Open Journal of Mathematical Analysis

PSR Press

Article

Identification of parameters in parabolic partial differential equation from final observations using deep learning

Khalid Atifi[1,*], El-Hassan Essoufi[2], and Khadija Rizki[2]

[1] Laboratoire de Mathématiques Appliquées et Informatique (MAI) Université Cadi Ayyad, Marrakech, Morocco
[2] Laboratoire de Mathématiques Informatique et Sciences de l'ingénieur (MISI) Université Hassan 1, Settat 26000, Morocco.
* Correspondence: rahimie@shirazu.ac.ir

Communicated by: Absar Ul Haq
Received: 15 May 2023; Accepted: 20 June 2023; Published: 30 June 2023.

Abstract: In this work, we propose a deep learning approach for identifying parameters (initial condition, a coefficient in the diffusion term and source function) in parabolic partial differential equations (PDEs) from scattered final observations in space and noisy a priori knowledge. In Particular, we approximate the unknown solution and parameters by four deep neural networks trained to satisfy the differential operator, boundary conditions, a priori knowledge and observations. The proposed algorithm is mesh-free, which is key since meshes become infeasible in higher dimensions due to the number of grid points explosion. Instead of forming a mesh, the neural networks are trained on batches of randomly sampled time and space points. This work is devoted to the identification of several parameters of PDEs at the same time. The classical methods require a total a priori knowledge which is not feasible. While they cannot solve this inverse problem given such partial data, the deep learning method allows them to resolve it using minimal a priori knowledge.

Keywords: Deep learning; Heat equation; Hybrid method; Inverse problem; Model-driven solution; Neural networks; Optimization; Tikhonov regularization.

MSC: 35C05

1. Introduction

According to J.B. Keller [1], two problems are said to be inverses of each other if the formulation of one put the other involved. A more operational definition is that an inverse problem is to determine causes knowing effects. Thus, this problem is the inverse of the one called direct problem, consisting in deducing the effects, the causes being known.

In a large number of applications based on partial differential equations, we are interested in the identification of some properties of the system from partial knowledge. We consider the following boundary problem:

$$\begin{cases} \partial_t u - \partial_x(a\partial_x u) = f, \ \ \forall x \in \Omega, \ \forall t \in]0, T[, \\ u|_{\partial\Omega} = g(t), \ \ \forall t \in]0, T[, \\ u(0, x) = u_0(x), \ \ \forall x \in \Omega. \end{cases} \tag{1}$$

Heat equation (diffusion equation) is the archetype of parabolic models. In the unidimensional problem case, (1) models the evolution of the temperature $u(t, x)$ in a conductor bar, the initial temperature distribution at $t = 0$ is given by the function $u_0(x)$, on the edge the temperature is maintained at $g(t)$, the heat sources are modeled by the given function $f(t, x)$, the specific heat $a(x)$ of a material is the amount of heat energy that it takes to raise one unit of mass of the material by one unit of temperature.

There are several inverse problems or identification problems associated with the heat equation. Given temperature u (some measures of $u(t, x)$), each question among the following models an inverse problem:

- What are the boundary conditions which we must supply this system to have the temperature u?

- What is the initial condition to reach the temperature u?
- What is the source we should provide to the system to have u as a solution of (1)?

Sometimes an inverse problem is modeled by a combination of questions above. We should notice that for other values of observations we can get other parameters. On other hand we can search $a(x)$, which we call an identification problem, since it is a characterization which identifies the material. Whatever observations we get the same coefficient value.

The reformulation of an inverse problem as minimization of an error between the real measures and the solution of the direct problem is the most important step. We seek to identify a parameter $P \in E_P$ which can be the initial condition u_0, the source f or the coefficient a, from an observed function u^{obs} (measures). The following equation of state relates implicitly the parameter P and the state u:

$$F(P, u) = 0. \tag{2}$$

Let

$$u = \phi(P), \tag{3}$$

be the solution of the problem (2).
The inverse problem consists in identifying P such as:

$$\phi(P) = u^{obs}. \tag{4}$$

The application ϕ is defined implicitly. It is non-linear, even though the equation of state (2) is linear. This obviously makes it more difficult to solve the inverse problem. We refer the interested reader in inverse problems to see [2,3].

What we said previously suggests that equation (4) may not have a solution, and even if it does, the inverse application is not necessarily continuous. We are therefore going to introduce a formulation, a priori weaker, which has proved its usefulness. We replace equation (4) with the following minimization problem:

$$\begin{cases} \text{Find } \ P \in A_{ad} \ \text{ such as} \\ J_T(P) = \min\limits_{h \in A_{ad}} J_T(h), \end{cases}$$

where the space A_{ad} is the admissible set of P and the cost function J_T is defined as follows:

$$J_T(h) = \frac{1}{2T} \left\| \phi(h) - u^{obs} \right\|^2.$$

This formulation is called a least squares method, and J_T is the cost function. The observation being given once and for all, to evaluate the functional J_T in one parameter P, we start by solving the equation of state (2), then the observation equation 4, and we compare the simulated observation to the measured one. Regarding the observation, measurements are made for a few points of space and at some points in time. So, we need to do an interpolation.

The first section is devoted to the presentation of classical methods of resolution, their limitations and our motivation. In the second section we prove that the minimization problem is well-posed. We will calculate the gradient of J_T in the third section which we will use for classical methods in the fourth one. The fifth section is devoted to the presentation of our approach. We will prove, under certain conditions, the convergence of the L^2 error defined as (24) using the smoothness of the neural network approximations in the sixth section. We will present in the following section some numerical results of our algorithm which we will compare with classical algorithms in the last section.

2. Related work

The mathematical model leads to a non-convex minimization problem. The objective function can have several local minimums, as a result a descent algorithm ends at the meeting of the first local minimum. To solve this problem we can apply two different approaches.

The first approach is Tikhonov regularization which consists in defining the solution of the inverse problem as being the minimum of one least squares criterion penalized by a quadratic term:

$$J_T(h) = \frac{1}{2T}\left\|\phi(h) - u^{obs}\right\|^2 + \frac{\varepsilon}{2}\left\|h - h^b\right\|^2,$$

where ε is the regularization coefficient and h^b is a priori knowledge (background) of exact parameter h.

An essential difficulty with this approach stems from the fact that it requires a priori knowledge at all points of the domain of resolution. Moreover, in practice, the knowledge h^b is partial (30%), the size of h^b is much less than that of h. This makes it difficult to find the regularization coefficient.

To solve this problem a second approach called hybrid method is proposed in [4,5] . The idea of this method is to marry the genetic algorithm with a gradient descent method. The individual in the genetic algorithm is the parameter h. Let's start with an initial population, we apply the genetic algorithm with its operators (crossing and mutation) to produce a new population of which each individual serves as a starting point for the steep descent method. This method is very effective for inverse problems associated with identifying a single parameter of PDEs for example a or u_0 or f. For the identification of several parameters of the equation this method does not solve the problem of partial knowledge.

However, there are some drawbacks associated with those methods. The gradient of functional J_T is computed by using adjoint method, which requires the resolution of another boundary problem whose parameters are dependent on u (the solution of the main problem). The first drawback is that they require, in each descent iteration, resolution of two PDEs by classical methods (finite difference, finite element, .etc). It means that to solve the inverse problem we should solve direct problem several times and manage its complexity which makes those methods costly.

The second drawback is that they are not meshfree. Their application for high dimensional problems and complex domains is difficult since classical methods become infeasible in higher dimensions due to the explosion in the number of grid points. They also require using interpolation and numerical integrals.

The last drawback is that they require a complete a priori knowledge. The solution reconstruction cannot be made from partial data.

Recently, deep learning emerged as a powerful technique in many applications. It is used in many works related to PDEs like [6–8] where authors solve many types of PDEs by deep learning method. This success inspires us to apply such a technique to identifying parameters of PDEs from final observations.

This work is devoted to the identification of a, u_0 and f of (1) from final observations by a new approach based on deep learning method which is meshfree and does not require solving direct problem, interpolation or numeric integrals. This new approach requires only partial a priori knowledge.

The sought after solution is driven by the model since the relation between all parameters of the system (diffusion phenomenon model) is the main information for the resolution.

3. Well-posedness

In this section we consider the case of homogeneous boundary data, i.e.,$g(t) = 0$.

We specify some notations we shall use.

$$S_a = S_{u_0} = L^2(\Omega), \quad S_f = L^2\left(]0,T[;L^2(\Omega)\right),$$

$$S = S_a \times S_{u_0} \times S_f.$$

Let $P = (a, u_0, f) \in S$

$$\|P\|_S = \|a\|_{S_a} + \|u_0\|_{S_{u_0}} + \|f\|_{S_f}.$$

Definition 1. *(Variational formulation [9])*
The variational formulation of the problem (1) is the following:
Find $u(t)$ function of $]0,T[$ in $H_0^1(\Omega)$ such as:

$$\begin{cases} \frac{d}{dt}\langle u(t), v\rangle_{L^2(\Omega)} + \langle a u_x(t), v_x\rangle_{L^2(\Omega)} = \langle f(t), v\rangle_{L^2(\Omega)}, \ \forall v \in H_0^1(\Omega), \ \forall t \in]0,T[, \\ u(x,0) = u_0(x), \ \forall x \in \Omega. \end{cases} \tag{5}$$

Definition 2. *(Weak solution)*
A weak solution of (1) is a solution of the variational formulation 5.

Let $a \in L^\infty(\Omega)$ where $a(x) > \gamma$, $\gamma > 0$, $\forall\, x \in \overline{\Omega}$.
The existence and uniqueness of the solution are given by the following theorem:

Theorem 1. *[10] Let $u_0 \in L^2(\Omega)$ and $f \in L^2(]0,T[\,; L^2(\Omega))$, there is only one weak solution of the problem (1) such as:*

$$u \in L^2(]0,T[\,; H_0^1(\Omega)) \cap C([0,T]\,; L^2(\Omega)),$$

with the following estimation:

$$\|u\|_{L^2(]0,T[;H_0^1(\Omega))} + \|u\|_{C([0,T];L^2(\Omega))} \leq C\left(\|f\|_{L^2(]0,T[;L^2(\Omega))} + \|u_0\|_{L^2(\Omega)}\right), \tag{6}$$

the constant C depends only on Ω and T. □

Theorem 2. *Let u a weak solution of (1) with $P = (a, u_0, f)$.*
The function

$$\begin{aligned} \phi \ : S &\longrightarrow L^2(]0,T[\,; H_0^1(\Omega)) \cap C([0,T]\,; L^2(\Omega)) \\ P &\longmapsto u \end{aligned}$$

is continuous, and the functional J_T is continuous. Therefore, J_T has a unique minimum in A_{ad}.□

Proof. Let $\delta P = (\delta a, \delta u_0, \delta f) \in S$ be a small variation such that $P + \delta P \in A_{ad}$. Consider $\delta u = u^\delta - u$, where u is the weak solution of (1) with P, and u^δ is the weak solution of (1) with $P^\delta = P + \delta P$. We have

$$\begin{cases} \partial_t u - \partial_x(a\partial_x u) = f(x,t), \ \forall x \in \Omega, \ \ \forall t \in]0,T[, \\ u(0,t) = u(1,t) = 0, \ \ \forall t \in]0,T[, \\ u(x,0) = u_0(x), \ \ \forall x \in \Omega, \end{cases} \tag{7}$$

and

$$\begin{cases} \partial_t u^\delta - \partial_x\left(a^\delta u_x^\delta\right) = f^\delta(x,t), \ \forall x \in \Omega, \ \ \forall t \in]0,T[, \\ u^\delta(0,t) = u^\delta(1,t) = 0, \ \ \forall t \in]0,T[, \\ u^\delta(x,0) = u_0^\delta(x), \ \ \forall x \in \Omega. \end{cases} \tag{8}$$

(8)−(7) yield

$$\begin{cases} \partial_t \delta u - \partial_x\left(a^\delta \partial_x \delta u\right) - \partial_x\left(\delta a \partial_x u\right) = \delta f, \ \ \forall x \in \Omega, \ \ \forall t \in]0,T[, \\ \delta u(0,t) = \delta u(1,t) = 0, \ \ \forall t \in]0,T[, \\ \delta u(x,0) = \delta u_0(x), \ \ \forall x \in \Omega. \end{cases} \tag{9}$$

The variational formulation of (9) is:

$$\begin{cases} \displaystyle\int_\Omega \partial_t \delta u v \, dx + \int_\Omega \left(a^\delta \partial_x \delta u\right) \partial_x v \, dx - \int_\Omega \partial_x\left(\delta a \partial_x u\right) v \, dx = \int_\Omega \delta f v \, dx, \ \ \forall v \in H_0^1(\Omega), \\ \delta u(x,0) = \delta u_0(x), \ \ \forall x \in \Omega. \end{cases}$$

Take $v = \delta u$, then

$$\int_\Omega \partial_t(\delta u)\delta u \, dx + \int_\Omega a^\delta \left(\partial_x \delta u\right)^2 \, dx - \int_\Omega \partial_x\left(\delta a \partial_x u\right) \delta u \, dx = \int_\Omega \delta f \delta u \, dx.$$

We have

$$\int_\Omega a^\delta \left(\partial_x \delta u\right)^2 \, dx \geq 0,$$

this implies that

$$\int_\Omega \partial_t(\delta u)\delta u \, dx \leqslant \int_\Omega \partial_x\left(\delta a \partial_x u\right) \delta u \, dx + \int_\Omega \delta f \delta u \, dx,$$

and consequently

$$\int_\Omega \partial_t(\delta u)\delta u\, dx \leqslant \int_\Omega \partial_x\left(\delta a \partial_x u\right)\delta u\, dx + \int_\Omega \delta f \delta u\, dx,$$

then

$$\int_\Omega \partial_t(\delta u)\delta u\, dx \leqslant \| \delta a \|_{L^\infty(\Omega)} \int_\Omega |\partial_x u \partial_x \delta u|\, dx + \int_\Omega |\delta f \delta u|\, dx.$$

By integrating between 0 and t with $t \in [0,T]$ we obtain

$$\frac{1}{2}\| \delta u(t) \|^2_{L^2(\Omega)} - \frac{1}{2}\| \delta u_0 \|^2_{L^2(\Omega)} \leqslant \| \delta a \|_{L^\infty(\Omega)} \int_0^T\int_\Omega |\partial_x u \partial_x \delta u|\, dxdt + \int_0^T\int_\Omega |\delta f \delta u|\, dx,$$

since $u,\ \delta u \in H^1(0,T;L^2(\Omega))$,
there exist $C_1 > 0$ and $C_2 > 0$, such that

$$\sup_{t\in[0,T]} \| \delta u(t) \|^2_{L^2(\Omega)} \leqslant 2C_1 \| \delta a \|_{L^\infty(\Omega)} + \| \delta u_0 \|^2_{L^2(\Omega)} + C_2 \|\delta f\|_{L^2(]0,T[;L^2(\Omega))},$$

which give

$$\| \delta u \|^2_{C([0,T];L^2(\Omega))} \leqslant C\left(\| \delta a \|_{L^2(\Omega)} + \| \delta u_0 \|^2_{L^2(\Omega)} + \|\delta f\|_{L^2(]0,T[;L^2(\Omega))}\right)$$

where $C = max(2C_1, 1, C_2)$.
We have:

$$H^1(\Omega) \underset{compact}{\hookrightarrow} L^2(\Omega), \tag{10}$$

$$H^1_0(\Omega) \underset{compact}{\hookrightarrow} L^2(\Omega). \tag{11}$$

Let

$$F = \left\{ g \in L^2\left([0,T]\,; H^1(\Omega)\right) \,/\, \partial_t g \in L^2\left([0,T]; L^2(\Omega)\right)\right\},$$

By Aubin-Lions Lemma [11], :

$$F \underset{compact}{\hookrightarrow} L^2\left([0,T]; L^2(\Omega)\right). \tag{12}$$

Let

$$A^{r_1}_{u_0} = \left\{ u_0 \in H^1_0(\Omega);\ \|u_0\|_{H^1_0(\Omega)} \leq r_1 \right\},$$

$$A^{r_2}_{a} = \left\{ a \in H^1(\Omega);\ \|a\|_{H^1(\Omega)} \leq r_2 \right\},$$

$$A^{r_3}_{f} = \left\{ f \in F;\ \|f\|_{L^2([0,T];H^1(\Omega))} \leq r_3 \right\},$$

where $r_1, r_2, r_3 \in \mathbb{R}^{*+}$. Let

$$A_{ad} = A^r_a \times A^r_{u_0} \times A^r_f,$$

where $r = min(r_1, r_2, r_3)$.
A^r_a, $A^r_{u_0}$ and A^r_f are bounded in $H^1(\Omega)$, $H^1_0(\Omega)$ and F respectively. From, (11) and (12), they are compacts in $L^2(\Omega)$, $L^2(\Omega)$ and $L^2\left([0,T]\,; L^2(\Omega)\right)$ respectively. Consequently A_{ad} is compact in S.

The functional J_T is continuous. Therefore, J has a unique minimum in A_{ad}. □

4. Gradient of J_T by adjoint state method

We define the Gâteaux derivative of ϕ at $P = (a, u_0, f)$ in the direction $q = (q_a, q_{u_0}, q_f) \in S$, by

$$\hat{u} = \lim_{\varepsilon\to 0} \frac{\phi(P+\varepsilon q) - \phi(P)}{\varepsilon},$$

$\phi(P+\varepsilon q)$ is the weak solution of (1) with parameters $P+\varepsilon q$, and $\phi(P)$ is the weak solution of (1) with parameters P.

The Gâteaux (directional) derivative of ϕ at P in the direction $q \in S$, is the weak solution of the following tangent linear model:

$$\begin{cases} \partial_t \hat{u} - \partial_x(a\partial_x \hat{u}) - \partial_x(q_a \partial_x u) = q_f(x,t),\ \forall x \in \Omega,\ \ \forall t \in]0,T[, \\ \hat{u}(t,0) = \hat{u}(t,1) = 0,\ \ \forall t \in]0,T[, \\ \hat{u}(0,x) = q_{u_0}(x),\ \ \forall x \in \Omega. \end{cases}$$

We introduce the adjoint variable A, and we integrate:

$$\int_0^T \int_0^1 \partial_t \hat{u} A - \partial_x \left(a\partial_x \hat{u}\right) A - \partial_x \left(q_a \partial_x u\right) A \, dxdt = \int_0^T \int_0^1 q_f A \, dxdt.$$

Calculate separately each term :

$$\int_0^T \int_0^1 \partial_t \hat{u} A \, dxdt = \int_0^1 \left[\hat{u}A\right]_0^T \, dx - \int_0^1 \int_0^T \partial_t A \hat{u} \, dtdx,$$

$$\begin{aligned} \int_0^T \int_0^1 \partial_x \left(a\partial_x \hat{u}\right) A \, dxdt &= \int_0^T \left[\left(a\partial_x \hat{u}\right) A\right]_0^1 \, dt - \int_0^T \int_0^1 a\partial_x \hat{u} \partial_x A \, dxdt \\ &= \int_0^T \left[\left(a\partial_x \hat{u}\right) A\right]_0^1 \, dt - \int_0^T \left[a\hat{u}\partial_x A\right]_0^1 \, dt + \int_0^T \int_0^1 \partial_x \left(a\partial_x A\right) \hat{u} \, dxdt \\ &= \int_0^T \left[\left(a\partial_x \hat{u}\right) A\right]_0^1 \, dt + \int_0^T \int_0^1 \partial_x \left(a\partial_x A\right) \hat{u} \, dxdt, \end{aligned}$$

$$\int_0^T \int_0^1 A \left(\partial_x \left(q_a \partial_x u\right)\right) \, dxdt = \int_0^T \left[q_a \partial_x u A\right]_0^1 \, dt - \int_0^T \int_0^1 q_a \partial_x u \partial_x A \, dxdt.$$

We pose $A(t,0) = A(t,1) = 0$ and $A(T,x) = 0$, we obtain

$$\int_0^T \left\langle \hat{u}, \partial_t A + \partial_x(a\partial_x A)\right\rangle_{L^2(\Omega)} dt = \left\langle q_a, \int_0^T \partial_x u \partial_x A \, dt \right\rangle_{L^2(\Omega)} - \left\langle q_{u_0}, A(0,x)\right\rangle_{L^2(\Omega)} - \left\langle q_f, A \right\rangle_{L^2(]0,T[\times\Omega)},$$

so

$$\int_0^T \left\langle \partial_t A + \partial_x(a\partial_x A, \hat{u})\right\rangle_{L^2(\Omega)} dt = -\left\langle q, h\right\rangle_S, \tag{13}$$

where

$$h = \left(-\int_0^T \partial_x u \partial_x A \, dt, A(0,x), A\right).$$

Let

$$J_T(P) = \frac{1}{2T}\int_0^T \left\| u(t) - u^{obs}(t)\right\|^2_{L^2(\Omega)} dt.$$

The Gâteaux derivative of J at P in the direction q is given by:

$$\hat{J}_P(q) = \lim_{\varepsilon \to 0} \frac{J_T(P+\varepsilon q) - J_T(P)}{\varepsilon},$$

After some computations, we arrive at

$$\hat{J}_P(q) = \frac{1}{T}\int_0^T \left\langle u - u^{obs}, \hat{u}\right\rangle_{L^2(\Omega)} dt. \tag{14}$$

The adjoint model is:

$$\begin{cases} \partial_t A + \partial_x \left(a\partial_x A\right) = \frac{1}{T}\left(u - u^{obs}\right), \\ A(t,0) = A(t,1) = 0,\ \ \forall t \in]0,T[, \\ A(T,x) = 0,\ \ \forall x \in \Omega, \end{cases} \tag{15}$$

the problem is retrograde, we make the change of variable $t \longleftrightarrow T - t$, then

$$\begin{cases} \partial_t A - \partial_x \left(a \partial_x A\right) = \frac{1}{T}\left(\bar{u}^{obs} - \bar{u}\right), \\ A(t,0) = A(t,1) = 0, \ \ \forall t \in]0,T[, \\ A(0,x) = 0, \ \ \forall x \in \Omega, \end{cases} \tag{16}$$

with $\bar{u}(t) = u(T-t)$.
From equations (13), (14) et (15), the gradient of J_T is given by

$$\nabla_P J_T = \left(\frac{\partial J_T}{\partial a}, \frac{\partial J_T}{\partial u_0}, \frac{\partial J_T}{\partial f}\right) = -h. \tag{17}$$

By the change of variable we have:

$$\frac{\partial J_T}{\partial a} = \int_0^T \partial_x u \partial_x A \, dt \ ; \quad \frac{\partial J_T}{\partial u_0} = -A(T,x) \ ; \quad \frac{\partial J_T}{\partial f} = -A \ ;$$

5. Approach, Model and Algorithm

Consider a parabolic PDE with d spatial dimensions:

$$\begin{cases} \partial_t u - div_x \left(a(x) I_d \nabla_x u\right) = f, \ \ \forall x \in \Omega, \ \forall t \in]0,T[, \\ u|_{\partial\Omega} = g(t), \ \ \forall t \in]0,T[, \\ u(0,x) = u_0(x), \ \ \forall x \in \Omega, \end{cases} \tag{18}$$

where $\Omega \subset \mathbb{R}^d$, I_d is the identity matrix of order d.

The goal is to find $P = (a, u_0, f) \in S_P$ such that $U^{ob} \in \mathbb{R}^k$ are k final observations of u solution of (18) at K, with

$$S_P = L^2(\Omega) \times L^2(\Omega) \times L^2\left(]0,T[\times \Omega\right),$$
$$K = \{(t_i, x_i) \in \{T\} \times \Omega\}_{0 \leqslant i \leqslant k}$$

and $P^b = (a^b, u_0^b, f^b) \in \mathbb{R}^h$ is h a priori knowledge of P at H^b with some bruit b where $h = (h_1, h_2, h_3) \in \mathbb{N}^3$:

$$H^b = \{x_i \in \Omega\}_{0 \leqslant i \leqslant h_1} \times \{x_i \in \Omega\}_{0 \leqslant i \leqslant h_2} \times \{(t_i, x_i) \in]0,T[\times \Omega\}_{0 \leqslant i \leqslant h_3}$$

The optimization problem associated is finding P which minimizes the function:

$$\mathcal{J}(v) = \left\| u(t=T) - u^{ob} \right\|^2_{L^2(\Omega)} + \left\| v - p^b \right\|^2_{S_P}, \tag{19}$$

where u^{ob} and p^b are interpolations of U^{ob} in $L^2(\Omega)$ and P^b in S_P.

The deep learning method for inverse problem (DL-IP) approach approximates a, u_0, f and u by four deep neural networks with common inputs and loss function. Figure (1) shows a visualization of the overall architecture:

$$a(x) \simeq \tilde{a}(x) = N_a(x; \theta_a),$$
$$u_0(x) \simeq \tilde{u_0}(x) = N_{u_0}(x; \theta_{u_0}),$$
$$f(x,t) \simeq \tilde{f}(x,t) = N_f(x,t; \theta_f),$$
$$u(x,t) \simeq \tilde{u}(x,t) = N_u(x,t; \theta_u),$$

where $\theta = (\theta_a, \theta_{u_0}, \theta_f, \theta_u)$ are parameters of neural networks.

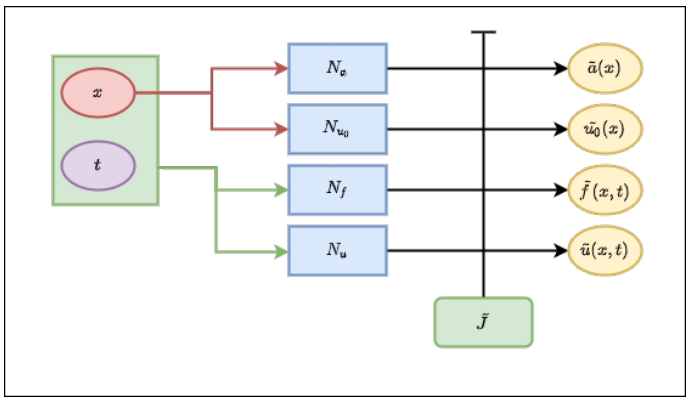

Figure 1. DL-IP's model architecture.

Our goal now is to find θ such that U^{ob} are observations of N_u, and P^b is the a priori knowledge of (N_a, N_{u_0}, N_f).

The cost function is constructed as follows:

$$\tilde{\mathcal{J}}(\theta) = \left\|\partial_t \tilde{u} + \mathcal{L}(\tilde{a}, \tilde{u}) - \tilde{f}\right\|^2_{L^2(]0,T[\times\Omega)} + \left\|\tilde{u}^{ob} - U^{ob}\right\|^2_{\mathbb{R}^k} + \left\|\tilde{P}^b - P^b\right\|^2_{\mathbb{R}^h} + \left\|\tilde{u}|_{\partial\Omega} - g\right\|^2_{L^2(]0,T[\times\partial\Omega)} + \left\|\tilde{u}(t=0) - \tilde{u_0}\right\|^2_{L^2(\Omega)}, \tag{20}$$

where $\mathcal{L}(a,u) = -div_x\left(a(x) I_d \nabla_x u\right)$, $\tilde{u}^{ob}$ are values of $\tilde{u}$ at K, $\tilde{P}^b$ are values of $(\tilde{a}, \tilde{u_0}, \tilde{f})$ at H^b.

The loss functional $\tilde{\mathcal{J}}$ consists of five parts:

1. A measure of how well $\tilde{u}$ and $\tilde{P}$ satisfy the differential operator:

$$\tilde{\mathcal{J}}_1(\theta) = \left\|\partial_t \tilde{u} + \mathcal{L}(\tilde{a}, \tilde{u}) - \tilde{f}\right\|^2_{L^2(]0,T[\times\Omega)}$$

2. A measure of how well the approximation $\tilde{u}$ satisfies the observation value:

$$\tilde{\mathcal{J}}_2(\theta) = \left\|\tilde{u}^{ob} - U^{ob}\right\|^2_{\mathbb{R}^k}$$

3. A measure of how well $\tilde{P}$ satisfies the a priori knowledge:

$$\tilde{\mathcal{J}}_3(\theta) = \left\|\tilde{P} - P^b\right\|^2_{\mathbb{R}^h}$$

4. A measure of how well the approximation $\tilde{u}$ satisfies the boundary condition:

$$\tilde{\mathcal{J}}_4(\theta) = \left\|\tilde{u}|_{\partial\Omega} - g\right\|^2_{L^2(]0,T[\times\partial\Omega)}$$

5. A measure of how well the approximation $\tilde{u}$ satisfies the approximation of initial condition $\tilde{u_0}$:

$$\tilde{\mathcal{J}}_5(\theta) = \left\|\tilde{u}(t=0) - \tilde{u_0}\right\|^2_{L^2(\Omega)}$$

The derivatives of $\tilde{u}$ can be evaluated using automatic differentiation (see [12]) since it is parameterizing as a neural network.

The architecture of a neural network can be crucial to its success. Frequently, different applications require different architectures. For example, convolution networks are essential for image recognition while long short-term networks (LSTMs) are useful for modeling sequential data. In DL-IP's model presented by Figure (1), we use four neural networks with the same architecture described in figure (2).

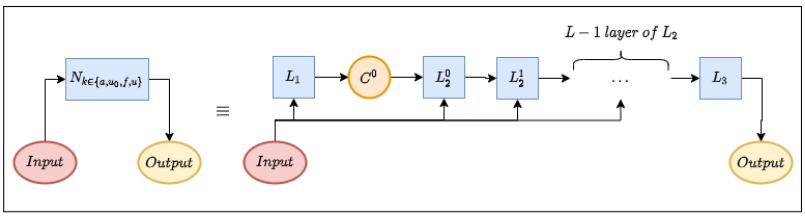

Figure 2. The structure of networks used in figure (1).

Each neural network $N_{k\in\{a,u_0,f,u\}}$ uses three types of layers L_1, L_2 and L_3.

- L_1 takes as input $I = (x,t)$ or $I = x$ and returns as output C^0:

$$C^0 = \sigma(\theta^0.I + \beta^0)$$

- L_2^{n-1} takes as inputs $I = (x,t)$ or $I = x$ and the output C^{n-1} of the previous layer L_1 if $n = 1$ or L_2^{n-2} if $n > 1$ and returns as output C^n. The inputs are transformed through a series of operations. Below, we present the architecture in the equations along with a visual representation of a single L_2^{n-1} layer in Figure (3):

$$\begin{aligned} Z_1^{n-1} &= \sigma(\theta_1^n.I + \gamma_1^n.C^{n-1} + \beta_1^n), \\ Z_2^{n-1} &= \sigma(\theta_2^n.I + \gamma_2^n.C^{n-1} + \beta_2^n), \\ Z_3^{n-1} &= \sigma(\theta_3^n.I + \gamma_3^n.C^{n-1} + \beta_3^n) \\ Z_4^{n-1} &= \theta_4^n.I + \gamma_4^n.(C^{n-1} \odot Z_1^{n-1}) + \beta_4^n), \\ C^n &= (1 - Z_2^{n-1}) \odot Z_4^{n-1} + Z_3^{n-1} \odot C^{n-1}, \end{aligned} \tag{21}$$

where $1 \leqslant n \leqslant L$, $\odot$ denotes Hadamard multiplication, $\theta^n = \{\theta_i^n, \gamma_i^n, \beta_i^n\}_{1\leqslant i\leqslant 4}$ are parameters of L_2^{n-1} layer and L is the total number of L_2 layers.

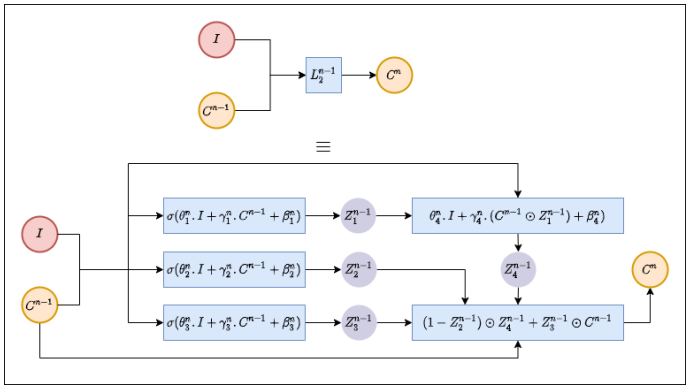

Figure 3. Operations within layerL_2^{n-1}.

- L_3 takes as input the output C^L of the last layer L_2^L and returns:

$$C^{L+1} = \theta^{L+1}.I + \beta^{L+1}.$$

σ is an activation function such as the *tanh* function or sigmoidal function, θ_k are parameters of the N_k neural network and θ are parameters of the DL-IP's model, where

$$\theta_k = \left\{\theta^0, \beta^0, \ldots, \theta^n, \ldots, \theta^{L+1}, \beta^{L+1}\right\}_{1\leqslant n\leqslant L}$$

$$\theta = \{\theta_k\}_{k\in\{a,u_0,f,u\}}$$

Algorithm 3. DL-IP Algorithm

1. *Define boundary conditions*

2. *Define the architecture of neutral networks by setting the number of layers, number of neurons in each layer and activation functions.*
3. *Generate random training set:*

$$D_M = \left\{ (t_i, x_i)_{0 \leqslant i \leqslant \tilde{M}} \in [0,T] \times \Omega,\ (\overline{t_i}, \overline{x_i})_{0 \leqslant i \leqslant \overline{M}} \in [0,T] \times \partial\Omega,\ (x_i^0)_{0 \leqslant i \leqslant M^0} \in \Omega \right\},$$

where $M = (\tilde{M}, \overline{M}, M^0) \in \mathbb{N}^{*3}$.
4. *Initialize the parameter set* θ_0 *and the learning rate* α_0.
5. *Repeat until convergence criterion is satisfied.*

1 Randomly sample a mini-batch d_m *of training examples from* D_M :

$$d_m = \left\{ (t_i, x_i)_{0 \leqslant i \leqslant \tilde{m} \leqslant \tilde{M}},\ (\overline{t_i}, \overline{x_i})_{0 \leqslant i \leqslant \overline{m} \leqslant \overline{M}},\ (x_i^0)_{0 \leqslant i \leqslant m^0 \leqslant M^0} \right\} \subset D_M.$$

2 Compute the loss functional for the sampled mini-batch d_m*:*

$$\tilde{J}_1^m(\theta_n) = \frac{1}{m} \sum_0^m \left(\partial_t \tilde{u}(t_i, x_i) + \mathcal{L}(\tilde{a}(t_i, x_i), \tilde{u}(t_i, x_i)) - \tilde{f}(t_i, x_i) \right)^2,$$

$$\tilde{J}_2^k(\theta_n) = \frac{1}{k} \sum_{(t_i, x_i) \in K} \left(\tilde{u}(t_i, x_i) - U_i^{ob} \right)^2,$$

$$\tilde{J}_3^h(\theta_n) = \frac{1}{h} \sum_{(t_i, x_i) \in B} \left(\left(\tilde{a}(x_i) - a_i^b \right)^2 + \left(\tilde{f}(t_i, x_i) - f_i^b \right)^2 + \left(\tilde{u_0}(t_i, x_i) - u_{0_i}^b \right)^2 \right),$$

$$\tilde{J}_4^{\overline{m}}(\theta_n) = \frac{1}{\overline{m}} \sum_{i=0}^{\overline{m}} \left(\tilde{u}(\overline{t_i}, \overline{x_i}) - g(\overline{x_i}) \right)^2,$$

$$\tilde{J}_5^{m^0}(\theta_n) = \frac{1}{m^0} \sum_{i=0}^{m^0} \left(\tilde{u}(0, x_i^0) - \tilde{u_0}(x_i^0) \right)^2,$$

$$\tilde{J}(\theta_n, d_m) = \tilde{J}_1^m(\theta_n) + \tilde{J}_2^k(\theta_n) + \tilde{J}_3^h(\theta_n) + \tilde{J}_4^{m^0}(\theta_n) + \tilde{J}_5^m(\theta_n). \tag{22}$$

3 Compute the gradient $\nabla_{\theta_n} \tilde{J}(\theta_n, d_m)$ *for the sampled mini-batch* d_m *using backpropagation.*
4 Use the estimated gradient to take a descent step at d_m *with learning rates (23) to update* θ_{n+1}*:*

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_{\theta_n} \tilde{J}(\theta_n, d_m)$$

$$\alpha_n = \begin{cases} 10^{-4}, & n \leqslant 5000, \\ 5 \times 10^{-4}, & 5 < n \leqslant 10000, \\ 10^{-5}, & 10 < n \leqslant 20000, \\ 5 \times 10^{-5}, & 20 < n \leqslant 30000, \\ 10^{-6}, & 5 < 30 \leqslant 40000, \\ 5 \times 10^{-6}, & 40 < n \leqslant 50000, \\ 10^{-7}, & 50000 < n. \end{cases} \tag{23}$$

6. *Save model to be used for any* $x \in \Omega$ *and* $t \in]0, T[$.

6. Convergence of the L^2 error $\tilde{\mathcal{J}}(\tilde{u}, \tilde{P})$

Let the L^2 error $\tilde{\mathcal{J}}(\tilde{u}, \tilde{P})$ measure how well neural networks $\tilde{u}$ and $\tilde{P} = (\tilde{a}, \tilde{u_0}, \tilde{f})$ satisfy observations, differential operator, boundary condition, initial condition and a priori knowledge.

$$\begin{aligned} \tilde{\mathcal{J}}(\tilde{u}, \tilde{P}) = &\left\| \partial_t \tilde{u} + \mathcal{L}(\tilde{a}, \tilde{u}) - \tilde{f} \right\|_{L^2(]0,T[\times \Omega)}^2 + \left\| \tilde{u}(t=T) - u^{ob} \right\|_{L^2(\Omega)}^2 + \left\| \tilde{P} - p^b \right\|_{S_P}^2 + \\ &\left\| \tilde{u}|_{\partial\Omega} - g \right\|_{L^2(]0,T[\times \partial\Omega)}^2 + \left\| \tilde{u}(t=0) - \tilde{u_0} \right\|_{L^2(\Omega)}^2, \end{aligned} \tag{24}$$

where u^{ob} and p^b being interpolations of U^{ob} in $L^2(\Omega)$ and P^b in S_P.

$\tilde{\mathcal{J}}(\tilde{u}, \tilde{P})$ can be directly calculated from the PDE (18) for any approximations $(\tilde{u}, \tilde{P})$. The goal is to construct functions $(\tilde{u}, \tilde{P})$ for which $\tilde{\mathcal{J}}(\tilde{u}, \tilde{P})$ is as close to 0 as possible. Define $\mathfrak{C}^n$ and ${\mathfrak{C}_t}^n$ as classes of neural networks with a single hidden layer and n hidden units. Let $(\tilde{u}^n, \tilde{P}^n)$ be neural networks with n hidden units which minimize $\tilde{\mathcal{J}}(\tilde{u}, \tilde{P})$. We prove that, under certain conditions,

$$\text{there exists } (\tilde{u}^n, \tilde{P}^n) \in \mathfrak{C}_4^n \text{ such that } \tilde{\mathcal{J}}(\tilde{u}^n, \tilde{P}^n) \to 0, \text{ as } n \to \infty.$$

strongly in $L^2([0,T] \times \Omega) \times S$.

The proof requires the joint analysis of the approximation power of neural networks as well as the continuity properties of partial differential equations. We show that the neural networks can satisfy observations, the differential operator, boundary condition, and initial condition arbitrarily well for sufficiently large n,

$$\tilde{\mathcal{J}}(\tilde{u}^n, \tilde{P}^n) \to 0 \text{ as } n \to \infty.$$

In this section, we present a theorem guaranteeing the existence of multilayer feed forward networks $(\tilde{u}^n, \tilde{P}^n)$ able to universally approximate solutions (u, P) in the sense that there is $(\tilde{u}^n, \tilde{P}^n)$ that makes the objective function $\tilde{\mathcal{J}}$ arbitrarily small. To do so, we use the results of [13] on universal approximation of functions and their derivatives. We denote

$$\Omega_T = [0,T] \times \Omega \qquad \mathcal{O}[u,P] = \partial_t u - div_x\left(a(x) I_d \nabla_x u\right) - f.$$

Universal approximation results for single functions and their derivatives have been obtained under various assumptions in [13–15]. In this paper, we use Theorem 3 of [13]. Let us recall the setup appropriately modified for our case of interest. Let ψ be an activation function, e.g., of sigmoid type, of the hidden units and define the sets

$${\mathfrak{C}_t}^n(\psi) = \left\{ \zeta_t(t,x) : \mathbb{R}^{1+d} \mapsto \mathbb{R} : \zeta_t(t,x) = \sum_{i=1}^{n} \beta_i^t \psi\left(\alpha_{0,i}^t t + \sum_{j=1}^{d} \alpha_{j,i}^t x_j + c_j^t\right) \right\}, \tag{25}$$

where $\theta_t = (\beta_1^t, \ldots, \beta_n^t, \alpha_{0,1}^t, \ldots, \alpha_{d,n}^t, c_1^t, \ldots, c_d^t) \in \mathbb{R}^{2n+d(1+n)}$ compose the elements of the parameter space, and

$$\mathfrak{C}^n(\psi) = \left\{ \zeta(x) : \mathbb{R}^d \mapsto \mathbb{R} : \zeta(x) = \sum_{i=1}^{n} \beta_i \psi\left(\sum_{j=1}^{d} \alpha_{j,i} x_j + c_j\right) \right\}, \tag{26}$$

where $\theta = (\beta_1, \ldots, \beta_n, \alpha_{1,1}, \ldots, \alpha_{d,n}, c_1, \ldots, c_n) \in \mathbb{R}^{n+d(1+n)}$ compose the elements of the parameter space. We consider the product spaces:

$$\mathfrak{C}_4^n(\psi) = {\mathfrak{C}_t}^n(\psi) \times \mathfrak{C}^n(\psi) \times \mathfrak{C}^n(\psi) \times {\mathfrak{C}_t}^n(\psi) \;\; ; \quad \mathfrak{C}_4(\psi) = \mathfrak{C}_t(\psi) \times \mathfrak{C}(\psi) \times \mathfrak{C}(\psi) \times \mathfrak{C}_t(\psi),$$

where

$$\mathfrak{C}_t(\psi) = \bigcup_{n \geq 1} {\mathfrak{C}_t}^n(\psi) \;\; ; \qquad \mathfrak{C}(\psi) = \bigcup_{n \geq 1} \mathfrak{C}^n(\psi).$$

Then we have the following result.

Theorem 4. *Let $\mathfrak{C}^n(\psi)$ and ${\mathfrak{C}_t}^n(\psi)$ be given by (26) and 25 respectively where ψ is assumed to be in $C^2(\mathbb{R})$, bounded and non-constant. Assume that Ω_T is compact. In addition, assume that the problem has a unique solution (u, P). Then, for every $\epsilon > 0$, there exists a positive constant $K > 0$ such that there exists a $(\tilde{u}, \tilde{P}) \in \mathfrak{C}_4(\psi)$ that satisfies*

$$\tilde{\mathcal{J}}(\tilde{u}, \tilde{P}) \leq K\epsilon.\square$$

Proof.

By Theorem 3 of [13] we know that there is a function $(\tilde{u}, \tilde{P}) \in \mathfrak{C}_4(\psi)$ that is uniformly dense on compacts of C_4. Where $C_4 = \mathcal{C}^2(\mathbb{R}^{1+d}) \times \mathcal{C}^2(\mathbb{R}^d) \times \mathcal{C}(\mathbb{R}^d) \times \mathcal{C}(\mathbb{R}^{1+d})$. This means that for $(u, a, u_0, f) \in C_4$ and $\epsilon > 0$, there is $(\tilde{u}, \tilde{a}, \imath\tilde{u}_0, \tilde{f}) \in \mathfrak{C}_4(\psi)$ such that

$$\sup_{(t,x) \in \Omega_T} |\partial_t u(t,x) - \partial_t \tilde{u}(t,x;\theta_u)| + \max_{|\alpha| \leq 2} \sup_{(t,x) \in \Omega_T} |\partial_x^{(\alpha)} u(t,x) - \partial_x^{(\alpha)} \tilde{u}(t,x;\theta_u)| < \epsilon$$

$$\max_{|\alpha|\leq 2}\sup_{x\in\bar{\Omega}}|\partial_x^{(\alpha)}a(x)-\partial_x^{(\alpha)}\tilde{a}(x;\theta_a)|<\epsilon$$
$$\sup_{x\in\bar{\Omega}}|u_0(x)-\tilde{u_0}(x;\theta_{u_0})|<\epsilon$$
$$\sup_{(t,x)\in\Omega_T}|f(t,x)-\tilde{f}(t,x;\theta_f)|<\epsilon$$

For the objective function (note that $\mathcal{O}[u,P](t,x)=0$ for (u,P) that solves the PDE)

$$\begin{aligned}\tilde{\mathcal{J}}(\tilde{u},\tilde{P}) &= \left\|\mathcal{O}[\tilde{u},\tilde{P}]-\mathcal{O}[u,P]\right\|^2_{L^2(\Omega_T)}+\left\|\tilde{u}(t=T)-u^{ob}\right\|^2_{L^2(\Omega)}+\left\|\tilde{P}-p^b\right\|^2_{S_P}+\\ &\|\tilde{u}|_{\partial\Omega}-g\|^2_{L^2(]0,T[\times\partial\Omega)}+\|\tilde{u}(t=0)-\tilde{u_0}\|^2_{L^2(\Omega)}\\ &\leq \|\partial_t u(t,x)-\partial_t\tilde{u}(t,x;\theta_u)\|^2_{L^2(\Omega_T)}+\left\|f(t,x)-\tilde{f}(t,x;\theta_f)\right\|^2_{L^2(\Omega_T)}+\|\mathcal{L}(a,u)-\mathcal{L}(\tilde{a},\tilde{u})\|^2_{L^2(\Omega_T)}+\\ &\left\|\tilde{u}(t=T)-u^{ob}\right\|^2_{L^2(\Omega)}+\left\|\tilde{f}(t,x;\theta_f)-f^b(t,x)\right\|^2_{L^2(\Omega_T)}+\left\|\tilde{a}(x;\theta_a)-a^b(x)\right\|^2_{L^2(\Omega)}+\\ &\left\|\tilde{u_0}(x;\theta_{u_0})-u_0^b(x)\right\|^2_{L^2(\Omega)}+\|\tilde{u}|_{\partial\Omega}-g\|^2_{L^2(]0,T[\times\partial\Omega)}+\|\tilde{u}(t=0)-\tilde{u_0}\|^2_{L^2(\Omega)}\\ &\leq K\epsilon\end{aligned}$$

□

7. Numerical results and Analysis

7.1. One dimensional Heat equation

Let us start with the heat equation with Dirichlet boundary conditions. In one space dimension, the problem reads as:

$$\begin{cases}\partial_t u-\partial_x(a\partial_x u)=f,\ \ \forall x\in\,]0,1[\,,\ \forall t\in\,]0,1[\,,\\ u(t,0)=u(t,1)=0,\ \ \forall t\in\,]0,1[\,,\\ u(0,x)=u_0(x),\ \ \forall x\in\,]0,1[\,.\end{cases}\tag{27}$$

From measures at 50 couples of (T,x) created from 50 space points at final instant, we try to find $P=(a,u_0,f)$. We represent the solution u and P by four neural networks with 5 L_2 layers and 10 neurons in each sub-layer. As for the activation functions, we use *tanh*. In general, the choice of a neural network's architecture (number of layers, neurons and form of activation functions) is crucial and in many cases still remains an art that relies on one's ability to balance the trade off between expressivity and trainability of the neural network. Our tests indicate that deeper and wider networks are usually more expressive but are often more costly to train (i.e., a feed-forward evaluation of the neural network takes more time and the optimizer requires more iterations to converge). The neural networks are trained by minimizing the sum of squared errors loss of equation (22).

To illustrate the effectiveness of our approach, we applied method of manufactured solution (MMS), i.e. construct some equations with analytic solutions and then compare the numerical solution with the analytical solutions..

Consider the analytical solutions of (27):

$$a(x)=\frac{1+e^x}{4};\quad u(t,x)=\frac{(e^x-1)(x-1)^2}{1+t},$$

from a and u below, we can calculate the analytical source f and initial condition u_0. The neural network parameters are initialized using normal initialization and updated using the well-known ADAM algorithm (see [16]) with a decaying learning rate schedule (23). We use approximately 100 epochs. The data set of training is generated randomly by uniform distribution. 90 time points and 90 space points are chosen

randomly to create 8100 couples of (t, x) to train the neural networks. The data is sampled to mini-batches of size 96. We implement the algorithm using TensorFlow which is a software library for deep learning.

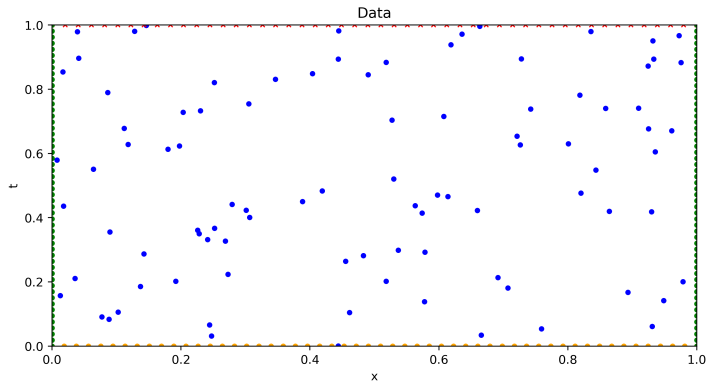

Figure 4. Data generation for training neural networks. Here • represent the sampling points within the domain, • represent the sampling points on the boundary, • represent the initial sampling points and × represent the points of observations.

After training, we test the trained model on a testing data of 2916 couples of (t, x) which are uniformly generated. The results below are calculated on the testing data.

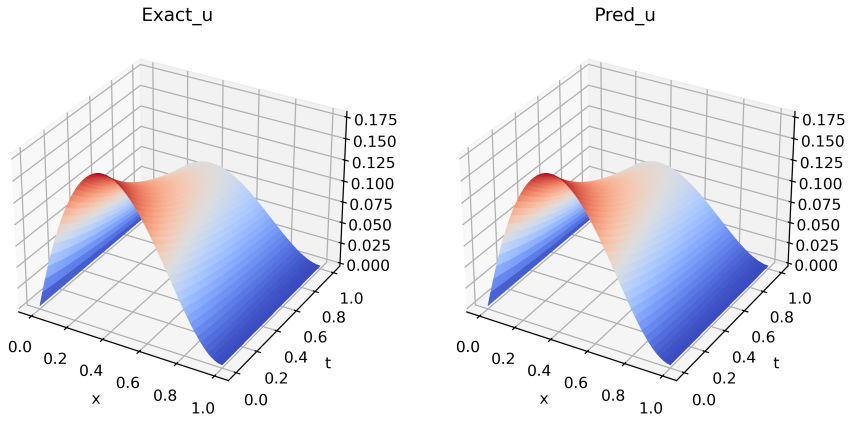

Figure 5. The exact solution u of equation (27) (left panel) is compared to the solution $\tilde{u}$ predicted by DL-IP (right panel). It is hard to distinguish predicted solution and exact solution by eyes.

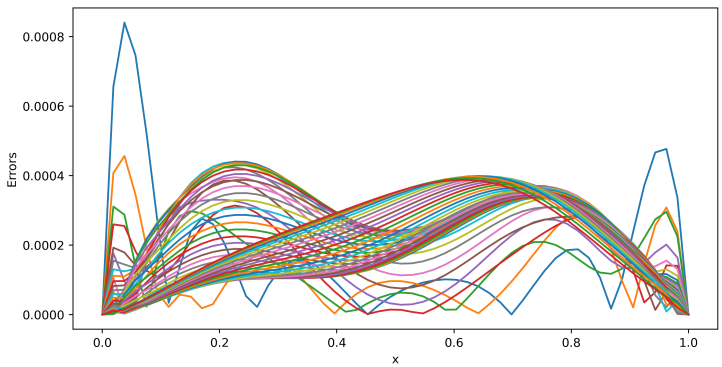

Figure 6. Absolute space errors between predicted solution and exact solution at all points of time. The maximal error value is $8.3997e-04$. On the training data points it is $2.0591e-06$.

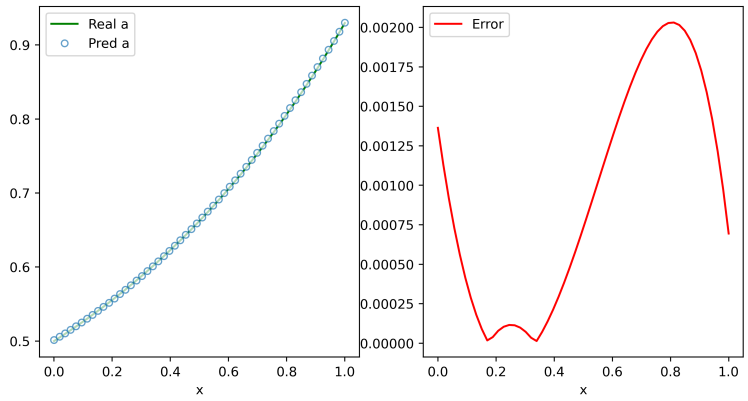

Figure 7. The exact coefficient a is presented by $-$ and the coefficient $\tilde{a}$ predicted by DL-IP is presented by $\circ$ (left panel). Right panel represents the absolute error between the solutions. It is also hard to distinguish predicted and exact coefficients by eyes. The maximal error value is $2.0306e-03$. On the training data points it is $4.3610e-07$.

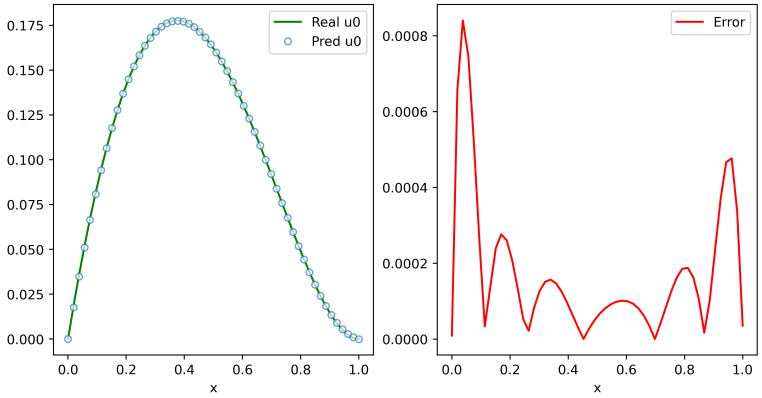

Figure 8. The exact initial condition u_0 is presented by $-$ and the predicted one $\tilde{u_0}$ is presented by $\circ$ (left panel). Right panel represents the absolute error between the solutions. It is also hard to distinguish predicted and exact initial conditions by eyes. The maximal error value is $8.3997e-04$. On the training data points it is $8.1583e-08$.

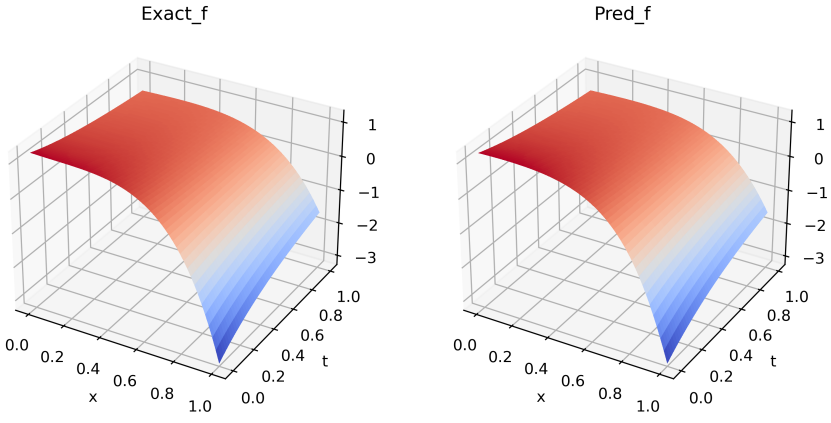

Figure 9. The exact source f of equation (27) (left panel) is compared to $\tilde{f}$ predicted by DL-IP (right panel). It is hard to distinguish exact and predicted sources by eyes.

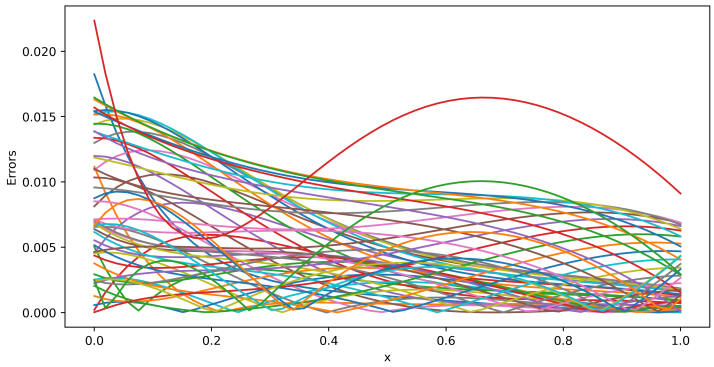

Figure 10. Absolute space errors between exact and predicted sources at all time points of testing data. The maximal error value is $2.2345e-02$. On the training data points it is $3.2232e-05$.

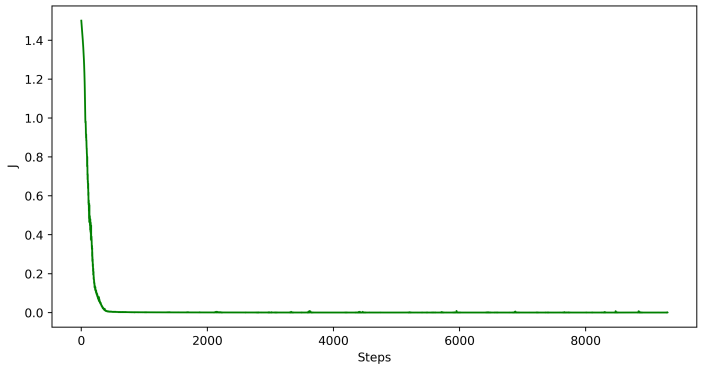

Figure 11. Variation of functional $\tilde{J}$ within 100 epochs which are equivalent to 9200 steps. Across the iterations the loss is decreasing even if there are some oscillations; its minimal value is $3.1398e-05$.

Furthermore, we performed two systematic studies of the impact of noise levels and a priori knowledge amounts by keeping the total number of training observations as well as the neural network architectures fixed to the settings described above. The results of the first study are summarized in table (1) and figure (12). For the second one we present the results in table(2) and figure (13).

Table 1. Variation of functional $\tilde{J}$ in terms of b the noise between real value of P and the used one P^b, it means $P^b = P + b$. $(b = 0)$ is corresponding to the clear data. Here, the a priori knowledge is kept fixed at 100 points.

Noise b	0	$10e-5$	$10e-4$	$10e-2$
$\tilde{J}$	$4.9279e-05$	$3.8593e-05$	$4.6976e-05$	$5.8740e-05$
$error_a$	$1.3595e-03$	$3.0966e-03$	$2.544641e-03$	$1.0210e-01$
$error_{u_0}$	$6.5393e-04$	$6.3966e-04$	$1.419791e-03$	$1.0136e-01$
$error_f$	$3.9217e-02$	$3.7173e-02$	$3.065276e-02$	$1.1334e-01$
$error_u$	$6.5393e-04$	$2.8137e-03$	$1.4197e-03$	$1.0136e-01$

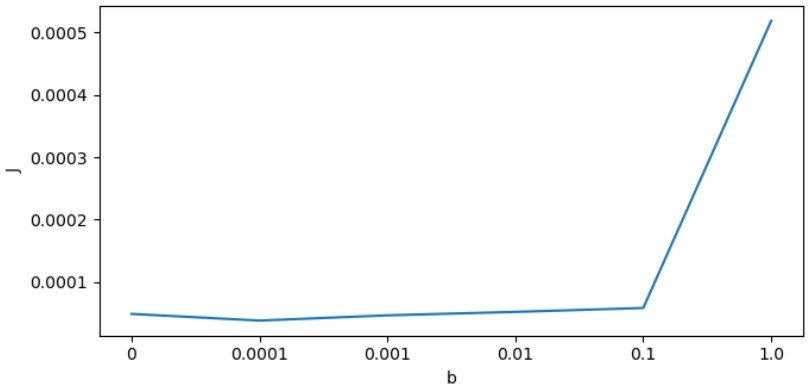

Figure 12. Graph of table (1) results. By increasing noise b minimal value of $\tilde{J}$ increases too.

Table 2. Variation of functional $\tilde{J}$ in terms of h the amount of a priori knowledge. Here, the noise is kept fixed at 0.

h	25	900	2500	3600	6400
$\tilde{J}$	$5.4914e-05$	$5.4113e-05$	$3.6522e-05$	$3.5627e-05$	$3.2917e-05$
$error_a$	$6.9173e-03$	$1.8087e-03$	$1.6346e-03$	$1.0738e-03$	$3.4335e-03$
$error_{u_0}$	$6.9173e-03$	$1.6645e-03$	$2.0992e-04$	$3.5799e-04$	$3.3736e-04$
$error_f$	$1.5644e-01$	$5.8674e-02$	$6.7514e-02$	$5.3573e-02$	$5.1853e-02$
$error_u$	$6.9173e-03$	$1.1827e-03$	$4.7244e-04$	$2.5719e-04$	$5.7335e-04$

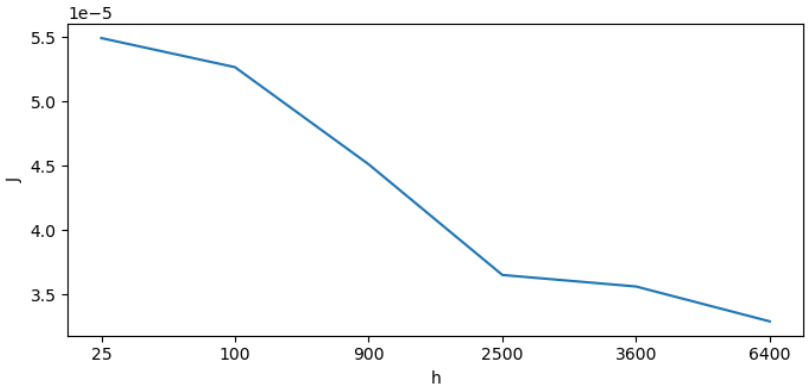

Figure 13. Graph of table (2) results. By increasing the amount of a priori knowledge h minimal value of $\tilde{J}$ decreases.

The key observation here is that less noise in the data enhances the performance of the algorithm. Our experience so far indicates that the negative consequences of more noise in the data can be remedied to some extent by obtaining more data.

7.2. Limitation of DL-IP method

For inverse or identification problem associated to the evolution problem like heat equation, it is interesting to reconstruct solutions from final observations and without a priori knowledge $h = 0$. DL-IP method can reconstruct the solutions from final observations but with at least 25 of a priori knowledge. This is the main drawback associated with DL-IP method. We present in figures below some results of this limitation.

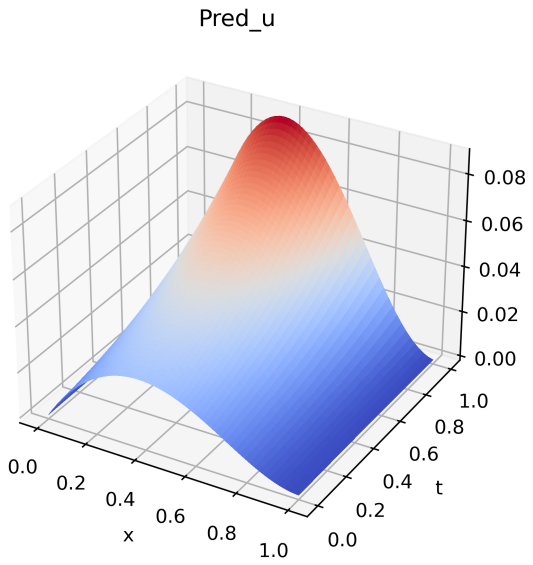

Figure 14. The solution $\tilde{u}$ predicted by DL-IP without any a priori knowledge. The maximal error value is 0.0423.

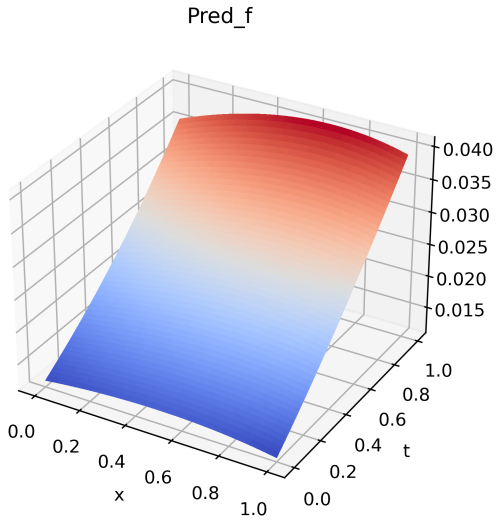

Figure 15. The solution $\tilde{f}$ predicted by DL-IP without any a priori knowledge. The maximal error value is $3.4016e+01$.

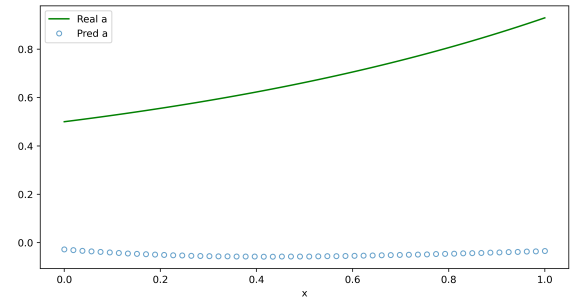

Figure 16. The exact coefficient a is presented by — and the coefficient $\tilde{a}$ predicted by DL-IP without any a priori knowledge is presented by ∘. The maximal error value is $9.6417e-01$.

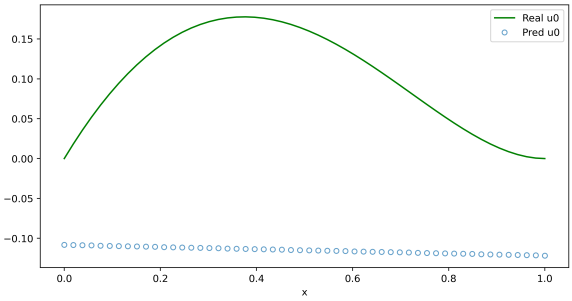

Figure 17. The exact initial condition u_0 is presented by — and the predicted one without any a priori knowledge $\tilde{u_0}$ is presented by ∘. The maximal error value is $2.9096e-01$.

It should also be noted that from an a priori final knowledge the reconstruction cannot be done. Even if the coefficient a and the initial condition u_0 are well predicted, the source term f is poorly predicted since it is a spatio-temporal function.

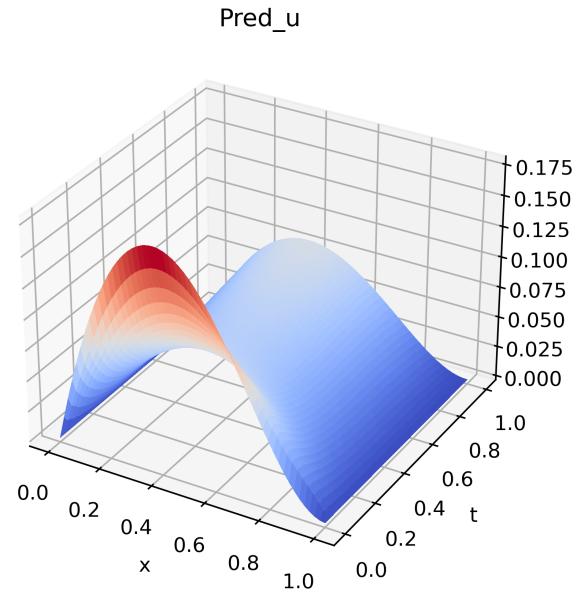

Figure 18. The solution $\tilde{u}$ predicted by DL-IP with final a priori knowledge. The maximal error value is $7.2011e-02$.

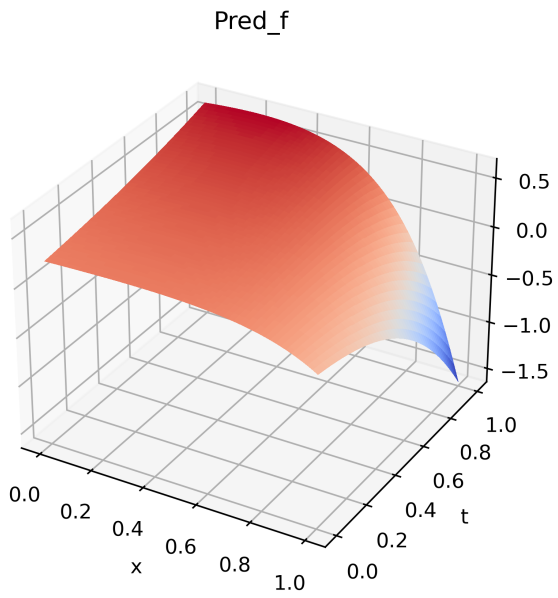

Figure 19. The solution $\tilde{f}$ predicted by DL-IP without any a priori knowledge. The maximal error value is $3.1105e+00$.

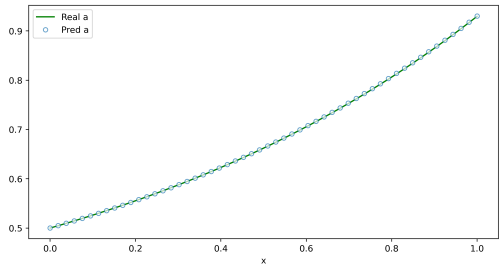

Figure 20. The exact coefficient a is presented by — and the coefficient $\tilde{a}$ predicted by DL-IP with final a priori knowledge is presented by ○. The maximal error value is $1.1479e-03$.

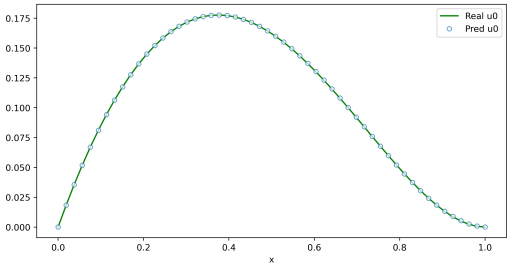

Figure 21. The exact initial condition u_0 is presented by — and the predicted one with final a priori knowledge $\tilde{u_0}$ is presented by ○. The maximal error value is $3.0998e-04$.

8. Comparison of methods

In this section we make a comparison between DL-IP's model and classical methods such as Tikhonov regularization and hybrid approach.

Firstly we make a general comparison of algorithms in table (3) by answering the following questions:

- Is the method meshfree (MF)?

- Does it require interpolations (I)?
- Does it need Numeric Integration (NI)?
- How many direct problems we need to solve (DP)?

Table 3. General comparison of algorithms.

	DL-IP model	Tikhonov regularization	Hybrid Method
MF	No	Yes	Yes
I	No	Yes	Yes
NI	No	Yes	Yes
DP	0	2 by descent iteration	thead2 by descent iteration + 1 by fitness call

Secondly, we consider the example (7.1), we solve the problem by Tikhonov regularization (table 4), Hybrid Method (table 5), and we make a comparison with results of DL-IP's model (table 6) by considering amount of knowledge h, noise b, minimal values of $\tilde{J}$ or J_T and errors between exact solution and predicted one (absolute error $error_u$ and relative error $\%error_u$). We consider 50 observations, the size of mesh is $S = MN$ with $M = N = 100$. Parameters of Armijo-Goldstein's rule are $\alpha = 0.3$, $\beta = 10e-4$, $\alpha_i = 0.9$. Regularization coefficient is $\varepsilon = 0.005$. we turn hybrid approach with an initial population of size 10000, 500 individuals and 500 generations.

- **Tikhonov regularization.**

Table 4. Results of the Tikhonov approach. By increasing noise b or decreasing a priori knowledge h, minimal values of J_T and errors increase. Solution cannot be reconstructed from partial knowledge

h	b	J_T	$error_u$	$\%error_u$
100	$10e-2$	$2.3471e-05$	$3.1232e-01$	$1.8347e+00$
	$10e-5$	$2.9642e-05$	$3.1758e-01$	$1.7483e+00$
900	$10e-2$	$1.8643e-05$	$3.0199e-01$	$2.0137e+00$
	$10e-5$	$2.7292e-05$	$2.9533e-01$	$1.8559e+00$
S	$10e-2$	$1.1090e-06$	$1.0162e-01$	$2.2864e-01$
	$10e-5$	$1.0894e-06$	$3.6399e-03$	$2.4319e-02$

- **Hybrid method**

Table 5. Results given by the hybrid method with $b = 10e-4$. By decreasing a priori knowledge h, minimal values of J_T and errors increase. Solution cannot be reconstructed from partial knowledge.

h	J_T	$error_u$	$\%error_u$
100	$1.8332e-05$	$3.0238e-01$	$1.8283e+00$
900	$1.6953e-05$	$3.0160e-01$	$1.9380e+00$
S	$1.9603e-09$	$2.3773e-03$	$2.2907e-03$

- **DL-IP's model**

Table 6. Results given by DL-IP's model.

h	b	$\tilde{J}$	$error_u$	$\%error_u$
25	$10e-2$	$2.7900e-05$	$1.1098e-01$	$2.3676e-01$
	$10e-5$	$3.0141e-05$	$2.2934e-02$	$1.6950e-02$
100	$10e-2$	$9.7133e-05$	$1.0189e-01$	$2.3837e-01$
	$10e-5$	$3.7580e-05$	$9.8341e-04$	$2.3096e-03$
900	$10e-2$	$4.0489e-05$	$1.0180e-01$	$2.3888e-01$
	$10e-5$	$3.6766e-05$	$6.7920e-04$	$3.1624e-03$

According to the results of tables below (3), (4), (5), (6), we can see that:

- Solutions cannot be reconstructed from partial knowledge by Tikhonov regularization or hybrid method even from 900 points of a priori data.

- From 25 points of a priori knowledge we can reconstruct PDE's solution with $error_u = 2.2934e-02$ by deep learning approach.
- With maximal a priori knowledge, DL-IP is more stable to noise b than classical methods. With $b = 10e-3$, we get a solution with $error_u = 2.7851e-05$ by DL-IP on testing data, and with $error_u = 2.0916e-01$ by regularization approach.
- For classical methods we should use a percent of a priori knowledge calculated as $h\% = \frac{100h}{S}$. Since our tests indicate that to get some precision the amount increases by increasing size of mesh.
- Solution reconstructed by hibrid approach from total data is better than the solution of Tikhonov regularization.

9. Conclusion

We have presented in this paper a new approach based on deep learning for the determination of parameters $P = (a, u_0, f)$ in linear parabolic equation from final data observations and partial a priori knowledge. Since the classical methods are not meshfree and require total a priori knowledge, our algorithm covers a great need in the domain of inverse problems.

Deep learning could become a valuable approach for solving inverse problems, which are important in physics, engineering, and finance. The PDE solution and its parameters can be approximated with deep neural networks, which are trained to satisfy the differential operator, initial condition, final observations, a priori knowledge, and boundary conditions. Our deep learning algorithm for solving inverse problems is mesh-free, which is key since meshes become infeasible in higher dimensions. Instead of forming a mesh, the neural network is trained on batches of randomly sampled time and space points. The algorithm can be easily modified to apply to hyperbolic, elliptic, and nonlinear equations. The algorithm remains the same for these other types of PDEs. However, the numerical performance of these other types of PDEs remains to be investigated. Developing deep learning algorithms that do not require any a priori knowledge or at least require final a priori knowledge would certainly be interesting. We leave this question for future work.

Conflicts of Interest: The author declares no conflict of interest.

Data Availability: All data required for this research is included within this paper.

Funding Information: This research is funded by Higher Education Commission of Pakistan.

References

[1] Keller, J. B. (1976). Inverse problems. *The American Mathematical Monthly, 83*, 107-118.

[2] Kern, M. (2016). *Numerical methods for inverse problems*. John Wiley & Sons.

[3] Hoppe, F., & Neitzel, I. (2021). A-posteriori reduced basis error-estimates for a semi-discrete in space quasilinear parabolic PDE. *Computational Optimization and Applications*, 1-30.

[4] Atifi, K., Balouki, Y., Essoufi, E. H., & Khouiti, B. (2017). Identifying initial condition in degenerate parabolic equation with singular potential. *International Journal of Differential Equations*, 2017.

[5] Atifi, K., Essoufi, E. H., & Sidi, H. O. (2018). Identification of a Diffusion Coefficient in Degenerate/Singular Parabolic Equations from Final Observation by Hybrid Method. *Open Journal of Mathematical Analysis, 2*(2), 142-155.

[6] Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics, 375*, 1339-1364.

[7] Ali, A., Correia, A., Naiff, D., Jardim, G., & Saporito, Y. (2018). Solving Nonlinear and High-Dimensional Partial Differential Equations via Deep Learning. Retrieved from https://doi.org/10.48550/arXiv.1811.08782

[8] Xu, K., Shi, B., & Yin, S. (2018). *Deep Learning for Partial Differential Equations*. Stanford University.

[9] Allaire, G. (2004). Analyse numérique et optimisation: une introduction à la modélisation mathématique et à la simulation numérique. École polytechnique, Département de mathématiques appliquées.

[10] Evans, L. C. (2022). *Partial differential equations* (Vol. 19). American Mathematical Society.

[11] Chen, X., Jüngel, A., & Liu, J. G. (2014). A note on Aubin-Lions-Dubinskii lemmas. *Acta Applicandae Mathematicae, 133*, 33-43.

[12] Dussault, J. P. (2008). La différentiation automatique et son utilisation en optimisation. *RAIRO-Operations Research, 42*(2), 141-155.

[13] Chen, T., Chen, H., & Liu, R. W. (1995). Approximation capability in C (R /sup n/) by multilayer feedforward networks and related problems. *IEEE Transactions on Neural Networks, 6*(1), 25-30.

[14] Hornik, K., Stinchcombe, M., & White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks, 3*(5), 551-560.

[15] Lewicki, G., & Marino, G. (2004). Approximation of functions of finite variation by superpositions of a Sigmoidal function. *Applied Mathematics Letters, 17*(10), 1147-1152.

[16] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

© 2023 by the authors; licensee PSRP, Lahore, Pakistan. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).