Code description for

On the locality of local neural operator in learning fluid dynamics

By Ximeng Ye, Hongyu Li, Jingjie Huang, Guoliang Qin

Program structure

- ***Train_Validation***: training and validation of LNO
  - *main*.py*:* main program to run the training or validation of LNO, including 3 functions:
    - *train_test_save():* train, validate, and save an LNO
    - *load_test(args.out_name):* load and validate a trained LNO
    - *test_receptive(args.out_name):* load a trained LNO and analyze its receptive field
  - *receptive*.py: analyze the receptive field of an initialized LNO
  - *lib:* supportive functions and supportive data
    - *legendres:* the discrete kernel $\varphi_{m,i}$ and $\psi_{m,i}$ for 5^{th}~41^{st}-order Legendre polynomials used in the spectral path
    - *networksNS*.py*:* the network of local neural operator for 2D compressible Navier-Stokes 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
  - *receptives*: saved data of receptive field
  - *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
  - *mainCircularCylinder*.py: main program to solve the flow around a circular cylinder
  - *mainVehicle*.py: main program to solve the flow around a vehicle
  - *IBM*.py: function to implement immersed boundary method
  - *geometry:* geometry files of circular cylinder with different diameters and different shapes of vehicles
  - *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
- *CircularCylinder & Vehicle*: folders to store the predicted flow fields of the two examples

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 calculate the receptive field of a trained LNO:

  Enable `test_receptive(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 *receptives* named *run_name_receptive*.mat.

- To calculate the receptive field of an initialized LNO:

  Run command:

  ```
  python receptive.py
  ```

  The results of predicting the validation data samples will be in *receptives* named *initial_n{}_N{}_M{}_K{}*.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 mainCircularCylinder.py
  ```

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

- To solve the flow around a vehicle:

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

  ```
  python mainVehicle.py
  ```

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

Dataset description

Data samples

Data samples for each learning task to be learned are stored in one independent folder named *ComNS128Re{}Ma{}*. In each folder, data samples are stored in pieces with name *folder_name_order*.mat, e.g., Com*NS128Re100Ma2_1*.mat is the 1st data sample for $Re = 100, Ma = 0.2$, 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/13ZRXGuaBujk6zlA98tWIlQ

Code: sy84

Unzip the folder at /Train_Validation/Data/

Main adjustable parameters

Parameter	Description	Options
	Train_Validation	
`data_dir`	Path of dataset	/
`data_name`	Name of folder according to Table 1	/
`Re`	Reynolds number of Navier-Stokes equation to be learned	/
`Ma`	Mach number 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	/
`num_blocks`	Number of blocks n	4 (recommend)
`N`	Order of spectral transform N	5~41
`M`	Selected first m lowest modes M	$\leq$ n
`K`	Number of repetitions K	2 (recommend)
`Norm_facotrs`	Factors to normalize the input according to Appendix E	[0.5, 0.5, 5, 10] (recommend)
`Init_weight`	Factors to initialize the weights in LNO according to Appendix E	$[\sqrt{3}, \{\}, \sqrt{6}, \sqrt{3}, \sqrt{6}, \sqrt{3}]$ (recommend, see Table E1 for the 2nd value)
`if_ln`	Set input ρ, T to their logarithm for normalization according to Appendix E	True (recommend)
	Application	
`N`	Order of spectral transform for choosing LNOs with different receptive ranges, 8 for insufficient, 12 for compatible, 20 for excessive range	8,12,20
`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	/

	center of the solid wall geometry in the left, down, up, and right sides	
`delta_x`	Size of spatial discretization according to the training samples	1/64 (default)
`u_lid`	Velocity of the inflow	1 (recommend)
`filename`	Geometry file name of the solid obstacle	'CircularCylinderD{}.mat' for circular cylinder 'vehicle_sports.mat' for vehicle
`p_l`	Equidistant points on the solid wall of the obstacle	/
`delta_s`	Distance between points in `p_l`, should be around `delta_x`	/
`ShapeNum`	Number of obstacles	/
`d`	Diameter of circular cylinder	/
`cycle_num`	Total number of time-marching steps	/