The Journal of Open Source Software

ODINN.jl: Scientific machine learning glacier modelling

Jordi Bolibar[1,2¶], Facundo Sapienza[3,4], Alban Gossard[1], Mathieu le Séac'h[1], Vivek Gajadhar[2], Fabien Maussion[5,6], Bert Wouters[2], and Fernando Pérez[4]

1 Univ. Grenoble Alpes, CNRS, IRD, G-INP, Institut des Géosciences de l'Environnement, Grenoble, Franc **2** Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands **3** Department of Geophysics, Stanford University, Stanford, United States **4** Department of Statistics, University of California, Berkeley, United States **5** Bristol Glaciology Centre, School of Geographical Sciences, University of Bristol, Bristol, UK **6** Department of Atmospheric and Cryospheric Sciences, University of Innsbruck, Innsbruck, Austria ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

`ODINN.jl` is a glacier model, leveraging scientific machine learning (SciML) methods, to perform forward and reverse simulations of large-scale glacier evolution. It can simulate both surface mass balance and ice flow dynamics, through a modular architecture which enables the user to easily modify model components. For this, `ODINN.jl` is in fact an ecosystem composed of multiple packages, each one handling a specific task:

- `Sleipnir.jl`: Handles all the basic types, functions and datasets, common through the whole ecosystem.
- `Muninn.jl`: Handles surface mass balance processes, via different types of models.
- `Huginn.jl`: Handles ice flow dynamics, by solving the ice flow partial differential equations (PDEs) using numerical methods. It can accommodate multiple types of ice flow models.
- `ODINN.jl`: Acts as the interface to the whole ecosystem, and provides the necessary tools to differentiate and optimize any model component. It can be seen as the SciML layer, enabling different types of inverse methods, using hybrid models combining differential equations with data-driven models.

The ODINN ecosystem extends beyond this suite of Julia packages, by leveraging the data preprocessing tools of the Open Global Glacier Model (OGGM). We do so via an auxiliary library named `Gungnir`, which is responsible for downloading all the necessary data to force and initialize the model, such as glacier outlines from the Randolph Glacier Inventory (RGI), digital elevation models (DEMs), ice thickness observations from GlaThiDa, ice surface velocities from different studies and many different sources of climate reanalyses and projections. This implies that `ODINN.jl`, like OGGM, is virtually capable of simulating any of the 200,000 glaciers on Earth.

The Journal of Open Source Software

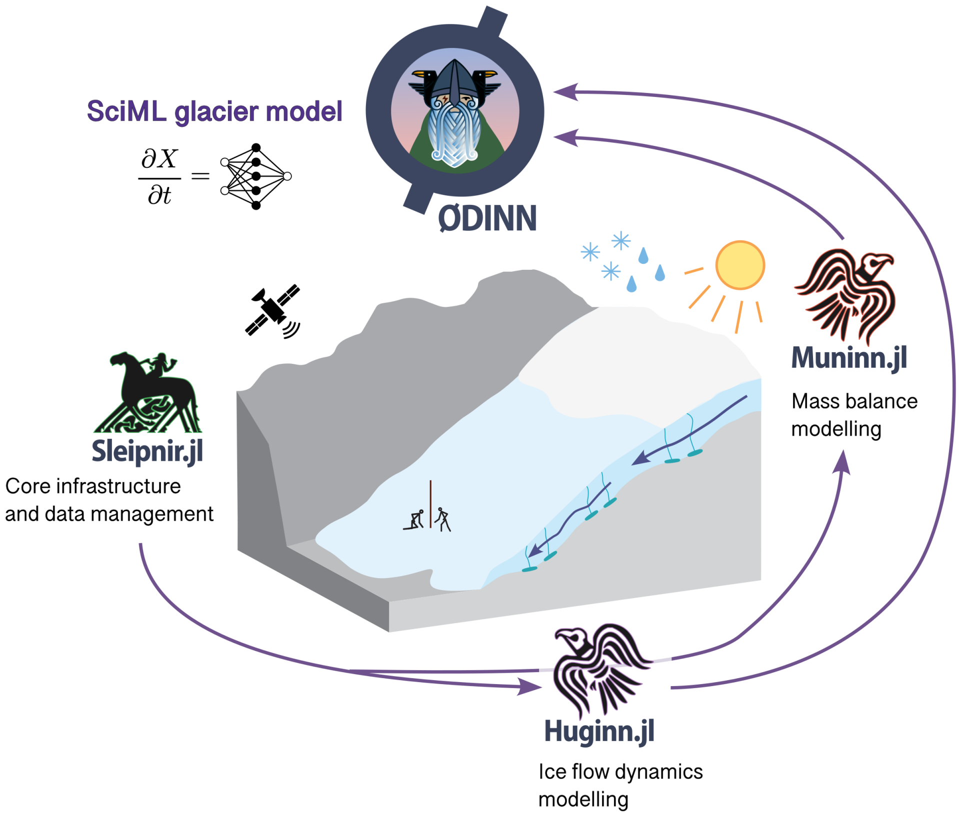

Figure 1: Figure 1: Overview of the ODINN.jl ecosystem.

`ODINN.jl` provides a high-level user-friendly interface, enabling the user to swap and replace most elements of a glacier simulation in a very modular fashion. The main elements of a simulation, such as the `Parameters`, a `Model` and a `Simulation` (e.g. a `Prediction` or a `FunctionalInversion`), are all objects that can be easily modified and combined. In a few lines of code, the user can automatically retrieve all necessary information for most glaciers on Earth, compose a `Model` based on a specific combination of surface mass balance and ice flow models, and incorporate data-driven models (e.g. a neural network) to parametrize specific physical processes of any of these components. Both forward and reverse simulations run in parallel using multiprocessing, leveraging Julia's speed and performance. GPU compatibility is still not ready, due to the difficulties of making everything compatible with automatic differentiation (AD). Nonetheless, it is planned for future versions.

The most unique aspect of `ODINN.jl` is its differentiability and capabilities of performing all sorts of different hybrid modelling. Since the whole ecosystem is differentiable, we can optimize almost any model component, providing an extremely powerful framework to tackle many scientific problems. `ODINN.jl` can optimize, separately or together, in a steady-state or transient way:

- The initial or intermediate state of glaciers (i.e. their ice thickness `H`) or the equivalent ice velocities `V[x,y]`.
- Model parameters (e.g. the ice viscosity `A` in a 2D Shallow Ice Approximation), in a gridded or scalar format. This can be done for multiple time steps where observations (e.g. ice surface velocities) are available.
- The parameters of a regressor (e.g. a neural network), used to parametrize a subpart or one or more parameters of an ice flow or surface mass balance model. This enables the exploration of empirical laws describing physical processes of glaciers.

For this, it is necessary to use reverse differentation to compute the required vector-jacobian products (VJPs). We have two strategies to achieve this: (1) manual adjoints, which have been implemented using AD via `Enzyme.jl`, as well as fully manual implementations of the discrete and continuous adjoints; and (2) automatic adjoints using `SciMLSensitivity.jl`, providing both continuous and discrete versions and available with different AD back-ends. These two approaches are complementery, with the manual adjoints being ideal for high-performance tasks, and serving as a ground truth for benchmarking and testing automatic adjoint methods from `SciMLSensitivity.jl`.

Beyond all these inverse modelling capabilities, `ODINN.jl` can also act as a more conventional forward glacier model, simulating glaciers in parallel, and easily customizing almost every possible detail of the simulation. Its high modularity, combined with the easy access to a vast array of datasets coming from OGGM, makes it very easy to run simulations, even with a simple laptop. `Huginn.jl` is responisble for the ice flow dynamics models, with an architecture capable of integrating and easily swapping various models. For now, we have implemented a 2D Shallow Ice Approximation (SIA), but in the future we plan to incorporate other models, such as the Shallow Shelf Approximation (SSA). In terms of surface mass balance, `Muninn.jl` incorporates for now simple temperature-index models. Nonetheless, the main addition of the upcoming version will be the machine learning-based models from the `MassBalanceMachine`, which will become the de-facto solution.

Statement of need

`ODINN.jl` has been designed to address the need for a glacier model which can leverage both the interpretability and established knowledge coming from the literature in the form of mechanistic models based on differential equations, with the flexibility and data-assimilation capabilities of data-driven models. The combination of these two paradigms enables a targetted approach to inverse methods for learning parametrizations of glacier physical processes, learning only the unknown physics and keeping a reliable structure in the dynamics in the form of a differential equation. While purely mechanistic and data-driven modelling approaches exist in glaciology, there is a need for flexible models which can leverage existing widely available observations at the glacier surface, to simulate complex physical processes of glaciers, such as basal sliding, creep or calving. Existing laws do not necessarily map available observations with these physical processes, difficulting the finding and calibration of parametrizations and laws. Approaches based on functional inversions and differentiable programming offer the needed flexilibity to derive new empirical laws based on carefully-chosen input proxies, which can help to test hypothesis of what can constitute and drive new parametrizations.

At the same time, a good representation of this complex and poorly represented physical processes is key to accurate predictions of glacier evolution, crucial for their impact to both freshwater resources and sea-level rise. Therefore, with `ODINN.jl`, we provide a unified modelling ecosystem, capable of both flexibile and advance inverse methods for model calibration, as well as efficient and modular methods for forward simulations for large-scale glacier modelling.

Acknowledgements

We acknowledge the help of Chris Rackauckas for the debugging and discussion of issues related to the SciML Julia ecosystem, Redouane Lguensat for scientific discussions on the first prototype of the model. JB acknowledges financial support from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek, Stichting voor de Technische Wetenschappen (Vidi grant 016.Vidi.171.063). FS acknowledges funding from the National Science Foundation (EarthCube programme under awards 1928406 and 1928374).

References