Solving Singular Liouville Equations Using Deep Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep learning has been applied to solving high-dimensional PDEs and successfully breaks the curse of dimensionality. However, it has barely been applied to finding singular solutions to certain PDEs, whose boundary conditions are absent and singular behavior is not a priori known. In this paper, we treat one example of such equations, the singular Liouville equations, which naturally arise when studying the celebrated Einstein equation in general relativity by using deep learning. We introduce a method of jointly training multiple deep neural networks to dynamically learn the singular behaviors of the solution and successfully capture both the smooth and singular parts of such equations.

1 Introduction

Most of the mathematical models arising from real world problems, if not all, are governed by Partial Differential Equations (PDEs). In most cases, closed form solutions to such equations do not exist. Recently, deep learning has been applied to finding numerical solutions to PDEs and showed the ability to handle high-dimensional PDEs [5, 4, 13, 11, 7, 2, 1]. However, apart from the high dimensional problem, the complex nature of PDEs also lies in the fact that their solutions may admit singularities. For example, the geometric shape of an American football can be described by a singular solution to Liouville equations. The singularities occur at the two "tips" on the American football, compared to European footballs where no such tips exist.

Deep learning has been barely applied to solving such singular PDEs due to the following reasons. First, the exact singular behavior of solutions is a priori unknown, such that one deep network model is not adequate to capture all the information. Moreover, there exists no efficient and satisfactory way for a neural network to approximate singular functions near singular points, to the best of our knowledge, despite the fact that smooth function can be approximated by neural networks [3, 6].

In this paper, we addressed the above challenges by encoding the different smooth and singular information of the PDEs into several loss functions and train multiple neural networks to dynamically learn the behavior of singular solutions near each singularity. More precisely, we treat the singular Liouville equations which govern metrics of "singular" spheres such as the mentioned American football. Our neural networks are designed to encode both the smooth and singular parts of the solutions, while at the same time each neural network is trained to only learn a smooth function. Our method is inspired by the success of physics-informed neural networks (PINNs) [12, 10, 9], which have been applied in a wide range of applications. PINNs are appropriate for solving nonlinear PDEs in the small data setting, which bears similarities to our objective Liouville equations.

Main contributions of our work are as follows. To the best of our knowledge, our work is the first to solve these types of singular PDEs numerically and our method provides a way of solving

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

PDEs whose boundary or initial conditions are present yet not a priori known. Our proposed method achieves good accuracy in terms of approximating singular solutions. Moreover, in this work we solve the singular Liouville equations on a closed manifold, i.e., a compact manifold without boundary. We address the issue that boundary conditions are absent for the PDEs by choosing appropriate coordinate charts for the objective manifold. Our work shows the potential of applying deep learning method to solving equations numerically on curved manifolds, not only the flat Euclidean space.

The motivation to numerically solve singular Liouville equations originated from authors' study on deformation of geometric shapes governed by such equations under change of parameters. Thus, the numerical results presented in the paper help understand the asymptotic behavior of singular manifolds from a differential geometric point of view. Moreover, we expect to extend our method to the situation where one deals with singularity of PDEs that exists continuously on divisors, not only discretely at points.

2 Methodology

2.1 Mathematical Standpoint of the Problem and a Baseline Method

We aim to solve the following singular Liouville equation numerically:

$$\begin{aligned} &\Delta u + e^{2u} = 0, \text{ in } \mathbb{R}^2 \setminus \{z_1, \dots, z_m\}, \\ &u(z) = v_i(z) + \log |z - z_i|^{\beta_i - 1}, \text{ for } z \text{ near } z_i, \ i = 1, \dots, m, \end{aligned} \tag{1}$$

where $\beta_i \in (0, 1)$ are known constants, each v_i is continuous, and most importantly, v_i is a priori unknown. In other words, solving (1) requires to find both a function u that satisfies (1) and a family of continuous functions v_i such that the difference of u and v_i is a log norm function. The solution u can be regarded as having singular asymptotic behavior at marked points z_i.

Remark 2.1. *Singular Liouville equation* (1) *naturally arises as we study the existence of constant curvature metric on* $\mathbb{S}^2$ *that has conical singularities. With the help of differential geometry, one knows that solution to* (1) *does exist. However, it is extremely difficult to find the analytical form of the solution, especially when* $m > 2$.

We ignore the easiest $m = 1$ case in (1) since then there does not exist solution to (1) [14]. From now on, fix $m = 2$. In such case there exists a unique solution to (1). Moreover, in this case the two cone angles β_1 and β_2 must be equal and z_1, z_2 must be the origin and the infinity point in $\mathbb{R}^2$ respectively. More precisely, we can write down the explicit form of the solution to (1) when $m = 2$, for some $\beta_1 = \beta_2 = \beta$ and $z_1 = (0, 0), z_2 = \infty$:

$$u_\beta := \frac{1}{2} \log \frac{4\beta^2 |z|^{2\beta - 2}}{(1 + |z|^{2\beta})^2}. \tag{2}$$

Question 2.2. *Can we solve* (1) *numerically using deep learning? More precisely, can we recover the solution* (2) *by training neural networks to approximately solve* (1) *when* $m = 2$*? Due to the existence of singularities, this is challenging and has not yet been solved numerically.*

A potential solution as the baseline. The challenge to numerically solve (1) comes from the fact that exact singular behavior of solutions is not a priori known and a boundary condition is absent. However, when $m = 2$, due to the existence of a closed form solution (2), we may use this as a priori knowledge and pose a boundary condition for (1). Then we propose two approaches based on DeepXDE [8], a library for scientific machine learning and physics-informed learning which requires boundary conditions to solve the equation. Treating this method as a baseline, we leave the details of theoretical setup and empirical results of these methods to the Appendix. Note that, these methods can not be either applied to the general $m > 2$ case nor used to find unknown singular solutions. Besides, empirical evidence implies that our method of training multiple neural networks, which will be introduced below outperforms DeepXDE even in this regime.

2.2 Proposed Method

To capture both the smooth and singular information of the solutions to (1), we design and jointly train multiple neural networks to numerically solve the singular Liouville equation. More precisely, we aim to approximate both the solution u to (1) and the a priori unknown continuous functions v_i for each i. The advantage is then all the neural networks are used to approximate smooth functions.

In more detail, denote by w the parameters in NNs, we train NNs

$$\mathcal{N}_u(x,y,w), \quad \mathcal{N}_i(x,y,w), \quad i=1,\ldots,m$$

so that $\mathcal{N}_u$ satisfies the first equation in (1), $\mathcal{N}_i$ approximates each v_i in (1) and the difference

$$\mathcal{N}_u(x,y,w)-\mathcal{N}_i(x,y,w)\approx \log|x+\sqrt{-1}y-z_i|^{\beta_i-1}$$

prescribes the conical singularities.

To encode smooth and singular information into loss functions and construct the corresponding physics-informed models, we define different loss functions to update the parameters. The loss function of $\mathcal{N}_u$ is defined as follows:

$$\ell_u(w):=\frac{1}{N}\cdot\sum\nolimits_{i=1}^{N}[\Delta\mathcal{N}_u(x_i,y_i,w)+e^{2\mathcal{N}_u(x_i,y_i,w)}]^2.$$

Denote by $\mathcal{N}_v$ and $\mathcal{N}_{v'}$ the NNs to capture conical singularities. The loss function of $\mathcal{N}_v$ is defined as

$$\ell_v(w):=\frac{1}{M}\sum\nolimits_{j=1}^{M}[\mathcal{N}_u(\hat{x}_j,\hat{y}_j,w)-\mathcal{N}_v(\hat{x}_j,\hat{y}_j,w)-\frac{\beta-1}{2}\log(\hat{x}_j^2+\hat{y}_j^2)]^2.$$

To approximate the singularity at ∞, we indeed train $\mathcal{N}_{v'}$ so that

$$\mathcal{N}_u\approx\mathcal{N}_{v'}-(1+\beta)\log|z|$$

for z restricted to $\{z\in\mathbb{R}^2:\delta_1<|z|<\delta_2\}$, where δ_1 and δ_2 are large. Thus, we sample points in this region and denote them by $\{(\tilde{x}_s,\tilde{y}_s)\}_{1\le s\le M}$. Then the loss function of $\mathcal{N}_{v'}$ are defined as follows:

$$\ell_{v'}(w):=\frac{1}{M}\sum\nolimits_{s=1}^{M}[\mathcal{N}_u(\tilde{x}_s,\tilde{y}_s,w)-\mathcal{N}_{v'}(\tilde{x}_s,\tilde{y}_s,w)+\frac{\beta+1}{2}\log(\tilde{x}_s^2+\tilde{y}_s^2)]^2.$$

In our experiments, we jointly optimize the NNs with the above losses $\ell_u(w)$, $\ell_v(w)$ and $\ell_{v'}(w)$. The next section introduces our experiment settings and results in detail.

3 Experiments

3.1 Experiment Settings

Data sampling: We first sample N points from $C_3:=\{(x,y)\in\mathbb{R}^2:\sqrt{x^2+y^2}<3\}$. As under stereographic projection, the area corresponding to C_3 on $\mathbb{S}^2$ covers 90% of the surface $\mathbb{S}^2$. We use $(x_i,y_i), i=1,\ldots,N$ to denote such data points. Next, we sample M points close to each prescribed singular point $z_i, i=1,\ldots,m$, by taking points from $C_{\epsilon,i}:=\{(x,y)\in\mathbb{R}^2:|(x,y)-z_i|<\epsilon\}$. Usually ϵ is set to 0.001. Denote by $(\hat{x}_j,\hat{y}_j), j=1,\ldots,M$ these data. In experiments, we set $N=1000$ and $M=1000$. For the sake of data precision, we avoid having points in $C_{\epsilon,i}$ that are too close to z_i. In our experiments, we avoid points whose distance to the singular point is less than 0.01ϵ. In practice, since we assume $m=2$, we indeed sample points that are close to $(0,0)\in\mathbb{R}^2$ or close to the infinity point, i.e., restricted to $\{z\in\mathbb{R}^2:\delta_1<|z|<\delta_2\}$ where δ_1 and δ_2 are large.

Implementation details: We train several multilayer perceptrons (MLPs) and denote by $d=4$ the number of layers and $k=50$ the number of units in each layer. We use the hyperbolic tangent function $\tanh(x):=\frac{e^{2x}-1}{e^{2x}+1}$. Note that the ReLU function is not appropriate for the problem since the second order derivatives of ReLU are trivial and thus it can not be used to approximate the solution to any second (or higher) order differential equations. In our experiments we find that a combined use of both first and second order methods gives the best result. In practice we first employ Adam and then L-BFGS to achieve smallest loss.

Evaluation metrics: We introduce two different metrics to evaluate the trained model $\mathcal{N}(w,x,y)$. Denote by $\{(x_i,y_i)\}_{i=1,\ldots,N}$ the test data points. The first metric is defined to measure if the model solves the PDE:

$$\ell_1(w):=\sum\nolimits_{i=1}^{N}\frac{1}{N}(\Delta\mathcal{N}(w,x_i,y_i)+e^{2\mathcal{N}(w,x_i,y_i)})^2. \tag{3}$$

Another metric is defined to measure if the model coincides with the expected solution u_β as in (2):

$$\ell_2(w):=\sum\nolimits_{i=1}^{N}\frac{1}{N}(\mathcal{N}(w,x_i,y_i)-u_\beta(x_i,y_i))^2. \tag{4}$$

3.2 Experiment Results

Comparison to the ground-truth. In the training process, we define the total loss to be $0.4(\ell_v + \ell_{v'}) + 0.2\ell_u$. After training the weighted loss is 0.0114 while $\ell_u = 0.0133, \ell_v = 2.7209e-5, \ell_{v'} = 0.0218$. The test loss ℓ_1 is 0.0134 and $\ell_2 = 0.4973$. The heat map of the difference between the numerical solution and the real solution (2) is shown in Figure 1. We find that close to the origin there admits higher error due to the data precision issue and singular nature of (2), which are inevitable.

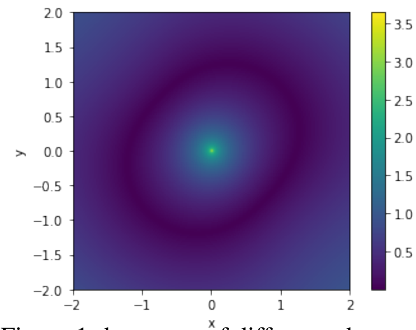

Figure 1: heat map of difference between numerical solution and real solution

Comparison to the baseline method. A comparison between our proposed method and the baseline method, which is introduced in the Appendix is shown in Table 1. By "using prior knowledge" we mean using a priori known solution (2) to add boundary conditions to (1). Existence of boundary condition is essential for using DeepXDE in the baseline method. The loss functions ℓ_1 as in (3) measures if the model solves the PDE and ℓ_2 as in (4) measures how close the model is to the real solution (2).

Method	Using prior knowledge or not	ℓ_1	ℓ_2
Baseline method (version 1)	Yes	2e-6	20.7779
Baseline method (version 2)	Yes	109.9671	0.2035
Proposed method	Yes	4e-7	6.6847
Proposed method	No	0.0134	0.4973

Table 1: Comparison between different methods in solving singular Liouville equation

Table 1 indicates that the baseline method can only capture either the smooth part or the singular part of the solution. Indeed, the baseline method tends to learn an almost constant function as the solution which is trivial for practical use.

In contrast, the proposed method is able to recover singular solutions, even without using a priori knowledge. Note that ℓ_2 in the last row of Table 1 is an average loss and the main contribution comes from test points close to the origin, which reflect the singular nature of (2) and is inevitable to some extent considering data precision limit.

Ablation study. In our proposed method, we could also use a priori knowledge. More precisely, one still trains $\mathcal{N}_u$ to solve (1), and to capture the singularity we only need to train $\mathcal{N}_v$ such that it approximates

$$\mathcal{N}_v \approx u_\beta - (\beta - 1)\log|z|$$

given the information from the solution (2).

The empirical results are shown in the third row in Table 1. We found that adding a priori knowledge does not help learn the singular behaviors of solutions. In other words, our proposed method without need of any a priori knowledge is flexible to capture the singularities in the solutions by dynamically learning their behaviors near singular points.

4 Conclusion

In this paper we proposed a deep learning method where we design multiple different neural networks to solve the singular Liouville equations, which are fully nonlinear, boundary conditions free but low-dimensional. Our method, to the best of our knowledge, is the first to treat PDEs that have continuous (not a jump) singularities. The proposed model outperforms existing readily available PDE solvers that employ deep neural networks when solving singular Liouville equations. The numerical solutions to the singular equation are precise enough to reflect the singular nature of the objective solution and can be further applied to the study of deformation of specific geometric shapes from the theoretical point of view.

References

[1] Christian Beck, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen. Solving the kolmogorov pde by means of deep learning. *Journal of Scientific Computing*, 88(3):1–28, 2021.

[2] Julius Berner, Markus Dablander, and Philipp Grohs. Numerically solving parametric families of high-dimensional kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33:16615–16627, 2020.

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[4] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, 5(4):349–380, 2017.

[5] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[6] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[7] Arnulf Jentzen, Diyora Salimova, and Timo Welti. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

[8] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[9] 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. *Journal of Computational Physics*, 378:686–707, 2019.

[10] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[11] Maziar Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

[12] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[13] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[14] Marc Troyanov. Metrics of constant curvature on a sphere with two conical singularities. *Differential geometry*, pages 296–306, 1989.

Appendix

In order to use existing package to solve (1), we "cheat" by using a priori known solution (2) to add boundary condition so that the original PDE is converted to a Dirichlet problem. Then we propose two approaches based on DeepXDE [8] to solve the new problems.

The first approach: We solve the following Dirichlet problem:

$$
\begin{aligned}
&\Delta u + e^{2u} = 0, \quad \text{in } \{z \in \mathbb{R}^2 : \epsilon < |z| < \delta\}, \\
&u = \frac{1}{2} \log \frac{4\beta^2 |z|^{2\beta-2}}{(1+|z|^{2\beta})^2}, \quad \text{on } \{z \in \mathbb{R}^2 : |z| = \epsilon \text{ or } \delta\}
\end{aligned}
\tag{5}
$$

using the package DeepXDE [8].

We expect to recover the solution u_β in (2) by solving (5) as we impose small/large enough ϵ and δ. We use ℓ_1 in (3) and ℓ_2 in (4) to test the accuracy of the model after training process.

The second approach: Instead of solving (1) for u, under the assumption of $m = 2$, we may reformulate the problem of solving (1) to another PDE for the function v_i. Indeed, in this case, globally there holds

$$u = v + \log |z|^{\beta-1}, \quad \text{in } \mathbb{R}^2,$$

where β is the cone angle (note that in this case the two cone angles at 0 and ∞ must be the same). The key point is that v is a smooth function. To solve u in (1) it is then enough to solve v.

Lemma .1. *When $m = 2$, the function v in* (1) *satisfies the following PDE:*

$$\Delta v + e^{2v} |z|^{2\beta-2} = 0, \quad \textit{in } \mathbb{R}^2. \tag{6}$$

Proof. From (1), we obtain

$$v = u - (\beta - 1) \log |z|.$$

Then taking Laplacian of v, we get

$$
\begin{aligned}
\Delta v &= \Delta u - \Delta(\beta - 1) \log |z| \\
&= \Delta u \\
&= -e^{2u}, \quad \text{using (1)} \\
&= -e^{2v} |z|^{2\beta-2}.
\end{aligned}
$$

□

Note that the PDE (6) has singularity since $2\beta - 2 < 0$. In other words, although v as a smooth function has better regularity compared to u, the PDE for v has a singular term, which does not happen for u.

To solve (6) using DeepXDE, we need to modify it to a Dirichlet problem. To achieve this, we solve v in the unit disk with the needed boundary condition:

$$
\begin{aligned}
\Delta v + e^{2v} |z|^{2\beta-2} = 0, &\quad \text{in the unit disk}, \\
v = \log \beta, &\quad \text{on the boundary}.
\end{aligned}
\tag{7}
$$

We only need to solve v in the unit disk due to the fact that it is enough to solve v for the two hemispheres, while each hemisphere corresponds to the unit disk under stereographic projection. The boundary condition in (7) comes from the prior knowledge of u_β in (2), which serves as the solution to (1) when $m = 2$.

We also use ℓ_1 in (3) as the test loss. Besides, for this approach we modify ℓ_2 to the average L^2 difference between the model $\mathcal{N}_v$ and the real solution v_β:

$$\ell_2 = \ell(\mathcal{N}_v, v_\beta) := \frac{1}{N} \sum_{i=1}^{N} |\mathcal{N}_v(x_i, y_i) - v_\beta(x_i, y_i)|^2,$$

for test data set $\{(x_i, y_i)\}_{1 \le i \le N}$.