Blind Deconvolution of turbulence flows using Neural Networks

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

1 Problem background

1.1 Convolution and Deconvolution

In applied mathematics and computer science (and, in particular, functional analysis) convolution is a mathematical operation on two functions (f and g) that produces a third function, which is typically viewed as a modified version of one of the original functions. This modified function gives the integral of the point-wise multiplication of the two functions as a function of the amount that one of the original functions has been translated.

The convolution of f and g is written $f{*}g$, using an [*] or star. It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of [[integral transform]]:

(f * g)(t) $\stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$
$= \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$

Deconvolution, on the other hand is an algorithm-based process used to reverse the effects of convolution on recorded data. The concept of deconvolution is extensively used in the techniques of signal processing and image processing.

1.2 Turbulence modeling and the Closure Problem

In fluid dynamics, turbulence or turbulent flow is any pattern of fluid motion characterized by chaotic changes in pressure and flow velocity. It is in contrast to a laminar flow regime, which occurs when a / the fluid flows in parallel layers, with no disruption between those layers.

In Turbulence modeling, one constructs and uses a model to predict the effects of turbulence. A turbulent fluid flow has features on many different length scales, which all interact with each other. A common or naive approach is to average the governing equations of the flow, in order to focus on large-scale and non-fluctuating features of the flow.

The velocity and pressure of a fluid flow is governed by the Navier–Stokes equations. In a turbulent flow, each of these quantities may be decomposed into a mean part and a fluctuating part. On averaging the equations, we get the Reynolds-averaged Navier–Stokes (RANS) equations, which govern the mean flow. The non-linearity of the Navier–Stokes equations however means that the velocity fluctuations still appear in the RANS equations, in the nonlinear term $\rho\overline{v_i^\prime v_j^\prime}$ from the convective acceleration. This term is known as the Reynolds stress, R_{ij}.[2] Its effect on the mean flow is like that of a stress term, such as from pressure or viscosity.

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

To obtain equations containing only the mean velocity and pressure, we need to close the RANS equations by modelling the Reynolds stress term R_{ij} as a function of the mean flow, removing any reference to the fluctuating part of the velocity. This is the closure problem.

2 Our Objectives

The major motivation for this project is due to the recent advances in the image processing community which employ general machine learning techniques used for reconstruction of noisy or blurred images. In particular, our objective is to implement Artifical Neural Network (ANN)-based machine learning strategies to recover subfilter-scale features in turbulence closure modeling.

We aim to develop a single-layer feed forward ANN to identify a non-linear relationship between the low-pass spatially filtered and coarse-grained (but unfiltered) field variables for settings in two-dimensional (2D) and three-dimensional (3D) homogenous isotrpoic turbulence as well as a stratified turbulence case exhibiting moderate compressibility in the limit of infinite Reynold's numbers.

The approach outlined in our study is analogous to the approximate deconvolution methodology (Stolz and Adams 1999) to recover subfilter contributions of low-pass spatially filtered flow fields, and the only difference is the lack of assumption of any filtering kernel (Gaussian or otherwise) - which is why we call the deconvolution 'blind'.

3 Approach

The first step was to import and preprocessed the data - we extracted the dataset (described below) from the JHTDB website, and pre-processed it by using the shifting strategy, filtering it and finally generating the training and test sets for our algorithms.

3.1 The Dataset

The datasets we used were taken from the John Hopkins Turbulence Databases (http://turbulence.pha.jhu.edu/) -

Forced isotropic turbulence which was generated from direct numerical simulation (DNS) using $1,024^3$ nodes. Here the Navier-Stokes equation was solved using pseudo-spectral method, and the energy was injected by keeping constant the total energy in shells such that $|k|$ is less or equal to 2. After the simulation reached a statistical stationary state, 5,028 frames of data with 3 velocity components and pressure were stored in the database. The Taylor-scale Reynolds number fluctuates around R 433. In one dataset ("coarse") there are 5028 timesteps available, for time t between 0 and 10.056 (the frames are stored at every 10 time-steps of the DNS) and the intermediate times can be queried using temporal-interpolation. In the other dataset ("fine"), single time-step of the DNS is stored, for testing purposes. Times available are for t between 0.0002 and 0.0198). This data was used for the 3D case.

Forced Isotropic Turbulence Dataset on 4096^3 Grid which was generated from direct numerical simulation (DNS) using 4096^3 nodes. The Navier-Stokes equation was solved using pseudo-spectral method, and the time integration uses second-order Runge-Kutta method. While the simulation is de-aliased using phase-shifting and truncation, energy is injected by keeping the energy density in the lowest wavenumber modes prescribed following the approach of Donzis Yeung. After the simulation reached a statistical stationary state, a frame of data, which includes the 3 components of the velocity vector and the pressure, were generated and written in files that could be accessed directly by the database (FileDB system).

3.2 Data Preprocessing

The 3D numerical experiments were generated using 512^3 degrees of freedom while the 2D case was generated using 2048^2 grid points. A coarse-grained large eddy simulation was mimicked through the subsequent sampling of these high-quality datasets. The caorse-graining procesure was then applied with a sub-sampled field of 64^3 and 256^2 degrees of freedom in the 3D and 2D test cases, respectively. In other words, our coarse-graining procedure involves the selection of every eighth point in the high-fidelity uniform grid data.

We devised several shifted data sets (each possessing 64^3 degrees of freedom in 3D and 256^2 degrees of freedom in 2D) which continue to represent the physics of the fine-grained dataset. The missing data points at boundaries were reconstructed through the use of the periodic boundary conditions for the given test cases. A simplified shifting schematic was employed (shown in figure 1 below),

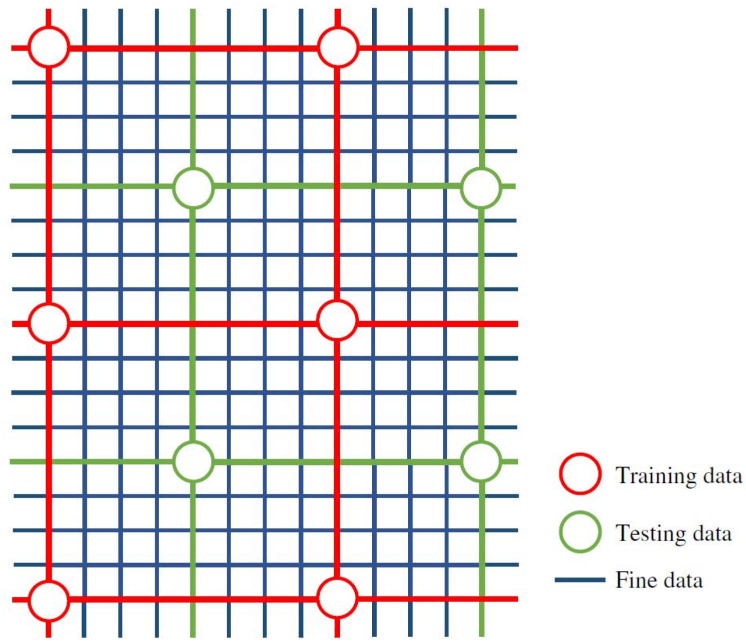

Figure 1: Schematic of spatial shifting strategy for a simplified two-dimensional grid showing two different data-sets

where a simple technique of generating two coarse data sets from fine data is demonstrated. In short, the coarse-graining and shifting procedures allow us to devise up to 63 and 511 completely different data sets in two and three dimensions, respectively. For cross-validation we randomly chose any four of these multiple data sets for the generation of three sets of testing data and one set of training data.

After our four different randomly generated data sets are identified, we added perturbations corresponding to the type of behaviour demanded of the proposed artificial neural network. To test the deconvolution ability of our network, our training data (i.e., one of the four data sets) was filtered with an appropriate Gaussian smoothing (for our inputs to the network) and unfiltered training data were utilized as outputs to the network. Three testing data sets are generated in a similar manner, with one of these data sets being filtered with the same filter radius and the others being filtered with a 10% larger and 10% smaller filter radius, respectively (Figure 2).

The trained ANN was then utilized to recover deconvolved approximations to the true field for these three test data sets. This ensured that the trends of the trained network were not due to overfitting or 'data-memory' but through an implicit learning of the inverse filtering. A similar procedure was also utilized to cross-validate the regularization ability of the closure wherein the randomly chosen data sets were perturbed through Gaussian noise. A table describing the filter radii and magnitudes of noise for our training and testing data sets is shown in Figure 3 below.

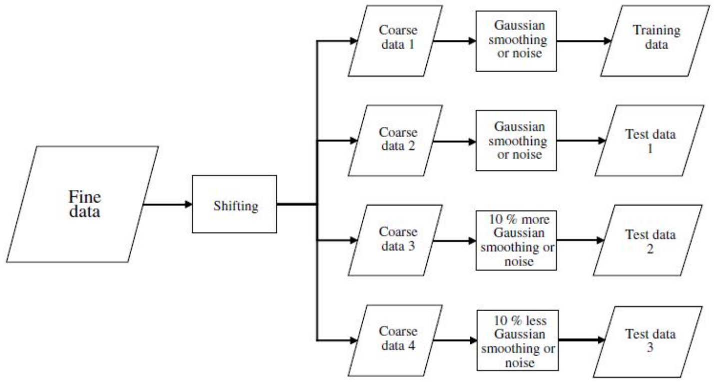

Figure 2: A flow chart explaining the generation of cross-validation data sets using the shifting operation and different perturbations.

Deconvolution		Regularization	
Data set	Filter radius (σ)	Data set	Noise (μ)
Training data	1.0	Training data	0.2
Test data 1	1.0	Test data 1	0.2
Test data 2	1.1	Test data 2	0.22
Test data 3	0.9	Test data 3	0.18

Figure 3: Cross-validation data sets for the proposed data-driven blind deconvolution closure

4 Extreme learning machine

5 Single layer feed-forward NN with keras

6 Two layer feed-forward NN with keras

7 Results

8 Conclusion

9 References

[1] https://en.wikipedia.org/wiki/Convolution

[2] https://en.wikipedia.org/wiki/Deconvolution

[3] https://en.wikipedia.org/wiki/Turbulence

[4] A neural network approach for the blind deconvolution of turbulent flows,R. Maulik and O. San, Journal of Fluid Mechanics, Get access Volume 831,25 November 2017, pp. 151-181, https://doi.org/10.1017/jfm.2017.637