Graphical Abstract

Equation-Based and Data-Driven Modeling: Open-Source Software Current State and Future Directions

LaGrande Gunnell, Bethany Nicholson, John D. Hedengren

Highlights

Equation-Based and Data-Driven Modeling: Open-Source Software Current State and Future Directions

LaGrande Gunnell, Bethany Nicholson, John D. Hedengren

- Review of open-source modeling software in scientific computing
- Critical evaluation of innovation with open-source modeling tools
- Perspective on the future of open-source tools in specialization and integration

Equation-Based and Data-Driven Modeling: Open-Source Software Current State and Future Directions

LaGrande Gunnell[a], Bethany Nicholson[b], John D. Hedengren[a]

[a]*Department of Chemical Engineering, Brigham Young University, 330 EB, Provo, 84602, UT, USA*
[b]*Center for Computing Research, Sandia National Laboratories, 1611 Innovation Pkwy SE, Albuquerque, 87123, NM, USA*

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 meet business needs while keeping the foundational tools 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. Integration of data-driven and principles-based 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 necessary to sustain the momentum and maintenance of critical foundational packages.

Keywords: Modeling, Open-source, Optimization, Simulation, Solver

1. 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. The influence of large language models on the speed of innovation will also be discussed. 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).

2. Current Trends in Scientific Computing

Programming jobs in software development, quality assurance, analysis, and testing will grow +22%, about 3 times faster than other occupations, over the next decade according to the US Bureau of Labor Statistics [1]. Python is the most popular programming language according to indices that track online searches [2]. Other scientific computing languages in the top 50 most popular programming languages include C/C++ (2/4), R (16), MATLAB

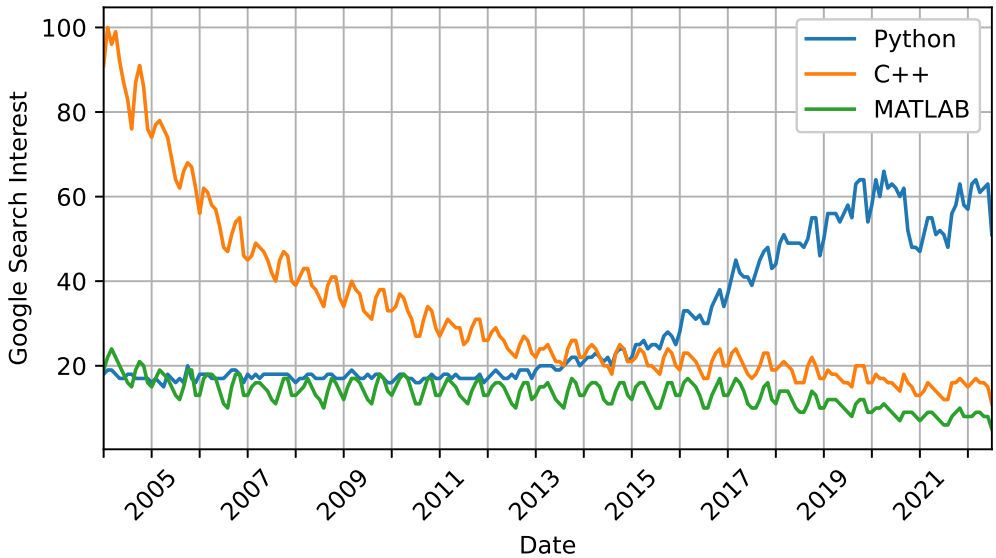

Figure 1: Trends of Python, C++, and MATLAB search interest from January 2004 to June 2022

(24), FORTRAN (26), Julia (28), Simulink (47), and LabVIEW (48) as of June 2022 [3].

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.

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. While Python is catered to more general usage, Julia is specifically designed for numerical and scientific computing, and many operations are both faster and easier to access than Python. Polymorphism implemented via multi-

ple dispatch in Julia enables more versatility in scientific computing as well. However, as it is a newer language, there is a limited set of libraries available in Julia.

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 (Downloads), Q+A Forum Posts, Citations
- Developers: Latest Release, Documentation, Support for Multiple Operating Systems, GitHub Insights (number of contributors, rate of development, number of issues, issue handling time, etc.)

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. Packages are more likely to be used if they are actively maintained and tested; compatibility and stability are necessary qualities to ensure long-term engagement, and are more common issues for open-source packages than for commercial software.

2.1. Large Language Models in Scientific Computing

Another important trend in scientific computing is that of generative AI in the form of large language models (LLMs) and the applications resulting from LLM developments. The most prevalent and available of these models is OpenAI's ChatGPT [4]. Google Scholar reports that the keyword "ChatGPT" has been used in the title or content of over 9000 articles as of 2023. ChatGPT has also become widely known and recognized in only a few short months, as shown in Figure 2.

Many of these large language models are trained on large collections of data that are open and available online, and extensive open source documentation has allowed these models to be used as coding and development aides. Chen et al. [5] evaluates the performance of different LLMs to write Python code and introduces the Codex model that powers Github Copilot. LLMs are massive and typically require billions of parameters, requiring high

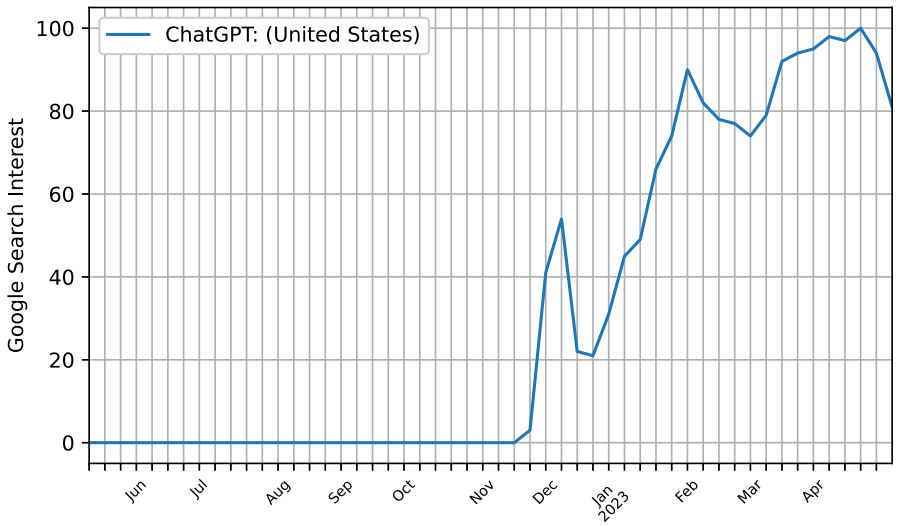

Figure 2: Google Trend of "ChatGPT" searches since May 2022.

power and memory capabilities. While most LLMs are not open source and only usable through an API, efforts like those from Meta AI have been made to make open source transformer-based models available for research and investigation at a less prohibitive memory requirement [6].

Advancements in LLMs have helped in the use and application of open source modeling platforms. As the documentation is open and available for many of the previously described packages, large language models like ChatGPT are trained on the documentation and can be used to write new code. More specific LLMs like Codex for Github copilot can be used to support model development and aid in the use of lower level tools to develop models.

Beyond code generation, there are a myriad of potential applications of LLM in the scientific computing community. LLMs have been applied to high-performance computing to evaluate and optimize modeling frameworks [7]. Multimodal analysis like Vision Language Models combine computer vision and natural language processing to enable a model to describe or caption an image [8]. Research and data extraction from the internet is more accessible and versatile with LLMs like ChatGPT [9]. Galactica, an open source LLM trained on scientific papers, achieves better scores and

higher reasoning on domain-specific questions than GPT-3 [10]. As LLMs are a quickly developing field, the number of applications are expected to grow; better pre-trained foundation models can lead to better performance with down-stream tasks, which will likely have a significant impact on the scientific community in the future.

3. Open-Source Modeling Packages

Modeling requires equations or data that characterize the system being studied. Should equations be available, the model can be developed and studied using Algebraic Modeling Languages (AMLs) like Pyomo, CasADi, Gekko, and JuMP. For systems where only data is available, data-driven packages, such as TensorFlow, PyTorch, or scikit-learn, can be used for development and training of many machine learning models for prediction purposes. This section will discuss and compare modeling packages and use metrics, applications of these modeling platforms, and many open source extensions that have been developed for these platforms.

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.

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.

3.1. 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. Solvers need information such as an objective function, constraints, and equation residuals. The user must also know the type of problem that is being described; linear programming (LP) solvers may not be able to handle mixed-integer programming (MIP) or nonlinear programming (NLP) problems. Some AMLs, like Pyomo, allow the problem to be described in the same syntax but can connect to different back-end solvers for different problem structures. Many solvers also require Jacobian and Hessian evaluations, so AMLs can 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 3. 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.

Pyomo [11] 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 a part of the Coopr software library but was released as its own package beginning in 2015. As of July 2022 there were 1270 Pyomo tagged questions on Stack Overflow. There is also a Google Group forum where users can post questions.

CasADi [12] is available in MATLAB, Python, and C++. It was originally a framework for automatic differentiation but has evolved into a complete

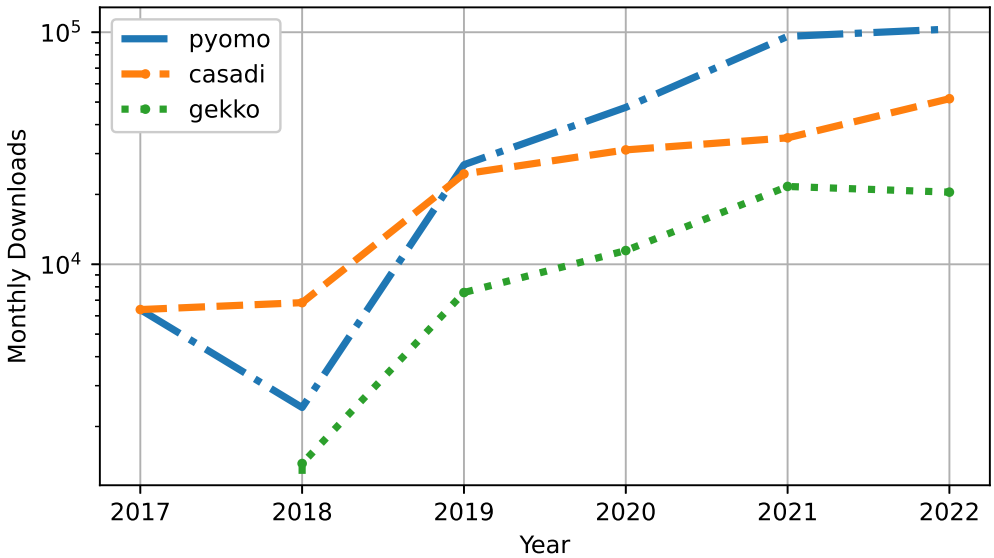

Figure 3: AML downloads for one month (June) each year (source: Google BigQuery).

modeling language. Casadi was first released in 2017 and as of July 2022, there were 72 questions on Stack Overflow. Most of the support questions are posted to a Google Group forum that delivers messages with email.

Gekko [13] 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 were 605 questions on Stack Overflow. Questions are also posted to a Google Group forum.

JuMP [14] 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 were 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 soft-

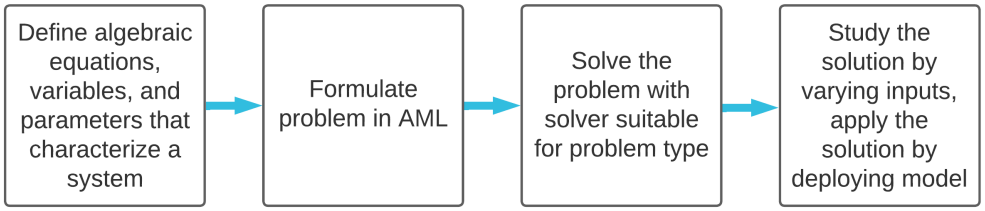

Figure 4: Flowchart of using an AML to model a system.

ware packages are available for optimization and control including ACADO [15], ACADOS [16], AIMMS [17], AMPL [18], CProS [19], CVX [20], CVX-OPT [21], DIDO [22], Dymos [23], GAMS [24], GPkit [25], GPOPS II [26], Gravity [27], IMPL [28], InfiniteOpt [29], MUSCOD-II [30], NLPy [31], OMPR [32], OpenMDAO [33], OpenOpt [34], OPTANO [35], OR-tools [36], PICOS [37], PROPT [38], PSOPT [39], PuLP [40], PyOpt [41], PySCIPOpt [42], Python-MIP [43], and YALMIP [44].

AMLs require equations to model a problem or system; these equations can be derived from first principles for a physics based system, or defined in the problem construction. Common problems that can be described in an AML and solved include differential equations, parameter regression and curve fitting, Model Predictive Control (MPC), and portfolio optimization. These problems can be solved and studied following a general flowchart shown in Figure 4.

Parameter regression is a common problem that can be formulated in an AML; presented here is an example of least squares linear regression. Given a set of linear data, the sum of squared errors between a linear model and the true data can be described symbolically in an AML and minimized to find optimal parameters. Although this particular regression method is simple and can be implemented with more efficient data-driven tools, the formulation of problems in an AML allows addition of constraints and creation of more complex models and systems in an equation-based framework. Below in Listings 1, 2, 3, and 4 are examples of the syntax used for Pyomo, CasADi, Gekko, and JuMP.

Listing 1: Pyomo Linear Regression

```
from pyomo.environ import *
m = ConcreteModel()
x = [1, 5, 10, 15, 20]
y = [8, 20, 35, 50, 65]
m.a = Var()
m.b = Var()
# Define the objective function
m.obj = Objective(expr=sum((m.a * x[i] + m.b - y[i]) ** 2 for i in range(5)))
SolverFactory('ipopt').solve(m) # Solve the optimization problem
print("a:", value(m.a)) # Display the results
print("b:", value(m.b))
```

Listing 2: CasADi Linear Regression

```
import casadi as ca
x_data = [1, 5, 10, 15, 20]
y_data = [8, 20, 35, 50, 65] # 3x + 5
a = ca.MX.sym("a")
b = ca.MX.sym("b")
params = ca.vertcat(a, b)
y_pred = [a * x_data[i] + b for i in range(5)]
SSE = sum((y_pred[i] - y_data[i])**2 for i in range(5))
nlp_prob = {'f': SSE, 'x': params}
opts = {'ipopt.print_level': 0, 'print_time': 0, 'ipopt.sb': 'yes'}
solver = ca.nlpsol('solver', 'ipopt', nlp_prob, opts)
# Solve the optimization problem
sol = solver(x0=[1, 1])
print(sol['x'][0]) # value of a
print(sol['x'][1]) # value of b
```

Listing 3: Gekko Linear Regression

```
from gekko import GEKKO
m = GEKKO()
x = [1,5,10,15,20]
y = [8,20,35,50,65]
a = m.Var()
b = m.Var()
ypred = m.Array(m.Var,5)
for i in range(5):
    ypred[i] = a*x[i] + b
SSE = m.sum((ypred - y)**2)
m.Obj(SSE)
m.options.IMODE = 2
m.solve(disp=False)
print(a.value[0])
print(b.value[0])
```

Listing 4: JuMP Linear Regression

```
using JuMP, Ipopt
x = [1, 5, 10, 15, 20]
y = [8, 20, 35, 50, 65]
m = Model(optimizer_with_attributes(Ipopt.Optimizer, "print_level" => 0))
@variable(m, a)
@variable(m, b)
@objective(m, Min, sum((a * x[i] + b - y[i])^2 for i in 1:5))
optimize!(m)
a_opt = value(a)
b_opt = value(b)
println("a: ", a_opt)
println("b: ", b_opt)
```

These code snippets were generated with the help of ChatGPT. After describing the model in Gekko, the code was translated with minimal user adjustments to the other AMLS. As ChatGPT was trained on all of the

FLOSS documentation, support forums, and other applicable text, ChatGPT was able to debug any minor problems after an additional prompt. Even so, the final implementation offered by the language model may not be optimal. With further advancements of LLMs, the process of learning AML syntax and then describing a system will be made simpler. Models would be able to be translated between packages with less difficulty, allowing for greater insight and model customization.

A performance benchmark of FLOSS AMLs and a review of open-source solvers is beyond the scope of this review. Recent work has been done to benchmark and compare AMLs based on features and performance [45]. There are many examples of benchmark problems to test AMLs, but most problems can be formulated differently leading to more or less efficient solutions. A recent blog post by GAMS developers sought to benchmark the GAMS, Pyomo, GurobiPy, and JuMP AMLs [46]. The post concluded that GAMS had better performance over the alternatives on a specific model problem. The JuMP developers responded with their own post showing how a reformulation of the problem can lead to a faster solution, and concluded that such a comparison is difficult to produce without thorough community feedback [47]. The task of comparing AMLs is more difficult than a straightforward performance or scale comparison; as there is more flexibility in problem formulation, there is more room for error. Additionally, as AMLs offer different syntax and capabilities, the notion of best may be dependent on user preference. A notable emerging trend is tighter coupling between the solver and modeling language for callbacks, adaptive programming, and meta-algorithm development.

3.2. Data-Driven Modeling

The dramatic rise of data-driven modeling can be attributed to increased data availability, decreased compute cost, and powerful data-driven software tools. Two of the most popular packages for deep learning are TensorFlow [48] and PyTorch [49] that were developed at Alphabet (Google Brain) and Meta AI (Facebook), respectively. The machine learning package scikit-learn [50] was started in 2007 as a Google Summer of Code project. Figure 5 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.

Part of the core business model of both Meta and Alphabet is to sell advertisements and online services rather than proprietary optimization software.

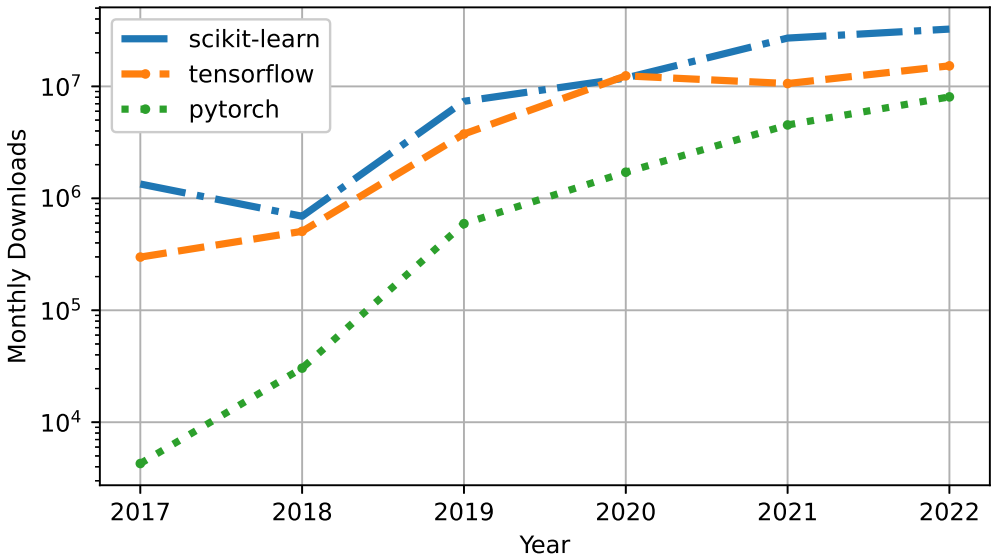

Figure 5: Data-driven package downloads for one month (June) each year (source: Google BigQuery).

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 energy efficiency or the rate of computation per unit of 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 the automotive, telecommunication, and medical industries. ASICS are designed for a specific task while a Central Processing

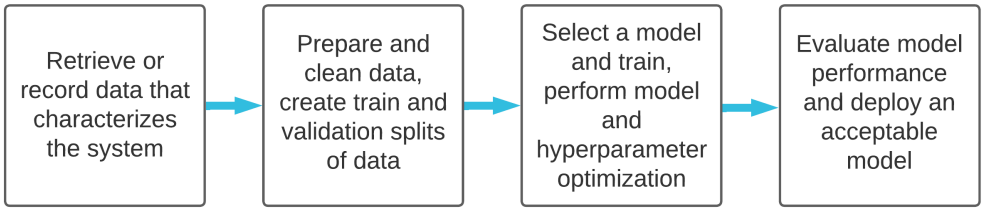

Figure 6: Flowchart of machine learning pipeline.

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 [51]. 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.

The availability of training datasets encourages exploration and improvement of open source machine learning models and enables performance comparisons between models and packages. The MNIST [52] dataset is a set of 60,000 training images of handwritten digits for classification. Available medicinal and financial datasets for regression purposes help improve the prediction models to be used in their respective fields. Websites like Kaggle and DataONE host and share thousands of datasets for public use and model development. For some websites where the API allows, data can be scraped and used for model development. One example of this is the training of large language models that are trained on text data retrieved from the internet.

The machine learning pipeline, visualized in Figure 6, shows a general flow of retrieving data, processing data, training models, and then model deployment. The data-driven packages described previously offer tools for model selection and training. Scikit-learn offers a wide selection of models that are user friendly to train, but does not have the same depth and customization of Pytorch or Tensorflow. With the performance advantages explained before, Tensorflow and Pytorch are specialized for deep learning and model development.

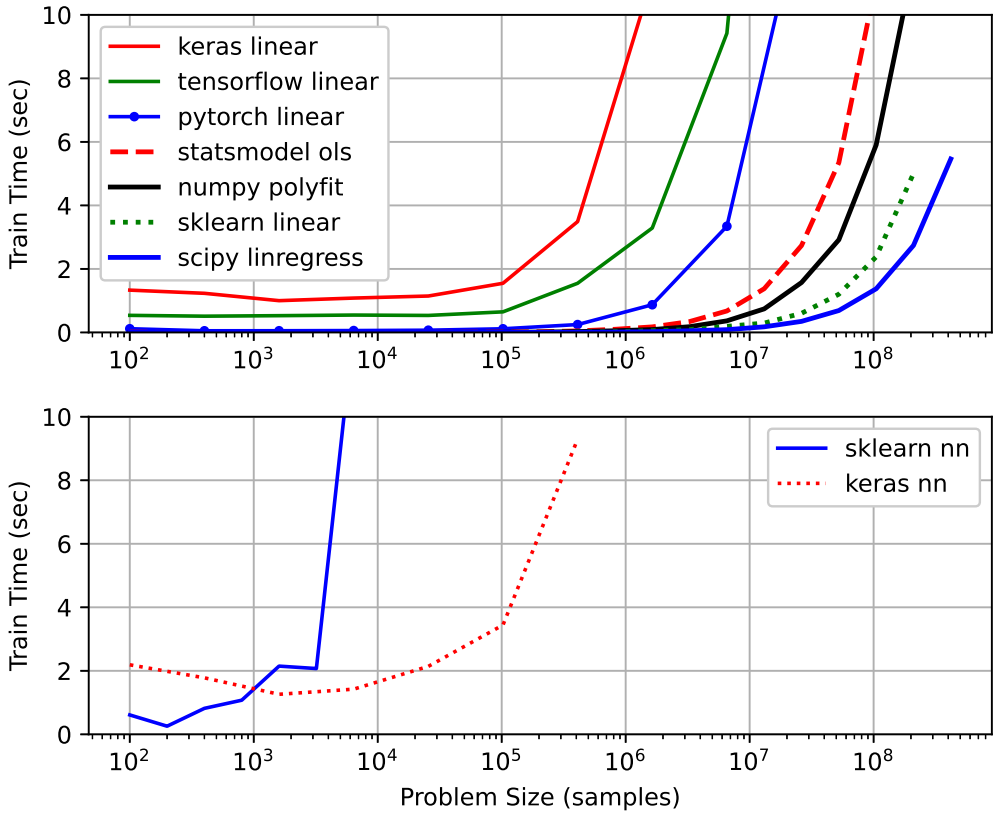

Figure 7: Linear regression scale-up comparison performed on AMD Ryzen 7 2700X Eight-Core Processor.

Figure 7 shows a performance comparison between several data-driven packages for linear regression. Different implementations of similar methods, like linear regression, are available in many of these packages. Although the figure shows performance comparisons on a CPU, Pytorch and Tensorflow allow GPU acceleration which would lead to better scale-up for larger problems. Scikit-learn and Numpy offer easy and efficient implementations of some models but do not offer the same level of customization. Overall, the best package for a model implementation depends on the scale of the problem, the available computing power, and the level of customization required.

3.3. Extensions and Applications of FLOSS Modeling Platforms

FLOSS platforms allow users to contribute to further developments of the code, leading to many tools and platforms that extend the capabilities of these packages. Extensions have allowed AMLs to interface to data driven modeling packages or provide application-specific functionality such as support for model predictive control. User extensions of data driven packages have enabled easier and faster algorithm selection and training and streamlined hyperparameter tuning and model deployment. In some cases, these tools have been incorporated into larger frameworks to solve a broader set of problems or to handle a specific application. Many FLOSS packages encourage users to contribute to public repositories to ensure the continual improvement and maintenance of these packages.

The combination of data-driven modeling and equation-based modeling can replace costly simulations and functions, improving the performance and usability of larger numerical optimization formulations. The OMLT [53] package allows neural network and gradient-boosted tree models to be represented in a Pyomo optimization problem. Similar work has been done with Gekko that integrates Gaussian process regression, support vector machines, and neural networks into gradient-based optimization [54].

Other extensions have expanded the capabilities of AMLs into Model Predictive Control (MPC). For example, do-mpc [55] and PolyMPC [56] are libraries for Model Predictive Control (MPC) built on CasADi. Other packages have used neural networks from Pytorch for machine learning MPC in CasADi [57].

Data-driven modeling packages have been improved and extended with user made tools, enabling better model selection and training. Keras [58] is a user-friendly API for Tensorflow, and an example of how deep learning can be simplified and streamlined with extensions. Open source automated machine learning packages like auto-sklearn [59], auto-Keras [60], H2O AutoML [61], and TPOPT [62] automate the pre-processing and training of models. Other tools like ONNX provide a standard for interfacing between data-driven modeling packages for better interoperability when training deep learning networks.

Open source packages can be incorporated into larger frameworks to solve larger scale problems, particularly in the field of energy system optimization and process systems engineering. One example of such a framework is the IDAES Integrated Platform [63], visualized in Figure 8, which builds on

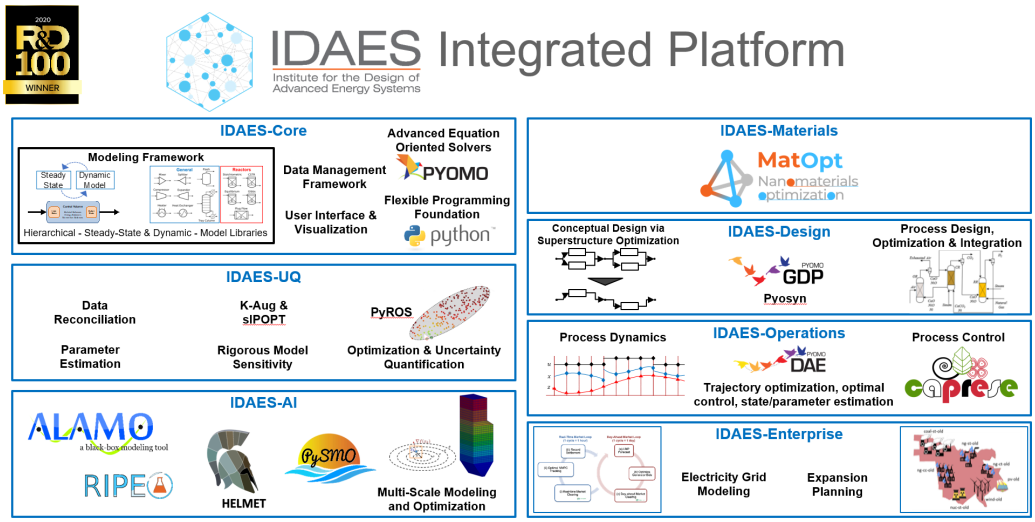

Figure 8: Overview of the IDAES platform

Pyomo to provide tools for the design and optimization of complex, interacting energy and process systems. Similar frameworks have been developed for the modeling and optimization of renewable energy systems based on Pyomo, GAMS, and C++; these include urbs, Balmorel [64], GENeSYS-MOD, GENESYS-2 [65], and oemof. FOQUS is another framework used for process systems engineering that connects to both IDAES and Pyomo. As these frameworks are open-source, evaluations and comparisons between similar frameworks can help improve modeling performance by identifying missing features [66]. There are many more examples of application-specific packages, further illustrating the value these generic FLOSS optimization platforms provide to the community.

4. Impacts of Open-Source Tools

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 [67], staff scheduling [68], mathematical research [69], renewable energy grid optimization [70], control of electric vehicle charging in smart communities [71], chemical reactor design [72], blockchain computing

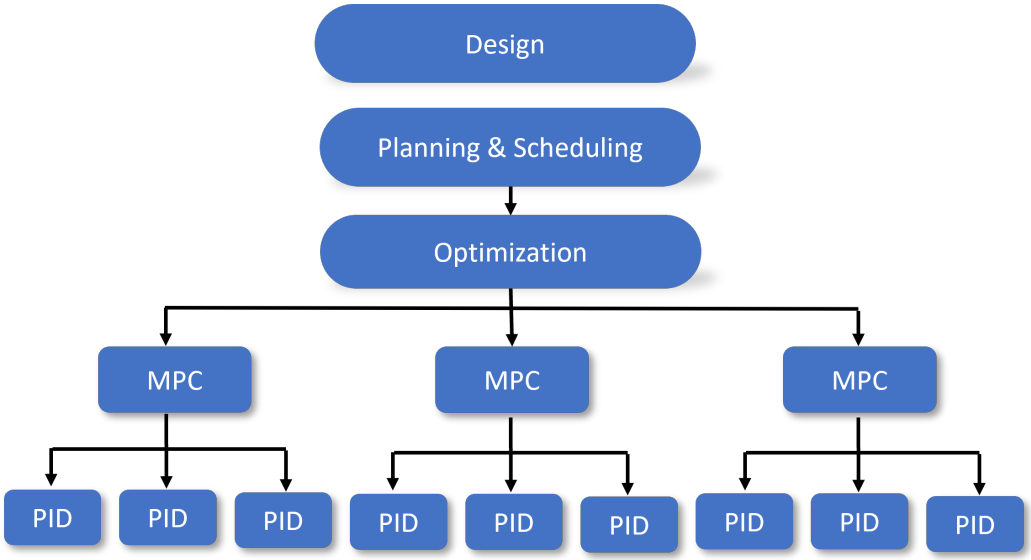

Figure 9: Hierarchy of applications in the chemical and process industries. New modeling and optimization developments are finding synergies between areas that were formerly separate.

optimization for Industrial Internet of Things (IoT) [73], safety systems on Liquified Natural Gas (LNG) vessels [74], robotic hand automation [75], autonomous unmanned aerial vehicles [76, 77], low-activity waste loading for long-term storage with vitrification [78], and fish-like robots [79]. Many other applications are cited, giving strong evidence of user adoption with innovative application areas. As of October 2023, there were 1633 citations of Pyomo [11], 516 citations of APMonitor and Gekko [13], 2987 citations of CasADi [12], and 1681 citations of JuMP [80].

The pace of innovation is likewise supported by FLOSS tools in data-driven modeling and optimization with notable advances in natural language processing [81], self-driving cars [82], image classification [83], medical diagnosis [84], precision agriculture [85], autonomous unmanned aerial vehicles [86], and many other areas [87]. As of October 2023, there were 27,689 citations of TensorFlow [88, 48], 33,617 citations of PyTorch [49, 89], and 80,652 citations of Scikit-learn [50].

Model predictive control (MPC) has benefited from recent advancements with data-driven modeling and machine learning. While classical approaches to MPC use linear methods, data-driven modeling allows for more complex

and general nonlinear behavior [90]. Reinforcement learning in particular has found great success for control of robotics and self driving cars, and may have a future in the process industry with MPC [91]. For process systems with large volumes of historical data, machine learning models can be trained as digital twins for further analysis and control. Integration of AI-based digital twins into large processes can lead to higher quality data and better performing models [92]. Advanced models that capture total system behavior can develop synergies between different hierarchies in control schemes, shown in Figure 9. Additionally, smaller scale industrial tasks such as detecting anomalies or equipment faults have been greatly facilitated by machine learning advancements [93].

5. 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.

5.1. 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 [94], integration of machine learning models in Pyomo [53], constrained optimization with physics-based modeling priors in PyTorch [95], and Gekko interfaces to GPflow [96] and scikit-learn. Developments with package interoperability will continue to accelerate in the next 5 years.

There are new development resources for code auto-completion such as GitHub Copilot [97] 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 [98].

Data engineering, organizing and preparing data for the purpose of extracting useful information [99], will also be increasingly important.

5.2. 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.

A full review of the ethics of sharing FLOSS tools is beyond the scope of this paper; however, a brief discussion is presented here. FLOSS can be potentially used by rogue actors for malicious intent. The emergence of machine learning and artificial intelligence is comparable in societal impact to the emergence of nuclear power in the 1950s, yet nuclear power centered software is largely export controlled for purposes of national security. Contributors with malicious intentions may take advantage of the open source nature of these packages to bypass the security of users of FLOSS tools. Advanced LLM tools that become popular and widely used have the potential to enable dishonest behavior in education, publication, and communication. A further in-depth consideration of the potential ethical impacts of FLOSS is needed for future work.

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. While LLMs can alleviate the skill barrier, domain specific advancements are needed if LLMs are to be applied in fields like PSE [100]. 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 [101] and more progress will continue to blend paradigms.

5.3. Final Thoughts

The future of FLOSS tools in scientific computing is promising. Software will continue to develop and improve as the scientific community grows; state of the art modeling methods are frequently released on platforms like Hugging Face for community use and continued research [102]. Important key performance indicators of successful open source tools are the same indicators of current trends; user count, stability, citations, and interfaces to other platforms. As there is a growing interest in large and compute heavy models (especially LLMs), there will be a larger focus on performance, parallelization, and scalability for data-driven modeling packages. Pytorch and Tensorflow are widely used because of accessible distributive training strategies and GPU hardware acceleration. For AMLs, both usable syntax and performance will be important. One review has found that AMPL and GAMS allow for the easiest problem description and fastest model solve times [45]. As previously mentioned, comparing AMLs is a more difficult task due to different problem formulations and capabilities. As the Julia language grows, popular scientific computing packages may be translated to or remade for the newer language.

There are a few challenges that may shape the future of open source software. As software and hardware develops, FLOSS tools will need continual updates and maintenance; platforms like COIN-OR will need more support for future use. For some modeling ideas with high impact, like LLMs, competition between groups can lead to certain software and architectures to become trade secret. The GPT-4 architecture is not publicly available, unlike its predecessors [4]. Finally, FLOSS tools will need to be more usable

and preferable to commercial alternatives for continued success in industrial cases.

The packages mentioned in this review are considered successful because they are popular, usable, and performant. Going forward, these packages will need to be continually maintained and improved for future success. Interfaces between platforms, like ONNX for neural networks, will bolster usage and impact of open source platforms [103]. As packages become more interconnected, tools that are not updated will become obsolete. With more accessible tools and greater interest in modeling, the scientific community is expected to grow, leading to further innovation and improvement with FLOSS tools.

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.

References

[1] US Bureau of Labor Statistics, Occupational outlook handbook: Software developers, quality assurance analysts, and testers (2022). URL `https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm`

[2] PYPL, PYPL index (2022). URL `https://pypl.github.io/PYPL.html`

[3] TIOBE, TIOBE index (2022). URL `https://www.tiobe.com/tiobe-index/`

[4] OpenAI, Gpt-4 technical report (2023). `arXiv:2303.08774`.

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba, Evaluating Large Language Models Trained on Code, arXiv:2107.03374 [cs] (Jul. 2021). doi:10.48550/arXiv.2107.03374.
URL http://arxiv.org/abs/2107.03374

[6] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer, OPT: Open Pre-trained Transformer Language Models, arXiv:2205.01068 [cs] (Jun. 2022). doi:10.48550/arXiv.2205.01068.
URL http://arxiv.org/abs/2205.01068

[7] L. Chen, P.-H. Lin, T. Vanderbruggen, C. Liao, M. Emani, B. de Supinski, LM4HPC: Towards Effective Language Model Application in High-Performance Computing, in: S. McIntosh-Smith, M. Klemm, B. R. de Supinski, T. Deakin, J. Klinkenberg (Eds.), OpenMP: Advanced Task-Based, Device and Compiler Programming, Lecture Notes in Computer Science, Springer Nature Switzerland, Cham, 2023, pp. 18–33. doi:10.1007/978-3-031-40744-4_2.

[8] W. Hu, Y. Xu, Y. Li, W. Li, Z. Chen, Z. Tu, BLIVA: A Simple Multimodal LLM for Better Handling of Text-Rich Visual Questions, arXiv:2308.09936 [cs] (Aug. 2023). doi:10.48550/arXiv.2308.09936.
URL http://arxiv.org/abs/2308.09936

[9] R. Krosnick, S. Oney, Promises and Pitfalls of Using LLMs for Scraping Web UIs (2023).

[10] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, R. Stojnic, Galactica: A Large Language Model for Science, arXiv:2211.09085 [cs, stat] (Nov. 2022).

doi:10.48550/arXiv.2211.09085.
URL http://arxiv.org/abs/2211.09085

[11] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, D. L. Woodruff, Pyomo–optimization modeling in python, 3rd Edition, Vol. 67, Springer Science & Business Media, 2021.

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

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

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

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

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

[17] J. Bisschop, AIMMS - Optimization Modeling, Lulu.com, 2006.

[18] R. Fourer, D. Gay, B. Kernighan, Ampl, Danvers, MA: Boyd & Fraser 117 (1993).

[19] S. Misra, L. R. Buttazoni, V. Avadiappan, H. J. Lee, M. Yang, C. T. Maravelias, CProS: A web-based application for chemical production scheduling, Computers & Chemical Engineering 164 (2022) 107895. doi:https://doi.org/10.1016/j.compchemeng.2022.107895.
URL https://www.sciencedirect.com/science/article/pii/S0098135422002332

[20] M. Grant, S. Boyd, Graph implementations for nonsmooth convex programs, in: V. Blondel, S. Boyd, H. Kimura (Eds.), Recent Advances

in Learning and Control, Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110.

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

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

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

[24] J. Bisschop, A. Meeraus, On the development of a general algebraic modeling system in a strategic planning environment, in: Applications, Springer, 1982, pp. 1–29.

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

[26] M. A. Patterson, A. V. Rao, 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) (2014) 1.

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

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

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

[30] D. B. Leineweber, A. Schäfer, H. G. Bock, J. P. Schlöder, 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) (2003) 167–174.

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

[32] D. Schumacher, OMPR: Model and Solve Mixed Integer Linear Programs, r package version 1.0.2 (2022).
URL `https://CRAN.R-project.org/package=ompr`

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

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

[35] Y. DEMİR, Mathematical programming with c#. net, Electronic Letters on Science and Engineering 17 (2) (2021) 96–104.

[36] L. Perron, Operations research and constraint programming at google, in: International Conference on Principles and Practice of Constraint Programming, Springer, 2011, pp. 2–2.

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

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

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

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

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

[42] S. Maher, M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, F. Serrano, PySCIPOpt: Mathematical programming in python with the SCIP optimization suite, in: Mathematical Software – ICMS 2016, Springer International Publishing, 2016, pp. 301–307. doi:10.1007/978-3-319-42432-3_37.

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

[44] J. Löfberg, Yalmip : A toolbox for modeling and optimization in matlab, in: In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004, pp. 284–289.

[45] V. Jusevičius, R. Oberdieck, R. Paulavičius, Experimental Analysis of Algebraic Modelling Languages for Mathematical Optimization, Informatica 32 (2) (2021) 283–304, publisher: Vilnius University Institute of Data Science and Digital Technologies. doi:10.15388/21-INFOR447. URL https://informatica.vu.lt/journal/INFORMATICA/article/1216

[46] J. Broihan, Performance in optimization models: A comparative analysis of gams, pyomo, gurobipy, and jump (Jul 2023). URL https://www.gams.com/blog/2023/07/performance-in-optimization-models-a-comparative-analysis-of-gams-pyomo-guro

[47] JuMPjl, JuMP, GAMS, and the IJKLM model (Jul. 2023). URL https://jump.dev/2023/07/20/gams-blog/

[48] M. Abadi, 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, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL https://www.tensorflow.org/

[49] A. Paszke, 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, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[50] F. Pedregosa, 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, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[51] L. Eeckhout, Is Moore's Law slowing down? what's next?, IEEE Micro 37 (04) (2017) 4–5.

[52] L. Deng, The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web], IEEE Signal Processing Magazine 29 (6) (2012) 141–142, conference Name: IEEE Signal Processing Magazine. doi:10.1109/MSP.2012.2211477.

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

[54] L. L. Gunnell, K. Manwaring, X. Lu, J. Reynolds, J. Vienna, J. Hedengren, Machine Learning with Gradient-Based Optimization of Nuclear Waste Vitrification with Uncertainties and Constraints, Processes 10 (11) (2022) 2365, number: 11 Publisher: Multidisciplinary Digital Publishing Institute. doi:10.3390/pr10112365.
URL https://www.mdpi.com/2227-9717/10/11/2365

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

[56] P. Listov, C. Jones, 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) (2020) 709–727.

[57] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, M. Ryll, Real-time Neural-MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms, IEEE Robotics and Automation Letters 8 (4) (2023) 2397–2404, arXiv:2203.07747 [cs, eess]. doi:10.1109/LRA.2023.3246839.
URL http://arxiv.org/abs/2203.07747

[58] F. Chollet, others, Keras (2015).
URL https://keras.io

[59] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and Robust Automated Machine Learning, in: Advances in Neural Information Processing Systems 28 (2015), 2015, pp. 2962–2970.

[60] H. Jin, F. Chollet, Q. Song, X. Hu, AutoKeras: An AutoML Library for Deep Learning, Journal of Machine Learning Research 24 (6) (2023) 1–6.
URL http://jmlr.org/papers/v24/20-1355.html

[61] E. LeDell, S. Poirier, H2O AutoML: Scalable Automatic Machine Learning, 7th ICML Workshop on Automated Machine Learning (AutoML) (Jul. 2020).
URL https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf

[62] R. S. Olson, J. H. Moore, TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning, in: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, The Springer Series on Challenges in Machine Learning, Springer International Publishing, Cham, 2019, pp. 151–160.

doi:10.1007/978-3-030-05318-5_8.
URL https://doi.org/10.1007/978-3-030-05318-5_8

[63] A. Lee, 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., The idaes process modeling framework and model library—flexibility for process simulation and optimization, Journal of Advanced Manufacturing and Processing 3 (3) (2021) e10095.

[64] F. Wiese, R. Bramstoft, H. Koduvere, A. P. Alonso, O. Balyk, J. G. Kirkerud, Å. G. Tveten, T. F. Bolkesjø, M. Münster, H. Ravn, Balmorel open source energy system model, Energy strategy reviews 20 (2018) 26–34.

[65] C. Bußar, Investigation of optimal transformation paths until 2050 for the successful implementation of a sustainable reduction of carbon dioxide emissions in the field of power generation (2019).

[66] S. Candas, C. Muschner, S. Buchholz, R. Bramstoft, J. van Ouwerkerk, K. Hainsch, K. Löffler, S. Günther, S. Berendes, S. Nguyen, A. Justin, Code exposed: Review of five open-source frameworks for modeling renewable energy systems, Renewable and Sustainable Energy Reviews 161 (2022) 112272. doi:10.1016/j.rser.2022.112272.
URL https://www.sciencedirect.com/science/article/pii/S1364032122001927

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

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

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

[70] B.-C. Lai, W.-Y. Chiu, Y.-P. Tsai, Multiagent reinforcement learning for community energy management to mitigate peak rebounds under

renewable energy uncertainty, IEEE Transactions on Emerging Topics in Computational Intelligence (2022).

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

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

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

[74] H. Nubli, J. M. Sohn, A. R. Prabowo, 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 (2022) 100443.

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

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

[77] J. Han, M.-J. Tahk, H.-L. Choi, Pseudospectral method-based safe motion planning for quadrotors in a cluttered environment, in: AIAA SCITECH 2022 Forum, 2022, p. 2545.

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

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

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

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

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

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

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

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

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

[87] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

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

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

[90] L. T. Biegler, A perspective on nonlinear model predictive control, Korean Journal of Chemical Engineering 38 (7) (2021) 1317–1332. doi:10.1007/s11814-021-0791-7.
URL https://doi.org/10.1007/s11814-021-0791-7

[91] D. Görges, Relations between Model Predictive Control and Reinforcement Learning, IFAC-PapersOnLine 50 (1) (2017) 4920–4928. doi:10.1016/j.ifacol.2017.08.747.
URL https://www.sciencedirect.com/science/article/pii/S2405896317311941

[92] L. Hewing, K. P. Wabersich, M. Menner, M. N. Zeilinger, Learning-Based Model Predictive Control: Toward Safe Learning in Control, Annual Review of Control, Robotics, and Autonomous Systems 3 (1) (2020) 269–296, _eprint: https://doi.org/10.1146/annurev-control-090419-075625. doi:10.1146/annurev-control-090419-075625.
URL https://doi.org/10.1146/annurev-control-090419-075625

[93] S. A. A. Taqvi, H. Zabiri, L. D. Tufa, F. Uddin, S. A. Fatima, A. S. Maulud, A Review on Data-Driven Learning Approaches for Fault Detection and Diagnosis in Chemical Processes, ChemBioEng Reviews 8 (3) (2021) 239–259, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cben.202000027. doi:10.1002/cben.202000027.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cben.202000027

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

[95] A. Tuor, J. Drgona, M. Skomski, J. Koch, Z. Chen, S. Dernbach, C. M. Legaard, D. Vrabie, NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations (2022).
URL https://github.com/pnnl/neuromancer

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

[97] D. Sobania, M. Briesch, F. Rothlauf, 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, 2022, pp. 1019–1027.

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

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

[100] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith, Don't Stop Pretraining: Adapt Language Models to Domains and Tasks, arXiv:2004.10964 [cs] (May 2020). doi:10.48550/arXiv.2004.10964.
URL http://arxiv.org/abs/2004.10964

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

[102] H. Face, Hugging face (2023).
URL http://huggingface.co

[103] O. R. developers, Onnx runtime, https://onnxruntime.ai/, version: x.y.z (2021).