Code description for

Local neural operator for solving transient partial differential equations on varied domains

by Hongyu Li, Ximeng Ye, Peng Jiang, Guoliang Qin, Tiejun Wang

Program structure

- ***Train_Validation***: training and validation of LNO
  - *main.*py*:* main program to run the training or validation of LNO
  - *lib:* supportive functions and supportive data
    - *chebyshevs:* the discrete kernel $\varphi_{m,i}$ in Eq. (17) and $\psi_{m,i}$ in Eq. (19) for 5^{th}~41^{st}-order Chebyshev polynomials used in the spectral path, calculated according to Eqs. (S21-S22)
    - *legendres:* the discrete kernel $\varphi_{m,i}$ in Eq. (17) and $\psi_{m,i}$ in Eq. (19) for 5^{th}~41^{st}-order Legendre polynomials used in the spectral path, calculated according to Eqs. (S21-S22)
    - *networks_LNO.*py*:* the network of local neural operator, including the networks for Navier-Stokes equation (and 2D viscous Burgers equations), Wave equation, and 1D viscous Burgers equation
    - *train.*py*:* functions to train LNO
    - *test.*py*:* functions to test trained LNO
    - *utils*.py*:* supportive functions to generate the kernel of spectral path
  - *Data:* functions to generate and extract data for training
    - *DatasetNS.*py: generate dataset for Navier-Stokes equations
    - *DatasetBurgers1D.*py: generate dataset for 1D viscous Burgers equations
    - *DatasetBurgers2D.*py: generate dataset for 2D viscous Burgers equations
    - *DatasetWave.*py: generate dataset for wave equations
  - *models:* the trained LNO models
  - *outputs:* predicted results on validation samples by trained LNOs
  - *logs:* the output logs during the training process

- ***Application***: apply pre-trained LNO to solve unseen problems
  - *mainSquareCylinder.*py: main program to solve the flow around a square cylinder
  - *mainCascade.*py: main program to solve the flow across a cascade
  - *IBMInterpolation.*py: function to implement immersed boundary method
  - *NACA0012_20.*mat: geometry file of the airfoil in the cascade
  - *lib*: supportive functions

- *networkNS*.py: the network of local neural operator for Navier-Stokes equation, almost the same as *networks_LNO*.py but padding operations are removed
- utils.py: supportive functions to generate the kernel of spectral path

- *models*: pre-trained LNO models for solving the unseen problems

How to use

- To train a new LNO and then test:

  Enable `train_test_save()` in *main*.py and run command:

  ```
  nohup python -u main.py -n run_name > logs/run_name.log 2>&1 &
  ```

  The trained LNO will be in models named *run_name_model*.pp and the training log will be in *logs* named *run_name*.log.

- To test a trained LNO:

  Enable `load_test(args.out_name)` in *main*.py and run command:

  ```
  nohup python -u main.py -n run_name > logs/run_name.log 2>&1 &
  ```

  The results of predicting the validation data samples will be in *outputs* named *run_name*.mat.

- To solve the flow around a square cylinder:

  Put a pre-trained LNO model file into *models* or select one from *models*, change `model_file` in *mainSquareCylinder*.py and run command:

  ```
  python mainSquareCylinder.py
  ```

  The predicted flow fields at different time levels will be named *SC_timelevel*.mat.

- To solve the flow across a cascade:

  Put a pre-trained LNO model file into *models* or select one from *models*, change `model_file` in *mainCascade*.py and run command:

  ```
  python mainCascade.py
  ```

  The predicted flow fields at different time levels will be named *Cascade_timelevel*.mat.

Dataset description

Data samples for each PDE to be learned are stored in one independent folder.

PDE	Folder name
Navier-Stokes equation	NS128Re{}t1000
1D viscous Burgers	Burgers128Re100t1000
2D viscous Burgers	Burgers2D128Re100t1000
Wave equation	Wave128t1000

In each folder, data samples are stored in pieces with name *folder_name_order*.mat, e.g., *NS128Re500t1000_1*.mat, which is the physical field calculated from one random initial condition. Each data file includes all the physical fields required for training, and each physical field is in the format [total_time_steps×field_value_in_a_time_level]. The example data samples can be found in https://pan.baidu.com/s/1QMH_1VvlODgivHOY-aTj1g

code: j3h7

Unzip the folder at */Train_Validation/Data/*

Main adjustable parameters

Parameter	Description	Options
	Train_Validation	
`PROBLEM`	The type of PDE to be learned	'NS' for Navier-Stokes equation 'Burgers1D' for 1D viscous Burgers equation 'Burgers2D' for 2D viscous Burgers equation 'Wave' for wave equation
`data_dir`	Path of dataset	/
`data_name`	Name of folder according to Table 1	/
`Re`	Viscosity of Navier-Stokes equation to be learned	/
`t_interval`	Controling the time step Δt of the learning task, $\Delta t = \Delta\tau \times t\_interval$, where $\Delta\tau = 0.01$ is the time step of training data samples	Positive integer
`learning_rate`	Initial learning rate	0.001 (recommend)
`reccurent`	Round number for recurrent training,	10 (recommend)
`epochs_overall`	Epoch number of training	200 (recommend)
`iterations`	Iteration number in each epoch	500 (recommend)
`orders_all`	Orders of all used data samples, including both training and validation samples	/
`orders_train`	Orders of training data samples	/
`n`	Order of spectral transform N	5~41
`m`	Selected first m lowest modes M	$\leq$ n
`k`	Number of repetitions K	2 (recommend)
	Application	
`model_file`	File name for the pre-trained LNO model	/
`NG_L,NG_D,NG_U,NG_R`	The size of the computational domain from the original coordinate in the left, down, up, and right sides	/
`u_lid`	Velocity of the inflow	1 (recommend)
`alpha`	Angle of attack α for the cascade	/