Algorithmic/automatic differentiation (AD) tools

Patrick Heimbach

MIT, EAPS, Cambridge, MA

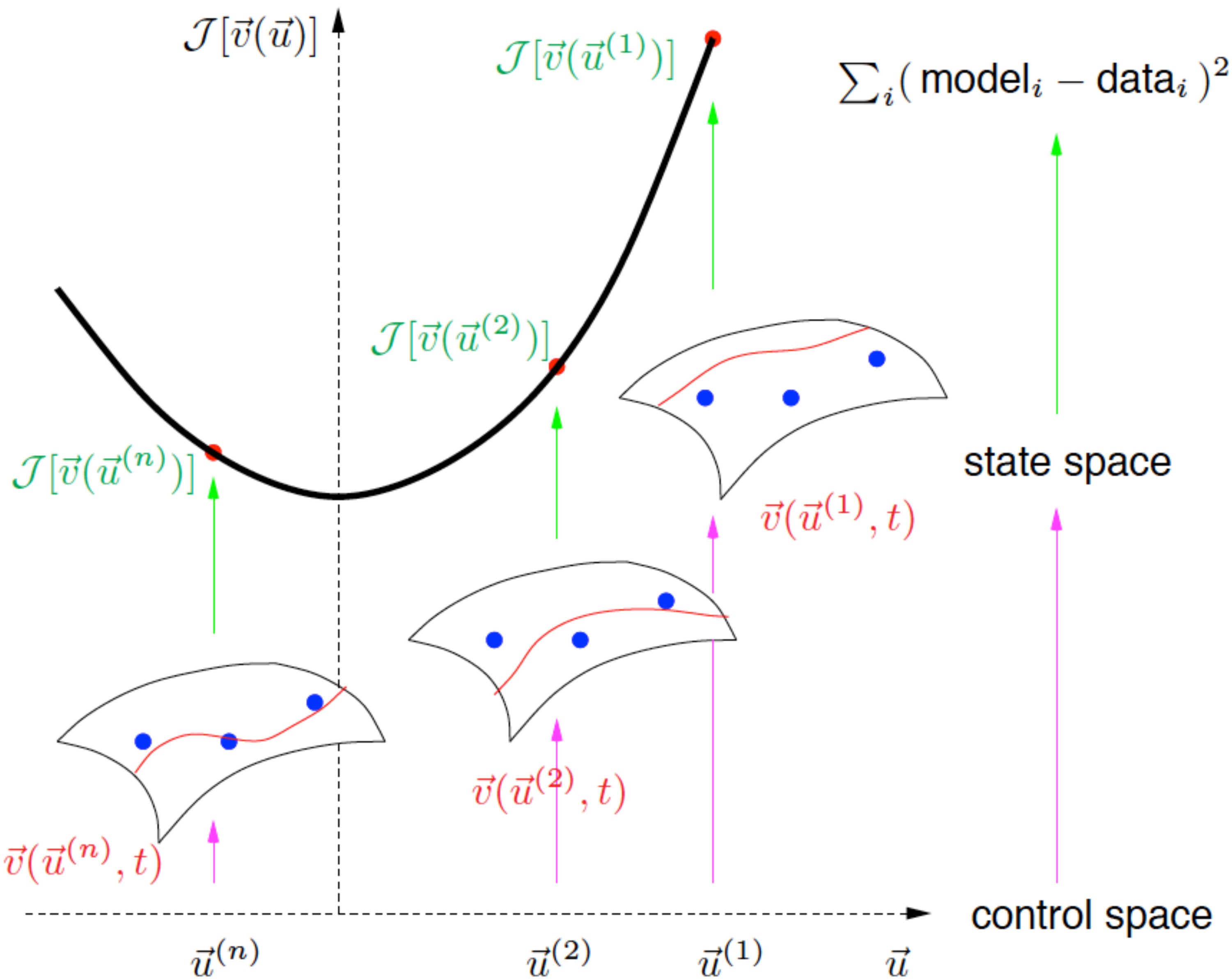
Outline

1. Why should we / do we care?

2. Tools & challenges

Why gradients/adjoints are good for you?

Data assimilation / state & parameter estimation



Why gradients/adjoints are good for you?

Comprehensive sensitivity studies

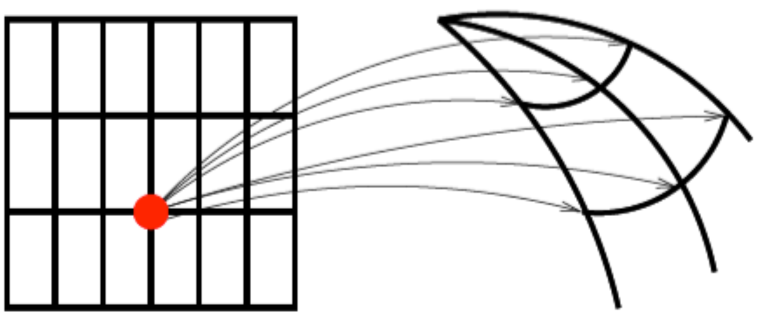
- **Finite difference approach:**
  - Take “guessed” anomaly (e.g. **SST**) and determine its impact on model output (**ice export**)
  - Perturb each input element (**SST**(i,j)) to determine its impact on output (**ice export**).

  Impact of ***one input*** on ***all outputs***

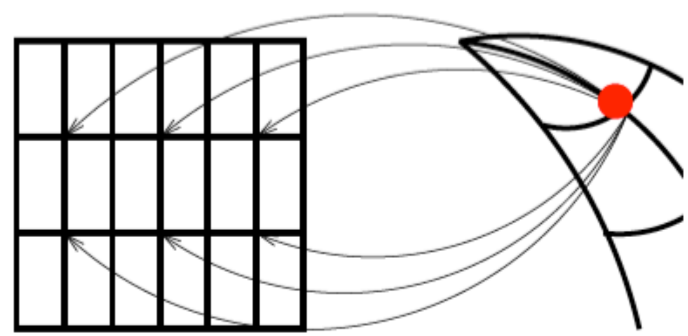
- **Reverse/adjoint approach:**
  - Calculates “full” sensitivity field $\frac{\partial \textbf{ice export}}{\partial \textbf{SST}(x,y,t)}$
  - Approach: Let $\mathcal{J} = \textbf{export}$, $\vec{u} = \textbf{SST}(i,j)$

    $$\longrightarrow \boxed{\vec{\nabla}_u \mathcal{J}(\vec{u})} = \frac{\partial \textbf{ice export}}{\partial \textbf{SST}(x,y,t)}$$

  Sensitivity of ***one output*** to ***all inputs***



forward or finite difference approach



adjoint approach

Why gradients/adjoints are good for you?
Non-normal transient amplification & predictability

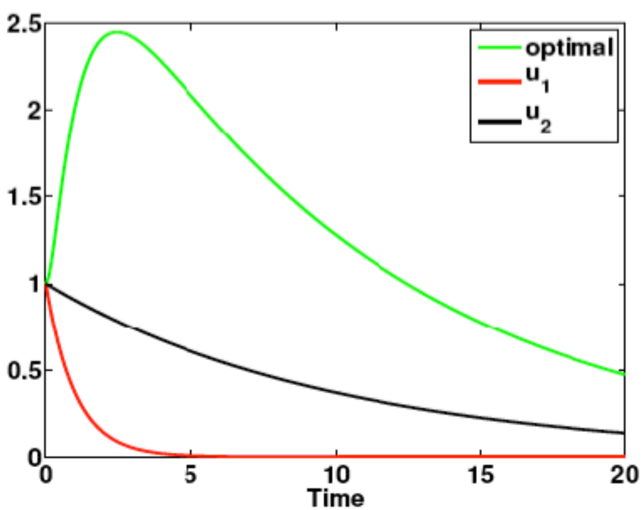
- Consider stable linear system

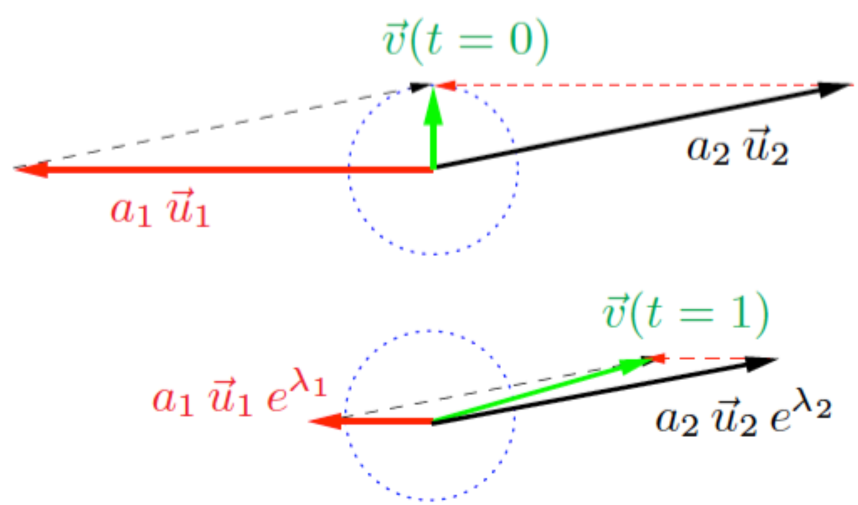
$$\frac{\mathrm{d}\,\vec{v}(t)}{\mathrm{d}\,t} = M\,\vec{v}(t), \quad \vec{v}(t) \to 0 \text{ for } t \to \infty$$

- If M is non-normal, $M \cdot M^T \neq M^T \cdot M$, non-orthogonal eigenvectors:

$$\vec{v}(t) = a_1\,\vec{u_1}\,e^{\lambda_1 t} + a_2\,\vec{u_2}\,e^{\lambda_2 t}$$

- If decay timescales very different, i.e. $\lambda_1 << \lambda_2 < 0$, then
  - $a_1\,\vec{u_1}\,e^{\lambda_1 t}$ decays quickly,removing partial cancelation of EV's
  - causing transient amplification for $t \approx 1$
  - leaving mostly $\vec{v}(t) \approx a_2\,\vec{u_2}\,e^{\lambda_2 t} \to 0$ for $t \to \infty$.





Why gradients/adjoints are good for you?
Formal uncertainty characterization & quantification

- Consider linear approx. of cost function

$$\mathcal{J}(\vec{u}) = \frac{1}{2}\left(\mathcal{M}(\vec{u}) - \vec{d}\right)^T W \left(\mathcal{M}(\vec{u}) - \vec{d}\right)^T$$

$$\approx \frac{1}{2}(\vec{u} - \vec{u}_0)^T \left(\frac{\partial \mathcal{M}}{\partial u}\right)^T W \left(\frac{\partial \mathcal{M}}{\partial u}\right)(\vec{u} - $$

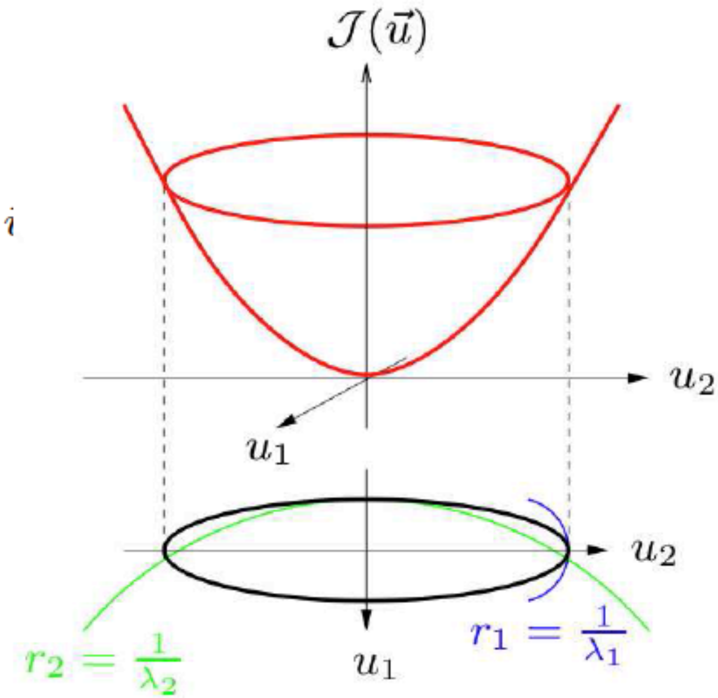
- Compare to multivariate Gaussian distribution

$$\mathcal{N}(\vec{u}_0, \Sigma) \propto \exp\left[(\vec{u} - \vec{u}_0)^T \Sigma^{-1} (\vec{u} - \vec{u}_0)\right]$$

- posterior error covariance matrix Σ is inverse of Hessian H of $\mathcal{J}(\vec{u})$ at minimum:

$$H = d_u^2 \mathcal{J}(\vec{u}_{opt})$$

$$= \left(\frac{\partial \mathcal{M}}{\partial u}\right)^T W \left(\frac{\partial \mathcal{M}}{\partial u}\right) + \left(\frac{\partial^2 \mathcal{M}_k}{\partial u_i \partial u_j}\right) W \left(\mathcal{M}(\vec{u}) - \vec{d}\right)$$



- Eigenvalues of H: principal curvatures

$$\frac{\text{largest EV}}{\text{smallest EV}} = \text{conditioning number}$$

- r_i: principal curvatures
- $\det(H^{-1})$: Gauss curvature
- $\text{trace}\,(H^{-1})$: mean curvature

Some algebra

Need $\vec{\nabla}_u \mathcal{J}|_{u_0}$ of $\mathcal{J}(\vec{u}_0) \in I\!R^1$ w.r.t. control variable $\vec{u} \in I\!R^m$

$\mathcal{J}$	:	$\vec{u}$	$\mapsto$	$\vec{v} = \mathcal{M}(\vec{u})$	$\mapsto$	$\mathcal{J}(\mathcal{M}(\vec{u}))$
TLM	:	$\delta\vec{u}$	$\mapsto$	$\delta\vec{v} = M \cdot \delta\vec{u}$	$\mapsto$	$\delta\mathcal{J} = \vec{\nabla}_u \mathcal{J} \cdot \delta\vec{u}$
ADM	:	$\delta^*\vec{u} = \vec{\nabla}_u \mathcal{J}^T$	$\leftarrow\!\!\dashv$	$\delta^*\vec{v}$	$\leftarrow\!\!\dashv$	$\delta\mathcal{J}$

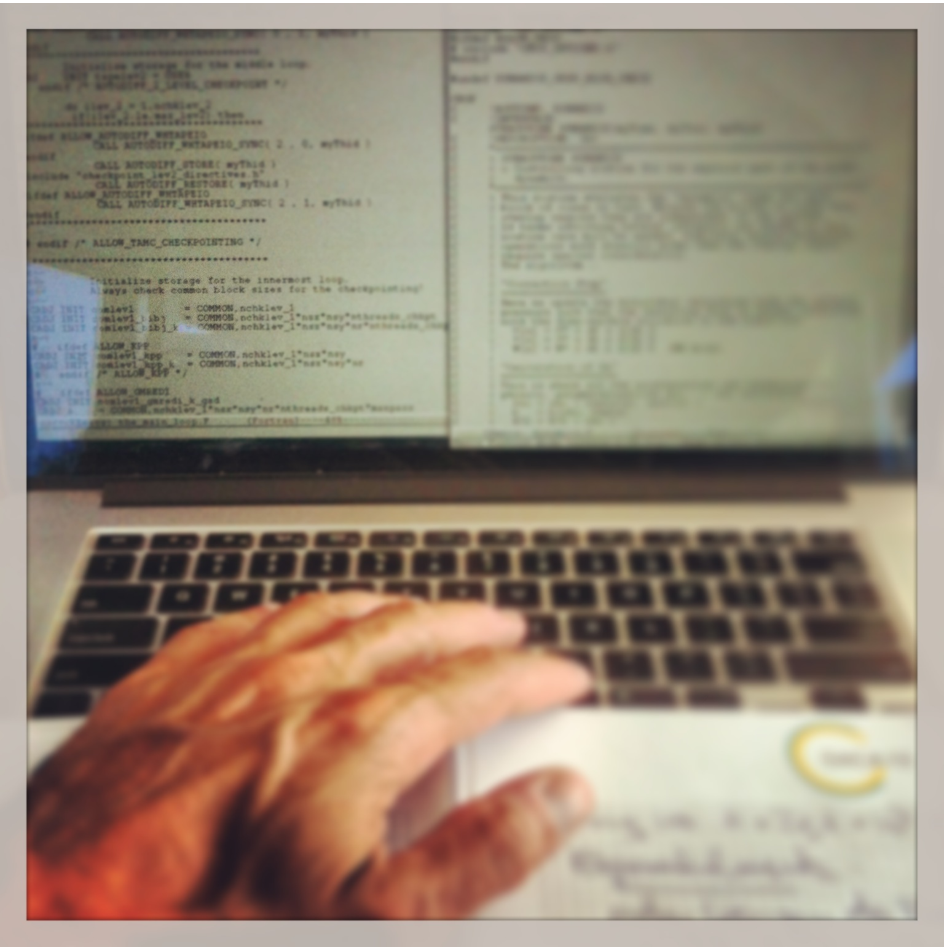
- $\vec{v} = \mathcal{M}(\vec{u})$ nonlinear model
- $M \; , \; M^T$ tangent linear (TLM) / adjoint (ADM)
- $\delta\vec{u} \; , \; \delta^*\vec{u}$ perturbation / dual (or sensitivity)

$$\boxed{\vec{\nabla}_u \mathcal{J}^T|_{\vec{u}} \quad = \quad M^T|_{\vec{v}} \cdot \vec{\nabla}_v \mathcal{J}|_{\vec{v}}}$$

TLM : $m\,(\sim n_x n_y n_z)$ integrations @ $1 \cdot$ (#forward)

ADM : 1 integration @ $\gamma \cdot$ (#forward)

How to get an adjoint model?



hand-written adjoint

automatic differentiation

Adjoint code generation via Automatic / Algorithmic Differentiation (AD) applied to the MITgcm code (100,000+ lines of code)

- Nonlinear model code

$$\vec{v} = \mathcal{M}_\Lambda\left(\mathcal{M}_{\Lambda-1}\left(\ldots\ldots\left(\mathcal{M}_0\left(\vec{u}\right)\right)\right)\right)$$

- Adjoint model code

$$\delta^*\vec{u} = M_0^T \cdot M_1^T \cdot \ldots\ldots \cdot M_\Lambda^T \cdot \delta^*\vec{v}$$

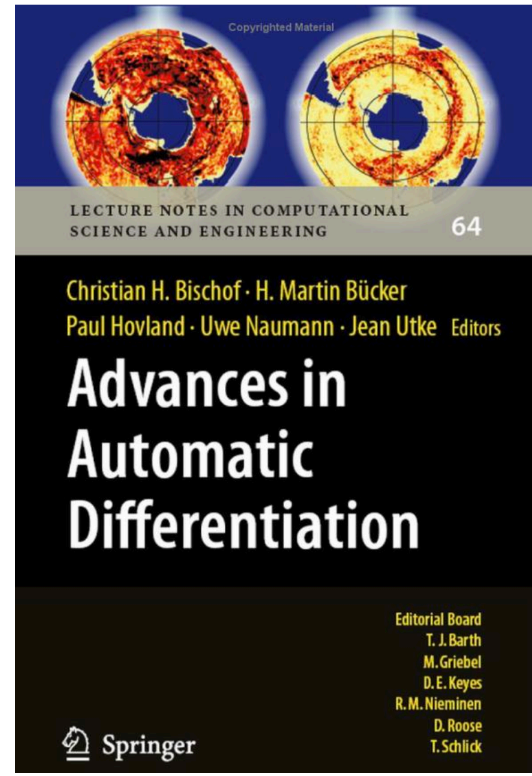
- Automatic differentiation:

Marotzke et al. JGR 1999

Stammer et al. JGR 2002

Heimbach et al. FGCS, 2005

each line of code is elementary operator $\mathcal{M}_\lambda$
$\longrightarrow$ rules for differentiating elementary operations
$\longrightarrow$ yield elementary Jacobians M_λ
$\longrightarrow$ composition of M_λ's according to chain rule
$\longrightarrow$ transpose M_λ^T gives adjoint
yield full tangent linear / adjoint model



- Source-to-source AD tools:
  TAF (Giering & Kaminski, 1998), commercial
  OpenAD (Utke et al., 2008), open-source

- model $\mathcal{M}$
- independent $\vec{u}$
- dependent $\mathcal{J}$

$\xrightarrow{\text{TAF, OpenAD}}$

TLM M, or
ADM M^T, or
gradient $\delta^*\vec{u} = \vec{\nabla}_u \mathcal{J}$

Implementation options (L. Harscoet & K. Narayanan, pers. comm.)

Implementation-wise, several strategies may be available

Strategies differ in:

- efficiency of derivative code
- suitability for the sort of derivatives required
- ease of use
- tool development investment
- dependence on the application language

Options:

- Source-to-source transformation vs. operator overloading
- Diff. variables association-by-name vs. association-by-address
- Reverse retrieval by storage or by re-computation

Strengths & weaknesses (L. Harscoet & K. Narayanan, pers. comm.)

- operator overloading:

  + few restrictions, flexibility, ease of coding language

  - adjoint tape size; interpreted => slower*3; overhead; source code preparation

- source-to-source transformation:

  + smaller adjoint tape; global analyses; compiler optimizations => better efficiency / performance

  - lagging behind language features; development cost

Other aspects:

- assoc. by address: maintaining connection, locality
- assoc. by type: readability

Available AD tools:
http://autodiff.org

- Source-to-source transformation:
  - TAF/TAC++ (Germany, commercial) – *MITgcm (ocean & ice)*
  - OpenAD/F & ADIC (Argonne NL, USA) – *MITgcm (ocean & ice)*
  - Tapenade (INRIA, France)
- Operator overloading:
  - ADOL-C (Argonne NL, USA) – *Ice Sheet System Model (ISSM)*
  - NAGware-95 (RWTH, Germany)

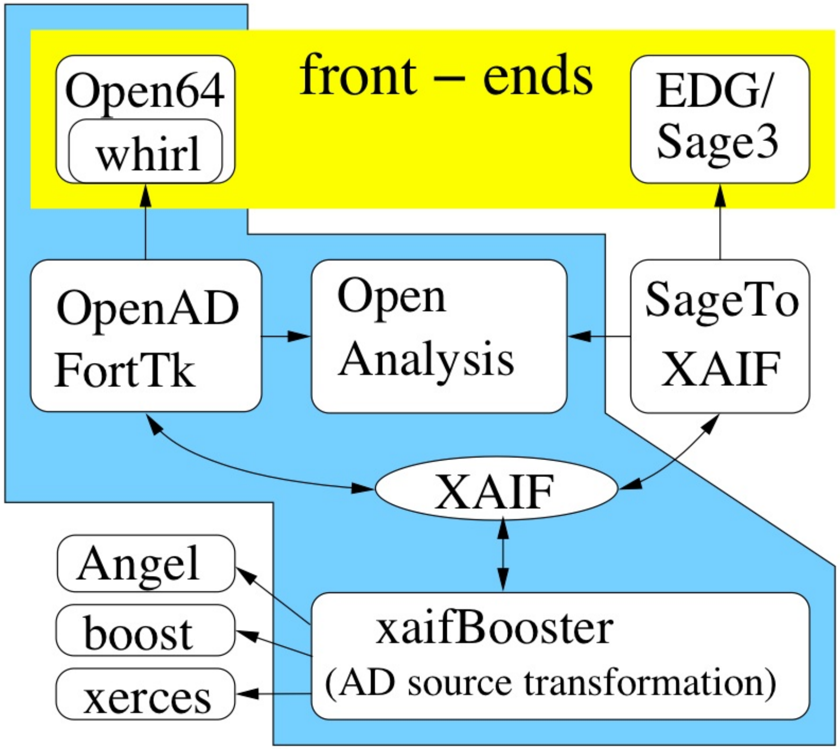
Shameless self-promotion:

- MITgcm/ECCO framework has been *flagship application* for AD
- can now be differentiated using **both** *TAF* **and** *OpenAD*
- **both** ocean GCM **and** ice sheet model differentiated

Significant national & international (e.g., UK, Germany, Norway, …) interest in accessibility to open-source AD tools

OpenAD: an open-source algorithmic differentiation tool

http://www.mcs.anl.gov/OpenAD



Argonne NATIONAL LABORATORY
Massachusetts Institute of Technology
RICE
RWTH AACHEN
RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

Tool design emphases

- modularity
- flexibility
- use of open-source components
- new algorithmic approaches:
  - XML-based language-independent transformation
  - basic block preaccumulation
  - other optimal elimination methods
  - control flow & call graph reversal
  - hierarchical checkpointing

Immediate needs:

Tool support at agency level
for climate applications (esp. DOE)

(started with NSF-CMG & NASA support)

Conclusion

- Gradient information are powerful ingredients in climate research (DA, sensitivity, predictability, UQ, …)
- can be efficiently obtained via adjoint model
- obtaining adjoint of full-fledged model is challenging
- algorithmic differentiation (AD) has proven feasible
  - is generating increasing interest in modeling community
- think of using AD tool like driving a Formula 1 car
  - requires skillful driver
  - highly tuned: tool improves with each new application
  - requires AD tool support
- strong desire for better access to (open-source) AD tools

Specific recommendation: increase *OpenAD* tool support at agency level for climate applications (esp. DOE)