Physics informed neural networks for the quantum harmonic oscillator

This repository contains a implementation of physics informed neural networks for the 2D Schrödinger equation especally the quantum harmonic oscillator.

Architecture

Physics informed neural network is an approach to solve partial differential equations data-driven with neural networks. The implemenation of physics informed neural networks is defined in the paper Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. In our approach we builded a physics informed neural network (PINN) to solve the 2D Schrödinger Eqauion especally the quantum harmonic oscillator. The training of the PINN is splitted in two phases. The first phase is the pretraining phase. In the pretraining phase the network will be trainied on the initial condition (t=0). And in the second phase the network will trained on the complete domain. A weighted PDE-Loss was introduced for the second phase to balance the influence of the initial condition.

Implementations

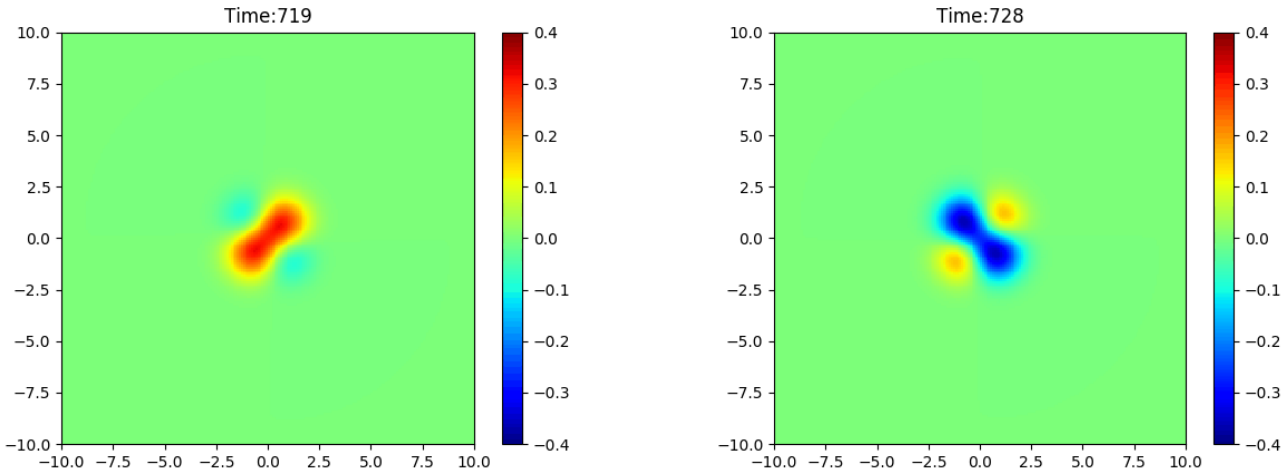
This repository contains following implementations:

Implementation	Description
Schrödinger2D_baseline.py	Training of the quantum harmonic oscillator as a supervised learning task. The algorithm is parallelized by Horovod
Schrödinger2D_torch.py	Imlementation of physics informed neural in pytorch
Schrödinger2D_hvd.py	Imlementation of physics informed neural in pytorch and parallelized by horovod
Schrödinger2D_param.py	Extented version of the horovod implementation parameterized by the frequency of the harmonic oscillator f
Schrödinger2D_transfer.py	Implementation of the Transferlearning-Approach. Needs a pretrained model.

Quantum harmonic oscillator

In general,the quantum harmonic oscillator (QHO) is a special form ofthe Schrödinger Equation and describes the behavior of oneparticle in a harmonic potential. We use this special form of the Schrödinger

Equation because it can be solved analitically. The solutions of qho look as follows (left real part, right imaginary part):



The analytical solution is also provided in form of a Python-Script. An example call of the analytical solver script that calculates the solution of the quantum harmonic oscillator with frequency = 2 for 2000 steps and write them into the folder:"solution"

```
python SchrodingerAnalytical.py --prefix "solution" --f 2
```

It is possible to configure the solver by your self. You can see all options with:

```
python SchrodingerAnalytical.py --help
```

Usage of implementations

In the Script Folder are different bash scrips that running different experiment runs which defined in the experiment.csv file. Here an example of an bash script:

```
#!/bin/bash -l

#SBATCH -p ml
#SBATCH -t 120:00:00
#SBATCH --nodes=10
#SBATCH --ntasks=60
#SBATCH --hint=multithread #important for multiple node horovod
#SBATCH --cpus-per-task=28
#SBATCH --gres=gpu:6
#SBATCH --exclude=taurusml1
#SBATCH -o experiment19.out
#SBATCH -e experiment19.err
#SBATCH --job-name="exp19"


module load modenv/ml
module load OpenMPI/3.1.4-gcccuda-2018b
module load PythonAnaconda/3.6
module load cuDNN/7.1.4.18-fosscuda-2018b
module load CMake/3.11.4-GCCcore-7.3.0

source activate horovod #conda enviorment

cd /home/s7520458/AIPP/2D_Schrodinger.

srun python3.6 Schrodinger2D_hvd.py --identifier experiment19 \ identifier for ten
sorboard
                                --batchsize 1700\ #Size of batch
                                --numbatches 5000 \ # Number of Batches
                                --initsize 15000 \ # Number of sampling points for i
                                --numlayers 8 \ # Number of layers
                                --numfeatures 1000 \ # Number of neurons per layer
                                --epochssolution 1000 \ #Number of epochs for first
                                --epochsPDE 7000 \ #Number of epochs for second trai
                                --energyloss 1 \ # enable (1) or disable(0) probabil
                                --pretraining 1 \ # enable (1) or disable(0) pretrai
                                --alpha 12 \  # weight for weighted loss
                                --lhs 0 # enable (1) or disable(0) lhs sampling
```

Dependencies

- PyTorch
- Horovod
- Tensorboard
- TensorboadX