World Academy of Science, Engineering and Technology
International Journal of Aerospace and Mechanical Engineering
Vol:12, No:5, 2018

# A Fast, Portable Computational Framework for Aerodynamic Simulations

Mehdi Ghommem, Daniel Garcia, Nathan Collier, Victor Calo

***Abstract*—**We develop a fast, user-friendly implementation of a potential flow solver based on the unsteady vortex lattice method (UVLM). The computational framework uses the Python programming language which has easy integration with the scripts requiring computationally-expensive operations written in Fortran. The mixed-language approach enables high performance in terms of solution time and high flexibility in terms of easiness of code adaptation to different system configurations and applications. This computational tool is intended to predict the unsteady aerodynamic behavior of multiple moving bodies (e.g., flapping wings, rotating blades, suspension bridges...) subject to an incoming air. We simulate different aerodynamic problems to validate and illustrate the usefulness and effectiveness of the developed computational tool.

***Keywords*—**Unsteady aerodynamics, numerical simulations, mixed-language approach, potential flow.



## I. Introduction

THE performance of aerodynamic systems, such as air vehicles, suspension structures, and wind turbines, could be assessed at the earliest stages of design through the deployment of computational tools. Recent advances in computer hardware and software have overcome the computational burden associated with the numerical integration of the equations governing the aerodynamic performance of the aforementioned systems. However, large scale and intensive computations still require the use of compiled languages (e.g., C/C++, Fortran) to obtain reasonable simulation times. Integrating the set of flow governing equations into a single large code (for instance, completely written in Fortran) may be complex for users and scientists and would lack flexibility and easiness in adapting different configurations and applications. The demanded flexibility can be hardly achieved by deploying such high-performance programming language. Python presents a convenient and open source working environment but remains inappropriate for intensive computation. As such, the mixed-language approach seeks to resolve the issues related to both performance and flexibility. The concept of a computational platform supporting data exchange between scripts programmed in different languages has been previously employed for multi-disciplinary optimization (e.g., pyOPT [1], and pyMDO [2]), high-fidelity simulations of nonlinear hyperbolic PDEs (e.g., SOLVCON [3]), and isogeometric analysis (e.g., PetIGA [4]).

In this paper, we present a fast and efficient numerical implementation of a potential flow solver based on the unsteady vortex lattice method (UVLM). This computational tool is designed to simulate the unsteady aerodynamic behavior of a wide range of moving bodies. The aerodynamic loads are obtained by pressure differences across the body surface resulting from acceleration- and circulation-based phenomena. The UVLM formulation accounts for unsteady effects such as added mass forces, the growth of bound circulation, and the wake. The UVLM has been widely used for the analysis and design of avian-like flapping wings in forward and hover flights [5]-[8], modeling of wind turbines [9]-[11], and dynamic analysis of offshore structures [12].

The goal of the computational tool is to provide a friendly framework for aerodynamic simulations based on UVLM. We achieve this by using a mixed-language programming paradigm. We employ Python for high-level management of grid objects and processing the aerodynamic quantities while Fortran is used for the computationally-demanding kernels.

Mehdi Ghommem is with the American University of Sharjah, United Arab Emirates (e-mail: mghommem@aus.edu).

## II. Aerodynamics Simulator Description

The unsteady vortex lattice method (UVLM) facilitates the study of the aerodynamic effects in slender bodies. Vorticity is the main fluid quantity used in UVLM to compute the dynamic evolution of fluids. The vorticity is generated along the boundary layer as a consequence of the viscous forces and advected downstream into the fluid. The region of the fluid enclosing the vorticity is called wake. In UVLM, the regions confining the vorticity are considered thin enough to be described as sheets of vorticity. This consideration agrees with the exclusion of viscous effects due to the assumed ideal potential flow. The vortex sheet description for the case of a lifting body submerged in a fluid is illustrated in Fig. 1. The vortex sheet description consists of a bound vortex and a free vortex. The bound vortex sheet describes the boundary layer of the lifting body. The displacement of the bound vortex is prescribed as a result of the solid surface of the body and a jump in the pressure may exist across this segment of the sheet. The free vortex sheet describes the wake and has no prescribed displacement implying that the sheet deforms freely. In the free vortex, there is no jump in pressure due to the free deformation of the wake. The only flow separation considered in the vortex sheet description is performed at the trailing edge connecting the bound vortex and the free vortex, implying the Kutta condition is satisfied in UVLM. The implementation of UVLM is based on discretizing the vortex sheets in a lattice of vortex elements known as vortex rings. The generated aerodynamics loads (lift and induced drag) are computed from the vortex rings' circulations obtained by imposing the no-penetration condition along the body surface.

The computational tool is designed to handle a wide range of aerodynamic problems of different scales. However, as

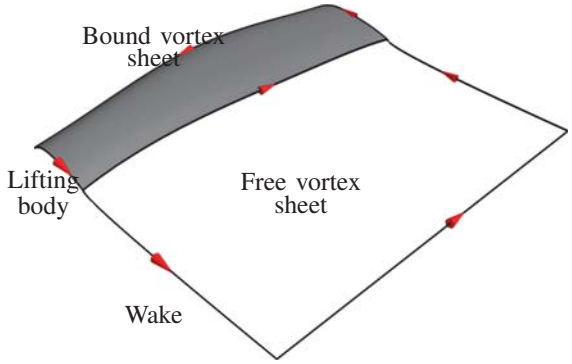
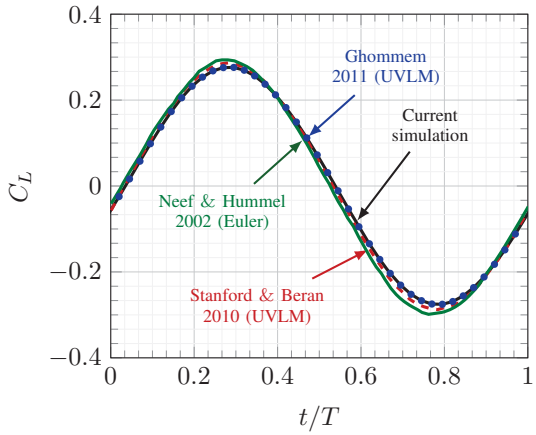
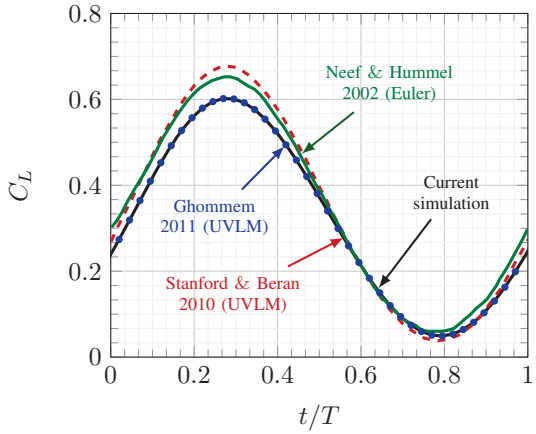



Fig. 1 Bound vortex description




(a) Pitch $\alpha = 0°$




(b) Pitch $\alpha = 4°$

Fig. 2 Variations of the lift coefficient over one flapping cycle: comparison against other flow solvers

the domain grows, the number of pairwise interactions to be accounted for grows significantly. This leads to a major increment in the computational expenses of forming the algebraic system and inverting the resulting dense matrix. Thus, we propose to decompose the domain into two regions and compute the contributions of the vortex segments in each part with two different mechanisms. In the proximal part of

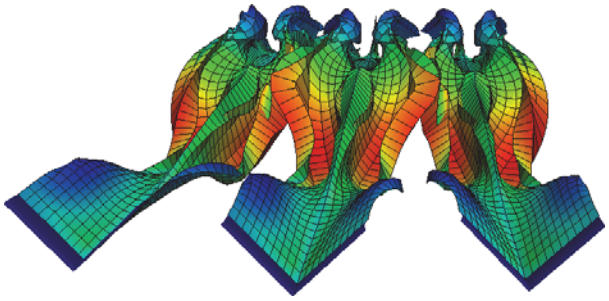


Fig. 3 Simulations of the wake patterns of the three flapping wings. Contour color levels denote the vorticity strength of the wake

the domain to the target point, we use the analytical expression based on Biot-Savart law to compute the contributions of the vortex sheets to the velocities. In the distal part of the domain, away from the point, we develop a pointwise approximation to compute the contributions. The process is recursively repeated until the contribution of all the vortex segments is evaluated on each target point. The partitioning of the domain allows to speed up further the simulations.

Our proposed numerical tool combining the use of compiled language (Fortran) and user-interactive problem solving environment (Python) presents several advantages:

- *Easy-to use and open source tool*. The computational tool will be made available as an open source software tool to serve the needs of users and developers seeking to simulate potential flow around moving bodies.
- *High flexibility and high performance*. The computational tool is designed to cover a broad range of applications (e.g., flapping wings, formation flight, rotating blades, and suspension bridges). The code interface uses Python to provide a flexible working environment for users. The code back-end uses Fortran, which is a low-level compiling language, to ensure reasonable simulation times.
- *Efficient and reliable*. The UVLM which presents the basis of the developed computational tool has been widely verified and used for the design and performance assessment of various aerodynamic systems. Furthermore, the tool has the capability to simulate multi-body interactions and enclosure effects (i.e., the proximity to obstacles such as ground and walls).
- *Potential extensibility*. The computational tool is intended to provide a powerful and extensible platform for the numerical simulations of aeroelastic systems. The tool also provides a solid foundation for being coupled to multidisciplinary optimization tools for design purposes.

## III. Results and Discussion

We simulate the response several aerodynamic systems to demonstrate the capability of the computational framework to handle a broad range of applications. The main purpose of these simulations is to compare the performance and accuracy of the proposed numerical approximation against standard UVLM implementation and other numerical methods.

We consider a flapping/twisting wing in forward flight. The wing has a rectangular shape with a root chord $c = 1\,cm$ and

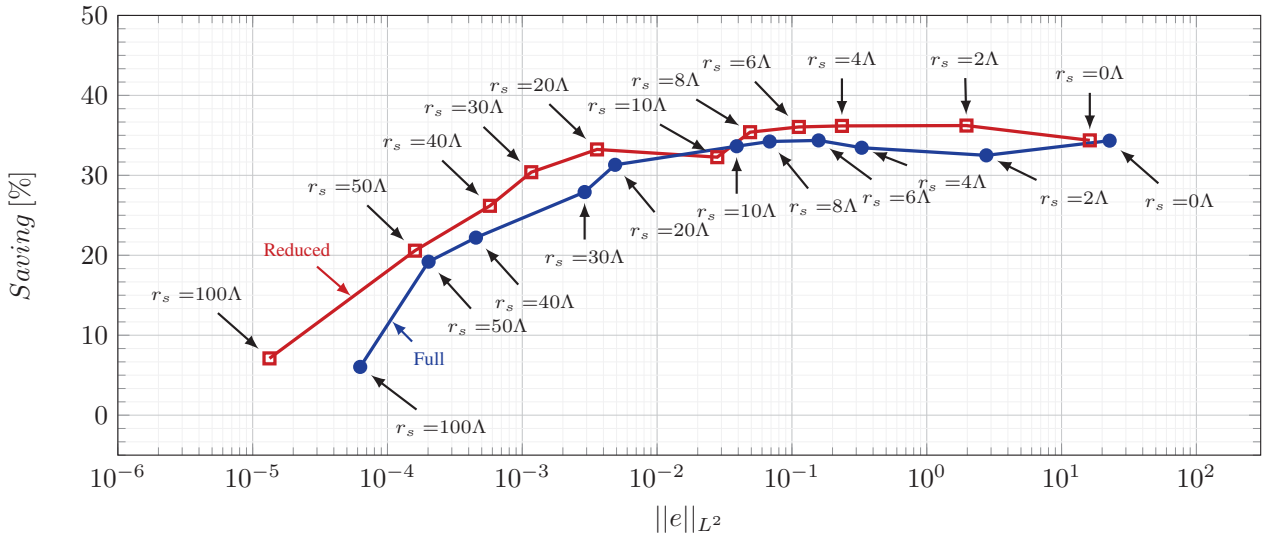



Fig. 4 Computational saving vs error for the three birds flapping wing problem

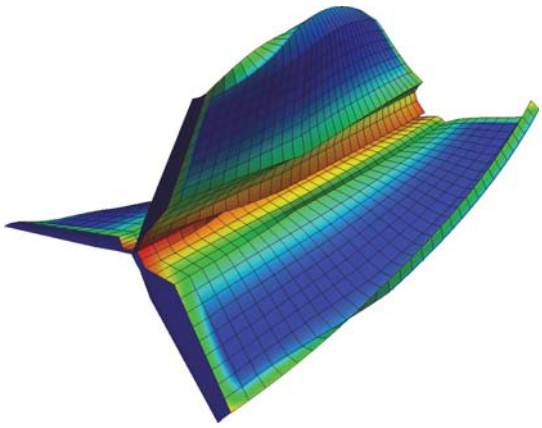


Fig. 5 Simulation of the flow past rotating blades

a wingspan $b = 8\,cm$. The transversal section of the wing consists of a NACA 0012 airfoil. The wing moves forward at a speed $\underline{v}$ and flaps with a maximum angle of $\theta_{max}$ and at a flapping frequency of $\omega_f$. A linear deformation along the span of the wing is introduced with a maximum twisting angle $\beta = 4°$ at the tip of the wing. The twisting is out of phase with the flapping by $\Psi_\beta = 90°$. The parameters used in the test case are presented in Table I.

TABLE I
FLAPPING/TWISTING WING PROBLEM DATA

| Parameter | |
|---|---|
| Flight velocity ($\underline{v}$ in $m/s$) | 10 |
| Flapping frequency ($\omega_f$ in $Hz$) | 2 |
| Reduce frequency ($k = \omega_f/2/\mathrm{u}_\infty$) | 0.1 |
| Flapping amplitude ($\theta_{max}$ in °) | 15 |
| Maximum twisting ($\beta$ in °) | 4 |
| Twisting phase angle ($\Psi_\beta$ in °) | 90 |
| Pitch ($\alpha$ in °) | 0/4 |
| Flapping period ($T = 2\pi/\omega$ in $s$) | $\pi$ |
| Time step ($t = T/40$ in $s$) | $\pi/40$ |

In Fig. 2, we plot the lift coefficient for one flapping cycle. The pitch angle is set equal to $0^o$ and $4^o$. The current simulation results compare very well with those obtained from the Euler flow simulations preformed in [13] and previous studies using UVLM-based implementations [5, 6], showing the capability of the computational tool to predict accurately the unsteady aerodynamic loads generated from flapping wings interacting with air.

The flight of three birds in a *V-shape* formation flight is considered as one of the numerical test problems. We use the computational tool to simulate the flow over flapping wings flying in grouping arrangements and in proximity of each other. This numerical example illustrates the ability of the developed tool to handle multiple bodies. The analysis of formation flights can be helpful to develop an understanding of the potential power saving that birds can achieve through organized patterns when traveling over long distances without stopping and feeding [14]. Fig. 3 shows the three flapping wings and the shed wake. This aerodynamic problem is solved using two methods: the first uses the full domain and the second assumes a reduced domain by considering the symmetry of the problem.

To assess the potential of the proposed pointwise approximation to speed up the simulations while maintaining good accuracy, the $L^2$ error norm of the circulation on the bound sheet, given by (1), is evaluated. Note that only the error norm in the circulation of the bound sheet is considered as this circulation determines all the aerodynamics loads [15].

$$||\bar{e}||_{L^2} = \left\langle \left( \sum_{i=0}^{elements} |\Gamma_\infty^i - \Gamma_{r_s}^i|^2 \right)^{0.5} : 0 < r_s < \infty \right\rangle^{max}. \quad (1)$$

Here, $\Gamma_\infty^i$ refers to the circulation of the $i$-th element on the bound sheet computed with the analytical method and $\Gamma_{rs}^i$ is the circulation computed with the proposed approach and using the close-to-target parameter $r_s$ that splits the simulated domain into two regions. $\langle\rangle^{max}$ refers to the maximum value of error norm among all the time steps. The computational time of solving one time step is used to estimate

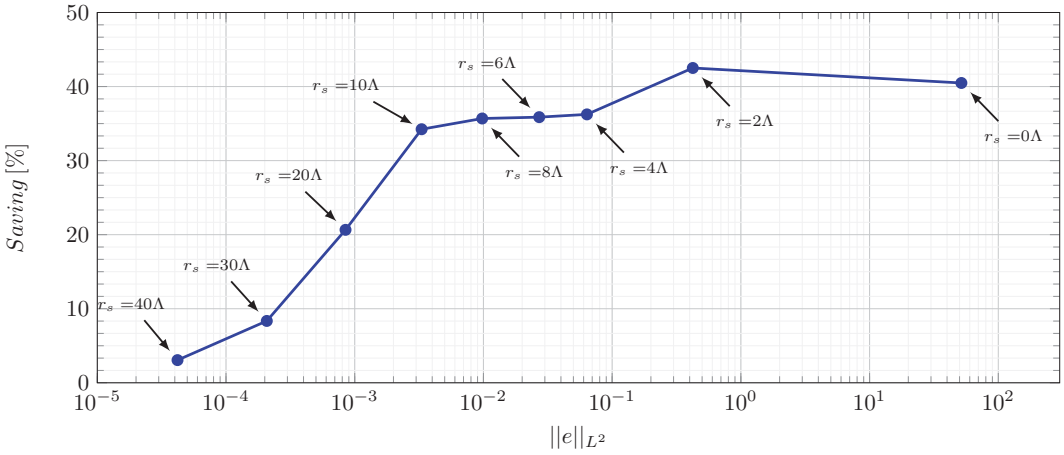



Fig. 6 Computational saving vs error for the rotating blades problem

the computational cost. Each time step, the computational tool builds an algebraic system to compute the bound circulation, updates the lattice position, transports the bound circulation into the free vortex and finds the new wake distribution. The cost is estimated as the average computational time over all the time steps.

A relevant measure to assess the performance of the tool is the saving in the computational time, achieved when employing the proposed pointwise approximation, relative to the error. The computation of this saving is performed based on the estimates of cost and error as described above. The average saving in the computational time relative to the error obtained when varying the values of the close-to-target parameter $r_s$ is presented in Fig. 4. The maximum saving in the computational time that the pointwise approximation yields is $32\%$ when considering the full domain and $44\%$ in the case of the reduced domain. These values are not entirely practical because the maximum saving results in high error. Reasonable savings in computational time are achieved when the close-to-target parameter $r_s$ is taken bigger than $40$. For instance, the error observed with $r_s = 40$ is of order $\mathcal{O}(10^{-3})$ and the saving reaches a maximum of $29\%$. Moreover if an error of order $\mathcal{O}(10^{-5})$ is required in the solution, a saving in computational time of $5\%$ is still achievable.

Next, we simulate the aerodynamic response of rigid rotating blades subjected to an incoming steady freestream flow. We consider a wind turbine composed of three blades, each $120$ degrees out of phase from each other. We discretize the problem using six elements along the chordwise direction and $14$ elements along the blade spanwise direction. The transversal section of the blade is a NACA83XX airfoil.

We use the computational framework to simulate the interactions of the multiple grids (three blades) while varying the values of the parameter $r_s$. Fig. 5 shows the formation and shedding of the bound vortices into the wake. The roll-up of the wake vortices shed from the blades trailing edge can be observed.

To show the tradeoffs between the possible computational saving (obtained from the pointwise approximation) and the loss in the aerodynamic solution accuracy, we plot in Fig. 6 the variations of the computational saving with the error when varying the values of the parameter $r_s$. We observe a nearly-linear trend followed by a stabilization indicating that setting the parameter $r_s$ greater than a certain value (8 $\Lambda$) does not speed up significantly the simulations but this has a noticeable effect on the accuracy of the solution as can be deduced from the large errors. Assuming an error of approximation in the order of $\mathcal{O}\left(10^{-3}\right)$, a maximum saving in time of $20\%$ is obtained. These results demonstrate the capability of the computational framework to speed up the aerodynamic simulations while keeping a good level of accuracy. Furthermore, the user has the possibility to select the potential saving in the computational cost based on the required accuracy of the simulated aerodynamic problem and the objective of the study. For instance, optimization and sensitivity analyses require running the forward aerodynamic problem several times and then considering ways to speed up the simulations such as the proposed pointwise approximation will be helpful to conduct such analyses within a reasonable time frame.

Other aerodynamic problems are also simulated to validate the numerical prediction of the aerodynamic loads against higher-fidelity solvers and assess further the performance of the developed computational tool.

## IV. Conclusions

In this work, we developed and tested a fast and user-friendly computational framework for aerodynamic simulations. The computational tool is based on mixed-language programming approach combining Python for high-level management of grid objects and processing the aerodynamic quantities and Fortran for the computationally-demanding kernels. We used this tool to simulate a set of aerodynamic problems and demonstrate its flexibility and efficiency. We also showed the potential computational saving in the simulations thanks to the new implementation of the unsteady vortex lattice method.

## REFERENCES

[1] R. E. Perez, P. W. Jansen, J. R. R. A. Martins, pyOpt: a python-based object-oriented framework for nonlinear constrained optimization, Structural and Multidisciplinary Optimization 45 (1) (2012) 101 – 118.
[2] J. J. Alonso, P. LeGresley, E. van der Weide, J. R. R. A. Martins, J. J. Reuther, pymdo: A framework for high-fidelity multi-disciplinary optimization, in: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA 20044480, 2004.
[3] Y.-Y. Chen, D. L. Bilyeu, L. Yang, S.-T. J. Yu, Solvcon: A python-based cfd software framework for hybrid parallelization, in: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, AIAA 2011-1065, 2011.
[4] L. Dalcin, N. Collier, P. Vignal, A. M. A. Cortes, V. M. Calo, Petiga: A framework for high-performance isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 308 (2016) 151–181.
[5] M. Ghommem, M. R. Hajj, D. T. Mook, B. K. Stanford, P. S. Beran, L. T. Watson, Global optimization of actively-morphing flapping wings, Journal of Fluids and Structures 30 (2012) 210–228.
[6] B. K. Stanford, P. S. Beran, Analytical sensitivity analysis of an unsteady vortex-lattice method for flapping-wing optimization, Journal of Aircraft 47 (2010) 647–662.
[7] A. T. Nguyen, J.-K. Kim, J.-S. Han, J.-H. Han, Extended unsteady vortex-lattice method for insect flapping wings, Journal of Aircraft 0 (2016) 1–10.
[8] J. D. Colmenares, O. D. Lpez, S. Preidikman, Computational study of a transverse rotor aircraft in hover using the unsteady vortex lattice method, Mathematical Problems in Engineering 2015, article ID 478457.
[9] A. Rosenberg, A. Sharma, A prescribed-wake vortex lattice method for preliminary design of co-axial, dual-rotor wind turbines, Journal of Solar Energy Engineering 138 (2016) 1–9.
[10] B. F. Ng, H. Hesse, R. Palacios, J. M. R. Graham, E. C. Kerrigan, Aeroservoelastic state-space vortex lattice modeling and load alleviation of wind turbine blades, Wind Energy 18 (2015) 1317–1331.
[11] G. Tescione, C. S. Ferreira, G. van Bussel, Analysis of a free vortex wake model for the study of the rotor and near wake flow of a vertical axis wind turbine, Renewable Energy 87 (2016) 552–563.
[12] M. Jeona, S. Leea, S. Leeb, Unsteady aerodynamics of offshore floating wind turbines in platform pitching motion using vortex lattice method, Renewable Energy 65 (2014) 207–212.
[13] M. F. Neef, D. Hummel, Euler Solutions for a Finite-Span Flapping Wing in Mueller T. J. (ed.), Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications, American Institute of Aeronautics and Astronautics, Inc., Reston, 2004.
[14] M. Ghommem, V. Calo, Flapping wings in line formation flight: a computational analysis, The Aeronautical Journal 118 (2014) 485–501.
[15] J. Katz, A. Plotkin, Low-Speed Aerodynamics, Cambridge University Press, MA, 2001.