Data-Driven Solution of Nonlinear Partial Differential Equations by AI Algorithm

Dostdar Ali
Departmential Seminar(Speaker)

Department of Mathematics
Univrsity of Baltistan, Skardu
Gilgit Baltistan

May 2, 2024

Table of Contents

Abstract

- This discussion I will explore the fusion of physics informed neural networks and differential equations in the optimization of numerical solutions to nonlinear of partial differential equation (PDE).
- We delve into the architecture of physics-informed neural networks (PINNs), discussing their integration of known physics into the learning process.
- Through example and case study, I illustrate the efficacy of PINNs in diverse domains and highlight their potential to revolutionize computational modeling.

Abstract

- This discussion I will explore the fusion of physics informed neural networks and differential equations in the optimization of numerical solutions to nonlinear of partial differential equation (PDE).
- We delve into the architecture of physics-informed neural networks (PINNs), discussing their integration of known physics into the learning process.
- Through example and case study, I illustrate the efficacy of PINNs in diverse domains and highlight their potential to revolutionize computational modeling.

Abstract

- This discussion I will explore the fusion of physics informed neural networks and differential equations in the optimization of numerical solutions to nonlinear of partial differential equation (PDE).
- We delve into the architecture of physics-informed neural networks (PINNs), discussing their integration of known physics into the learning process.
- Through example and case study, I illustrate the efficacy of PINNs in diverse domains and highlight their potential to revolutionize computational modeling.

Limitations of Traditional Numerical Methods

- **Non-linearities** Inability to capture complex non-linearities and multi-scale phenomena in many real-world PDEs.
- **Physical Law's** Reliance on pre-defined mathematical models that may not fully represent the underlying physics.
- **Accuracy** Difficulty in handling irregular geometries and boundary conditions, leading to reduced accuracy.

Limitations of Traditional Numerical Methods

- **Non-linearities** Inability to capture complex non-linearities and multi-scale phenomena in many real-world PDEs.
- **Physical Law's** Reliance on pre-defined mathematical models that may not fully represent the underlying physics.
- **Accuracy** Difficulty in handling irregular geometries and boundary conditions, leading to reduced accuracy.

Limitations of Traditional Numerical Methods

- **Non-linearities** Inability to capture complex non-linearities and multi-scale phenomena in many real-world PDEs.
- **Physical Law's** Reliance on pre-defined mathematical models that may not fully represent the underlying physics.
- **Accuracy** Difficulty in handling irregular geometries and boundary conditions, leading to reduced accuracy.

Physics-Informed Neural Networks(PINNs)

Definition

Physics-informed neural networks are the type of universal function approximators that can embed the knowledge of any physical laws that govern a given data-set in learning process, and can be described by partial differential equations (PDEs).

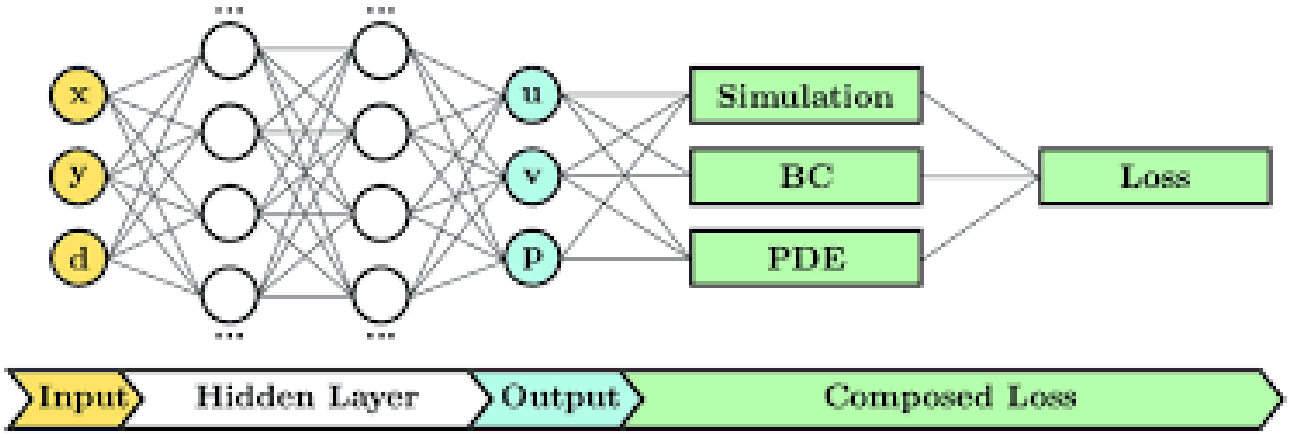

Figure: Integrating Neural Network with PDE.

Modeling and computation

A general nonlinear partial differential equation can be:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{1}$$

where, **u(t,x)** denotes the latent (hidden) solution, $N[.; \lambda]$ is a nonlinear differential operator, is a nonlinear operator parametrized by λ, and Ω is a subset of $\mathbb{R}^D$.

- This general form of governing equations summarizes a wide range of problems in mathematical physics, such as conservative laws, diffusion process, advection-diffusion systems, and kinetic equations.
  PINNs can be designed to solve two classes of problems:
- Data-driven solution of partial differential equations.
- Data-driven discovery of partial differential equations.

Modeling and computation

A general nonlinear partial differential equation can be:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{1}$$

where, **u(t,x)** denotes the latent (hidden) solution, $N[.; \lambda]$ is a nonlinear differential operator, is a nonlinear operator parametrized by λ, and Ω is a subset of $\mathbb{R}^D$.

- This general form of governing equations summarizes a wide range of problems in mathematical physics, such as conservative laws, diffusion process, advection-diffusion systems, and kinetic equations.
  PINNs can be designed to solve two classes of problems:
- Data-driven solution of partial differential equations.
- Data-driven discovery of partial differential equations.

Modeling and computation

A general nonlinear partial differential equation can be:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{1}$$

where, **u(t,x)** denotes the latent (hidden) solution, $N[.; \lambda]$ is a nonlinear differential operator, is a nonlinear operator parametrized by λ, and Ω is a subset of $\mathbb{R}^D$.

- This general form of governing equations summarizes a wide range of problems in mathematical physics, such as conservative laws, diffusion process, advection-diffusion systems, and kinetic equations.
  PINNs can be designed to solve two classes of problems:
- Data-driven solution of partial differential equations.
- Data-driven discovery of partial differential equations.

Modeling and computation

A general nonlinear partial differential equation can be:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{1}$$

where, **u(t,x)** denotes the latent (hidden) solution, $N[.; \lambda]$ is a nonlinear differential operator, is a nonlinear operator parametrized by λ, and Ω is a subset of $\mathbb{R}^D$.

- This general form of governing equations summarizes a wide range of problems in mathematical physics, such as conservative laws, diffusion process, advection-diffusion systems, and kinetic equations.
  PINNs can be designed to solve two classes of problems:
- Data-driven solution of partial differential equations.
- Data-driven discovery of partial differential equations.

Modeling: Data-driven solution of PDE's

Computes the hidden state **u(t,x)** of the system given boundary data and/or measurements z, and fixed model parameters λ. We solve:

$$u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{2}$$

- By defining the residual $f(t, x)$ as

$$f := u_t + N[u] = 0 \tag{3}$$

- Approximating **u(t,x)** by a deep neural network. This network can be differentiated using automatic differentiation.
- The parameters of **u(t,x)** and $f(t, x)$ can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{4}$$

Modeling: Data-driven solution of PDE's

Computes the hidden state **u(t,x)** of the system given boundary data and/or measurements z, and fixed model parameters λ. We solve:

$$u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{2}$$

- By defining the residual $f(t, x)$ as

$$f := u_t + N[u] = 0 \tag{3}$$

- Approximating **u(t,x)** by a deep neural network. This network can be differentiated using automatic differentiation.
- The parameters of **u(t,x)** and $f(t, x)$ can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{4}$$

Modeling: Data-driven solution of PDE's

Computes the hidden state **u(t,x)** of the system given boundary data and/or measurements z, and fixed model parameters λ. We solve:

$$u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{2}$$

- By defining the residual $f(t, x)$ as

$$f := u_t + N[u] = 0 \tag{3}$$

- Approximating **u(t,x)** by a deep neural network. This network can be differentiated using automatic differentiation.
- The parameters of **u(t,x)** and $f(t, x)$ can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{4}$$

Modeling: Data-driven solution of PDE's

Computes the hidden state **u(t,x)** of the system given boundary data and/or measurements z, and fixed model parameters λ. We solve:

$$u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{2}$$

- By defining the residual $f(t, x)$ as

$$f := u_t + N[u] = 0 \tag{3}$$

- Approximating **u(t,x)** by a deep neural network. This network can be differentiated using automatic differentiation.
- The parameters of **u(t,x)** and $f(t, x)$ can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{4}$$

Modeling: Data-driven discovery of PDE's

Computing the unknown state $u(t, x)$ and learning model parameters λ that best describe the observed data and it reads as follows:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{5}$$

- By defining $f(t, x)$ as

$$f := u_t + N[u; \lambda] = 0 \tag{6}$$

- Approximating $u(t, x)$ by a deep neural network, $f(t, x)$ results in a PINN. This network can be derived using automatic differentiation.
- The parameters of $u(t, x)$ and $f(t, x)$, together with the parameter λ of the differential operator can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{7}$$

Modeling: Data-driven discovery of PDE's

Computing the unknown state $u(t, x)$ and learning model parameters λ that best describe the observed data and it reads as follows:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{5}$$

- By defining $f(t, x)$ as

$$f := u_t + N[u; \lambda] = 0 \tag{6}$$

- Approximating $u(t, x)$ by a deep neural network, $f(t, x)$ results in a PINN. This network can be derived using automatic differentiation.
- The parameters of $u(t, x)$ and $f(t, x)$, together with the parameter λ of the differential operator can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{7}$$

Modeling: Data-driven discovery of PDE's

Computing the unknown state $u(t, x)$ and learning model parameters λ that best describe the observed data and it reads as follows:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{5}$$

- By defining $f(t, x)$ as

$$f := u_t + N[u; \lambda] = 0 \tag{6}$$

- Approximating $u(t, x)$ by a deep neural network, $f(t, x)$ results in a PINN. This network can be derived using automatic differentiation.
- The parameters of $u(t, x)$ and $f(t, x)$, together with the parameter λ of the differential operator can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{7}$$

Modeling: Data-driven discovery of PDE's

Computing the unknown state $u(t, x)$ and learning model parameters λ that best describe the observed data and it reads as follows:

$$u_t + N[u; \lambda] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{5}$$

- By defining $f(t, x)$ as

$$f := u_t + N[u; \lambda] = 0 \tag{6}$$

- Approximating $u(t, x)$ by a deep neural network, $f(t, x)$ results in a PINN. This network can be derived using automatic differentiation.
- The parameters of $u(t, x)$ and $f(t, x)$, together with the parameter λ of the differential operator can be then learned by minimizing the following loss function L_{tot}:

$$L_{tot} = L_u + L_f \tag{7}$$

Architecture and Training of PINNs

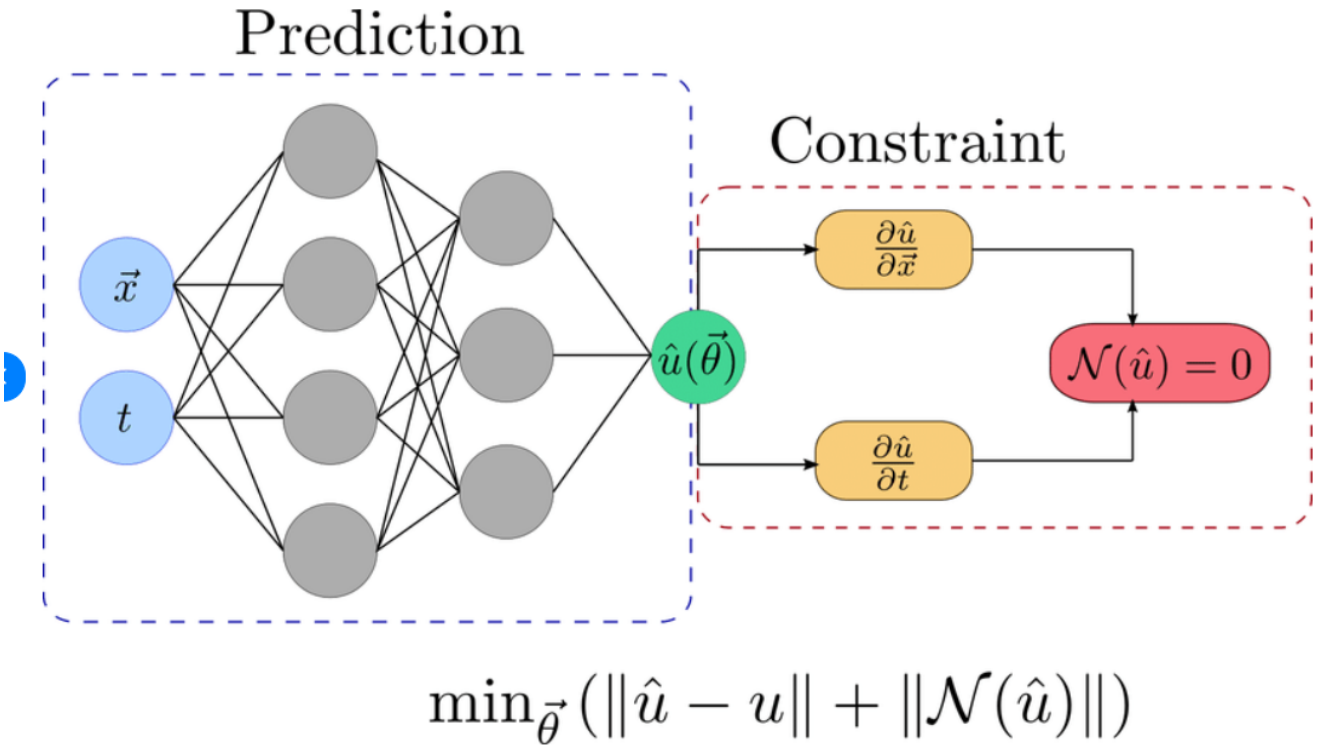

Figure: A sample schematic of a PINN architecture for the linear transport equation.

Modeling: Example Burgers' Equation

let us consider the Burgers' equation. In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1] \tag{8}$$

$$u(0, x) = -sin(\pi x) \tag{9}$$

$$u(t, -1) = u(t, 1) = 0 \tag{10}$$

- Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx} = 0, \tag{11}$$

- Proceed by approximating $u(t, x)$ by a deep neural network.

Modeling: Example Burgers' Equation

let us consider the Burgers' equation. In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1] \tag{8}$$

$$u(0, x) = -sin(\pi x) \tag{9}$$

$$u(t, -1) = u(t, 1) = 0 \tag{10}$$

- Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx} = 0, \tag{11}$$

- Proceed by approximating $u(t, x)$ by a deep neural network.

Modeling: Example Burgers' Equation

let us consider the Burgers' equation. In one space dimension, the Burger's equation along with Dirichlet boundary conditions reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1] \tag{8}$$

$$u(0, x) = -sin(\pi x) \tag{9}$$

$$u(t, -1) = u(t, 1) = 0 \tag{10}$$

- Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx} = 0, \tag{11}$$

- Proceed by approximating $u(t, x)$ by a deep neural network.

Architecture: for Burger Equation

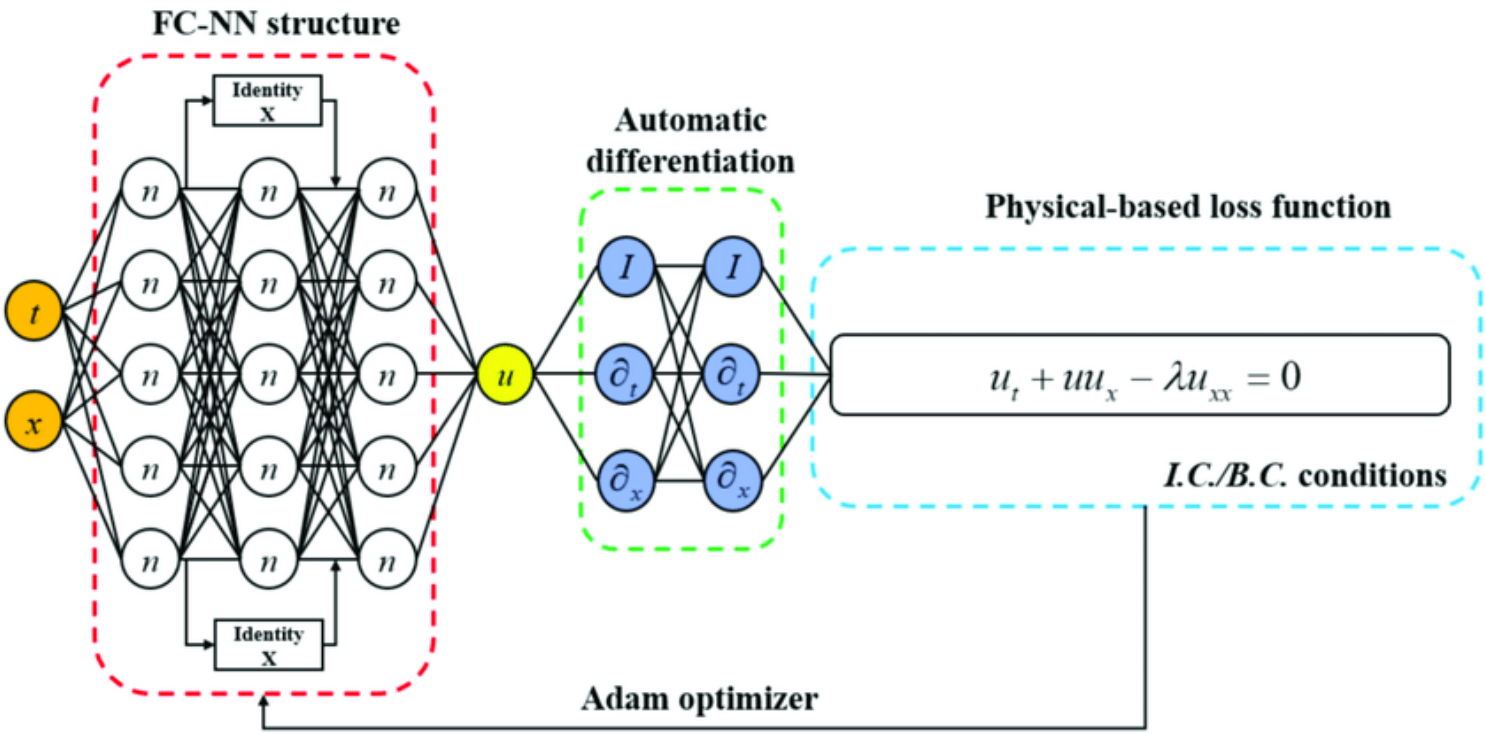

Figure: PINNs Structure Model for Burger Equation

Architecture: Behind Mathematical Working

The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss,

$$MSE = MSE_u + MSE_f \tag{12}$$

- where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \tag{13}$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \tag{14}$$

- here, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\{t_u^i, x_u^i\}_{i=1}^{N_u}$ specify the collocations points for $f(t, x)$. The loss MSE_u corresponds to the initial and boundary data while MSE_f enforces the structure by equation.

Architecture: Behind Mathematical Working

The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss,

$$MSE = MSE_u + MSE_f \tag{12}$$

- where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \tag{13}$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \tag{14}$$

- here, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\{t_u^i, x_u^i\}_{i=1}^{N_u}$ specify the collocations points for $f(t, x)$. The loss MSE_u corresponds to the initial and boundary data while MSE_f enforces the structure by equation.

Architecture: Behind Mathematical Working

The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss,

$$MSE = MSE_u + MSE_f \tag{12}$$

- where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \tag{13}$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \tag{14}$$

- here, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\{t_u^i, x_u^i\}_{i=1}^{N_u}$ specify the collocations points for $f(t, x)$. The loss MSE_u corresponds to the initial and boundary data while MSE_f enforces the structure by equation.

Key Findings: Solution

Predicted solution u(t, x) along with the initial and boundary training data.

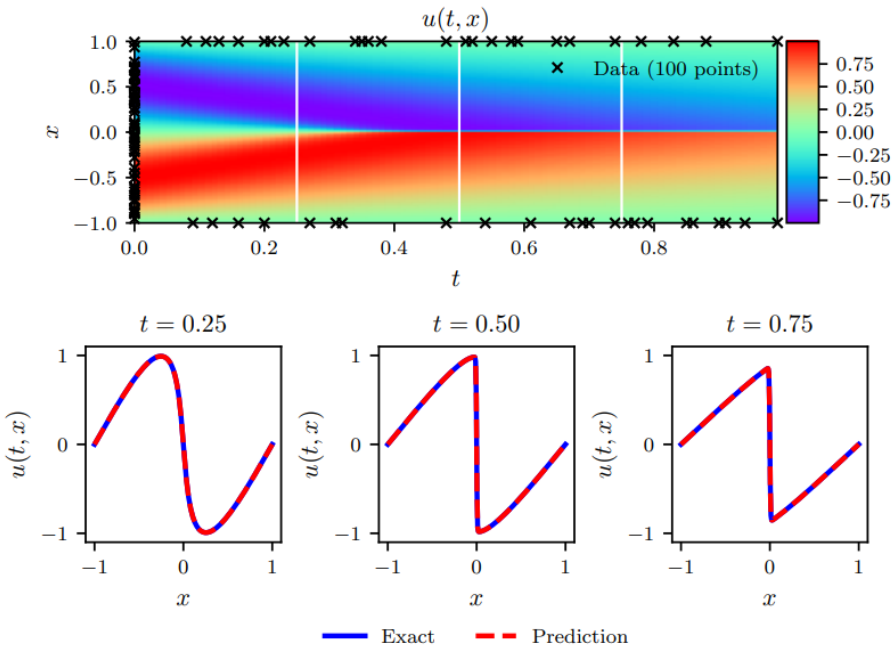

Figure: Solution of Burger Equation

- Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel. The relative L_2 error for this case is 6.7×10^4

Key Findings: Solution

Predicted solution u(t, x) along with the initial and boundary training data.

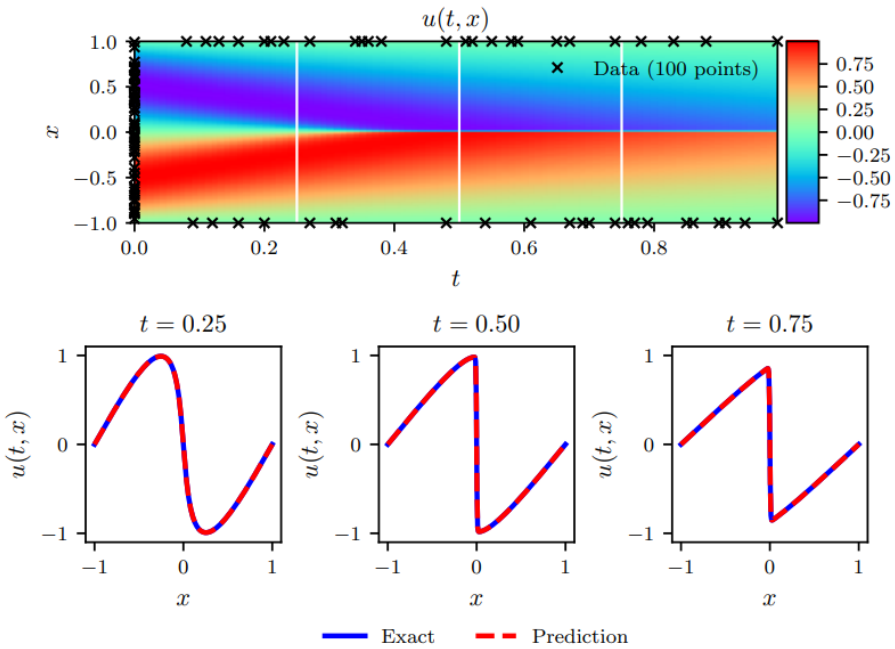

Figure: Solution of Burger Equation

- Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel. The relative L_2 error for this case is 6.7×10^4

Case study: Navier-Stokes Equation

- Our next example involves a realistic scenario of incompressible fluid flow as described by the Navier-Stokes equations. Navier-Stokes equations describe the physics of many phenomena.
- The Navier-Stokes equations in their full and simplified forms help with the design of the study of blood flow.
- Let us consider the Navier-Stokes equations in two dimensions (2D) given explicitly by

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{15}$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{16}$$

- where, $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and$p(t, x, y)$ the pressure. Here,$\lambda(\lambda_1\ ,\ \lambda_2)$ are the unknown parameters

Case study: Navier-Stokes Equation

- Our next example involves a realistic scenario of incompressible fluid flow as described by the Navier-Stokes equations. Navier-Stokes equations describe the physics of many phenomena.
- The Navier-Stokes equations in their full and simplified forms help with the design of the study of blood flow.
- Let us consider the Navier-Stokes equations in two dimensions (2D) given explicitly by

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{15}$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{16}$$

- where, $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and$p(t, x, y)$ the pressure. Here,$\lambda(\lambda_1\ ,\ \lambda_2)$ are the unknown parameters

Case study: Navier-Stokes Equation

- Our next example involves a realistic scenario of incompressible fluid flow as described by the Navier-Stokes equations. Navier-Stokes equations describe the physics of many phenomena.
- The Navier-Stokes equations in their full and simplified forms help with the design of the study of blood flow.
- Let us consider the Navier-Stokes equations in two dimensions (2D) given explicitly by

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{15}$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{16}$$

- where, $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and$p(t, x, y)$ the pressure. Here,$\lambda(\lambda_1 , \lambda_2)$ are the unknown parameters

Case study: Navier-Stokes Equation

- Our next example involves a realistic scenario of incompressible fluid flow as described by the Navier-Stokes equations. Navier-Stokes equations describe the physics of many phenomena.
- The Navier-Stokes equations in their full and simplified forms help with the design of the study of blood flow.
- Let us consider the Navier-Stokes equations in two dimensions (2D) given explicitly by

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{15}$$

$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{16}$$

- where, $u(t, x, y)$ denotes the x-component of the velocity field, $v(t, x, y)$ the y-component, and$p(t, x, y)$ the pressure. Here,$\lambda(\lambda_1\ ,\ \lambda_2)$ are the unknown parameters

Case study: Physics of Navier-Stokes

- Solutions to the Navier-Stokes equations are searched in the set of divergence-free functions; i.e.,

$$u_x + u_y = 0 \tag{17}$$

- This extra equation is the continuity equation for incompressible fluids that describes the conservation of mass of the fluid. We make the assumption that

$$u = \psi, \quad v = -\psi_x \tag{18}$$

for some latent function $\psi(t, x, y)$. Under this assumption, the continuity equation will be automatically satisfied.

- we are interested in learning the parameters λ as well as the pressure and define as,

$$f := u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{19}$$

$$g := v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{20}$$

Case study: Physics of Navier-Stokes

- Solutions to the Navier-Stokes equations are searched in the set of divergence-free functions; i.e.,

$$u_x + u_y = 0 \tag{17}$$

- This extra equation is the continuity equation for incompressible fluids that describes the conservation of mass of the fluid. We make the assumption that

$$u = \psi, \quad v = -\psi_x \tag{18}$$

for some latent function $\psi(t, x, y)$. Under this assumption, the continuity equation will be automatically satisfied.

- we are interested in learning the parameters λ as well as the pressure and define as,

$$f := u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{19}$$

$$g := v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{20}$$

Case study: Physics of Navier-Stokes

- Solutions to the Navier-Stokes equations are searched in the set of divergence-free functions; i.e.,

$$u_x + u_y = 0 \tag{17}$$

- This extra equation is the continuity equation for incompressible fluids that describes the conservation of mass of the fluid. We make the assumption that

$$u = \psi, \quad v = -\psi_x \tag{18}$$

  for some latent function $\psi(t, x, y)$. Under this assumption, the continuity equation will be automatically satisfied.
- we are interested in learning the parameters λ as well as the pressure and define as,

$$f := u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}) \tag{19}$$

$$g := v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}) \tag{20}$$

Note

and proceed by jointly approximating $[\psi(t,x,y)P(t,x,y)]$ using a single neural network with two outputs

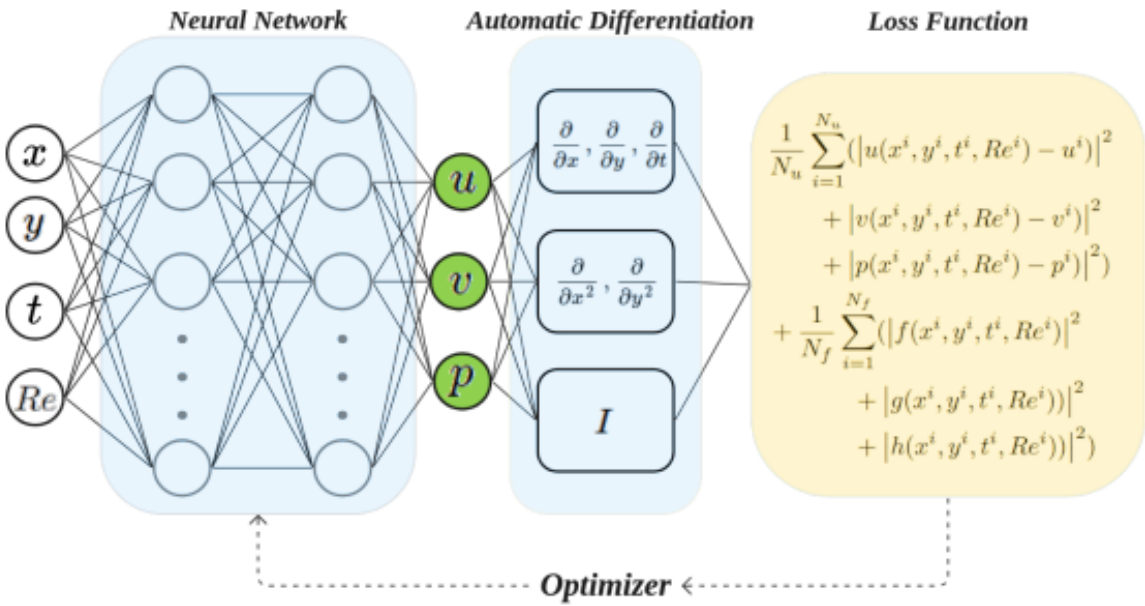

Figure: PINN architecture for solving Navier-Stokes Equation.

Key Findings: Solution of model PDE's, Exact and PINNs

The PINN's Predictions of vorticity (curl of the velocity field) for fluid flow around horizontal cylinder. The model is trained on velocity and pressure data at Re $=$ [100, 200, 333.$\bar{}$3, 400, 500], the vorticity is computed with automatic differentiation.

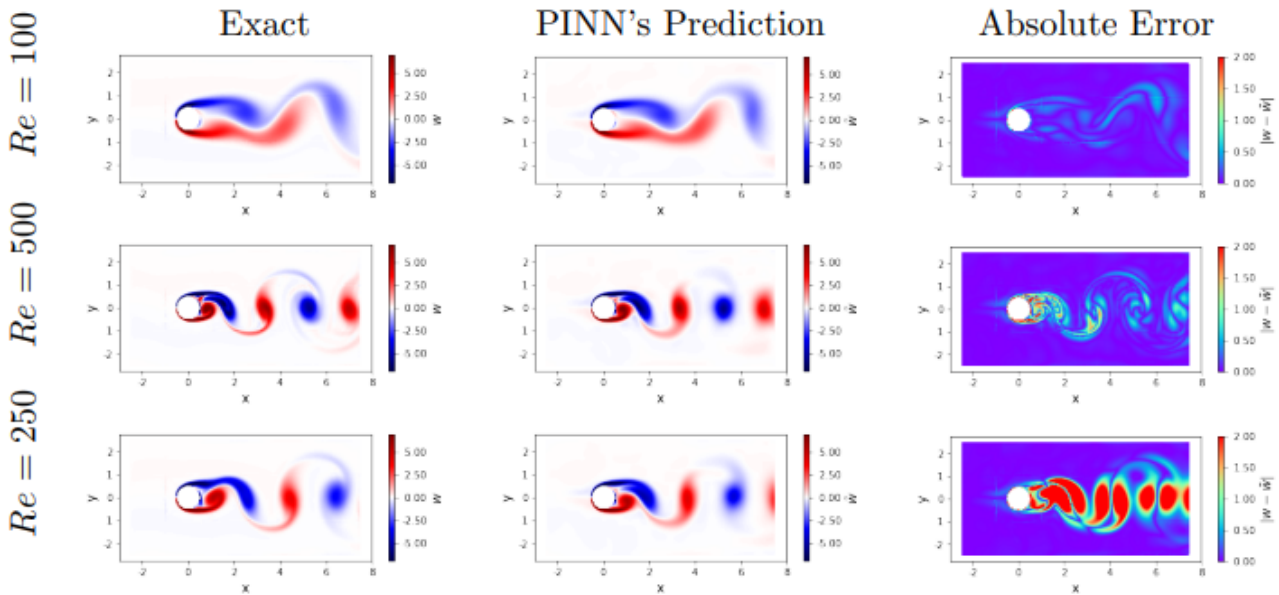

Figure: PINN architecture for solving Navier-Stokes Equation.

Take to Home Massage

- The seminar underscores the significance of physics-informed neural networks in addressing the challenges posed by traditional numerical methods in solving nonlinear PDEs.
- By embedding the knowledge of physical laws into the learning process, PINNs offer a promising approach to enhance accuracy and efficiency in computational modeling.
- The ability of PINNs to handle complex nonlinearities and irregular geometries opens up new possibilities for advancing numerical solutions in diverse domains.
- Overall, the seminar showcases the potential of AI algorithms in transforming the field of computational modeling through innovative approaches like PINNs.

Take to Home Massage

- The seminar underscores the significance of physics-informed neural networks in addressing the challenges posed by traditional numerical methods in solving nonlinear PDEs.
- By embedding the knowledge of physical laws into the learning process, PINNs offer a promising approach to enhance accuracy and efficiency in computational modeling.
- The ability of PINNs to handle complex nonlinearities and irregular geometries opens up new possibilities for advancing numerical solutions in diverse domains.
- Overall, the seminar showcases the potential of AI algorithms in transforming the field of computational modeling through innovative approaches like PINNs.

Take to Home Massage

- The seminar underscores the significance of physics-informed neural networks in addressing the challenges posed by traditional numerical methods in solving nonlinear PDEs.
- By embedding the knowledge of physical laws into the learning process, PINNs offer a promising approach to enhance accuracy and efficiency in computational modeling.
- The ability of PINNs to handle complex nonlinearities and irregular geometries opens up new possibilities for advancing numerical solutions in diverse domains.
- Overall, the seminar showcases the potential of AI algorithms in transforming the field of computational modeling through innovative approaches like PINNs.

Take to Home Massage

- The seminar underscores the significance of physics-informed neural networks in addressing the challenges posed by traditional numerical methods in solving nonlinear PDEs.
- By embedding the knowledge of physical laws into the learning process, PINNs offer a promising approach to enhance accuracy and efficiency in computational modeling.
- The ability of PINNs to handle complex nonlinearities and irregular geometries opens up new possibilities for advancing numerical solutions in diverse domains.
- Overall, the seminar showcases the potential of AI algorithms in transforming the field of computational modeling through innovative approaches like PINNs.

Appendix I

```
# To highlight the simplicity in implementing this idea let us i
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u


# Correspondingly, the physics informed neural network f(t,x) ta
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

Questions and Answers

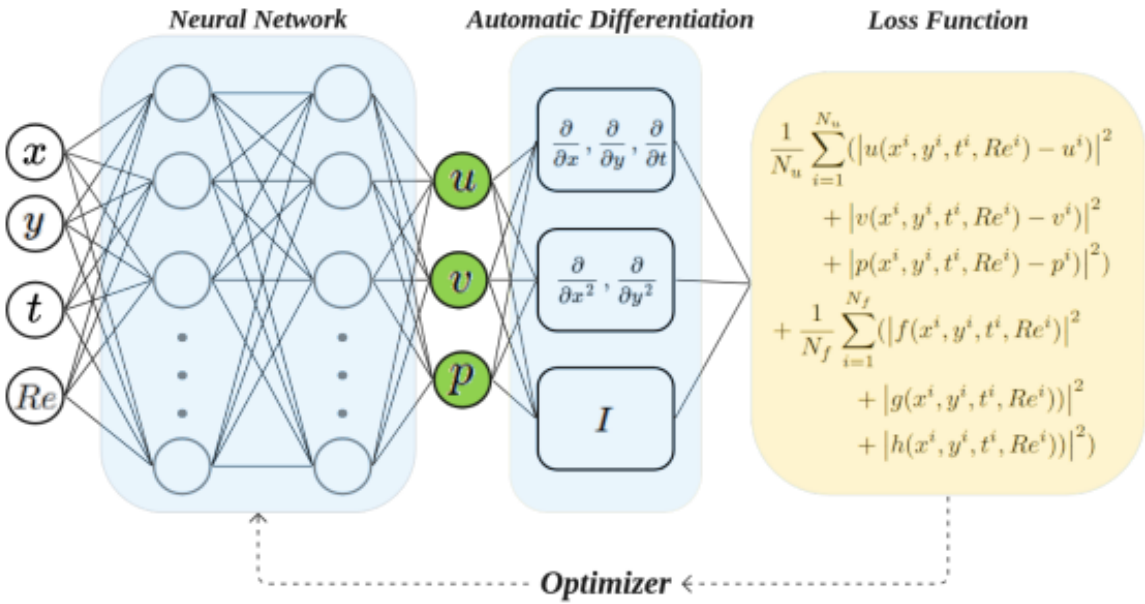

Figure: Searcher and git recourse.