-BFGS algorithms to minimize the loss functions of mean square error (MSE)

\[
MSE = w_bMSE_b + w_iMSE_i + w_fMSE_f + w_gMSE_g,
\]

where \( MSE_i \) and \( MSE_b \) correspond to the initial and boundary data while \( MSE_f \) and \( MSE_g \) enforce the structure imposed by system (1) at a finite set of collocation points,

\[
\begin{align*}
  MSE_b &= \frac{1}{N} \sum_{i=1}^{N} \left| B(t^i, u^i, v^i) \right|^2, \\
  MSE_i &= \frac{1}{N} \sum_{i=1}^{N} \left| I(x^i, u^i, v^i) \right|^2, \\
  MSE_f &= \frac{1}{N} \sum_{j=1}^{N_f} \left| f(\tilde{t}_j, \tilde{x}_j) \right|^2, \quad MSE_g = \frac{1}{N_g} \sum_{j=1}^{N_g} \left| g(\tilde{t}_j, \tilde{x}_j) \right|^2,
\end{align*}
\]

where and \( w_b, w_i, w_f \) and \( w_g \) are the weights, \( \{t^i, x^i, u^i, v^i\}_{i=1}^{N} \) are the initial and boundary training data set and \( \{\tilde{t}_j, \tilde{x}_j\}_{j=1}^{N} \) denote the collocation points for \( f \) and \( g \). Note that in this study we choose the weights \( w_b = w_i = w_f = w_g = 1 \) in both PINN and the SPINN below, thus in SPINN we do not state them again.

Generally speaking, the total number of training datas \( N \) is relatively small (a few hundred up to a few thousand points) while the number of collocations points are large. Alternatively, one can enforce the boundary or initial conditions exactly and automatically by modifying the network architecture, which eliminates the loss term of boundary conditions \[20, 21\].
2.2 Symmetry of PDEs

The classical method for obtaining Lie symmetry admitted by Eq. (1) is to find a local one-parameter Lie group of infinitesimal transformation

\[ x^* = x + \varepsilon \xi(x, t, u, v) + O(\varepsilon^2), \]
\[ t^* = t + \varepsilon \tau(x, t, u, v) + O(\varepsilon^2), \]
\[ u^* = u + \varepsilon \eta(x, t, u, v) + O(\varepsilon^2), \]
\[ v^* = v + \varepsilon \phi(x, t, u, v) + O(\varepsilon^2), \]

which leaves system (1) invariant. Lie’s fundamental theorem shows that such group is completely characterized by the infinitesimal operator \( X = \xi \partial_x + \tau \partial_t + \eta \partial_u + \phi \partial_v \), where we briefly denote \( \xi = \xi(x, t, u, v), \tau = \tau(x, t, u, v), \eta = \eta(x, t, u, v) \) and \( \phi = \phi(x, t, u, v) \), thus we will not differentiate the Lie group (5) and the corresponding operator \( X \) and call them as Lie symmetry. Then Lie’s infinitesimal criterion requires \( X \) satisfying

\[ \text{pr}^{(2)} X(f)_{|\{\Delta\}} - \text{pr}^{(2)} X(g)_{|\{\Delta\}} = 0, \]

where, here and below, the symbol \( \mathring{\mathcal{L}}(\Delta) \) means that the computations work on the solution space of \( \Delta = 0 \), and

\[ \text{pr}^{(2)} X = X + \eta^{(1)}_t \partial_{ut} + \eta^{(1)}_x \partial_{ux} + \cdots + \eta^{(r)}_x \partial_{ur} + \phi^{(1)}_t \partial_{vt} + \phi^{(1)}_x \partial_{vx} + \cdots + \phi^{(r)}_x \partial_{vr}, \]

stands for \( r \)-th order prolongation of \( X \), \( u_i = \partial^i u/\partial x^i \) with \( i = 1, 2, \ldots, r \), the coefficients \( \eta^{(1)}_t, \eta^{(1)}_x \) and \( \phi^{(1)}_t, \phi^{(1)}_x \) can be calculated by the well-known prolongation formulae [1][2],

\[ \eta^{(1)}_t = D_t (\eta - \tau u_t - \xi u_x) + \tau D^2_t u + \xi D_x D_t u, \]
\[ \eta^{(1)}_x = D^i_x (\eta - \tau u_t - \xi u_x) + \tau D^i_x D_u u + \xi D^i_x D^2_x u, \]
\[ \phi^{(1)}_t = D_t (\phi - \tau v_t - \xi v_x) + \tau D^2_t v + \xi D_x D_t v, \]
\[ \phi^{(1)}_x = D^i_x (\phi - \tau v_t - \xi v_x) + \tau D^i_x D^2_x v + \xi D^i_x D_v v. \]

The symbols \( D_t \) and \( D_x \) in (7) indicate the total derivatives with respect to \( t \) and \( x \) respectively,

\[ D_t = \partial_t + u_t \partial_u + u_x t \partial_{ux} + u_{tt} \partial_{ut} + \cdots, \]
\[ D_x = \partial_x + u_t \partial_u + u_x \partial_{ux} + u_{tt} \partial_{ux} + \cdots, \]

and \( D^0_t(u) = u, D^0_x = D_t (D_t^{-1}) \) and similar for \( D_x \).

The Lie symmetry admitted by (1) can be used to construct similarity solutions which also satisfy the ISC

\[ ISC_1 := \eta - \tau u_t - \xi u_x = 0, \quad ISC_2 := \phi - \tau v_t - \xi v_x = 0. \]

Hence, the ISC (8) add new constraints for finding solutions of the system (1) if the system admits the Lie symmetry (5). Moreover, the methods for finding symmetries of system (1) with initial-boundary value conditions (2) are well-studied [1][7][8], thus it is feasible to use the ISC to improve the accuracy of learned solutions by deep neural network.

If the system (1) is compatible with the ISC (8), then on the common solution manifold of the two systems, the conditions

\[ \text{pr}^{(2)} X(f)_{|\{1, 8\}} - \text{pr}^{(2)} X(g)_{|\{1, 8\}} = 0, \]

generate the non-classical symmetry which obviously contains Lie symmetry as a particular case [9]. The non-classical symmetry also keeps the graph of common solutions of systems (1) and (8) invariant and thus exerts the same role of Lie symmetry to learn high precision solutions by deep neural network.
2.3 SPINN for solving PDEs

In addition to considering PDEs, there exist no ‘physical’ elements expressed in PINN [3]. However, the predicted solutions of PDEs via the PINN contain the physical properties, such as the Lie symmetries in subsection 2.2, which are not reflected in the training procedure of neural networks. Observe that the solutions of the initial boundary problem associated with system (1) are also restricted by the ISC, thus it is reasonable to embed the ISC into the loss functions of PINN to further improve the efficiency of neural network. Meanwhile, the considered symmetries, Lie symmetry or non-classical symmetry, characterize the inherent properties of PDEs and thus their induced ISC can accelerate the learning efficiency.

Specifically, the loss function in SPINN is

\[ \text{MSE} = \text{MSE}_i + \text{MSE}_b + \text{MSE}_f + \text{MSE}_g + \text{MSE}_{ISC}, \]  \hspace{1cm} (9)

where \( \text{MSE}_i, \text{MSE}_b \) and \( \text{MSE}_f, \text{MSE}_g \) are defined in (4) and

\[ \text{MSE}_{ISC} = \frac{1}{N} \sum_{i=1}^{N} (| \text{ISC}_1(\tilde{t}_j, \tilde{x}_j, \tilde{u}, \tilde{v})|^2 + | \text{ISC}_2(\tilde{t}_j, \tilde{x}_j, \tilde{u}, \tilde{v})|^2 ). \]

We state the whole procedure of SPINN in Algorithm 1 and depict the schematic diagram in Figure 1.

---

**Algorithm 1 : Steps for SPINN**

**Input:** Initial and boundary data set: \( \{ t^i, x^i, u^i, v^i \}_{i=1}^{N} \); Initial value of weights and bias: Xavier initialization; Loss functions: MSE in (9);

**Output:** Learned solutions: \( (\tilde{u}, \tilde{v}) \) and MSE;

1: Find the Lie symmetry or non-classical symmetry of PDEs: \( \mathcal{X} \);
2: Get the ISC according the obtained the Lie symmetry or non-classical symmetry of PDEs: \( \text{ISC}_1 \) and \( \text{ISC}_2 \);
3: Train the neural network with the loss function given in (9);
4: return the error and predicted solution \( (\tilde{u}, \tilde{v}) \) satisfying the given error bound;

---

Compared with PINN, the optimization problem in SPINN is to find the minimum value of the loss function, composed of PDEs residual and the ISC residual, by optimizing the weights and biases. As we will show in the numerical experiments, SPINN improves the accuracy of predicted solutions, and requires less training points and simpler networks to achieve the accuracy of PINN with more training points.

3 Numerical experiments

In this section, we apply the proposed SPINN to study three PDEs: KdV equation with the time and space translation symmetry, Heat equation with Lie symmetry and the potential Burgers equation with non-classical symmetry. In the three examples, we choose the tanh as the activation function and use the L-BFGS algorithm, a full batch gradient descent optimization algorithm based on the quasi-Newton method [22], to optimize the loss function to gather the \( L_2 \) norm error.

For each example, keep all other elements of neural network and data unchanged, we perform ten independent experiments for two cases respectively: the changes of collocation points and the changes of numbers of neurons. The experiment results show that SPINN outperforms than PINN with the same architecture and parameters.
3.1 KdV equation

The nonlinear KdV equation reads as

\[ u_t + uu_x + u_{xxx} = 0, \quad x \in [0, 1], \quad t \in [0, 1], \quad (10) \]

which arises in the theory of long waves in shallow water and the physical systems in which both nonlinear and dispersive effects are relevant \[23\], and the initial boundary conditions are

\[ u(0, x) = 12 \text{sech}^2 x, \quad u(t, 0) = 12 \text{sech}^2(-4t), \quad u(t, 1) = 12 \text{sech}^2(1 - 4t). \quad (11) \]

Eq. (10) together with the initial boundary conditions (11) are admitted by the combination of time and space translation symmetry \( X_{kdv} = \partial t + c \partial x \), where \( c = 4 \) is called the speed of travelling wave.

By means of the symmetry \( X_{kdv} \), we get a celebrated soliton solution \( u = 12 \text{sech}^2(x - 4t) \) and the ISC \( u_t + 4u_x = 0 \). Let

\[ f := u_t + uu_x + u_{xxx}, \]
\[ g := u_t + 4u_x. \]

Then the shared parameters of the neural networks \((u, f, g)\) with SPINN can be learned by minimizing the mean squared error

\[ MSE = MSE_u + MSE_f + MSE_g, \quad (12) \]

where

\[ MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u(0, x_i) - 12 \text{sech}^2(x_i) \right|^2\]
\[ + \left| u(t_i, 0) - 12\text{sech}^2(-4t_i) \right|^2 + \left| u(t_i, 1) - 12\text{sech}^2(1 - 4t_i) \right|^2 \right], \]

\[ MSE_f = \frac{1}{N} \sum_{j=1}^{\tilde{N}} \left| f(t_j, x_j) \right|^2, \quad MSE_g = \frac{1}{N} \sum_{j=1}^{\tilde{N}} \left| g(t_j, x_j) \right|^2. \]

where \( MSE_u \) corresponds to the loss on the initial and boundary data \( \{t_i, x_i, u_i\}_{i=0}^{N_u} \), \( MSE_f \) penalizes the KdV equation not being satisfied on the collocation points \( \{t_j, x_j\}_{j=0}^{\tilde{N}} \) while \( MSE_g \) corresponds to the loss of the ISC on the same collocation points. Note that the loss function of PINN is \( MSE = MSE_u + MSE_f \) which differentiates the one of SPINN is the ISC term \( MSE_g \).

To obtain the training data set, we divide the spatial region \( x \in [0, 1] \) and time region \( t \in [0, 1] \) into \( N_x = 256 \) and \( N_t = 100 \) discrete equidistance points respectively. Thus the solutions \( u \) is discretized into \( 256 \times 100 \) data points in the given spatiotemporal domain \( [0, 1] \times [0, 1] \).

To investigate the performance of PINN and SPINN for the KdV equation (10), we perform two groups of ten independent experiments to compare the prediction accuracy of the two methods where each initial seeds are selected randomly, the training points \( N_u = 100 \) are randomly sampled initial-boundary data set and the number of hidden layers is 2, and

Group I. The number of collocation points \( \tilde{N} \) varies from 50 to 2050 with step 100 and each layer has 20 neurons;

Group II. The number of neurons in each layer changes from 10 to 100 with step 5 simultaneously and the number of collocation points keep \( \tilde{N} = 700 \) via the Latin hypercube sampling method.

The \( L_2 \) relative errors of PINN and SPINN are displayed in Figure 2 where the left one corresponds to Group I and the right one is for Group II. The red and green lines respectively correspond to the mean errors of ten experiments while the shade regions depict the max-min errors of ten experiments where the nodes show the locations of the collocation points or neurons for training. In Group I, the mean values of \( L_2 \) relative errors for SPINN outperform at least one order of magnitude than PINN, even to three orders at \( \tilde{N} = 1500 \). SPINN reaches \( 10^{-5} \) \( L_2 \) relative error by using only 50 collocation points while PINN can not get the same accuracy.
within 2050 collocation points. Moreover, the range of $L_2$ relative errors via ten experiments for SPINN is much more smaller than PINN, which demonstrate that the $L_2$ relative error of SPINN is more stable than PINN. In Group II, as the increasing numbers of the neurons in graph B of Figure 2 the $L_2$ relative errors of SPINN always keep stable around $10^{-4}$ and far better than the ones of PINN which fluctuate around $2 \times 10^{-3}$ and have big fluctuations at 10, 15 and 90 neurons respectively.

However, there exist intersections between the shade regions of SPINN and PINN, because the best $L_2$ relative error of PINN and the worst one of SPINN just overlap in certain cases of ten experiments. In fact, under the complete same conditions, SPINN still performs better than PINN. We show it by choosing the worst cases of SPINN in the two cases respectively and list their results explicitly in Table 1 where the error reduction rate (ERR) is computed according to the $L_2$ relative error of PINN ($Re_1$) and the one of SPINN ($Re_2$) \[13\],

$$ERR = \frac{Re_1 - Re_2}{Re_1}.$$  

**Table 1: KdV equation: $L_2$ relative errors of PINN and SPINN and ERR**

| Solution         | Method   | PINN       | BPINN      | ERR       |
|------------------|----------|------------|------------|-----------|
| $u(N = 850)$     |          | 3.704e-04  | 1.065e-04  | 71.24%    |
| $u(20 \text{ neurons})$ |          | 6.628e-03  | 5.435e-05  | 99.18%    |

Table 1 shows that, even in the worst cases of SPINN, the ERR still has large drops, 71.24% reduction at 850 collocation points and two orders of magnitude reduction at 20 neurons. Furthermore, in the two worst cases, we compute the absolute errors of PINN and SPINN and show them in Figure 3. From the error distribution graphs A for Group I in Figure 3 the absolute errors of the PINN mainly distribute on the interval $(10^{-4}, 10^{-2})$ while the absolute errors of the SPINN gather on $(2 \times 10^{-5}, 2 \times 10^{-3})$. The peak value of SPINN appear at $5 \times 10^{-4}$ and the peak value of PINN emerges at $10^{-3}$, thus SPINN is more easily to get high precision numerical solutions of the KdV equation. And the absolute errors of SPINN with one order of magnitude improvement appear much usually. For the Group II, in graph D, the distribution of absolute error of SPINN gathers on $(10^{-5}, 2 \times 10^{-3})$ which is remarkable superior than $(10^{-1}, 10^{-3})$ of PINN, while the peak value of SPINN is about $2 \times 10^{-5}$ which is more close to the zero than PINN and thus SPINN can usually give higher precision solutions of the KdV equation. The three-dimensional distribution of absolute error in C (Group I) and F (Group II) for SPINN are stable and all far better than the ones in B (Group I) and E (Group II) for PINN which have big fluctuations.

The experiment results for KdV equation show that enforcing the inherent physical properties, the ISC induced by the Lie symmetry, to the loss function can further constrain the PDEs and thus learn more accurate data-efficient solutions. As expected, the accuracy of the SPINN is higher than that of the PINN, and in some cases even three orders of magnitude higher. In addition, the travelling transformation, $\partial t + c \partial x$ with wave speed $c > 0$, leaves invariant all the PDEs in which the independent variables $t$ and $x$ do not appear explicitly, thus the proposed SPINN has a good prospect of application and extension.

### 3.2 Heat equation

We consider the linear heat equation

$$f := u_t - u_{xx} = 0,$$  

(14)
subject to the initial and boundary conditions

\[
\begin{align*}
    u\left(\frac{1}{2}, x\right) &= 2\sqrt{2} x \, e^{-\frac{x^2}{2}}, \\
    u(t, 0) &= 0, \quad u(t, 1) = t^{-\frac{3}{2}} e^{-\frac{x^2}{4}}. 
\end{align*}
\]

(15)

Physically, the initial and boundary problems (14) and (15) represent a model for the heat flow in an insulated wire of which the two ends are kept at 0°C and the time-dependent temperature \(t^{-3/2} e^{-1/(4t)}\) respectively, and the initial temperature distribution at \(t = 1/2\) is given as \(2\sqrt{2} x \, e^{-x^2/2}\) [24].

Eq. (14) together with (15) admit a Lie symmetry \([2]\)

\[
X_{\text{heat}} = x t \partial_x + t^2 \partial_t - \left(\frac{x^2}{4} + \frac{t}{2}\right) u \partial_u,
\]

(16)

which generates an exact solution \(u = x t^{-3/2} e^{-x^2/4}\). Meanwhile, the ISC induced by \(X_{\text{heat}}\) is

\[
g := \left(\frac{x^2}{4} + \frac{t}{2}\right) u + x t u_x + t^2 u_t = 0.
\]

(17)

To obtain the training data, we divide the time region \(t \in [0.5, 1.5]\) into \(N_t = 100\) and spatial region \(x \in [0, 1]\) into \(N_x = 256\) equidistance points respectively, then the solution \(u\) is discredited into \(100 \times 256\) data points in the given spatio-temporal domain \([0.5, 1.5] \times [0, 1]\). The loss function of SPINN is given by

\[
MSE = MSE_u + MSE_f + MSE_g.
\]
where

\[ MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left[ |u(\frac{1}{2}, x_i) - 2\sqrt{2} x_i e^{-\frac{x_i^2}{2}}|^2 + |u(t_i, 0)|^2 + |u(t_i, 1) - \frac{1}{t_i^2} e^{-\frac{1}{t_i^2}}|^2 \right], \]

\[ MSE_f = \frac{1}{\tilde{N}} \sum_{j=1}^{\tilde{N}} |f(\tilde{t}_j, \tilde{x}_j)|^2, \quad MSE_g = \frac{1}{\tilde{N}} \sum_{j=1}^{\tilde{N}} |g(\tilde{t}_j, \tilde{x}_j)|^2. \]

where \( MSE_u \) corresponds to the loss on the initial and boundary data \( \{t_i, x_i, u_i\}_{i=0}^{N_u}, MSE_f \) and \( MSE_g \) corresponds to the loss functions of Eq.(14) and the ISC on the collocation points \( \{\tilde{t}_j, \tilde{x}_j\}_{j=0}^{\tilde{N}} \) respectively.

To compare the performances of PINN and SPINN for Eq.(14), we perform two groups of ten independent experiments where one group aims at the influences of number of collocation points, from 500 to 2500 with step 100, and the other considers the effects of the variation of number of neurons per layer, from 5 to 100 with step 5. In the experiments, the initial seeds are randomly selected to compare the \( L_2 \) relative errors of the two methods which are shown in Figure 4. In Figure 4, the left graph shows the variations of \( L_2 \) relative errors of the two methods under different collocation points \( \tilde{N} \), where the training data \( N_u = 100 \) randomly are sampled points from the initial-boundary data set and the neural network is a 3-layer with 40 neurons per layer. The green line for SPINN slightly fluctuate around \( 10^{-4} \) which are particular better than the one of PINN which is depicted by red line with the mean value \( 10^{-3} \). The green shaded area denotes \( L_2 \) relative errors of ten experiments for SPINN and remains relatively steady, fluctuating from \( 5 \times 10^{-5} \) to \( 2 \times 10^{-4} \), while the amplitude of red shaded for PINN is obvious amplified. For the group of variations of neurons, we randomly select \( N_u = 150 \) training points and proceed by sampling \( \tilde{N} = 1000 \) collocation points via the Latin hypercube sampling method, and fix the deep neural network as 3-layer. The right graph in Figure 4 shows the effects of variations of

![Figure 4: Heat equation: Comparison of \( L_2 \) relative errors of PINN and SPINN. (A) Keeping the number of training points unchanged, the \( L_2 \) relative errors of \( u \) for PINN and SPINN using different numbers of collocation points. (B) Keeping the number of training points unchanged, the \( L_2 \) relative errors of \( u \) for PINN and SPINN using different numbers of neurons. The line and shaded region represent the mean and max-min of 10 independent runs.](image)
neurons for the two methods where the green line of SPINN has no big fluctuations and is far below the red line of PINN. Except for the two unhoped cases $\tilde{N} = 1800$ in graph A and the 80 neurons in graph B, the two shade regions has no intersection totally which demonstrate that SPINN outperform PINN better.

In order to further show the superiority of SPINN, we choose the worst cases of SPINN in the above two group experiments and listed the $L_2$ relative errors and ERR defined by (13) in Table 2. Even in the worst cases of SPINN there has big improvements of $L_2$ relative errors, i.e.

| Solution                  | Method | PINN      | SPINN     | ERR     |
|---------------------------|--------|-----------|-----------|---------|
| $u(N = 1800)$             | PINN   | 4.5847e-04| 1.4963e-04| 67.36%  |
| $u(neurons = 80)$         | SPINN  | 6.5529e-04| 1.5036e-04| 77.34%  |

67.36% and 77.34%. Furthermore, the absolute error of two worst cases of SPINN are depicted in Figure 5. The graphes A and D shows the absolute error distributions of the two worst cases of SPINN respectively, where the distribution of SPINN in the two groups are more close to zero. It is obvious that SPINN has greater superiority than PINN. The graphes B and E further shows the absolute error surfaces of $u$ for PINN, where both of them have big fluctuations and thus PINN is sensitive to the variations of collocation points and neurons, but the graphes C and F for SPINN exhibit flat tendency. Obviously, the absolute errors are more stable for the two groups of experiments.

Figure 5: Heat equation: Two graph sets (A,B,C) and (D,E,F) correspond to the two worst cases in the left and right experiments in Figure 4. (A and D): Absolute error distributions of $u$ for the two worse cases. Comparisons of absolute errors between PINN and SPINN for the two worse cases (B and C for the left experiment in Figure 4 E and F for the right one).
3.3 Potential Burgers equations

The last example is the nonlinear potential Burgers equations

\[ f(t, x) := v_x - u = 0, \]
\[ g(t, x) := v_t - u_x + \frac{1}{2}u^2 = 0, \]  \(18\)

together with the initial and boundary conditions

\[ u(0, x) = -\frac{4(x + 2)}{x(4 + x)}, \quad v(0, x) = -2 \ln \left(\frac{x}{3} + \frac{x^2}{12}\right), \]
\[ u(t, 0.1) = \frac{-840}{200t + 41}, \quad v(t, 0.1) = -2 \ln \left(\frac{t}{6} + \frac{41}{1200}\right), \]
\[ u(t, 1.1) = \frac{-1240}{200t + 561}, \quad v(t, 1.1) = -2 \ln \left(\frac{t}{6} + \frac{187}{400}\right). \]  \(19\)

The compatible condition \(v_{xt} = v_{tx}\) in system (18) yields the celebrated Burgers equation \(u_t + uu_x - u_{xx} = 0\) which describes the interaction of convection and diffusion in turbulent fluid and is similar with Navier-Stokes equations \([25]\).

System (18) together with initial and boundary conditions (19) admit a non-classical symmetry \([26]\)

\[ X_{burgers} = -\frac{1}{x + 1} \partial x + \partial t + \frac{1}{(x + 1)^2} \left[ \frac{1}{6}(x + 1)ue^\frac{x}{2} - u - \frac{1}{3}e^\frac{x}{2} \right] \partial v + \frac{1}{3(x + 1)}e^\frac{x}{2} \partial u, \]

which generate a set of exact solution

\[ u(t, x) = \frac{-4(x + 2)}{2t + 4x + x^2}, \quad v(t, x) = -2 \ln \left(\frac{t}{6} + \frac{x}{3} + \frac{x^2}{12}\right). \]  \(20\)

The ISCs associated with the non-classical symmetry \(X_{burgers}\) are

\[ l(t, x) := u_t - \frac{1}{x + 1} u_x - \frac{1}{3(x + 1)} e^\frac{x}{2} = 0, \]
\[ p(t, x) := v_t - \frac{1}{x + 1} v_x - \frac{1}{(x + 1)^2} \left[ \frac{1}{6}(x + 1)ue^\frac{x}{2} - u - \frac{1}{3}e^\frac{x}{2} \right] = 0. \]

The shared parameters of the neural network \((u(t, x), v(t, x))\) can be learned by minimizing the mean squared error loss

\[ MSE = MSE_u + MSE_v + MSE_f + MSE_g + MSE_l + MSE_p, \]  \(21\)

where \(MSE_u\) and \(MSE_v\) correspond to the loss on the initial and boundary data \(\{t_i, x_i, u^i, v^i\}_{i=0}^\hat{N}\), \(MSE_f\) and \(MSE_g\) penalize the Burgers potential equations not being satisfied on the collocation points \(\{\tilde{t}_j, \tilde{x}_j\}_{j=0}^\hat{N}\), \(MSE_l\) and \(MSE_p\) correspond to the loss of ISC,

\[ MSE_u = \frac{1}{\hat{N}} \sum_{i=1}^{\hat{N}} \left[ \left| u(0, x_i) - \frac{4(x_i + 2)}{x_i(4 + x_i)} \right|^2 + \left| u(t_i, 0.1) + \frac{840}{200t_i + 41} \right|^2 \right. \]
\[ + \left. \left| u(t_i, 1.1) + \frac{1240}{200t_i + 561} \right|^2 \right]. \]
We divide the spatial region $x \in [0.1, 1.1]$ and time region $t \in [0, 1]$ into $N_x = 256$ and $N_t = 100$ discrete equidistance points respectively. Thus, the solutions $u$ and $v$ are discretized into $256 \times 100$ data points in the given spatio-temporal domain $[0.1, 1.1] \times [0, 1]$. In what follows, the variations of two indexes, collocation points and neurons, are used to verify the overall effectiveness of SPINN for system (18). We exhibit the $L_2$ relative error, defined by $\text{error}_u + \text{error}_v$, where $\text{error}_u$ and $\text{error}_v$ stand for $L_2$ relative error of $u$ and $v$ respectively, via PINN and SPINN in the left graph in Figure 6, where the numbers of collocation points $\tilde{N}$ varies from 50 to 2050 with step 100, the training points keeps unchanged $N_u = 100$ randomly sampled from the initial and boundary data set, and the neural network architecture is 2 layers with 60 neurons per layer. The green line for SPINN, the mean value of ten independent runs with random seeds, is far below the red line which stands for the mean value by PINN. The $L_2$ relative errors for SPINN witness a downward trend as the increasing of collocation points. Moreover, SPINN reaches $L_2$ relative error $3 \times 10^{-4}$ by using only 150 collocation points, while PINN can not reach the same accuracy with 2050 collocation points.

![Figure 6: Potential Burgers equations: Comparison of the $L_2$ relative errors of PINN and SPINN.](image)

(A) Keeping the number of training points unchanged, the sum of the $L_2$ relative errors of $u$ and $v$ for PINN and SPINN using different numbers of collocation points. (B) Keeping the number of layers unchanged, the sum of the $L_2$ relative errors of $u$ and $v$ for PINN and SPINN using different numbers of neurons. The line and shaded region represent the mean and max-min of 10 independent runs.

The right graph in Figure 6 shows $L_2$ relative error for the 2-layer neural network with the number of neurons per layer, varying from 5 to 100 with step 5, where the randomly training
points are $\hat{N} = 100$ and the collocation points are $\tilde{N} = 500$ by means of the Latin hypercube sampling method. Again, the green line for SPINN is totally below the red line for PINN, and SPINN gets $3 \times 10^{-4}$ by using only 15 neurons while PINN can not reach the same accuracy with 100 neurons. Moreover, as the numbers of neuron increase in graph B of Figure 6, the mean value of $L_2$ relative errors for SPINN keeps stable after 60 neurons but the one for PINN still has fluctuations.

However, we cannot guarantee that, among the 10 experiments, the worst $L_2$ relative error of SPINN is better than the best one of PINN, which leads to the intersection of the two shaded areas in Figure 3. However, it is true that in the case of the same initial seed, the error of SPINN is lower than that of PINN. We might as well respectively choose the worst case of SPINN from the two sets of experiments, and take them out separately to compare the $L_2$ relative error and absolute error with PINN. Table 3 shows the comparisons of the two worse case of SPINN with PINN where the ERR is defined by (13) and the improvements of $L_2$ relative errors in both cases for $u$ is better than $v$.

### Table 3: Potential Burgers equations: $L_2$ relative errors of PINN and SPINN and ERR.

| Solution          | Method   | PINN       | SPINN      | ERR     |
|-------------------|----------|------------|------------|---------|
| $u(\hat{N} = 250)$|          | 7.462e-04  | 1.954e-04  | 73.81%  |
| $v(\hat{N} = 250)$|          | 3.619e-04  | 2.099e-04  | 42.00%  |
| $u(\text{neurons} = 35)$ |          | 4.457e-04  | 9.613e-05  | 78.43%  |
| $v(\text{neurons} = 35)$ |          | 2.351e-04  | 1.050e-04  | 55.34%  |

Figure 7 shows the absolute errors of the two worse cases for SPINN where the graphs A-E depict for the case of $\tilde{N} = 250$ collocation points while the graphs F-J for the case of 35 neurons. The distributions of absolute error A and D for the both cases demonstrate that the tendency of peak values of SPINN is more close to zero than PINN, where the improvement of $u$ is more remarkable than $v$. The pictures of three-dimensional absolute error also expose the effectiveness of SPINN for $u$, approximately flat in the given region, is better than the one of $v$ which has extreme fluctuations in the same region, thus it is possible that the effects of $v$ mainly give rise to the intersections of the shaded area in Figure 6.

### 3.4 Computational cost of SPINN

The above three numerical experiments show that SPINN with the same network structure or collocation points achieves higher accuracy and is more stable than PINN. Since the loss functions in SPINN are added by the ISC, we take for granted that SPINN spends more training time than PINN. However, the final results overturn the idea. We use the relative computational cost of SPINN to PINN, defined by the training time of SPINN divided by the training time of PINN [8], to quantify the computational overhead of SPINN. All the computations in this section are performed using the Intel Core i5-11300H CPU.

For the KdV equation in Section 3.1, the relative computational costs for different numbers of collocation points and neurons in one trial are shown in Figure 8. In the left graph for the twenty-one collocation points, the relative computational costs below one are thirteen times, taking 62%, the others take 38% which includes six times between one and two and two times above two. While in the right graph for the nineteen neurons, the relative computational costs below one are twelve times, taking 63%, the others are seven times between one and two and take 37%. It means that SPINN for the KdV equation takes less time than PINN over 60% cases. Moreover, together with the fact in Figure 2 that SPINN with less collocation points or less neurons outperforms
Figure 7: Burgers potential equations: Two graph sets (A,B,C,D,E) and (F,G,H,I,J) correspond to the worst cases in the left and right experiments in Figure 6 respectively. (A and F): Absolute error distributions of $u$ and $v$ for the two worse cases. (B,C,D,E) and (G,H,I,J): Comparisons of absolute errors between PINN and SPINN for the two worse cases.
PINN, we find that SPINN for KdV equation has great advantages than PINN in terms of both accuracy and training time.

The left graph in Figure 8 presents the totally reverse results for the relative computational costs of the Heat equation with twenty one collocation points, 38% less than one and 62% bigger than one, but the relative computational costs bigger than one mostly activates around one and only two collocation points surpass two. While in the right graph for the cases of variations of neurons, the relative computational costs for all cases fluctuate around one and do not exceed 1.5. In Figure 10, the graph A for the collocation points shows the same results as the KdV equation but the graph B for the variations of neurons presents not good scenario where only seven cases are less than one.
Figure 10: Relative computational cost distribution of the Potential Burgers equations. (A) Keeping the network structure consistent, the number of collocation points varies from 50 to 2050 with step 100. (B) Keeping the number of selected data points unchanged, the number of neurons varies from 10 to 100 with step 5.

Furthermore, we list the average values of relative computational costs of SPINN to PINN for the above three examples respectively in Table 4. The average relative computational costs change between 0.93 to 1.26 and has at most 0.26 fluctuations around one. Therefore, though an additional loss term ISC is added into the loss function in SPINN, SPINN still has certain superiorities than PINN in terms of training time. For example, for the KdV equation with different numbers of neurons and the potential Burgers equation with different collocation points, the average relative computational costs are 0.97 and 0.93 respectively.

| Examples                     | Relative computational costs |
|------------------------------|-----------------------------|
| 3.1 KdV equation($\tilde{N}$) | 1.03                        |
| 3.1 KdV equation(neurons)    | 0.97                        |
| 3.2 Heat equation($\tilde{N}$) | 1.18                        |
| 3.2 Heat equation(neurons)   | 1.03                        |
| 3.3 Potential Burgers equations($\tilde{N}$) | 0.93                        |
| 3.3 Potential Burgers equations(neurons) | 1.26                        |

4 Conclusion

In this paper, we propose a new SPINN, which incorporates the ISC induced by Lie symmetry into the loss function of PINN, to improve the accuracy and reliability of solutions of PDEs. Numerical experiments for the three illustrated PDEs show that the proposed SPINN clearly outperforms PINN where the $L_2$ relative error, error reduction rate as well as the absolute error are used to test the effectiveness. Moreover, SPINN with less training points can reach a high-precision value which can not be obtained by PINN within the tested intervals. Moreover, the neural network architecture used in the experiments are comparably simple, 2-layer in KdV and potential Burgers equation and 3-layer in Heat equation, but the $L_2$ relative error of SPINN can reach $10^{-5}$ which
is seldom before, and generally has one order of magnitude improvement than PINN, even two orders of magnitude. Furthermore, the relative computational cost of the proposed SPINN to PINN does not increase dramatically, where only few particular cases exceed two times of PINN, while the average relative computational costs keep small increases or decreases. Our results further demonstrate that the deep neural network enforced by the inherent physical properties of PDEs such as the symmetry information is very effective for finding high precision numerical solutions of PDEs and is worthy of further exploring, for example, the collection of SPINN with the gradient-enhanced PINN, inserting generalized symmetry into the loss function of PINN, etc. Such works are under consideration and will be reported in the future.

Acknowledgements

The paper is supported by the Beijing Natural Science Foundation (No. 1222014), the National Natural Science Foundation of China (No. 11671014) and the Cross Research Project for Minzu University of China; R&D Program of Beijing Municipal Education Commission (Nos. KM202110009006 and KM201910009001).

Declarations of interest: none

References

[1] G.W. Bluman, S.C. Anco, Symmetry and Integration Methods for Differential Equations, Springer-Verlag: New York 2002.

[2] P.J. Olver, Applications of Lie Groups to Differential Equations, Springer, New York, 1993.

[3] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686-707.

[4] B.A. Pearlmutter, Automatic Differentiation in Machine Learning: a Survey. computer science, 2015.

[5] L. Wang, Z.Y. Yan, Data-driven rogue waves and parameter discovery in the defocusing nonlinear Schrödinger equation with a potential using the PINN deep learning, Phys. Lett. A 404 (2021) 127408.

[6] J. Pu, J. Li, Y. Chen, Solving localized wave solutions of the derivative nonlinear Schrödinger equation using an improved PINN method, Nonlinear Dynam. 105 (2021) 1723-1739.

[7] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208-228.

[8] J. Yu, L. Lu, X.H. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114823.

[9] N. Lin, Y. Chen, A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions, J. Comput. Phys. 457 (2022) 111053.

[10] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, P. Gentine, Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems, Phys. Rev. Lett. 126 (2021) 098302.
[11] W. Zhu, W. Khademi, E.G. Charalampidis, P.G. Kevrekidis, Neural networks enforcing physical symmetries in nonlinear dynamical lattices: The case example of the Ablowitz-Ladik model, Physica D 434 (2022) 133264.

[12] L. Yang, X.H. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comput. Phys. 425 (2021) 109913.

[13] H. Gao, M.J. Zahr, J.X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502.

[14] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002-2041.

[15] X. Meng, Z. Li, D. Zhang, G.E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, Comput. Methods Appl. Mech. Engrg. 370 (2020) 113250.

[16] E. Kharazmi, Z.Q. Zhang, G.E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547.

[17] Z.Y. Zhang Z, Y.F. Chen. Classical and nonclassical symmetries analysis for initial value problems, Phys. Lett. A, 374(9) (2010) 1117-1120.

[18] J. Goard, Finding symmetries by incorporating initial conditions as side conditions, Euro. J. Appl. Math. 19 (2008) 701-715.

[19] G.W. Bluman, J.D. Cole, The general similarity solution of the heat equation, J. Math. Mech. 18 (1969) 1025-1042.

[20] P.L. Lagari, L.H. Tsoukalas, S. Safarkhani, I.E. Lagaris, Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions, Int. J. Artif. Intell. Tools 29 (5) (2020).

[21] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105-B1132.

[22] D.C. Liu, J.Nocedal, On the limited memory BFGS method for large scale optimization. Math. Progr. 45 (1989) 503-528.

[23] D. J. Korteweg, G. de Vries, On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves, Phil. Mag. 39 (1895)422-443.

[24] D.V. Widder, The Heat Equation, Academic Press, New York, 1975.

[25] J.M. Burgers, A mathematical model illustrating the theory of turbulence author links open overlay panel, Adv. Appl. Mech. 1(1948) 171-199.

[26] M.L. Gandarias, M.S. Bruzon, Nonclassical potential symmetries for the Burgers equation, Nonlinear Anal. 71 (2009) e1826-e1834.