Evaluation of a Model for the 3-Point 3^{rd}-Order Velocity Correlation in High Reynolds Number Isotropic Turbulence

H. Chang & R. D. Moser
Institute for Computational Engineering and Science, University of Texas at Austin
rmoser@ices.utexas.edu (512)-471-4946
9/1/2007

This document describes the semantics of the data used to describe the model for the three-point third-order velocity correlation in high Reynolds number isotropic turbulence reported in the Physics of Fluids paper entitled "An inertial range model for the three-point third-order velocity correlation," by Chang & Moser (2007, referred to as CM from here on). Also described is a set of Fortran 90 subroutines that read the data and evaluate the correlation tensor for given separation vectors.

As described in CM, the three-point third-order correlation tensor $\mathbb{T}_{ijk}(\mathbf{r},\mathbf{s})$ in homogeneous turbulence is defined

$$\mathbb{T}_{ijk}(\mathbf{r},\mathbf{s}) = \langle v_i(\mathbf{x}) v_j(\mathbf{x}+\mathbf{r}) v_k(\mathbf{x}+\mathbf{s}) \rangle \tag{1}$$

where v_i is the velocity. The third separation vector $\mathbf{t} = \mathbf{r} - \mathbf{s}$ is also needed. The magnitudes of the separations vectors are denoted r, s and t respectively. The inertial range model for this correlation is written in terms of 5 basis vectors $\mathbb{T}^n$. Each of the basis tensors is written in terms of 14 tensor functions Γ^m of $\mathbf{r}$ and $\mathbf{s}$ as follows:

$$\begin{array}{lll}
\Gamma^1_{ijk} = r_i r_j r_k & \Gamma^2_{ijk} = r_i r_j s_k & \Gamma^3_{ijk} = r_i s_j r_k \\
\Gamma^4_{ijk} = s_i r_j r_k & \Gamma^5_{ijk} = s_i s_j s_k & \Gamma^6_{ijk} = s_i s_j r_k \\
\Gamma^7_{ijk} = s_i r_j s_k & \Gamma^8_{ijk} = r_i s_j s_k & \Gamma^9_{ijk} = r_i \delta_{jk} \\
\Gamma^{10}_{ijk} = r_j \delta_{ik} & \Gamma^{11}_{ijk} = r_k \delta_{ij} & \Gamma^{12}_{ijk} = s_i \delta_{jk} \\
\Gamma^{13}_{ijk} = s_j \delta_{ik} & \Gamma^{14}_{ijk} = s_k \delta_{ij} &
\end{array}$$

Each basis tensor in the model is then given by

$$\mathbb{T}^n_{ijk} = \sum_{m=1}^{14} f^{n,m}(r,s,t) \Gamma^m_{ijk} \tag{2}$$

where the $f^{n,m}$ are rational functions. Kolmogorov inertial range theory and dimensional analysis implies that $\mathbb{T}$ scales linearly with the dissipation ϵ and a length scale related to the separations, and we choose the maximum of r, s and t to be that length scale. Using the symmetries of $\mathbb{T}$ (see CM), we can always rearrange arguments so that (for example) $s \le r \le t$. Thus in what follows, without loss of generality, we can assume that t is the largest separation, so that t is the scaling length. Define $\tilde{\mathbf{r}} = \mathbf{r}/t$, $\tilde{\mathbf{s}} = \mathbf{s}/t$, $\tilde{r} = r/t$ and $\tilde{s} = s/t$ and let $\tilde{f}^{n,m}(\tilde{r},\tilde{s}) = f^{n,m}(\tilde{r},\tilde{s},1)$. The rational functions $\tilde{f}$ can be expressed:

$$\tilde{f}^{n,m}(\tilde{r},\tilde{s}) = \frac{1}{N^n} \sum_{l=1}^{M^{n,m}} c_l^{n,m} \tilde{r}^{p_l^{n,m}} \tilde{s}^{q_l^{n,m}} \tag{3}$$

where N^n is an integer that normalizes the basis function to be consistent with $\epsilon = 1$, $M^{n,m}$ is the number of rational terms in $\tilde{f}^{n,m}$, $c_l^{n,m}$ are the integer coefficients for each term, $p_l^{n,m}$ are the integer powers of r, which can be negative, $q_l^{n,m}$ are the integer powers of s, which can be negative. Thus, our final expression for the n^{th} basis function is:

$$\mathbb{T}^n_{ijk}(\mathbf{r},\mathbf{s}) = \frac{t}{N^n} \sum_{m=1}^{14} \Gamma^m_{ijk}(\tilde{\mathbf{r}},\tilde{\mathbf{s}}) \sum_{l=1}^{M^{n,m}} c_l^{n,m} \tilde{r}^{p_l^{n,m}} \tilde{s}^{q_l^{n,m}} \tag{4}$$

and the five basis functions are defined by specifying the integers N^n, $M^{n,m}$, $c_l^{n,m}$, $p_l^{n,m}$, $q_l^{n,m}$.

Using the procedure described in CM, these numbers defining the basis tensors were determined and are provided in the accompanying data file `basisdata.txt`, which can be read to allow evaluation of the tensor model using the Fortran 90 routines provided in `evalTijk.f90`. These routines ignore text (comments) on each line that follows an "!", which are provided in `basisdata.txt` to make the file more human readable. The data in the file is written with the following structure:

```
foreach n = 1 to 5
    N^n
    foreach m = 1 to 14
        M^{n,m}
        c_l^{n,m} for l = 1 to M^{n,m}
        p_l^{n,m} for l = 1 to M^{n,m}
        q_l^{n,m} for l = 1 to M^{n,m}
    end foreach m
end foreach n
```

where each line in the outline above that does not start with `foreach` or `end foreach` represents a line in the data file, while the `foreach` — `end foreach` pairs indicate structure that is repeated for the different values of the indicated variable. The `for` structure indicates quantities that are looped through on the same line.

In the Fortran 90 module `evalTijk.f90`, five user callable subroutines and functions are provided. They are as follows:

- `read_basis(datafile)` Reads the data describing the 5 basis tensors in the format described above. `datafile` is a string containing the name of the data file to read. The representation is stored in an internal data structure for later use in evaluating the correlations.
- `print_basis` Prints the basis characterization data to `stdout` in the format described above, though without comments. This is a debugging diagnostic.
- `build_composite(weights,normalize)` Constructs a representation for a linear combination of the 5 basis tensors specified by the weights stored in the double precision real array of 5 coefficients `weights`. The result is stored in an internal data

structure for later use in evaluation. The parameter `normalize` is a logical, which if true, will cause the weights to be normalized so that they sum to one, resulting in a correlation consistent with $\epsilon = 1$. If not normalized, the dissipation related to the correlation is the sum of the coefficients in `weights`. If an individual basis function is to be evaluated then `weights` should include all zeros, except for a one for the basis tensor to be evaluated.

- `print_composite` Similar to `print_basis`, prints the characterization of the composite in a format like that described above. The differences are that there is no loop in n and the values $c_l^{n,m}$ are reals rather than integers. This is primarily a debugging diagnostic.

- `eval_composite(i,j,k,r_,s_,checkprecision)` is a function returning the value of $\mathbb{T}_{ijk}(\mathbf{r},\mathbf{s})$ for the composite tensor defined by `build_composite`. `i, j, k` are integers, while `r_` and `s_` are three-dimensional double precision real vectors representing $\mathbf{r}$ and $\mathbf{s}$. The argument `checkprecision` is a logical. If true, `eval_composite` will return a `NaN` if the magnitude of the smallest separation vector is more than a factor of 100 smaller than the largest separation vector. This is useful because for such large aspect ratios, the evaluation of the model is subject to pollution by round-off errors. When this aspect ratio is less than 100, the round-off error will be less than a part in 10^5.

To use the routines to evaluate $\mathbb{T}_{ijk}(\mathbf{r},\mathbf{s})$, one would normally first call `read_basis` pointing it at `basisdata.txt`. Then one would call `build_composite` with coefficients determined from a fit of the correlations to appropriate data. For example, from CM, we would have:

```
double precision ::  weights(5)
weights = (/0.884,-2.692,-6.099,-5.853,14.760/)
```

An example program `eval_2pt_3or_tophat.f90` is used to compute the two-point third-order correlations of top-hat filtered (integrated over cubical volume) velocities, with separation in the direction normal to the cubical faces.

These routines are intended primarily as an example of how to use the description of the correlation model, and as a means of comparison. However, there are a number of ways in which these routines could be improved, in general, or for particular purposes. Two obvious examples are: 1) avoid the problem of large aspect ratio separations by using in this case an asymptotic representation valid when one separation is very small; 2) improve efficiency when computing all components of the tensor, by not recomputing the rational functions $\tilde{f}$ for each component. There are likely many other improvements. If you develop an improvement that might be of general interest, please forward it on to us. We will maintain an updated versions of the codes and data at `http://turbulence.ices.utexas.edu/Tijk.html`.