Computational Tool for a Mini-Windmill Study with SOFT

M. Garbey, M. Smaoui, N. De Brye, and C. Picard

Department of Computer Science, University of Houston, Houston, TX 77204 USA
garbey@cs.uh.edu

1 Introduction and Motivation

In this paper, we present a parallel computational framework for the completely automated design of a Vertical Axis Fluid Turbine (VAFT) . Simulation, Optimum design, Fabrication and Testing (SOFT) of the VAFT is integrated into a hardware/software environment that can fit into a small office space.

The components of the four steps design loop are as follows

1. **Simulation**: We use a parallel CFD algorithm to run a direct simulation of the fluid structure interaction problem. We derive from that computation the torque and the average rotation speed for a given friction coefficient on the rotor shaft and an average flow speed. Our objective is to get the most power out of the windmill, consequently the highest rotation speed possible.
2. **Optimization**: We optimize the shape of the blade section with a genetic algorithm and/or a surface response. The evaluation of the objective function (average rotation speed) corresponds to the direct simulation of the Navier Stokes flow interacting with the rotating turbine, until reaching a stationary regime. Because this simulation is compute-intensive, we distribute the evaluation of the objective function for the different shapes (gene or parameter combinations) on a network of computers using an embarrassingly parallel algorithm.
3. **Fabrication**: The optimization procedure results in a supposedly optimum shape in the chosen design space. This shape is sent to a 3-D printer that fabricates the real turbine. This turbine is set up such that it can be easily mounted on a standard base equipped with an electric alternator/generator.
4. **Testing**: The windmill is tested in a mini wind tunnel. The electric output is measured and a video camera can directly monitor the windmill rotation through the transparent wall of the wind tunnel. This information can be analyzed by the computer system and comparison with the simulation is assessed. Figure 1 gives a graphical overview of the SOFT concept.

This four-steps loop can be repeated as many times as needed. Eventually, artificial intelligence tools such as Bayesian networks can be added to close the design

loop efficiently. This component would decide when to test other classes of design characterized by the number of blades, the number of stages in the turbine, the use of baffles to channel the flow etc.; see Fig. 2

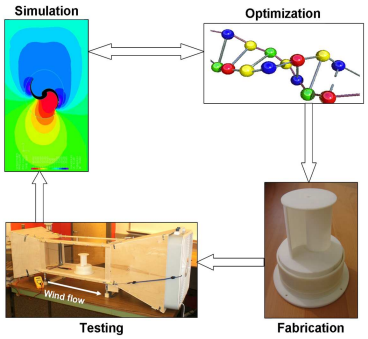

Fig. 1. SOFT concept

Fig. 2. Collection of VAFT Shapes

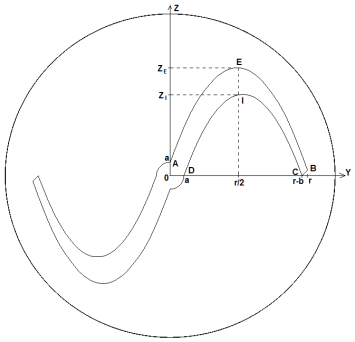

Fig. 3. Design of the turbine

We will concentrate here on two dimensional computation with a simplified two scoop blade that is symmetric with respect to its shaft as in Fig. 3.

We have chosen to optimize the VAFT in low speed flow condition, with Reynolds number in the range (100–2000). We do not need a priori to deal with complex turbulent flow neither stability issues in the fluid structure interaction. One of the possible applications is to power remote sensors with VAFT when other energy sources are more difficult to manage. We are also interested in low Reynolds number flows that are characteristic of micro air vehicle [8].

This project has some obvious pedagogic components that can motivate undergraduate students to do science! However, in reality, a critical step in the process is obviously the CFD method to test the VAFT performance: the numerical simulator should be robust, extremely fast but accurate enough to discriminate bad design from good design. We will discuss an immersed boundary method and domain decomposition solver that we have tentatively developed to satisfy this ambitious program. We first describe the incompressible Navier Stokes Solver.

2 Flow Solver

We use the penalty method introduced by Caltagirone and his co-workers [3] that is simpler to implement than our previous boundary fitted methods [4] and applies naturally to flow in a domain with moving walls [7].

The flow of incompressible fluid in a rectangular domain $\Omega = (0, L_x) \times (0, L_y)$ with prescribed values of the velocity on $\partial\Omega$ obeys the NS equations:

$$\begin{aligned} \partial_t U + (U \cdot \nabla)U + \nabla p - \nu \nabla \cdot (\nabla U) = f \quad & \text{in } \Omega \\ div(U) = 0 \quad & \text{in } \Omega \\ U = \mathbf{g} \quad & \text{on } \partial\Omega. \end{aligned}$$

We denote by $U(x,y,t)$ the velocity with components (u_1, u_2) and by $p(x,y,t)$ the normalized pressure of the fluid. ν is a kinematic viscosity.

With an immersed boundary approach the domain Ω is decomposed into a fluid subdomain Ω_f and a moving rigid body subdomain corresponding to the blade Ω_b. In the L_2 penalty method the right hand side f is a forcing term that contains a mask function Λ_{Ω_b}

$$\begin{aligned} \Lambda_{\Omega_b}(x,y) = 1 \quad & \text{if } (x,y) \in \Omega_b, \\ 0 \quad & \text{elsewhere,} \end{aligned}$$

and is defined as

$$f = -\frac{1}{\eta} \Lambda_{\Omega_b} \{U - U_b(t)\}. \tag{1}$$

U_b is the velocity of the moving blade and η is a small positive parameter that tends to 0.

A formal asymptotic analysis helps us to understand how the penalty method matches the no slip boundary condition on the interface $S_b^f = \bar{\Omega}_f \bigcap \bar{\Omega}_b$ as $\eta \to 0$. Let us define the following expansion:

$$U = U_0 + \eta\, U_1, \quad p = p_0 + \eta\, p_1.$$

Formally we obtained at leading order,

$$\frac{1}{\eta} \Lambda_{\Omega_b} \{U_0 - U_b(t)\} = 0,$$

that is

$$U_0 = U_b \quad \text{for } (x,y) \in \Omega_b.$$

The leading order terms U_0 and p_0 in the fluid domain Ω_f satisfy the standard set of NS equations:

$$\begin{aligned} \partial_t U_0 + (U_0 \cdot \nabla)U_0 + \nabla p_0 - \nu \nabla \cdot (\nabla U_0) = 0 \quad & \text{in } \Omega_f \\ div(U_0) = 0 \quad & \text{in } \Omega. \end{aligned}$$

At next order we have in Ω_b,

$$\nabla p_0 + U_1 + Q_b = 0, \tag{2}$$

where

$$Q_b = \partial_t U_b + (U_b \cdot \nabla) U_b - \nu \nabla \cdot (\nabla U_b).$$

Further the wall motion U_b must be divergence free which is the case for a rigid body. In conclusion, the flow evolution is dominated by the NS equations in the flow domain, and by the Darcy law with very small permeability inside the rotor.

In this framework, the efficiency of the NS code relies essentially on the design of robust and efficient parallel solvers for linear operators of the following two types

$$-\varepsilon \Delta + \delta u \cdot \nabla + Id \quad \text{and} \quad -\Delta.$$

To be more specific, time stepping uses a multi-step projection scheme. Space discretization is done with a staggered grid. The convection is processed with the method of characteristic. Since the penalty term is linear it is trivial to make that term implicit in time stepping. Finally we use a combination of Aitken-Schwarz as the domain decomposition solver with block LU decomposition per subdomain. We refer to [5] for an extensive report on the performance of that parallel solver on multiple computer architecture and a comparison with other solvers such as multigrid or Krylov methods. We are going now to describe a key aspect that is the Fluid Structure Interaction (FSI) approach we have followed here.

3 FSI

First let us discuss the computation of the torque. The main difficulty with the penalty method is that the flow field is not differentiable at the fluid structure interface. The computation of the drag forces exerted on the blade cannot be done directly with the standard formula

$$F = \int_{\partial \Omega_b} \sigma(U,p) n \, d\gamma,$$

where $\sigma(U,p) = \frac{1}{2}\nu(\nabla U + (\nabla U)^t) - p\, I$.

Using the observation of [2], we can compute this force with an integral on the gradient of pressure *inside* the blade:

$$F = \lim_{\eta \to 0} \int_{\Omega_b} \nabla p \, dx,$$

which ends up with the simple formula using the momentum equation:

$$F = \lim_{\eta \to 0} \frac{-1}{\eta} \int_{\Omega_b} U - U_b \, dx. \tag{3}$$

The computation of the torque is done by summing up the contribution of (3) to the torque, cell-wise and inside the blade. We take into account only the interior cells

to avoid the singularity at the wall and leave out from the calculation all the cells that intersect the boundary of the blade $\partial\Omega_b$. The verification on the computational efficiency of this technic has been done with static torque calculation. We checked that when the penalty parameter goes to zero, $\eta \to 0$, the numerical error is rapidly dominated by the grid accuracy. As $h \to 0$, we observe first order convergence. Finally we did compare our torque computation with Adina's computation. Adina is a commercial finite element code that uses an arbitrary Lagrangian-Eulerian formulation with displacements compatibility and traction equilibrium at the blade interface.

We did some fine mesh calculation with Adina of the static toque, i.e for fixed orientation of the blade, and use that numerical solution as reference. We found that it was easy to maintain a 10% accuracy compare to the reference solution computed with Adina with moderated grid size and Reynolds number of order a few hundreds, provided that the tip of the blade had a thickness of at least 3 to 4 mesh points.

Let us discuss now the FSI algorithm based on this torque calculation. It is classic to apply the second Newton law and advance the rotor accordingly: we alternate then the flow solver and solid rotation. Unfortunately, while the penalty method is very robust, this solution is ill-conditioned, due to the stiffness of the coupling and sensitivity to the noisy calculation of the Torque. As a matter of fact, we can expect small high frequency oscillation in time of the torque calculation with rotating blades as Cartesian cell enter/leaves the domain of computation Ω_b.

"Thinking parallel" leads to a completely different new solution to solve this FSI. Based on extensive FSI simulations with Adina of various blade designs, we have observed that the velocity of the rotor can be represented accurately with few Fourier modes:

$$\frac{\partial \Phi}{\partial t} = \Phi_0 + \Phi_1 \sin(\Theta) + \Phi_2 \cos(\Theta) + \cdots .$$

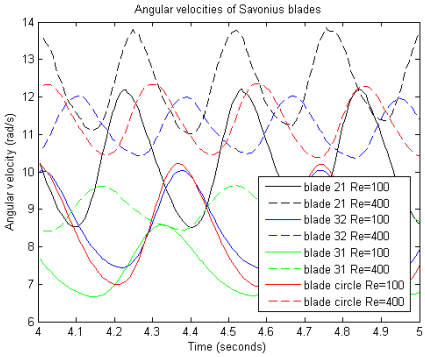

Fig. 4. Comparison of various Blades at different Reynolds number

Figure 4 gives a representative example of such an Adina calculation of the rotating velocity speed of the blade. Since we are interested in comparing design to take decision, it does not take a lot of accuracy to compare blade performances [6].

Our solution is then to apply a forcing speed to the blade and compute the torque exerted on the rotor. We first generate a surface response that approximates the torque with a family of a few coefficient in the Fourier expansion. The idea is second to optimize the periodic rotating speed $\frac{\partial \Phi}{\partial t}$ function on the surface response that satisfies at best the second Newton law:

$$\min_{(\varphi_0,\varphi_1,\varphi_2,\cdots)} \left\| I\frac{\partial^2 \Phi}{\partial t^2} - T \right\|_{(0,P)}. \tag{4}$$

Solving this minimization problem requires a regularization. We can indeed post-process the noisy results of the torque calculation with various angular speed obtained with the penalty method and modest grid size; see Fig. 5. Figure 6 is the result of Fourier filtering on the data of Fig. 5 with a second order filter. This regularization makes the minimization of (4) easy to process and robust with respect to noisy torque calculation. More importantly once the surface response for $T(\varphi_0, \varphi_1, \varphi_2, \dots)$ is generated, we can solve the optimum design with different load on the wind mill, by changing our objective function (4) only.

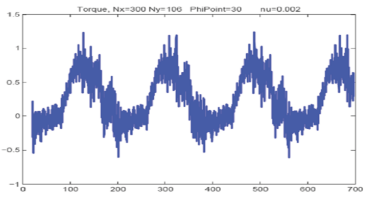

Fig. 5. Non-filtered torque (Reynolds = 250)

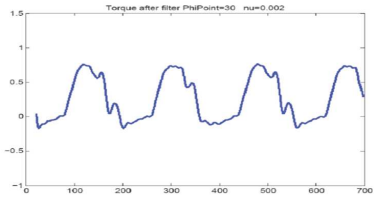

Fig. 6. Filtered torque (Reynolds = 250)

We shall now discuss the potential of this method for parallel processing.

4 Parallel Computing Scenario and Conclusion

In the short history of parallel computing, the tendency has been to solve larger and larger problems to get performance rather than reducing the exectution time for fixed (modest) size problems. The second is needed in optimum design while the first is for grand challenge problems only. Nowadays computers have hundreds of processors, and most standard algorithms with modest grid size problems cannot take advantage of this potential. The Sicortex system for example offers a very cost effective 72 cores parallel system in a standard desktop PC box format, that uses 200 Watts only. This sounds as a good motivation to come up with algorithms for small problem such as the two dimension NS FSI problem considered in this paper and which can take advantage of such a resource.

We observe that sampling the space for low order speed approximation (4) generates $O(100)$ independent tasks with embarrassing parallelism. Comparing design between various blade shapes can be done either by surface response and/or stochastic algorithms such as genetic algorithm or alternatively particle swarm algorithm. This adds a second level of large scale parallelism. We speculate that this approach that relies heavily on the robustness of the (parallel) domain decomposition CFD solver can run with volunteer computing effort such as offered by BOINC [1].

To conclude this paper, we have presented the SOFT concept to design VAFT *automatically* and a domain decomposition algorithm that can be a robust numerical engine for the FSI simulation of the VAFT. We found in our recent experience that this project had a positive impact to motivate our students in science and possibly improve our student enrollment. It is somewhat fascinating to our students to build real turbine with a numerical algorithm. We are currently running simulations to test the limits of our FSI/Immersed Boundary approach with Reynolds numbers much larger than in the present paper. This is, indeed, a very critical issue for the applicability of our method. However we believe that in principle one can reuse any existing NS codes into our FSI and optimization design framework to tackle larger windmill designs, that have to run in the turbulent boundary atmospheric layer where urban VAFTs should operate.

References

[1] Anderson, D.P., Christensen, C., Allen, B.: Designing a runtime system for volunteer computing. In *SC '06. Proceedings of the ACM/IEEE SuperComputing 2006 Conference*, pages 33–43, November 2006.

[2] Angot, P., Bruneau, C.-H., Fabrie, P.: A penalization method to take into account obstacles in incompressible viscous flows. *Numer. Math.*, 81(4):497–520, February 1999.

[3] Arquis, E., Caltagirone, J.P.: Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide-milieux poreux: Application a la convection naturelle. *C. R. Acad. Sci. Paris Sér. II*, 299:1–4, 1984.

[4] Garbey, M., Vassilevski, Y.V.: A parallel solver for unsteady incompressible 3d navier-stokes equations. *Parallel Comput.*, 27(4):363–389, March 2001.

[5] Hadri, B., Garbey, M.: A fast Navier-Stokes flow simulator tool for image based CFD. *J. Algorithms Comput. Technol.*, 2(4): 527–556, 2008.

[6] Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Kevin, T.P.: Surrogate-based analysis and optimization. *Progr. Aerospace Sc.*, 41(1):1–28, January 2005.

[7] Schneider, K., Farge, M.: Numerical simulation of the transient flow behaviour in tube bundles using a volume penalization method. *J. Fluids Struct.*, 20(4): 555–566, May 2005.

[8] Shyy, W., Lian, Y., Tang, J., Viieru, D., Liu, H.: *Aerodynamics of Low Reynolds Number Flyers*. Cambridge Aerospace Series. Cambridge University Press, Cambridge, MA, 2007.