FELiCS: A Versatile Linearized Flow Solver for Multi-Physics Applications

S.J. Knechtel [1], S. Demange [1], M. Goldack [1], L.M. Fuchs [1], M. Casel [1], X. Song [1], J.S. Müller [1], A. Villié [1], F. Bergner [1], M.T. Degner [1], A.R. Talasikar [1], A. Nehls [1], M. Matthaiou [1], P. Kuhn [1], K. Oberleithner [1], and T.L. Kaiser [1]

1 Laboratory for Flow Instabilities and Dynamics, Technische Universität Berlin, 10623 Berlin, Germany

DOI: 10.21105/joss.10241

Software

- Review
- Repository
- Archive

Editor: Mojtaba Barzegari

Reviewers:

- @AndreWeiner
- @JaroslavHron
- @vijaysm

Submitted: 02 December 2025
Published: 20 August 2026

Summary

The analysis and control of flow instability are crucial in many fields of engineering and science, ranging from aerodynamics to combustion. While computational fluid dynamics solvers can simulate the full nonlinear behavior of flows, the analysis of small perturbations is often more efficiently handled by linearized models. These low-order models provide valuable insights into flow stability, receptivity, and input-output interactions and facilitate flow control. Recently, those already well-known methods have been extended from the analysis of base flows and laminar flows to turbulent flows, by linearizing around the mean solution, which opens the possibility for a wide range of new applications. Consequently, open-source software tools for linear stability analysis represent an increasingly relevant asset for researchers and engineers studying complex flow dynamics.

`FELiCS` is both an easy-to-use, monolithic codebase executed via configuration files, as well as a flexible Python package designed for the linear stability and sensitivity analysis of complex fluid flows. It solves the linearized equations governing flow motion, turbulence transport, acoustics, and reacting flows using the Finite Element Method (FEM). Designed as a versatile tool, `FELiCS` enables global stability analysis, resolvent analysis, and input-output analysis (or "forcing-response-analysis") within a single framework. In addition, its modular functionalities can be used in the form of a Python package to incorporate linear equations into personal Python scripts, such as for base-flow computations, data assimilation or sensitivity calculations among others. Thus, it addresses the need for a user-friendly tool that combines these diverse analytical capabilities.

Statement of need

Mean flow analysis has emerged as a rapidly developing research field over the past decade and is beginning to attract growing interest from industry. As a result, computational tools that enable such analyses are becoming increasingly important for both the research community and practical engineering applications. In parallel, linearized flow analysis has evolved from its initial focus on relatively simple incompressible configurations to encompass complex multi-physics environments. These now include compressible and supersonic flows, turbulence modeling, and reacting flows, often occurring in arbitrary combinations.

There is a need for a well-documented and actively maintained research code that supports linear analysis in such multi-physics settings. Increasing application to realistic problems requires the ability to handle complex geometries through unstructured meshes and flexible discretizations. Usability is critical: researchers often possess mean flows from DNS, LES, or experiments and require linear results without programming expertise or detailed knowledge of numerical methods. Lowering this barrier is also important for industrial adoption, where tools must operate reliably without specialized insight into their implementation. The expanding use of linear models further demands software that can be extended easily to incorporate new equations and physical models. Because linearized operators are integral to tasks such as base-flow computation, data assimilation, and sensitivity analyses, the tool should enable straightforward access to these operators and support modular use within custom workflows. Accordingly, a suitable tool must (i) handle complex geometries, (ii) allow incremental extension to new physical models, and (iii) support both ready-to-use applications and flexible integration into research workflows.

To address these requirements, `FELiCS` (Kaiser, Demange, et al., 2023) provides a well-documented, unified framework for linearized multi-physics flow analysis on unstructured meshes. The software is implemented in Python and can be used either as a ready-to-run monolithic application or as a modular library, enabling flexible integration into research workflows. `FELiCS` supports a broad range of physical models, including incompressible and compressible Navier–Stokes equations, low-Mach-number density variations, and chemically reacting flows with progress variables, which can be combined as needed. The modeling capabilities include heat and mass transport, eddy-viscosity turbulence closures, and sponge layers for boundary damping, with planned extensions such as linearized Reynolds-averaged Navier–Stokes equations and additional reaction models. The monolithic interface enables common analyses—such as global modal analysis, resolvent analysis, and input–output analysis—without requiring prior Python expertise. At the same time, the library interface allows users to construct customized analyses and integrate linearized operators into external workflows, facilitating tasks such as data assimilation, sensitivity analysis, and gradient-based optimization. Embedded post-processing tools further support direct analysis of results within the finite-element framework.

Thus, `FELiCS` is designed to serve the needs of three different types of users:

- applied users can execute complete workflows via configuration files without programming knowledge;
- advanced users can import `FELiCS` as a library within their own Python codes, accessing equations and solvers in an object-oriented manner for complex tasks like data assimilation or gradient-based optimization;
- developers can extend the modular Python codebase to incorporate new physical models and methods.

State of the field

Several tools exist for stability analysis, which are specialized for specific flow regimes, flow equations and/or high-performance computing. For instance, NekStab (Frantz et al., 2023) offers efficient parallel computation for incompressible flows while using Nek5000's spectral elements discretization (Fischer et al., 2007), written in FORTRAN. Similarly, Nektar++ (Cantwell et al., 2015) offers eigenvalue computations with finite element discretization for various equations, while relying on complex high-order spectral/hp elements in C++. Other tools like Linstab2D (Martini & Schmidt, 2024) and BROADCAST (Poulain et al., 2023) (based on MATLAB and FORTRAN, respectively) discretize with finite differences, while using structured or quasi-structured meshes. The Python toolkit Resolvent4py (Padovan et al., 2025) provides matrix-free and high-performance parallelized linear analysis capabilities for large problems which requires the user to supply the already discretized linear operator. In addition to the aforementioned software, many researchers rely on in-house, unpublished stability codes tailored to specific applications.

Software design

The software offers an accessible interface, available both as a command-line tool and as a Python-importable module. Users control solver parameters via simple JSON configuration files and can import meshes from GMSH (Geuzaine & Remacle, 2009) while exporting results to H5 and XDMF formats for visualization in ParaView (Ahrens et al., 2005). Comprehensive user support, including tutorials, how-to guides, and detailed descriptions of the governing equations and models, is provided in the documentation.

Under the hood, the software ensures high performance and numerical stability by leveraging the PETSc/SLEPc toolkit (Balay et al., 2019; Hernandez et al., 2005) for linear algebra and the FEniCSx package (Baratta et al., 2023) for the FEM framework. The modular, object-oriented architecture further facilitates the extension of new physical models. All these functionalities can also be used by importing `FELiCS` as a Python package and do custom computations such as, for example, shape optimization or gradient based flow control, in relatively short scripts (see the how-tos documentation). The code ensures robustness through both minimalist test cases in a separate repository and automated unit tests.

`FELiCS` aims to support different combinations of momentum, energy, species, turbulence, and reacting-flow models, as well as different variable sets. It adopts a compositional approach, implemented in an object-oriented manner: individual partial differential equations (PDEs) and physical models are represented as modular components. Based on the user's selection, a corresponding closed set of linearized equations is first constructed and then discretized using standard FEM procedures.

Research impact statement

`FELiCS` supports a diverse range of applications demonstrated in numerous published studies. These include the analysis of combustion instabilities in turbulent jet flames and gas turbines (Casel et al., 2022; Kaiser, Oberleithner, et al., 2019; Kaiser, Lesshafft, et al., 2019; Kaiser, Varillon, et al., 2023; von Saldern et al., 2025; Wang et al., 2022) and aeroacoustic noise prediction for airfoils (Demange et al., 2024, 2025). It has also been applied to flow instabilities in gas (Moczarski et al., 2024) and water turbines (Müller et al., 2022), turbulent mixing (Kaiser & Oberleithner, 2021a, 2021b; Lückoff et al., 2021), coherent structure modeling in turbulent jets (Kuhn et al., 2021; von Saldern et al., 2024), and linear analysis based on PIV measurements (Fuchs et al., 2026; Klopsch et al., 2026). Further use cases encompass sensitivity and adjoint-based analyses (Knechtel et al., 2024; Müller et al., 2024) as well as shape optimization (Reumschüssel et al., 2026).

To illustrate the capabilities of the software, Figure 1 presents results from various multi-physics applications and displays incompressible resolvent modes in a cardiovascular flow (Figure 1(a)) (Villié et al., 2025), a compressible resolvent mode of a flow around an airfoil (Figure 1(b)) (Demange et al., 2025) and an incompressible resolvent mode in a turbulent separation bubble (Figure 1(c)) (Fuchs et al., 2026). Figure 1(d) shows a resolvent response mode of a turbulent methane-air flame (Kaiser, Varillon, et al., 2023).

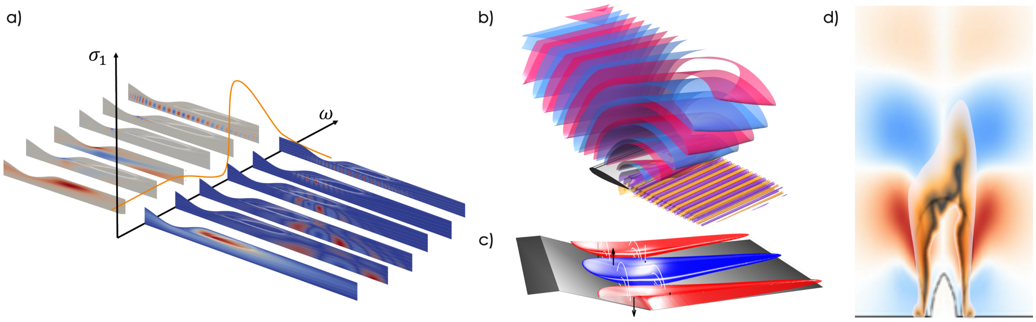

Figure 1: Visualization examples: a) Optimal forcing and response resolvent modes of a stenotic flow; b) Compressible resolvent response mode, two contour levels showing hydrodynamic and acoustic pressure fluctuations around an airfoil; c) Isocontour of the optimal resolvent mode in a turbulent separation bubble; d) Compressible resolvent mode with a flame model around a reacting jet.

Figure 2 presents post-processing analyses examples. Figure 2(a) demonstrates shape optimization applied to an unstable eigenvalue with respect to wall-normal shape deformations, while Figure 2(b) shows the structural sensitivity analysis in a 2D cylinder flow.

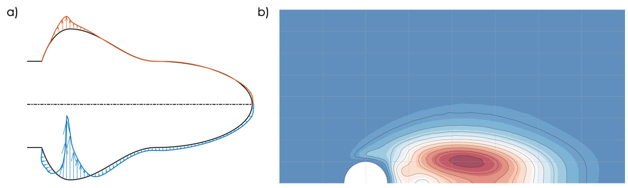

Figure 2: Post-processing examples: a) Growth rate sensitivity of the most unstable eigenvalue with respect to wall-normal shape deformations of a runner crown geometry in a turbulent Francis turbine flow; b) Structural sensitivity analysis in a 2D cylinder flow.

AI usage disclosure

Modern development environments integrate AI-assisted code completion tools (e.g., GitHub Copilot in VS Code). Such tools were used during the recent development of `FELiCS` to assist with code completion. All AI-suggested code fragments were systematically reviewed, tested, and validated by the developers prior to inclusion in the repository.

Large Language Model (LLM) tools were also used to assist in drafting parts of the software documentation, including selected docstrings for the API reference. For this purpose, a structured in-house prompting workflow was developed. All AI-assisted docstrings were generated in a dedicated documentation branch and underwent a strict internal review process prior to merging. Each module containing AI-assisted documentation was reviewed by a developer familiar with the corresponding code section to verify correctness, consistency, and technical accuracy.

LLM tools were additionally used to support grammar and spelling checks during the preparation of this manuscript. The content and technical descriptions were written by the authors. The entire manuscript was manually proofread multiple times to ensure that the meaning and scientific accuracy of the text were preserved.

AI tools were not used for algorithm design, numerical analysis, result generation, figure production, or validation of scientific conclusions.

Acknowledgements

The authors thank all current and past contributors to FELiCS. The FELiCS community is grateful to the developers of FEniCS, PETSc, and SLEPc for their foundational open-source infrastructure.

The authors acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through the following project numbers: 458062719 (S.D.), 504349109 (L.M.F.), 441269395, 523881008 (T.L.K.), 429772199 (J.S.M.). This work also received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101137955 (HORIZON-JTI-CLEANH2-2023-04-02), and from the State of Berlin via the Elsa-Neumann-Fellowship. The work was also supported by the German Federal Ministry for Economic Affairs and Energy (BMWi, grant 0324314B) and by computing time on SuperMUC-NG at Leibniz Supercomputing Centre, provided by the Gauss Centre for Supercomputing.

References

Ahrens, J., Geveci, B., & Law, C. (2005). ParaView: An end-user tool for large data visualization. In *Visualization handbook*. Elesvier. https://doi.org/10.1016/b978-012387582-2/50038-1

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W., & others. (2019). *PETSc users manual*. https://doi.org/10.2172/2998643

Baratta, I. A., Dean, J. P., Dokken, J. S., Habera, M., Hale, J. S., Richardson, C. N., Rognes, M. E., Scroggs, M. W., Sime, N., & Wells, G. N. (2023). *DOLFINx: The next generation FEniCS problem solving environment*. preprint. https://doi.org/10.5281/zenodo.10447666

Cantwell, C. D., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., De Grazia, D., Yakovlev, S., Lombard, J.-E., Ekelschot, D., & others. (2015). Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications*, *192*, 205–219. https://doi.org/10.1016/j.cpc.2015.02.008

Casel, M., Oberleithner, K., Zhang, F., Zirwes, T., Bockhorn, H., Trimis, D., & Kaiser, T. L. (2022). Resolvent-based modelling of coherent structures in a turbulent jet flame using a passive flame approach. *Combustion and Flame*, *236*, 111695. https://doi.org/10.1016/j.combustflame.2021.111695

Demange, S., Oberleithner, K., Yuan, Z., Hanifi, A., & Cavalieri, A. V. G. (2025). Flow Structures Driving Broadband Trailing-Edge Noise: A Resolvent-Based Model. *AIAA Journal*, 1–13. https://doi.org/10.2514/1.J065730

Demange, S., Yuan, Z., Jekosch, S., Hanifi, A., Cavalieri, A. V. G., Sarradj, E., Kaiser, T. L., & Oberleithner, K. (2024). Resolvent model for aeroacoustics of trailing edge noise. *Theoretical and Computational Fluid Dynamics*. https://doi.org/10.1007/s00162-024-00688-z

Fischer, P., Lottes, J., & Tufo, H. (2007). *Nek5000*. Argonne National Laboratory (ANL), Argonne, IL (United States). https://doi.org/10.11578/dc.20210416.29

Frantz, R. A., Loiseau, J.-C., & Robinet, J.-C. (2023). Krylov methods for large-scale dynamical systems: Application in fluid dynamics. *Applied Mechanics Reviews*, *75*(3), 030802. https://doi.org/10.1115/1.4056808

Fuchs, L. M., Steinfurth, B., von Saldern, J. G. R., Weiss, J., & Oberleithner, K. (2026). Standing-wave dynamics in low-frequency breathing of a turbulent separation bubble. *Journal of Fluid Mechanics*, *1030*, A34. https://doi.org/10.1017/jfm.2026.11191

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Hernandez, V., Roman, J. E., & Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, *31*(3), 351–362. https://doi.org/10.1145/1089014.1089019

Kaiser, T. L., Demange, S., Müller, J. S., Knechtel, S., & Oberleithner, K. (2023). FELiCS: A versatile linearized solver addressing dynamics in multi-physics flows. *AIAA Aviation 2023 Forum*, 3434. https://doi.org/10.2514/6.2023-3434

Kaiser, T. L., Lesshafft, L., & Oberleithner, K. (2019). Prediction of the flow response of a turbulent flame to acoustic perturbations based on mean flow resolvent analysis. *Journal of Engineering for Gas Turbines and Power*, *141*, 111021. https://doi.org/10.1115/1.4044993

Kaiser, T. L., & Oberleithner, K. (2021a). A global linearized framework for modelling shear dispersion and turbulent diffusion of passive scalar fluctuations. *Journal of Fluid Mechanics*, *915*, A111. https://doi.org/10.1017/jfm.2021.151

Kaiser, T. L., & Oberleithner, K. (2021b). Modeling the transport of fuel mixture perturbations and entropy waves in the linearized framework. *Journal of Engineering for Gas Turbines and Power*, *143*(11), 111001. https://doi.org/10.1115/1.4051714

Kaiser, T. L., Oberleithner, K., Selle, L., & Poinsot, T. (2019). Examining the effect of geometry changes in industrial fuel injection systems on hydrodynamic structures with biglobal linear stability analysis. *Journal of Engineering for Gas Turbines and Power*. https://doi.org/10.1115/1.4045018

Kaiser, T. L., Varillon, G., Polifke, W., Zhang, F., Zirwes, T., Bockhorn, H., & Oberleithner, K. (2023). Modelling the response of a turbulent jet flame to acoustic forcing in a linearized framework using an active flame approach. *Combustion and Flame*, *253*, 112778. https://doi.org/10.1016/j.combustflame.2023.112778

Klopsch, R., Fuchs, L. M., Rigas, G., Oberleithner, K., & von Saldern, J. G. (2026). Spectral analysis of attached and separated turbulent flows over a gaussian-shaped bump. *Journal of Fluid Mechanics*, *1036*, A67. https://doi.org/10.1017/jfm.2026.11629

Knechtel, S. J., Kaiser, T. L., Orchini, A., & Oberleithner, K. (2024). Arbitrary-order sensitivities of the incompressible base flow and its eigenproblem. *Journal of Fluid Mechanics*, *985*, A32. https://doi.org/10.1017/jfm.2024.195

Kuhn, P., Soria, J., & Oberleithner, K. (2021). Linear modelling of self-similar jet turbulence. *Journal of Fluid Mechanics*, *919*, A7. https://doi.org/10.1017/jfm.2021.292

Lückoff, F., Kaiser, T. L., Paschereit, C. O., & Oberleithner, K. (2021). Mean field coupling mechanisms explaining the impact of the precessing vortex core on the flame transfer function. *Combustion and Flame*, *223*, 254–266. https://doi.org/10.1016/j.combustflame.2020.09.019

Martini, E., & Schmidt, O. (2024). Linstab2D: Stability and resolvent analysis of compressible viscous flows in MATLAB. *Theoretical and Computational Fluid Dynamics*, *38*(5), 665–685. https://doi.org/10.1007/s00162-024-00706-0

Moczarski, L., Treleaven, N. C. W., Oberleithner, K., Schmidt, S., Fischer, A., & Kaiser, T. L. (2024). Interacting linear modes in the turbulent flow of an industrial swirled combustor. *AIAA Journal*, *62*, 979–988. https://doi.org/10.2514/1.J063639

Müller, J. S., Lückoff, F., Kaiser, T. L., & Oberleithner, K. (2022). On the relevance of the runner crown for flow instabilities in a francis turbine. *IOP Conference Series: Earth and Environmental Science*, *1079*, 12053. https://doi.org/10.1088/1755-1315/1079/1/012053

Müller, J. S., von Saldern, J. G. R., Kaiser, T. L., & Oberleithner, K. (2024). Linear amplification of inertial-wave-driven swirl fluctuations in turbulent swirling pipe flows: A resolvent analysis approach. *Journal of Fluid Mechanics*, *1000*, A91. https://doi.org/10.1017/jfm.2024.679

Padovan, A., Anantharaman, V., Rowley, C. W., Vollmer, B., Colonius, T., & Bodony, D. J. (2025). Resolvent4py: A parallel python package for analysis, model reduction and control of large-scale linear systems. *SoftwareX*, *31*, 102286. https://doi.org/10.1016/j.softx.2025.102286

Poulain, A., Content, C., Sipp, D., Rigas, G., & Garnier, E. (2023). BROADCAST: A high-order compressible CFD toolbox for stability and sensitivity using algorithmic differentiation. *Computer Physics Communications*, *283*, 108557. https://doi.org/10.1016/j.cpc.2022.108557

Reumschüssel, M., Müller, J., Demange, S., Knechtel, S., & Oberleithner, K. (2026). Shape optimization for control of flow unsteadiness via linearized methods and higher-order bayesian optimization. *Social Science Research Network Preprint*. https://doi.org/10.2139/ssrn.6065726

Villié, A., Schmitter, S., von Saldern, J. G., Demange, S., & Oberleithner, K. (2025). Physics-informed neural networks for enhancing medical flow magnetic resonance imaging: Artifact correction and mean pressure and reynolds stresses assimilation. *Physics of Fluids*, *37*(2). https://doi.org/10.1063/5.0252852

von Saldern, J. G., Beuth, J. P., Reumschüssel, J. M., Jaeschke, A., Paschereit, C. O., & Oberleithner, K. (2025). Low-frequency streaky structures in turbulent hydrogen jet flames. *Combustion and Flame*, *278*, 114231. https://doi.org/10.1016/j.combustflame.2025.114231

von Saldern, J. G., Schmidt, O. T., Jordan, P., & Oberleithner, K. (2024). On the role of eddy viscosity in resolvent analysis of turbulent jets. *Journal of Fluid Mechanics*, *1000*, A51. https://doi.org/10.1017/jfm.2024.922

Wang, C., Kaiser, T. L., Meindl, M., Oberleithner, K., Polifke, W., & Lesshafft, L. (2022). Linear instability of a premixed slot flame: Flame transfer function and resolvent analysis. *Combustion and Flame*, *240*, 112016. https://doi.org/10.1016/j.combustflame.2022.112016