Numerical Experiment

1 Neural Network Models

1.1 General Methodology

- We define the deep hidden physics model to be:

$$f := u_t - N(t, x, u, u_x, u_{xx}, ...) \tag{1}$$

- We obtain the derivatives of the neural network u with respect to time t and space x by applying the chain rule for differentiating compositions of functions using automatic differentiation

- Parameters of the neural networks u and N neural network can be learned by minimizing the sum of squared errors:

$$\sum_{i=1}^{N}(|u(t_i, x_i) - u_i|^2 + |f(t_i, x_i)|^2) + \lambda R(\theta) \tag{2}$$

where the set $\{t_i, x_i, u_i\}$ is the training data on u and $R(\theta)$ is the regularization (in numerical experiment we use L_1 regularization) and λ is a penalized constant for regularization (we choose $\lambda = 0.001$).

1.2 Inviscid Burgers

- The invsicid Burgers PDE equation is of the form:

$$u_t + (\frac{u^2}{2})_x = 0 \tag{3}$$

  In the domain $(x, t) \in [-8, 8] \times [0, 2]$

- The first initial condition we consider is $u_0(x) = -\sin\left(\frac{\pi x}{8}\right)$ and the second initial condition we consider is $u_0(x) = \cos\left(\frac{-\pi x}{8}\right)$.

- General description of the problem in neural network: the nerual network of inviscid Burgers equation:

$$u_t = \mathrm{N}(u, u_x) \tag{4}$$

- We represent the solution u by a 5-layer deep neural network with 50 neurons per hidden layer. Furthermore, we let N to be a neural network with 2 hidden layers and 100 neurons per hidden layer and in the input layer this N takes input of 2.

- Out of this data-set (generated by chebfun in Matlab), we generate a smaller training subset, scattered in space and time, by randomly sub-sampling 10000 data points from time t = 0 to t = 1.8. The rest of the domain from time t = 1.8 to the final time t = 2.0 will be referred to as the test portion. Given the training data, we are interested in learning N as a function of the solution u and its derivatives up to the 2nd order.

1.3 kdV Equation

- The kdV equation is of the form:

$$u_t + uu_x + u_{xxx} = 0 \tag{5}$$

  in the domain $(x, t) \in [-8, 8] \times [0, 5]$

- The first initial condition we concern is $u_0(x) = -\sin\left(\frac{\pi x}{8}\right)$ and the second initial condition we consider is $u_0(x) = \cos\left(\frac{-\pi x}{8}\right)$.

$$u_t = \mathrm{N}(u, u_x, u_{xx}, u_{xxx}) \tag{6}$$

- We represent the solution u by a 5-layer deep neural network with 50 neurons per hidden layer. Furthermore, we let N to be a neural network with 2 hidden layers and 100 neurons per hidden layer. These two networks are trained by minimizing the sum of squared errors loss of equation

- Out of this data-set (generated by chebfun in Matlab), we generate a smaller training subset, scattered in space and time, by randomly sub-sampling 10000 data points from time t = 0 to t = 4.0 . The rest of the domain from time t = 4.0 to the final time t = 5.0 will be referred to as the test portion. Given the training data, we are interested in learning N as a function of the solution u and its derivatives up to the 3rd order.

1.4 Viscid Burgers Equation

- The viscid Burgers PDE equation is of the form:

$$u_t + (\frac{u^2}{2})_x = \epsilon u_{xx} \tag{7}$$

  In the domain $(x, t) \in [-8, 8] \times [0, 10]$ and in particular $\epsilon = 0.01$.

- The first initial condition we consider is $u_0(x) = -\sin\left(\frac{\pi x}{8}\right)$ and the second initial condition we consider is $u_0(x) = \exp(-(x+2)^2)$.

- General description of the problem in neural network: the nerual network of inviscid Burgers equation:

$$u_t = \mathrm{N}(u, u_x, u_{xx}) \tag{8}$$

- We represent the solution u by a 5-layer deep neural network with 50 neurons per hidden layer. Furthermore, we let N to be a neural network with 2 hidden layers and 100 neurons per hidden layer and in the input layer this N takes input of 3.

- Out of this data-set (generated by chebfun in Matlab), we generate a smaller training subset, scattered in space and time, by randomly sub-sampling 10000 data points from time t = 0 to t = 6.7. The rest of the domain from time t = 6.7 to the final time t = 10.0 will be referred to as the test portion. Given the training data, we are interested in learning N as a function of the solution u and its derivatives up to the 2nd order.