AMSC664 Final Presentation

Exploration on Applications of Machine Learning Methods in Approximating Parameter-Dependent Partial Differential Equations

Jiajing Guan
Advisor: Howard Elman

University of Maryland

May 4, 2021

Overview

Problem Definition

Let $u(\boldsymbol{x}, t; \boldsymbol{\mu})$ be the exact solution to the following parameter-dependent PDE problem:

$$u_t + \mathcal{N}[u; \boldsymbol{\mu}] = 0,$$

where $\boldsymbol{\mu}$ denote parameters. Let $f(u(\boldsymbol{x}, t; \boldsymbol{\mu})) := u_t + \mathcal{N}[u; \boldsymbol{\mu}]$.

We would like to obtain approximations $u_{nn}(\boldsymbol{x}, t; \boldsymbol{\mu})$ produced by trained neural networks at discrete points $\{(\boldsymbol{x}_{test}, t_{test})^{(j)}\}_{j=1}^{N_{grid}}$ and set of parameters $\{\boldsymbol{\mu}_{test}^{(i)}\}_{i=1}^{N_{test}}$.

Project Definition

Goal

Studying existing machine learning methods that approximate solutions to parameter-dependent PDEs.

Existing Methods:

- Non-intrusive reduced order modeling of nonlinear problems using neural networks[1]
- Physics-Informed Neural Networks[2]

[1] J. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks," *Journal of Computational Physics*, vol. 363, pp. 55 –78, 2018, ISSN: 0021-9991.

[2] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686 –707, 2019, ISSN: 0021-9991.

POD-NN RB [1]

Procedure:

1. sample N_{train} number of parameters $\{\boldsymbol{\mu}_{train}^{(j)}\}_{j=1}^{N_{train}}$;
2. compute a collection of snapshots $\{\boldsymbol{u}_h(\boldsymbol{\mu}_{train}^{(j)})\}_{j=1}^{N_{train}}$ at $\{(\boldsymbol{x}_{test}, t_{test})^{(j)}\}_{j=1}^{N_{grid}}$ using traditional solvers such as FDM;
3. using proper orthogonal decomposition (POD) Galerkin Reduced Basis (RB) method, create change of basis matrix V from left singular vectors of snapshots. Need to choose dimension of reduced basis L.
4. produce sample outputs $\{V^T\boldsymbol{u}_h(\boldsymbol{\mu}_{train}^{(j)})\}_{j=1}^{N_{train}}$ and train neural network with loss function

$$L(\boldsymbol{\theta}) = \frac{1}{2N_{train}} \sum_{j=1}^{N_{train}} \left\| \boldsymbol{u}_{nn}(\boldsymbol{\mu}_{train}^{(j)}; \boldsymbol{\theta}) - V^T \boldsymbol{u}_h(\boldsymbol{\mu}_{train}^{(j)}) \right\|_2^2,$$

   where $\boldsymbol{u}_{nn}(\boldsymbol{\mu}; \boldsymbol{\theta})$ is output of the neural network given parameter $\boldsymbol{\mu}$ and weights $\boldsymbol{\theta}$.
5. produce approximations with parameter set $\{\boldsymbol{\mu}_{test}^{(i)}\}_{i=1}^{N_{test}}$ as $\{V\boldsymbol{u}_{nn}(\boldsymbol{\mu}_{test}^{(i)}; \boldsymbol{\theta})\}_{i=1}^{N_{test}}$.

PINN [2]

Procedure:

1. sample over initial and boundary training data as $\{(\boldsymbol{x}_{IB}, t_{IB}, \boldsymbol{\mu}_{IB}, u_{IB})^{(j)}\}_{j=1}^{N_i}$ and points in the domain as $\{(\boldsymbol{x}_F, t_F, \boldsymbol{\mu}_F)^{(z)}\}_{z=1}^{N_f}$.
2. train neural network with the following loss function:

$$L(\boldsymbol{\theta}) = \frac{1}{2N_i}\sum_{j=1}^{N_i}(u_{nn}(\boldsymbol{x}_{IB}^{(j)}, t_{IB}^{(j)}, \boldsymbol{\mu}_{IB}^{(j)}; \boldsymbol{\theta}) - u_{IB}^{(j)})^2 + \frac{1}{2N_f}\sum_{z=1}^{N_f}(f(u_{nn}(\boldsymbol{x}_F^{(z)}, t_F^{(z)}, \boldsymbol{\mu}_F^{(z)}; \boldsymbol{\theta})))^2$$

3. produce approximations at discrete points $\{(\boldsymbol{x}_{test}, t_{test})^{(j)}\}_{j=1}^{N_{grid}}$ and parameter set $\{\boldsymbol{\mu}_{test}^{(i)}\}_{i=1}^{N_{test}}$ as $\{\{u_{nn}(\boldsymbol{x}_{test}^{(j)}, t_{test}^{(j)}, \boldsymbol{\mu}_{test}^{(i)}; \boldsymbol{\theta})\}_{j=1}^{N_{grid}}\}_{i=1}^{N_{test}}$.

What We Set Out to Do

Original Goal

Test the influence of following factors on the performance of POD-NN RB and PINN on parameter-dependent PDE problems:

- network depth
- network width
- network structure
- number of samples
- type of problems

Good News

We used an unsteady Burger's equation and a nonlinear diffusion equation as our test problems. We found:

- Network depth, width, network structure do not influence the performance of POD-NN RB significantly;
- POD-NN RB performs similarly if sufficient samples are given.

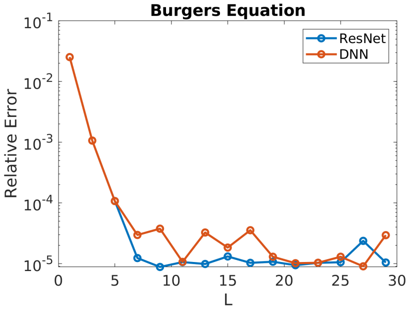


Figure: This figure shows that the network structure (Dense Neural Network or ResNet) does not influence the performance of POD-NN RB on the Burger's equation. The network were trained with Levenberg-Marquardt (LM), as were all the following results.

Problems Encountered

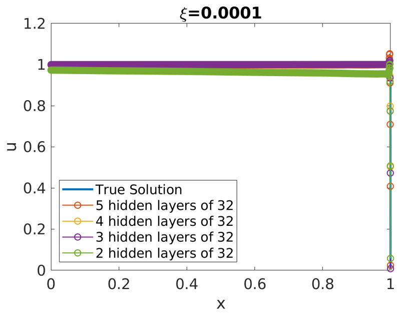
Issue 1:

- PINN unable to produce good approximation results as the parameters approach their asymptotes in a simple 1D Convection-Diffusion equation problem:

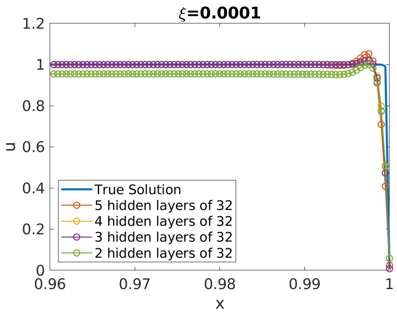
$$\begin{aligned} -\xi u^{''} + u^{'} &= 0 \quad \text{for } x \in (0,1) \\ u(0) &= 1 - e^{-1/\epsilon} \\ u(1) &= 0 \end{aligned} \tag{1}$$

  Here, the parameter is $\epsilon = 10^a$, where a is chosen from a uniform distribution of $[-4, 0]$.

Issue 1



(a)



(b)

Figure: 2a shows the approximations obtained by PINN on Equation 1 with a DNN of different depth. 2b shows the same figure zoomed in on $x \in [0.96, 1]$.

Problems Encountered

Issue 2:

- Unpredictable behavior of PINN

Number of hidden layers	Final loss value	Runtime	Relative Error
2	7.036142e-03	567.494288	0.12503777
3	702.5287	1022.886320	7.454208
4	665.6986	2154.073805	5.666144
5	1.140639e-02	1752.942492	0.14456806

Table: This table shows the effects of depth of a DNN on PINN solving a nonlinear diffusion equation. The networks were trained using LM.

$\Longrightarrow$ **narrow down the scope of the problem and understand the performance of PINN on parameter-dependent PDE problems with DNN.**

Convection-Diffusion Equation

Without loss of generality, one-dimension Convection-Diffusion equations can be modeled as[3]:

$$\begin{aligned} Lu &:= -\epsilon u'' + b(x)u' + c(x)u = f(x), \quad \text{for } x \in (0,1), \\ u(0) &= u(1) = 0, \quad \text{with } c(x) \geq 0 \text{ for } x \in [0,1]. \end{aligned} \tag{2}$$

where u'' corresponds to diffusion, u' represents convection, u is a source term and f is a driving term.

Singularly Perturbed

Differential equations (ordinary or partial) that depend on a small positive parameter ϵ and whose solutions (or their derivatives) approach a discontinuous limit as ϵ approaches zero. Such problems are said to be singularly perturbed, where ϵ is a perturbation parameter.

[3] H.-G. Roos, M. Stynes, and L. Tobiska, *Robust numerical methods for singularly perturbed differential equations: convection-diffusion-reaction and flow problems*. Springer Science & Business Media, 2008, vol. 24.

Original Problem

$$-\epsilon u'' + u' = 1, \quad x \in (0,1) \tag{3}$$

with Dirichlet boundary conditions $u(0) = u(1) = 0$ and $\epsilon \in [10^{-4}, 1]$. The exact solution to Equation (3) is:

$$u_{ex}(x) = x - \frac{\exp(-\frac{1-x}{\epsilon}) - \exp(-\frac{1}{\epsilon})}{1 - \exp(-\frac{1}{\epsilon})} \tag{4}$$

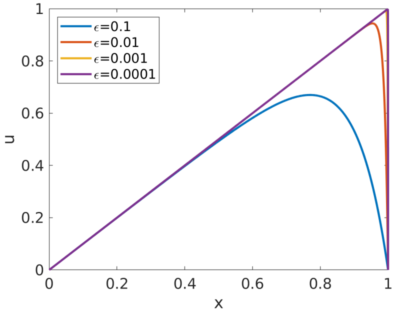
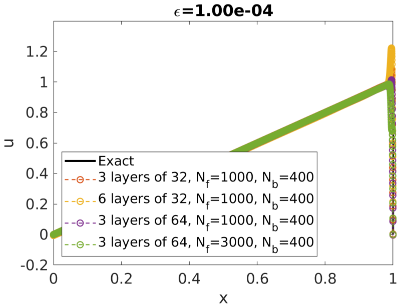


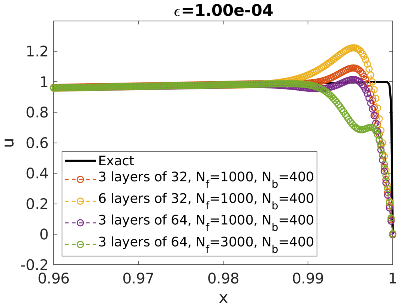
Figure: This figure shows the exact solution $u_{ex}(x)$ to Equation (3) for various ϵ values.

Results without Transformation

Let the residual function be $f(u) = -\epsilon u'' + u' - 1$. How good of a solution can we get?



(a)



(b)

Figure: 4a shows the approximation done by the neural net of different configurations compared to the exact solution. 4b is the same figure to their left, zoomed in on $x \in [0.96, 1]$

Transformed Problem

The linear transformation is $\xi = \frac{1-x}{\epsilon}$. Suppose $v(\xi) = u(x)$. With this change of variable, the original PDE problem becomes:

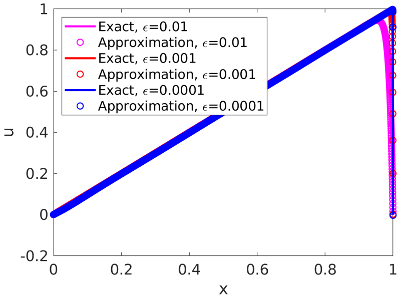
$$-\frac{1}{\epsilon}\frac{d^2 v}{d\xi^2} - \frac{1}{\epsilon}\frac{dv}{d\xi} = 1, \quad \xi \in (0, \frac{1}{\epsilon}) \tag{5}$$

with Dirichlet boundary conditions $v(0) = v(1/\epsilon) = 0$. The exact solution to Equation (5) is:

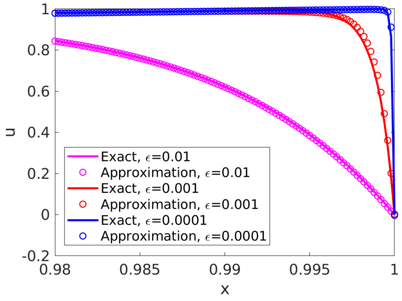
$$v_{ex}(\xi) = 1 - \epsilon\xi - \frac{\exp(-\xi) - \exp(-\frac{1}{\epsilon})}{1 - \exp(-\frac{1}{\epsilon})} \tag{6}$$

Transformed Problem Result

Let the residual function be $f(v) = (\frac{d^2 v}{d\xi^2} + \frac{dv}{d\xi} + \epsilon)/\epsilon$.



(a)



(b)

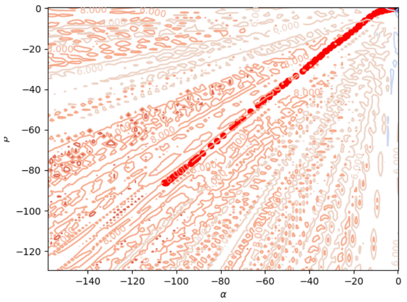
Figure: 5a shows the approximation done by the neural net (with 3 hidden layers, 32 nodes per layer) compared to the exact solution with $N_f = 1000$ and $N_b = 400$. 5b is the same figure to their left, zoomed in on $x \in [0.98, 1]$

Visualization of Optimization Trajectory

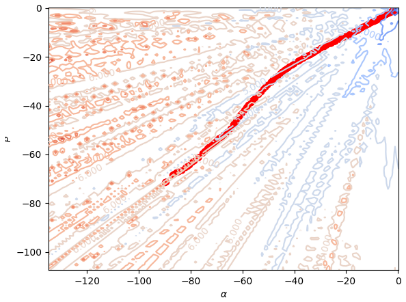
1. Store the weights at each training epoch i, denoted as θ_i;
2. Suppose the last epoch is N. Construct the matrix $[\theta_0 - \theta_N; \theta_1 - \theta_N; \dots; \theta_{N-1} - \theta_N]$ and apply PCA to this matrix;
3. Select the two vectors corresponding to the two largest singular values and treat them as d_1 and d_2;
4. Compute the projection of $\theta_i - \theta_N$, for $i = 0, \dots, N$, onto the two directions and store them as the trajectory points of training;
5. Create a mesh that include the min and max of the trajectory points and visualize the loss contour with trajectory points plotted.

Effects of Residual Function - Trajectory

Problems are too complicated to obtain useful information from the loss landscape.



(a)



(b)

Figure: 6a shows the loss landscape created with PCA selected directions of the original problem and its optimization trajectory. 6b shows the same graph for the transformed problem.

A General Transformation

One question one might ask is that: How essential is the boundary layer location in the transformation? Let $\xi = \frac{a-x}{\epsilon}$, for $a \in [0, 1]$. Then the original problem becomes:

$$-\frac{1}{\epsilon}\frac{d^2v}{d\xi^2} - \frac{1}{\epsilon}\frac{dv}{d\xi} = 1, \quad \xi \in (\frac{a-1}{\epsilon}, \frac{a}{\epsilon}) \tag{7}$$

with Dirichlet boundary conditions $v(\frac{a-1}{\epsilon}) = v(\frac{a}{\epsilon}) = 0$. The exact solution to Equation (7) is:

$$v_{ex}(\xi) = a - \epsilon\xi - \frac{\exp(-\xi - \frac{1-a}{\epsilon}) - \exp(-\frac{1}{\epsilon})}{1 - \exp(-\frac{1}{\epsilon})} \tag{8}$$

General Transformation Results

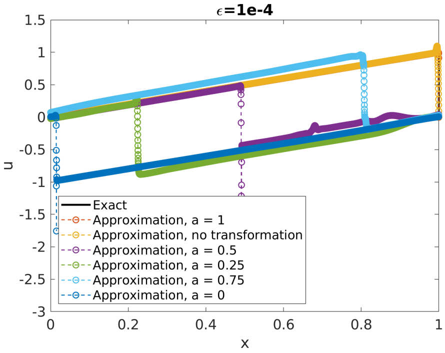


Figure: This figure shows the approximation done by the neural net with transformation $\xi = (a - x)/\epsilon$, for $a = 0, 0.25, 0.5, 0.75, 1$. All approximations are generated from a DNN with 3 hidden layers, 32 nodes per layer. They are trained with 1000 residual samples a

Sample Distribution

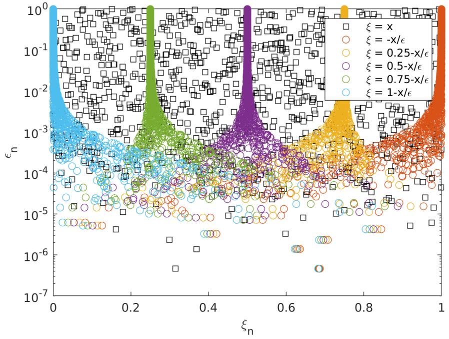
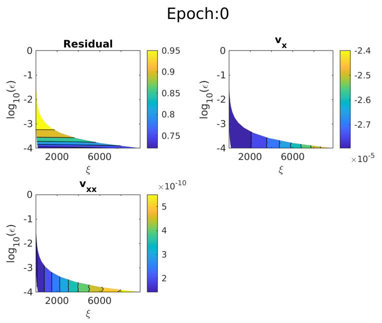
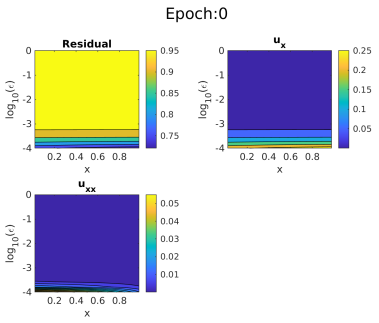


Figure: This figure shows the normalized distribution of samples of transformations $\xi = (a - x)/\epsilon$, for $a = 0, 0.25, 0.5, 0.75, 1$. Here, both ξ and ϵ are normalized using the formula $\text{Input}_{\text{Normalized}} = \frac{\text{Input} - \text{Lower Bound}}{\text{Upper Bound} - \text{Lower Bound}}$

Change of Residual During Training: $a = 1$



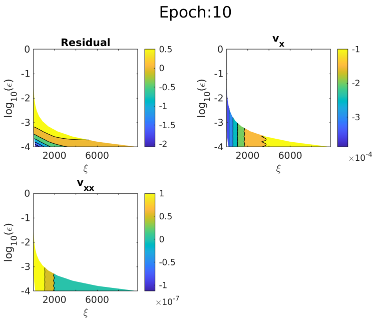
(a)



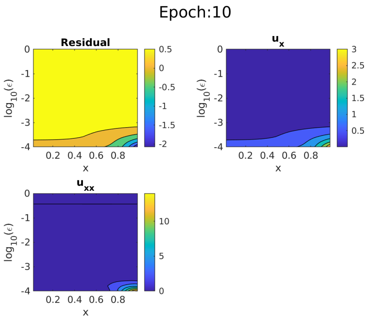
(b)

Figure: 9a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 0, i.e. the initialization of weights. 9b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 0.

Change of Residual During Training: $a = 1$



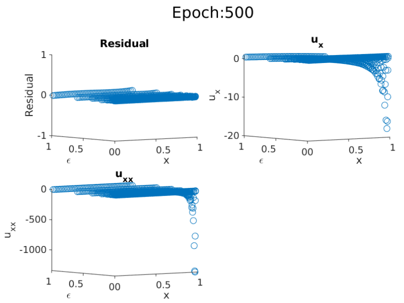
(a)



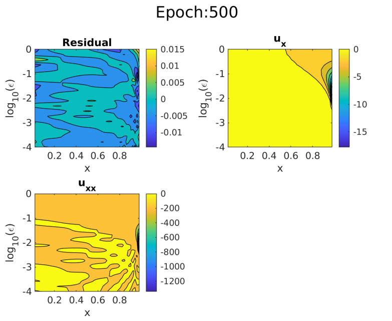
(b)

Figure: 10a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 10. 10b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 10.

Change of Residual During Training: $a = 1$



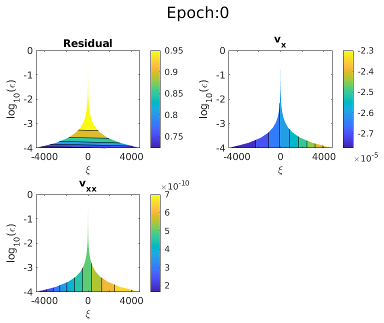
(a)



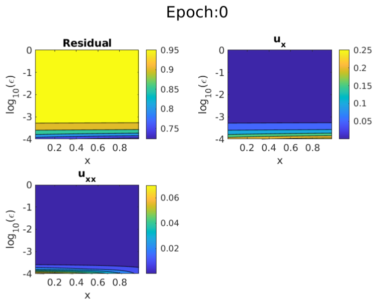
(b)

Figure: 11b shows the contour of residual value (upper left), the contour of $\frac{du}{dx}$ (upper right) and the contour of $\frac{d^2u}{dx^2}$ (bottom left) at epoch 500. 11a shows the 3D plots of the contours to illustrate details missing from the contour plots.

Change of Residual During Training: $a = 0.5$



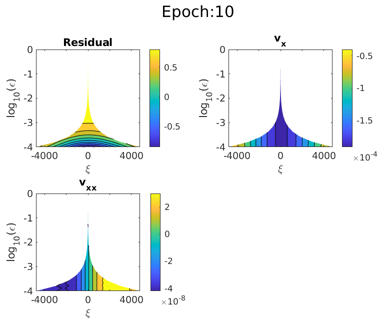
(a)



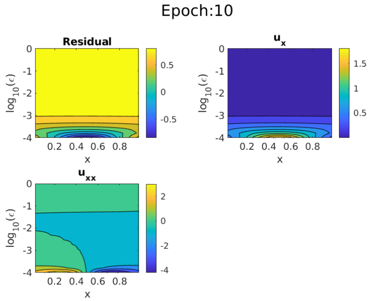
(b)

Figure: 12a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 0, i.e. the initialization of weights. 12b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 0.

Change of Residual During Training: $a = 0.5$



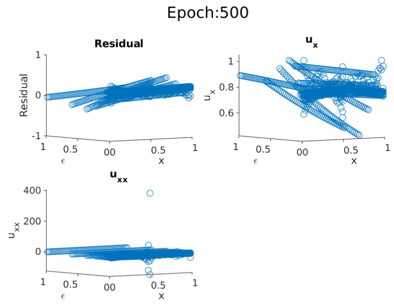
(a)



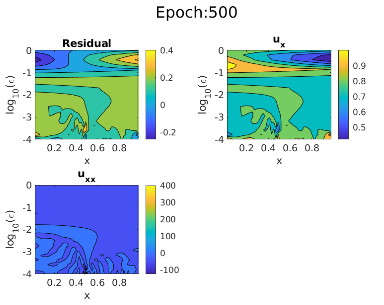
(b)

Figure: 13a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 10. 13b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 10.

Change of Residual During Training: $a = 0.5$



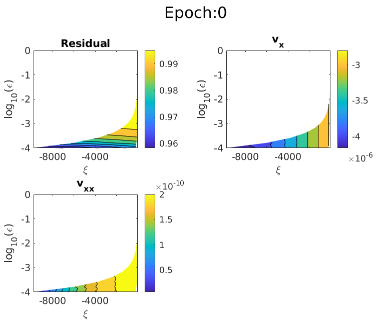
(a)



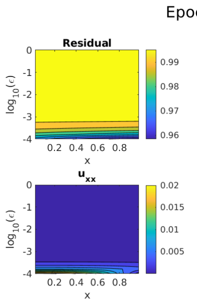
(b)

Figure: 14b shows the contour of residual value (upper left), the contour of $\frac{du}{dx}$ (upper right) and the contour of $\frac{d^2u}{dx^2}$ (bottom left) at epoch 500. 14a shows the 3D plots of the contours to illustrate details missing from the contour plots.

Change of Residual During Training: $a = 0$



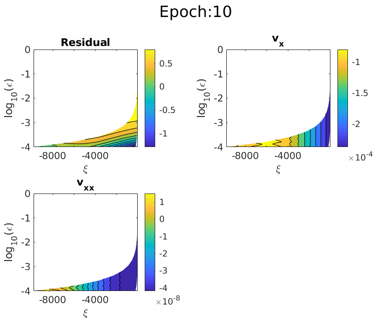
(a)



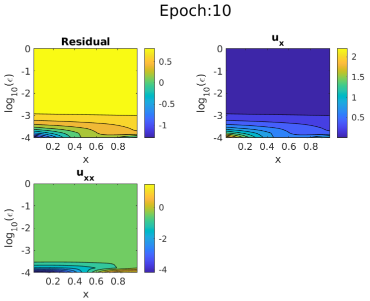
(b)

Figure: 15a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 0, i.e. the initialization of weights. 15b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 0.

Change of Residual During Training: $a = 0$



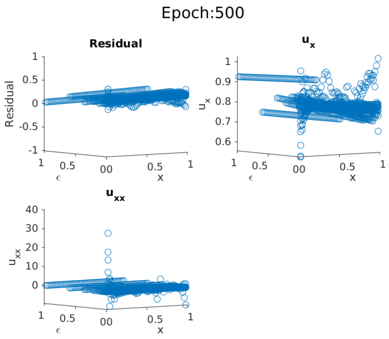
(a)



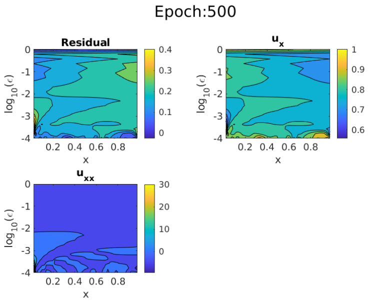
(b)

Figure: 16a shows the contour of residual value (upper left), the contour of $\frac{dv}{d\xi}$ (upper right) and the contour of $\frac{d^2v}{d\xi^2}$ (bottom left) at epoch 10. 16b shows the same graph in (x, ϵ) space. It shows the contours of the residual, $\frac{du}{dx}$ and $\frac{d^2u}{dx^2}$ at epoch 10.

Change of Residual During Training: $a = 0$



(a)



(b)

Figure: 17b shows the contour of residual value (upper left), the contour of $\frac{du}{dx}$ (upper right) and the contour of $\frac{d^2u}{dx^2}$ (bottom left) at epoch 500. 17a shows the 3D plots of the contours to illustrate details missing from the contour plots.

Explanation

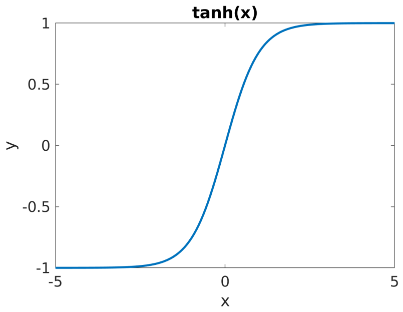


Figure: This figure shows the graph of $\tanh(x)$.

Derivative of $\tanh x$ is the largest when x is around $0 \implies$ one wants to ensure that the boundary layer is transformed to have input values around 0.

Revisit the Old Problem

$$
\begin{aligned}
-\xi u^{''} + u^{'} &= 0 \quad \text{for } x \in (0,1) \\
u(0) &= 1 - e^{-1/\epsilon} \qquad (9) \\
u(1) &= 0
\end{aligned}
$$

Here, the parameter is $\epsilon = 10^a$, where a is chosen from a uniform distribution of $[-4, 0]$. We could compute the exact analytical solution to be $u(x) = 1 - e^{(x-1)/\epsilon}$.

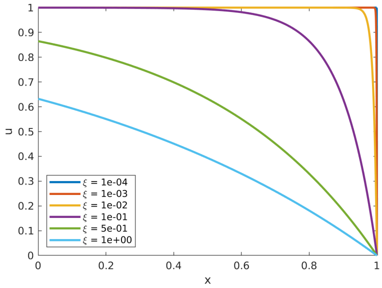


Figure: This figure shows how solutions change as ϵ increases from 10^{-4} to 1.

Transformed Problem

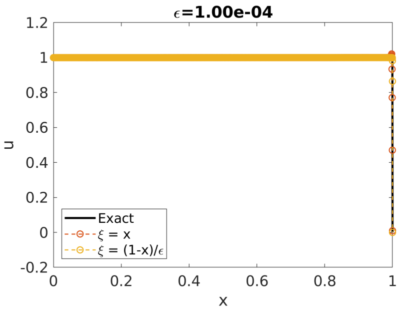
Suppose $v(\xi) = u(x)$. With $\xi = (1-x)/\epsilon$, the original PDE problem becomes:

$$\begin{aligned} -\frac{1}{\epsilon}\frac{d^2 v}{d\xi^2} - \frac{1}{\epsilon}\frac{dv}{d\xi} &= 0 \quad \text{for } \xi \in (0, \frac{1}{\epsilon}) \\ v(0) &= 0 \\ v(1/\epsilon) &= 1 - e^{-1/\epsilon} \end{aligned} \tag{10}$$

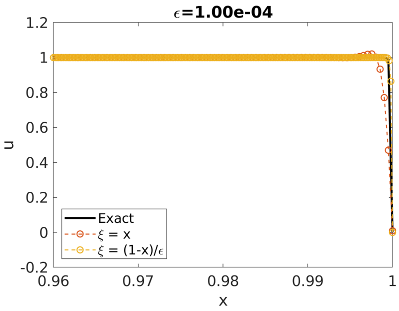
The exact solution is:

$$v_{ex}(\xi) = 1 - \exp(-\xi) \tag{11}$$

Transformed Results



(a)



(b)

Figure: 20a shows the approximation produced from the original problem, the transformed problem and the exact solution side-by-side. 20b shows the same figure zoomed in on $x \in [0.96, 1]$. Results above are trained using a DNN of 3 hidden layers with 32 nodes per layer, 1000 residual samples and 400 boundary samples.

2D Convection-Diffusion Equation

Now we will test if our theory holds for 2D problems. We grabbed this problem from[4].

$$
\begin{aligned}
&-\epsilon(u_{xx}+u_{yy})+u_y=0, \quad (x,y)\in(-1,1)\times(-1,1)\\
&u(-1,y)\approx -1, \quad u(1,y)\approx 1\\
&u(x,-1)=x, \quad u(x,1)=0
\end{aligned} \tag{12}
$$

Here $\epsilon \in [10^{-4}, 1]$. The exact solution is:

$$
u_{ex}(x,y)=x\left(\frac{1-\exp((y-1)/\epsilon)}{1-\exp(-2/\epsilon)}\right) \tag{13}
$$

[4] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scie, 2014.

Exact Solution

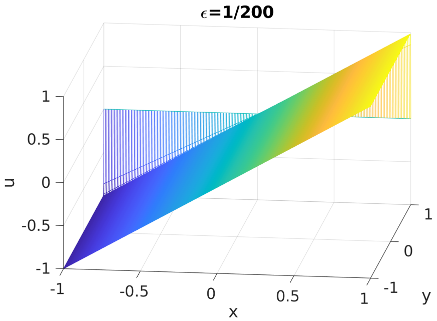
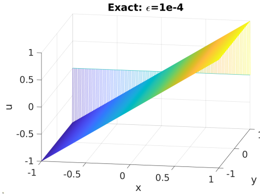
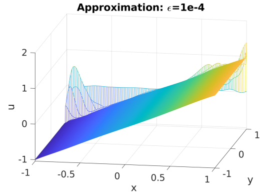


Figure: This figure shows the exact solution $u_{ex}(x)$ to the 2D Convection-Diffusion equation for $\epsilon = 1/200$.

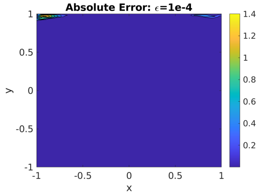
Results without Transformation



(a)



(b)



(c)

Figure: 22a shows the exact solution when $\epsilon = 1e-4$. 22b shows the approximation produced by PINN with a DNN of 3 hidden layers, 32 nodes per layer, 2000 residual samples and 400 boundary samples. 22c shows the contour plot of the absolute error between the exact solution and the approximation.

Transformed Problem

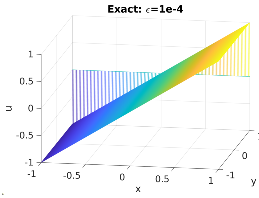
Let $v(\xi,\eta) = u(x,y)$. Suppose $\xi = x$ and $\eta = (1-y)/\epsilon$, we then get:

$$\begin{aligned}
&-\epsilon v_{\xi\xi} - (v_{\eta\eta} + v_\eta)/\epsilon = 0, \quad (\xi,\eta) \in (-1,1) \times (0, 2/\epsilon) \\
&v(-1,\eta) \approx -1, \quad v(1,\eta) \approx 1 \\
&v(\xi, 2/\epsilon) = \xi, \quad v(\xi, 0) = 0
\end{aligned}$$

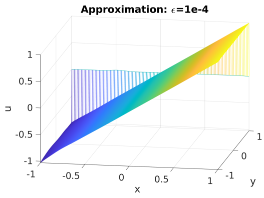
Here $\epsilon \in [10^{-4}, 1]$. The exact solution is:

$$v_{ex}(\xi,\eta) = \xi \left(\frac{1 - \exp(-\eta)}{1 - \exp(-2/\epsilon)} \right) \tag{14}$$

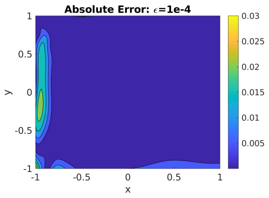
Results with Transformation



(a)



(b)

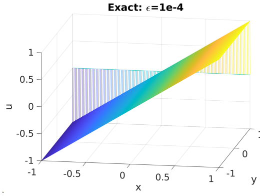


(c)

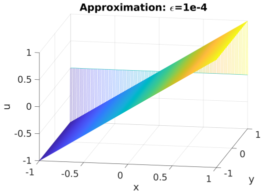
Figure: 23a shows the exact solution when $\epsilon = 1e - 4$. 23b shows the approximation produced by PINN with a DNN of 3 hidden layers, 32 nodes per layer, 2000 residual samples and 400 boundary samples on the transformed problem. 23c shows the contour plot of the absolute error between the exact solution and the approximation.

Can we do better?

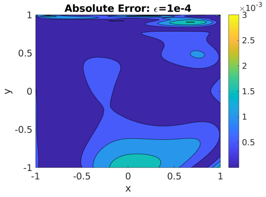
Now increase the number of residual samples from 2000 to 8000:



(a)



(b)



(c)

Figure: 24a shows the exact solution when $\epsilon = 1e - 4$. 24b shows the approximation produced by PINN with a DNN of 3 hidden layers, 32 nodes per layer, 8000 residual samples and 400 boundary samples on the transformed problem. 24c shows the contour plot of the absolute error between the exact solution and the approximation.

1D Convection-Diffusion Equation: Neumann Boundary

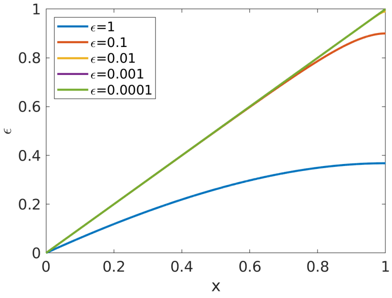
What if we have zero-flux boundary condition at $x = 1$? We then have the following governing equation:

$$-\epsilon u'' + u' = 1, \quad x \in (0,1) \tag{15}$$

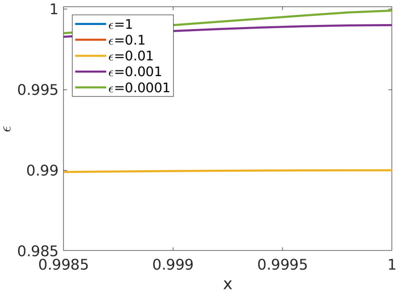
with Dirichlet boundary condition $u(0) = 0$, Neumann boundary condition $u'(1) = 0$ and $\epsilon \in [10^{-4}, 1]$. The exact solution to Equation (15) is:

$$u_{ex}(x) = x - \epsilon[\exp(-(1-x)/\epsilon) - \exp(-1/\epsilon)]. \tag{16}$$

Characteristics of the Exact Solution



(a)



(b)

Figure: 25a shows how solutions to the Neumann boundary problem change as ϵ increases from 10^{-4} to 1. 25b shows the same figure zoomed in on $x \in [0.9985, 1]$.

Transformed Problem

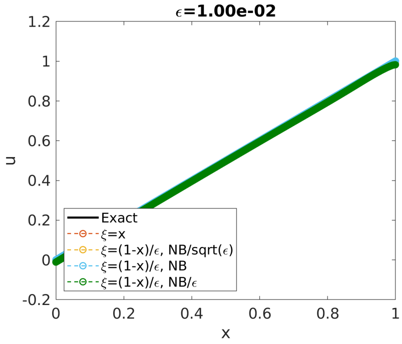
Let $v(\xi) = u(x)$. Suppose $\xi = (1-x)/\epsilon$. Then we have:

$$-\frac{1}{\epsilon}\frac{d^2v}{d\xi^2} - \frac{1}{\epsilon}\frac{dv}{d\xi} = 1, \quad \xi \in (0, \frac{1}{\epsilon}) \tag{17}$$

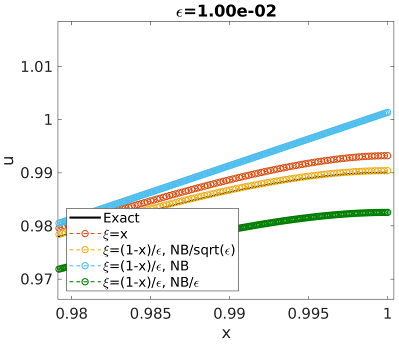
with Dirichlet boundary condition $v(1/\epsilon) = 0$ and Neumann boundary condition $\frac{dv}{\epsilon d\xi}(0) = 0$. The exact solution is:

$$v_{ex}(\xi) = 1 - \epsilon\xi - \epsilon[\exp(-\xi) - \exp(-1/\epsilon)] \tag{18}$$

Results



(a)



(b)

Figure: 26a compares the approximation obtained using the original problem ($\xi = x$), using the transformed problem ($\xi = (1-x)/\epsilon$), and the exact solution. 26b shows the same figure zoomed in on $x \in [0.98, 1]$. Here NB=$\frac{dv}{d\xi}$. The different "NB" options denotes different weightings of the Neumann boundary during training. All approximations results are produced from a DNN with 3 hidden layers, 32 nodes per layer, 1000 residual samples, 200 Dirichlet boundary and 200 Neumann boundary samples.

Possible Explanation

Why would Neumann boundary condition being $\frac{dv}{\sqrt{\epsilon}d\xi}$ obtain the best result?

First, one needs to recognize that the width of the boundary layer at $x = 1$ is of $O(\sqrt{\epsilon})$[4], instead of $O(\epsilon)$ like the Dirichlet boundaries. Thus, the stretching factor at the Neumann boundary should be $\sqrt{\epsilon}$, instead of ϵ.

Future Directions

- Study effects of different factors on PINN using the transformed problems to ensure the approximations are accurate;
- Study singularly perturbed problems with more than one boundary layers;
- Study singularly perturbed problems with more than one perturbation parameter;
- Adjust PINN automatically without knowing the characteristics of the problem.

References I

[1] J. Hesthaven and S. Ubbiali, "Non-intrusive reduced order modeling of nonlinear problems using neural networks," *Journal of Computational Physics*, vol. 363, pp. 55 –78, 2018, ISSN: 0021-9991.

[2] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686 –707, 2019, ISSN: 0021-9991.

[3] H.-G. Roos, M. Stynes, and L. Tobiska, *Robust numerical methods for singularly perturbed differential equations: convection-diffusion-reaction and flow problems*. Springer Science & Business Media, 2008, vol. 24.

[4] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scie, 2014.

[5] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[6] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314.

References II

[7] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[8] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.

[9] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993. [Online]. Available: `https://www.aclweb.org/anthology/J93-2003`.

[10] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[11] O. Le Maître and O. Knio, *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics*. Jan. 2010, ISBN: ISBN-10: 9048135192. DOI: `10.1007/978-90-481-3520-2`.

[12] K. Q. Ye, "Orthogonal column latin hypercubes and their application in computer experiments," *Journal of the American Statistical Association*, vol. 93, no. 444, pp. 1430–1439, 1998, ISSN: 01621459. [Online]. Available: `http://www.jstor.org/stable/2670057`.

References III

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[14] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[15] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *arXiv preprint arXiv:1712.09913*, 2017.

Test Problems

- Unsteady Burger's Equations

$$\begin{cases} u_t + uu_x = \xi u_{xx}, & \text{for } x \in [-1,1],\ t \in [0,1] \\ u(0,x) = -\sin(\pi x), & \\ u(t,-1) = u(t,1) = 0 & \end{cases},$$

where $\xi = 10^p$, where p is sampled on an uniform distribution of $[-3,-2]$.

Burgers' Equation Solutions

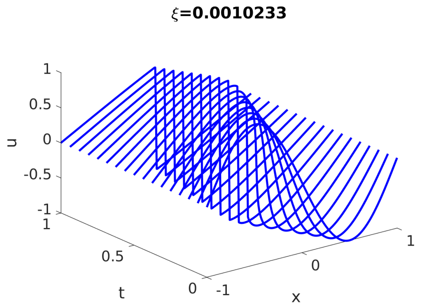


Figure: This figure shows the solution to the inviscous Burgers' equation when $\xi = 0.0010233$

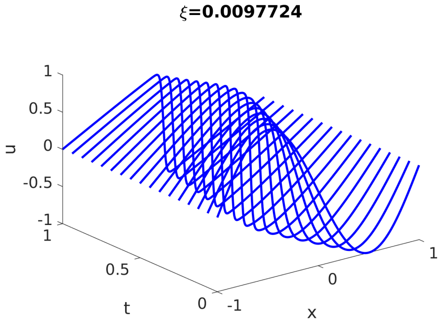


Figure: This figure shows the solution to the inviscous Burgers' equation when $\xi = 0.0097724$.

Test Problems

- Nonlinear Diffusion Equation

$$\begin{cases} -(exp(u(x;\boldsymbol{\xi}))u(x;\boldsymbol{\xi})')' = s(x;\boldsymbol{\xi}), & \text{for } x \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ u(\pm\pi/2;\boldsymbol{\xi}) = \xi_2 \sin(2 \pm \frac{\xi_1 \pi}{2}) exp(\pm\frac{\xi_3 \pi}{2}) \end{cases}$$

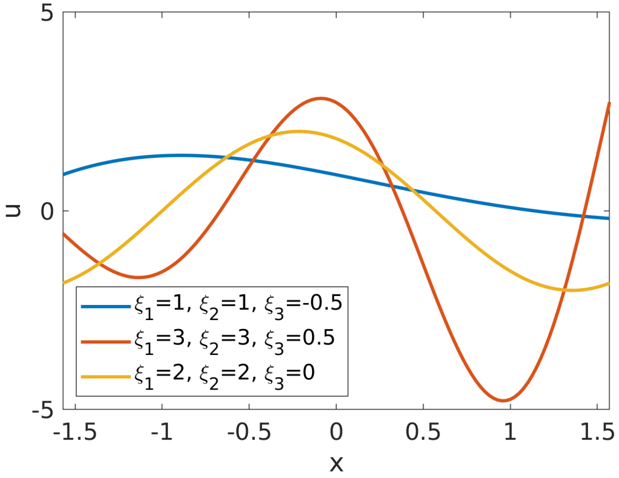
where $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$ are sampled on uniform distribution of $[1,3] \times [1,3] \times [-0.5, 0.5]$ and

$$\begin{aligned} s(x;\boldsymbol{\xi}) = &- \xi_2 exp(\xi_2 \sin(2+\xi_1 x) exp(\xi_3 x) + \xi_3 x) \\ &* [2\xi_1\xi_3 \cos(2+\xi_1 x) + (\xi_3^2 - \xi_1^2)\sin(2+\xi_1 x) \\ &+ exp(\xi_3 x)[\xi_1 \cos(2+\xi_1 x) + \xi_3 \sin(2+\xi_1 x)]^2] \end{aligned}$$

Here, $s(x;\boldsymbol{\xi})$ is calculated such that the exact solution is

$$u_{ex}(x;\boldsymbol{\xi}) = \xi_2 \sin(2+\xi_1 x) exp(\xi_3 x)$$

Diffusion Equation Solutions



Importance of Stretching Factor

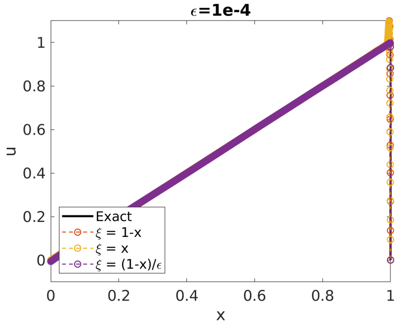
What if for the 1D Convection-Diffusion equation with Dirichlet problem, we have $\xi = 1 - x$? Then the governing equation will become the following equation:

$$-\epsilon \frac{d^2 v}{d\xi^2} - \frac{dv}{d\xi} = 1, \quad v \in (0, 1/\epsilon) \tag{19}$$

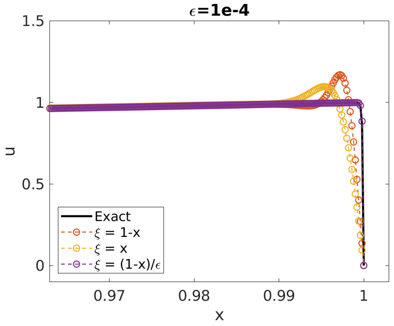
with Dirichlet boundary conditions $v(0) = v(1/\epsilon) = 0$ and $\epsilon \in [10^{-4}, 1]$. The exact solution to Equation (19) is:

$$v_{ex}(\xi) = 1 - \xi - \frac{\exp(-\frac{-\xi}{\epsilon}) - \exp(-\frac{1}{\epsilon})}{1 - \exp(-\frac{1}{\epsilon})} \tag{20}$$

Results



(a)



(b)

Figure: 29a compares the approximation obtained using the original problem ($\xi = x$), using the transformed problem ($\xi = (1-x)/\epsilon, \xi = 1-x$), and the exact solution. 29b shows the same figure zoomed in on $x \in [0.96, 1]$. All approximations results are produced from a DNN with 3 hidden layers, 32 nodes per layer, 1000 residual samples, 400 Dirichlet boundary samples.