SoftwareX 13 (2021) 100655

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

FLUBIO—An unstructured, parallel, finite-volume based Navier–Stokes and convection–diffusion like equations solver for teaching and research purposes

Edoardo Alinovi, Joel Guerrero *

DICCA, Scuola Politecnica, University of Genoa, 1 via Montallegro, 16145 Genoa, Italy

ARTICLE INFO

Article history:
Received 14 September 2020
Received in revised form 20 November 2020
Accepted 29 December 2020

Keywords:
Finite volume method
CFD
Convection–diffusion
Navier–Stokes
PETSc

ABSTRACT

FLUBIO is a parallel, unstructured, finite-volume based solver for the solution of the Navier–Stokes equations and convection–diffusion like equations. The solver is written using modern Fortran (2003+ standard), it is object-oriented, and is organized in such a way that it is easy to understand and modify. The solver is targeted at students, academics, personal users and practitioners to help them understand the general theory behind modern CFD solution methods and discretization techniques. The solver is also general and capable enough to deal with industrial and academic problems found in the field of fluid dynamics.

Code metadata

Current code version	version 1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00046
Legal Code License	CC BY-NC
Code versioning system used	Git
Software code languages, tools, and services used	Fortran, MPI, PETSc, JSON, VTK
Compilation requirements, operating environments & dependencies	Unix-Linux like OS, gfortran, MPI, PETSc. The installation instructions are provided in the document FLUBIO_INSTALL.pdf located in the repository
If available Link to developer documentation/manual	https://gitlab.com/alie89/flubio-code-fvm
Support email for questions	Gitlab issue tracker section

1. Motivation and significance

Computational Fluid Dynamics (CFD) deals with the numerical solution of the governing equations of fluid dynamics. This discipline finds a wide range of applications within the aerospace, automotive, naval, chemical, biomedical, pharmaceutical, energy generation, and environmental industries, among many. Furthermore, it is an essential tool for academic research in any field that deals with fluid dynamics, parallel computing, and numerical methods.

Among the numerical methods used to solve the conservation laws, which are the base of the governing equations of fluid dynamics, maybe the finite volume method (FVM) is the most popular one. Its popularity relies on its theoretical simplicity and the fact that it enforces mass conservation (local and global). The FVM method is the kernel of many commercial and open-source CFD solvers, to name a few, Ansys Fluent [1], StarCCM+ [2], OpenFOAM [3], and SU2 [4].

As any other numerical technique, the FVM is strongly linked to computer science, since the bridge between theory and practice takes the form of scientific software. Writing a solver, even if it is a simple one used to solve the convection–diffusion equation, is the most instructive task that one could do in order to understand the details of the theory behind the FVM and to become functional in the use of modern programming languages and parallel computing.

Among many of the open-source CFD solvers available, perhaps OpenFOAM [3] and SU2 [4] are the most popular ones. These software libraries are well designed and offer a broad set of capabilities that allow new users to deal with complex multi-physics problems and advanced optimization techniques

* Corresponding author.
E-mail addresses: edoardo.alinovi@dicca.unige.it (Edoardo Alinovi), joel.guerrero@unige.it (Joel Guerrero).

https://doi.org/10.1016/j.softx.2020.100655

Table 1
Different discretization techniques implemented in FLUBIO. Further information about the discretization techniques can be found in Refs. [5–9].

Term	Brief description
Time derivative	Methods for iterative and time marching. Methods implemented: Steady-state, Euler, Backward differencing, Crank–Nicolson.
Diffusive term	Weighted central differences with non-orthogonal corrections. Three methods available, namely, over-relaxed, orthogonal correction, and minimum correction.
Convective term	State of the art convective schemes applicable to unstructured meshes. Many methods implemented, to name a few, upwind, central differences, second-order upwind, QUICK, minmod, vanleer, superbee. Fluxes are assembled in a fully implicit way or by using the deferred correction approach.
Gradient terms	Cell-based Green–Gauss and least-squares. To enforce monotonicity and improve solution stability, gradient limiters can be used. Face-limited and cell-limited versions are available.

(e.g., adjoint optimization). However, they are difficult to understand, as many of the implementation details are obfuscated due to the use of advanced programming techniques that entry-level users do not understand. Also, the lack of documentation makes it even harder to understand how to use or program on these libraries. In addition, the information (and misinformation) spread among many forums (often non-official ones) can confuse new users. Doing simple modifications to these libraries can be a daunting task for newcomers.

As a result, many students and practitioners are often left to face complex CFD solvers with little support and a very poor understanding of what is happening under the hood. And even worse, many beginner CFD practitioners get frustrated and try to tackle complex problems using CFD solvers with limited capabilities, do not use parallel computing, are not compatible with the physics of interest, or have not undergone proper validation and verification.

With the aim at offering an easy to understand and easy to modify CFD solver, and at the same time with discretization capabilities similar to those available in the most popular open-source solvers available in the web (e.g., OpenFOAM [3] and SU2 [4]), we introduce FLUBIO, an unstructured, parallel, finite volume based Navier–Stokes and convection–diffusion like equations solver. The solver is cell-centered, and it uses segregated methods to deal with the pressure–velocity coupling of the Navier–Stokes equations (e.g., SIMPLE method [10] and PISO method [11]). To avoid the checkerboard instability, the Rhie–Chow interpolation is used [12]. Many high-order and high-resolution methods are implemented, which guarantees at least second-order accurate solutions. The time-derivative, convective terms, diffusive terms, and gradient terms appearing in the governing equations can be discretized in a term-by-term basis; that is, the user can use any combination of different numerical schemes. Finally, to enforce the monotonicity principle, slope limiters are implemented.

FLUBIO can deal with two-dimensional and three-dimensional domains. It uses unstructured meshes, and in theory, can handle any cell element type (tetrahedrons, hexes, prisms, pyramids, and general polyhedrons). It has been fully parallelized by using domain decomposition and the message passing interface library Open MPI [13]. To solve the linear systems arising from the discretization of the governing equations, FLUBIO uses the high performance computing library PETSc [14], which is a collection of routines for efficient and scalable scientific computing.

FLUBIO is written using plain Fortran 2003 standard, and with object orientation in mind. Derived types and classes are used to encapsulate data and methods at the core solver level, making it easy to reuse and modify. Inheritance is kept at a single level almost everywhere. Bespoke names for variables and subroutines are employed, and the same coding standard is adopted all over the code for clarity and understandability.

Finally, the code has been extensively commented, so new users will not get lost when reading the source code. In addition, easy to browse documentation is automatically generated using Ford [15]. We aim to translate what can be studied in the textbooks into an easy to read computer code, with no tricks, and in the simplest possible way to maintain a close link with theory. The code has been thought to provide a fast access to the core discretization techniques used in CFD. On the one hand, newcomers in the field of CFD and computer programming can get a better view of the building blocks and how to implement CFD solvers. On the other hand, experienced users can take advantage of the code to develop more sophisticated solvers for their research activity in the field of CFD.

2. Software description

The general transport equation (Eq. (1)), integrated over a given control volume V_P, is used throughout this discussion to briefly describe the FVM discretization practices underlying FLUBIO. In this equation, ϕ is the transported quantity, *i.e.*, velocity, pressure, temperature, and so on, and Γ_ϕ is the diffusion coefficient of the transported quantity.

$$\int_{V_P} \underbrace{\frac{\partial \rho\phi}{\partial t} dV}_{\text{time derivative}} + \int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u}\phi)\, dV}_{\text{convective term}} - \int_{V_P} \underbrace{\nabla \cdot \left(\rho \Gamma_\phi \nabla\phi\right) dV}_{\text{diffusive term}} \ldots = \int_{V_P} \underbrace{S_\phi(\phi)\, dV}_{\text{source term}} \tag{1}$$

FLUBIO has several dedicated modules covering state-of-the-art discretization techniques targeting each term appearing in Eq. (1), which serves as a building block for any solver implemented in FLUBIO. An overview of FLUBIO's discretization options is given in Table 1.

After spatial and temporal discretization of the governing equations in every control volume of the domain, a system of differential algebraic equations of the form,

$$[A]\,[\phi] = [R]\,, \tag{2}$$

is assembled. In Eq. (2), $[A]$ is a sparse matrix with coefficients a_P in the diagonal and a_N off the diagonal, $[\phi]$ is the vector containing the unknown quantity ϕ in all control volumes, and $[R]$ is the vector containing boundary conditions, source terms, and explicit contributions arising from the discretization process. The coefficients a_P include the contribution from all terms corresponding to $[\phi_P^n]$, that is, the temporal derivative, convection and diffusion terms, and the linear part of the source term corresponding to the current time-step n. The coefficients a_N include the contributions corresponding to each term of the neighboring control volumes. The summation is done over all the control volumes that share a face with the current control volume V_P. The right-hand side of Eq. (2) (the vector R), includes all terms that can be evaluated without knowing the new value of the vector $[\phi]$, namely, the constant part of the source term, boundary conditions contribution, and the explicit contributions at the old-time step $n-1$ of

the convective and diffusive terms. When this system is solved, we obtain the solution for the vector $[\phi]$ at the new time step n. To solve the linear system arising from the discretization of the governing equations (Eq. (2)), we use the library PETSc [14], which provides a set of efficient, highly scalable, and validated iterative solvers and preconditioners targeted for the solution of large sparse linear system.

FLUBIO has been parallelized using the standard message passing interface communications protocol, implemented in Open MPI [13]. In particular, non-blocking communication between partitions is employed to exchange the values at neighboring cells, which are essential for fields interpolation and fluxes evaluation. The non-blocking nature of the communication allows the code to send messages and to receive messages from other partitions, avoiding in this way deadlocks. The MPI functionalities are implemented using low-level programming. Therefore, the user does not need to deal with MPI programming when using the code or when implementing new solvers or boundary conditions.

The solver supports fully unstructured meshes made up of hexes, prisms, tetras, and pyramids in 3D. In 2D, the solver supports quadrilateral and triangular elements. Hybrid meshes, made up by any combination of the cell types mentioned above, are also supported. In practice, the solver can support general polyhedral cells, but this feature will be added as soon as a clear need arises. For the moment, the solver does not handle hanging nodes.

FLUBIO does not provide built-in meshing capabilities, and in the current state, it relies on the OpenFOAM mesh format. The code can handle both serial and decomposed meshes. The domain decomposition must be carried out within the OpenFOAM environment by using the application decomposePar. Together with the mesh format, this is the only connection that FLUBIO shares with OpenFOAM. We are working on removing this constraint by implementing our own stand-alone decomposition application. We are also working to offer support to more general mesh and data structure formats, such as CGNS [16] or HDF5 [17].

Finally, the most commonly used boundary conditions are already implemented, and others can be easily added by following the existing templates. For further information about how to use the solver, we invite the interested reader to read the tutorial manual, which comes with the source code and is located in the examples directory.

3. Software architecture

FLUBIO is implemented as a library with high-level functions for the solution of the governing equations. In Fig. 1, we illustrate a nested diagram of the main building blocks in FLUBIO. This diagram gives a general view of the hierarchical organization of FLUBIO, where first the mesh is read, followed by the fields initialization. At this point, interpolation and gradient operations are applied to the fields; then, each term appearing in the governing equations is discretized, and finally, a solution method is selected to solve the equations. All these modules, constitute a solver in FLUBIO.

Let us address in more detail the nested diagram depicted in Fig. 1. At the base level of FLUBIO's organization, we find the mesh data structure and methods required to decompose the domain into a set of non-overlapping finite volume cells. This block is composed of a set of classes that can handle all the geometrical operations involved in the FVM. To name a few, computation of cells volume, cells center, faces area and surface normal vectors, cell neighbors, interpolation weights, and so on.

Once the mesh has been defined, the fields at each cell center and boundary face can be assigned and initialized. A field in FLUBIO is the numerical counterpart of a variable. It can be seen as a container on which many operations can be carried out, such as field interpolation and field gradient computation (which are fundamental for the evaluation of the face fluxes).

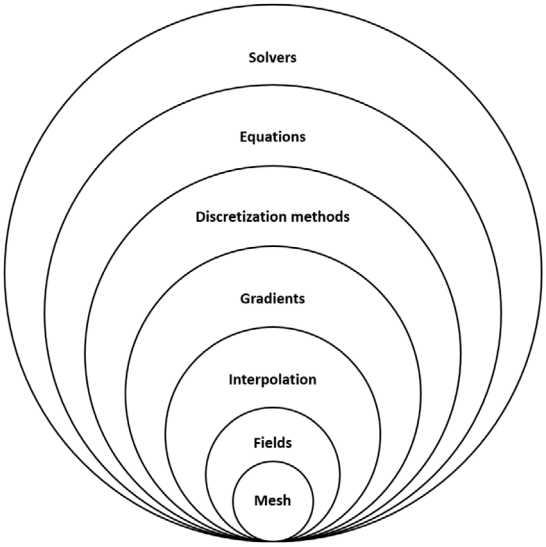

Fig. 1. FLUBIO building blocks.

Then, on top of the mesh and field operations, discretization methods are defined for each term appearing in Eq. (2). In FLUBIO, many discretization methods are available, and they can be assigned on a term-by-term basis. That is, each term appearing in the governing equations can use different methods with different order of accuracy.

Finally, different methods are implemented to deal with the solution of the governing equations. For example, to deal with the pressure–velocity coupling of the Navier–Stokes equations, the user can use any of the following segregated methods, SIMPLE [10], SIMPLEC [18], or PISO [11].

In FLUBIO, solvers and user cases are located in different directories, and they can be run from any directory in the system. Thus, each case is organized in a stand-alone folder, which must respect the following structure:

```
Case directory
├─ BCs: boundary conditions input files.
├─ grid: mesh.
│  ├─ processor*: decomposed mesh (for parallel runs).
│  ├─ serial: serial mesh.
│  └─ bin: mesh saved in binary format.
├─ physical: physical constants required by the solver.
├─ settings: numerical settings and simulation controls.
└─ postProc: post-processed fields and monitors.
   ├─ fields: fields saved in binary format.
   ├─ monitors: monitors and sampled output.
   └─ VTKfields: fields in binary XML format (*.vtu).
```

To understand how to use FLUBIO, the case structure, and basic case setup, we encourage the interested user to read the Ford documentation and the tutorial guide distributed with the source code.

3.1. Software functionalities

FLUBIO comes with several solvers, each one dealing with different governing equations commonly encountered in CFD, as listed in Table 2. For turbulence modeling, both Reynolds Averaged Navier–Stokes (RANS) and Large Eddy Simulation (LES)

Table 2
Solvers currently implemented in FLUBIO.

Solver Name	Target equation	Brief description
flubio_poisson	$\frac{\partial \phi}{\partial t} = \nabla \cdot \Gamma \nabla \phi + f$	Transient and steady state solver for the Poisson equation.
flubio_potential	$\nabla^2 \phi = 0$	Steady state potential flow solver.
flubio_transport	$\frac{\partial \phi}{\partial t} + \nabla \cdot \boldsymbol{u}\phi = \nabla \cdot \Gamma \nabla \phi + f$	Transient and steady state solver for the convection–diffusion equation.
flubio_simple	$\nabla \cdot \boldsymbol{u} = 0, \nabla \cdot \rho \boldsymbol{u}\boldsymbol{u} = -\nabla p + \nabla \cdot \mu \nabla \boldsymbol{u} + \boldsymbol{f}$	Steady state, incompressible, Navier–Stokes solver using the SIMPLE algorithm [10].
flubio_piso	$\nabla \cdot \boldsymbol{u} = 0, \frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot \rho \boldsymbol{u}\boldsymbol{u} = -\nabla p + \nabla \cdot \mu \nabla \boldsymbol{u} + \boldsymbol{f}$	Transient, incompressible, Navier–Stokes solver using the PISO algorithm [11].

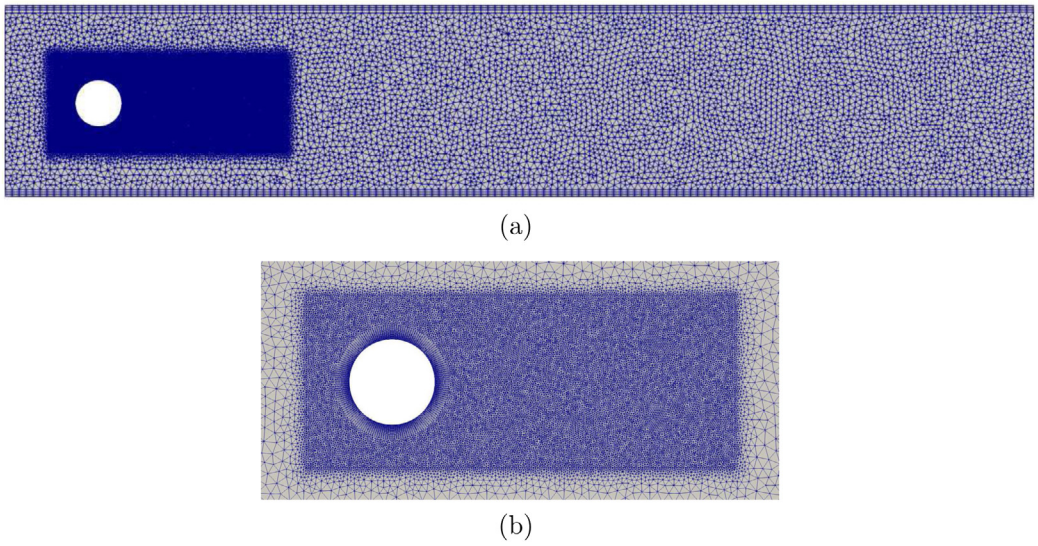

Fig. 2. (a) Computational domain and hybrid mesh used in the laminar flow around a cylinder study. (b) Close up of the hybrid mesh and refinement box in the vicinity of the cylinder.

models are available. To name a few of the turbulence models available, Spalart–Allmaras [19] and $k-\omega$ [20] for RANS modeling, and Smagorinsky [21] and Wall-adapting Local Eddy-viscosity (WALE) [22] for LES modeling. Due to the library's modular organization, new turbulence models can be added with minimal effort by simply following the existing templates.

To setup the simulation options and general run-time parameters, input files in ASCII format are parsed by the code at run-time. Many control options are exposed to the user, some of them having default values to avoid overwhelming the user with settings, especially for those users using the library for the first time or those unfamiliar with the FVM. A list of all the dictionary entries, and whenever they are mandatory or not, is specified in the documentation distributed along with the code.

4. Illustrative examples

4.1. Laminar flow around a cylinder

This test-case corresponds to the laminar flow around a cylinder at $Re = 100$, as described in Refs. [23,24]. The problem has been made non-dimensional using the cylinder's diameter D as length scale, the mean inlet flow velocity U_0 as velocity scale, and the ratio D/U_0 as the time scale. The results are compared against OpenFOAM [3] (same mesh and equivalent setup), and FeatFlow [25]. The comparison with FeatFlow is made using grid level number 4 (as reported in Refs. [23,24]), which has 42016 degrees of freedom. In Fig. 2, we show the mesh used in this case. The mesh is made up of triangular and rectangular elements and has 59428 hybrid cells. We used 20 prism layers at the cylinder wall, and at the top and bottom walls, we used 5 prism layers. Also, a refinement box was used in the vicinity of the cylinder to better resolve the near wake.

The backward Euler method is used to advance the simulation in time, with a time step equal to 5×10^{-3} (non-dimensional time units). The cell-based Green–Gauss method is used to approximate the gradients, while the second-order upwind scheme is used for the convective term. The diffusive terms are computed using centered differences with non-orthogonal corrections.

In Fig. 3, we show the contours of the velocity magnitude, relative pressure, and vorticity fields. In the same figure, we also show the time signal of the lift and drag coefficients. In Table 3, we compare the outcome of FLUBIO against OpenFOAM and FeatFlow. In the table, the lift coefficient C_L and drag coefficient C_D, are computed as follows,

$$C_L = \frac{L}{0.5\rho U_\infty^2 S_{ref}}, \quad C_D = \frac{D}{0.5\rho U_\infty^2 S_{ref}} \tag{3}$$

In Eq. (3), L is the lift force, D is the drag force, ρ is the reference density, U_∞ is the reference velocity, and S_{ref} is a reference surface (the cylinder diameter per unit depth in this case). In overall, we obtained a good agreement with the data found in the literature [23,24].

It is worth mentioning that this case is also used to demonstrate the capability of FLUBIO to deal with hybrid meshes, as well as the capacity of using non-uniform boundary conditions (the inlet velocity corresponds to a parabolic profile, as described in Refs. [23,24]).

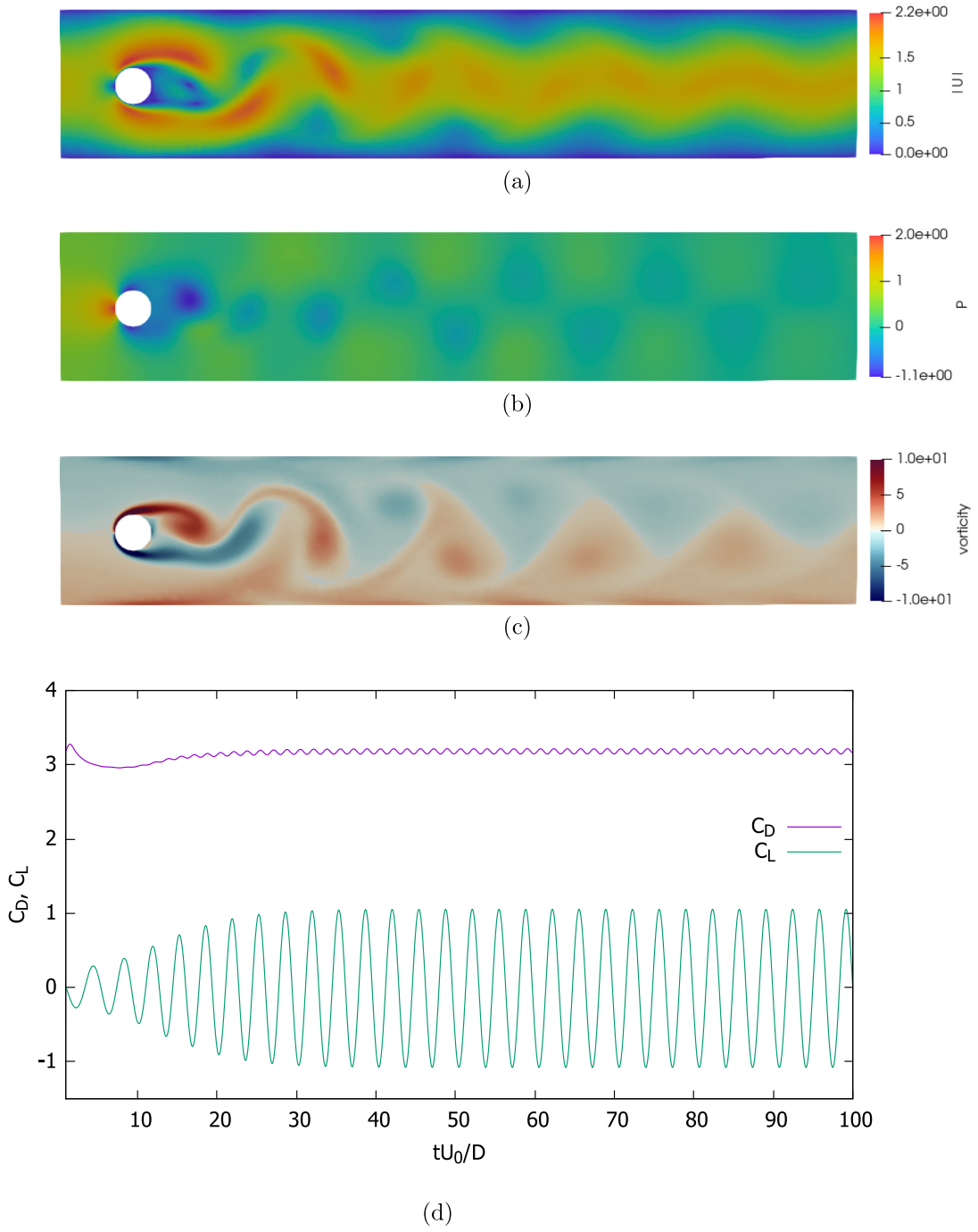

Fig. 3. Laminar flow around a cylinder. (a) Contours of velocity magnitude (m/s). (b) Contours of relative pressure (Pa). (c) Contours of vorticity ω_z (1/s). (d) Drag coefficient C_D and lift coefficient C_L time series.

Table 3
Comparison of the aerodynamic coefficients C_D and C_L (per unit depth), and Strouhal number (St). The superscripts *min* and *max* stand for minimum and maximum value, respectively.

Solver	C_D	C_D^{min}	C_D^{max}	C_L	C_L^{min}	C_L^{max}	St
FLUBIO	3.19	3.15	3.22	−1.048	1.013	−0.0175	0.29
OpenFOAM	3.20	3.17	3.24	−1.087	1.052	−0.0178	0.29
FeatFlow	3.16	3.13	3.20	−1.017	0.986	−0.0172	0.30

4.2. Turbulent flow past the ONERA-M6 wing

The ONERA-M6 wing is a widely used geometry in CFD to validate compressible solvers [26–28]. For the moment, FLUBIO cannot deal with compressible flows; therefore, we added a twist to this case in the sense that we worked with a turbulent incompressible flow.

The computational domain and mesh are shown in Fig. 4, and we used the mesh available from the NPARC verification and validation archive [29]. The mesh consists of 294912 hexahedral elements. The Reynolds number is equal to 10^6, and the wind

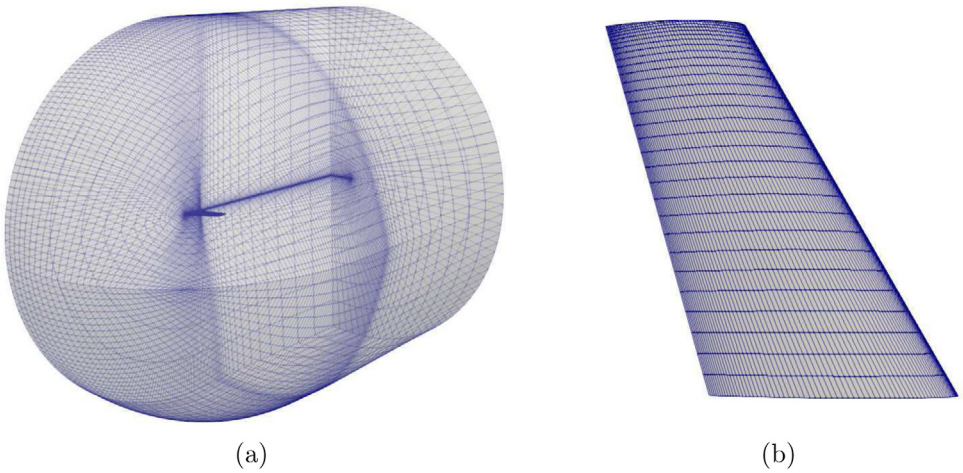

Fig. 4. Onera M6 test case. (a) Volume mesh and domain. (b) Wing surface mesh detail.

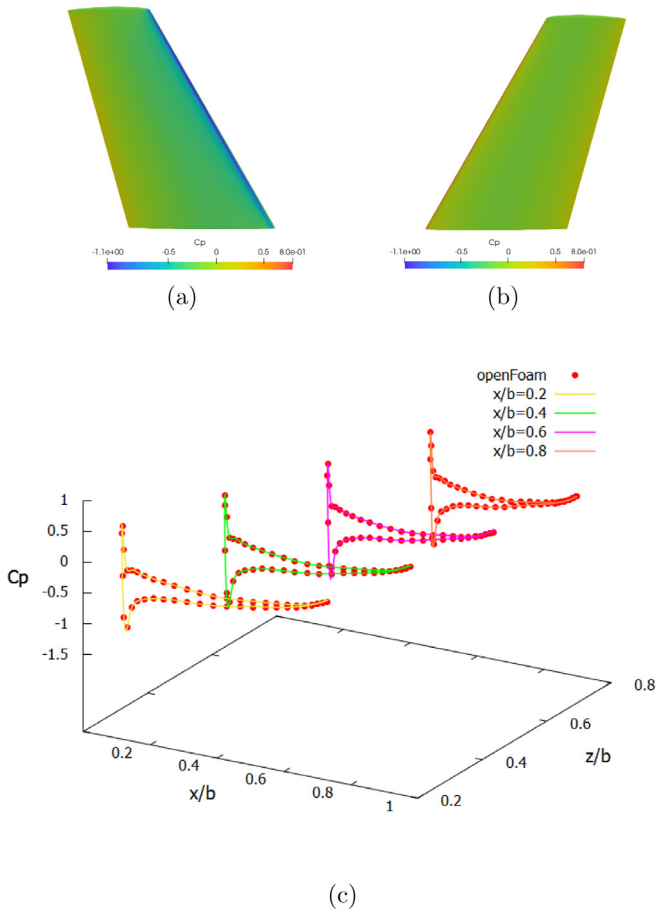

Fig. 5. Incompressible flow past the Onera-M6 wing. (a) Contours of pressure coefficient C_p on the suction side. (b) Contours of pressure coefficient C_p on the pressure side. (c) Pressure coefficient C_p at different span locations, where the continuous lines represent the solution obtained with FLUBIO, and the dots represent the solution obtained with OpenFOAM.

angle of attack is equal to 3.06°. For turbulence modeling, we use the RANS Spalart–Allmaras turbulence model. The discretization method used is similar to that described in Section 4.1.

In Table 4, we show a comparison of the outcome obtained with FLUBIO against OpenFOAM and Ansys Fluent [1]. In overall, the agreement is acceptable. In Fig. 5(a) and 5(b) we show the contours of pressure coefficient C_p on the wing surface. In Fig. 5(c), we compare the C_p distribution at four different locations of the wing, and as it can be seen, the agreement is good between FLUBIO and OpenFOAM.

Table 4
Comparison of the wing drag coefficient C_D and lift coefficient C_L. The force coefficients were computed using Eq. (3), where we used the wing projected area as reference surface.

Solver	C_D	C_L
FLUBIO	6.7×10^{-3}	8.65×10^{-2}
OpenFOAM	7.1×10^{-3}	8.47×10^{-2}
Ansys Fluent	6.6×10^{-3}	8.66×10^{-2}

5. Software impact and conclusions

FLUBIO is a CFD solver targeted at students, academics, personal users and practitioners to help them understand the general theory behind modern CFD solution methods and discretization techniques. The solver is general and capable enough so it can deal with simple problems involving the solution of the convection–diffusion equation, or the solution of complex academic and industrial problems involving turbulence modeling. The solver can be run in serial and parallel environments, and thanks to the use of the PETSc library, FLUBIO is suitable for large problems and potentially exascale computing.

FLUBIO is written using plain Fortran 2003 standard, with object orientation and code re-usability in mind, making it easier to understand and modify. The source code is commented extensively, so users can know the scope of the functions, classes, and derived types without the need of translating the source code into actionable information.

FLUBIO addresses many of the frustrations and difficulties found by the authors when using legacy solvers, and solvers lacking documentation. We aim with this contribution to address the need of many students and researchers to have a code easy to understand and to modify. A solver to use to test hypotheses but still able to deal with non-trivial geometries and complex flow physics.

We have made a great effort into translating the standard FVM and CFD theory found in the literature [5–9,30–36], into an easy to read computer code, with no tricks, which maintains a close link to the fundamentals of CFD and the FVM.

From a future perspective, plenty of exciting new developments are in the pipeline and we are happy to accept contribution from the community interested in using this new framework.

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

The authors kindly acknowledge the PETSc development team for the helpful support received through their mailing list. EA would like to thank the authors of Ref. [5] for their inspirational work and for having provided such a detailed reference on the FVM.

References

[1] Ansys Academic Research, Release 2020, Help System, Ansys Fluent Theory Guide. 2020.
[2] STAR-CCM+ 2020.1 Help Document. 2020.
[3] OpenFOAM. The Open Source CFD Toolbox. User Guide. URL http://www.openfoam.org.
[4] Economon TD, Palacios F, Copeland SR, Lukaczyk TW, Alonso JJ. SU2: An open-source suite for multiphysics simulation and design. AIAA J 2016;54(3):828–46. http://dx.doi.org/10.2514/1.J053813.
[5] Moukalled F, Mangani L, Darwish M. The finite volume method in computational fluid dynamics. 1st ed. Springer International Publishing; 2016, http://dx.doi.org/10.1007/978-3-319-16874-6.
[6] Ferziger JH, Peric M, Street RL. Computational methods for fluid dynamics. 4th ed. Springer International Publishing; 2020, http://dx.doi.org/10.1007/978-3-319-99693-6.
[7] Blazek J. Computational fluid dynamics. 3rd ed. Butterworth-Heinemann; 2015.
[8] Versteeg H, Malalasekera W. An introduction to computational fluid dynamics. The finite volume method. 2nd ed. Prentice Hall; 2007.
[9] Mazumder S. Numerical methods for partial differential equations. Finite difference and finite volume methods. 1st ed. Academic Press; 2016.
[10] Patankar SV, Spalding DB. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. Int J Heat Mass Transfer 1972;15(10):1787–806. http://dx.doi.org/10.1016/0017-9310(72)90054-3.
[11] Issa RI. Solution of the implicitly discretized fluid flow equations by operator-splitting. J Comput Phys 1985;62(1):40–65. http://dx.doi.org/10.1016/0021-9991(86)90099-9.
[12] Rhie CM, Chow WL. Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA J 1983;21(11):1525–32. http://dx.doi.org/10.2514/3.8284.
[13] 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: Proceedings, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary. 2004. p. 97–104.
[14] Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Dener A, Eijkhout V, Gropp WD, Karpeyev D, Kaushik D, Knepley MG, May DA, McInnes LC, Mills RT, Munson T, Rupp K, Sanan P, Smith BF, Zampini S, Zhang H, Zhang H. PETSc Users Manual. Tech. Rep. ANL-95/11 - Revision 3.13, Argonne National Laboratory; 2020, URL https://www.mcs.anl.gov/petsc.
[15] MacMackin C. FORD. June 2018. http://dx.doi.org/10.5281/zenodo.1422473.
[16] CGNS. CFD Genral Notation System. URL https://cgns.github.io/.
[17] HDF5. Hierarchical data format version 5. URL http://www.hdfgroup.org/HDF5.
[18] van Doormaal JP, Raithby GD. Enhancements of the SIMPLE method for predicting incompressible fluid flows. Int J Heat Mass Transfer 1984;7(2):147–63. http://dx.doi.org/10.1080/01495728408961817.
[19] Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. Rech Aerosp 1994;(1):5–21. http://dx.doi.org/10.2514/6.1992-439.
[20] Wilcox DC. Reassessment of the scale-determining equation for advanced turbulence models. AIAA J 1994;26(11):1299–310. http://dx.doi.org/10.2514/3.10041.
[21] Smagorinsky J. General circulation experiments with the primitive equations. Mon Weather Rev 1963;91:99–164. http://dx.doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.
[22] Nicoud F, Ducros F. Subgrid scale stress modelling based on the square of the velocity gradient tensor. Flow Turbul Combust 1999;62(3):183–200. http://dx.doi.org/10.1023/A:1009995426001.
[23] DFG flow around cylinder benchmark 2D-2, time-periodic case Re=100. URL http://www.featflow.de/en/benchmarks/cfdbenchmarking/flow/dfg_benchmark2_re100.html.
[24] Schäfer M, Turek S, Durst F, Krause E, Rannacher R. Benchmark computations of laminar flow around a cylinder. 1996, p. 547–66. http://dx.doi.org/10.1007/978-3-322-89849-4_39.
[25] Featflow. High performance finite elements. URL http://www.featflow.de/en/index.html.
[26] Schmitt V, Charpin F. Pressure distributions on the ONERA-M6-wing at transonic Mach numbers. Report AGARD Report 138, Brussels, Belgium: NATO; 1979.
[27] Guerrero J, Sanguineti M, Wittkowski K. CFD study of the impact of variable cant angle winglets on total drag reduction. Aerospace 2018;5. http://dx.doi.org/10.3390/aerospace5040126.
[28] Guerrero J, Sanguineti M, Wittkowski K. Variable cant angle winglets for improvement of aircraft flight performance. Meccanica 2020. http://dx.doi.org/10.1007/s11012-020-01230-1.
[29] NPARC Verification and Validation Web Site. URL https://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing01/m6wing01.html.
[30] Bernard P. Turbulent fluid flow. 1st ed. Wiley; 2019.
[31] Hirsch C. Numerical computation of internal and external flows. 2nd ed. Butterworth-Heinemann; 2007.
[32] Leveque R. Finite volume methods for hyperbolic problems. Cambridge texts in applied mathematics, 1st ed. 2002.
[33] Tucker P. Advanced computational fluid and aerodynamics. 1st ed. Cambridge University Press; 2016.
[34] Laney C. Computational gasdynamics. 1st ed. Cambridge University Press; 1998.
[35] Patankar S. Numerical heat transfer and fluid flow. 1st ed. CRC Press; 1980.
[36] Wilcox DC. Turbulence modeling for CFD. 3rd ed. DCW Industries; 2010.