OPEN-SOURCE MODELING PLATFORMS

John Hedengren [a,1], Bethany Nicholson [b]

[a] Department of Chemical Engineering, Brigham Young University, Provo, UT 84602

[b] Center for Computing Research, Sandia National Laboratories, Albuquerque, NM 87123

Abstract

A review of current trends in scientific computing reveals a broad shift to open-source and higher-level programming languages such as Python and growing career opportunities over the next decade. Open-source modeling tools accelerate innovation in equation-based and data-driven applications. Significant resources have been deployed to develop data-driven tools (PyTorch, TensorFlow, Scikit-learn) from tech companies that rely on machine learning services to accelerate business needs. The data and applications of the software are proprietary but the foundational tools are open. Open-source equation-based tools such as Pyomo, CasADi, Gekko, and JuMP are also gaining momentum according to user community and development pace metrics. The future of open-source modeling tools is in specialization and interfaces to other specialized packages. Integration of data-driven (empirical) and equation-based (principles, knowledge-driven) tools is emerging. New compute hardware, productivity software, and training resources have the potential to radically accelerate progress. However, long-term support mechanisms are still needed to sustain momentum and maintenance for key foundational packages.

Keywords

Modeling, Open-source, Optimization, Simulation, Solver.

Introduction

The pace of innovation in scientific computing is accelerated with free and open-source foundational contributions such as programming languages, modeling platforms, and solvers. The decision to create and support open-source packages is counter-intuitive from the aspect of direct compensation for the time and effort put into creating and supporting the software. While there are non-monetary awards and recognition for creating useful software, there are many business, regulatory, and scientific drivers that influence the decision to release open-source software. Open-source is sometimes required by the sponsoring agency, such as a government contract that requires the source code. Business drivers for open-source include spreading the development burden across the industry instead of isolating it to a specialized team of developers within a single company. Scientific drivers for open-source include verifying results and advancing science with the ability to more easily build on and extend existing work. The value of open-source software is amplified by a strong community of users and developers that mutually support each other through online tutorials, support forums, bug reports, feature requests, and documentation. Community momentum is a critical metric to observe so that organizations can build upon open-source software that is actively developed and supported and find skilled workers already familiar with the software, limiting the need for extensive training.

The organization of this paper is to first present a high-level view of current trends in scientific computing. In particular, there has been a shift from proprietary software to open-source programming languages (MATLAB to Python). There has also been a performance sacrifice for increased usability, functionality, and higher-level abstractions (C++ to Python). Next, this paper compares momentum for equation-based modeling platforms and data-driven modeling platforms and discusses the pace of innovation and how this can be accelerated with open-source initiatives. Finally, the future of open-source software is considered focusing on two areas: (1) current developments and features that are recently released or planned to be released in the next few years and (2) long-term needs for open-source software development within Process Systems Engineering (PSE).

Current Trends in Scientific Computing

Programming jobs in software development, quality assurance, analysis, and testing will grow +22%, about 3 times

[1] Corresponding author. Email: john_hedengren@byu.edu.

faster than other occupations, over the next decade according to US Bureau of Labor Statistics (2022). Python is the most popular programming language according to indices that track online searches (PYPL, 2022). Other scientific computing languages in the top 50 most popular programming languages include C/C++ (2/4), R (16), MATLAB (24), FORTRAN (26), Julia (28), Simulink (47), and LabVIEW (48) as of June 2022 (TIOBE, 2022).

Python has gained popularity relative to other scientific programming languages in recent years as shown in Fig. 1. Python has risen in popularity because of its accessibility, ease of learning, documentation, online community support, and library availability but is criticized for its performance relative to compiled languages like C/C++. Many popular Python packages for scientific computing interface to lower-lever programming languages to offload compute-intensive tasks. In addition, JIT (Just In Time) compilers such as Numba and PyPy or AOT (Ahead of Time) compilers such as Cython can be used to speed up Python code.

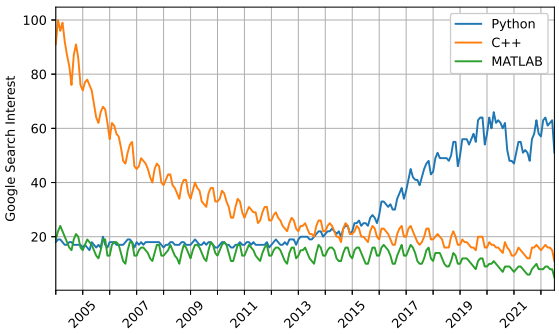

Figure 1: Trends of Python, C++, and MATLAB search interest from January 2004 to June 2022

In contrast, Julia is a much younger programming language that is starting to gain momentum in the scientific computing community. It offers many of the same features of Python in terms of usability with the added benefit of computational performance comparable to lower-level compiled languages. However, as it is a newer language, there is a limited set of libraries available in Julia.

Open-Source Momentum Metrics and Definitions

Open-source modeling packages gain momentum by having an active development team and by growing a user community. The momentum of open-source modeling packages can be compared by examining the number of users and developers actively engaged with the software. Some metrics for measuring software engagement include:

- Users: Install Rate, Q+A Forum Posts, Citations
- Developers: Latest Release, Documentation, OS Support, GitHub Insights

Other factors are also important such as whether the software is easy to install, extensible, scales to large-scale problems, solves popular benchmark problems, is tailored to unique solutions not available elsewhere, and has auto-completion in advanced tools such as GitHub Copilot.

There is an important distinction between Open-Source Software (OSS), Free Software (FS), and Free and Open-Source Software (FOSS). OSS can have a proprietary license and FS can be closed-source. Free/Libre and Open-Source Software (FLOSS) emphasizes that free software refers to freedom and not to price. The focus of this review is on FLOSS modeling frameworks with permissive licenses (allowing for the use, copy, and modification of the source code) that are openly shared to encourage developers to voluntarily adapt and improve the software. FLOSS is in contrast to proprietary codes that have restrictive licenses or unavailable source code. Proprietary software has an important role to provide customer support, graphical user interfaces, and customized solutions. Some industries are dominated by FLOSS such as Python in data science and TensorFlow / PyTorch in deep learning. Other segments of scientific computing are dominated by proprietary software, such as solvers CPLEX and Gurobi for Mixed Integer Linear Programming (MILP) and Simulink for graphical and embedded control, that have less competitive but emerging open-source alternatives. Distributed Control Systems (DCS) and Programmable Logic Controllers (PLC) are likewise dominated by proprietary solutions. Open Process Automation (OPA) is an industry-led initiative to create interoperability standards in the industrial control domain (Bartusiak et al., 2022). The trend in many industries is the adoption of standards or open-source alternatives.

While many open-source packages are initially developed in academia, there are several FLOSS modeling platforms that have been created and supported by industry. The term "mind share" is frequently cited as a reason to release commercial software as FLOSS and distribute development costs among industry participants. The software becomes more useful with broad adoption, an online support community, searchable knowledge base, and extensions of the software capabilities. Some of the challenges of FLOSS are lack of standards for benchmark performance, shifting community momentum, long-term support, and selection among many alternatives.

Open-Source Algebraic Modeling Languages

The primary purpose of an Algebraic Modeling Language (AML) is to facilitate the expression and solution of equation-based optimization problems. The AML serves as a front-end translator for mathematical expressions, converting these expressions into a form for solvers to attempt a solution. At a minimum, solvers need information such as objective function values and equation residuals. Many solvers also require first and second derivative information so most AMLs also provide automatic differentiation capabilities.

This section gives an overview of equation-based FLOSS modeling platforms, with particular emphasis on control and optimization AMLs. The monthly download rates for three popular FLOSS Python AML packages is shown in Figure 2. These numbers are inflated with downloads from automated clone repositories but give a general picture of the growth in

users over time. The list of packages described below is not comprehensive, but is an attempt to share some of the popular options with their distinguishing capabilities.

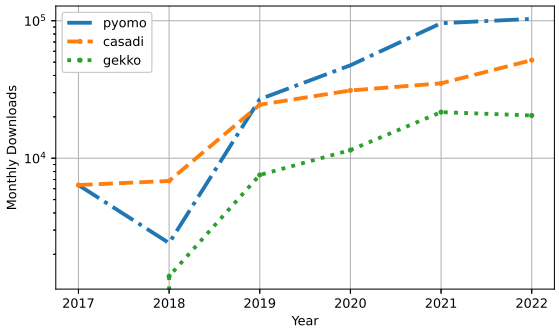

Figure 2: AML downloads for one month (June) each year (source: Google BigQuery).

Pyomo (Bynum et al., 2021) is a Python-based AML. It includes interfaces to a variety of optimization solvers either through standardized file formats (LP or NL) or by interfacing directly with a solver's Python API. Automatic differentiation is achieved using the AMPL Solver Library (ASL) with NL files. An advantage of this package is that it includes many extensions for handling high-level modeling constructs (e.g. differential equations and logical disjunctions). Pyomo was first released in 2009 as the Coopr software library but was released as its own package beginning in 2015. As of July 2022 there are 1270 Pyomo tagged questions on Stack Overflow. User questions are also posted to a Google Group forum.

CasADi (Andersson et al., 2019) is available in MATLAB, Python, and C++. It was originally a framework for automatic differentiation but has evolved into a complete modeling language. Casadi was first released in 2017 and as of July 2022, there are 72 questions on Stack Overflow. Most of the support questions are posted to a Google Group forum that delivers messages with email.

Gekko (Beal et al., 2018) is a Python package for machine learning and optimization of mixed-integer and differential algebraic equations. It is coupled with large-scale solvers for optimization, parameter regression, and predictive control. Gekko was first released in 2018 and as of July 2022, there are 605 questions on Stack Overflow. Questions are also posted to a Google Group forum.

JuMP (Dunning et al., 2017a) is a Julia-based modeling language for optimization with automatic differentiation for solution of linear, nonlinear, and mixed-integer problems with many solver interfaces. As of July 2022, there are 313 questions on Stack Overflow and about 800 questions on the Julia Language support forum. A direct comparison to pip installs is not possible because many of the pip downloads are for cloning. A total of 93,424 unique IP addresses downloaded JuMP between Sept 2021 and July 2022 for a monthly download rate of ∼8500. Besides anonymized or dynamic IP addresses, this is a much closer count to number of users than the pip install numbers that are inflated with clone repository downloads.

In addition to those listed above, many other FLOSS and proprietary software packages are available for optimization and control including ACADO (Houska et al., 2011), ACADOS (Verschueren et al., 2022), AIMMS (Bisschop, 2006), AMPL (Fourer et al., 1993), CProS (Misra et al., 2022), CVX (Grant and Boyd, 2008), CVXOPT (Andersen et al., 2011), DIDO (Ross, 2004), Dymos (Falck et al., 2021), GAMS (Bisschop and Meeraus, 1982), GPkit (Burnell et al., 2020), GPOPS II (Patterson and Rao, 2014), Gravity (Hijazi et al., 2018), IMPL (Kelly and Menezes, 2019), InfiniteOpt (Pulsipher et al., 2022), MUSCOD-II (Leineweber et al., 2003), NLPy (Orban, 2014), OMPR (Schumacher, 2022), OpenMDAO (Gray et al., 2019), OpenOpt (Kroshko, 2007), OPTANO (DEMİR, DEMİR), OR-tools (Perron, 2011), PICOS (Sagnol and Stahlberg, 2022), PROPT (Rutquist and Edvall, 2010), PSOPT (Becerra, 2010), PuLP (Mitchell et al., 2011), PyOpt (Perez et al., 2012), PySCIPOpt (Maher et al., 2016), Python-MIP (Santos and Toffolo, 2019), and YALMIP (Löfberg, 2004).

Many of the FLOSS optimization platforms mentioned above are also enabling new, application-specific packages. For example, do-mpc (Lucia et al., 2017) and PolyMPC (Listov and Jones, 2020) are libraries for Model Predictive Control (MPC) built on CasADi and IDAES (Lee et al., 2021) is a Pyomo-based multi-scale process modeling framework for the design and optimization of complex, interacting energy and process systems. There are many more examples of application-specific packages, further illustrating the value these generic FLOSS optimization platforms provide to the community.

Review of FLOSS optimization solvers is beyond the scope of this review. However, a notable emerging trend is tighter coupling between the solver and modeling language for callbacks, adaptive programming, and meta-algorithm development.

Open-Source Data-Driven Modeling

The dramatic rise of data-driven modeling is driven by increased data availability, decreased compute cost, and powerful data-driven software tools. Two of the most popular packages for deep learning are TensorFlow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019) that were developed at Alphabet (Google Brain) and Meta AI (Facebook), respectively. The machine learning package scikit-learn (Pedregosa et al., 2011) was started in 2007 as a Google Summer of Code project. Figure 3 shows the number of monthly downloads of scikit-learn, TensorFlow, and PyTorch. The download rate is not an accurate count of users but does give qualitative trends on relative adoption rates and community momentum.

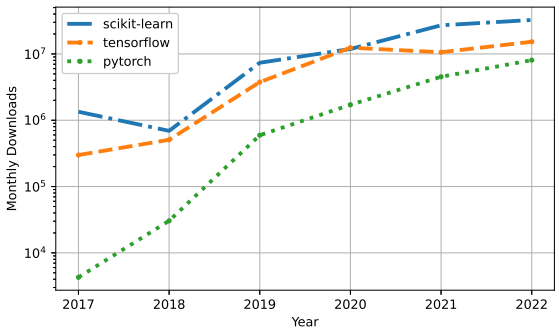

Figure 3: Data-driven package downloads for one month (June) each year (source: Google BigQuery).

Part of the core business model of both Meta and Alphabet is to sell advertisements and online services rather than proprietary optimization software. Forecasting, natural language processing, facial recognition, advertisement selection, and web-page ranking are Artificial Intelligence (AI) enabled aspects for increasing click-through rates. The decision to open-source and support AI tools is one of the factors that has increased speed of innovation and impact within major US tech companies such as Alphabet, Meta, Amazon, Tesla, Apple, and Netflix. There is an abundance of AI startup companies now penetrating traditional industries (manufacturing, automotive, aerospace, etc) with data engineering, data science, and machine learning services with Industry 4.0 innovation and disruption. Many of the software solutions are based on open-source tools such as Tesla Autopilot built on PyTorch.

A key performance metric for data-driven models is the performance per energy consumed. Specialized compute hardware has been created to reduce the power consumption such as Application Specific Integrated Circuits (ASICS) for processing financial transactions on blockchain, embedded controls, smart phones, wearable devices, and other applications in automotive, telecommunication, and medical industries. ASICS are designed for a specific task while a Central Processing Unit (CPU) is a more configurable platform for computing. Alphabet designed the Tensor Processing Unit (TPU) to reduce power consumption by up to 80 times relative to contemporary CPUs or GPUs (Eeckhout, 2017). They achieved this improvement by using 8-bit integers and a complex instruction set to calculate a neural network prediction. Instead of relying on the pace of innovation that is driven by other industries, Alphabet created new hardware to drive a key performance index for data-driven models that are used in search, street view, photos, and translation. The CPU, GPU, and TPU kernels are freely available in web browsers through a Google Colab run-time option for machine learning prediction functions.

Impacts on Speed of Innovation

FLOSS tools have greatly impacted the rate of innovation in the optimization and control community. Equation-based and data-driven modeling and optimization software are two concrete examples of tools that accelerate innovation. Equation-based tools have been applied in chemical and process industries (Mowbray et al., 2022), staff scheduling (Abouee Mehrizi et al., 2022), mathematical research (Hernandez et al., 2022), renewable energy grid optimization (Lai et al., 2022), control of electric vehicle charging in smart communities (Zhou et al., 2022), chemical reactor design (Frumkin et al., 2022), blockchain computing optimization for Industrial Internet of Things (IoT) (Wang et al., 2022), safety systems on Liquified Natural Gas (LNG) vessels (Nubli et al., 2022), robotic hand automation (Hammoud et al., 2021), autonomous unmanned aerial vehicles (Alhelaly et al., 2022; Han et al., 2022), low-activity waste loading for long-term storage with vitrification (Lu et al., 2021), and fish-like robots (Barbosa et al., 2021). Many other applications are cited, giving strong evidence of user adoption with innovative application areas. There are 1074 citations of Pyomo (Hart et al., 2012), 353 citations of APMonitor and Gekko (Beal et al., 2018), 1478 citations of CasADi (Andersson et al., 2019), and 1195 citations of JuMP (Dunning et al., 2017b).

The pace of innovation is likewise supported by FLOSS tools in data-driven modeling and optimization with notable advances in natural language processing (Brown et al., 2020), self-driving cars (Gupta et al., 2021), image classification (Guo et al., 2016), medical diagnosis (Razzak et al., 2018), precision agriculture (Kamilaris and Prenafeta-Boldú, 2018), autonomous unmanned aerial vehicles (Fraga-Lamas et al., 2019), and many other areas (LeCun et al., 2015). There are 26,958 citations of TensorFlow (Abadi et al., 2016, 2015), 27,720 citations of PyTorch (Paszke et al., 2019, 2017), and 58,656 citations of Scikit-learn (Pedregosa et al., 2011).

Future of Open-Source Tools

Most of the tools discussed in this paper were recently developed around the time of the last CPC/FOCAPO meeting in 2017. Since then many of these tools have seen significant adoption by the PSE community. This section looks to the future and tries to predict how these tools will evolve over the next 5-10 years.

What's Next

A new trend in open-source tools is to specialize to an important task and create interfaces to other packages that complement those capabilities. There is TensorFlow support in CasADi, PyTorch linear and integer programming with Pyomo (Tang and Khalil, 2022), integration of machine learning models in Pyomo (Ceccon et al., 2022), constrained optimization with physics-based modeling priors in PyTorch (Tuor et al., 2022), and Gekko interfaces to GPflow (Matthews et al., 2017) and scikit-learn. Developments with package interoperability will continue to accelerate in the next 5 years.

There are new development resources for code autocompletion such as GitHub Copilot (Sobania et al., 2022) which could accelerate the adoption of certain FLOSS modeling tools. Additional AI-trained tools and auto-ML tools will move optimization engineers, data scientists, and machine learning specialists to new levels of abstraction with

higher levels of productivity (He et al., 2021).

Data engineering, organizing and preparing data for the purpose of extracting useful information (Wu et al., 2013), will also be increasingly important.

What's Needed

A well-known issue for open-source foundational tools is long-term support and maintenance. This was recently emphasized within the PSE community in a December 2021 COIN-OR news post seeking support for a full-time employee to work on the development, documentation, and distribution of COIN-OR projects such as Clp, Cbc, and Ipopt. Without this support, the COIN-OR initiative may be retired. Support is also needed for adaptation to new computing platforms (quantum, cloud, edge, embedded), new interfaces such as higher level abstractions to define optimization problems, and support for issue tracking and resolution. Without financial incentives, better recognition of code and software contributions could be another method of motivating development and support of open-source tools. A long-term support strategy for FLOSS tools will become increasingly important as these tools see broader adoption in the optimization community, especially for packages that require a high level of skill to develop and maintain.

The difference in development pace and resources is apparent with data-driven and equation-based software. The open-source model accelerates user feedback and spreads the cost of development to the broader community. For example, many companies reduced in-house technical expertise in favor of contracting out development services in the energy, power, and chemical industries. The companies rely on proprietary packages that sometimes have not had significant core technological advances from the first deployments in the 1980s-90s. How would the situation be different if key companies, similar to Meta and Alphabet, had released open-source tools for broad adoption? Equation-based modeling tools benefit from academic development and government funding, but have not had a similar accelerated pace as data-driven methods with strong initial open-source support. The pace of innovation is robust but lags data-driven tools that design specialized software (TensorFlow and PyTorch) and hardware (TPUs) to accelerate adoption.

In addition to a robust funding model for equation-based tools, more emphasis is needed on coding and software engineering skill-sets in undergraduate and graduate student engineering curriculum to meet growing demand. Current tools are fragmented and have limited interoperability. Additional resources are needed to blend data-driven and equation-based modeling and optimization methods. Recent progress has been made in physics-informed neural networks (Cai et al., 2022) and more progress will continue to blend paradigms.

Disclaimer: Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND2022-12379C

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283.

Abouee Mehrizi, H., A. Aminoleslami, J. Darko, E. Osei, and H. Mahmoudzadeh (2022). Staff scheduling during a pandemic: The case of radiation therapy department. *Available at SSRN 4104581*.

Alhelaly, S., A. Muthanna, and I. A. Elgendy (2022). Optimizing task offloading energy in multi-user multi-uav-enabled mobile edge-cloud computing systems. *Applied Sciences 12*(13), 6566.

Andersen, M., J. Dahl, Z. Liu, and L. Vandenberghe (2011). Interior-point methods for large-scale cone programming. *Optimization for machine learning*, 55–83.

Andersson, J. A., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2019). Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation 11*(1), 1–36.

Barbosa, A. S., L. Z. Tahara, and M. M. da Silva (2021). Motion planning of a fish-like piezoelectric actuated robot using model-based predictive control. *Journal of Vibration and Control*, 10775463211048255.

Bartusiak, R. D., S. Bitar, D. L. DeBari, B. G. Houk, D. Stevens, B. Fitzpatrick, and P. Sloan (2022). Open process automation: A standards-based, open, secure, interoperable process control architecture. *Control Engineering Practice 121*, 105034.

Beal, L. D., D. C. Hill, R. A. Martin, and J. D. Hedengren (2018). Gekko optimization suite. *Processes 6*(8), 106.

Becerra, V. M. (2010). Solving complex optimal control problems at no cost with psopt. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pp. 1391–1396. IEEE.

Bisschop, J. (2006). *AIMMS - Optimization Modeling*. Lulu.com.

Bisschop, J. and A. Meeraus (1982). On the development of a general algebraic modeling system in a strategic planning environment. In *Applications*, pp. 1–29. Springer.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems 33*, 1877–1901.

Burnell, E., N. B. Damen, and W. Hoburg (2020). Gpkit: A human-centered approach to convex optimization in engineering design. In *Proceedings of the 2020 chi conference on human factors in computing systems*, pp. 1–13.

Bynum, M. L., G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff (2021). *Pyomo–optimization modeling in python* (Third ed.), Volume 67. Springer Science & Business Media.

Cai, S., Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis (2022). Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 1–12.

Ceccon, F., J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener (2022). OMLT: Optimization & machine learning toolkit. *arXiv preprint arXiv:2202.02414*.

DEMİR, Y. Mathematical programming with c#. net. *Electronic Letters on Science and Engineering 17*(2), 96–104.

Dunning, I., J. Huchette, and M. Lubin (2017a). Jump: A modeling language for mathematical optimization. *SIAM Review 59*(2), 295–320.

Dunning, I., J. Huchette, and M. Lubin (2017b). Jump: A modeling language for mathematical optimization. *SIAM review 59*(2), 295–320.

Eeckhout, L. (2017). Is moore's law slowing down? what's next? *IEEE Micro 37*(04), 4–5.

Falck, R., J. S. Gray, K. Ponnapalli, and T. Wright (2021). dymos: A python package for optimal control of multidisciplinary systems. *Journal of Open Source Software 6*(59), 2809.

Fourer, R., D. Gay, and B. Kernighan (1993). Ampl. *Danvers, MA: Boyd & Fraser 117*.

Fraga-Lamas, P., L. Ramos, V. Mondéjar-Guerra, and T. M. Fernández-Caramés (2019). A review on iot deep learning uav systems for autonomous obstacle detection and collision avoidance. *Remote Sensing 11*(18), 2144.

Frumkin, J. A., V. Khanna, and M. F. Doherty (2022). Innovation in chemical reactor engineering practice and science. *Computers & Chemical Engineering 161*, 107699.

Grant, M. and S. Boyd (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura (Eds.), *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited.

Gray, J. S., J. T. Hwang, J. R. Martins, K. T. Moore, and B. A. Naylor (2019). Openmdao: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization 59*(4), 1075–1104.

Guo, Y., Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew (2016). Deep learning for visual understanding: A review. *Neurocomputing 187*, 27–48.

Gupta, A., A. Anpalagan, L. Guan, and A. S. Khwaja (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array 10*, 100057.

Hammoud, A., A. Diouf, and V. Perdereau (2021). A robotic in-hand manipulation dictionary based on human data. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 961–967. IEEE.

Han, J., M.-J. Tahk, and H.-L. Choi (2022). Pseudospectral method-based safe motion planning for quadrotors in a cluttered environment. In *AIAA SCITECH 2022 Forum*, pp. 2545.

Hart, W. E., C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola (2012). *Pyomo-optimization modeling in python*, Volume 67. Springer.

He, X., K. Zhao, and X. Chu (2021). Automl: A survey of the state-of-the-art. *Knowledge-Based Systems 212*, 106622.

Hernandez, M., R. Lecaros, and S. Zamorano (2022). Averaged turnpike property for differential equations with random constant coefficients. *Mathematical Control and Related Fields*.

Hijazi, H., G. Wang, and C. Coffrin (2018). Gravity: A mathematical modeling language for optimization and machine learning.

Houska, B., H. J. Ferreau, and M. Diehl (2011). Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods 32*(3), 298–312.

Kamilaris, A. and F. X. Prenafeta-Boldú (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture 147*, 70–90.

Kelly, J. D. and B. C. Menezes (2019). Industrial modeling and programming language (impl) for off-and on-line optimization and estimation applications. In *Optimization in Large Scale Problems*, pp. 75–96. Springer.

Kroshko, D. (2007). Openopt: Free scientific-engineering software for mathematical modeling and optimization. *URL http://www. openopt. org*.

Lai, B.-C., W.-Y. Chiu, and Y.-P. Tsai (2022). Multiagent reinforcement learning for community energy management

to mitigate peak rebounds under renewable energy uncertainty. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature 521*(7553), 436–444.

Lee, A., J. H. Ghouse, J. C. Eslick, C. D. Laird, J. D. Siirola, M. A. Zamarripa, D. Gunter, J. H. Shinn, A. W. Dowling, D. Bhattacharyya, et al. (2021). The idaes process modeling framework and model library—flexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing 3*(3), e10095.

Leineweber, D. B., A. Schäfer, H. G. Bock, and J. P. Schlöder (2003). An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization: Part ii: Software aspects and applications. *Computers & chemical engineering 27*(2), 167–174.

Listov, P. and C. Jones (2020). Polympc: An efficient and extensible tool for real-time nonlinear model predictive tracking and path following for fast mechatronic systems. *Optimal Control Applications and Methods 41*(2), 709–727.

Löfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan.

Lu, X., D.-S. Kim, and J. D. Vienna (2021). Impacts of constraints and uncertainties on projected amount of hanford low-activity waste glasses. *Nuclear Engineering and Design 385*, 111543.

Lucia, S., A. Tătulea-Codrean, C. Schoppmeyer, and S. Engell (2017). Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice 60*, 51–62.

Maher, S., M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, and F. Serrano (2016). PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pp. 301–307. Springer International Publishing.

Matthews, A. G. d. G., M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman (2017). Gpflow: A gaussian process library using tensorflow. *J. Mach. Learn. Res. 18*(40), 1–6.

Misra, S., L. R. Buttazoni, V. Avadiappan, H. J. Lee, M. Yang, and C. T. Maravelias (2022). CProS: A web-based application for chemical production scheduling. *Computers & Chemical Engineering 164*, 107895.

Mitchell, S., S. M. Consulting, and I. Dunning (2011). Pulp: A linear programming toolkit for python. The University of Auckland, Auckland, New Zealand.

Mowbray, M., M. Vallerio, C. Perez-Galvan, D. Zhang, A. D. R. Chanona, and F. J. Navarro-Brull (2022). Industrial data science–a review of machine learning applications for chemical and process industries. *Reaction Chemistry & Engineering*.

Nubli, H., J. M. Sohn, and A. R. Prabowo (2022). Layout optimization for safety evaluation on lng-fueled ship under an accidental fuel release using mixed-integer nonlinear programming. *International Journal of Naval Architecture and Ocean Engineering 14*, 100443.

Orban, D. (2014). Nlpy—a large-scale optimization toolkit in python. *Cahier du GERAD G-2014-xx, GERAD, Montréal, QC, Canada. In preparation*.

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). Automatic differentiation in pytorch.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc.

Patterson, M. A. and A. V. Rao (2014). Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS) 41*(1), 1.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Perez, R. E., P. W. Jansen, and J. R. Martins (2012). pyopt: a python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization 45*(1), 101–118.

Perron, L. (2011). Operations research and constraint programming at google. In *International Conference on Principles and Practice of Constraint Programming*, pp. 2–2. Springer.

Pulsipher, J. L., W. Zhang, T. J. Hongisto, and V. M. Zavala (2022). A unifying modeling abstraction for infinite-dimensional optimization. *Computers & Chemical Engineering 156*, 107567.

PYPL (2022). PYPL index.

Razzak, M. I., S. Naz, and A. Zaib (2018). Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps*, 323–350.

Ross, I. M. (2004). User's manual for DIDO: A matlab application package for solving optimal control problems. *Tomlab Optimization, Sweden*, 65.

Rutquist, P. E. and M. M. Edvall (2010). Propt-matlab optimal control software. *Tomlab Optimization Inc 260*(1).

Sagnol, G. and M. Stahlberg (2022). Picos: A python interface to conic optimization solvers. *Journal of Open Source Software 7*(70), 3915.

Santos, H. G. and T. A. Toffolo (2019). Tutorial de desenvolvimento de métodos de programação linear inteira mista em python usando o pacote python-mip. *Pesquisa Operacional para o Desenvolvimento 11*(3), 127–138.

Schumacher, D. (2022). *OMPR: Model and Solve Mixed Integer Linear Programs*. R package version 1.0.2.

Sobania, D., M. Briesch, and F. Rothlauf (2022). Choose your programming copilot: a comparison of the program synthesis performance of github copilot and genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1019–1027.

Tang, B. and E. B. Khalil (2022). Pyepo: A pytorch-based end-to-end predict-then-optimize library for linear and integer programming. *arXiv preprint arXiv:2206.14234*.

TIOBE (2022). TIOBE index.

Tuor, A., J. Drgona, M. Skomski, J. Koch, Z. Chen, S. Dernbach, C. M. Legaard, and D. Vrabie (2022). NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations.

US Bureau of Labor Statistics (2022). Occupational outlook handbook: Software developers, quality assurance analysts, and testers.

Verschueren, R., G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl (2022). acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation 14*(1), 147–183.

Wang, T., S. Ai, Z. Tian, B. B. Gupta, and C. Shan (2022). A blockchain-based distributed computational resource trading system for industrial internet of things considering multiple preferences. *arXiv preprint arXiv:2201.09539*.

Wu, X., X. Zhu, G.-Q. Wu, and W. Ding (2013). Data mining with big data. *IEEE transactions on knowledge and data engineering 26*(1), 97–107.

Zhou, F., Y. Li, W. Wang, and C. Pan (2022). Integrated energy management of a smart community with electric vehicle charging using scenario based stochastic model predictive control. *Energy and Buildings 260*, 111916.