SBMLTOODEJAX: Efficient Simulation and Optimization of Biological Network Models in JAX

Mayalen Etcheverry*
Flowers Team
Inria, Univ. Bordeaux (France)
mayalen.etcheverry@inria.fr

Michael Levin
Allen Discovery Center
Tufts University (USA)
michael.levin@@tufts.edu

Clément Moulin-Frier
Flowers Team
Inria, Univ. Bordeaux (France)
clement.moulin-frier@inria.fr

Pierre-Yves Oudeyer
Flowers Team
Inria, Univ. Bordeaux (France)
pierre-yves.oudeyer@inria.fr

Abstract

Advances in bioengineering and biomedicine demand a deep understanding of the dynamic behavior of biological systems, ranging from protein pathways to complex cellular processes. Biological networks like gene regulatory networks and protein pathways are key drivers of embryogenesis and physiological processes. Comprehending their diverse behaviors is essential for tackling diseases, including cancer, as well as for engineering novel biological constructs. Despite the availability of extensive mathematical models represented in Systems Biology Markup Language (SBML), researchers face significant challenges in exploring the full spectrum of behaviors and optimizing interventions to efficiently shape those behaviors. Existing tools designed for simulation of biological network models are not tailored to facilitate interventions on network dynamics nor to facilitate automated discovery. Leveraging recent developments in machine learning (ML), this paper introduces SBMLTOODEJAX, a lightweight library designed to seamlessly integrate SBML models with ML-supported pipelines, powered by JAX. SBMLTOODEJAX facilitates the reuse and customization of SBML-based models, harnessing JAX's capabilities for efficient parallel simulations and optimization, with the aim to accelerate research in biological network analysis.

1 Introduction

Developing methods to explore, predict and control the dynamic behavior of biological systems, from protein pathways to complex cellular processes, is an essential frontier of research for bioengineering and biomedicine [1]. Systems such as gene regulatory networks and protein pathways play pivotal roles in directing embryogenesis, influencing cellular activities, and governing intricate physiological processes. Understanding the range of behaviors that these systems can exhibit is crucial for comprehending and intervening in various disease states, including cancer [2–4], and for the development of novel bioengineered constructs in synthetic morphology contexts [5–7].

Thus, significant effort has gone in computational inference and mathematical modeling of biological systems, which has resulted in the development of large collections of publicly-available models, typically stored and exchanged on online platforms (such as the BioModels Database [8, 9]) using

*Source code, documentation and online tutorials can be found at http://developmentalsystems.org/sbmltoodejax. Please contact Mayalen Etcheverry for any additional questions.

NeurIPS 2023 AI for Science Workshop.

the Systems Biology Markup Language (SBML), a standard format for representing mathematical models of biological systems [10, 11].

Yet, despite the wealth of available SBML models, scientists still lack an in-depth understanding of the range of possible behaviors that these models can exhibit under different initial data and environmental stimuli, and lack effective methods to reveal and optimize those behaviors via external interventions. Except for a set of simple networks where system behavior and response to stimuli can be well understood analytically (or with exhaustive enumeration methods), onerous sampling of the parameter space and time-consuming numerical simulations are needed. This remains a major roadblock for progress in biological network analysis.

On the other hand, recent progress in machine learning (ML) has led to the development of novel computational tools that leverage high-performance computation, parallel execution and differentiable programming and that promise to accelerate research across multiple areas of science, including biological network analysis [12] and applications in drug discovery and molecular medicine [13, 14].

However, to our knowledge, there is no software tool that allows seamless integration of existing mathematical models of cellular molecular pathways (SBML files constructed by biologists) with ML-supported pipelines and programming frameworks. Whereas there exists many software tools for manipulation and numerical simulation of SBML models, they typically rely either on specialized simulation platforms limiting the flexibility for customization and scripting (such as COPASI [15, 16], Virtual Cell [17, 18] and Cell Designer [19, 20]) or provide scripting interfaces in Python or Matlab but rely on backend engines that do not support hardware acceleration or automatic differentiation (like Tellurium [21, 22] and SBMLtoODEpy [23] Python packages, or the Systems Biology Format Converter (SBFC) which generates MATLAB and OCTAVE code [24]).

In this paper we present SBMLTOODEJAX, a lightweight library that seeks to bridge that gap by bringing SBML simulation to the JAX ecosystem, a thriving community of JAX libraries that aim to accelerate research in machine learning and beyond, with diverse applications spanning molecular dynamics [25], protein engineering [26], quantum physics [27], cosmology [28], ocean modeling [29], photovoltaic research [30], acoustic simulations [31] and fluid dynamics [32].

SBMLTOODEJAX aims to integrate this ecosystem and provide tools to accelerate research in biological network analysis. In this paper, we explain how SBMLTOODEJAX facilitates the re-use of existing biological network models, as well as their manipulation in Python projects and AI pipelines, while tailoring them to take advantage of JAX main features for efficient and parallel computations.

2 SBMLTOODEJAX

SBMLTOODEJAX is a lightweight library that allows to automatically parse and convert SBML models into Python models written end-to-end in JAX, a high-performance numerical computing library with automatic differentiation capabilities [33]. SBMLTOODEJAX is targeted at researchers that aim to incorporate SBML-specified ordinary differential equation (ODE) models into their pPthon projects and machine learning pipelines, in order to perform efficient numerical simulation and optimization with only a few lines of code.

Taking advantage of JAX's core transformation features, one can easily boost the speed of ODE models time-course simulations and perform efficient search and optimization by running simulations in parallel and/or using automatic differentiation to find derivatives.

SBMLTOODEJAX extends SBMLtoODEpy, a Python library developed in 2019 for converting SBML files into Python files written in Numpy/Scipy [23]. The chosen conventions for the generated variables and modules are slightly different from the standard SBML conventions (and from the conventions used in the original SBMLtoODEpy package) with the aim to accommodate for more flexible manipulations while preserving JAX-like functional programming style.

We refer to the documentation available at `https://developmentalsystems.org/sbmltoodejax/` for additional details on SBMLTOODEJAX's design principles, chosen conventions for the generated variables and modules, and full API docs. Various hands-on tutorial notebooks for loading and simulating biomodels, running parallel executions and performing gradient descent optimization with the generated ODE models are also provided.

3 Why use SBMLTOODEJAX?

Simplicity and extensibility SBMLTOODEJAX retains the simplicity of the SBMLtoODEpy library to facilitate incorporation of the ODE models into one's own Python projects. As shown in Figure 1, with only a few lines of Python code, one can load and simulate existing SBML files. Moreover, one can easily refactor the models to its need.

```python
from sbmltoodejax.utils import load_biomodel

# load and simulate model
model, _, _, _ = load_biomodel(10)
n_secs = 150*60
n_steps = int(n_secs / model.deltaT)
ys, ws, ts = model(n_steps)

# plot time-course simulation
...
```

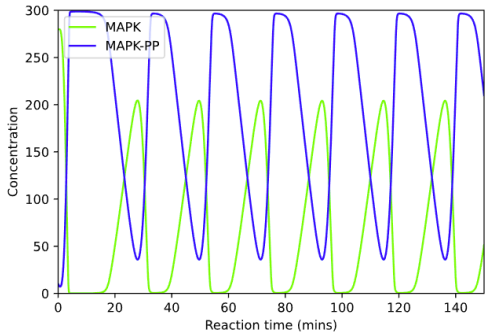

Figure 1: Example code (left) and output snapshot (right) reproducing original simulation results of Kholodenko 2000's paper [34] hosted on BioModels.

JAX-friendly The generated Python models are tailored to take advantage of JAX main features. Model rollouts use `jit` transformation and `scan` primitive to reduce compilation and execution time of the recursive ODE integration steps, which is particularly useful when running large numbers of steps (long reaction times). Models also inherit from the Equinox module abstraction [35] and are registered as PyTree containers, which facilitates the application of JAX core transformations to any SBMLTOODEJAX object.

Efficiency simulation and optimization The application of JAX core transformations, such as just-in-time compilation (`jit`), automatic vectorization (`vmap`) and automatic differentiation (`grad`), to the generated models make it very easy (and seamless) to efficiently run simulations in parallel. For instance, as shown in Figure 2, with only a few lines of Python code one can vectorize calls to model rollout and perform batched computations, which is especially efficient for large batch sizes.

```python
# vector of initial conditions
batched_y0 = ...

# batch model
batched_model = vmap(model,
                     in_axes=(None, 0),
                     out_axes=(0, 0, None))

# run simulation in batch mode
batched_ys, batched_ws, ts = batched_model(n_steps,
                                           batched_y0)
```

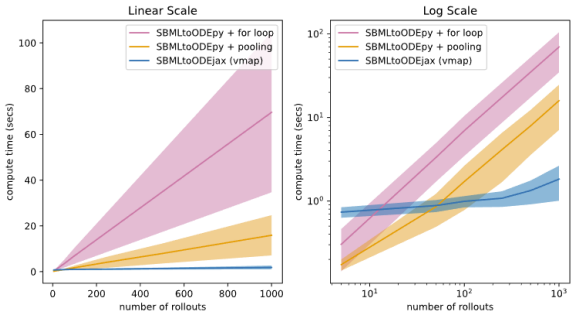

Figure 2: (left) Example code to vectorize calls to model rollout (right) Results of a (rudimentary) benchmark comparing the average simulation time of models implemented with SBMLtoODEpy versus SBMLTOODEJAX (for different number of rollouts i.e. batch size). Mean and standard-deviation results of the benchmark, run on 5 models, are displayed. For additional details on the comparison, please refer to our Benchmarking notebook.

As shown in Figure 3, SBMLTOODEJAX models can also be integrated within Optax pipelines, a gradient processing and optimization library for JAX [36], allowing to optimize model parameters and/or external interventions with stochastic gradient descent.

Altogether, the parallel execution capabilities and the differentiability of the generated models opens interesting possibilities to design and optimize intervention strategies.

```python
# Load Model
model, y0, w0, c = load_model(model_idx)

# Optax pipeline
@jit
def loss_fn(c, model):
    """loss function"""
    ys, ws, ts = model(n_steps, y0, w0, c)
    loss = jnp.sqrt(jnp.square(ys[y_indexes["Ca_Cyt"]]
                               -target_Ca_Cyt).sum())
    return loss

@jit
def make_step(c, model, opt_state):
    """update function"""
    loss, grads = value_and_grad(loss_fn)(c, model)
    updates, opt_state = optim.update(grads, opt_state)
    c = optax.apply_updates(c, updates)
    return loss, c, opt_state

n_optim_steps = 1000
optim = optax.adam(1e-3)
opt_state = optim.init(c)
train_loss = []
for optim_step_idx in range(n_optim_steps):
    loss, c, opt_state = make_step(c, model, opt_state)
    train_loss.append(loss)

# Plot train loss
plt.plot(train_loss)
plt.ylabel("training loss")
plt.xlabel("train steps")
plt.show()
```

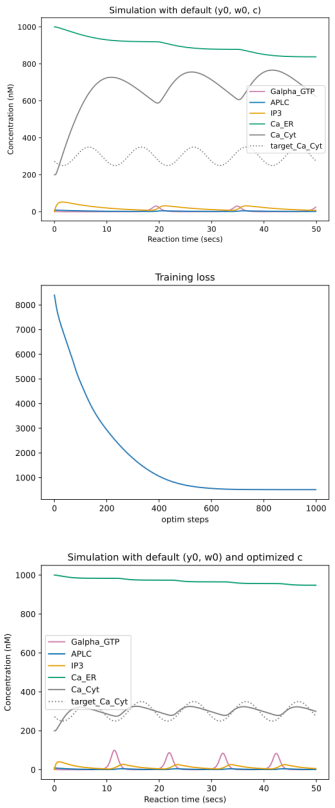

Figure 3: (left) Example of seamless integration with Optax optimization pipeline, with Adam optimizer, over the model kinematic parameters c of the biomodel #145 which models ATP-induced intracellular calcium oscillations. (right-top) Default simulation results of biomodel #145 and (arbitrary) target sine-wave pattern for Ca_Cyt concentration (shown in light gray). (right-middle) Training loss obtained after running the Optax optimization loop. (right-bottom) Simulation results obtained after optimization where we see that the obtained calcium oscillations (dark gray) align with the target one (light gray). The full example is available at our Gradient Descent tutorial.

4 Discussion

The development of SBMLTOODEJAX aims to address the need for efficient integration of Systems Biology Markup Language (SBML) models with machine learning frameworks, particularly the JAX ecosystem. This tool simplifies the utilization of SBML-based models in Python projects and machine learning pipelines, and has the potential to be reused in several contexts at the intersection of machine learning and biology.

As an illustration, SBMLTOODEJAX has been used in a recent publication presenting a novel methodology, using diversity search AI algorithms, for exploring the space of possible behaviors of gene regulatory networks, from the perspective of generic problem-solving agents navigating their environment [37]. The authors present several implications that the constructed "behavioral catalogs" could in turn have for biomedicine (control of gene expression via stimuli, not genetic rewiring) and for gene circuit engineering (optimization of network parameters to perform a desired functionality). An interactive version of their paper is available at `https://developmentalsystems.org/curious-exploration-of-grn-competencies/`.

However, SBMLTOODEJAX is still in its early phase and there are several developments that could be pursued in future work. First, it does not yet handle all possible cases of SBML files, for instance SBML models with events i.e. discrete occurrences that can trigger discontinuous changes in the model. Similarly, SBMLTOODEJAX only integrates one ODE solver for now (`jax.experimental.odeint`), but could benefit from more [38]. Finally, whereas SBMLTOOD-

EJAX is to our knowledge the first software tool enabling gradient backpropagation through the SBML model rollout, applying it in practice can be hard. GRN models are recurrent networks that are generally run for many steps, with each step calling the ODE solver. This can sometimes lead to gradient issues and long backpropagation compute times. Further optimizing the models to be efficiently differentiable might benefit their broader usage.

References

[1] H. Kitano, "Systems Biology: A Brief Overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, Mar. 2002.

[2] A. J. Singh, S. A. Ramsey, T. M. Filtz, and C. Kioussi, "Differential gene regulatory networks in development and disease," *Cellular and Molecular Life Sciences*, vol. 75, no. 6, pp. 1013–1025, Mar. 2018.

[3] G. Qin, L. Yang, Y. Ma, J. Liu, and Q. Huo, "The exploration of disease-specific gene regulatory networks in esophageal carcinoma and stomach adenocarcinoma," *BMC Bioinformatics*, vol. 20, no. 22, p. 717, Dec. 2019.

[4] H. Fazilaty, L. Rago, K. Kass Youssef, O. H. Ocaña, F. Garcia-Asencio, A. Arcas, J. Galceran, and M. A. Nieto, "A gene regulatory network to control EMT programs in development and disease," *Nature Communications*, vol. 10, no. 1, p. 5115, Nov. 2019.

[5] J. Davies and M. Levin, "Synthetic morphology via active and agential matter," Jun. 2022.

[6] S. Toda, W. L. McKeithan, T. J. Hakkinen, P. Lopez, O. D. Klein, and W. A. Lim, "Engineering synthetic morphogen systems that can program multicellular patterning," *Science*, vol. 370, no. 6514, pp. 327–331, Oct. 2020.

[7] C. Ho and L. Morsut, "Novel synthetic biology approaches for developmental systems," *Stem Cell Reports*, vol. 16, no. 5, pp. 1051–1064, May 2021.

[8] M. Glont, T. V. N. Nguyen, M. Graesslin, R. Hälke, R. Ali, J. Schramm, S. M. Wimalaratne, V. B. Kothamachu, N. Rodriguez, M. J. Swat, J. Eils, R. Eils, C. Laibe, R. S. Malik-Sheriff, V. Chelliah, N. Le Novère, and H. Hermjakob, "BioModels: Expanding horizons to include more modelling approaches and formats," *Nucleic Acids Research*, vol. 46, no. D1, pp. D1248–D1253, Jan. 2018.

[9] R. S. Malik-Sheriff, M. Glont, T. V. N. Nguyen, K. Tiwari, M. G. Roberts, A. Xavier, M. T. Vu, J. Men, M. Maire, S. Kananathan, E. L. Fairbanks, J. P. Meyer, C. Arankalle, T. M. Varusai, V. Knight-Schrijver, L. Li, C. Dueñas-Roca, G. Dass, S. M. Keating, Y. M. Park, N. Buso, N. Rodriguez, M. Hucka, and H. Hermjakob, "BioModels—15 years of sharing computational models in life science," *Nucleic Acids Research*, vol. 48, no. D1, pp. D407–D415, Jan. 2020.

[10] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and and the rest of the SBML Forum:, "The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, Mar. 2003.

[11] M. Hucka, F. T. Bergmann, C. Chaouiya, A. Dräger, S. Hoops, S. M. Keating, M. König, N. L. Novère, C. J. Myers, B. G. Olivier, S. Sahle, J. C. Schaff, R. Sheriff, L. P. Smith, D. Waltemath, D. J. Wilkinson, and F. Zhang, "The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 2 Core Release 2," *Journal of Integrative Bioinformatics*, vol. 16, no. 2, p. 20190021, Jun. 2019.

[12] G. Muzio, L. O'Bray, and K. Borgwardt, "Biological network analysis with deep learning," *Briefings in Bioinformatics*, vol. 22, no. 2, pp. 1515–1530, Mar. 2021.

[13] D. M. Camacho, K. M. Collins, R. K. Powers, J. C. Costello, and J. J. Collins, "Next-Generation Machine Learning for Biological Networks," *Cell*, vol. 173, no. 7, pp. 1581–1592, Jun. 2018.

[14] M. AlQuraishi and P. K. Sorger, "Differentiable biology: Using deep learning for biophysics-based and data-driven modeling of molecular mechanisms," *Nature Methods*, vol. 18, no. 10, pp. 1169–1180, Oct. 2021.

[15] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, "COPASI—a COmplex PAthway SImulator," *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, Dec. 2006.

[16] F. T. Bergmann, "Copasi/basico: Release 0.48," Mar. 2023.

[17] L. M. Loew and J. C. Schaff, "The Virtual Cell: A software environment for computational cell biology," *Trends in Biotechnology*, vol. 19, no. 10, pp. 401–406, Oct. 2001.

[18] B. M. Slepchenko, J. C. Schaff, I. Macara, and L. M. Loew, "Quantitative cell biology with the Virtual Cell," *Trends in Cell Biology*, vol. 13, no. 11, pp. 570–576, Nov. 2003.

[19] A. Funahashi, M. Morohashi, H. Kitano, and N. Tanimura, "CellDesigner: A process diagram editor for gene-regulatory and biochemical networks," *BIOSILICO*, vol. 1, no. 5, pp. 159–162, Nov. 2003.

[20] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano, "CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks," *Proceedings of the IEEE*, vol. 96, no. 8, pp. 1254–1265, Aug. 2008.

[21] K. Choi, J. K. Medley, M. König, K. Stocking, L. Smith, S. Gu, and H. M. Sauro, "Tellurium: An extensible python-based modeling environment for systems and synthetic biology," *Biosystems*, vol. 171, pp. 74–79, Sep. 2018.

[22] J. K. Medley, K. Choi, M. König, L. Smith, S. Gu, J. Hellerstein, S. C. Sealfon, and H. M. Sauro, "Tellurium notebooks—An environment for reproducible dynamical modeling in systems biology," *PLOS Computational Biology*, vol. 14, no. 6, p. e1006220, Jun. 2018.

[23] S. M. Ruggiero and A. N. F. Versypt, "SBMLtoODEpy: A software program for converting SBML models into ODE models in Python," *Journal of Open Source Software*, vol. 5, no. 53, p. 1643, Sep. 2019.

[24] N. Rodriguez, J.-B. Pettit, P. Dalle Pezze, L. Li, A. Henry, M. P. van Iersel, G. Jalowicki, M. Kutmon, K. N. Natarajan, D. Tolnay, M. I. Stefan, C. T. Evelo, and N. Le Novère, "The systems biology format converter," *BMC Bioinformatics*, vol. 17, no. 1, p. 154, Apr. 2016.

[25] S. Schoenholz and E. D. Cubuk, "JAX MD: A Framework for Differentiable Physics," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 11 428–11 441.

[26] E. J. Ma and A. Kummer, "Reimplementing Unirep in JAX," p. 2020.05.11.088344, May 2020.

[27] G. Carleo, K. Choo, D. Hofmann, J. E. T. Smith, T. Westerhout, F. Alet, E. J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, M. Mauri, G. Mazzola, C. B. Mendl, E. van Nieuwenburg, O. O'Reilly, H. Théveniaut, G. Torlai, F. Vicentini, and A. Wietek, "NetKet: A machine learning toolkit for many-body quantum systems," *SoftwareX*, vol. 10, p. 100311, Jul. 2019.

[28] J.-E. Campagne, F. Lanusse, J. Zuntz, A. Boucaud, S. Casas, M. Karamanis, D. Kirkby, D. Lanzieri, Y. Li, and A. Peel, "JAX-COSMO: An End-to-End Differentiable and GPU Accelerated Cosmology Library," *The Open Journal of Astrophysics*, vol. 6, p. 10.21105/astro.2302.05163, Apr. 2023.

[29] D. Häfner, R. Nuterman, and M. Jochum, "Fast, Cheap, and Turbulent—Global Ocean Modeling With GPU Acceleration in Python," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 12, p. e2021MS002717, 2021.

[30] S. Mann, E. Fadel, S. S. Schoenholz, E. D. Cubuk, S. G. Johnson, and G. Romano, "∂PV: An end-to-end differentiable solar-cell simulator," *Computer Physics Communications*, vol. 272, p. 108232, Mar. 2022.

[31] A. Stanziola, S. R. Arridge, B. T. Cox, and B. E. Treeby, "J-Wave: An open-source differentiable wave simulator," *SoftwareX*, vol. 22, p. 101338, May 2023.

[32] D. A. Bezgin, A. B. Buhendwa, and N. A. Adams, "JAX-Fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows," *Computer Physics Communications*, vol. 282, p. 108527, Jan. 2023.

[33] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: Composable transformations of Python+NumPy programs," 2018.

[34] B. N. Kholodenko, "Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades," *European Journal of Biochemistry*, vol. 267, no. 6, pp. 1583–1588, Mar. 2000.

[35] P. Kidger and C. Garcia, "Equinox: Neural networks in JAX via callable PyTrees and filtered transformations," *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[36] I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, W. Stokowiec, L. Wang, G. Zhou, and F. Viola, "The DeepMind JAX Ecosystem," 2020.

[37] M. Etcheverry, C. Moulin-Frier, P.-Y. Oudeyer, and M. Levin, "Ai-driven automated discovery tools reveal diverse behavioral competencies of biological networks," 2023.

[38] P. Städter, Y. Schälte, L. Schmiester, J. Hasenauer, and P. L. Stapor, "Benchmarking of numerical integration methods for ODE models of biological systems," *Scientific Reports*, vol. 11, no. 1, p. 2696, Jan. 2021.