Analysis of Advection-Diffusion-Reaction Equation using Physics-informed neural networks

Sandeep kumar

20312032

Prof: Ameeya Kumar Nayak

Introduction

- About Advection-Diffussion-Reaction
- Mathematical representation of ADR
- Solving ADR
- PINN
- Physics-informed loss
- Structure for PINN
- Strategies for training points selection
- First derivative constraint (FDC)
- PINN Model
- Analysis of solution
- Numerical result and discussion

What is Advection-Diffussion-Reaction Equations?

- Advection-diffusion is a mathematical model that describes the transport of a substance in a fluid medium.
- It combines two physical processes: advection, which is the movement of a substance with the flow of the fluid, and diffusion, which is the spreading of the substance due to its random motion.
- It also include a chemical or biological reaction that occurs within the transported substance. This model is commonly used in environmental engineering, bioengineering, and chemical engineering.

Mathematical representation of ADR

- The advection-diffusion-reaction (ADR) model can be mathematically represented by a set of coupled partial differential equations (PDEs).
- Let's consider a one-dimensional domain with a substance that is transported by advection, diffusion and undergoes a chemical reaction. The ADR model can be written as:
- The mass conservation equation: $\frac{\partial C}{\partial t} = k\frac{\partial^2 C}{\partial x^2} - v\frac{\partial C}{\partial x} - f$
- Where C, is concentration, v is the advection velocity, k is the diffusion coefficient, and f is the reaction function.
- The advection term, $v\frac{\partial c}{\partial x}$, represents the transport of the substance due to the fluid flow,
- The diffusion term $k\frac{\partial^2 c}{\partial x^2}$,represents the random motion of the substance.
- The reaction term, f, represents the chemical or biological reaction that occurs within the transported substance.

Solving ADR

- Under the advection-dominated situations the ADR behaves more like a hyperbolic equation.

$$\frac{\partial C}{\partial t} = v\frac{\partial C}{\partial x} - f$$

- Where under the diffusion dominated situations, the ADR behaves more like a parabolic equation.

$$\frac{\partial C}{\partial t} = k\frac{\partial^2 C}{\partial x^2} - f$$

- This brings some challenges in solving it with general numerical methods.

.

- A variety of numerical methods can be used to solve the ODE-PDE equation, including Eulerian, semi-Lagrangian, and Lagrangian methods.
- As these method use numerical discrete calculation format, none of them is thought to be really space-time continuous.so numerical method not effectively solve ADR. These only works to approximate the solution by simplifying the equation.
- Also, these traditional methods are computationally expensive in solving ADR problems.
- So deep learning (DL) technology can be an attractive alternative.

PINN

- PINN stands for Physics-Informed Neural Networks. It is a machine learning technique that combines deep neural networks with the principles of physics to solve partial differential equations (PDEs).
- To train a PINN, the network is first initialized with random weights and biases. The network is then trained using backpropagation to minimize the total loss function, which consists of the sum of the mean squared error loss for the PDE residual and the physical constraints loss.
- PINN has been successfully applied to a wide range of problems in fluid dynamics, solid mechanics, and other fields.

PINN

- PINN is constructed by the NN structure and a physics informed loss function, where the solution of the PDE is approximated by NN.

- NN is defined as :-
- Input layer: $N^0 = X$,
- Hidden layers: $N^k = \sigma\left(W^k N^{k-1} + b^k\right) \quad \text{for } 1 \leq k \leq L-1$
- Output layer: $\mu_\theta(\text{X}) = \text{y} = N^L = \sigma\left(W^L N^{l-1} + b^l\right)$

- Where N^k , W^k, and b^k are the neuron, weight, and bias of the kth layer, respectively; $\sigma(.)$ is the activation function
- $\mu_\theta(.)$ is the solution approximated by NN

Physics-informed loss

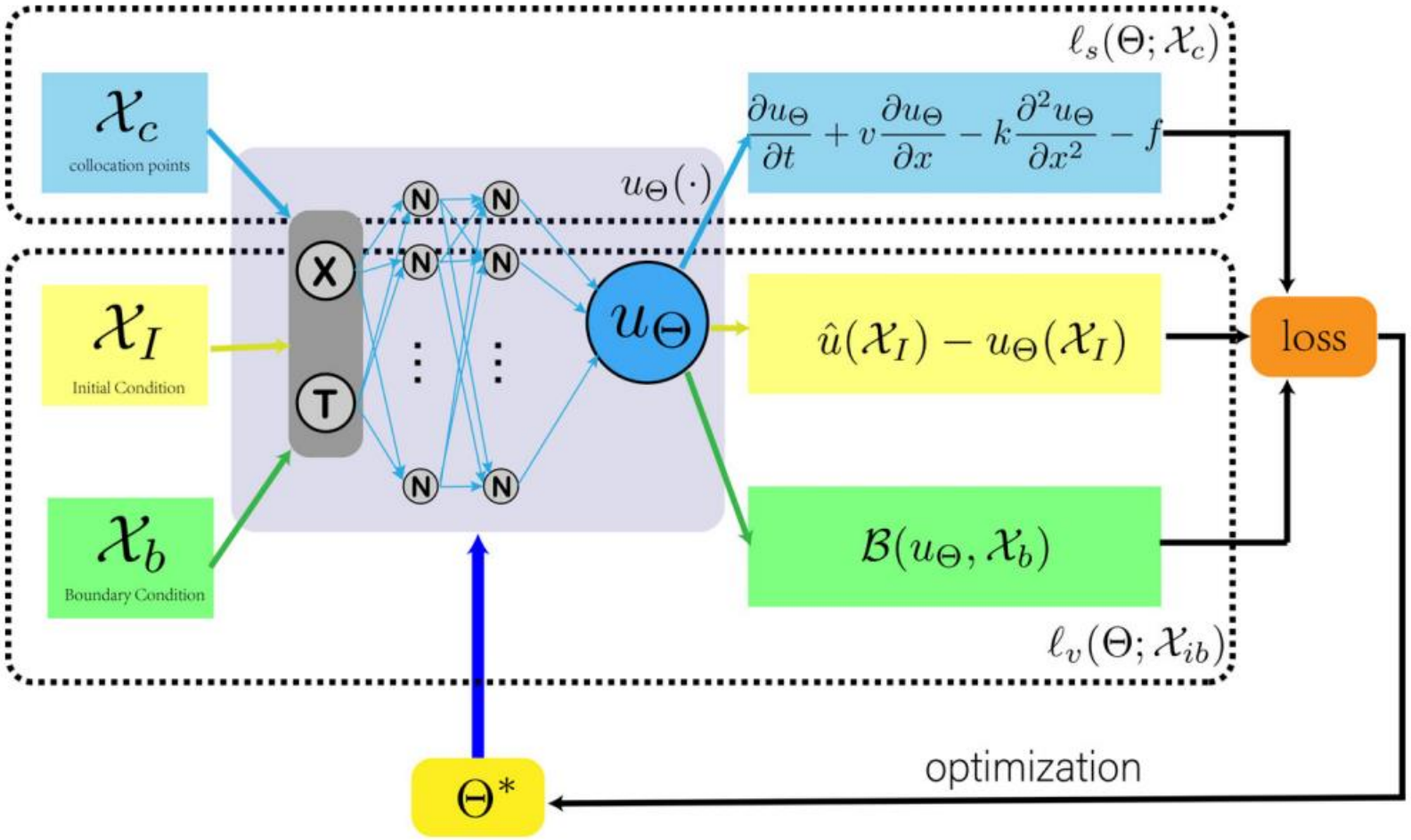
- In the NN, the losses are used to measure the difference between u_θ (.)and u(.)
- The PINN is implemented by imposing two types of loss.
  1. l_s controlled by the collocation points x_c
  2. l_v calculated on the initial-boundary points x_{ib}
- The discrepancy between u_θ (.)and u(.) is measured on the collocation points and boundary points using the loss function defined by the L_2 norm as

$$l(\theta, x) = \omega_s l_s(\theta; x_c) + w_v l_v(\theta; x_{ib}),$$

$$l_s(\theta; x_c) = \frac{1}{|x_c|} \sum_{x \in x_c} || \frac{\partial u_\theta}{\partial t} + v \frac{\partial u_\theta}{\partial x} - k \frac{\partial^2 y}{\partial x^2} - \text{f} ||_2 ,$$

$$l_v(\theta; x_{ib}) = \frac{1}{|x_{ib}|} \sum_{x \epsilon x_{ib}} ||\beta(\mu_\theta, x_{ib})||_2$$

Structure for PINN



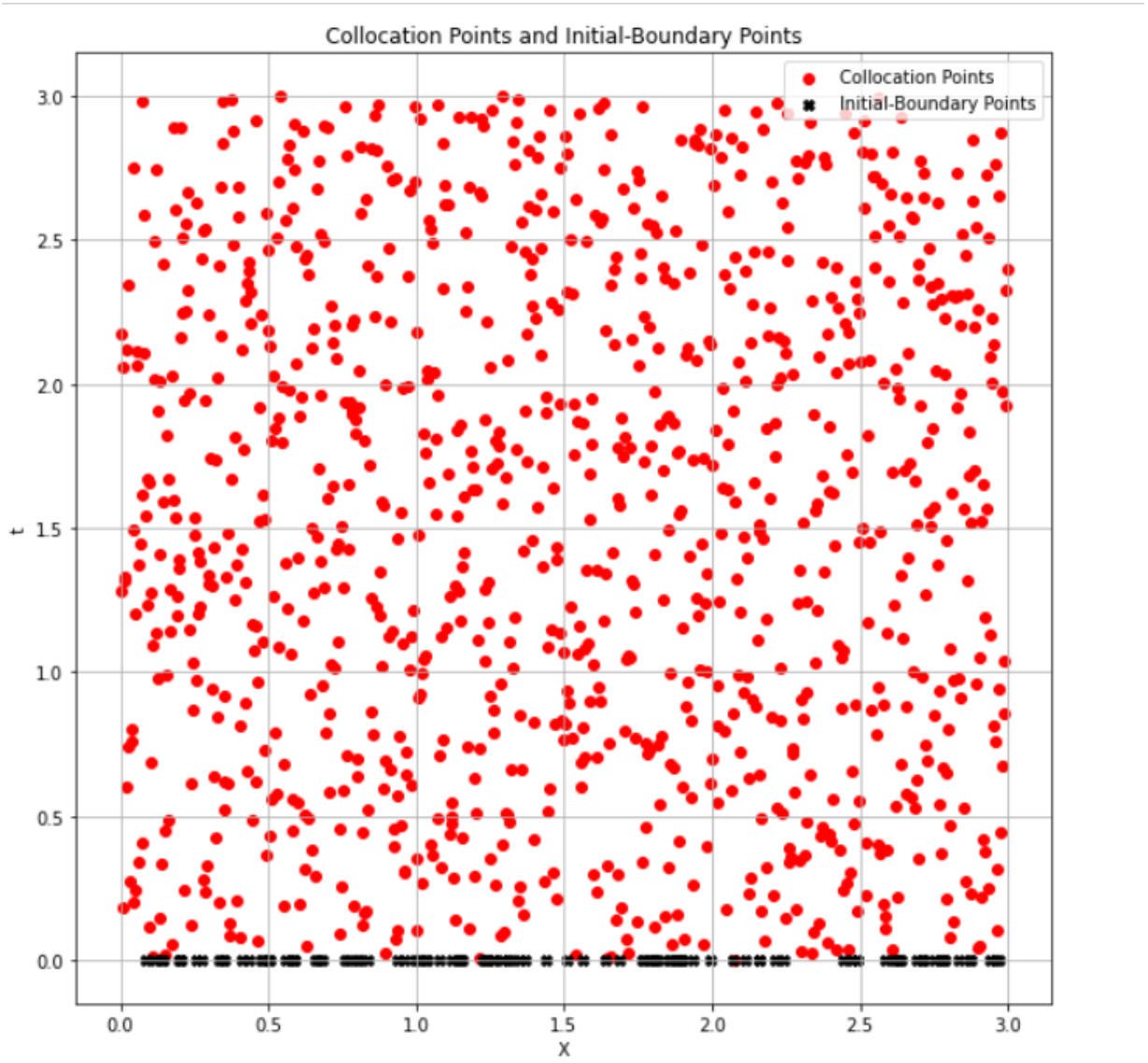
Strategies for training points selection

1 Use the LHS method for collocation points,
random selection method for initial-boundary points.

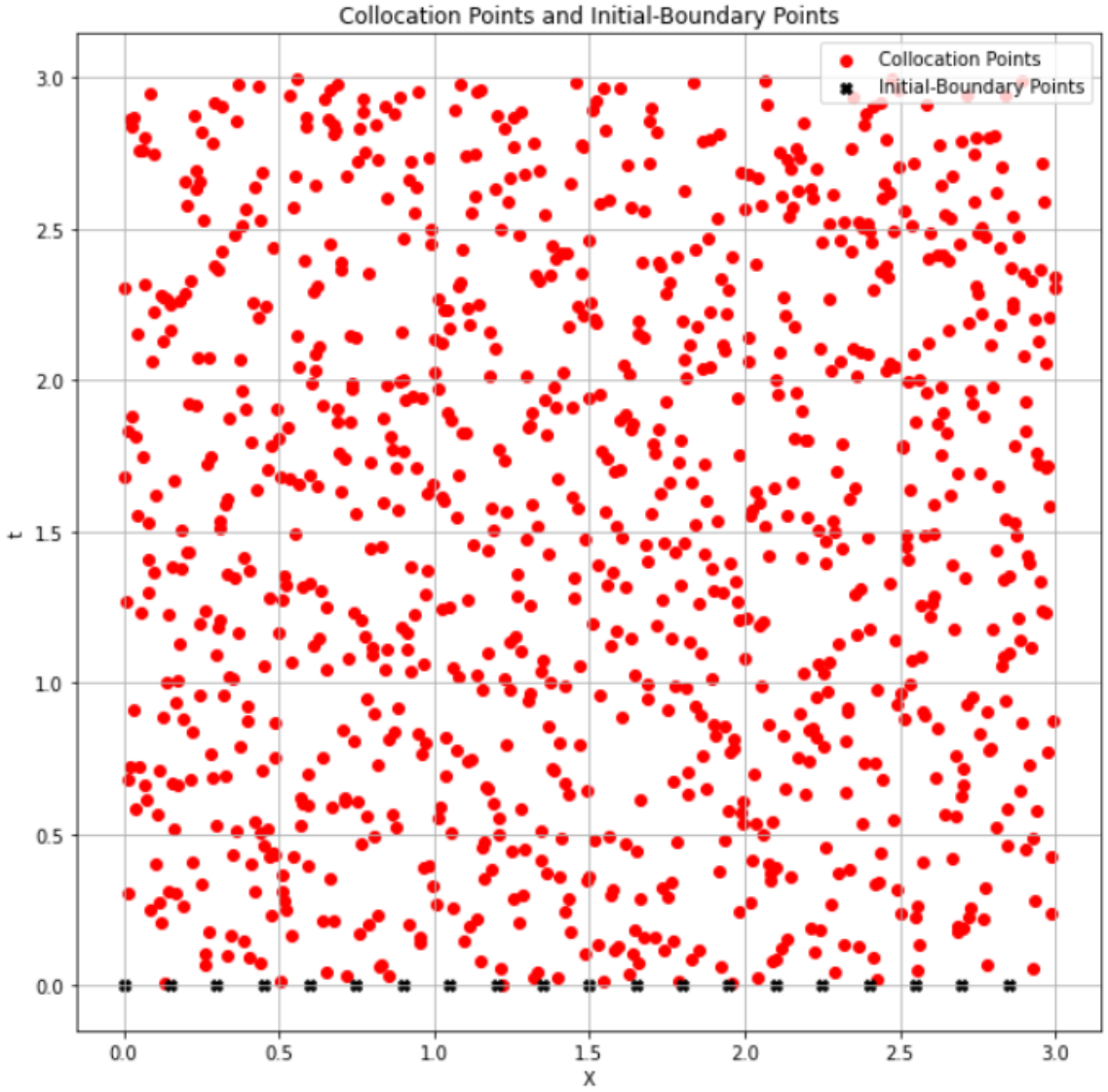
2 Use the LHS method for collocation points,
uniform selection method for initial-boundary points.

3 Use orthogonal grids for collocation points,
uniform selection method for initial-boundary points.

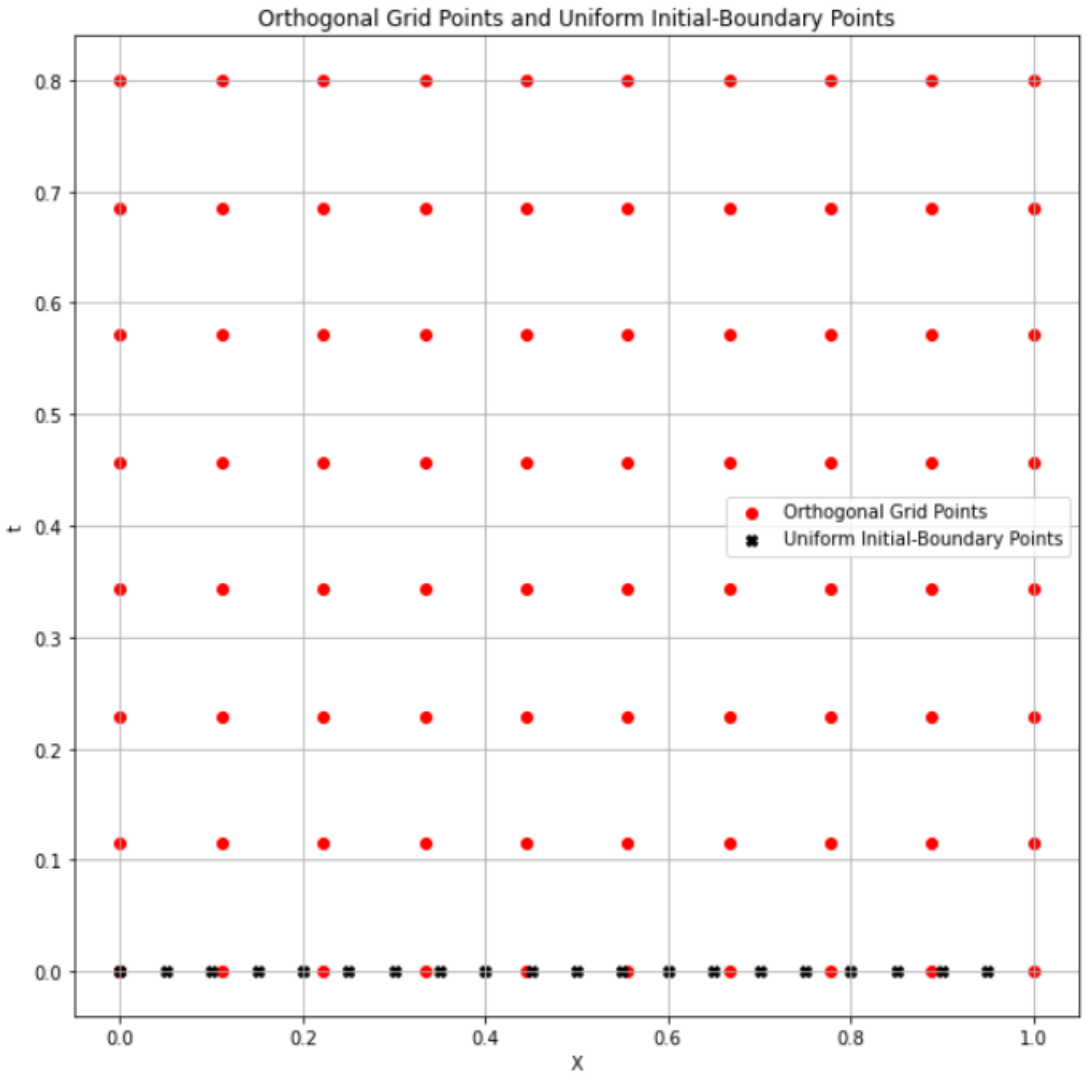
LHS method for collocation points and the random selection method for initial-boundary points.



Use the LHS method for collocation points and uniform selection method for initial-boundary points.



Use orthogonal grids for collocation points and a uniform selection method for initial-boundary points.



First derivative constraint (FDC)

- For ADR equations where only initial and boundary values are given, without any derivative information in the space-time domain, it is beneficial to include a new constraint on the derivative in the loss function.
- Since there are no additional high-order derivatives available, the first derivative of the governing equation can be used as a constraint for the second-order derivative, specifically for the diffusion term. This constraint helps improve the overall solution accuracy within the PINN framework.

By adding the first derivative of the governing equation as an additional constraint to the algorithm, the loss function is

$$l_{fx}(\theta; x_f) = \frac{1}{|x_f|} \sum_{x \in x_f} || \frac{\partial(\frac{\partial u_\theta}{\partial t} + \frac{\partial u_\theta}{\partial x} - k\frac{\partial^2 y}{\partial x^2} - f)}{\partial x} ||2$$

so, finally loss function become:

$$l(\theta, x) = \omega_s l_s(\theta; x_c) + w_{fx} l_{fx}(\theta; x_f) + w_v l_v(\theta; x_{ib})$$

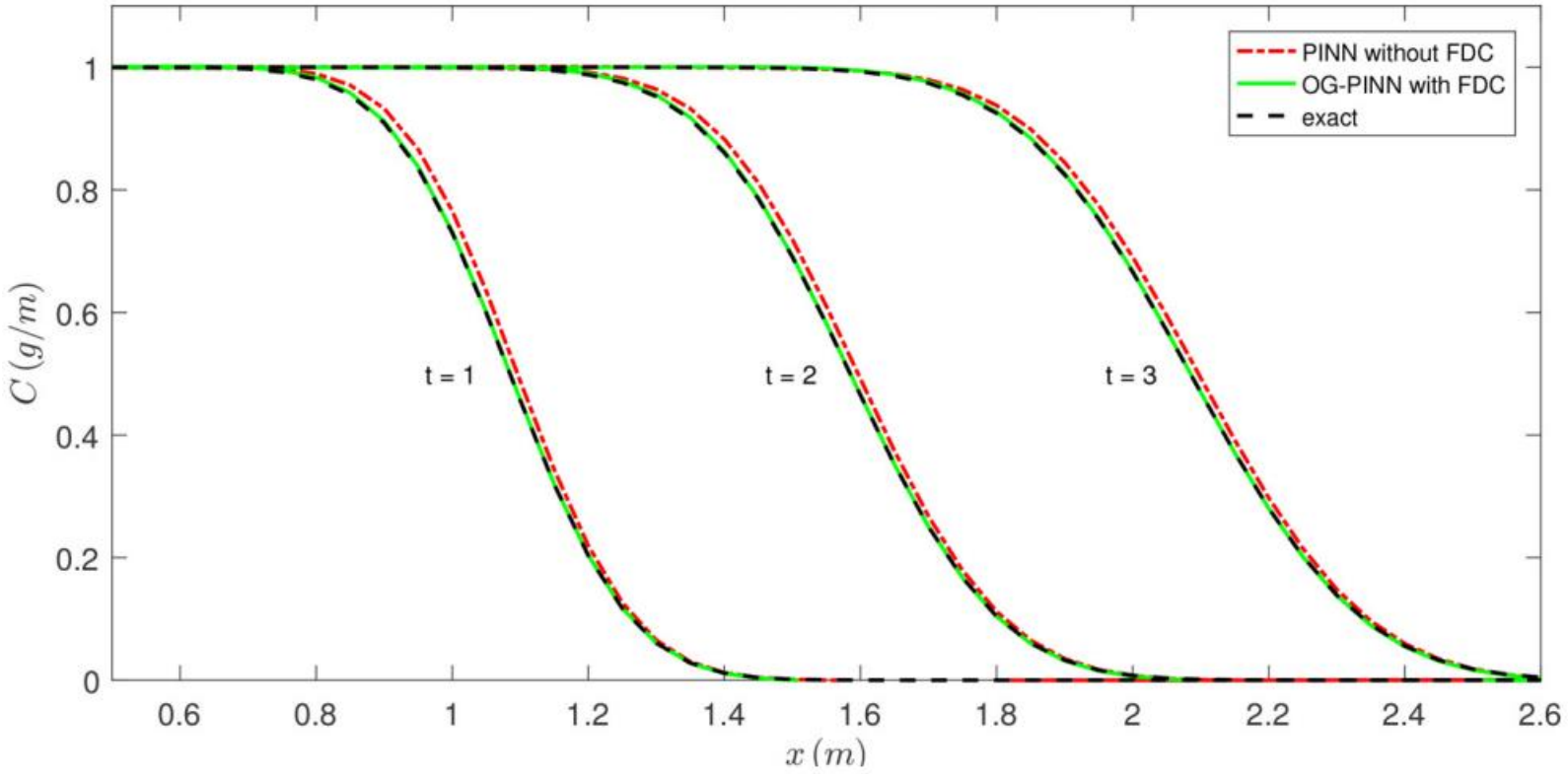
Model Parameter

- In all the tests, the activation function is tanh, the number of hidden layers in the network is five, and the number of neurons in each layer is 50. The weights x_s and x_{fx} are one, and x_v is 0.001, in order to ensure that each part of the loss is in the same order of magnitude. The Adam method is used as the optimization algorithm in the training process with

In [20]:
```python
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Define the parameters
activation_fn = 'tanh'
n_hidden_layers = 5
n_neurons = 50
x_s_weight = 1.0
x_fx_weight = 1.0
x_v_weight = 0.001
learning_rate = 0.001

# Define the model architecture
model = keras.models.Sequential()
model.add(keras.layers.Dense(n_neurons, activation=activation_fn, input_shape=(input_dim,)))
for a in range(n_hidden_layers - 1):
    model.add(keras.layers.Dense(n_neurons, activation=activation_fn))
model.add(keras.layers.Dense(output_dim, activation='linear'))

# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss=custom_loss)
```

Analysis of solution

The results of OG-PINN with FDC are compared with the exact and the original PINN solutions at three times levels of t = 1, 2, and 3 s, respectively. Both the original PINN and the OG PINN with FDC can give correct solutions, but the original PINN shows a more misfit with exact solution compair to PINN with FDC.



NUMERICAL RESULTS AND DISCUSSION

Solving ADR equation by previous model by all possible combination of selection of colocation point & initial boundary point and using and without using FDC

Mean and standard deviation of the error are:

Case	Strategy 1	Strategy 2	Strategy 3
Without FDC	0.041 84 ± 0.036 22	0.033 99 ± 0.020 68	0.041 59 ± 0.015 66
With FDC	0.029 13 ± 0.014 61	0.021 65 ± 0.020 66	0.003 19 ± 0.001 32

Experimental mean and standard deviation of the error

Conclusion

- By Comparing all these ways of solving ADR.

- The strategy of using orthogonal grids instead of LHS for collocation points and uniform instead of random for initial-boundary points is demonstrated as an effective method for point selection in PINN to solve the ADR equation.
- An additional high-order derivative constraint is added to the loss function, which differentiates the governing equation to improve the accuracy of each term in the governing equation.
- The effectiveness of these improvements is verified by numerical experiments. Both the mean and standard deviation of the error in OG-PINN with FDC are lower than those in the original PINN. These improvements have improved the accuracy and stability of the model without increasing the need for additional data during the training process.

References

- 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,” J. Comput. Phys. 378, 686–707 (2019).

- L. A. Khan and P. L. Liu, “An operator splitting algorithm for the three dimensional advection-diffusion equation,” Int. J. Numer. Methods Fluids 28, 461–476 (1998).

- S. Zhang and X. Xin, “Pollutant source identification model for water pollution incidents in small straight rivers based on genetic algorithm,” Appl. Water Sci. 7, 1955–1963 (2017).

- N. Karedla, J. C. Thiele, I. Gregor, and J. Enderlein, “Efficient solver for a special class of convection-diffusion problems,” Phys. Fluids 31, 023606 (2019).

- Y. Li and F. Mei, “Deep learning-based method coupled with small sample learning for solving partial differential equations,” Multimedia Tools Appl. 80, 17391–17413 (2021)