Armenian Journal of Mathematics
Volume 15, Number 8, 2023, 1–11
https://doi.org/10.52737/18291163-2023.15.8-1-11

A Mesh-Free Algorithm to Solve an Inverse Source Problem for Degenerate Two-Dimensional Parabolic Equation from Final Observations

K. Atifi

Abstract. The main purpose of this work is to propose a new network architecture model for deep learning applied to solve an inverse source problem for a two-dimensional degenerate parabolic equation from final observations with degeneracy occurring anywhere in the spatial domain.

Key Words: Deep Learning, Heat Equation, Inverse Problem, Degenerate Two-Dimensional Equation, Optimization
Mathematics Subject Classification 2010: 15A29, 47A52, 34A38, 93C20, 60J70, 35K05, 35K65

Introduction

Deep neural networks are machine learning models that have achieved remarkable success in a number of domains, from visual recognition and speech to natural language processing and robotics (see [6]).

Recently, Sirignano and Spiliopoulos [7] proposed to solve PDEs using a mesh-free deep learning algorithm. The method is similar in spirit to the Galerkin method but with several key changes using ideas from machine learning. The Galerkin method is a widely-used computational method which seeks a reduced-form solution to a PDE as a linear combination of basis functions. The deep learning algorithm, or the deep Galerkin method (DGM), uses a deep neural network instead of a linear combination of basis functions. The deep neural network is trained to satisfy the differential operator, initial condition, and boundary conditions using stochastic gradient descent at randomly sampled spatial points. By randomly sampling spatial points, the authors avoid the need to form a mesh and instead convert the PDE problem into a machine-learning problem.

Kailai and Eric [5] propose a new method that estimates the unknown distribution by matching the statistical properties between observed and simulated random processes. The authors approximate the unknown distribution and use a discriminative neural network for computing the statistical discrepancies between the observed and simulated random processes. Hyeontae et al. [4] construct approximated solutions of differential equations using the deep neural network. In the aim to resolve an inverse problem in medical imaging, Chang et al. [3] provide a systematic basis for learning the causal relationship regarding the structure of the training data suitable for deep learning to solve highly underdetermined problems.

In this paper, based on the idea proposed in [7], we introduce a new network architecture model for deep learning and apply it to solve an inverse source problem for a degenerate two-dimensional parabolic equation from final observations. This new deep neural network is trained to satisfy the differential operator, initial condition, boundary, and observability conditions. Our algorithm is mesh-free.

This work is the continuation of [1], in which we identified the initial state in a degenerate two-dimensional parabolic equation.

Now, we start by treating this inverse problem theoretically.

Consider the following problem:

$$\begin{cases} \partial_t u(x,y,t) - div\,(a(x,y)I_2\nabla u(x,y,t)) = h(t)R(x,y,t), \\ \qquad\qquad\qquad\qquad (x,y)\in\Omega, t\in]0;T[, \\ u(x,y,t)=0, \qquad\qquad (x,y)\in\partial\Omega, t\in]0;T[, \\ u(x,y,0)=u_0(x,y), \qquad (x,y)\in\Omega, \end{cases} \tag{1}$$

where Ω is an open bounded subset of $\mathbb{R}^2$, $a\in C^1(\bar{\Omega})$, $a\geqslant 0$ everywhere ($a(\cdot,\cdot)$ can be equal to zero at any point in Ω), $R\in L^2(\Omega\times]0,T[)$, and $h\in L^2(0,T)$.

Let us introduce the following functional space

$$H_a^1(\Omega)=\left\{u\in L^2(\Omega):\sqrt{a}\nabla u\in L^2(\Omega) \text{ and } u(x,y)=0 \text{ for any } (x,y)\in\partial\Omega\right\},$$

with

$$\| u \|^2_{H_a^1(\Omega)} = \| u \|^2_{L^2(\Omega)} + \| \sqrt{a}\nabla u \|^2_{L^2(\Omega)}.$$

Then the weak formulation of problem (1) is

$$\int_\Omega \partial_t uv\; dxdy + \int_\Omega a(x,y)\nabla u\nabla v dxdy = \int_\Omega fv dxdy, \qquad v\in H_0^1(\Omega).$$

Let us define the bilinear form

$$B[u,v]=\int_\Omega a(x,y)\nabla u\nabla v dxdy.$$

The bilinear form B is noncoercive.

Denote $\mathcal{U} = \{h \in H^1(0,T) : \|h\|_{H^1(0,T)} \leqslant r\}$, where r is a real strictly positive constant. Obviously, the set $\mathcal{U}$ is a compact of $L^2(0,T)$.

Inverse Source Problem (ISP). Let u be the solution to (1). Assuming R is known, determine the time-dependent source part h from the measured data u^{obs} at the final time $u(T,\cdot)$.

Remark 1 *It should be mentioned that we do not need the supplement distributed measurements to obtain the numerical solution of the inverse problem.*

We treat the ISP by interpreting its solution as a minimizer of the following optimization problem:

$$\min_{h\in\mathcal{U}} J(u,\ h), \quad J(u,\ h) = \frac{1}{2}\|u(\cdot,T) - u^{obs}\|^2_{L^2(\Omega)} + \frac{\varepsilon}{2}\left\|h - \bar{h}\right\|^2_{L^2(0,T)} \tag{2}$$
subject to u is a solution to (1),

where $\bar{h}$ is an *a priori* (background state) knowledge of the state h^{exact}, and $\varepsilon > 0$ is the regularization parameter.

Minimization problem (2) can be equivalently formulated as follows:

$$\begin{aligned}&\min_{h\in\mathcal{U}} \mathcal{J}(u,\ h),\\ &\mathcal{J}(u,\ h) = J(u,h) + \frac{1}{2}\|A(u) - f\|^2_{L^2(\Omega)} + \frac{1}{2}\|u\|^2_{L^2(\partial\Omega)\times L^2(0,T)}\\ &\qquad + \frac{1}{2}\|u(t=0) - u_0\|^2_{L^2(\Omega)},\end{aligned} \tag{3}$$

where

$$f(x,y,t) = h(t)R(x,y,t)$$

and

$$A(u) = \partial_t u(x,y,t) - div\left(a(x,y)I_2\nabla u(x,y,t)\right), \qquad (x,y,t) \in \Omega\times]0;T[.$$

We want to approximate $(u(x,y,t), h^{exact})$ with a deep neural network

$$z = (\tilde{u}(x,y,t;\theta_u), \tilde{h}(t;\ \theta_h)),$$

where $\theta_u,\ \theta_h \in \mathbb{R}^k$ are the neural network's parameters. The goal is to find a set of parameters $\theta = (\theta_u, \theta_h)$ such that the function z minimizes the error $\mathcal{J}(z)$. If the error $\mathcal{J}(z)$ is small, then $\tilde{u}(x,y,t;\theta_u)$ will closely satisfy the PDE differential operator, boundary conditions, and initial condition. Therefore, θ_u which minimizes $\mathcal{J}$ produces a reduced-form model $\tilde{u}(x,y,t;\theta_u)$ which approximates the PDE's solution $u(x,y,t)$.

1 Well-posedness of the problem

We recall the following result.

Theorem 1 *[1] For all $f \in L^2(\Omega\times]0,T[)$ and $u_0 \in H^1_a(\Omega)$, there exists a unique weak solution which solves problem* (1) *and is such that*

$$u \in L^2\left(0,T; H^1_a(\Omega)\right) \cap L^\infty\left(0,T; L^2(\Omega)\right), \qquad \partial_t u \in L^2(0,T;L^2(\Omega)),$$

and

$$\sup_{t\in[0,T]} \parallel u(t) \parallel^2_{L^2(\Omega)} + \int_0^T \parallel \partial_t u \parallel^2_{L^2(\Omega)} dt + \int_0^T \parallel \sqrt{a}\nabla u \parallel^2_{L^2(\Omega)}$$
$$\leqslant C\left(\parallel f \parallel^2_{L^2(0,T;L^2(\Omega))} + \parallel u_0 \parallel^2_{H^1_a(\Omega)}\right),$$

with constant C depending on Ω and T.

Lemma 1 *Let u be the weak solution of* (1) *corresponding to a given initial state u_0. Then the input-output operator*

$$\varphi : L^2(0,T) \longrightarrow L^2\left(0,T; H^1_a(\Omega)\right) \cap L^\infty\left(0,T; L^2(\Omega)\right), \qquad \varphi(h) := u, \quad (4)$$

is Lipschitz continuous.

Proof. Let $\delta h \in L^2(0,T)$ be a small variation such that $h + \delta h \in \mathcal{U}$. Consider $\delta u = u^\delta - u$, where u is the weak solution of (1) with initial state u_0 and u^δ is the weak solution of (1) with source term $h^\delta = h + \delta h$. Then δu is the solution to the following problem:

$$\begin{cases} \displaystyle\int_\Omega \partial_t \delta u v dxdy + \int_\Omega a(x,y)\nabla\delta u\nabla v dxdy = \int_\Omega \delta h R v dxdy, \quad v \in H^1_0(\Omega), \\ \delta u(x,y,t) = 0, \qquad (x,y) \in \partial\Omega, t \in]0;T[, \\ \delta u(x,y,0) = 0, \qquad (x,y) \in \Omega. \end{cases}$$

Hence, δu is a weak solution of (1). By Theorem 1, there exists C, depending only on Ω and T, such that

$$\parallel \delta u \parallel^2_{L^2(0,T;H^1_a(\Omega))} \leqslant C \parallel \delta h \parallel^2_{L^2(0,T)}$$

and

$$\parallel \delta u \parallel^2_{L^\infty(0,T;L^2(\Omega))} \leqslant C \parallel \delta h \parallel^2_{L^2(0,T)} .$$

This implies the Lipschitz continuity of the input-output operator (4). □

An immediate consequence of Lemma 1 is the following result.

Proposition 1 *The functional J is continuous on $\mathcal{U}$, and there exists a unique minimizer $h^\star \in \mathcal{U}$ of $J(h)$, i.e.,*

$$J(h^\star) = \min_{h\in\mathcal{U}} J(h).$$

The differentiability of the functional J is deduced from the differentiability of the input-output operator (4) where u is the weak solution of (1) with time-dependent source part h.

We have the following result.

Proposition 2 *Let u be the weak solution of* (1) *with time-dependent source part h. Then the input-output operator* (4) *is G-derivable.*

Proof. Let $h \in \mathcal{U}$ and $\delta h \in L^2(0,T)$ be a small variation such that $h+\delta h \in \mathcal{U}$. Define the function

$$\varphi'(h) : \delta h \in L^2(0,T) \longrightarrow \delta u,$$

where δu is the solution to the following variational problem:

$$\begin{cases} \displaystyle\int_\Omega \partial_t \delta u v dxdy + \int_\Omega a(x,y)\nabla\delta u \nabla v dxdy = \int_\Omega \delta h R v dxdy, \quad v \in H_0^1(\Omega), \\ \delta u(x,y,t) = 0, (x,y) \in \partial\Omega, \quad t \in]0;T[, \\ \delta u(x,y,0) = 0, \qquad\qquad (x,y) \in \Omega, \end{cases}$$

and put

$$\phi(h) = \varphi(h+\delta h) - \varphi(h) - \varphi'(h)\delta h.$$

We want to show that $\phi(h) = o(\delta h)$.

It can be easily verified that function ϕ is a solution to following variational problem:

$$\begin{cases} \displaystyle\int_\Omega \partial_t \phi v dxdy + \int_\Omega a(x,y)\nabla\phi \nabla v dxdy = \int_\Omega (\delta h - (\delta h)^2) R dxdy, \; v \in H_0^1(\Omega), \\ \phi(x,y,t) = 0, (x,y) \in \partial\Omega, \quad t \in]0;T[, \\ \phi(x,y,0) = 0, \qquad\qquad (x,y) \in \Omega. \end{cases}$$

By the same way that was used in the proof of Lemma 1, we deduce that

$$\| \phi \|^2_{L^2(0,T;H^1_a(\Omega))} \leqslant C \| \delta h - (\delta h)^2 \|^2_{L^2(0,T)}$$

and

$$\| \phi \|^2_{L^\infty(0,T;L^2(\Omega))} \leqslant C \| \delta h - (\delta h)^2 \|^2_{L^2(0,T)} .$$

Hence, the input-output operator $\varphi : h \longrightarrow u$ is G-derivable. $\square$

2 Model and algorithm

To approximate (u, h^{exact}), we present a new model described by Fig. 1, where $\tilde{u}$ is the output of layers fully connected and $\tilde{h}$ is a multi-layered composition output that gives a Taylor series approximation of the time-dependent source part.

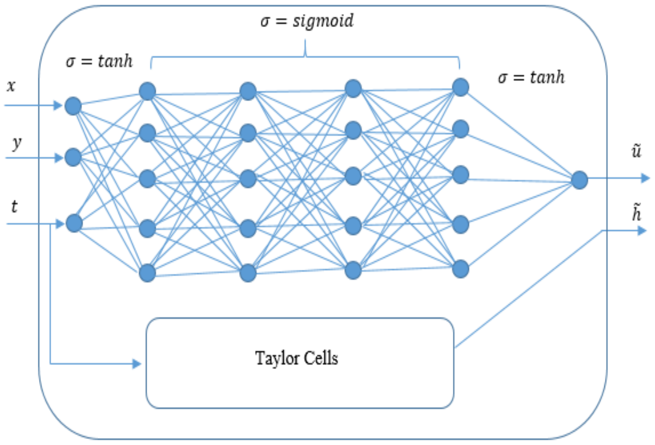

Figure 1: New model

The Taylor cell represented in Fig. 1 gives the following network architecture:

$$h = \sum_{k\leqslant n} W_k t^k,$$

which presents an expansion in Taylor series of order n.

The deep learning method for the inverse source problem (DL-ISP) approach approximates h and u by two deep neural networks with common inputs and loss function. Fig. 1 shows a visualization of the overall architecture given by

$$h(t) \simeq \tilde{h}(t) = N_h(t;\ \theta_h),\ \theta_h = (W_1, W_2, .., W_n),$$

$$u(x,y,t) \simeq \tilde{u}(x,y,t) = N_u(x,y,t;\theta_u),$$

where θ_u and θ_h are the parameters of neural networks.

Our goal now is to find $\theta = (\theta_u, \theta_h)$ such that u^{ob} is an observation of N_u, and $\bar{h}$ is the a priori knowledge of N_h.

The cost function is constructed as follows:

$$\begin{aligned}\tilde{\mathcal{J}}(\theta) = \frac{1}{2}\|A(\tilde{u}) - f\|^2_{L^2(\Omega)} + \frac{1}{2}\|\tilde{u}\|^2_{L^2(\partial\Omega)\times L^2(0,T)} + \frac{1}{2}\|\tilde{u}(t=T) - u^{obs}\|^2_{L^2(\Omega)} \\ + \frac{1}{2}\|\tilde{u}(t=0) - u_0\|^2_{L^2(\Omega)} + \frac{\varepsilon}{2}\|\tilde{h} - \bar{h}\|^2_{L^2(0,T)}.\end{aligned} \tag{5}$$

The derivatives of $\tilde{u}$ can be evaluated using automatic differentiation (see [2]) since it is parametrizing as a neural network.

Remark 2 *During the optimization, we observe that for all given $\tilde{h}$, one searches $\tilde{u}$ which minimizes the difference $A(\tilde{u}) - \tilde{h}(t)R(x, y, t)$. For that, the approximation $\tilde{h}$ is involved in the computation of $\tilde{u}$.*

DL-ISP 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.

4. Initialize the parameter set θ_0 and the learning rate α_0.

5. Repeat the following until convergence criterion is satisfied:

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

    2. Compute the loss functional $\tilde{J}(\theta_n, d_m)$ for the sampled mini-batch d_m.

    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 to update θ_{n+1}:

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

    The parameters are updated using the well-known ADAM algorithm with a decaying learning rate schedule.

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

We implement the algorithm using *TensorFlow*, which is software libraries for deep learning. *TensorFlow* has reverse mode automatic differentiation, which allows the calculation of derivatives for a broad range of functions. For example, *TensorFlow* can be used to calculate the gradient of the neural network 1 with respect to x or t, or θ. *TensorFlow* also allows the training of models on graphics processing units (GPUs).

3 Numerical results

For the simulations, in all tests below, we take the hyper parameters $L = 3$ (i.e., four hidden layers), $M = 50$ (number of units in each layer) for $\tilde{u}$, and $L = 6$ for $\tilde{h}$. The neural network parameters are initialized using the keras.initializers.glorot_normal initialization.

We do several tests, in order to show the performance of our approach. After training, we test the trained model on a testing data of 3000 couples of (t, x) which are uniformly generated. After 6000 epoch, we find the following results.

For the case $h^{exact}(t) = sin(\pi t)cos(\pi t) + 2$ and $n = 6$, see Fig. 2–4.

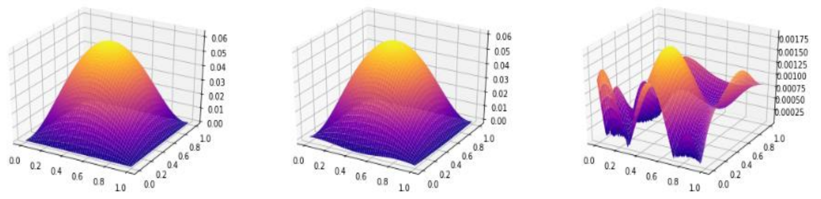

Figure 2: For $t = 0$, u constructed (left), u^{exact} (middle) and absolute error construction (right).

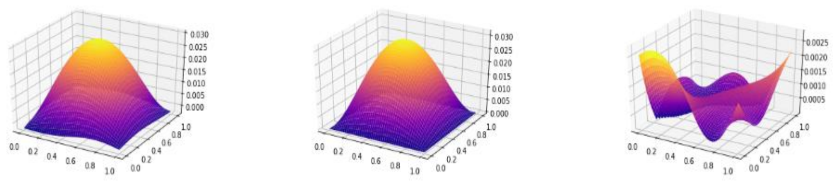

Figure 3: For $t = T$, u constructed (left), u^{obs} (middle) and absolute error construction (right).

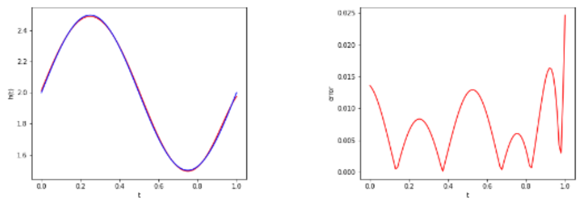

Figure 4: h constructed and h^{exact} (left), absolute errors between exact and predicted sources (right).

For the case $h^{exact}(t) = sin(\pi t)e^{\pi t} + 2$ and $n = 6$, see Fig. 5–7.

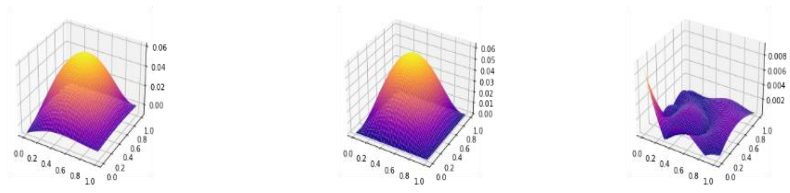

Figure 5: For $t = 0$, u constructed (left), u^{exact} (middle) and absolute error construction (right).

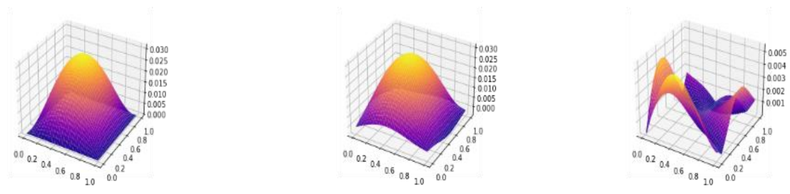

Figure 6: For $t = T$, u constructed (left), u^{obs} (middle) and absolute error construction (right).

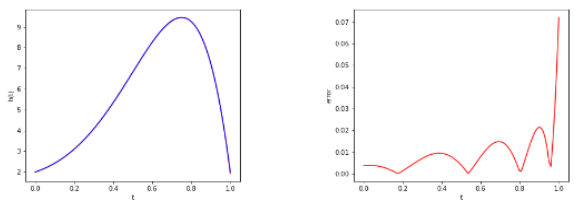

Figure 7: h constructed and h^{exact} (left), absolute errors between exact and predicted sources (right).

Finally, for the case $h^{exact}(t) = e^{sin(\pi t)}$ and $n = 6$, see Fig. 8–10.

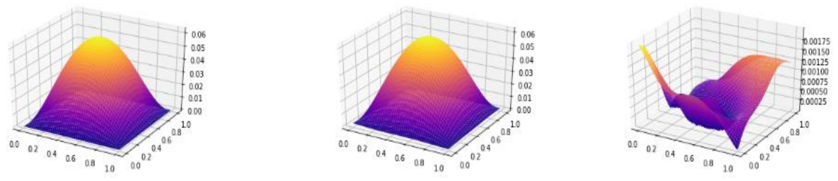

Figure 8: For $t = 0$, u constructed (left), u^{exact} (middle) and absolute error construction (right).

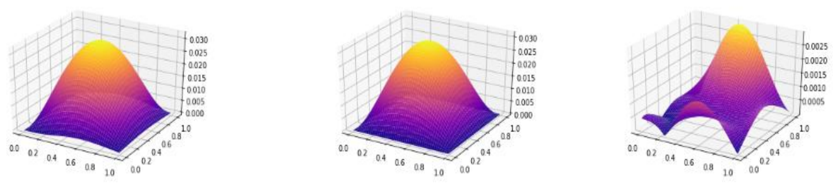

Figure 9: For $t = T$, u constructed (left), u^{obs} (middle) and absolute error construction (right).

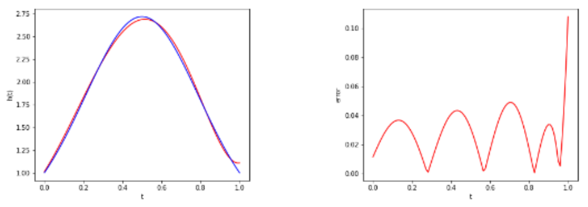

Figure 10: h constructed and h^{exact} (left), absolute errors between exact and predicted sources (right).

4 Conclusion

The deep learning DL-ISP Algorithm for solving PDEs presented in this paper is mesh-free, which is a key point 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. Moreover, suggested algorithm does not have a rounding errors caused by the discretization which have a very important role for the construction of solution.

The ease of implementing the DL-ISP Algorithm and the independence of this algorithm to PDEs make the method very efficient. Also, it presents an efficient way to solve nonlinear equations. But there remains the problem of the parametrization of the algorithm. The choice of the number of layers, number of units in each layer, and activation function turns out to be very important to have a good approximation of the solution.

References

[1] K. Atifi and E.-H. Essoufi, An inverse backward problem for degenerate two-dimensional parabolic equation. Opusc. Math., **40** (2020), no. 4, pp. 427–449. https://doi.org/10.7494/OpMath.2020.40.4.427

[2] J.-P. Dussault, La différentiation automatiqque et son utilisation en optimisation, RAIRO-Oper. Res., **155** (2008), no. 2, pp. 141–155. https://doi.org/10.1051/ro:2008007

[3] C.M. Hyun, S.H. Baek, M. Lee, S.M. Lee, and J.K. Seo, Deep learning-based solvability of underdetermined inverse problems in medical imaging. Preprint: arXiv:2001.01432v3, 2020.

[4] H. Jo, H. Son, H.J. Hwang and E. Kim, Deep neural network approach to forward-inverse problems. Preprint: arXiv:1907.12925v1, 2019.

[5] X. Kailai and D. Eric, Solving inverse problems in stochastic models using deep neural networks and adversarial training. Comput. Methods. Appl. Mech. Eng., **384** (2021), Article ID 113976. https://doi.org/10.1016/j.cma.2021.113976

[6] Y. LeCun, Y. Bengio and G. Hinton, Deep learning. Nature, **521** (2015), pp. 436–444. https://doi.org/10.1038/nature14539

[7] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations. J. Comput. Phys, **375** (2018), pp. 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029

Khalid Atifi
Faculty of Science and Technology,
Laboratory of Mathematics, Computer Science and Engineering Sciences (MISI), Hassan 1 University,
Settat 26000, Morocco.
k.atifi.uhp@gmail.com

Please, cite to this paper as published in
Armen. J. Math., V. **15**, N. 8(2023), pp. 1–11
https://doi.org/10.52737/18291163-2023.15.8-1-11