

# fastSF: A parallel code for computing the structure functions of turbulence

Shubhadeep Sadhukhan<sup>1</sup>, Shashwat Bhattacharya<sup>2</sup>, and Mahendra K. Verma<sup>1</sup>

<sup>1</sup> Department of Physics, Indian Institute of Technology Kanpur, Kanpur 208016, India <sup>2</sup> Department of Mechanical Engineering, Indian Institute of Technology Kanpur 208016, India

DOI: [10.21105/joss.02185](https://doi.org/10.21105/joss.02185)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Jed Brown](#) ↗

## Reviewers:

- [@cpgr](#)
- [@iljah](#)

Submitted: 06 February 2020

Published: 16 January 2021

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

Turbulence is a complex phenomenon in fluid dynamics involving nonlinear interactions between multiple scales. Structure functions are popular diagnostics in the study of statistical properties of turbulent flows (Frisch, 1995; Kolmogorov, 1941a, 1941b). Some of the earlier works comprising of such analysis are those of Gotoh et al. (2002), Ishihara et al. (2003), and Ishihara & Gotoh (2009) for three-dimensional (3D) hydrodynamic turbulence; Yeung et al. (2005) and Ray et al. (2008) for passive scalar turbulence; Biferale et al. (2004) for two-dimensional (2D) hydrodynamic turbulence; and Kunnen et al. (2008), Kaczorowski & Xia (2013), and Bhattacharya et al. (2019) for turbulent thermal convection. Structure functions are two-point statistical quantities; thus, an accurate computation of these quantities requires averaging over many points. However, incorporation of a large number of points makes the computations very expensive and challenging. Therefore, we require an efficient parallel code for accurate computation of structure functions. In this paper, we describe the design and validation of the results of fastSF, a parallel code to compute the structure functions for a given velocity or scalar field.

fastSF is a C++ application for computing the structure functions of scalar and vector fields on Cartesian grids of a 2D or 3D periodic box, stored as HDF5 files. The code employs MPI (Message Passing Interface) parallelization with equal load distribution and vectorization for efficiency on SIMD architectures. The user can select the range of the orders of the structure functions to be computed and the computed structure functions are written to HDF5 files that can be further processed by the user.

We are not aware of any other open source or commercial packages for computing structure functions; prior studies have relied on in-house software that was never publicly released. As an open source package, fastSF provides a standard high-performance implementation and thus facilitates wider use of structure functions.

fastSF uses MPI (Pacheco, 2011) for parallelism, HDF5 (Kozioł & Robinson, 2018) via H5SI (Chatterjee & Chatterjee, 2020) for reading gridded field data and writing structure functions, as well as blitz++ (Veldhuizen, 1998) for vectorized computation and yaml-cpp (Beder, 2019) for reading control parameters. In the next section, we will briefly explain the velocity and scalar structure functions in turbulent flows.

## Velocity and scalar structure functions

We denote the velocity and scalar fields using  $\mathbf{u}$  and  $\theta$  respectively. The velocity difference between any two points  $\mathbf{r}$  and  $\mathbf{r} + \mathbf{l}$  is  $\delta\mathbf{u} = \mathbf{u}(\mathbf{r} + \mathbf{l}) - \mathbf{u}(\mathbf{r})$ . The difference in the parallel

components of the velocity field along  $\mathbf{l}$  is  $\delta u_{\parallel} = \delta \mathbf{u} \cdot \hat{\mathbf{l}}$ . The corresponding difference in the perpendicular component is  $\delta u_{\perp} = |\delta \mathbf{u} - \delta u_{\parallel} \hat{\mathbf{l}}|$ . Assuming statistical homogeneity, we define the longitudinal velocity structure functions of order  $q$  as

$$S_q^{u_{\parallel}}(\mathbf{l}) = \langle (\delta u_{\parallel})^q \rangle = \langle [\{\mathbf{u}(\mathbf{r} + \mathbf{l}) - \mathbf{u}(\mathbf{r})\} \cdot \hat{\mathbf{l}}]^q \rangle, \quad (1)$$

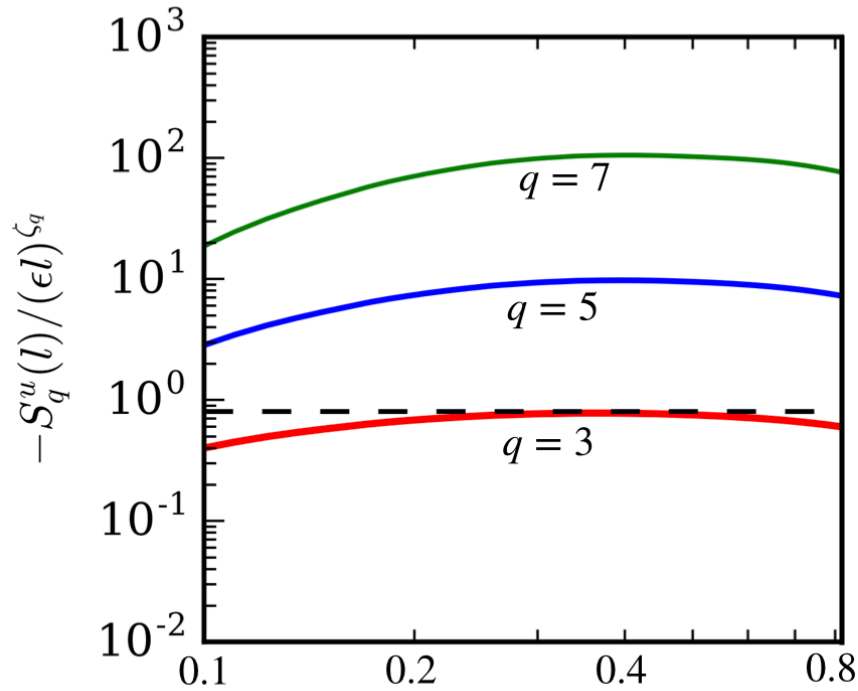
and the transverse velocity structure functions of order  $q$  as

$$S_q^{u_{\perp}}(\mathbf{l}) = \langle (\delta u_{\perp})^q \rangle = \langle |\delta \mathbf{u} - \delta u_{\parallel} \hat{\mathbf{l}}|^q \rangle. \quad (2)$$

Here,  $\langle \cdot \rangle$  denotes spatial averaging. Similarly, we can define the scalar structure functions for the scalar field as

$$S_q^{\theta}(\mathbf{l}) = \langle (\delta \theta)^q \rangle = \langle [\theta(\mathbf{r} + \mathbf{l}) - \theta(\mathbf{r})]^q \rangle. \quad (3)$$

For isotropic turbulence (in addition to being homogeneous), the structure functions become functions of  $l$ , where  $l = |\mathbf{l}|$ . The second-order velocity structure function  $S_2^{u_{\parallel}}(l)$  provides an estimate for the energy in the eddies of size  $l$  or less (Davidson, 2004).



**Figure 1:** For 3D homogeneous isotropic turbulence: plots of the negative of normalized third, fifth and seventh-order longitudinal velocity structure functions vs.  $l$ . The negative of the normalized third-order structure function is close to  $4/5$  (dashed line) in the inertial range.

For 3D incompressible hydrodynamic turbulence with homogeneity and isotropy, the third-order longitudinal velocity structure function in the inertial range (scales lying between the large-scale forcing regime and the small-scale dissipation regime) is given by (Frisch, 1995; Kolmogorov, 1941a, 1941b)

$$S_3^{u_{\parallel}}(l) = -\frac{4}{5} \epsilon l \sim -l, \quad (4)$$

where  $\epsilon$  is the viscous dissipation rate. For an arbitrary order  $q$ , She & Leveque (1994) proposed that the longitudinal structure functions scale as  $S_q^{u_{\parallel}}(l) \sim \zeta_q$ , where the exponent

$\zeta_q$  is given by

$$\zeta_q = \frac{q}{9} + 2 \left( 1 - \left( \frac{2}{3} \right)^{q/3} \right). \quad (5)$$

Figure~1 exhibits the plots of the negative of the normalized 3rd, 5th, and 7th-order longitudinal velocity structure functions computed using the simulation data of 3D hydrodynamic turbulence (Sadhukhan et al., 2019). The structure functions are normalized by  $(\epsilon l)^{\zeta_q}$ , where  $\zeta_q$  is given by Eq. (5). In the inertial range ( $0.2 < l < 0.7$ ), the normalized third-order longitudinal velocity structure function is fairly close to  $4/5$  (represented by dashed line), consistent with Kolmogorov's theory. Moreover, the normalized fifth and seventh-order structure functions show a plateau for the same range of  $l$ , thus exhibiting consistency with She-Leveque's model.

In the next section, we provide a brief description of the code.

## Design of the Code

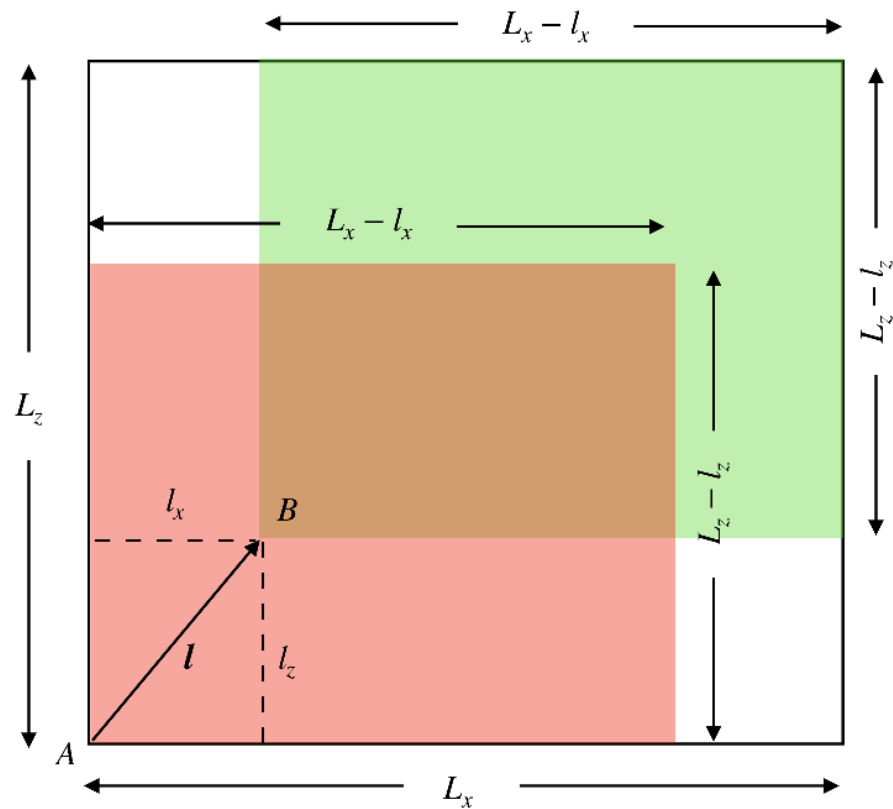
In this section, we present a sketch of the structure function computation for the velocity structure functions. We employ vectorization and loops over  $l$ , thus requiring three loops for 3D fields and two loops for 2D fields. In the following, we provide the algorithm for structure function computation for a 2D velocity field.

### Pseudo-code

*Data:* Velocity field  $\mathbf{u}$  in domain  $(L_x, L_z)$ ; number of processors  $P$ .

*Procedure:*

- Divide  $l$ 's among various processors. The process of data division among the processors has been described later in this section.
- For every processor:
  - for  $l = (l_x, l_z)$  assigned to the processor:
    - \* Compute  $\delta \mathbf{u}(l_x, l_z)$  by taking the difference between two points with the same indices in pink and green subdomains as shown in Fig. 2. This feature enables vectorized subtraction operation.
    - \*  $\delta u_{\parallel}(l_x, l_z) = \delta \mathbf{u} \cdot \hat{l}$  (Vectorized).
    - \*  $\delta u_{\perp}(l_x, l_z) = |\delta \mathbf{u} - \delta u_{\parallel} \hat{l}|$  (Vectorized).
    - \* for order  $q$ :
      - $S_q^{u_{\parallel}}(l_x, l_z) = \text{Average of } \delta u_{\parallel}^q$  (Vectorized).
      - $S_q^{u_{\perp}}(l_x, l_z) = \text{Average of } \delta u_{\perp}^q$  (Vectorized).
      - Send the values of  $S_q^{u_{\parallel}}(l_x, l_z)$ ,  $S_q^{u_{\perp}}(l_x, l_z)$ ,  $q$ ,  $l_x$ , and  $l_z$  to the root process.
- The root process stores  $S_q^{u_{\parallel}}(l_x, l_z)$  and  $S_q^{u_{\perp}}(l_x, l_z)$ .
- Stop



**Figure 2:** The velocity difference  $\delta u(\mathbf{l})$  is computed by taking the difference between two points with the same indices in the pink and the green subdomains. For example,  $\mathbf{u}(\mathbf{l}) - \mathbf{u}(0, 0) = \mathbf{u}_B - \mathbf{u}_A$ , where  $B$  and  $A$  are the origins of the green and the pink subdomains. This feature enables vectorization of the computation.

Since  $S_q^u(\mathbf{l})$  is important for intermediate scales (inertial range) only, we vary  $\mathbf{l}$  upto half the domain size, that is, upto  $(L_x/2, L_z/2)$ , to save computational cost. The  $\mathbf{l}$ 's are divided among MPI processors along  $x$  and  $z$  directions. Each MPI processor computes the structure functions for the points assigned to it and has access to the entire input data. After computing the structure function for a given  $\mathbf{l}$ , each processor communicates the result to the root process, which stores the  $S_q^{u_{\parallel}}(\mathbf{l})$  and  $S_q^{u_{\perp}}(\mathbf{l})$  arrays.

It is clear from Fig. 2 that the sizes of the pink or green subdomains are  $(L_x - l_x)(L_z - l_z)$ , which are function of  $\mathbf{l}$ 's. This function decreases with increasing  $\mathbf{l}$  leading to larger computational costs for small  $\mathbf{l}$  and less cost of larger  $\mathbf{l}$ . Hence, a straightforward division of the domain among the processors along  $x$  and  $z$  directions will lead to a load imbalance. Therefore, we assign both large and small  $\mathbf{l}$ 's to each processor to achieve equal load distribution. We illustrate the above idea using the following example.

Consider a one-dimensional domain of size  $L = 15$ , for which the possible  $\mathbf{l}$ 's are

$$\mathbf{l} = \{0, 1, 2, 3, \dots, 15\}.$$

We need to compute the structure functions for  $\mathbf{l}$  ranging from 0 to 7. We divide the task among four processors, with two  $\mathbf{l}$ 's assigned to each processor. The following distribution of

$l$ 's ensures equal load distribution:

$$\text{Processor 0: } l = \{0, 7\}, \quad \sum(L - l) = (15 - 0) + (15 - 7) = 23,$$

$$\text{Processor 1: } l = \{1, 6\}, \quad \sum(L - l) = (15 - 1) + (15 - 6) = 23,$$

$$\text{Processor 2: } l = \{2, 5\}, \quad \sum(L - l) = (15 - 2) + (15 - 5) = 23,$$

$$\text{Processor 3: } l = \{3, 4\}, \quad \sum(L - l) = (15 - 3) + (15 - 4) = 23.$$

Similarly, if two processors are used, then the following distribution results in load balance.

$$\text{Processor 0: } l = \{0, 7, 2, 5\},$$

$$\text{Processor 1: } l = \{1, 6, 3, 4\}.$$

This idea of load distribution has been implemented in our program and has been extended to higher dimensions.

Note that for 2D,  $l_x > 0$ , but  $l_z$  can take both positive and negative values. However, for isotropic turbulence, the structure functions for  $+l_z$  and  $-l_z$  are statistically equal. Therefore, in our computations, we restrict to  $l_x > 0$ ,  $l_z > 0$ . For anisotropic turbulence, not discussed here, the structure functions will depend on  $(l_x, l_z)$  rather than  $l$ ; our code will be extended to such systems in future.

For 3D turbulence, the structure functions will depend on  $(l_x, l_y, l_z)$ . We divide the tasks among processors over  $l_x$  and  $l_y$  as done above for 2D turbulence. The aforementioned algorithm can be easily extended to the 3D case. We employ a similar method for the computation of scalar structure functions as well.

In the next section, we discuss the scaling of our code.

## Scaling of fastSF

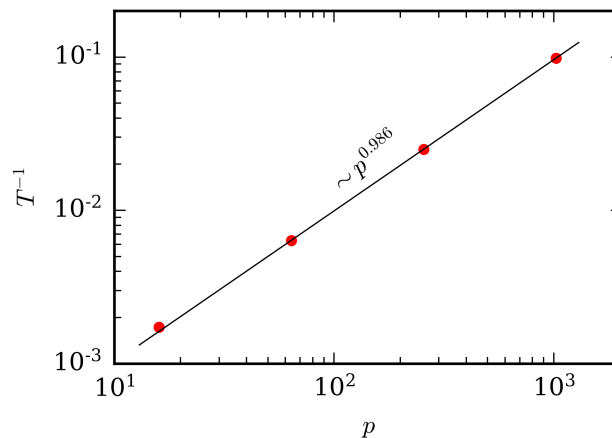
fastSF is scalable over many processors due to vectorization and equal load distribution. We demonstrate the scaling of fastSF for the third-order longitudinal structure function for an idealized velocity field on a  $128^3$  grid. For our computation we employ a maximum of 1024 cores. We take the velocity field as

$$\mathbf{u} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

We perform four runs on a Cray XC40 system (Shaheen II of KAUST) for this problem using a total of 16, 64, 256, and 1024 cores. We used 16 cores per node for each run. In Fig. 3, we plot the inverse of time taken in seconds versus the number of cores. The best fit curve for these data points yields

$$T^{-1} \sim p^{0.986 \pm 0.002},$$

Thus, the data-points follow  $T^{-1} \sim p$  curve to a good approximation. Hence, we conclude that our code exhibits good scaling.



**Figure 3:** Scaling of `fastSF` for the computation of third-order longitudinal velocity structure function using 16, 64, 256, and 1024 processors of Shaheen II. All the runs were conducted on a  $128^3$  grid. We observe a linear scaling.

## Conclusions

This paper provides a brief description of `fastSF`, an efficient parallel C++ code that computes structure functions for given velocity and scalar fields. This code is shown to be scalable over many processors. An earlier version of the code was used by [Bhattacharya et al. \(2019\)](#) for analyzing the structure functions of turbulent convection. We believe that `fastSF` will be useful to turbulence community as it facilitates wider use of structure functions.

## Acknowledgements

We thank Roshan Samuel, Anando Chatterjee, Soumyadeep Chatterjee, and Manohar Sharma for helpful discussions during the development of `fastSF`. We are grateful to Jed Brown, Ilja Honkonen, and Chris Green for a careful review of our work and their useful suggestions. Our computations were performed on Shaheen II at KAUST supercomputing laboratory, Saudi Arabia, under the project k1416.

---

## References

- Beder, J. (2019). *yaml-cpp*, A YAML parser and emitter for C++. [Computer Software] <https://github.com/jbeder/yaml-cpp>.
- Bhattacharya, S., Sadhukhan, S., Guha, A., & Verma, M. K. (2019). Similarities between the structure functions of thermal convection and hydrodynamic turbulence. *Physics of Fluids*, 31(11), 115107. <https://doi.org/10.1063/1.5119905>
- Biferale, L., Cencini, M., Lanotte, A. S., Sbragaglia, M., & Toschi, F. (2004). Anomalous scaling and universality in hydrodynamic systems with power-law forcing. *New Journal of Physics*, 6, 37. <https://doi.org/10.1088/1367-2630/6/1/037>

- Chatterjee, A., & Chatterjee, S. (2020). *H5SI*. [Computer Software] <https://github.com/anandogc/h5si>.
- Davidson, P. A. (2004). *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press. <https://doi.org/10.1093/acprof:oso/9780198722588.001.0001>
- Frisch, U. (1995). *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge University Press. <https://doi.org/10.1017/CBO9781139170666>
- Gotoh, T., Fukayama, D., & Nakano, T. (2002). Velocity field statistics in homogeneous steady turbulence obtained using a high-resolution direct numerical simulation. *Physics of Fluids*, 14(3), 1065–1081. <https://doi.org/10.1063/1.1448296>
- Ishihara, T., & Gotoh, T. (2009). Study of high-Reynolds number isotropic turbulence by direct numerical simulation. *Annual Review of Fluid Mechanics*, 41(1), 165–180. <https://doi.org/10.1146/annurev.fluid.010908.165203>
- Ishihara, T., Yokokawa, M., Itakura, K., & Uno, A. (2003). Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box. *Physics of Fluids*, 15(2), L21. <https://doi.org/10.1063/1.1539855>
- Kaczorowski, M., & Xia, K.-Q. (2013). Turbulent flow in the bulk of Rayleigh-Bénard convection: small-scale properties in a cubic cell. *Journal of Fluid Mechanics*, 722, 596–617. <https://doi.org/10.1017/jfm.2013.74>
- Kolmogorov, A. N. (1941a). Dissipation of Energy in Locally Isotropic Turbulence. *Doklady Akademii Nauk SSSR*, 32(1), 16–18. <https://doi.org/10.1098/rspa.1991.0076>
- Kolmogorov, A. N. (1941b). The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Doklady Akademii Nauk SSSR*, 30(4), 301–305. <https://doi.org/10.1098/rspa.1991.0075>
- Kozioł, Q., & Robinson, D. (2018). *HDF5*. [Computer Software] <https://doi.org/10.11578/dc.20180330.1>.
- Kunnen, R. P. J., Clercx, H. J. H., Geurts, B. J., Bokhoven, L. J. A. van, Akkermans, R. A. D., & Verzicco, R. (2008). Numerical and experimental investigation of structure-function scaling in turbulent Rayleigh-Bénard convection. *Physical Review E*, 77(1), 016302. <https://doi.org/10.1103/PhysRevE.77.016302>
- Pacheco, P. S. (2011). *An introduction to parallel programming*. Morgan Kaufmann. <https://doi.org/10.1016/C2009-0-18471-4>
- Ray, S. S., Mitra, D., & Pandit, R. (2008). The universality of dynamic multiscaling in homogeneous, isotropic Navier-Stokes and passive-scalar turbulence. *New Journal of Physics*, 10(3), 033003. <https://doi.org/10.1088/1367-2630/10/3/033003>
- Sadhukhan, S., Samuel, R., Verma, M. K., Stepanov, R., Plunian, F., & Samtaney, R. (2019). Enstrophy transfers in helical turbulence. *Physical Review Fluids*, 4, 84607. <https://doi.org/10.1103/PhysRevFluids.4.084607>
- She, Z.-S., & Leveque, E. (1994). Universal scaling laws in fully developed turbulence. *Physical Review Letters*, 72(3), 336–339. <https://doi.org/10.1103/PhysRevLett.72.336>
- Veldhuizen, T. L. (1998). Arrays in blitz++. *International Symposium on Computing in Object-Oriented Parallel Environments*, 223–230. [https://doi.org/10.1007/3-540-49372-7\\_24](https://doi.org/10.1007/3-540-49372-7_24)
- Yeung, P. K., Donzis, D. A., & Sreenivasan, K. R. (2005). High-Reynolds-number simulation of turbulent mixing. *Physics of Fluids*, 17(8), 081703. <https://doi.org/10.1063/1.2001690>