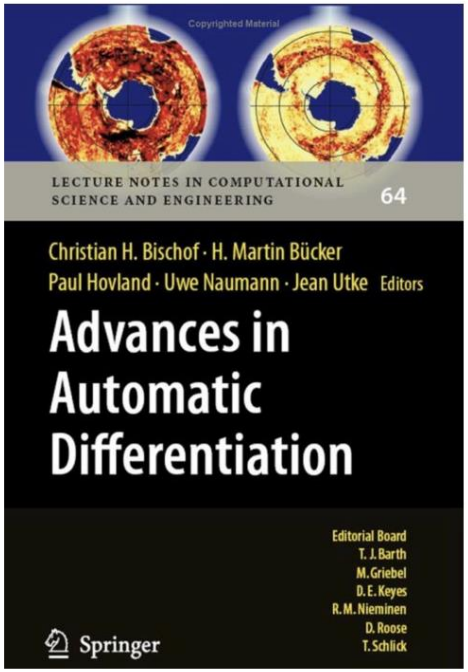
MITgcm and Tapenade

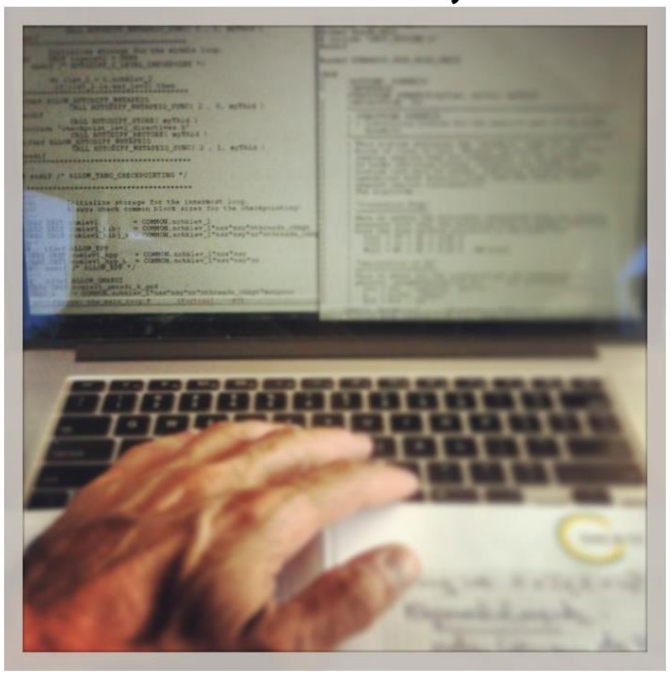
Shreyas Gaikwad, Sri Hari Krishna Narayanan, Laurent Hascoet,
Jean-Michel Campin, Helen Pillar, An Nguyen,
Jan Huckelheim, Paul Hovland, Patrick Heimbach

Why use Algorithmic Differentiation (AD)?

Generating and maintaining the adjoint of a state-of-the-art ocean GCM



hand-written adjoint



Automatic Differentiation

Giering & Kaminski (1998); Marotzke et al. (1999); Heimbach et al. (2005); Utke et al. (2007); Griewank & Walther (2008)

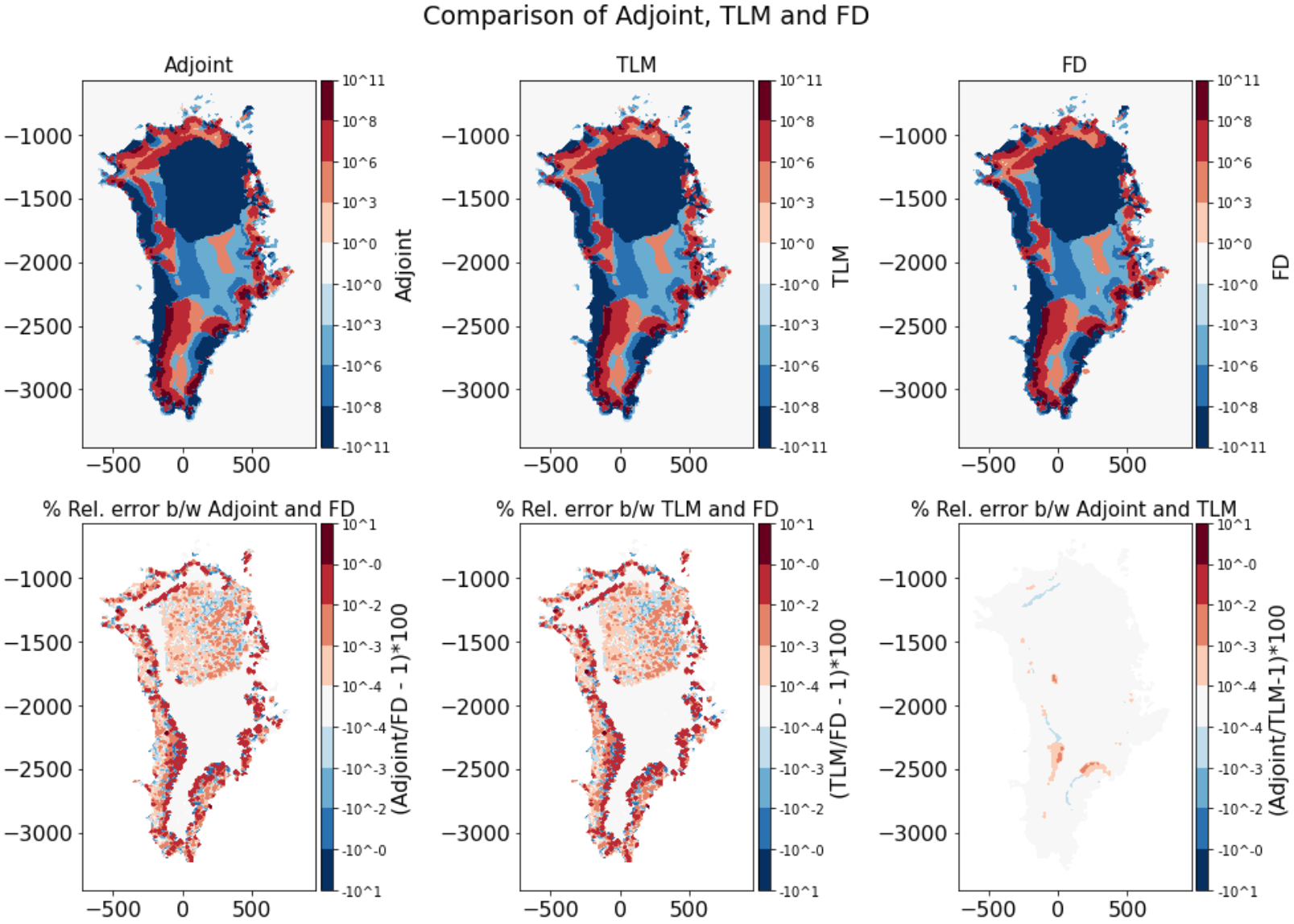
Advantages of open-source Algorithmic Differentiation (AD)

- Free of cost
- Science accessible to more users
- Adjoint code is readable and easier to modify/debug and test
- Access to codebase

Introduction to Tapenade

- An Open-Source Automatic Differentiation (AD) Engine developed at Inria, France
- Can differentiate Fortran77, Fortran90, Fortran95, or C codes
- Partial extensions to MPI, OpenMP, CUDA
- Previous applications -
  - Global sea-ice model CICE
  - Adjoinable Land Ice Flow model (ALIF), C clone of C++ model ISSM
  - Adjoint CFD code development (Mike Giles at Oxford)
  - River Hydraulics model, Optimal Design of supersonic planes, etc.
- In ML context - this is just like autograd, PyTorch, Tensorflow, etc.

Our first use - SICOPOLIS-AD v2

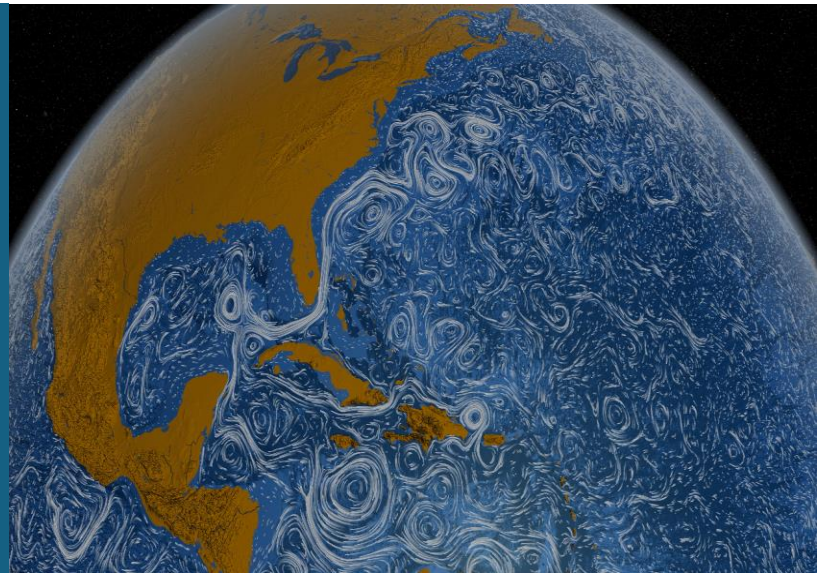


Adjoint/forward slowdown = 5.18022

Figure from Gaikwad et. al (2023), *Journal of Open Source Software*

MITGCM-AD V2:

OPEN SOURCE
INVERSE MODELING FRAMEWORK
FOR THE ATMOSPHERE & OCEANS
USING THE AD TOOL TAPENADE



Current capabilities with Tapenade

11 verification experiments

tutorial_tracer_adjsens	OpenAD
tutorial_global_oce_biogeo	global_with_exf
isomip	halfpipe_streamice
global_ocean.cs32x15	lab_sea
tutorial_barotropic_gyre	tutorial_baroclinic_gyre
global_oce_biogeo_bling	

24 packages*

exch2	cd_code	kpp	gfd	streamice	ptracers
gmredi	monitor	tapenade	exf	shelfice	thsice
cal	cost	ctrl	seaice	dic	grdchk
diagnostics	gchem	ggl90	ecco	profiles	mnc

*Not claiming that the whole package is compatible, but some measure of basic compatibility

Ease of use with Tapenade

- Setup time **< 1 hour (maybe not for MacOS users)**
- Code and documentation merged to master branch **(c68q)**
- Changes users make in their workflow are in red.
  - Mostly adding a flag and changing a few paths and make targets

```
$ make CLEAN

$ ../../../tools/genmake2 -tap -of ../../../tools/build_options/linux_amd64_ifort -mods
../code_tap

$ make depend
$ make -j 8 tap_adj

$ cd ../run
$ rm -r *
$ ln -s ../input_tap/* .
$ ../input_tap/prepare_run

$ ln -s ../build/mitgcmuv_tap_adj .
$ ./mitgcmuv_tap_adj > output_tap_adj.txt 2>&1
```

Tapenade vs TAF – tutorial_global_oce_biogeo

- J -> globally integrated air-sea flux of CO_2 on the final day of the integration.
- Control -> initial SST (theta).
- The relative difference between TAF and Tapenade results is negligible.

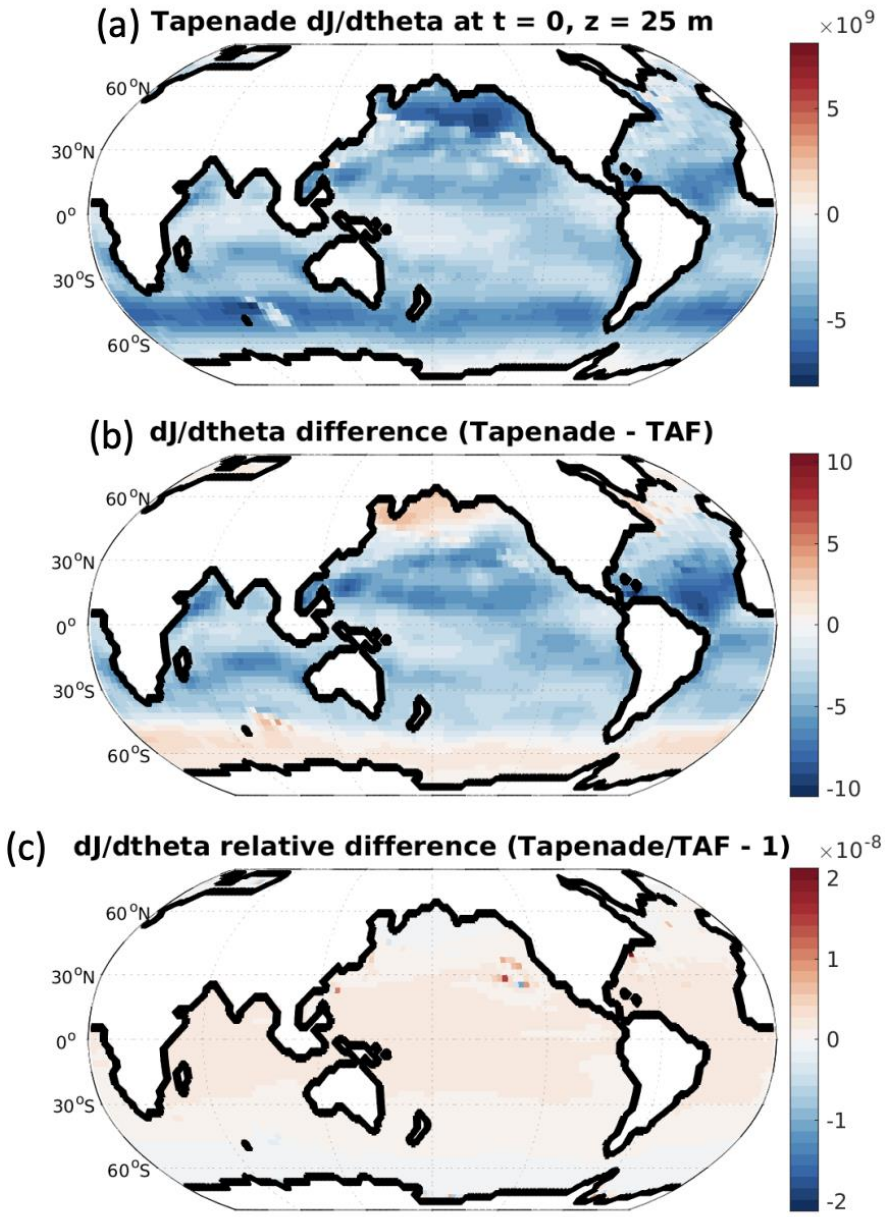


Figure from Gaikwad et. al (2024), *Submitted to JLESC-FGCS,*

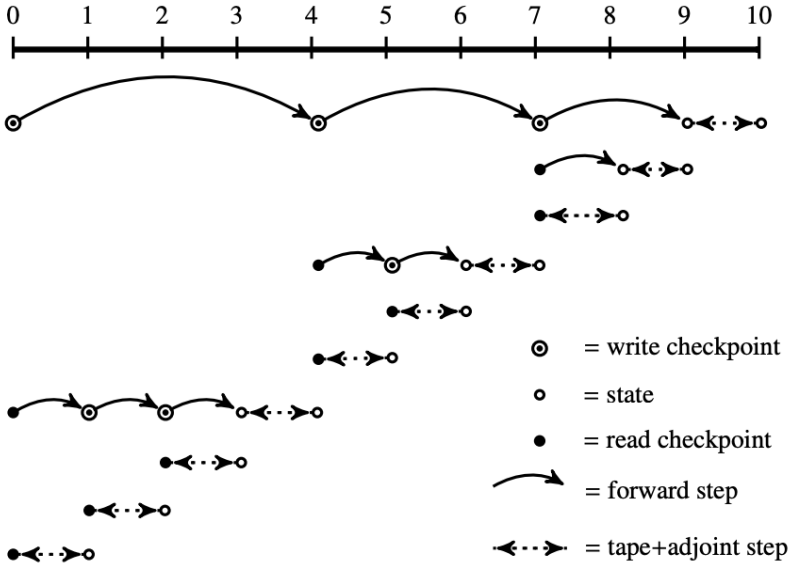
ArXiv preprint available

Tapenade vs TAF - Default

- TAF is by default recompute-all.
- TAF achieves its performance through targeted insertion of directives for storing (as opposed to recomputing) required variables.
- MITgcm's dynamical core has on the order of 350 such STORE directives, with another 600 directives in the different model packages. This results in highly tuned, albeit labor-intensive adjoint code performance tuning.
- Missing store directives can also lead to buggy code.
- Tapenade is by default store-all.
- This simplifies the implementation of efficient, AD-compatible code with Tapenade.
- **No special tuning has so far been performed for the Tapenade-generated adjoint code, except some preliminary testing.**

Tapenade vs TAF - Checkpointing

- Typical applications of the MITgcm to interrogate seasonal to multidecadal ocean variability will consist of $O(10^4)$ timesteps, each requiring $O(10^8)$ bytes
- Prohibitively large to be held in memory
- Checkpointing enables long time integrations of the model
- Checkpointing offers a trade-off b/w the recomputation of states and their storage
- TAF uses static 3-level checkpointing
- Tapenade instead implements built-in binomial checkpointing
  - optimal in the number of recomputations
  - No user effort required



Theorem 1. *Let c be the number of available checkpoints and $l = N_t$ the number of time steps. The minimal number of time steps evaluated during the adjoint calculation given by*

$$t(l,c) = \min_{1\le\hat{l}<l}\{\hat{l} + t(\hat{l},c) + t(l-\hat{l},c-1)\}\ \forall l>1, c>1,$$

$$t(1,c) = 0\ \ \forall c>0, \qquad t(l,1) = \frac{(l-1)l}{2}\ \ \forall l>0$$

has the explicit solution

$$t(l,c) = rl - \beta(c+1,r-1), \tag{8}$$

where $\beta(c,r) = \binom{c+r}{c}$ and the repetition number r is the unique integer, such that

$$\beta(c,r-1) < l \le \beta(c,r).$$

Tapenade vs TAF – Timing analysis

Table 3: Timing comparison for Tapenade (TAP) against TAF. Time in seconds. 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). Checkpointing intervals were optimized for the TAF runs to hold the maximum number of timesteps in memory; multi-level checkpointing was required only for `global_ocean.cs32x15`.

Experiment	Forward	TAF	TAP	TAP/TAF
global_with_exf	4.0	15.8	35.4	2.2
global_ocean.cs32x15	30.7	173.4	1112.2	6.4
global_oce_biogeo	26.4	53.4	165.0	3.1

Tapenade seems to be 2-7 times slower than TAF.

Tapenade has not been optimized for performance yet.

Table from Gaikwad et. al (2024), *Submitted to JLESC-FGCS,*

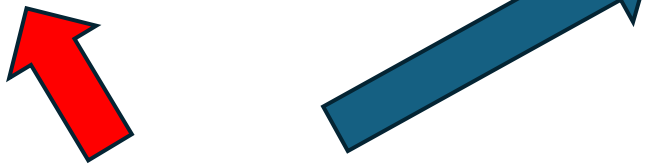
ArXiv preprint available

Tapenade vs TAF – Timing analysis

But some exciting new results

- Dan Goldberg's halfpipe-streamice with special differentiation of fixed-point loops.
- Adj_noOptim: No binomial checkpointing, default Tapenade behavior
- Adj_optim: No binomial checkpointing, no deep-stack checkpointing either

Setup	Stack (MB)	Time	Adj/forward slowdown
Forward	-	22.12	1
Adj_noOptim	187	85.04	3.84
Adj_optim	881	43.52	1.97



50% time savings with even more progress margin

Also note the increased stack size (time-memory tradeoff)

Preliminary conclusions and future outlook

- Tapenade is easy to use, actively maintained, and free
- Tapenade can handle forward AD mode, unlike OpenAD
- Tapenade might be slower, but that's because we have put no effort yet into optimizing for the speed. That is changing now.
- List of future updates documented in **Issue #735**
  - Further polishing of tapenadocker needed for use on MacOS
  - Better compatibility with the diagnostics package
  - Better compatibility with the ECCO package
  - Handling some NaNs floating around and out of bound arrays (seem benign)
  - …..

We need more people to use Tapenade for further improvement!

Thank you! Questions?

Find more details here -

- https://arxiv.org/abs/2401.11952
- MITgcm Documentation section 7.6
- Email - shreyas.gaikwad@utexas.edu