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 $\rho\overline{v_i' v_j'}$ 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.

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 and Approach

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'.

2.1 Data preprocessing

2.1.1 HIT data from JHU

2.1.2 Shifting strategy

2.1.3 Filtering

2.1.4 Training and test data

3 Extreme learning machine

4 Single layer feed-forward NN with keras