SoftwareX 11 (2020) 100395

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

nsCouette – A high-performance code for direct numerical simulations of turbulent Taylor–Couette flow

Jose Manuel López [a,*], Daniel Feldmann [b,*], Markus Rampp [c], Alberto Vela-Martín [d], Liang Shi [e], Marc Avila [b]

[a] Institute of Science and Technology Austria, Am Campus 1, 3400 Klosterneuburg, Austria
[b] University of Bremen, Center of Applied Space Technology and Microgravity (ZARM), Am Fallturm 2, 28359 Bremen, Germany
[c] Max Planck Computing and Data Facility (MPCDF), Gießenbachstraße 2, 85748 Garching, Germany
[d] School of Aeronautics, Universidad Politécnica de Madrid, Plaza del Cardenal Cisneros 3, 28040 Madrid, Spain
[e] Max Planck Institute for Dynamics and Self-Organization (MPIDS), Bunsenstraße 10, 37073 Göttingen, Germany

ARTICLE INFO

Article history:
Received 8 August 2019
Received in revised form 2 December 2019
Accepted 26 December 2019

Keywords:
Wall-bounded turbulence
Rotating shear-flow
Thermal convection
Direct numerical simulation (DNS)
Hybrid parallelization
GPU

ABSTRACT

We present nsCouette, a highly scalable software tool to solve the Navier–Stokes equations for incompressible fluid flow between differentially heated and independently rotating, concentric cylinders. It is based on a pseudospectral spatial discretization and dynamic time-stepping. It is implemented in modern Fortran with a hybrid MPI-OpenMP parallelization scheme and thus designed to compute turbulent flows at high Reynolds and Rayleigh numbers. An additional GPU implementation (C-CUDA) for intermediate problem sizes and a version for pipe flow (nsPipe) are also provided.

Code metadata

Current code version	1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_225
Legal Code License	GPLv3
Code versioning system used	git
Software code languages, tools, and services used	Fortran, C (for some housekeeping tasks), MPI, OpenMP
Compilation requirements, operating environments & dependencies	Developed and tested under Linux and IBM AIX. Compiler: A Fortran 2003 compiler which is OpenMP-3 compliant, a basic C compiler, an MPI library with support for MPI_THREAD_SERIALIZED, a serial BLAS/LAPACK library, a serial but fully thread-safe FFTW-3 installation or equivalent, for output and visualization (optional): an MPI-parallel HDF5 installation.
Link to developer documentation/manual	https://gitlab.mpcdf.mpg.de/mjr/nscouette
Support email for questions	nsCouette@zarm.uni-bremen.de

1. Motivation and significance

Flows in engineering and nature are often characterized by large Reynolds (*Re*) or Rayleigh (*Ra*) numbers. Examples are the flow of gas in astrophysical disks, atmospheric flows and the cooling of rotating machines. In most cases, it is impossible to resolve all scales of the turbulent flow in a direct numerical simulation (DNS). However, DNS provide reliable data to allow extrapolation to the large *Re* limit and to enable the development of adequate subgrid-scale models. Taylor–Couette (TC) flow – the flow between two independently rotating concentric cylinders – stands out as a testbed for these purposes [1,2]. It allows exploring a variety of physical mechanisms, including buoyancy, shear, rotation and boundary layers in the vicinity of curved walls.

* Corresponding authors.
E-mail addresses: jlopez@ist.ac.at (J.M. López), daniel.feldmann@zarm.uni-bremen.de (D. Feldmann).

https://doi.org/10.1016/j.softx.2019.100395

Our DNS code nsCouette integrates the incompressible Navier–Stokes equations for TC flow forward in time using cylindrical coordinates and primitive variables. Optionally, the cylinder walls can be differentially heated, in which case an additional equation for the temperature is solved. The goal of this paper is to make nsCouette publicly available and thus enable DNS of rotating turbulent shear flows to a wide range of users in the mathematics, physics and engineering communities.

2. Software description

2.1. Functionality

In nsCouette, the governing equations are discretized using a pseudospectral Fourier–Galerkin ansatz for the azimuthal (θ) and the axial (z) direction. High-order finite differences (FD) are used in r; the only inhomogeneous direction. The user can select the distribution of radial grid points at runtime and the stencil-length of the FD scheme is specified at compile time with a default of nine points.

Periodic boundary conditions (BC) are assumed in z to avoid the need for dense grids close to the vertical boundaries. The comparison between DNS with axially periodic BCs and laboratory experiments with solid end-plates is extremely satisfactory for a wide range of Re from laminar to highly turbulent flows [2]. Additionally, z-periodicity often provides a more accurate model of astrophysical and geophysical flows and prevents misleading physical interpretations due to undesired end-wall-effects [3,4]. Details of the method and implementation are published in [5].

The temporal integration scheme has been upgraded to a predictor–corrector method [6]. This enables a variable time-step size with dynamic control, which is of advantage if the flow state is suddenly modified (applying disturbances, changing rotation rates) or naturally undergoes strongly transient dynamics.

Another significant upgrade is the extension to heat transfer where a temperature difference between the cylinders is imposed. To this end, a Boussinesq-like approximation [7] has been implemented to account for buoyancy effects. In the distributed version of nsCouette, a negative temperature gradient in r is considered, whereas gravity is aligned in z. Other scenarios can be easily investigated by changing only a few lines of source code.

Additionally, divergence-free initial conditions can be used to easily excite selected Fourier modes. This enables the user to systematically investigate different transition scenarios.

A single input file defines all relevant parameters (number of points, modes and timesteps, rotation rates etc.) at runtime. At every restart, the spatial resolution can be changed using an automated interpolation and mode padding functionality.

This and many other convenient features, together with a number of example `Makefiles` for the most common high-performance computing (HPC) platforms, provide newcomers an easy start into the world of highly-resolved and massively-parallel DNS. A user guide is included to help non-expert users get started with nsCouette (compilation, setup, select proper resolution, analyze data, etc.).

2.2. Software architecture

Over time, nsCouette has been ported to all major CPU-based HPC platforms. Amongst IBM Power, BlueGene and `x86_64` architectures – including a few generations of the prevalent Intel Xeon multi-core processors – it has also been ported to Xeon Phi (KnightsLanding), AMD EPYC (Naples, Rome) and ARMv8.1 (Marvell ThunderX2) platforms. Developments for the NEC SX-Aurora vector architecture and multi-GPU clusters are underway.

Building the executable requires a modern Fortran compiler, a standard C compiler and only very few additional libraries: MPI, BLAS/LAPACK, FFTW and optionally HDF5. All of them are commonly available as high-quality, open-source software (e.g. GCC, OpenMPI [8], FFTW [9], OpenBLAS) and as vendor-optimized tool chains (e.g. Intel Parallel Studio `PSXE`).

Our code runs on laptops and – for large-enough problems – efficiently scales up to the largest HPC systems with tens of thousands of processor cores [10]. The basic architecture of nsCouette and our design choice for a hybrid MPI-OpenMP parallelization scheme (HPS) follows straightforwardly from the Fourier–Galerkin ansatz and from the current technological trend of multi-core processors with ever increasing core counts and stagnating per-core performance. The basis of our HPS is a one-dimensional (1d) MPI-only slab decomposition into Fourier modes, which can be treated independently of each other in the computation of the linear terms occurring in the governing equations. For computing the non-linear terms, global data transpositions (`MPI_Alltoall`) and task-local transposes are employed to gather all modes locally on each MPI task. Therefore, the number of radial grid points (N_r), which is typically much smaller than the number of Fourier modes, imposes a natural limit to the 1d MPI-only approach. We relax this limit by introducing an additional parallelization layer, such that multiple cores per MPI task will be allocated to accommodate multiple OpenMP threads. First, this allows to compute the linear terms in OpenMP-parallel loops over Fourier modes. Second, an OpenMP-coarse-grain parallelism can be exploited during the computations of the non-linear terms by overlapping the global transpositions and the Fourier transformations of state variables. Overall, when confronted with a 2d MPI-only domain decomposition (e.g. [11]), which might be superior for certain setups, our HPS is an excellent compromise between versatility, simplicity of the implementation and achievable peak parallel scalability.

Additionally, we provide a basic GPU version of nsCouette written in C-CUDA, which implements the exact same numerical schemes as the Fortran version, but currently provides less output and functionalities. These will be added in the future. A CUDA-capable GPU device with compute capability 2.0 (or higher), support for double-precision arithmetic, and NVIDIA's CUDA toolkit are required. The GPU version runs on single GPU devices in a massively parallel setup with thousands of GPU threads and highly efficient memory management. It relies on custom CUDA kernels for linear algebra, uses the highly-optimized `cuFFT` library to perform Fourier transforms without the need for inter-node communication and, therefore, provides a clear speedup with respect to the Fortran code for small problems that fit into the main memory (RAM) of one GPU.

2.3. Computational performance

The number of MPI tasks can be selected at program start-up and changed between runs with the only restriction that it must divide N_r. The HPS of nsCouette achieves a scalability well beyond the limitations imposed by 1d MPI-only approaches and maps naturally to the prevailing multi-node, multi-core HPC architectures. Specifically, the flexibility to choose appropriate combinations for the numbers of MPI tasks per node and OpenMP threads per MPI task has proven key for achieving good performance across a wide range of architectures from 16 to 64 cores per node. On such machines, nsCouette enables highly resolved DNS with $N_r = \mathcal{O}(10^3)$ using $\mathcal{O}(10^4)$ cores [5]. In absolute terms, for an intermediate problem size with $N_r = 512$ and 513×1025 Fourier modes (i.e. Re up to $\mathcal{O}(10^4)$), the computation of a single timestep takes less than a second on 64 nodes (2560 cores) of a contemporary HPC cluster (Fig. 1) and requires

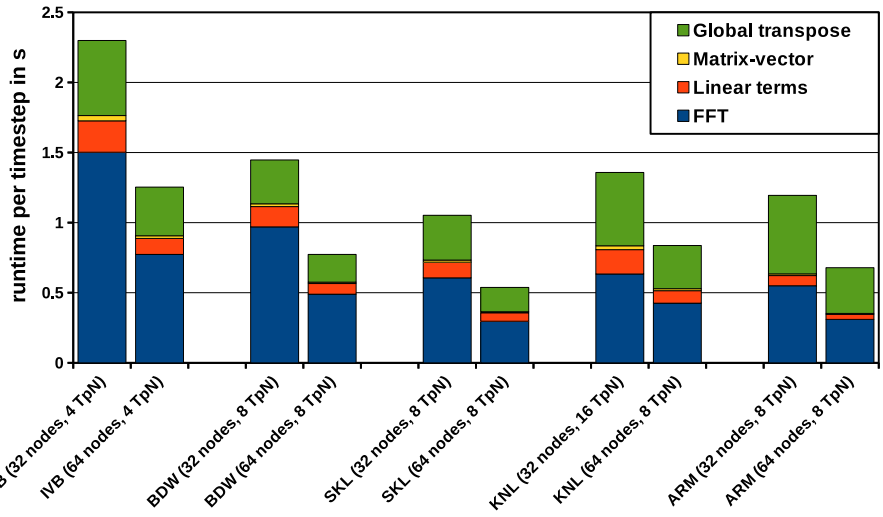

Fig. 1. Runtime per timestep and breakdown into the main algorithmic components (different colors) of a typical `nsCouette` run ($N_r = 512$ and 513×1025 Fourier modes) computed on 32 and 64 dual-socket nodes of various HPC clusters, using a platform-specific number of `MPI` tasks/node (TpN). IVB: Intel Xeon E5-2680v2 (IvyBridge), 20 cores/node. BDW: Intel Xeon E5-2698v4 (Broadwell), 40 cores/node. SKL: Intel Xeon 6148 (Skylake), 40 cores/node. KNL: Intel Xeon Phi 7230 (KnightsLanding), 64 cores/node. ARM: Marvell ThunderX2 ARM v8.1, 64 cores/node. The IVB and BDW clusters employ a Mellanox InfiniBand FDR network (56 Gbit s^{-1}), whereas SKL and KNL use Intel OmniPath (100 Gbit s^{-1}). The ARM cluster is interconnected with Cray Aries (80 Gbit s^{-1}). `nsCouette` was built using platform-optimized software tool chains (i.e. compilers and libraries) but no platform-specific optimization of the source code was performed. Corresponding `Makefiles` are shipped with the code. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

roughly 450 GB of RAM. On the SKL platform, this run achieves a performance of 1.5 TFlop/s, which, due to a rather moderate arithmetic intensity of the algorithm (0.3), is bounded by the memory bandwidth. When increasing the number of cores for a fixed problem size, the computations of the FFT and linear terms show very good strong scalability, whereas the global transposes ultimately limit the total parallel efficiency at large core counts (Fig. 1). A comprehensive study of the parallel scalability and efficiency of `nsCouette` has been presented in [5], and its potential to scale up to extremely high core counts was shown in [10]. The upgraded version presented here exhibits the same scalability and maintains consistent performance over a range of different HPC systems without the need for any platform-specific adaptations of the source code.

The performance of the GPU-accelerated version was tested on two NVIDIA graphics cards based on the Volta architecture: Titan V and Tesla V100 (Fig. 2). The runtime per timestep has been found to be similar on both cards over a wide range of problem sizes. The speed-up of the GPU version compared to the HPS version running on one node was shown to vary between a factor of three and 17, depending on the problem size and the particular choice of platform used as reference. Comparing a single GPU run against an `MPI-OpenMP` run on a single CPU node is a reasonable choice, since for server-class hardware both setups are roughly comparable in terms of price and electrical power consumption. However, 16 nodes (256 cores) were necessary to outperform the GPU version for small problems. Currently, the maximum problem size applicable to the GPU version is limited by the amount of RAM available on the graphics card.

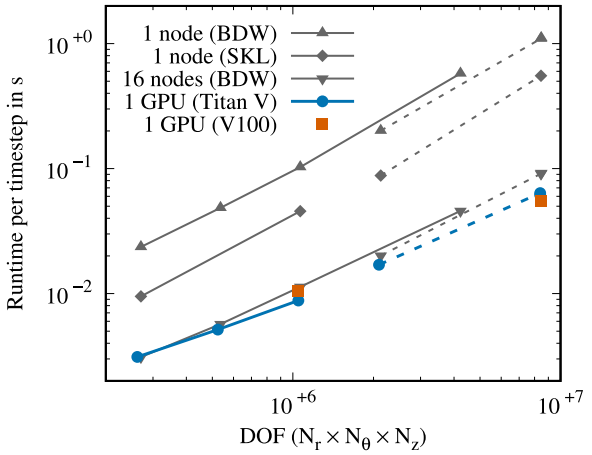

Fig. 2. Performance of the GPU-accelerated version of `nsCouette` compared to the `MPI-OpenMP` version. Runtime per timestep for different numbers of degrees of freedom (DOF). The GPU code ran on a single NVIDIA Titan V and a single Tesla V100 graphics card. It was built using NVIDIA's `CUDA` toolkit version 10.1. The hybrid code was built using Intel's `PSXE2018` and ran on one and 16 nodes of different platforms. BDW: Intel Xeon E5-2620v4 (Broadwell), 16 cores/node, Mellanox InfiniBand FDR network (56 Gbit s^{-1}). SKL: Intel Xeon 6148 (Skylake), 40 cores/node. Solid (dashed) lines represent runs with $N_r = 64$ ($N_r = 128$) radial points.

2.4. Data analysis and visualization

In `nsCouette`, the spectral coefficients and the primitive variables are dumped to individual files for each timestep at user-specified output intervals. It implements an easy-to-use checkpoint-restart mechanism based on the coefficients for handling long-running DNS. The primitive variables – velocity (u_r, u_θ, u_z), pressure (p) and optionally temperature (T) – are written in `HDF5` format, along with metadata in small `xdmf` files in order to facilitate analysis with common visualization software like `ParaView` and `VisIt`. Both tools allow loading sequences of `xdmf` files produced by `nsCouette`. Sample scripts based on the `Python` interface of `VisIt`, as well as a custom-made `ParaView` filter for handling the cylindrical coordinate system are distributed with the code. A detailed visualization tutorial is included in the user guide. This enables the user to easily perform comprehensive visual and quantitative analysis of the flow field.

2.5. Quality assurance

Verification and validation (V&V) of `nsCouette` is documented in [5]. For maintaining the correctness of the source code, we make extensive use of the continuous integration (CI)

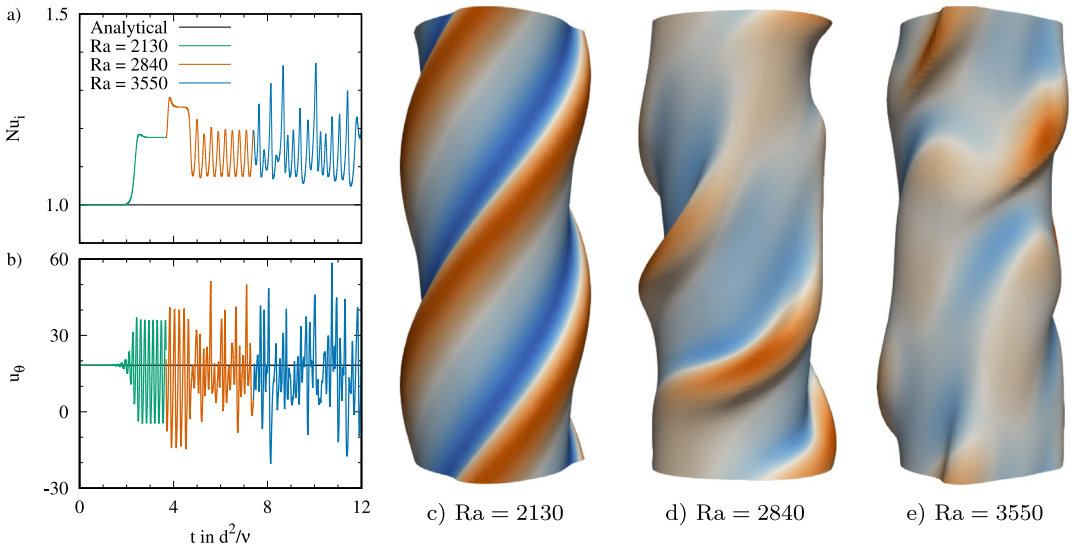

Fig. 3. Temporal evolution of the Nusselt number (Nu_i) at the inner cylinder wall and the streamwise velocity component (u_θ) at a mid-gap position as the Rayleigh number (Ra) increases for a fixed inner cylinder rotation ($Re_i = 50$). The final flow state for each Ra is visualized with instantaneous temperature iso-surfaces ($T = 0$), which are color-coded by inwards/outwards (blue/red) facing values of the wall-normal velocity component (u_r). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

functionality of `gitlab`. Upon every push to the repository, a number of regression tests are automatically triggered, including builds of the code in various configurations and a static code analysis using the `Forcheck` tool. In addition, a number of short test runs are automatically launched using runtime-checks and the tightest debug settings of the compiler to identify undefined variables, out-of-bounds errors and alike. The numerical results are then rigorously verified against previously recorded reference runs. A final validation run compares the wave speed of a simulated wavy vortex flow with an experimentally determined value [12], which is considered successful if the wave speeds match up to 10^{-4}. The entire CI configuration, the V&V results for every push, and an auto-generated source code documentation (`Ford`) – including dependencies and call graphs – are publicly accessible through the web interface of our development site.

3. Illustrative example

The fluid flow between a hot rotating cylinder and a cooled stationary enclosure is a simple model to investigate heat transfer in many engineering applications [13] such as cooling of rotating machinery [14]. At low angular speeds (Re_i) and small temperature differences (Ra), the heat transfer is purely conductive. In this simple case, termed basic state, the governing equations admit analytic solutions for u_θ and T, which only depend on r. The heat transfer can be enhanced by either increasing Re_i(forced convection) or by increasing Ra (natural convection). In both cases, the basic state exhibits a sequence of distinct instabilities ultimately leading to turbulent heat transfer [15]. A measure of the efficiency is given by the Nusselt number (Nu_i), which is the ratio of total heat transfer at the inner cylinder, normalized by that of the basic state at the same temperature difference.

Fig. 3 summarizes the results of three DNS with increasing temperature difference at a fixed rotation rate, using `nsCouette`. The first DNS was initialized by applying small single harmonic disturbances to the basic state at $Re_i = 50$ and $Ra = 2130$. Fig. 3a shows that initially $Nu_i \approx 1$, corresponding to purely conductive heat transfer. However, after roughly two viscous time units, a sharp increase of $Nu_i$is observed, indicating that the basic state has become unstable. This is confirmed by the time-series of u_θ at a fixed probe location in the computational domain (Fig. 3b). The final state of this run is visualized in Fig. 3c. It shows a three-dimensional rendering of a $T = 0$ iso-surface generated with `ParaView`. The spiral flow pattern rotates at a constant speed without changing its shape; like a barber pole. It passes repeatedly through the probe location, what explains why the u_θ signal reaches a periodic state ($2.5^{d^2}/_{\nu} > t > 3.5^{d^2}/_{\nu}$). Its constant shape explains why the integral heat flux (Nu_i), on the other hand, remains constant at the same time. The second and third DNS were initialized with the final state of the former runs and by increasing the Rayleigh number to $Ra = 2840$ and 3550, respectively. The time series and final states in Fig. 3 reveal, that the TC system undergoes a sequence of transitions to different flow states with increasing spatio-temporal complexity as Ra increases. These, and other illustrative examples, are documented in the tutorial section of the user guide.

4. Impact

Our software completes the list of publicly available Navier–Stokes solvers for the three most common prototypes of wall-bounded shear flow: plane Couette flow (`channelflow` [16]), pipe flow (`openpipeflow` [11]) and Taylor–Couette flow (`nsCouette`). It can be quickly installed and productively used by researchers interested in pattern formation and chaos, for which TC flow has long been a paradigm [17]. The example of Section 3 can be run in a laptop and is meant to illustrate how easy results can be obtained and analyzed.

We however stress that the main aim of providing `nsCouette` is to make the full potential of a massively-parallel and highly-scalable DNS code readily usable and, therefore, valuable for the broad scientific community. It enables users with little experience in HPC and DNS to easily perform highly-resolved simulations of turbulence. Among other software design choices, this was achieved by providing a user friendly build process that supports many pre-configured HPC architectures, easy runtime handling of all control parameters, tailored visualization tools, a comprehensive user guide and thorough internal quality assurance. Thus, `nsCouette` is a powerful tool to study highly-turbulent shear flows. For example, it has already contributed to a better understanding of astrophysical [18] and geophysical [19,20] flows.

Because of its modular structure, which closely follows the numerical formulation, and its moderate code complexity (in particular the HPS), new functionalities can be easily added to `nsCouette`, with given algorithmic domain knowledge and basic programming skills. As an example for this, we also provide `nsPipe`; a modified version to simulate flows in straight pipes. It follows the numerical formulation of `openpipeflow` [11] and uses the parallel infrastructure of `nsCouette`, as detailed in the user guide. Extensions including the modeling of polymer additives [21] and two-phase flows [22] have already been developed and tested and will be released in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Max Planck Society and partially funded by the German Research Foundation (DFG) through the priority programme Turbulent Superstructures (SPP1881). AVM was supported by the European Research Council (ERC) through the *COTURB* project (ERC-2014.AdG-669505). Computational resources provided by the following institutions are gratefully acknowledged: The Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility (DE-AC02-06CH11357). The Isambard UK National Tier-2 HPC Service operated by GW4 and the UK Met Office, and funded by EPSRC (EP/P020224/1). Further computations were performed on the HPC systems Hydra, Draco and Cobra at the MPCDF in Garching.

References

[1] Bazilevs Y, Akkerman I. Large eddy simulation of turbulent Taylor–Couette flow using isogeometric analysis and the residual-based variational multiscale method. J Comput Phys 2010;229(9):3402–14. http://dx.doi.org/10.1016/j.jcp.2010.01.008.

[2] Grossmann S, Lohse D, Sun C. High-Reynolds number Taylor–Couette turbulence. Annu Rev Fluid Mech 2016;48(1):53–80. http://dx.doi.org/10.1146/annurev-fluid-122414-034353.

[3] Edlund EM, Ji H. Nonlinear stability of laboratory quasi–Keplerian flows. Phys Rev E 2014;89(2). 021004. http://dx.doi.org/10.1103/PhysRevE.89.021004.

[4] Lopez JM, Avila M. Boundary-layer turbulence in experiments on quasi–Keplerian flows. J Fluid Mech 2017;817:21–34. http://dx.doi.org/10.1017/jfm.2017.109.

[5] Shi L, Rampp M, Hof B, Avila M. A hybrid MPI-OpenMP parallel implementation for pseudospectral simulations with application to Taylor–Couette flow. Comput & Fluids 2015;106:1–11. http://dx.doi.org/10.1016/j.compfluid.2014.09.021.

[6] Guseva A, Willis AP, Hollerbach R, Avila M. Transition to magnetorotational turbulence in Taylor–Couette flow with imposed azimuthal magnetic field. New J Phys 2015;17(9). 093018. http://dx.doi.org/10.1088/1367-2630/17/9/093018.

[7] Lopez JM, Marques F, Avila M. The Boussinesq approximation in rapidly rotating flows. J Fluid Mech 2013;737:56–77. http://dx.doi.org/10.1017/jfm.2013.558.

[8] Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, Sahay V, Kambadur P, Barrett B, Lumsdaine A, Castain RH, Daniel DJ, Graham RL, Woodall TS. Open MPI: Goals, concept, and design of a next generation MPI implementation. In: Kranzlmüller D, Kacsuk P, Dongarra J, editors. Recent advances in parallel virtual machine and message passing interface. Lecture notes in computer science, vol. 3241, Berlin, Heidelberg: Springer; 2004, p. 97–104. http://dx.doi.org/10.1007/978-3-540-30218-6_19.

[9] Frigo M, Johnson S. The design and implementation of FFTW3. Proc IEEE 2005;93(2):216–31. http://dx.doi.org/10.1109/JPROC.2004.840301.

[10] Rampp M, Lopez JM, Shi L, Hof B, Avila M. Extreme scaling of nscouette, a pseudospectral DNS code. In: INSIDE: innovative supercomputing in Germany, Vol. 12. (2):2014, p. 48–50.

[11] Willis AP. The openpipeflow Navier–Stokes solver. SoftwareX 2017;6:124–7. http://dx.doi.org/10.1016/j.softx.2017.05.003.

[12] King GP, Lee W, Li Y, Swinney HL, Marcus PS. Wave speeds in wavy Taylor-vortex flow. J Fluid Mech 1984;141:365–90. http://dx.doi.org/10.1017/S0022112084000896.

[13] Ali M, Weidman PD. On the stability of circular Couette flow with radial heating. J Fluid Mech 1990;220:53–84. http://dx.doi.org/10.1017/S0022112090003184.

[14] Howey DA, Childs PRN, Holmes AS. Air-gap convection in rotating electrical machines. IEEE Trans Ind Electron 2012;59(3):1367–75. http://dx.doi.org/10.1109/TIE.2010.2100337.

[15] Lopez JM, Marques F, Avila M. Conductive and convective heat transfer in fluid flows between differentially heated and rotating cylinders. Int J Heat Mass Transfer 2015;90:959–67. http://dx.doi.org/10.1016/j.ijheatmasstransfer.2015.07.026.

[16] Gibson JF. Channelflow: A spectral Navier–Stokes simulator in C++. Tech. rep., University of New Hampshire; 2014, www.channelflow.org.

[17] Fardin MA, Perge C, Taberlet N. "The hydrogen atom of fluid dynamics" – introduction to the Taylor–Couette flow for soft matter scientists. Soft Matter 2014;10(20):3523. http://dx.doi.org/10.1039/c3sm52828f.

[18] Shi L, Hof B, Rampp M, Avila M. Hydrodynamic turbulence in quasi-Keplerian rotating flows. Phys Fluids 2017;29(4). 044107. http://dx.doi.org/10.1063/1.4981525.

[19] Leclercq C, Partridge JL, Augier P, Caulfield C-CP, Dalziel SB, Linden PF. Nonlinear waves in stratified Taylor–Couette flow. Part 1. Layer formation. 2016, arXiv:1609.02885.

[20] Leclercq C, Partridge JL, Caulfield C-CP, Dalziel SB, Linden PF. Nonlinear waves in stratified Taylor–Couette flow. Part 2. Buoyancy flux. 2016, arXiv:1609.02886.

[21] Lopez JM, Choueiri GH, Hof B. Dynamics of viscoelastic pipe flow at low reynolds numbers in the maximum drag reduction limit. J Fluid Mech 2019;874:699–719. http://dx.doi.org/10.1017/jfm.2019.486.

[22] Song B, Plana C, Lopez JM, Avila M. Phase-field simulation of core-annular pipe flow. Int J Multiph Flow 2019;117:14–24. http://dx.doi.org/10.1016/j.ijmultiphaseflow.2019.04.027.