Accepted Manuscript

PyFly: A fast, portable aerodynamics simulator

D. Garcia, M. Ghommem, N. Collier, B.O.N. Varga, V.M. Calo

PII: S0377-0427(18)30123-7
DOI: https://doi.org/10.1016/j.cam.2018.03.003
Reference: CAM 11550

To appear in: *Journal of Computational and Applied Mathematics*

Received date : 11 March 2017
Revised date : 1 March 2018

PyFly: A Fast, Portable Aerodynamics Simulator

D. Garcia[a,b], M. Ghommem[c,*], N. Collier[d], B.O.N. Varga[e], V. M. Calo[f,g]

[a] *Basque Center for Applied Mathematics, (BCAM), Bilbao, Spain*
[b] *University of the Basque Country (UPV/EHU), Leioa, Spain*
[c] *Department of Mechanical Engineering, American University of Sharjah (AUS), Sharjah, United Arab Emirates*
[d] *Oak Ridge National Laboratory, Oak Ridge, TN, USA*
[e] *Earth Science & Engineering, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia*
[f] *Applied Geology Department, Curtin University, Perth, WA, Australia*
[g] *Mineral Resources, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Kensington, WA, Australia*

Abstract

We present a fast, user-friendly implementation of a potential flow solver based on the unsteady vortex lattice method (UVLM), namely PyFly. UVLM computes the aerodynamic loads applied on lifting surfaces while capturing the unsteady effects such as the added mass forces, the growth of bound circulation, and the wake while assuming that the flow separation location is known a priori. This method is based on discretizing the body surface into a lattice of vortex rings and relies on the Biot-Savart law to construct the velocity field at every point in the simulated domain. We introduce the pointwise approximation approach to simulate the interactions of the far-field vortices to overcome the computational burden associated with the classical implementation of UVLM. The computational framework uses the Python programming language to provide an easy to handle user interface while the computational kernels are written in Fortran. The mixed language approach enables high performance regarding solution time and great flexibility concerning easiness of code adaptation to different system configurations and applications. The computational tool predicts the unsteady aerodynamic behavior of multiple moving bodies (e.g., flapping wings, rotating blades, suspension bridges) subject to incoming air. The aerodynamic simulator can also deal with enclosure effects, multi-body interactions, and B-spline representation of body shapes. We simulate different aerodynamic problems to illustrate the usefulness and effectiveness of PyFly.

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

1. 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 C++) may be complex for users and scientists and may lack flexibility and easiness in adapting different configurations and applications. The needed flexibility is difficult to achieve in high-performance programming languages. 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]), and high-fidelity simulations of PDEs (e.g., SOLVCON [3], and FEniCS [4]). The same concept has been also used for constructing

*Corresponding author
Email address: mghommem@aus.edu (M. Ghommem)