A SIMULATION FRAMEWORK FOR TESTING FLOW CONTROL STRATEGIES

Marek Gayer*, Milan Milovanovic and Ole Morten Aamo
Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU)
O.S. Bragstads plass 2D, N-7491, Trondheim
Norway
e-mail: marek.gayer@itk.ntnu.no, milan.milovanovic@itk.ntnu.no, aamo@ntnu.no

ABSTRACT
We present a framework dedicated for simulation and control of incompressible fluid flows. The simulation is based on the Navier-Stokes model with possible control actuation from a customizable control module. We describe components of our simulation architecture: Computational Fluid Dynamics (CFD) code VISTA featuring a Navier-Stokes solver, Utility library with shared functionality for customizable Flow Control modules and the configuration system allowing to define separate simulation cases. Flow Control modules are responsible for performing control calculations, reading flow fields values, actuation, storing results for post-processing in each time step. By creating new Flow control modules, we can simulate different flow control behaviour and strategies. The simulation solution and it's components are realized in C++ using VISTA and Diffpack API. We present a case study based on this framework, where the objective is to suppress vortex shedding around cylinders in the 2D space domain by feedback control based module with kernel coefficients pre-calculated in MATLAB using Ginzburg-Landau model.

KEY WORDS
Flow control simulation, Navier-Stokes, Vortex shedding.

1 Introduction

1.1 Flow control

Flow control involves controlling a flow field using passive or active devices in order to bring on desired changes in the behaviour of the flow. For instance, laminar flow, which is characterized by parallel layers of fluid moving in a very regular and deterministic way, is associated with considerable less drag, or friction, at wall-fluid interfaces, than its counterpart, turbulent flow, which is characterized by small scale velocity components that appear to be stochastic in nature. Usually, laminar flows are unstable, and will unless controlled evolve into turbulent flows. Common control objectives for flows include: Delaying or advancing transition from laminar to turbulent flow, suppressing or enhancing turbulence, preventing or provoking separation and preventing or provoking certain flow regimes for multicomponent flows.

This paper presents a Computational Fluid Dynamics (CFD) simulation framework dedicated for simulation of flow control based on Navier-Stokes model and using customizable Flow control modules, allowing to implement general flow controllers.

1.2 Vortex shedding simulation and control

Dynamics of flow around cylinder has been studied for many years. Modelling and simulation of the cylinder wake behaviour are proposed in various experiments and models in [1], [2], [3], [4], [5] and [6]. For flows past submerged obstacles with sufficiently large Reynolds numbers, vortex shedding forms. A popular model flow for studying vortex suppression by means of open-loop or feedback control, is the flow past a two-dimensional circular cylinder, as sketched in Fig. 1 where the so-called von Kármán vortex street is visualized using passive tracer particles. Various simple feedback control configurations, such as [7] and [8] were proposed for successful suppression of vortex shedding in numerical simulations. A simple model, based on the Ginzburg-Landau equation with appropriate coefficients, has been found to well model the dynamics of vortex shedding in [9], [10] and [11]. Systematic control designs for finite dimensional approximations of the model from [11] were developed in [12], [13] for the linearized model, and in [14] for the nonlinear model. In [15] and [16], boundary control laws based on the non-discretized, linearized Ginzburg-Landau equation were rigorously derived for the state feedback and boundary output feedback cases, respectively. They were shown to effectively attenuate vortex shedding for the nominal case with Ginzburg-Landau as the system plant as well as the design model. In the test case study presented in this paper, the state feedback proposed in [15] is applied to the Navier-Stokes solver.

2 Simulation framework overview

The major parts of the framework are a CFD software code named VISTA with a Navier-Stokes solver, Utility library

*This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

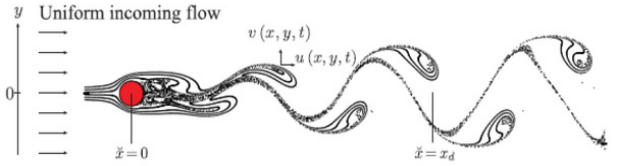

Figure 1. Vortex shedding from a cylinder visualized by passive tracer particles.

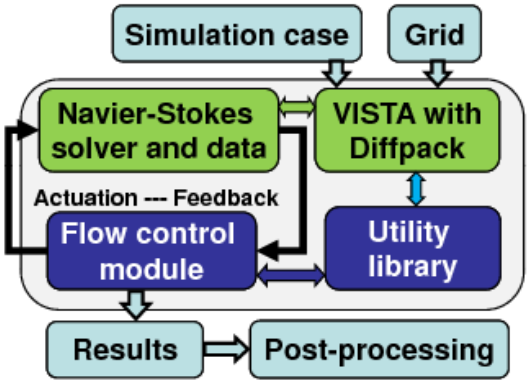

Figure 2. Our simulation framework architecture overview.

containing general functionality common for flow simulation and control modules and a customizable Flow control module. The architecture is schematically depicted on Figure 2.

2.1 CFD code for solving Navier-Stokes model

Our simulation framework is based on the Navier-Stokes model. As the basis for our CFD simulations of the flow control we use VISTA, developed by SINTEF Applied Mathematics[1]. VISTA is a general purpose, extendible CFD code using the finite element method and solving wide range of problems. The core of this code is a finite element method based solver of incompressible Navier-Stokes equation:

$$\rho\frac{\partial u}{\partial t} + \rho(u \cdot \nabla u) + \nabla p - \mu\nabla u = f \qquad (1)$$
$$\nabla \cdot u = 0$$

where ρ is the density of the fluid, u is the velocity vector with velocities in x and y direction, p is the pressure, μ is the dynamic viscosity and f an external body force. The boundary conditions for the equation can be set separately for each simulation cases.

The VISTA is written in C++ and uses Diffpack numerical package [17] for simulation based on the finite element method. Beside the Navier-Stokes solver, it has also functionalities for reading grids, obtaining simulated values from the domain and setting Dirichlet boundary conditions in subsequent time steps, reading configuration of simulation cases (VISTA menu system) and store some results in files. We will describe these possibilities later.

[1] http://www.sima.sintef.no/

2.2 Utility library

The Utility library efforts to provide general functionality shared by customizable Flow control modules. This code is realized using programming VISTA API and Diffpack [18]. The vector and matrix variables for data-structures are provided as Diffpack real and complex array classes, allowing us to use some basic mathematical operations over these arrays (e.g. adding, multiplication), bound checking as well as possibility for passing these data types as arguments of C++ methods and functions. Utility library defines additional routines over basic and limited work with these structures provided by Diffpack. Utility library contains routines for reading input files and writing output data sets. Vector and matrices files are represented as saved MATLAB script files with a special syntax format, containing assignments for vector items. Such files with this syntax can be both read from and written to Diffpack data types using Diffpack API and we have a script to generate these files in MATLAB.

2.3 Flow control modules

A Flow control module allows realization of a simulation flow controller. At the beginning of a simulation, a module is responsible for the initialization and reading of eventual input values related to a control case. It is responsible for processing an algorithm for a concrete controller computation in each time step. A module is responsible for applying the computed values back to the Navier-Stokes model. A module during each time step also updates various output files with read and computed values, which can be used for monitoring and post-processing. For realization some of its functionalities, it uses the Utility library. A sample of such module will be described in detail further. All Flow control modules are implemented as C++ inherited classes. Some inherited methods are invoked from VISTA, such as on initialization, each time step and on finishing the simulation. By creating new Flow control modules, we can simulate different flow control behaviour and strategies. We plan to enrich the functionality of the Utility library during development of such modules.

2.4 Input parameters in the VISTA menu system

Various input parameters for each of the independent simulation cases are specified in a structured text file format, which is parsed and accessed by VISTA and is called VISTA menu system. This allows Vista CFD configuration system to set various variables defining the simulation case. The menu system is also accessed by our Utility

library and Flow control modules, since custom variables can be defined and read from the menu system. This way, we can for instance define the Reynolds number used during simulation, starting time and length of simulation, time steps, locations of files containing mesh and control coefficients, controller fade in period, etc. Further, parameters controlling creation of VTF data sets[2] and allowing eventual restarting of the simulation from an already simulated state (or using a saved state as the basis for a simulation with modified parameters) can be defined as well. There are also additional plain text files referred and used in the VISTA menu system. In these files, we can define boundary conditions and points on which characteristics will be extracted.

2.5 Computational grids

General purpose grid generators can be used to produce computational grids for VISTA, since the VISTA code supports meshes in Diffpack and VTF formats. During development of our simulation framework, including the later described test case study, we have used the grid generator GRIDDLER[3] to generate finite element meshes in the VTF format. GRIDDLER supports general block-structured meshes in 2D and 3D. The vertices, curves, surfaces and blocks are specified in an input file. The file name of the grid is specified in the VISTA menu system and is automatically read by VISTA.

2.6 Results storage and post-processing

The characteristics from the computed flow field in whole time and space domain are stored in VTF format using GLview Express Writer API[4], embedded in VISTA and also some files are produced by our Utility library in plain text files, which can be imported in MATLAB. We post-process these results by using both the MATLAB and GLview Inova[5].

In GLview Inova, we can easily render animations from VTF files or review simulation progress of flow control, by visualizing various vector and scalar properties such as pressure, vorticity and velocities. During and after finishing of the simulation, we can monitor in the MATLAB various parameters: applied control values, velocities, pressures, force drag and lift and many others parameters for 2D and 3D plots. This functionality is maintained by availability of our MATLAB scripts processing the stored data sets.

Each of simulation cases is using different directories for storing it's results, thus keeping order in the file system and offering possibilities of easy migrations and backups. Various characteristics are stored in separate files to allow quick loading of subsets of output parameters in simulated case, both in post-processing as well as during simulation monitoring.

[2] http://www.ceetron.com/en/vtf_format.aspx

[3] Developed by Karstein Sørli at SINTEF ICT Applied Mathematics, http://sourceforge.net/projects/griddler/

[4] http://www.ceetron.com/en/glview_express1.aspx

[5] http://www.ceetron.com/en/glview_inova1.aspx

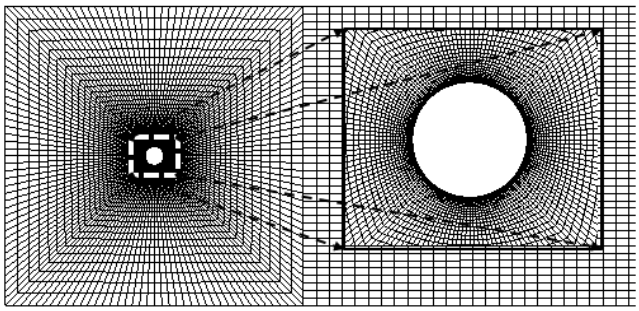

Figure 3. Computational grid with zoom around cylinder.

3 Case study of a simulation and control of vortex shedding around cylinders

This case study is realized within our simulation framework. It provides simulation of vortex shedding around cylinders in 2D. It uses custom boundary conditions as well as the mesh for the VISTA Navier-Stokes solver. Fig. 3 shows the grid generated by GRIDDLER, which we use for testing. A Flow control module has been implemented based on the control law developed in [15].

3.1 Our boundary conditions for Navier-Stokes solver

We assume free stream flow behaviour, flow at the boundaries is not affected by the cylinder. The following boundary conditions for Eq. 1 can be then assumed:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_1 \quad (2)$$

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_2 \cup \Gamma_4 \quad (3)$$

$$\boldsymbol{\tau} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_2 \cup \Gamma_4 \quad (4)$$

$$\boldsymbol{u} = \boldsymbol{u}_{in} \qquad \text{on } \Gamma_3 \quad (5)$$

$$\boldsymbol{u} = \boldsymbol{u}_c(t) \qquad \text{on } \Gamma_5 \quad (6)$$

where the boundaries are denoted by Γ_i, and are depicted in figure 4. Furthermore $\boldsymbol{u}_{in}$ is the inlet velocity vector at Γ_3 and $\boldsymbol{u}_c(t)$ is the velocity vector on the cylinder boundary Γ_5 due to control actuation. Besides, $\boldsymbol{n}$ and $\boldsymbol{\tau}$ are the unit outer normal and tangent vectors, while $\boldsymbol{\sigma}$ is the stress tensor given by:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \mu\nabla\boldsymbol{u} \quad (7)$$

Since the VISTA uses a continuous projection method for solving the Navier–Stokes equations, where the computation of the velocity and pressure is split, the boundary conditions for the numerical scheme looks slightly different. The boundary condition on the outlet is:

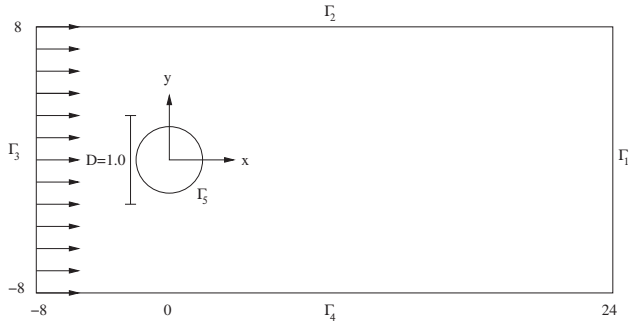

Figure 4. Computational domain with boundaries.

$$p = 0, \quad \nabla u \cdot n = 0 \qquad \text{on } \Gamma_1 \tag{8}$$

and a zero gradient condition is used for the pressure on the remaining part of the boundary

$$\nabla p \cdot n = 0 \qquad \text{on } \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \cup \Gamma_5 \tag{9}$$

The inflow velocity $\boldsymbol{u}_{in}$ is chosen by the user before the simulation is started, while the velocity $\boldsymbol{u}_c(t)$ is computed by the controller as the simulation proceeds.

3.2 State feedback model using Ginzburg-Landau based kernels

In [11] a simplified model of the computation of cylinder wake dynamics was suggested in terms of the Ginzburg-Landau equation.

$$\frac{\partial A}{\partial t} = a_1 \frac{\partial^2 A}{\partial \breve{x}^2} + a_2(\breve{x}) \frac{\partial A}{\partial \breve{x}} + a_3(\breve{x}) A + a_4 |A|^2 A + \delta(\breve{x} - 1) u, \tag{10}$$

where A is a complex-valued function of one spatial variable, $\breve{x} \in \mathbb{R}$, and time, $t \in \mathbb{R}_+$. The boundary conditions are $A(\pm\infty, t) = 0$. The control input, denoted u, is in the form of point actuation at the location of the cylinder, and the coefficients a_i, $i = 1, .., 4$, were fitted to data from laboratory experiments in [11]. δ denotes the Dirac distribution.

In [15, 16] a simplified form of Ginzburg-Landau equation was considered. The resulting system is given by

$$\frac{\partial A}{\partial t} = a_1 \frac{\partial^2 A}{\partial \breve{x}^2} + a_2(\breve{x}) \frac{\partial A}{\partial \breve{x}} + a_3(\breve{x}) A \tag{11}$$

for $\breve{x} \in (0, x_d)$, with boundary conditions

$$A(0, t) = u(t) \text{ and } A(x_d, t) = 0, \tag{12}$$

where $A : [0, x_d] \times \mathbb{R}_+ \to \mathbb{C}$, $a_2 \in C^2([0, x_d]; \mathbb{C})$, $a_3 \in C^1([0, x_d]; \mathbb{C})$, $a_1 \in \mathbb{C}$, $t \in \mathbb{R}_+$is time, and $u : \mathbb{R}_+ \to \mathbb{C}$ is the control input. a_1 is assumed to have strictly positive real part. The boundary conditions are $A(\pm\infty, t) = 0$. The control input, denoted u, is in the form of point actuation at the location of the cylinder, and the coefficients a_i, $i = 1, .., 3$, were fitted to data from laboratory experiments in [11].

In [15], stabilizing state feedback boundary control laws for system (11)–(12) were derived based on the backstepping methodology [19, 20, 21, 22]. The state feedback controller designed in [15] takes the form

$$u(t) = \int_0^{x_d} \frac{1}{x_d} [k_1(\frac{x_d - \breve{x}}{x_d}) - i k_{c,1}(\frac{x_d - \breve{x}}{x_d})] A(\breve{x}, t) \times \exp\left(\frac{1}{2a_1} \int_0^{\breve{x}} a_2(\tau) d\tau\right) d\breve{x} \tag{13}$$

where i denotes the imaginary unit, and k_1 and $k_{c,1}$ are the controller kernels which can be computed as the solution of a certain hyperbolic partial differential equation. Further details can be found in [15]. The controller kernels are precalculated in our MATLAB codes.

3.3 Connecting the state feedback controller with Navier-Stokes solver

$A(\breve{x}, t)$ in Eq. 13 may represent any physical variable (velocities (u, v) or pressure p), or derivations thereof, along the centerline of the 2D cylinder flow. The choice will have an impact on the performance of the Ginzburg-Landau model. In our case, we associate A with the transverse fluctuating velocity $v(\breve{x}, \breve{y} = 0, t)$ which seems to be a particularly good choice [23]. As pointed out in [13], the model is derived for Reynolds numbers close to the critical Reynolds number for onset of vortex shedding, but has been shown to remain accurate far outside this vicinity for a wide variety of flows. For the interior flow, we define two curves originating at the actuation slots and converging at the flow centerline as they stretch into the 2D flow domain (see left of Fig. 5). Transverse velocity is measured along these curves, denoted $v_l(\breve{x}, t)$ and $v_u(\breve{x}, t)$ in left of Fig. 5, and related to A as

$$Re(A) := v(\breve{x}, t) = \frac{1}{2}(v_l(\breve{x}, t) + v_u \breve{x}, t)). \tag{14}$$

The imaginary part of A is reconstructed based on modes of the Ginzburg-Landau equation having the form

$$A(\breve{x}, t) = \Phi(\breve{x}) exp(i\kappa\breve{x} - i\omega t). \tag{15}$$

We approximate $Im(A)$ by shifting $Re(A)$ in space corresponding roughly to $\Delta\breve{x} = TU/4$, where T is the shedding period and U is the streamwise velocity. Having obtained A as described, the control input is computed using Eq. 13. With the real part of A representing transverse velocity, the control actuation which is a Dirichlet boundary condition at the actuation end in the Ginzburg-Landau model, can be realized in the CFD code by jets on the cylinder surface. We

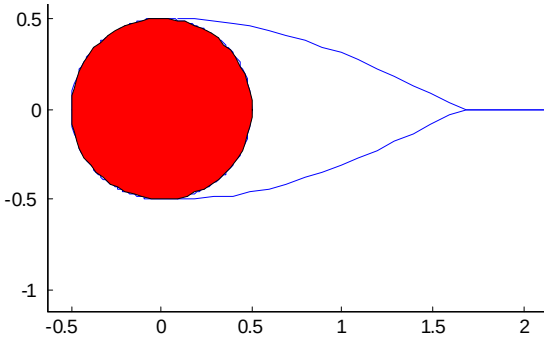

Figure 5. Curves along which transverse velocity is measured.

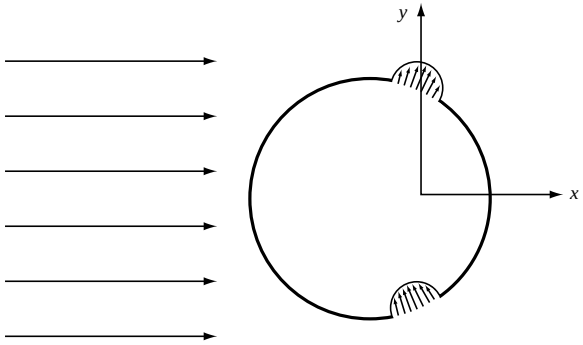

Figure 6. Suction/blowing actuation slots.

do this by employing two slots configured as shown in right of Fig. 6, close to the separation layers for effectiveness [24]. In our case, the actuation is done by using blowing and suction slots at $\pm 110^{o}$ from the stagnation point with a span of 15^{o}. The velocity profile of the jet is set according to:

$$\boldsymbol{u}_c(\alpha, t) \cdot \boldsymbol{n} = u(t) sin(\frac{\pi}{\alpha_{span}}(\alpha_{N_i} - \alpha_{start})). \quad (16)$$

We recalculate values of $u(t)$ and apply them each time step to the cylinder slots. This is effectively providing a suitable control parameter for the jet. Besides, α_{N_i} is the angle of the actual node in the mesh, while $\alpha_{start,end}$ are the start and end angles of the jet. For the purpose of solver stability, the actuation in all simulations is faded-in linearly in approximately one natural vortex shedding period ($@Re = 60, T_{fadein} = 6[s]$).

When one slot applies suction the other applies an equal amount of blowing, thereby maintaining the mass balance in the system. The actuation is thus suction and blowing whose size is given by the real part of u computed from Eq. 13. In practice, the physical actuation could be for instance micro/synthetic jet actuators, distributed on the cylinder surface, and a possible choice for the physical sensing could be pressure sensors distributed on the cylinder surface.

3.4 The Flow control module for this case

The Flow control module at the initialization reads the control kernels. During reading the controller kernels, we use Utility library to interpolate, resize and scale dimensions of smaller vector arrays into longer vector arrays. Each time step, the module is responsible for processing transversal velocities from sensing devices represented in the Navier-Stokes solver, for recalculating state feedback controller by evaluating Eq. 13. In each time step of the code, we retrieve beside transversal velocities also additional characteristics for the controller simulation and monitoring on both surface of the cylinder as well as on the horizontal line leading from the center of the cylinder, see Fig. 5. These include longitudal velocity, vorticity, shear stress, nodal coordinates and pressure.

For applying the computed control values back to the cylinder, we set time dependent Dirichlet boundary conditions along the surface points of the cylinder. The module computes actuation values and sets boundary conditions during each time step. They are passed to the Navier-Stokes solver, in form of boundary velocities on the surface on the cylinder, in place of suction and blowing jets and are distributed according to Eq. 16. The values are applied to set of singular points on the cylinder. The Flow control module reads parameters of controller jets (such as radius, center, span) using the VISTA menu system.

3.5 Simulation testing and results

We have tested our simulation solution with a case with fully developed vortex shedding at $Re = 60$. We can visualize suppression of developing of the vortex shedding in the cylinder of both scalar and vector fields (velocities, pressures, etc). Vortex shedding subjects the bluff body, in this case the cylinder, to periodic forcing, and would in any physical system induce potentially destructive vibrations. The controller applied in subsequent time steps effectively attenuates the forcing. Fig. 7 shows a vorticity map of the flow at $t = 0$. The von Kármán vortex street is clearly visible. Vortex shedding subjects the bluff body, in this case the cylinder, to periodic forcing, and would in any physical system induce potentially destructive vibrations. The controller applied in subsequent timesteps effectively attenuates the forcing. Fig. 8 and Fig. 9 show vorticity maps at $t = 20s$ and $t = 100s$ (when vortex shedding is completely removed). Fig. 10, 11 and 12 visualize transversal velocities and also depict progress of gradual suppression of vortex shedding in time after applying control. A sample of 3D plots, monitoring pressures on the cylinder boundary, is on 15 and 16. A sample of 2D monitoring, which is very useful during ongoing simulation (e.g. figure is updated each 10 seconds) is depicted on Fig. 14. During and after finishing simulations, we are able to monitor and perform 3D visualizations of changing velocities, distance and time, as it is on Fig. 13.

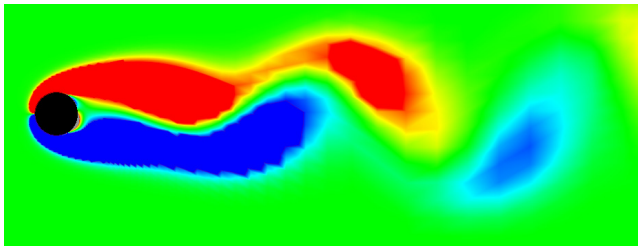

Figure 7. Vorticity map at $t = 0$. Fully developed vortex shedding.

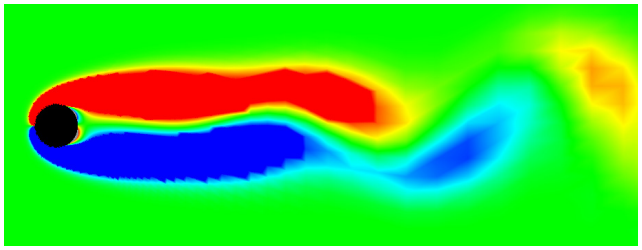

Figure 8. Vorticity map at $t = 20s$. Attenuated vortex shedding after applying actuation.

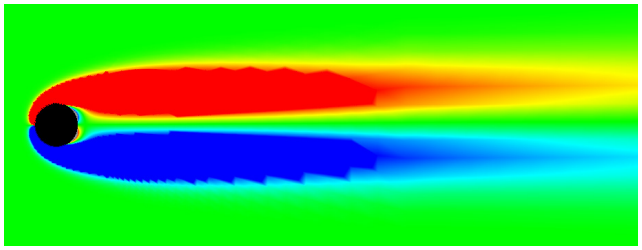

Figure 9. Vorticity map at $t = 100$. Completely stabilized flow.

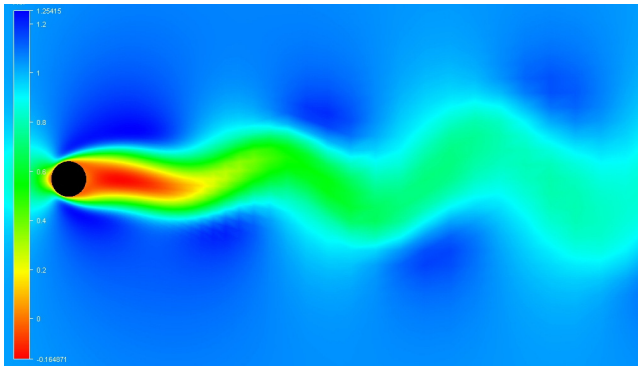

Figure 10. Transversal velocity map at $t = 0$.

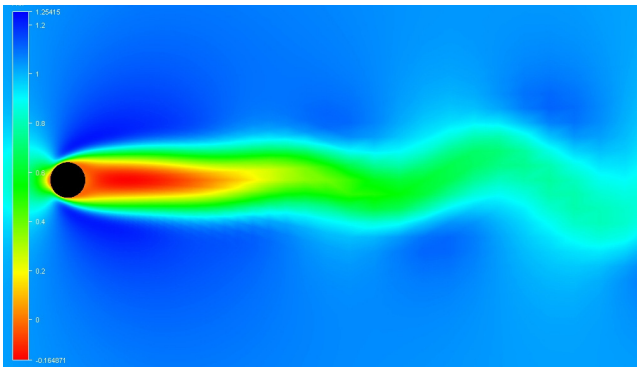

Figure 11. Transversal velocity map at $t = 20$.

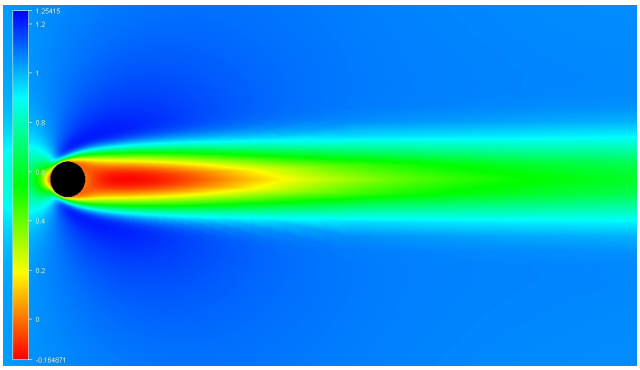

Figure 12. Transversal velocity map at $t = 100$.

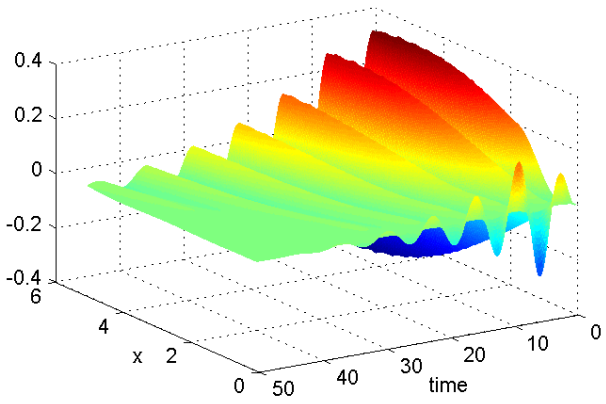

Figure 13. Suppression of velocities in distances up to 6 m from the cylinder in time from 0 to 50 seconds.

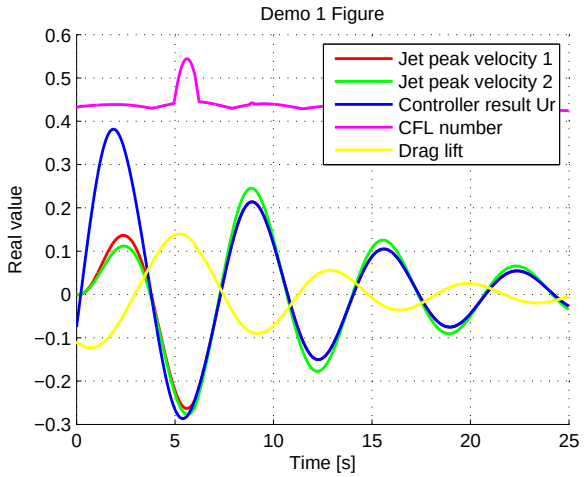

Figure 14. Monitoring of simulation control related parameters.

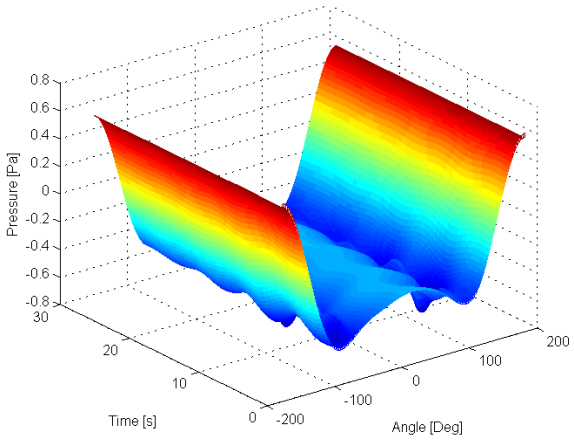

Figure 15. 3D plot of pressures around the cylinder boundary (±180º).

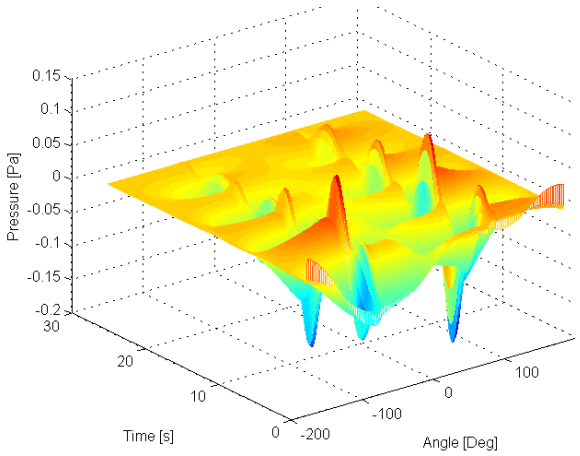

Figure 16. 3D plot of pressures around the cylinder boundary (±180º) with substraction of static pressure values.

4 Implementation and portability notes

To run the simulations, our primary focus are installations on remote multiprocessor Linux servers, so that we can eventually compute general, usually complex 3D simulations flow control cases and in future with parallel processing. We use SSH tunnelling to allow mapping of remote disk storage's as local drives.

While this option offers advantages over local installations, namely in terms of performance, we also found this option on the other hand often impractical namely due to lack of development tools on the server and certain problems with queuing system and performance of transferring simulated data sets over the network. Therefore, during development of the framework and for our 2D case study described in the paper, we had preferred a local installation.

VISTA CFD code currently runs only on Linux, and is dependent on the Diffpack library. To overcome the portability limitation, as well as difficult installation of both the VISTA and Diffpack, we decided to use VmWare virtual machines with Debian Etch Linux as the guest the operating system for the base for our local installation. In our case, this allows us to make experiments and run simulations in one operating system, since we use MATLAB, GLview Inova and other applications in Windows. For sharing data between simulation environment and host operating system in VmWare and other applications, we use network drives on Windows servers and using Samba to access them in Linux.

5 Conclusion

In this paper, we have presented a simulation framework for general incompressible fluid flows control. Our proposal features the CFD code VISTA computing 2D flow around the cylinder using Navier-Stokes solver and a customizable Flow control module. In each time step a Flow control module is responsible for reading and writing state values from Navier-Stokes solver, recalculating controller code actuation values, and applying them back to the flow field. With different Flow control modules, we can simulate various flow control problems and strategies. The Utility library offers general functionality for these modules and helps to generate results data for 2D and 3D postprocessing. Results are monitored using MATLAB scripts, while VISTA provides ability to automatically store VTF files for animation rendering.

We have presented a case study based on our simulation environment allowing control of vortex shedding in flows around cylinders with a state feedback controller, removing vortex shedding for the Reynolds number $Re = 60$. The simulations can be potentially repeated for higher Reynolds numbers, under the condition of providing recomputed kernels for state feedback controller.

For the future work, we plan to extend our simulation design and codes to work with the output feedback problem, which was solved for the Ginzburg-Landau equation

in [16], develop simulation cases and Flow control modules for 3D control of fluid flows, and implement parallel processing.

6 Acknowledgment

This work was supported by the BVV program at NTNU (Program on Computational Science and Visualization).

References

[1] C. P. Jackson, "A finite-element study of the onset of vortex shedding in flow past variously shaped bodies," *Journal of Fluid Mechanics*, vol. 182, pp. 23–45, 1987.

[2] P. Strykowski and K. Sreenivasan, "On the formation and suppresion of vortex 'shedding' at low Reynolds numbers," *Journal of Fluid Mechanics*, vol. 218, p. 71, 1990.

[3] C.-Y. Wen and C.-Y. Lin, "Two-dimensional vortex shedding of a circular cylinder," *Physics of Fluids*, vol. 13, pp. 557–560, 2001.

[4] M. König, B. R. Noack, and H. Eckelmann, "Discrete shedding modes in the von Karman vortex street," *Physics of Fluids*, vol. 5, pp. 1846–1848, 1993.

[5] O. Inoue, T. Yamazaki, and T. Bisaka, "Numerical simulation of forced wakes around a cylinder," *International Journal of Heat and Fluid Flow*, vol. 16, pp. 327–332(6), 1995.

[6] M. Facchinetti, E. Langre, and F. Biolley, "Vortex shedding modeling using diffusive van der Pol oscillators," *Comptes Rendus Mecanique*, vol. 330, pp. 451–456, 2002.

[7] D. Park, D. Ladd, and E. Hendricks, "Feedback control of von Kármán vortex shedding behind a circular cylinder at low Reynolds numbers," *Physics of fluids*, vol. 7, no. 7, pp. 2390–2405, 1994.

[8] M. Gunzburger and H. Lee, "Feedback control of Karman vortex shedding," *Transactions of the ASME*, vol. 63, pp. 828–835, 1996.

[9] P. Huerre and P. Monkewitz, "Local and global instabilities in spatially developing flows," *Annual Review of Fluid Mechanics*, vol. 22, pp. 473–537, 1990.

[10] P. Albarede and P. A. Monkewitz, "A model for the formation of oblique shedding and 'chevron' patterns in cylinder wakes," *Physics of Fluids*, vol. 4, pp. 744–756, 1992.

[11] K. Roussopoulos and P. Monkewitz, "Nonlinear modelling of vortex shedding control in cylinder wakes," *Physica D: Nonlinear Phenomena*, vol. 97, pp. 264–273, 1996.

[12] E. Lauga and T. Bewley, "H infinity control of linear global instability in models of non-parallel wakes," in *Proceedings of the Second International Symposium on Turbulence and Shear Flow Phenomena*, Stockholm, Sweden, 2001.

[13] E. Lauga and T. Bewley, "Performance of a linear robust control strategy on a nonlinear model of spatially-developing flows," *Journal of Fluid Mechanis*, vol. 512, pp. 343–374, 2004.

[14] O. M. Aamo and M. Krstic, "Global stabilization of a nonlinear Ginzburg-Landau model of vortex shedding," *European Journal of Control*, vol. 10, no. 2, 2004.

[15] O. M. Aamo, A. Smyshlyaev, and M. Krstic, "Boundary control of the linearized Ginzburg-Landau model of vortex shedding," *SIAM Journal on Control and Optimization*, vol. 43, no. 6, pp. 1953–1971, 2005.

[16] O. M. Aamo, A. Smyshlyaev, M. Krstic, and B. Foss, "Output feedback boundary control of a Ginzburg-Landau model of vortex shedding," *IEEE Transactions on Automatic Control*, vol. 52, no. 4, pp. 742–748, 2007.

[17] A. Bruaset and H. Langtangen, "A comprehensive set of tools for solving partial differential equations; diffpack," in *Numerical Methods and Software Tools in Industrial Mathematics*, 1996.

[18] Langtangen and H. P, *Computational Partial Differential Equations - Numerical Methods and Diffpack Programming*, vol. 1 of *Texts in Computational Science and Engineering*. Springer -Verlag, 2 ed., 2003.

[19] A. Smyshlyaev and M. Krstic, "Closed form boundary state feedbacks for a class of 1D partial integro-differential equations," *IEEE Transactions on Automatic Control*, vol. 49, pp. 2185–2202, 2004.

[20] W. J. Liu, "Boundary feedback stabilizaton of an unstable heat equation," *SIAM Journal on Control and Optimization*, vol. 42, pp. 1033–1043, 2003.

[21] M. Krstic and A. Smyshlyaev, *Boundary Control of PDEs: A Course on Backstepping Designs*. SIAM, 2008.

[22] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*. Wiley, 1995.

[23] P. Monkewitz, *Private communication*.

[24] P. Catalano, M. Wang, G. Iaccarino, I. F. Sbalzarini, and P. Koumoutsakos, "Optimization of cylinder flow control via zero net mass flux actuators," in *Proceedings of the CTR summer program*, Center for Turbulence Research, Stanford University, 2002.