The Journal of Open Source Software

# SICOPOLIS-AD v2: tangent linear and adjoint modeling framework for ice sheet modeling enabled by automatic differentiation tool Tapenade

**Shreyas Sunil Gaikwad**[1¶], **Laurent Hascoet**[2], **Sri Hari Krishna Narayanan**[3], **Liz Curry-Logan**[1], **Ralf Greve**[4,5], **and Patrick Heimbach**[1,6,7]

**1** Oden Institute for Computational Engineering and Sciences, University of Texas at Austin, USA **2** Institut National de Recherche en Informatique et Automatique, France **3** Mathematics and Computer Science Division, Argonne National Laboratory, USA **4** Institute of Low Temperature Science, Hokkaido University, Japan **5** Arctic Research Center, Hokkaido University, Japan **6** Jackson School of Geosciences, University of Texas at Austin, USA **7** Institute for Geophysics, University of Texas at Austin, USA ¶ Corresponding author

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Open Journals

**Reviewers:**

- @openjournals

**Submitted:** 01 January 1970
**Published:** unpublished



## Summary

We present a new framework for generating derivative code, i.e., tangent linear, adjoint, or Hessian models, of the open-source SImulation COde for POLythermal Ice Sheets (SICOPOLIS). These derivative operators are powerful computational engines to efficiently compute comprehensive gradients or sensitivities of scalar-valued model output, including least-squares model-data misfits or important quantities of interest, to high-dimensional model inputs (such as model initial conditions, parameter fields, or boundary conditions). The new version 2 (SICOPOLIS-AD v2) framework is based on the source-to-source automatic differentiation (AD) tool Tapenade which has recently been open-sourced. The switch from a previous AD tool (OpenAD) used in SICOPOLIS-AD version 1 to Tapenade overcomes several limitations outlined here. In addition, we provide several convenient support tools and workflows for code generation, validation, and data analysis. They include GitLab's Continuous Integration feature for code validation, python scripts that support model configuration, invoking AD, code compilation, and model execution. A new documentation and several tutorials help users to get started. The framework is integrated with the SICOPOLIS model's main trunk and is freely available along with Tapenade.

## Statement of need

Ice sheets are grounded ice masses of continental size, with an area greater than $50,000\text{km}^2$. The two contemporary ice sheets, Greenland and Antarctica, are dynamic entities whose evolution is governed by a set of nonlinear partial differential equations (PDEs) that describe the conservation of mass, momentum, and energy, as well as constitutive laws for the material properties of ice. In general, these equations cannot be solved analytically but must be solved numerically. Ice sheet models are a computer representation of these PDEs. They require as input parameters (i) initial conditions of the state of the ice sheet, (ii) surface boundary conditions, such as temperature and precipitation, (iii) basal boundary conditions, such as bed topography and geothermal flux, and (iv) model parametric uncertainties, such as basal friction or flow law parameter. Despite recent advances in numerical modeling of ice sheets, the effects of ad-hoc initialization and the uncertainties in these independent input parameters propagate to the model's output, in particular to quantities of interest (QoI), such as future projections of sea-level rise, which is of economic and societal importance (Schinko et al., 2020).

Understanding a system's response to perturbations in its input is at the heart of scientific analysis and understanding. It is thus desirable to evaluate the sensitivities of our QoI to these independent input variables. In the context of ice sheet modeling, sensitivities of model-data misfits or other quantities of interest are a key ingredient for performing model calibration, state estimation, or uncertainty quantification, all of which ultimately guide the improvement of model simulations through PDE-constrained gradient-based optimization. Posterior uncertainties around the optimal set of input variables can be quantified by characterizing them as probability density functions. An approximate approach to efficiently completing such a task requires the set of dominant eigenvalues and eigenvectors of the prior-preconditioned linearized misfit Hessian matrix. Both the gradient and the eigen-characteristics of the Hessian can be evaluated in a matrix-free manner using automatic differentiation (AD) generated adjoint and tangent linear codes.

The conventional approach for calculating system sensitivities consists of perturbing each component (e.g., spatially discretized elements of a field) of each input parameter one at a time and evaluating the change in the QoI. For a high dimensional parameter space, the cost of this brute force method is prohibitive, since it requires $\mathcal{O}(N)$ nonlinear forward model evaluations, where $N$ is the dimension of the input parameter space. A more efficient method is to develop an adjoint model, which flows in the reverse sense to our forward model and evaluates the sensitivities for the entire parameter space in a single model run. While the code for the adjoint model can be written manually, this approach becomes cumbersome for a large system of equations. Furthermore, this method suffers from the limitation of having to update the adjoint code manually every time the forward code is updated, which is error-prone and time-consuming.

Instead, `SICOPOLIS-AD v2` leverages the recently open-sourced automatic differentiation (AD) tool `Tapenade` (Hascoet & Pascual, 2013) to generate code for the adjoint model of the open-source ice sheet model, SImulation COde for POLythermal Ice Sheets, or `SICOPOLIS` (Greve, 1997; Greve et al., 2011; Greve & Blatter, 2009). Sensitivities can be calculated using a single forward and adjoint model evaluation, instead of the $\mathcal{O}(N)$ forward model evaluations. Empirically, one adjoint model evaluation is about 5-10 times as expensive as a forward model run (this number depends on the specific checkpointing scheme employed to recompute variables required to evaluate nonlinear or control flow expressions). Thus, the adjoint computation is highly efficient for calculating sensitivities when $N$ is large (in our case, $N \sim 10^4 - 10^5$).

Finally, the functionality to generate a tangent linear version of the forward model is also included, which was not available in `SICOPOLIS-AD v1`. This is valuable for uncertainty quantification (UQ) of the inferred parameters, as well as uncertainty propagation to QoIs. It can also be used to verify the results of the adjoint model.

Let us assume a set of independent parameters $\mathbf{x} \in \mathbb{R}^N$ and dependent scalar-valued output QoI $\mathcal{J} \in \mathbb{R}$ (also known as the cost function or objective function), which depends on another variable $\mathbf{y} \in \mathbb{R}^M$. Mathematically, $\mathcal{J} = \mathcal{J}(\mathbf{y})$, where $\mathbf{y} = A(\mathbf{x})$. Here, $A$ is our nonlinear forward model, represented by the `SICOPOLIS` code. The tangent linear model is given by the Jacobian matrix of operator $A$, $\mathbf{B}(\mathbf{x})$, such that $d\mathbf{y} = \mathbf{B}(\mathbf{x})\, d\mathbf{x}$. This matrix is implicitly represented by the `Tapenade` generated tangent linear code acting on vector $d\mathbf{x}$. The sensitivity of $\mathcal{J}(\mathbf{y}(\mathbf{x}))$ to $\mathbf{x}$ can be shown to be $\nabla_{\mathbf{x}}\mathcal{J}^T = \mathbf{B}^T(\mathbf{x})\, \nabla_{\mathbf{y}}\mathcal{J}^T$ (for example Errico (1997)). Here, $\mathbf{B}^T(\mathbf{x})$ is the linear adjoint operator, whose action is implicitly represented by the `Tapenade` generated adjoint code. This is explained in greater detail in Tutorial 1.

## Target Audience

This package is intended as a resource that enables sensitivity analysis, model calibration, and uncertainty quantification of a continent-scale ice sheet model. Since the numerical discretization scheme in SICOPOLIS is finite differences and it employs the shallow ice

approximation, it is relatively less computationally expensive than the models solving the full Stokes equations using the modern finite element methods. This makes it suitable for simulations over long time scales, i.e. paleoclimate simulations. Its coding structure is modular and amenable to AD source transformation. Thus, our package is also intended to serve as a guide for future work in the application of open-source AD tools for physics-based simulation codes written in `Fortran`.

## Features

Source transformation AD tools such as the commercial tool `TAF` and the open-source tool `OpenAD` have been used previously with `SICOPOLIS` (Heimbach & Bugnion, 2009; Logan et al., 2020). `OpenAD` is no longer actively developed because it is based on the `Open64` compiler which ceased development in 2011. The differentiation of `SICOPOLIS`, therefore, must be performed using a different tool. Compared to `OpenAD`, the `Tapenade` enabled implementation has the following advantages:

- It is up-to-date with the latest `SICOPOLIS` code.
- The AD tool `Tapenade` is open-source and actively maintained.
- A new tangent linear code generation capability is introduced.
- The AD-generated codes can accept inputs in the NetCDF format.
- The external Library of Iterative Solvers for Linear Systems (LIS), its tangent linear code, and adjoint code are correctly incorporated which can improve the simulation of Antarctic ice shelves and Greenland outlet glaciers.
- Gitlab-CI, a customized Docker, and the pytest framework are leveraged for Continuous Integration (CI) to track changes in the trunk that change the output or "break" the AD-based code generation.
- The entire code is parsed by `Tapenade`, including the initialization subroutines, thus preventing cumbersome manual maintenance of subroutines to initialize the adjoint runs.
- Convenient and easy-to-use Python scripts are provided for quick setup of the compilation, I/O, and execution processes based on the metadata the user provides in a `JSON` file.
- The entire setup is well-documented, along with tutorials.
  - Tutorial 1 is an exercise in writing an adjoint code by hand, in Python. This can be helpful for users looking to delve deeper into AD-generated codes and for debugging purposes.
  - Tutorial 2 describes the use of Tapenade to get the sensitivities in the case of a simple, but realistic mountain glacier model, based on (**Oerlemans1981?**) (see also Van der Veen (2013)). It also explains how the user can get a smaller memory footprint, which is beneficial for very large systems, by using binomial checkpointing (Griewank, 1992). A smaller memory footprint, however, comes at a higher computation cost.
  - Tutorial 3 is a simple validation exercise for the adjoint and tangent linear values using finite differences computed with SICOPOLIS. We use the header file `v5_grl16_bm4_ss25ka`, which is one of the regression tests provided with the standard `SICOPOLIS` distribution. It describes a steady state run for Greenland Ice Sheet with modern climate conditions. We evaluate the sensitivity of the total ice volume to time-invariant geothermal flux. This is also discussed in the example below.

## Software requirements and external usage

SICOPOLIS-AD v2 is built on top of the SICOPOLIS ice sheet model, an open-source code written in Fortran. It uses Tapenade, a recently open-sourced AD tool, to differentiate this code. Therefore, all the prerequisites of using SICOPOLIS and Tapenade need to be satisfied, and both need to be installed correctly. A Python installation is needed to use the convenient automation functions in `test_ad/tapenade_config.py`.

## Implementation

The code for SICOPOLIS-AD v2 is kept mostly independent from the base SICOPOLIS code, allowing non-AD users to avoid it completely. All of the AD-related support routines and data files can be found in `src/subroutines/tapenade`. Similarly, all utilities and testing files for AD simulations are stored in the `test_ad` directory. A separate Makefile is provided for AD purposes - `src/MakefileTapenade`.

SICOPOLIS-AD v2 can be run directly by interacting with the Makefile and the Fortran code. The user has to prepare a suitable header file for the base SICOPOLIS code and add a few more preprocessing directives to run the adjoint code with the same header file. This header file, along with a set of dependent and independent variables, is given to the Makefile. The Makefile executes the workflow for differentiating and compiling the code depending on the mode selected by the user (tangent linear, adjoint, finite differences, code coverage evaluation). The user must then insert the I/O statements in the Tapenade-generated code depending on what they wish to analyze. This is followed by recompilation and execution of the code.

Alternatively, the user can use the functions in `test_ad/tapenade_config.py` to automatically customize the setup. These functions are written for automated sensitivity studies. They can be easily modified to serve other purposes such as model calibration and UQ.

## Example

To illustrate the use of our tool using a simple example, we use the header file `v5_grl16_bm4_ss25ka` provided as a reference template in SICOPOLIS. It describes a steady state run for Greenland Ice Sheet with modern climate conditions. We shorten the total integration time to 100 simulated years to keep the computational cost of the tangent linear and finite differences reasonable. Our QoI (i.e., dependent variable) is the total volume of the ice sheet at the end of the run (`fc`). The sensitivity is evaluated with respect to the geothermal heat flux, `q_geo` (independent variable), a 19,186-dimensional field. The results are shown in Figure 1.

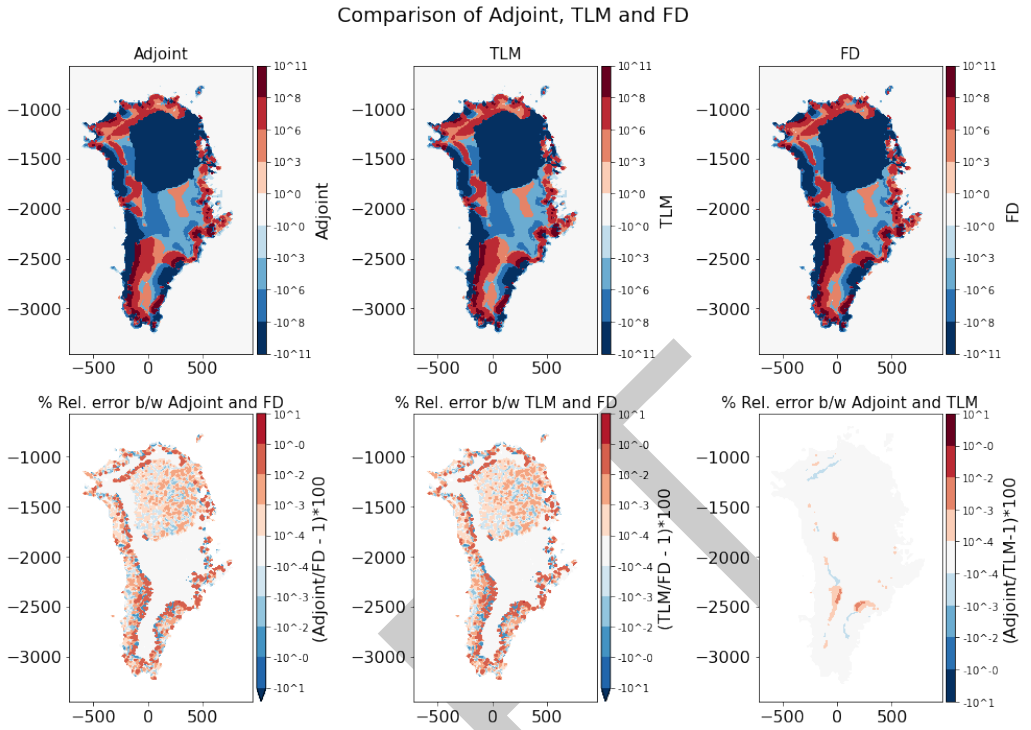


**Figure 1:** Validation exercise for adjoint (ADM) and tangent linear (TLM) models using the finite differences (FD) results for the sensitivity of `fc` with respect to `q_geo`. The upper row shows the sensitivities computed using the adjoint model (reverse-mode AD), tangent linear (forward-mode AD), and finite differences, respectively. The bottom run illustrates the relative error between (ADM, FD), (TLM, FD), and (ADM, TLM) respectively. For the bottom row, note that the values of relative error are only shown for points where the value of the gradient is "significant", i.e. within 4 orders of magnitude of the maximum absolute value of the gradient.

The results show good agreement between all three modes used to evaluate this sensitivity. The error is less than $2\%$ between AD-generated (adjoint/tangent linear codes) and finite differences at all points with "significant" gradient values, i.e. within 4 orders of magnitude of the maximum absolute value of the gradient. The relative error between the AD-generated adjoint and tangent linear models is less than $0.0025\%$ at all points with values within 4 orders of magnitude of the maximum absolute value of the gradient. However, the adjoint model is much faster than the other two, as shown in Table 1, because the number of evaluations of the tangent linear model and finite differences scales linearly with the parameter dimension ($\sim\mathcal{O}(N)$), unlike the adjoint. Thus, the discrepancy will be even larger if a finer mesh is used.

**Table 1:** Comparison of the time taken by various methods for gradient calculation to evaluate the gradient for a scalar objective function with respect to a 19,186-dimensional 2D field (16 km mesh) in a typical SICOPOLIS run. Model initialization and I/O times are not included. The runs are performed on Intel Xeon CPU E5-2695 v3 nodes (2.30 GHz clock rate, 35.84 MB L3 cache, 63.3 GB memory).

| Gradient calculation method | Time (in seconds) for 16 km mesh |
|---|---|
| Finite Differences | $1.645 \times 10^5$ |
| Tangent Linear Model | $2.140 \times 10^4$ |
| Adjoint Model | $2.113 \times 10^1$ |

The Journal of Open Source Software

## Acknowledgements

This work was supported in part by the Applied Mathematics activity within the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program, under contract number DE-AC02-06CH11357, and by National Science Foundation OPP/P2C2 grant #1903596.

## Copyright



## References

Errico, R. M. (1997). What is an adjoint model? *Bulletin of the American Meteorological Society*, *78*(11), 2577–2592. https://doi.org/10.1175/1520-0477(1997)078%3C2577:WIAAM%3E2.0.CO;2

Greve, R. (1997). Application of a polythermal three-dimensional ice sheet model to the greenland ice sheet: Response to steady-state and transient climate scenarios. *Journal of Climate*, *10*(5), 901–918. https://doi.org/10.1175/1520-0442(1997)010%3C0901:AOAPTD%3E2.0.CO;2

Greve, R., & Blatter, H. (2009). Dynamics of ice sheets and glaciers. *Dynamics of Ice Sheets and Glaciers by Ralf Greve*. https://doi.org/10.1007/978-3-642-03415-2

Greve, R., Saito, F., & Abe-Ouchi, A. (2011). Initial results of the SeaRISE numerical experiments with the models SICOPOLIS and IcIES for the greenland ice sheet. *Annals of Glaciology*, *52*(58), 23–30. https://doi.org/10.3189/172756411797252068

Griewank, A. (1992). Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, *1*(1), 35–54. https://doi.org/10.1080/10556789208805505

Hascoet, L., & Pascual, V. (2013). The Tapenade Automatic Differentiation tool: principles, model, and specification. *ACM Transactions on Mathematical Software*, *39*(3). https://doi.org/10.1145/2450153.2450158

Heimbach, P., & Bugnion, V. (2009). Greenland ice-sheet volume sensitivity to basal, surface and initial conditions derived from an adjoint model. *Annals of Glaciology*, *50*(52), 67–80. https://doi.org/10.3189/172756409789624256

Logan, L. C., Narayanan, S. H. K., Greve, R., & Heimbach, P. (2020). SICOPOLIS-AD v1: An open-source adjoint modeling framework for ice sheet simulation enabled by the algorithmic differentiation tool OpenAD. *Geoscientific Model Development*, *13*(4), 1845–1864. https://doi.org/10.5194/gmd-13-1845-2020

Schinko, T., Drouet, L., Vrontisi, Z., Hof, A., Hinkel, J., Mochizuki, J., Bosetti, V., Fragkiadakis, K., Vuuren, D. van, & Lincke, D. (2020). Economy-wide effects of coastal flooding due to sea level rise: A multi-model simultaneous treatment of mitigation, adaptation,

and residual impacts. *Environmental Research Communications*, *2*(1), 015002. https://doi.org/10.1088/2515-7620/ab6368

Van der Veen, C. J. (2013). *Fundamentals of glacier dynamics*. CRC press. https://doi.org/10.3189/2014JoG13J214