**xtImportant Links:**

DeepXDE - a library for physics-informed neural networks:

<https://github.com/lu-group/pinn-sampling/blob/main/src/burgers/Uniform.py>

Comparison of different adaptive sampling methods:

[**https://reader.elsevier.com/reader/sd/pii/S0045782522006260?token=376307ABE3E76299DCA090B4F8945A953BA0ABDD20A80A1B2192D70BE8E599639F88445259FAEB03D149D37654A02F1D&originRegion=eu-west-1&originCreation=20221103193035**](https://reader.elsevier.com/reader/sd/pii/S0045782522006260?token=376307ABE3E76299DCA090B4F8945A953BA0ABDD20A80A1B2192D70BE8E599639F88445259FAEB03D149D37654A02F1D&originRegion=eu-west-1&originCreation=20221103193035)

**14th Aug 2022, Saturday**

**Results:**

| 8 layers : [200, 200, 200, 200, 200, 200, 200, 200]  No of epochs: 700  Training error: 0.006857  Test error: 0.188 |
| --- |

| 8 layers : [200, 200, 200, 200, 200, 200, 200, 200]  No of epochs: 10000 (training stopped after 3800 epochs, error saturated)  Training error: 1.32e-4  Test error: 0.01622 |
| --- |

| 8 layers: 20 each  No of epochs: 10000  Training error: 6.93e-6  Test error: 6.86e-3 |
| --- |

| 8 layers: 20 each  No of epochs: 10000  Normalisation : (x - mu / sigma)  Training error: 1.339e-03  Test error: 2.956e-01 |
| --- |

| 4 layers: 200 each  No of epochs: 10000  Training error: 1.05e-4  Test error: 4.09e-2  ˀ. |
| --- |

**Observations**:

* (0, 1) Normalisation works better than (x - mu / sigma)
* Optimal number of neurons and layers are - 20 and 8 respectively
* Test error / training error ~ 100 times; therefore some regularisaiton has to be implemented
* Training of aprox 5000 is required in the given problem

**Following variations can be implemented and tested**:

# Initialization can be changed

# optimizer currently implemented - LBFGS; we can try with other optimisers

# add regularisation - L1/L2 penalty, dropout

**Date: 17th Aug, 2022**

Layer Normalisation:

* For Tanh: the training is getting finished after a very few iterations. This is due to diminishing gradients
* For ReLU it is happening much earlier

Batch Normalisation:

* For both the activation functions, training died very early. Model did not learn anything at all

| Xavier Weight Initialisation:   * Activation Function : Tanh * Training error : 2.08e-06 * Test error : 1.369e-03 |
| --- |

| Dropout:   * P = 0.05: training died after around 130 epochs * Training error : 0.1692 * Test error : 0.6515   Overall dropout is not useful |
| --- |

| Max\_iter = 20  Num\_epochs = 1000  Training error:  Iter 10641, Loss: 5.13110e-06, Loss\_u: 6.77857e-07, Loss\_f: 4.45324e-06  CPU times: user 3min 11s, sys: 5.72 s, total: 3min 17s  Wall time: 3min 13s  Training error:  Error u: 2.449404e-03 |
| --- |

**# 25th Aug**

Tasks to do:

1. Optimise using nelder mead
   1. The optimisation surface is highly non-convex. So, it is possible that the gradient is getting struck somewhere in the local mimina. In such cases nelder mead may turn out to be useful
2. ‘’ using genetic algorithm

Use adaptive-usage2 of collocation points

**# 25th Sep**

**Adaptive Collocation Points**

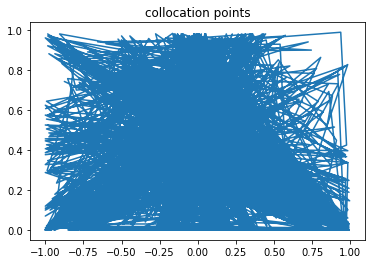
Corrected Algorithm results:

| Nf = 1000  Iter 8046, Loss: 8.21479e-05, Loss\_u: 1.31280e-05, Loss\_f: 6.90198e-05  CPU times: user 9min 41s, sys: 3.89 s, total: 9min 45s  Wall time: 9min 42s  Test error- Error u: 6.673294e-03 |
| --- |

| Training error;  Iter 7888, Loss: 1.75388e-05, Loss\_u: 1.74947e-06, Loss\_f: 1.57893e-05  CPU times: user 10min 59s, sys: 4.3 s, total: 11min 3s  Wall time: 11min  Test error- Error u: 6.814899e-03 |
| --- |

When prob[i][j] = (res)\*\*3

Test error: Error u: 3.425725e-02



Adaptive-gradient:

Take1:

| “Adapt-g”  1k collocation points  Iter 8005, Loss: 9.88699e-05, Loss\_u: 2.95401e-05, Loss\_f: 6.93298e-05  CPU times: user 2min 33s, sys: 2.65 s, total: 2min 35s  Wall time: 2min 37s  Test error: Error u: 5.999757e-02 |
| --- |

Take2:

| “Adapt-g”  10k points  Iter 7895, Loss: 5.40217e-05, Loss\_u: 1.74790e-05, Loss\_f: 3.65427e-05  CPU times: user 3min 59s, sys: 3.27 s, total: 4min 3s  Wall time: 4min  Test error: Error u: 1.023975e-02 |
| --- |

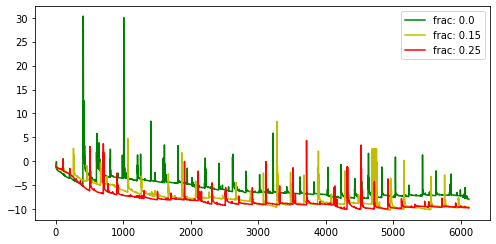
Some random plots of takes:

|  |
| --- |

| max\_iter=40  self.num\_epochs = 400  Self.e = 10  Iter 14529, Loss: 4.62503e-05, Loss\_u: 1.90334e-06, Loss\_f: 4.43470e-05  CPU times: user 4min 37s, sys: 5.3 s, total: 4min 42s  Wall time: 4min 39s  Test error: Error u: 4.538533e-03 |
| --- |

| max\_iter=40  self.num\_epochs = 400  Self.e = 10  Iter 9896, Loss: 1.91154e-05, Loss\_u: 1.92372e-06, Loss\_f: 1.71917e-05  CPU times: user 3min 9s, sys: 3.56 s, total: 3min 12s  Wall time: 3min 9s  Error u: 1.464214e-02 |
| --- |

# ADAM + LBFGS:



# Softmax function for probabilities:

Softmax with cosine annealing:

Temperature = 20

| Iter 7893, Loss: 1.50004e-05, Loss\_u: 1.21376e-06, Loss\_f: 1.37867e-05  CPU times: user 2min 38s, sys: 2.78 s, total: 2min 41s  Wall time: 2min 39s  Error u: 9.446155e-03 |
| --- |

| Iter 7892, Loss: 1.50353e-05, Loss\_u: 3.45056e-06, Loss\_f: 1.15847e-05  CPU times: user 2min 39s, sys: 2.76 s, total: 2min 42s  Wall time: 2min 41s  Error u: 3.502753e-02 |
| --- |

| Iter 7890, Loss: 1.00883e-05, Loss\_u: 1.23000e-06, Loss\_f: 8.85826e-06  CPU times: user 2min 38s, sys: 2.89 s, total: 2min 41s  Wall time: 2min 39s  Error u: 6.432609e-03 |
| --- |

Softmax without annealing:

Tried running several times, but same result:

| Temperature = 20  Iter 2024, Loss: 1.32538e+16, Loss\_u: 3.95233e+03, Loss\_f: 1.32538e+16 |
| --- |

| Temperature = 200  Iter 7895, Loss: 2.16511e-05, Loss\_u: 3.51170e-06, Loss\_f: 1.81393e-05  CPU times: user 2min 54s, sys: 2.77 s, total: 2min 57s  Wall time: 2min 55s  Error u: 4.868308e-02 |
| --- |

| Temperature = 100  Iter 879, Loss: 2.52207e+14, Loss\_u: 1.09691e+04, Loss\_f: 2.52207e+14  Iter 880, Loss: nan, Loss\_u: nan, Loss\_f: nan |
| --- |

| Temperature = 150  Iter 7827, Loss: 3.12462e-05, Loss\_u: 3.48002e-06, Loss\_f: 2.77661e-05  CPU times: user 2min 52s, sys: 2.8 s, total: 2min 54s  Wall time: 2min 52s  Error u: 6.156201e-03  Take2:  Iter 2013, Loss: 1.12617e+21, Loss\_u: 2.91748e+04, Loss\_f: 1.12617e+21  Iter 2014, Loss: nan, Loss\_u: nan, Loss\_f: nan |
| --- |

Observations:

1. For high temperatures, the results are good, comparable to that with annealing and without softmax
2. ‘Nan’ values are observed at lower epochs for low temperatures.
   1. Possible explanation is that for low lower epochs the model is not well trained and a lower temperature give a complete outlier, for which the model gives large errors and therefore gradients explode
3. The accuracy is approx the same as that with cosine annealing. But in cosine annealing, the possibility for gradients to explode is less!
4. Training is quite temperature sensitive

Future Scope:

1. Visualisation:

Make a 128x100 grid. Count the number of points in each square and plot a heat map. This will give the frequency of the number of points in a region

Problems Faced:

1. Even with lbfgs we are sometimes getting exploding gradients: getting ‘nan’ values for loss
2. With the softmax function we are converging to nan values much faster
   1. Possible explanation:

Whenever there is an entirely new set of points, it may lead to a huge error, therefore the gradients are more (concave upwards) and so it explodes!!!

**19th Oct, 2022:**

* When ‘e’ is small (=10) for simple probability distribution-> error ~ 4e-02
* When ‘e’ = 20 -> error ~ 3e-03
* Therefore, e should not be too small that it keeps on resampling near the points where error is more
* When Nf = 100 (extremely small) -> even the simple probability distribution was giving NaN
* Therefore, maybe for Gibbs sampling, we need to increase the number of collocation points a little
* Things are not reproducible even with simple prob distribution

| Nf = 10k  T = 500  Gibbs sampling (cosine annealing; temperature is constant)  Epochs = 10,000 (stagnated at around 7000)  Error u: 6.819832e-04 |
| --- |

| Nf = 10k  T = 200  Gibbs sampling (cosine annealing; temperature is constant)  Epochs = 10,000 (stagnated at around 6300)  Error u: 9.554199e-04 |
| --- |

| Nf = 10k  T = 500  Rate of decrease = 1%  Gibbs sampling  Epochs = 10,000  Error u: 6.236266e-03 |
| --- |

| Nf = 10k  T = 5  Sampling rate = 200  Gibbs sampling (cosine annealing; temperature is constant)  Epochs = 10,000 (stagnated at around 6300)  CPU times: user 7min 53s, sys: 4.7 s, total: 7min 58s  Wall time: 7min 55s !!!!  Error u: 2.331036e-03 |
| --- |

| Nf = 1000  T = 500  Rate of decrease = 1%  Gibbs sampling  Epochs = 10,000  Error u: 4.816266e-01 |
| --- |

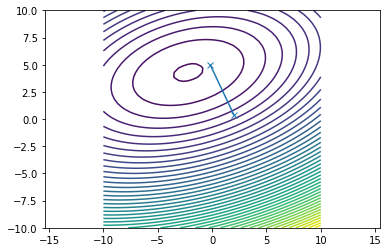
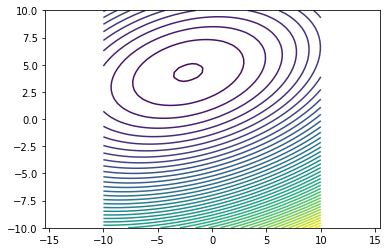
Conjugate-Gradient Method

Results:

Hyperparameter: Number of steps taken to reach the minima (Phi)

Method: Hyperparameter tuning

Example:



Results on burger’s equation:

Phi = 500

Training error: 0.0007384

Test error: 0.65667

Phi =1000

Training error: 0.00008459

Test error: 0.03599

Phi =1250

Training error: 3.6e-5

Test error: 2.1e-3

Phi=1300

Training error: 8.669e-6

Test error: 5.8e-3

Phi = 1350

Training error: 9.1e-6

Test error: 2e-3

Phi = 1500

Training error: 3.22e-7

Test error: 2.2e-3

Phi = 2500

Training error: 3.4e-7

Test error: 2.317e-3

Phi = 5000

Training error: 3.4e-7

Test error: 2.4e-3

Phi = 7000

Training error: 3.6e-7

Test error:2.4e-3

Phi= 8000

Training error: 3.7e-7

Test error: 2.57e-3

Phi=9000

Training error: 3.72e-7

Test error: 2.6e-3

Phi=10000

Training error: 3.7334e-7

Test error: 2.6e-3

Observations:

1. Reaching optimal minima and increase in a further number of steps increases the test error
2. After the “minima” Phi value, test error remained almost constant and the training process is clearly overfitting
3. Even at the minima, it cannot find the critical points where there is a change in slope and it is achieving a local minimum. Increasing the number of steps greater than 10000 is leading to memory outrun, hence sufficient time is not given to reach the global minima points.

<https://colab.research.google.com/drive/13tVo0e67-bpj99nRnxjmkYwM52rwY3JE#scrollTo=PfvdFN1PtBIV>

# **Adaptive Sampling Debugging**:

| Max\_iter = 100  E = 2  Temperature = 10  For uniform sampling: max prob ~ 3e-05  But at iter = 500: max prob ~ 1290e-05 (=0.0129) extremely high  Therefore, it is quite possible the same point is being sampled many times  To overcome this, what we can due is put a threshold such that all the points with residual values greater than a particular value will be considered = threshold value (but what should this threshold be??) |
| --- |

| Max\_iter = 100  E = 2  Temperature = 100 (constant)  Added boundary points to the collocation points  Max prob = 3.906e-05 (uniform sampling)  Results:  Error u: 1.331957e-03  Issues:  Number of collocation points finally = 2912 - 2x times of what it is supposed to be |
| --- |

**Adaptive training:**

Modified the loss to be weighted loss

Using momentum type of probabilities for proxy (p\_new + gamma\*p\_old)

Things we can try:

* Adding points to the current set sampled according to probabilities

| Using proportionality as probabilities:  Loss = L\_u + 0.001\*L\_f  Gamma = 0.1  Num of epochs = 10000  Num of collocation points = 2000  iter 9891, Loss: 5.79062e-06, Loss\_u: 7.19358e-07, Loss\_f: 5.07127e-03  Test error: Error u: 2.861200e-02 |
| --- |

| Uniform non-adaptive sampling  Loss = L\_u + 0.001\*L\_f  Num of epochs = 10000  Num of collocation points = 2000  iter 2476, Loss: 3.14011e-04, Loss\_u: 1.25729e-05, Loss\_f: 3.01438e-01  Error u: 3.479325e-01 |
| --- |

| Using Gibbs sampling: T = 20 - 1.0 (with a decrease of 10% after 100 epochs)  Loss = L\_u + 0.01\*L\_f  Gamma = 0.0  Num of epochs = 10000  Num of collocation points = 2000  iter 9824, Loss: 4.02361e-06, Loss\_u: 3.68469e-07, Loss\_f: 3.65514e-04  Error u: 8.753277e-03 |
| --- |

| Using Gibbs sampling: T = 20 - 0.5 (with a decrease of 10% after 100 epochs)  Loss = L\_u + 0.01\*L\_f  Gamma = 0.0  Num of epochs = 10000  Num of collocation points = 2000  iter 9825, Loss: 4.78648e-06, Loss\_u: 3.28305e-07, Loss\_f: 4.45818e-04  Error u: 8.919903e-03 |
| --- |

| Using Gibbs sampling: T = 50 - 2 (with a decrease of 5% after 100 epochs)  Loss = L\_u + 0.01\*L\_f  Gamma = 0.0  Num of epochs = 20000  Num of collocation points = 2000  iter 17295, Loss: 1.75890e-06, Loss\_u: 2.03338e-07, Loss\_f: 1.55556e-04  Error u: 3.925251e-03 |
| --- |

**Things to discuss with sir**:

1. Explain the problem that we were facing earlier with nan values for Gibbs sampling
2. Show that the same is happening with uniform adaptive sampling
3. Potential reasons? Remedies?
4. Discuss about tfc. Explain what it is and ask how to go about choosing the support functions? Monomials? Or what?
5. Give a small idea of hypernetworks. Explain its use and ask if its going to be of an use for us in solving HJB equation.

Results for PPT and Report

| iter: 7995 | max value in prob: 0.2665071487426758  iter: 8015 | max value in prob: 0.3245983123779297  CPU times: user 4min 18s, sys: 2.39 s, total: 4min 21s  Wall time: 4min 22s  Test error: Error u: 4.980397e-03 |
| --- |

| iter 20010, Loss: 6.75629e-06, Loss\_u: 1.71278e-06, Loss\_f: 5.04352e-06  Wall time: 9min 35s  Test Error u: 7.884466e-03 |
| --- |