A statistics toolbox for turbulent pipe flow in Nek5000

Saleh Rezaeiravesh[1], Ricardo Vinuesa[1,2], and Philipp Schlatter[1,2]

[1] *Linné FLOW Centre, KTH Mechanics SE-100 44, Stockholm, Sweden*
[2] *Swedish e-Science Research Centre (SeRC), Stockholm, Sweden*
{*salehr, rvinuesa, pschlatt*}*@mech.kth.se*

November 23, 2019

1 Introduction

The main aim of this document is to detail the numerical simulation of statistically fully-developed turbulent pipe flow using Nek5000 [2] and also to explain how the turbulence statistics can be obtained. In the streamwise direction, z, periodic boundary condition is applied. Additionally, the flow in the azimuthal direction θ is statically homogeneous. Therefore, by averaging in z, θ, and time the resulting profiles (shown by $\langle\cdot\rangle$) are 1D and vary only with the radial (wall-normal) coordinate r. Note that the radial coordinate is defined from $r = 0$ (at the pipe center) to $r = R$ (at the wall), where R is the radius of the pipe. We also define the wall distance $y = R - r$.

In the next section, an overview is given on different directories included in the Nek5000 case for fully-developed pipe flow. In Section 3, the required steps to setup a simulation including generating the mesh are briefly explained. This is followed by Section 4 where the details for obtaining 1D profiles of turbulence statistics are provided. Finally in Section 5, a set of sample results are presented.

The algorithm to compute the statistics has been refined over the years, starting with the work by El Khoury *et al.* [3] and Noorani *et al.* [6] for straight and bent pipes, respectively. The actual routines are adapted from [9] and simplified for the one-dimensional statistics used in fully-developed pipe flow.

2 Nek5000 Case

The Nek5000 case for fully-developed turbulent pipe flow can be downloaded from the following repository:

`https://github.com/KTH-Nek5000/turbPipe`

The present implementation is tailored for Nek5000 V17, but it can be easily adapted also to V19. Once this version is officially released, we will also upgrade our repository.

The `turbPipe` case (referred to as `<main>` in this document) contains different folders:

- `/compile`, which includes:
  - `SIZE`: Size of arrays for Nek5000,
  - `turbPipe.usr`,
  - `/stats`: The folder contains the codes corresponding to the statistical toolbox developed by Vinuesa et al. [9]. The most updated version of the toolbox can be download from,

    `https://github.com/KTH-Nek5000/PerHillStats`

  - The make file `makenek` and compiling script, `compile_script`,
  - Controller of simulation restart, `chkpoint.f`,
  - Input-output tools, `IO_tools.f`,
  - `/inc_src` which contains inputs and definitions for different codes.
- `/run`: The run directory contains

  - `turbPipe.par`: The dictionary of parameters and controllers of the pipe flow simulation,
  - different data files which are produced during the course of simulations.

- `/postPrcs`: where the post-processing of the turbulence statistics is performed. The details are provided in Section 4.

3 Simulation Setup

For the pipe flow, the bulk and friction-based Reynolds numbers Re_b and Re_τ, respectively, are defined as:

$$\mathrm{Re}_b = \frac{U_b \cdot 2R}{\nu}, \qquad \mathrm{Re}_\tau = \frac{u_\tau R}{\nu}. \tag{1}$$

Here U_b and u_τ are the bulk and friction velocities respectively, and ν is the kinematic viscosity. To set up a pipe flow, we set the value of Re_b. Therefore, Re_τ is a result of the simulation. Some pipe-flow correlations, such as the ones by Blasius and Colebrook [5], can be used to obtain an estimation of the required Re_b to obtain a certain value of Re_τ. Furthermore, the values obtained from the direct numerical simulation (DNS) data by El Khoury et al. [3] are given in Table 1.

Table 1: Bulk and friction Reynolds numbers, as well as the ratio between friction and bulk velocity, taken from the DNS by El Khoury et al. [3].

Re_b	Re_τ	u_τ/U_b
5300	181	0.0683
11700	361	0.0617
19000	550	0.0579
37700	1000	0.0530

Therefore, to setup a pipe flow simulation at a given Re_b,

1. Choose R and U_b (typically 1) which set the reference quantities.
2. Set `viscosity` in `/run/turbPipe.par` to $-2U_bR/\mathrm{Re}_b$.

3.1 Periodic boundary condition and initialization

Since the flow in the streamwise direction is periodic, we need to fix the mean flow rate in subroutine `usrdat2` in `turbPipe.usr`:

```
param(54) = -3 !Streamwise direction (3:z) along which flow rate is fixed

param(55) = 1.0 != Ub
```

Initialization of the velocity field is implemented in subroutine `useric`. There are random perturbations which are added to the initial mean velocity components. In the radial and azimuthal directions, the mean velocity components u_r and u_θ are initialized to zero. For axial mean velocity u_z, one can choose to use either a parabolic laminar profile or a profile given by a law of the wall, such as the Reichardt law [7].

3.2 Grid generation

For generating elements over the pipe flow domain, any code or software can be used. Here we use `gmsh` [4]. To have full control over the geometry and mesh parameters, a `.geo` script that can be downloaded from the following repository has been used:

`https://github.com/KTH-Nek5000/PipeMesh`

Inside folder `/meshGen`, there is a sample 3D mesh file `turbPipe.msh` which can be viewed in `gmsh`. To convert this file to the `.re2` format, the tool `gmsh2nek` is used which can be downloaded from Ref. [1]. During the conversion, the periodicity in the streamwise direction should be specified via choosing the tag of the inlet and outlet faces assigned in the `.geo` script. As a result, the only boundary condition

that is needed to be specified in subroutine `usrdat2` in `turbPipe.usr` is the wall condition. Note that in addition to the 3D mesh for the main simulation (i.e. `turbPipe`), we need a 2D mesh of the pipe circular cross section in the post-processing stage as will be pointed out in Section 4. The 2D mesh should be exactly the same as the 3D mesh at any constant z and can be produced by the above-mentioned `.geo` script. Figure 1 illustrates two meshes which are generated for `Nek5000`.

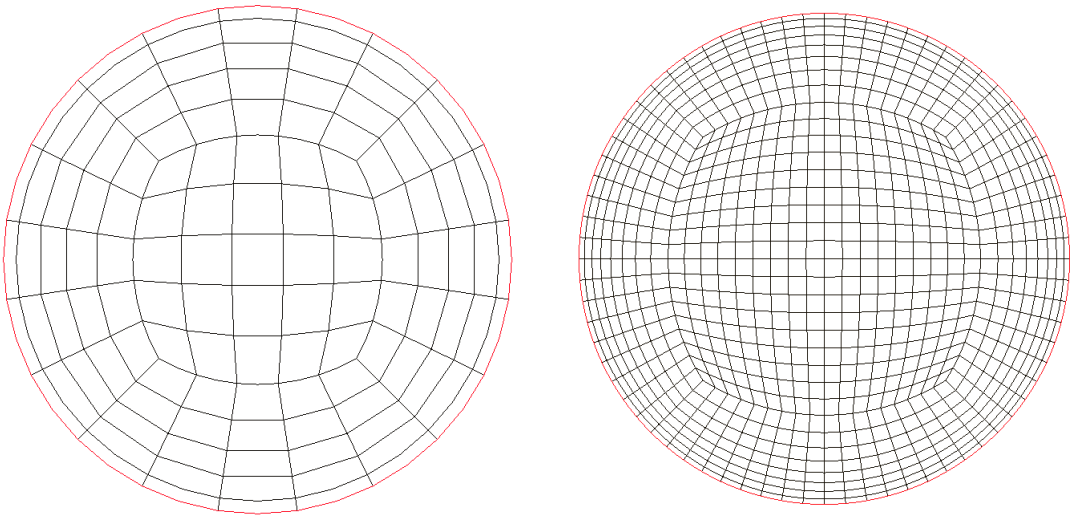

Figure 1: `Nek5000` elements over the pipe cross section for two different resolutions. Note that the quadrature points within the elements are not shown.

4 Post-processing the Simulations

As mentioned above, the toolbox described in [9] is used for gathering statistical data of the flow fields during the course of simulation. By default, averaging over time is considered. The frequency of time sampling is controlled via `userParam05` in `/run/turbPipe.par`. For the particular case of fully-developed pipe flow, the averaging in the axial direction z is also required. For this purpose, set `STAT3D=0` in `/compile/inc_src/STATS`. As a result of the on-the-fly averaging in time and z-direction a set of `ststurbPipe0.f*` files is produced in `/run`. The number of time steps between writing each of these files is controlled by `userParam06` in `/run/turbPipe.par`.

The rest of this section is devoted to describe in detail the steps required to obtain 1D profiles of the pipe flow statistics. The set of scripts used for this purpose are collected in the following three folders which are located in `/postPrcs`:

1. `/interpMesh`
2. `/ppNekOutData`
3. `/extractStats`

We need to follow this sequence of steps:

1. In `/interpMesh`, run[1] `interpMesh.m` to generate a 2D polar mesh over the circular cross section of the pipe. Hereafter, this mesh is referred to as the interpolating mesh. As shown in Fig. 2, the interpolating mesh is different from the mesh in the `Nek5000` simulations and is constructed so that it facilitates both averaging over θ and representing the final 1D profiles in the radial direction.

   Before running the script, set the grid specifications which include the pipe radius R, the distance from the wall of the first off-wall GLL (Gauss–Lobatto–Legendre) point in viscous units, Δr_w^+, and the number of mesh points in the radial and azimuthal directions, n_r and n_θ, respectively. Note that the viscous scaling is defined in terms of the friction velocity u_τ and the kinematic viscosity ν. Moreover, we need to specify kinematic viscosity ν, density ρ and nominal Re_τ (which is used

[1] Use either `GNU Octave` or `Matlab` to run the `.m` files.

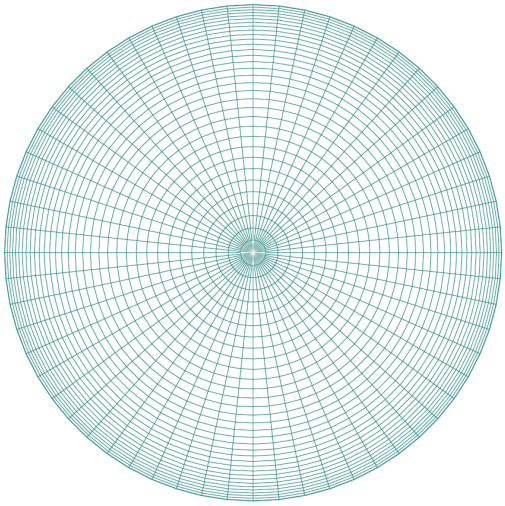

Figure 2: An interpolating mesh used to post-process the turbulence statistics.

only for finding Δr_w), exactly the same as the values in the main simulation. All these settings will be saved in `geo.mat` and will be used in the subsequent post-processing scripts.

By running `interpMesh.m`, the mesh coordinates $x = r\cos\theta$, $y = r\sin\theta$, where $0 \leq \theta \leq 2\pi$ and $0 \leq r \leq R$, along with the assigned settings are written in the following files:

`/postPrcs/ppNekOutData/ZSTAT/x.fort`

`/postPrcs/ppNekOutData/ZSTAT/y.fort`

`/postPrcs/extractStats/tempData/geo.m, angles.m`

Note that in the radial direction, the mesh coarsens from Δr_w toward the pipe center.

2. In `/ppNekOutData`, run `Nek5000` in post-processing mode to *(i)* concatenate the `ststurbPipe0.f*` files[2] and divide the gathered on-the-fly statistics by the elapsed averaging time, and, *(ii)* interpolate the resulting averaged fields to the interpolating mesh. We refer to this `Nek5000` case as `turbPipe_post`. Among the outputs (see below), we are only interested in `/ZSTAT/int_fld`.

   To attain the above targets,

   2(a) A 2D mesh over the pipe cross section is required. This 2D mesh is exactly the same as the cross section (at a constant z) of the 3D mesh used in the main simulation. Here we have used `gmsh` and then `gmsh2nek` to obtain `turbPipe_post.re2`.

   2(b) Adjust `SIZE`:

   - Set `ldim=2` and set `lelg` and `lelx` as the number of 2D elements (this can be read from the first line of `/turbPipe_post.re2`).
   - Make sure `lx1` (number of GLL points per element) and `lxd` are the same as those in `/<main>/compile/SIZE`.
   - Make sure `lhis` is large enough. This parameter specifies the maximum number of points in the interpolation mesh.

   2(c) Move `ststurbPipe0.f*` files from `/<main>/run` to `/ZSTAT`. We must ignore the `sts` files which contain the averaging data before completing transition to turbulence.[3] The time at the header of the last removed `sts` file is taken to be `mtimes` (needed below). After removing these files, manually re-number the remaining files starting from `ststurbPipe0.f00001`.

[2]Note that `sts` files are in `Nek5000` format and can be visualized (copy them to a separate folder, change the first line in `SESSION.NAME` to `ststurbPipe`, and run `visnek`).

[3]To inspect the flow development, the variation with time of the friction velocity averaged over the whole wall can be monitored. These data are written in the log file of the main simulation. For accurate evaluation of the wall friction velocity, the pipe radius and length `RAD` and `ZLENPIPE`, respectively, should be assigned at the beginning of `/compile/turbPipe.usr`. For more discussion on choosing the starting and ending times for averaging in numerical simulations of wall-bounded turbulence, see for instance Ref. [10].

2(d) In `turbPipe_post.par`,

- Set `numStep=0` to specify the post-process mode of `Nek5000`.
- Set `userParam05` to the frequency by which time-sampling was done in the main simulation (this parameter is the same as `userParam05` in `/<main>/run/turbPipe.par`).
- Set `userParam06` to the number of `sts` files which are considered.
- Set `userParam07` to `mtimes`, the time from which averaging of the turbulence statistics is started, excluding the initial transients.
- Set the density and viscosity as those in the main simulation.

2(e) Compile and run `Nek5000` for post-processing via `bash run_postprcs.sh`.

In summary, at the end of step 2 in `/postPrcs/ppNekOutData` we obtain the following files:

- `/ZSTAT/int_fld` which contains the fields averaged over time and z-direction and then interpolated over `x.fort` and `y.fort` (interpolating mesh). This file is written in binary format and is what we need for the next steps.
- `st1turPipe_post0.f00001`, `st2turbPipe_post0.f00001`, `UV_turPipe0.f00001`, and `WP_turbPipe0.f00001`: The same fields as in `int_fld` but in `Nek5000` format which can be visualized. There is no further use of these files except for the fact that they can be useful for checking the process.

3. Finally, in `/extractStats`, 1D turbulence statistics which are only dependent on the radial coordinate r are extracted. To this end,

(a) Run `turbPipe_stats.m` to compute turbulence statistics such as the mean velocity components, Reynolds stresses and their budgets. This script reads in `/ppNekOutData/ZSTAT/int_fld` and generates dataset `data_turbPipe_stat.mat` in `/tempData`. Here, one can plot the 2D contours of the time-z-averaged quantities over the 2D interpolating mesh.

(b) So far the (x, y, z) components of different statistical vector and tensor quantities are obtained on the interpolating mesh. By running `profs_turbPipe.m`, the vector fields in `/tempData/data_turbPipe_stat.mat` are transferred to the cylindrical coordinates (r, θ, z). Then by averaging over the azimuthal direction θ, the final 1D profiles of statistics in the radial direction are produced. These are written in different files in folder `/statsResults`.

Following the approach by Vinuesa et al. [8], here we shortly discuss coordinates transformation. For an arbitrary vector $\mathbf{A}$ we have,

$$\begin{bmatrix} A_r \\ A_\theta \\ A_z \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix},$$

where angle θ is measured with respect to the $+x$ axis in the counter-clockwise direction, so $0 \leq \theta \leq 2\pi$. The values of θ for the interpolation mesh are read from `/tempData/angles.m` produced by `../interpMesh/interpMesh.m`. The matrix on the right-hand-side of this expression is called rotation matrix $\mathbf{Q}$. If $\mathbf{A}$ is a second-order tensor (e.g. Reynolds stress tensor or velocity-gradient tensor), then the transformation reads as:

$$\begin{bmatrix} A_{rr} & A_{r\theta} & A_{rz} \\ A_{\theta r} & A_{\theta\theta} & A_{\theta z} \\ A_{zr} & A_{z\theta} & A_{zz} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} \mathbf{Q}^T.$$

5 Illustrative Example and Validation

The simulation case discussed in this tutorial can be run using `Nek5000` V17. The mesh has been generated by `gmsh` and transferred to `.re2` format. The corresponding files for the 2D mesh used in post-processing are available in `/postPrcs/ppNekOutData`. The 3D mesh has 1875 elements and is shown in Fig. 1 (left). Because of the relatively small number of degrees of freedom, the simulation can be performed on a local machine using 4–8 processors.
To compile and run the case:

1. Choose the compilers and also set the path to the `Nek5000` directory in:

   `/compile/makenek`

   `/compile/compile_script`

   `/postPrcs/ppNekOutData/makenek`

2. Compile and run the main simulation:

   In `/compile`, compile via `./compile_script --all`

   In `/run`, run by `bash runBash_local.sh` (after specifying the number of processors)

3. Compile and run `Nek5000` in post-processing mode:

   In `/postPrcs/ppNekOutData/` run `bash run_postprcs.sh`.

To validate the whole process, the same set of codes are used to simulate a fully-developed pipe flow at $\mathrm{Re}_\tau = 360$ over a fine mesh which is illustrated in Fig. 1 (right). This case has 37800 elements and the number of GLL points per element is 8^3. The pipe radius R and bulk velocity U_b are set to unity. The elements in the radial direction are constructed so that $\Delta r_w^+ = 0.5$. In the z-direction, the pipe length is $8.167\,R$ which is discretized by 42 equi-spaced elements. Therefore, the average distance between the GLL points within the elements in this direction is $\Delta z^+ = 10$. At the wall, the average distance between the GLL points in the azimuthal direction is $(R\Delta\theta)_w^+ \approx 4.5$. Discarding the initial transients, see Ref. [10], the statistical data are gathered over 46 flow-through times, corresponding to 376 convective time units. Note that the actually relevant number is eddy-turnover times, rather than flow-through or convective units.

For this simulation, Fig. 3 shows the contours of instantaneous streamwise velocity. Fig. 4 represents the excellent agreement of the computed profiles of the first- and second-order statistics of velocity, as well as the budget terms of the turbulent kinetic energy $k = (\langle u_r'^2\rangle + \langle u_z'^2\rangle + \langle u_\theta'^2\rangle)/2$ and Reynolds stress $\langle u_r' u_z'\rangle$ with the reference DNS data by El Khoury et al. [3].

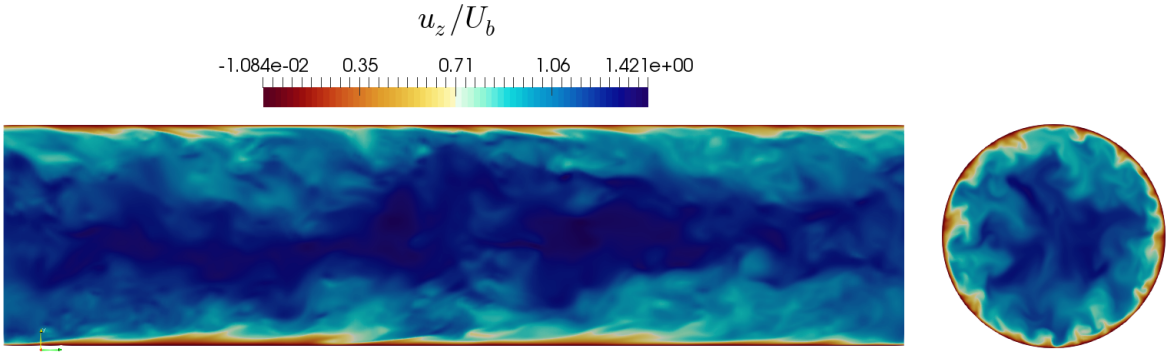

Figure 3: Contours of instantaneous streamwise velocity of a fully-developed turbulent pipe flow at $\mathrm{Re}_\tau = 360$ simulated using a grid of resolutions $\Delta r_w^+ = 0.5$, $\Delta z^+ = 10$ and $(R\Delta\theta)_w^+ = 4.5$.

6 Acknowledgement

Financial support by the Linné FLOW Centre for SR is gratefully acknowledged. The simulation shown in Figs. 3 and 4 was performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at HPC2N, Umeå University, Sweden.

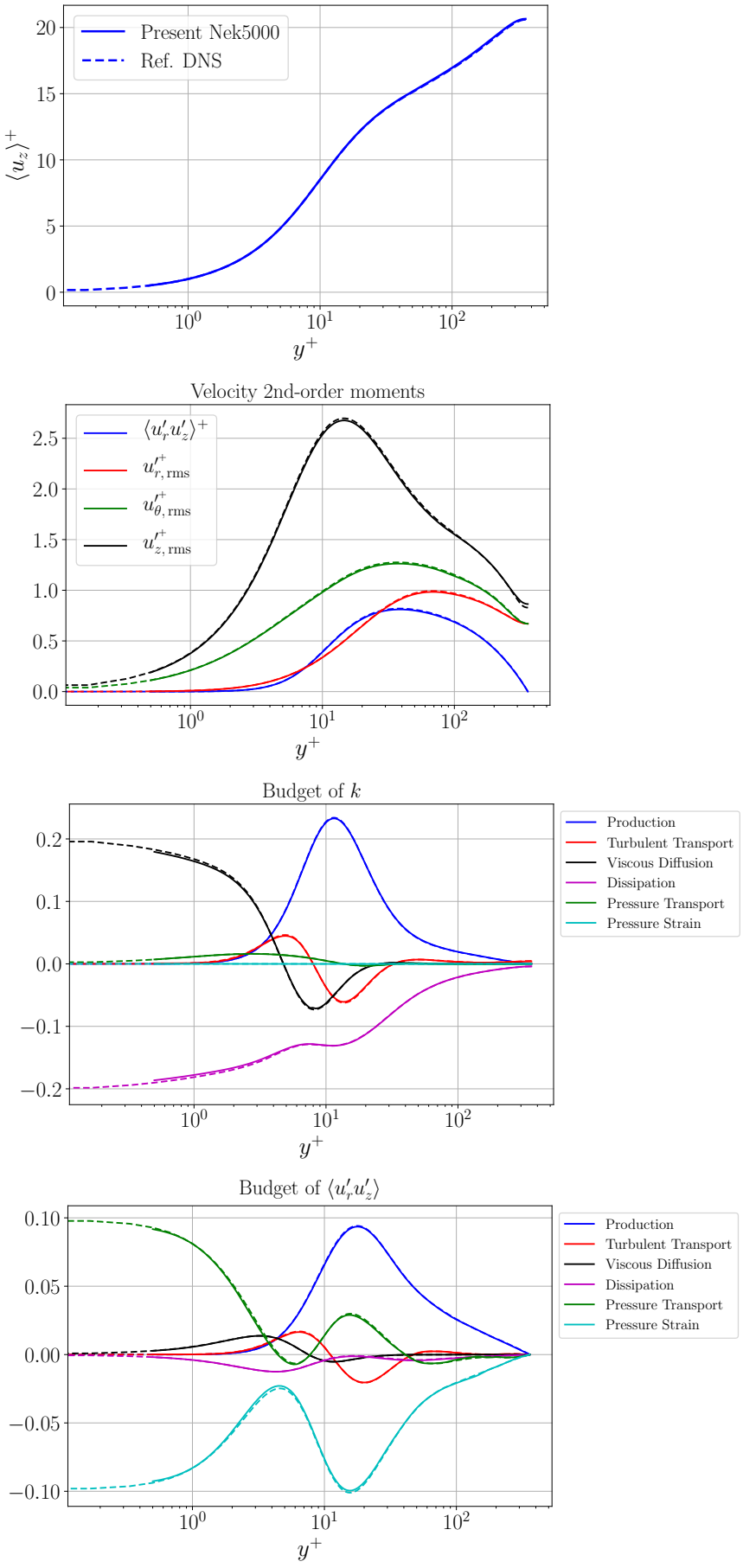

Figure 4: Turbulence statistics (in viscous units) for the pipe flow shown in Fig. 3: present `Nek5000` simulations (–), reference DNS data by El Khoury et al. [3] (--). Note that $y = R - r$.

References

[1] gmsh2nek: convert gmsh .msh (version 2, 2d/3d ascii/binary mesh) to Nek .re2 file. `https://github.com/yhaomin2007/Nek5000/tree/master/gmsh2nek_sourcecode/gmsh2nek`.

[2] Nek5000 Version 17. Argonne National Laboratory, Illinois. `https://nek5000.mcs.anl.gov`.

[3] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, and A. V. Johansson. Direct numerical simulation of turbulent pipe flow at moderately high Reynolds numbers. *Flow Turbul. Combust.*, 91(3):475–495, July 2013.

[4] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Num. Meth. Engng.*, 79(11):1309–1331, 2009.

[5] L. Moody. Friction factors for pipe flow. *Trans. ASME*, 66:671–684, 1944.

[6] A. Noorani, G. K. El Khoury, and P. Schlatter. Evolution of turbulence characteristics from straight to curved pipes. *Int. J. Heat Fluid Flow*, 41:16–26, 2013.

[7] V. H. Reichardt. Vollständige Darstellung der turbulenten Geschwindigkeitsverteilung in glatten Leitungen. *Z. angew. Math. Mech.*, 31(7):208–219, 1951.

[8] R. Vinuesa, S. M. Hosseini, A. Hanifi, D. S. Henningson, and P. Schlatter. Pressure-gradient turbulent boundary layers developing around a wing section. *Flow, Turbulence and Combustion*, 99(3):613–641, 2017.

[9] R. Vinuesa, A. Peplinski, M. Atzori, L. Fick, O. Marin, E. Merzari, P. Negi, A. Tanarro, and P. Schlatter. Turbulence statistics in a spectral-element code: a toolbox for high-fidelity simulations. Technical Report TRITA-SCI-RAP 2018:010, KTH Royal Institute of Technology, Sweden, 2018.

[10] R. Vinuesa, C. Prus, P. Schlatter, and H. M. Nagib. Convergence of numerical simulations of turbulent wall-bounded flows and mean cross-flow structure of rectangular ducts. *Meccanica*, 51(12):3025–3042, 2016.