The Journal of Open Source Software

Opyrability: A Python package for process operability analysis

Victor Alves[1], San Dinh[1,2], John R. Kitchin[2], Vitor Gazzaneo[1], Juan C. Carrasco[3], and Fernando V. Lima[1]

1 Department of Chemical and Biomedical Engineering, West Virginia University, Morgantown, West Virginia, USA **2** Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA **3** Department of Chemical Engineering, Universidad de Concepción, Concepción, Chile

DOI: 10.21105/joss.05966

Software

- Review
- Repository
- Archive

Editor: Kyle Niemeyer

Reviewers:

- @gmxavier
- @mustafaalsalmi1999

Submitted: 12 September 2023
Published: 06 February 2024

Summary

When designing a chemical process/plant, two main tasks naturally arise when considering the processing of raw materials into value-added products such as chemicals or energy:

1. **Process design decisions**: Which decisions should be made with respect to the design variables of a given process, in a way that its overall objectives are achieved? (e.g., economic profitability, constraints related to product purity/pollutant emissions, sustainability, etc.).
2. **Process control objectives**: Which variables should be controlled, yielding the maximum operability of the process? That is, can the process reach its maximum operational capacity, given the ranges of the manipulated/input variables when subject to disturbances?

Historically, Tasks 1 and 2 have been performed sequentially: Engineers/practitioners would come up with the design decisions, and only then the control objectives would be assessed. Unfortunately, this can yield a process that is designed in a way that its operability capabilities are hindered. In other words, because the control objectives were not considered early in the design phase, the process itself might be not controllable or operable at all. To give some perspective on how challenging this problem can be, there are reports dating back to the 1940s from well-known authors in the process control field such as Ziegler and Nichols (Ziegler & Nichols, 1943) mentioning the importance of interconnecting design and control.

Considering this, the need of quantifying achievability for a general nonlinear process naturally arises. The underlying motivation of determining whether it would be possible to measure the operability of a process to simultaneously achieve process design and control objectives led Georgakis and coworkers (Georgakis et al., 2003; Lima & Georgakis, 2010; Subramanian & Georgakis, 2005; Vinson & Georgakis, 2000) to formally define *process operability* and a metric called the *Operability Index (OI)*. The OI was conceptualized as a measure to quantify achievability of nonlinear processes (Vinson & Georgakis, 2000), which was proven to be independent of the control strategy and inventory control layer (Vinson & Georgakis, 2002). In addition, it allows for the efficient ranking of competing designs and/or control structures (Lima et al., 2010) and enables the systematic assessment of operability characteristics under the presence of disturbances. Hence, process operability was formalized as *a systematic framework to simultaneously assess design and control objectives early in the conceptual phase of industrial, typically large-scale, and nonlinear chemical processes.*

To achieve the systematic assessment of design and control objectives simultaneously, process operability is based on the definition of *operability sets*. These are spaces in the cartesian

system that are defined with respect to the available inputs of a given process, their respective achievable outputs, the desired regions of operation in the input and output spaces, and lastly, any expected disturbances that may be present. The thorough definitions of these spaces are readily available in the literature (Gazzaneo et al., 2020), as well as in the opyrability documentation.

Therefore, `opyrability` is a Python package for process operability analysis and calculations, with its API designed to provide a user-friendly interface to enable users to perform process operability analysis seamlessly. This has the aim of reducing the complexity of dealing with the programming aspects of nonlinear programming (Carrasco & Lima, 2017) and computational geometry (Gazzaneo & Lima, 2019) operations needed when performing process operability analyses.

Statement of need

`Opyrability` corresponds to a unified software tool to perform process operability analysis in a single-bundle fashion. In broader terms, opyrability provides a formal and mathematically tractable framework to systematically investigate the operability and achievability of industrial processes earlier in the conceptual phase. This eliminates the need for resorting to *ad-hoc*-type solutions to the design and control of industrial processes, which are inherently with loss of generality. The use of this framework thus guarantees a solution to the operability problem that is generalizable to any process, as long as a mathematical model of the given application is available. Hence, the introduction of `opyrability` in Python, a widely used and freely available programming language, is a significant advancement in the process operability field. Being open-source and hosted in a community-driven environment, it offers a valuable resource to the process systems engineering, computational catalysis and material sciences communities that would benefit from operability direct/inverse mappings. This package empowers researchers and practitioners to easily investigate the operability aspects of both emerging and existing large-scale industrial processes. Additionally, on a lab scale, it can aid in the examination of material properties that guide design decisions, such as reactions rate and membrane parameters that would be needed to reach certain product specifications.

Moreover, `opyrability` is built on well-known and developed packages such as (i) numpy (Harris et al., 2020) and (ii) scipy (Virtanen et al., 2020) for linear algebra and scientific computing; (iii) matplotlib (Hunter, 2007) for visualizing the operable regions in 2D/3D; (iv) cvxopt that allows access to glpk for linear programming; (v) polytope that enables efficient polytopic calculations; and (vi) cyipopt that allows access to IPOPT (Wächter & Biegler, 2006), a state-of-the-art nonlinear programming solver, enabling efficient inverse mapping operations within the operability framework. The inverse mapping task is further extended with full support for automatic differentiation, powered by JAX (Bradbury et al., 2018). This effort thus facilitates the dissemination of operability concepts and calculations in process systems engineering and other fields. Figure 1 illustrates the dependency graph for `opyrability`.

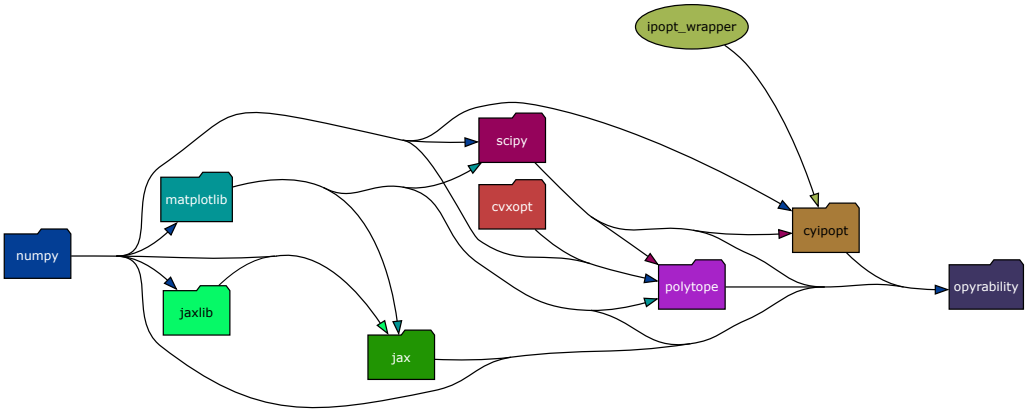

Figure 1: Dependency graph generated with pydeps illustrating all numerical packages and visualization tools used in `opyrability`.

Vignette

As a quick illustration of `opyrability's` capabilities, the example below available in the examples gallery of the proposed tool depicts the operability analysis of a continuous stirred tank reactor. In this example, the OI is evaluated for a desired region of operation for the concentration of reactants A and B (as outputs). In particular, it is desired to obtain insight on the design and operating region of this process, in terms of the reactor radius and its operating temperature (as inputs). Process operability is employed to systematically analyze which designs and operating temperatures are able to attain the requirements related to the concentrations of A and B.

The fundamental idea of `opyrability` is to be an environment for engineers and scientists that eliminates the burden of dealing with the transitioning among different software packages and environments to perform process operability calculations. In addition, the knowledge about computational geometry and constrained nonlinear programming can be limited to only theoretical rather than having the users implement the operability algorithms since `opyrability` already encapsulates all the necessary calculations.

In the example below, the user only needs to: (i) have simple Python programming knowledge, limited to be able to perform mathematical modeling and manipulate numpy arrays; and (ii) be able to interact with `opyrability's` functions, namely `multimodel_rep`, `OI_eval` and `nlp_based_approach`, as shown in the API documentation. Figure 2 illustrates the process and example in focus.

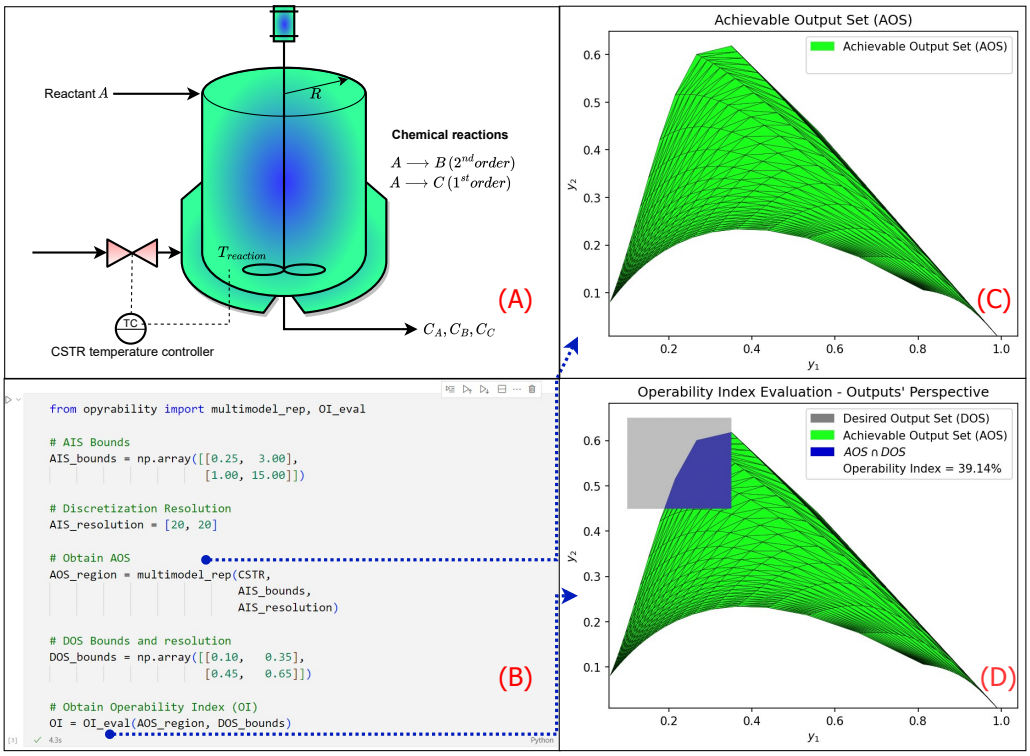

Figure 2: `Opyrability` multimodel representation. (A) Chemical reactor schematic. (B) Jupyter notebook illustrating the use of the `multimodel_rep` and `OI_eval` functions, as well as the set-up of these. (C) Visualization of the Achievable Output Set for the continuous stirred tank reactor example including the operable boundaries of the process studied. (D) Quantification of the Operability Index (OI), in which `opyrability` calculates that 39.14% of the desired operation can be achieved.

Lastly, Figure 3 depicts the use of `opyrability's` inverse mapping features by using the `nlp_based_approach` function, allowing the user to obtain from a desired region in the output space, the region in the input space that guarantees the desired operation.

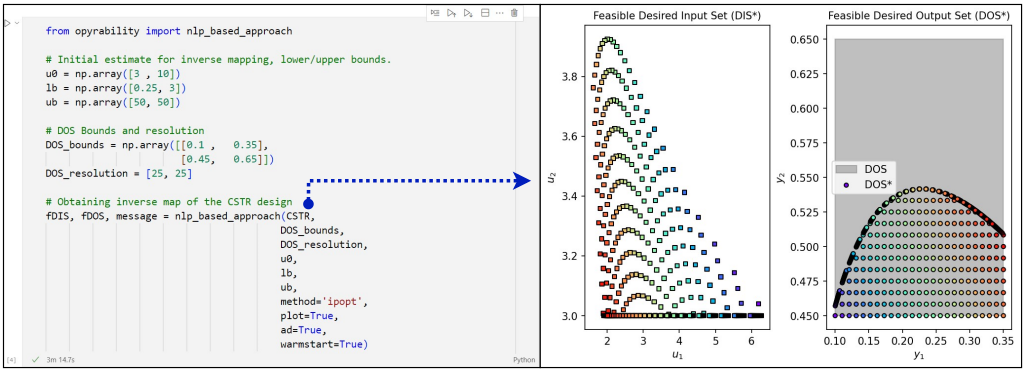

Figure 3: `opyrability's` inverse mapping using `nlp_based_approach`, in which the input space that guarantees the desired output set region is attained.

Availability

`Opyrability` is freely available in both PyPI and conda stores, as well as have its source code hosted on GitHub. In addition, its documentation contains not only a thorough description

of the API, but also a theoretical background discussion on process operability concepts, an examples gallery, and instructions on how to set up a process model following `opyrability` design principles. The idea is to supply both proper documentation to the users in the open-source software community as well as to give the users the necessary amount of theory allowing them to employ process operability principles in their specific application.

Acknowledgements

The authors acknowledge the support from the National Science Foundation CAREER Award 1653098.

References

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax

Carrasco, J. C., & Lima, F. V. (2017). Novel operability-based approach for process design and intensification: Application to a membrane reactor for direct methane aromatization. *AIChE Journal*, *63*(3), 975–983. https://doi.org/10.1002/aic.15439

Gazzaneo, V., Carrasco, J. C., Vinson, D. R., & Lima, F. V. (2020). Process operability algorithms: Past, present, and future developments. *Industrial & Engineering Chemistry Research*, *59*(6), 2457–2470. https://doi.org/10.1021/acs.iecr.9b05181

Gazzaneo, V., & Lima, F. V. (2019). Multilayer operability framework for process design, intensification, and modularization of nonlinear energy systems. *Industrial & Engineering Chemistry Research*, *58*(15), 6069–6079. https://doi.org/10.1021/acs.iecr.8b05482

Georgakis, C., Uztürk, D., Subramanian, S., & Vinson, D. R. (2003). On the operability of continuous processes. *Control Engineering Practice*, *11*(8), 859–869. https://doi.org/10.1016/S0967-0661(02)00217-4

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Lima, F. V., & Georgakis, C. (2010). Input-output operability of control systems: The steady-state case. *Journal of Process Control*, *20*(6), 769–776. https://doi.org/10.1016/j.jprocont.2010.04.008

Lima, F. V., Jia, Z., Ierapetritou, M., & Georgakis, C. (2010). Similarities and differences between the concepts of operability and flexibility: The steady-state case. *AIChE Journal*, *56*(3), 702–716. https://doi.org/10.1002/aic.12021

Subramanian, S., & Georgakis, C. (2005). Methodology for the steady-state operability analysis of plantwide systems. *Industrial & Engineering Chemistry Research*, *44*(20), 7770–7786. https://doi.org/10.1021/ie0490076

Vinson, D. R., & Georgakis, C. (2000). A new measure of process output controllability. *Journal of Process Control*, *10*(2), 185–194. https://doi.org/10.1016/S0959-1524(99)00045-1

Vinson, D. R., & Georgakis, C. (2002). Inventory control structure independence of the process operability index. *Industrial & Engineering Chemistry Research*, *41*(16), 3970–3983. https://doi.org/10.1021/ie0109814

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

Ziegler, J., & Nichols, N. (1943). Process lags in automatic-control circuits. *Transactions of the American Society of Mechanical Engineers*, *65*(5), 433–440. https://doi.org/10.1115/1.4018788