Adjoint model code generation via automatic differentiation and its application to ocean / sea ice state estimation

Patrick Heimbach and Dimitris Menemenlis

MIT and JPL

Some people involved:

- **MITgcm model development:**
  A. Adcroft, J.M. Campin, C. Hill, J. Marshall
- **ECCO environment development:**
  A. Köhl, D. Stammer, C. Wunsch
- **Sea ice model development:**
  D. Menemenlis, J. Zhang
- **Adjoint model aspects**
  R. Giering, D. Ferreira, G. Gebbie, M. Losch

Table of contents

Some references I

- **Some 'classics' on the adjoint method (subjective & incomplete):**

- F. Le Dimet and O. Talagrand: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus* **38A**, pp. 97–110.
- O. Talagrand and P. Courtier: Variational assimilation of meteorological observations with the adjoint vorticity equation. I: Theory. *Q. J. R. Meteorol. Soc.*, **113**, pp. 1311–1328, 1987.
- W.C. Thacker and R.B. Long: Fitting dynamics to data. *J. Geophys. Res.*, **93, C2**, pp. 1,227–1,240, 1988.
- W.C. Thacker: The role of the Hessian matrix in fitting models to measurements. *J. Geophys. Res.*, **94, C5**, pp. 6,177–6,196, 1989.
- E. Tziperman and W.C. Thacker: An optimal control / adjoint equation approach to studying the ocean general circulation. *J. Phys. Oceanogr.*, **19**, pp. 1471–1485, 1989.
- E. Tziperman, W.C. Thacker, R.B. Long and S. Hwang: Oceanic data analysis using a general circulation model. I: Simulations. *J. Phys. Oceanogr.*, **22**, 1434–1457, 1992.
- J. Marotzke and C. Wunsch: Finding the steady state of a general circulation model through data assimilation: Application to the North Atlantic Ocean. *J. Geophys. Res.*, **98, C11**, pp. 20,149–20,167, 1993.
- O. Talagrand: Assimilation of observations, an introduction. *J. Meteorol. Soc. Japan*, **75**, pp. 191–209, 1997.
- R. Errico: What is an adjoint model? *BAMS*, **78**, pp. 2577–2591, 1997.
- C. Wunsch: The ocean circulation inverse problem. *CUP*, 1996.

Some references II

- **Automatic/algorithmic differentiation (AD):**
  - Giering & Kaminski: *Recipes for Adjoint Code Construction*.
    ACM Transactions on Mathematical Software, vol. 24, no. 4, 1998
  - *MITgcm manual*, chapter 5, available at `http://mitgcm.org/sealion`
  - Heimbach, Hill & Giering: *An efficient exact adjoint of the MIT general circulation model, generated via automatic differentiation*.
    submitted to 'Future Generation Computer Systems', 2002.
  - A. Griewank: *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation.* Frontiers in Applied Mathematics, vol. 19, 369 pp., SIAM, Philadelphia, 2000.
  - A collection of resources on AD,
    `http://www.autodiff.org/` ,
    including an ongoing NSF-ITR project
    Adjoint Compiler Technology & Standards (ACTS)
    `http://www-unix.mcs.anl.gov/~naumann/ACTS/`
  - A collection of publications using TAMC/TAF:
    `http://www.fastopt.de/references/all.html`
- **Balancing storing vs. recomputing: n-level checkpointing**
  - A. Griewank: *Achieving logarithmic growth of temporal and spatial complexity in reverse AD*. Optimization Methods and Software **1**, pp. 35–54, 1992.
  - J.M. Restrepo, G.K. Leaf and A. Griewank: *Circumvening storage limitations in variational data assimilation studies*. SIAM J. Sci. Comput. **19**, pp. 1586–1605, 1998.

Some references III

- **Adjoint model applications to sensitivity studies:**

  Atmosphere:

  - M. Hall: Application of adjoint sensitivity theory to an atmospheric general circulation model. *J. Atmos. Sci.*, **43**, pp. 2644–2651, 1986.
  - R. Errico and T. Vukicevic: Sensitivity analysis using an adjoint of the PSU/NCAR mesoscale model. *Mon. Wea. Rev.*, **12**, pp. 1644–1660, 1992.
  - S. Corti and T. Palmer: Sensitivity analysis of atmospheric low-frequency variability. *Q. J. R. Meteorol. Soc.*, **123**, 2425–2447, 1997.

  Ocean:

  - J. Marotzke, R. Giering, K.Q. Zhang, D. Stammer, C. Hill and T. Lee: Construction of the adjoint MIT ocean general circulation model and application to Atlantic heat transport variability. *J. Geophys. Res.*, **104, C12**, pp. 29,529–29,547, 1999.
  - V. Bugnion: Driving the ocean's overturning: an adjoint sensitivity study. Ph.D. thesis, MIT/EAPS, Cambridge (MA), USA, June 2001.
  - M. Junge and T. Haine: Mechanisms of North Atlantic Wintertime Sea Surface Temperature Anomalies. *J. Clim.*, **14**, pp. 4560–4572, 2001.
  - E. Galanti, E. Tziperman, M. Harrison, A. Rosati, R. Giering and Z. Sirkes: The equatorial thermocline outcropping - A seasonal control on the tropical Pacific ocean-atmosphere instability. *J. Clim.*, **15**, pp. 2721–2739, 2002.

Some references IV

- **Adjoint model applications to ocean state estimation:**
  - J. Sheinbaum and D. Anderson: Variational assimilation of XBT data. Part 1. *J. Phys. Oceanogr.*, **20**, 672–688, 1990.
  - J. Schröter, U. Seiler, and M. Wenzel: Variational assimilation of Geosat data into an eddy-resolving model of the Gulf stream extension area. *J. Phys. Oceanogr.*, **23**, pp. 925–953, 1993.
  - B. Luong, J. Blum and J. Verron: A variational method for the resolution of a data assimilation problem in oceanography. *Inverse Problems*, **14**, 979–997, 1998.
  - Martin Losch, René Riedler and Jens Schröter: Estimating a mean ocean state from hydrography and sea-surface height data with a nonlinear inverse section model: twin experiments with a synthetic dataset. *J. Phys. Oceanogr.*, **32**, pp. 2096-2112, 2002.
  - Jens Schröter, Martin Losch, and Bernadette Sloyan: Impact of the Gravity field and steady-state Ocean Circulation Explorer (GOCE) mission on ocean circulation estimates. Volume and heat transports across hydrographic sections. *J. Geophys. Res.*, **107, C2**, 2002.
  - D. Stammer, C. Wunsch, R. Giering, C. Eckert, P. Heimbach, J. Marotzke, A. Adcroft, C.N. Hill and J. Marshall: The global ocean circulation and transports during 1992 – 1997, estimated from ocean observations and a general circulation model. *J. Geophys. Res.*, **107, C9**, pp. 3118, 2002.
  - D. Stammer, C. Wunsch, R. Giering, C. Eckert, P. Heimbach, J. Marotzke, A. Adcroft, C.N. Hill and J. Marshall: Volume, heat and freshwater transports of the global ocean circulation 1993 – 2000, estimated from a general circulation model constrained by WOCE data. *J. Geophys. Res.*, **108, C1**, pp. 3007, 2003.

Some references V

- **Data assimilation in highly nonlinear systems:**

  Atmosphere:

  - M. Tanguay, P. Bartello and P. Gauthier: Four-dimensional data assimilation with a wide range of scales. *Tellus*, **47A**, pp. 974–997, 1995.
  - C. Pires, R. Vautard, O. Talagrand: On extending the limits of variational assimilation in nonlinear chaotic systems. *Tellus*, **48A**, pp. 96–121, 1996
  - R. Miller, E. Carter and S. Blue: Data assimilation into nonlinear stochastic models. *Tellus*, **51A**, pp. 167–194, 1999.

  Ocean:

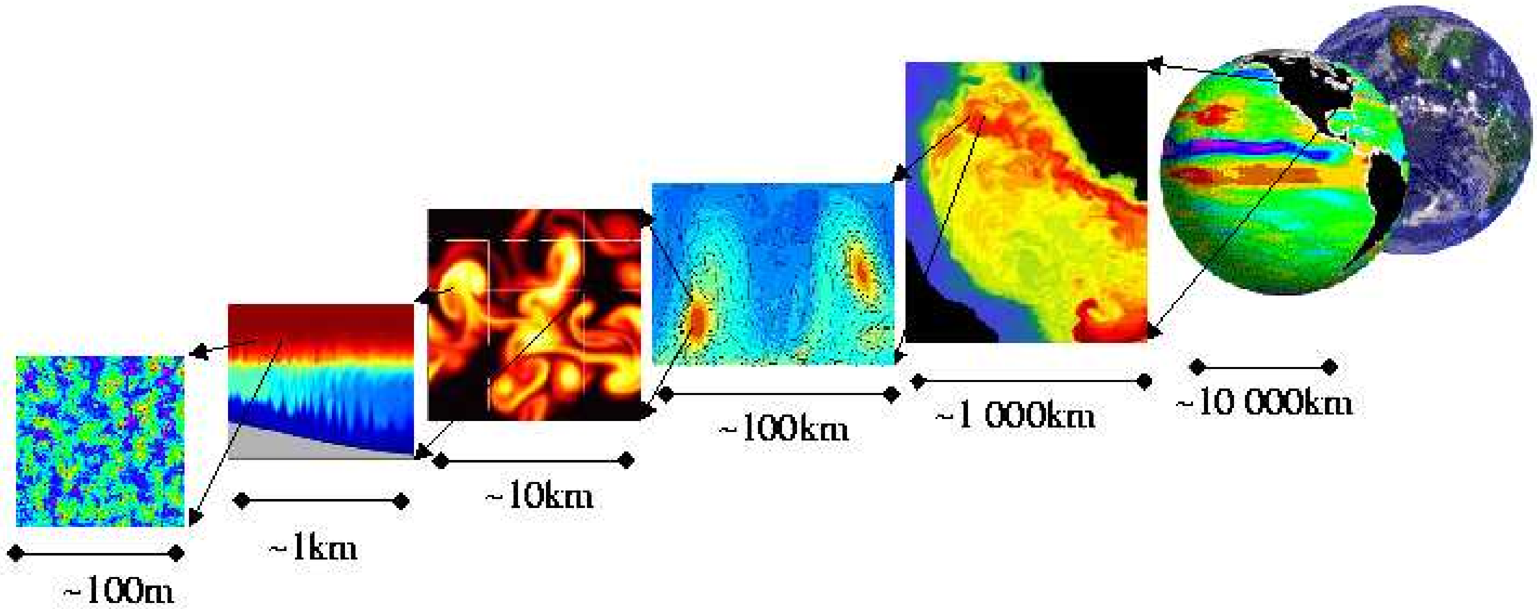
  - D. Lea and M. Allen and T. Haine: Sensitivity analysis of the climate of a chaotic system. *Tellus*, **52A**, pp. 523–532, 2000.
  - D. Lea, M. Allen, T. Haine, and J. Hansen: Sensitivity analysis of the climate of a chaotic ocean circulation model. *Q. J. R. Meteorol. Soc.*, **128**, pp. 2587-2605, 2002.
  - A. Köhl and J. Willebrand: An adjoint method for the assimilation of statistical characteristics into eddy-resolving ocean models. *Tellus*, **54A**, pp. 406–425, 2002.
  - A. Köhl and J. Willebrand: Variational assimilation of SSH variablity from TOPEX/POSEIDON and ERS-1 into an eddy-permitting model of the North Atlantic. *J. Geophys. Res.*, **108, C3**, pp. 3092, 2003.
  - G. Gebbie et. al., in preparation, 2003.

Some references VI

- **Reverse mode checkpointing:**
  - A. Griewank: Achieving logarithmic growth of temporal and spatial complexity in reverse Automatic Differentiation. *Optimization Methods and Software*, **1**, pp. 35–58, 1992.
  - J.M. Restrepo, G.K. Leaf and A. Griewank: Circumvening storage limitations in variational data assimilation studies. *SIAM J. Sci. Comput.*, **19**, pp. 1586–1605, 1998.
- **Box models of the thermohaline circulation**
  - H. Stommel: Thermohaline convection with two stable regimes of flow. *Tellus*, **13**, pp. 224–230, 1961
  - S. Griffies and E. Tziperman: A linear thermohaline oscillator driven by stochastic atmospheric forcing. *J. Clim.*, **8**, pp. 2440–2453.
  - I. Rivin and E. Tziperman: Linear versus self-sustained interdecadal thermohaline variability in a coupled box model. *J. Phys. Oceanogr.*, **27**, pp. 1216–1232, 1997.
  - E. Tziperman and P. Ioannou: Transient growth and optimal excitation of thermohaline variability. *J. Phys. Oceanogr.*, **32**, pp. 3427–3435, 2002.

The context of this state estimation system

- The ongoing MITgcm model development,
  `http://mitgcm.org/sealion/`
- The ongoing *Estimating the Circulation and Climate of the Ocean* (ECCO) project of global WOCE data–model synthesis,
  `http://www.ecco-group.org`
- The newly developed sea ice model by *Menemenlis & Zhang*
- A wealth of little-explored high-latitude ocean and sea ice data



Adjoint applications (I): Ocean State Estimation

- Given:
  - a set of (possibly different types of) observations
  - a numerical model & set of initial / boundary conditions
- Question: (estimation / optimal control problem)
  Find "*optimal*" model trajectory consistent with available observations within given prior errors
- Iterative approach:
  Minimize least square function $\mathcal{J}(\vec{u})$ measuring model vs. data misfi t
  $\longrightarrow$ seek $\boxed{\vec{\nabla}_u \mathcal{J}(\vec{u})}$ to infer update $\Delta\vec{u}$ from variation of controls $\vec{u}$

$$\vec{u}^{n+1} = \vec{u}^{n} + \Delta\vec{u}$$

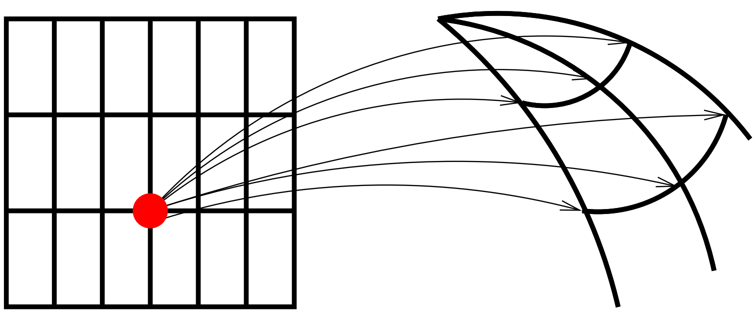
- Results:
  - optimal/consistent ocean state estimate
  - corrected initial/boundary value estimates

Adjoint applications (II): Sensitivity analysis

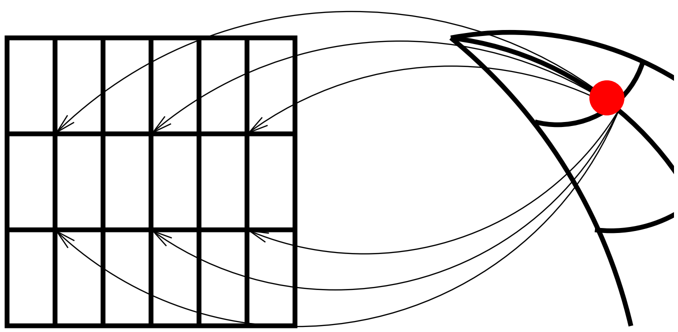
- Finite difference approach:
  - Take a "guessed" anomaly (SST) and determine its impact on model output (MOC)
  - Perturb each input element ($\mathrm{SST(i,j)}$) to determine its impact on output (MOC).

- Reverse/adjoint approach:
  - Calculates "full" sensitivity fi eld $\frac{\partial\,\mathrm{MOC}}{\partial\,\mathrm{SST}(x,y,t)}$
  - Approach:
    Let $\mathcal{J} = \mathrm{MOC}$, $\vec{u} = \mathrm{SST(i,j)}$

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



finite difference approach



adjoint approach

Adjoint applications (III): SVD / Optimal Perturbations

- For dynamical system $\vec{v}(t) = M\vec{v}(0)$, with $\vec{v}_0 = \vec{u}$, find initial conditions $\vec{u}$, such that model state $\vec{v}(t)$ maximizes a chosen norm

$$\langle \, \vec{v}(t) \, , \vec{v}(t) \, \rangle_t \; = \; \vec{v}(t)^T \cdot X_t \cdot \vec{v}(t)$$

$$\max_{\vec{u}} \Big\{ \, (M\vec{u})^T \cdot X_t \cdot M\vec{u} \; - \; \lambda \big[\, \vec{u}^T \cdot X_0 \cdot \vec{u} \; - \; 1 \, \big] \, \Big\}$$

  with Lagrange multipliers λ to enforce norm unity at $t = 0$

- Leads to generalized eigenvalue problem:

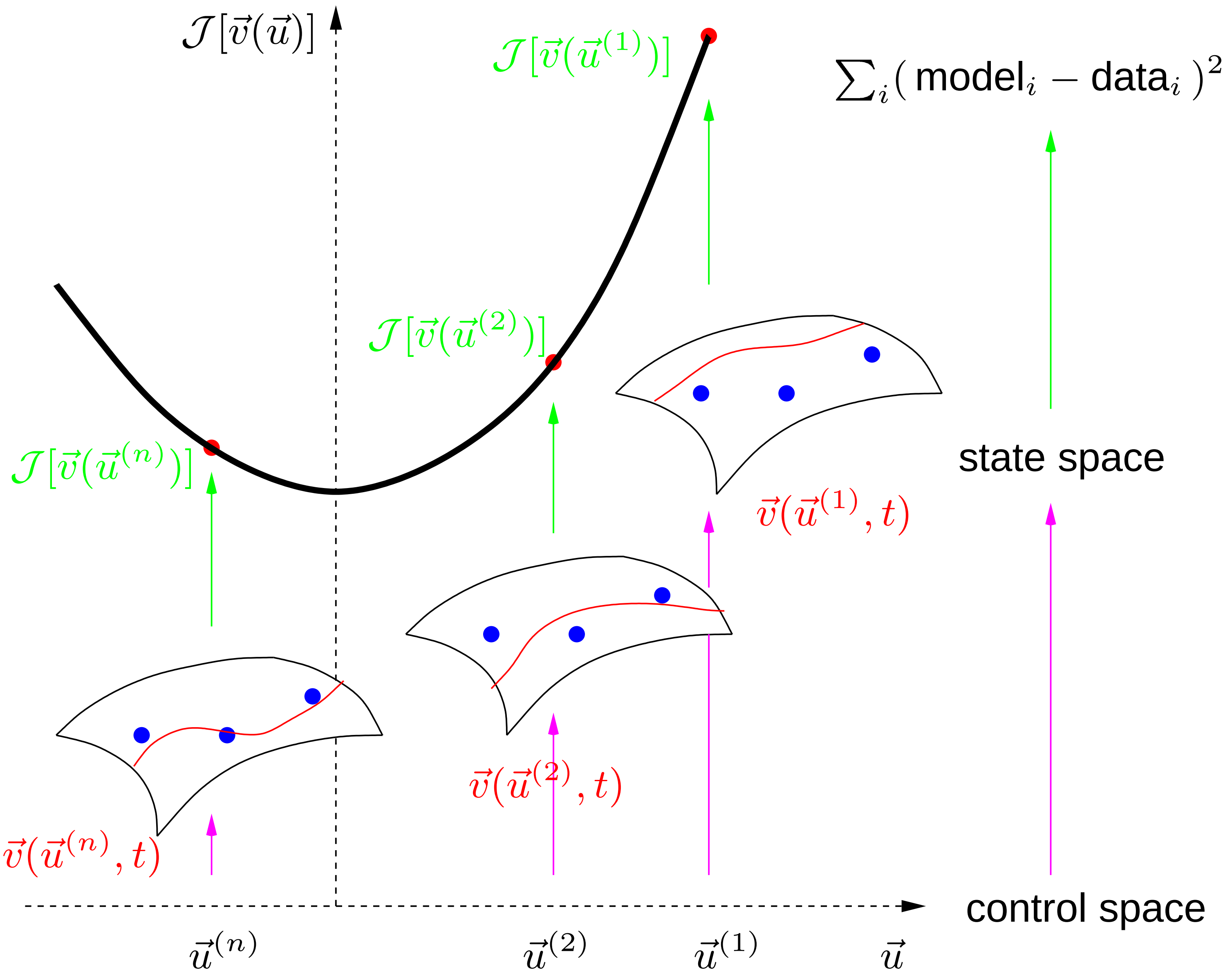
$$M^T \cdot X_t \cdot M \cdot \vec{u} \; = \; \lambda \; X_0 \cdot \vec{u}$$

  Thus, in terms of the tangent linear and adjoint operator:

$$\boxed{\mathtt{ADM} \cdot \mathtt{X_t} \cdot \mathtt{TLM} \cdot \mathtt{u} \; = \; \lambda \; \mathtt{X_0} \cdot \mathtt{u}}$$

  $\longrightarrow$ iterative solution for optimum $\vec{u}_{opt}$ via
  Implicit Restarted Arnoldi Iteration Routine ARPACK

Iterative optimization via gradient descent



Some definitions

$\vec{u}$	independent / control variables
	(e.g. initial T, S, surface forcing, background diffusivites)
$\vec{v}$	model state (at time t): T, S, U, V, W
$\mathcal{M}$	(nonlinear) model operator $\mathcal{M}(\vec{u}, t) = \vec{v}(t)$
$\mathcal{J}$	objective / cost function (e.g. least-square misfi t)

$\delta\vec{u}$	perturbation of independent /control variable
$\delta\vec{v}$	perturbed model state due to control perturbation
M	tangent linear operator of model operator, $M = (\partial\mathcal{M}_i \, / \, \partial u_j)$
$\delta\mathcal{J}$	cost function variation as result of control perturbation

Some algebra (i)

model $\mathcal{M}$:	*control space*	$\rightarrow$	*model state space*
tangent linear M:	δ*(control space)*	$\rightarrow$	δ*(model state space)*

Consider cost function $\mathcal{J}$

$$
\begin{array}{lcccccc}
\mathcal{J} : & U & \longrightarrow & V & \longrightarrow & I\!R \\
 & \vec{u} & \longmapsto & \vec{v} = \mathcal{M}(\vec{u}) & \longmapsto & \mathcal{J}(\vec{u}) = \mathcal{J}(\mathcal{M}(\vec{u}))
\end{array}
$$

$$
TLM : \quad \delta\vec{u} \quad \longmapsto \quad \delta\vec{v} = M \cdot \delta\vec{u} \quad \longmapsto \quad \delta\mathcal{J} = \vec{\nabla}_u \mathcal{J}^T \cdot \delta\vec{u}
$$

with tangent linear model

$$
M = \left(\frac{\partial \mathcal{M}_i}{\partial u_j} \right) \Big|_{\vec{u}_0}
$$

Minimize cost $\mathcal{J}$: Seek control variables $\vec{u}$ such that gradient vanishes:

$$
\min_{\vec{u}} \mathcal{J}(\vec{u}) \quad \Rightarrow \quad \vec{\nabla}_u \mathcal{J}(\vec{u}) = 0
$$

Some algebra (ii)

Expansion of $\mathcal{J}$ in terms of $\delta\vec{u}$ or $\delta\vec{v}$:

$$\begin{aligned}\mathcal{J} \; = \; \mathcal{J}_0 + \delta\mathcal{J} \; &= \; \mathcal{J}|_{\vec{u}_0} + \left\langle \nabla_u\mathcal{J}^T \, , \, \delta\vec{u} \right\rangle + O(\delta\vec{u}^2) \\ &= \; \mathcal{J}|_{\vec{v}_0} + \left\langle \nabla_v\mathcal{J}^T \, , \, \delta\vec{v} \right\rangle + O(\delta\vec{v}^2)\end{aligned}$$

Then, evaluate $\delta\mathcal{J}$:

$$\delta\mathcal{J} \; = \left\langle \nabla_v\mathcal{J}^T \, , \, \delta\vec{v} \right\rangle = \left\langle \nabla_v\mathcal{J}^T \, , \, M\,\delta\vec{u} \right\rangle = \left\langle M^T\,\nabla_v\mathcal{J}^T \, , \, \delta\vec{u} \right\rangle$$

The full gradient can be inferred via the **adjoint model (ADM)**

$$\begin{aligned}\nabla_u\mathcal{J}^T|_{\vec{u}} &= \; M^T \cdot \nabla_v\mathcal{J}^T|_{\vec{v}} \\ &= \; M^T \cdot \delta^*\vec{v} \\ &= \; \delta^*\vec{u}\end{aligned}$$

with M^T the adjoint (transpose) of the tangent linear operator M

Some algebra (iii)

Application of the chain rule:

$$\begin{aligned}\vec{v} &= \mathcal{M}(\vec{u}) &= \mathcal{M}_\Lambda(\mathcal{M}_{\Lambda-1}(\ldots(\mathcal{M}_\lambda(\ldots(\mathcal{M}_1(\mathcal{M}_0(\vec{u})))))))\\ \delta\vec{v} &= M\cdot\delta\vec{u} &= M_\Lambda\cdot M_{\Lambda-1}\cdot\ldots\cdot M_\lambda\cdot\ldots\cdot M_1\cdot M_0\cdot\delta\vec{u}\end{aligned}$$

Reveals the reverse nature of the adjoint calculation:

$$\begin{aligned}\delta\mathcal{J} &= \langle\ \nabla_v\mathcal{J}^T\ ,\ \delta\vec{v}\ \rangle\\ &= \langle\ \nabla_v\mathcal{J}^T\ ,\ M_\Lambda\cdot\ldots\cdot M_0\cdot\delta\vec{u}\ \rangle\\ &= \langle\ M_0^T\cdot\ldots\cdot M_\Lambda^T\cdot\nabla_v\mathcal{J}^T\ ,\ \delta\vec{u}\ \rangle\\ &= \langle\ \nabla_u\mathcal{J}^T\ ,\ \delta\vec{u}\ \rangle\end{aligned}$$

The Adjoint method

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 :	$\delta^*\vec{u} = \vec{\nabla}_u \mathcal{J}^T$	$\leftarrow\!\mid$	$\delta^*\vec{v}$	$\leftarrow\!\mid$	$\delta\mathcal{J}$

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

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

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

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

Automatic differentiation (AD)

- 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 code

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

- Automatic differentiation:

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
yield full tangent linear / adjoint model

- TAMC/TAF source-to-source tool (Giering & Kaminski, 1998)

• model $\mathcal{M}$ • independent $\vec{u}$ • dependent $\mathcal{J}$	TAMC/TAF $\longrightarrow$	TLM M, or ADM M^T, or gradient $\delta^* \vec{u} = \vec{\nabla}_u \mathcal{J}$

Example: Eli's 3-box model of the THC

```
DO t = 1, nTimeSteps
```

- calculate density

$$\rho = -\alpha T + \beta S$$

- calculate thermohaline transport

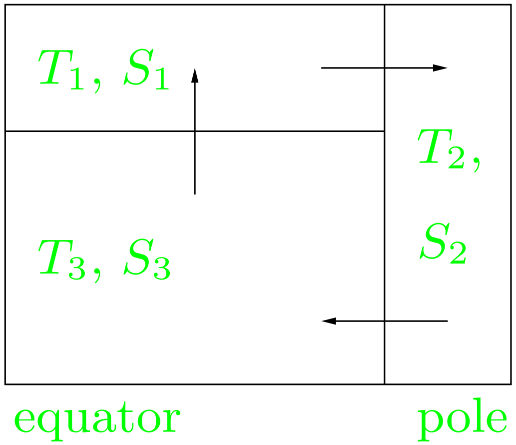
$$U = U(\rho(T, S))$$

- calculate tracer advection

$$\frac{d}{dt}Tr = f(Tr, U)$$

calculate timestepping and update tracer fields $Tr = \{T, S\}$

```
END DO
```



Focus on advection equation for T_3 (i)

$$\begin{aligned} \frac{dT_3}{dt} &= U(T_3 - T_2), \quad \text{for } U \geq 0 \\ \texttt{diffT3} &= \texttt{u} * (\texttt{T3} - \texttt{T2}) \end{aligned}$$

derivative $\delta\texttt{diffT3}$:

$$\delta\texttt{diffT3} = \frac{\partial\texttt{diffT3}}{\partial\texttt{U}}\delta\texttt{U} + \frac{\partial\texttt{diffT3}}{\partial\texttt{T}_2}\delta\texttt{T}_2 + \frac{\partial\texttt{diffT3}}{\partial\texttt{T}_3}\delta\texttt{T}_3$$

in matrix form:

$$\begin{pmatrix} \delta\texttt{diffT3} \\ \delta\texttt{T}_3 \\ \delta\texttt{T}_2 \\ \delta\texttt{U} \end{pmatrix}^{\lambda} = \begin{pmatrix} 0 & -\texttt{U} & \texttt{U} & \texttt{T3} - \texttt{T1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \delta\texttt{diffT3} \\ \delta\texttt{T}_3 \\ \delta\texttt{T}_2 \\ \delta\texttt{U} \end{pmatrix}^{\lambda-1}$$

Focus on advection equation for T_3 (ii)

Transposed relationship yields:

$$\begin{pmatrix} \delta^* \mathtt{diffT3} \\ \delta^* \mathtt{T}_3 \\ \delta^* \mathtt{T}_2 \\ \delta^* \mathtt{U} \end{pmatrix}^{\lambda-1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\mathtt{U} & 1 & 0 & 0 \\ \mathtt{U} & 0 & 1 & 0 \\ \mathtt{T3}-\mathtt{T1} & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \delta^* \mathtt{diffT3} \\ \delta^* \mathtt{T}_3 \\ \delta^* \mathtt{T}_2 \\ \delta^* \mathtt{U} \end{pmatrix}^{\lambda}$$

and thus adjoint code:

```
adT3     = adT3        - u*addiffT3
adT2     = adT2        + u*addiffT3
adU      = adu   + (T3-T2)*addiffT3
addiffT3 = 0
```

Note: state $\mathtt{T2}, \mathtt{T3}, \mathtt{U}$ are required to evaluate derivative at each time step, in reverse order!

$\longrightarrow$ *TANGENT* linearity

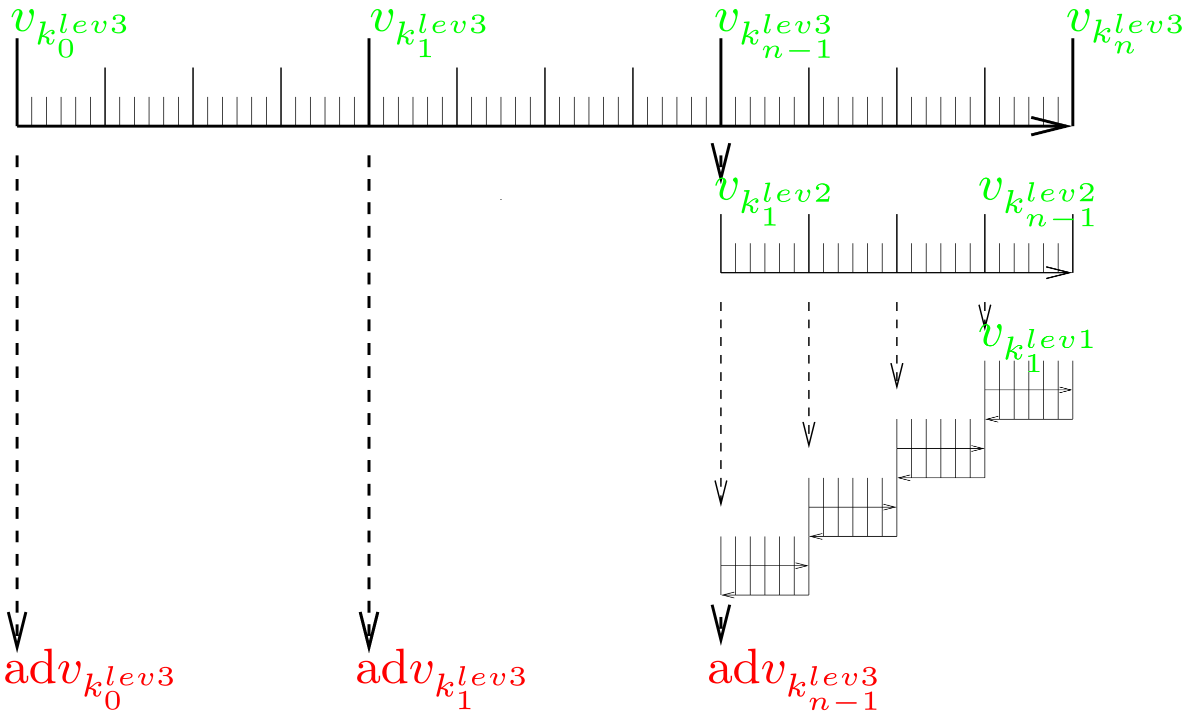
Reverse order integration (i)

```
DO istep = 1, nTimeSteps
  • call density(ρ)
  • call transport(U)
  • call timestep(T,S)
  • call update(T,S)
END DO
```

```
DO istep = nTimeSteps, 1, -1
C recompute required variables
  • DO iaux = 1, istep
     – call density(ρ)
     – call transport(U)
     – call timestep(T,S)
     – call update(T,S)
    END DO
C perform adjoint timestep
  • call adupdate(T,S)
  • call adtimestep(T,S)
  • call adtransport(U)
  • call addensity(ρ)
END DO
```

Reverse order integration (ii)

- *Adjoint = transpose of TLM*
  - → evaluated in reverse order
  - → model state at every time step required in reverse
  - → all state stored or recomputed
- *Solution: Checkpointing* (e.g. Griewank, 1992, Restrepo et al., 1998) balances storing vs. recomputation



Reverse order integration (iii)

`DO iOuter = 1, nOuter`

- *CADJ STORE* $T, S \longrightarrow$ *disk*
- `DO iInner = 1, nInner`
  - `call density`(ρ)
  - `call transport`(U)
  - `call timestep`(T, S)
  - `call update`(T, S)

  `END DO`

`END DO`

`DO iOuter = nOuter, 1, -1`

- *CADJ RESTORE* $T, S \longleftarrow$ *disk*
- `DO iInner = 1, nInner`
  - `call density`(ρ)
  - `call transport`(U)
  - *CADJ STORE* $T, S, U \longrightarrow$ *common*
  - `call timestep`(T, S)
  - `call update`(T, S)

  `END DO`
- `DO iInner = nInner, 1, -1`
  - `call adupdate`$(\mathrm{ad}T, \mathrm{ad}S)$
  - `call adtimestep`$(\mathrm{ad}T, \mathrm{ad}S)$
  - *CADJ RESTORE* $T, S, U \longleftarrow$ *common*
  - `call adtransport`$(\mathrm{ad}U)$
  - `call addensity`$(\mathrm{ad}\rho)$

  `END DO`

`END DO`

Reverse order integration (iv)

- e.g. 3-level checkpointing:

$$n_{TimeSteps} = n_1 \cdot n_2 \cdot n_3$$

$\rightarrow$ **Storing:** reduced from $n_1 \cdot n_2 \cdot n_3$ to

- disk: $n_2 + n_3$,
- memory: n_1

$\rightarrow$ **CPU:** $3 \cdot \text{forward} + 1 \cdot \text{adjoint} \approx 5.5 \cdot \text{forward}$

- Closely related to **adjoint dump & restart** problem.
  Available queue sizes at HPC Centres may be limited

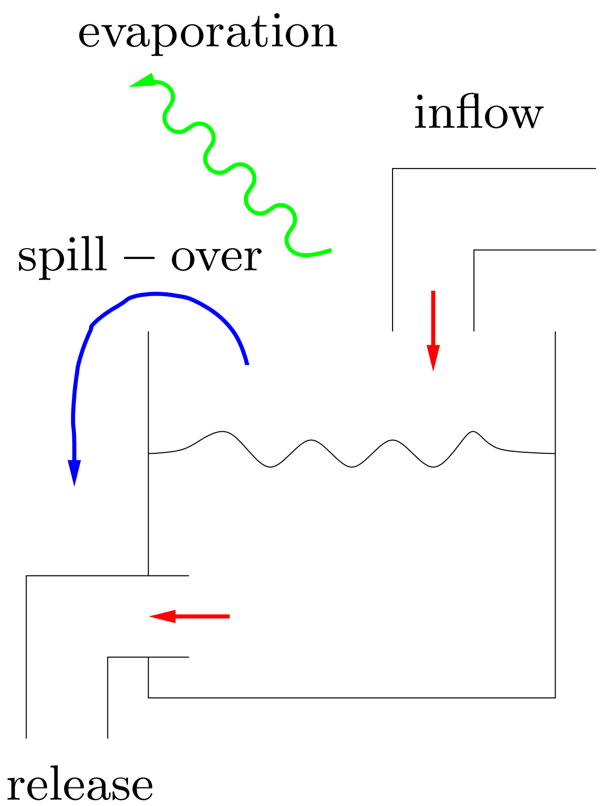
- Insertion of store directive requires detailed knowledge of code *and* AD tool behaviour
  $\longrightarrow$ not straightforward ("semi-automatic" differentiation only)

Thresholds: a hydrological reservoir model (I)

```
do t = 1, msteps
```

- get sources & sinks at time t
  inflow, evaporation, release
- calculate water release based on storage
  `release(t) = 0.8*storage(t-1)`$^{0.7}$
- calculate projected stored water,
  storage = storage+inflow-release-evaporation
  ```
  nominal = storage(t-1) +
     h*( infl(t)-release(t)-evap(t) )
  ```
- IF threshold capacity is exceeded, spill-over:
  `spill(t) = MAX(nominal-capac , 0.)`
- re-adjust projected stored water after spill-over:
  `storage(t) = nominal - spill(t)`
- determine outflow:
  `out(t) = release(t) + spill(t)/h`

```
end do
```



Thresholds: a hydrological reservoir model (II)

- The tangent linear model (*storage(0)* only as control)

```
do     t = 1, msteps
       g_release(t)   =   0.56*g_storage(t-1)*storage(t-1)**(-0.3)
       release(t)     =   0.8*storage(t-1)**0.7
       g_nominal      =   -g_release(t)*h+g_storage(t-1)
       nominal        =   storage(t-1)+h*(infl(t)-release(t)-evap(t))
       g_spill(t)     =   g_nominal*(0.5+SIGN(0.5,nominal-capac-0.))
       spill(t)       =   MAX(nominal-capac,0.)
       g_storage(t)   =   g_nominal-g_spill(t)
       storage(t)     =   nominal-spill(t)
       g_out(t)       =   g_release(t)+g_spill(t)/h
       out(t)         =   release(t)+spill(t)/h
end    do
```

- `g_release(t)` not defined for `storage(t-1) = 0.`
- `g_spill(t)` not defined for `nominal = capac`

Thresholds: a hydrological reservoir model (III)

- The adjoint model (with 2-level checkpointing implemented)

```
do    ilev2 = nlev2, 1, -1
      ...  recompute inner loop trajectory ...
      do    ilev1 = nlev1, 1, -1
            t                =   (ilev2-1)*nlev1+ilev1
            storage(t-1)     =   comlev1_storage_1h(ilev1)
            nominal          =   comlev1_nominal_2h(ilev1)
            adrelease(t)     =   adrelease(t)+adout(t)
            adspill(t)       =   adspill(t)+adout(t)/h
            adnominal        =   adnominal+adstorage(t)
            adspill(t)       =   adspill(t)-adstorage(t)
            adnominal        =   adnominal+adspill(t)
                                 *(0.5+sign(0.5d0,nominal-capac-0))
            adrelease(t)     =   adrelease(t)-adnominal*h
            adstorage(t-1)   =   adstorage(t-1)+adnominal
            adstorage(t-1)   =   adstorage(t-1)
                                 +0.56*adrelease(t)*storage(t-1)**(-0.3)
      end   do
end   do
```

Thresholds: a hydrological reservoir model (IV)

- *So far:*
  Control problem was formulated as initial state *storage(0)* as control only

- *Now:*
  Extend control problem to include part of the time-dependent forcing; choose *infbw(t)* as additional control variable

Two main messages:

- Observe, how time-dependent control variable acts in derivative code (here, TLM shown only)
- Observe, how derivative code gets (automatically) expanded, since variable `infl` which was previously passive, now becomes active (additional code is in red)

Thresholds: a hydrological reservoir model (V)

- The tangent linear model (*storage(0)* and *inflow(t)* as controls)

```
do    t = 1, msteps
      g_infl(t)      =   g_ctrlinfl(t)+g_infl(t)
      infl(t)        =   infl(t)+ctrlinfl(t)
      g_release(t)   =   0.56*g_storage(t-1)*storage(t-1)**(-0.3)
      release(t)     =   0.8*storage(t-1)**0.7
      g_nominal      =   g_infl(t)*height-g_release(t)*h+g_storage(t-1)
      nominal        =   storage(t-1)+h*(infl(t)-release(t)-evap(t))
      g_spill(t)     =   g_nominal*(0.5+SIGN(0.5,nominal-capac-0.))
      spill(t)       =   MAX(nominal-capac,0.)
      g_storage(t)   =   g_nominal-g_spill(t)
      storage(t)     =   nominal-spill(t)
      g_out(t)       =   g_release(t)+g_spill(t)/h
      out(t)         =   release(t)+spill(t)/h
end   do
```

Input/output – active file handling

I/O of active variables is common and needs to be accounted for in derivative

READ assigning a value to a variable

WRITE referencing a variable

code	hypothetical code	adjoint hypothetical code	adjoint code
`OPEN(8)`		`ADXD = 0.`	`OPEN(9)`
⋮	⋮	⋮	⋮
`WRITE(8) X`	`XD = X`	`ADXD = ADXD + ADZ`	`WRITE(9) ADZ`
		`ADZ = 0.`	`ADZ = 0.`
⋮	⋮	⋮	⋮
`READ(8) Z`	`Z = XD`	`ADX = ADX + ADXD`	`READ(9) ADXD`
		`ADXD = 0.`	`ADX = ADX + ADXD`
			`ADXD = 0.`
⋮	⋮	⋮	⋮
`CLOSE(8)`			`CLOSE(9)`

(from *Giering & Kaminski (1998)*)

Test / assure correctness of adjoint-derived gradient

Procedure to check that automatically derived gradient G_i^{ad} is correct;
consider perturbation of i-th control vector element u_i and $\Delta\vec{u}_i = \delta_{ij}$

finite difference vs. adjoint	tangent linear vs. adjoint
$G_i^{fd} = \frac{\mathcal{J}(u_i+\epsilon)-\mathcal{J}(u_i-\epsilon)}{2\epsilon}$	$G_i^{tl} = \vec{\nabla}_u \mathcal{J} \cdot \Delta\vec{u}_i = \left(\vec{\nabla}_u \mathcal{J}\right)_i$
$R_i^{fd} = 1 - \frac{G_i^{fd}}{G_i^{ad}}$	$R_i^{tl} = 1 - \frac{G_i^{tl}}{G_i^{ad}}$

$\longrightarrow$ can test 'correctness' of adjoint gradient G_i^{ad}

$\longrightarrow$ can test 'time horizon' of linearity assumption

Extension to vector-valued "cost"

For $\vec{\mathcal{J}}(\mathcal{M}(\vec{u})) \in I\!R^m$ previous expression for gradient is generalized to

$$M^T \cdot d_v \vec{\mathcal{J}}^T \cdot \delta \vec{\mathcal{J}} = d_u \vec{\mathcal{J}}^T \cdot \delta \vec{\mathcal{J}}$$

Thus, with $\delta \vec{u} \in I\!R^n$ and $\delta \vec{\mathcal{J}} \in I\!R^m$,

$$\left. \begin{array}{l} m \text{ adjoint} \\ n \text{ tangent linear} \end{array} \right\} \text{ runs for each perturbation } \left\{ \begin{array}{ll} (\delta \vec{\mathcal{J}})_j = \delta_{ij}, & j = 1, \ldots, m \\ (\delta \vec{u})_j = \delta_{ij}, & j = 1, \ldots, n \end{array} \right.$$

to recover full derivative $d_u \vec{\mathcal{J}}^T$.

$$\longrightarrow \quad \left. \begin{array}{c} \text{ADM} \\ \text{TLM} \end{array} \right\} \text{ is preferable, for } \left\{ \begin{array}{l} n > m \\ n < m \end{array} \right.$$

Examples

- Least-square model-vs.-data misfi t

$$\mathcal{J} = \langle \mathcal{H}(\vec{v}) - \vec{d}, \mathcal{H}(\vec{v}) - \vec{d} \rangle = \left(\mathcal{H}(\vec{v}) - \vec{d} \right)^T W \left(\mathcal{H}(\vec{v}) - \vec{d} \right)$$

e.g. for state estimation / data assimilation

with
- $\vec{d}$: data vector,
- $\mathcal{H}$: projector of model state space onto data space
- W: inverse prior error covariance (here variances only)

$$\nabla_v \mathcal{J}^T = 2\, H \cdot W \cdot \left(\mathcal{H}(\vec{v}) - \vec{d} \right)$$

$$\left(\nabla_v \mathcal{J}^T \right)_j = 2 \sum_k \left\{ \frac{\partial \mathcal{H}_k}{\partial v_j} W_{kk} \left(\mathcal{H}_k(\vec{v}) - d_k \right) \right\}$$

- Final state: $\vec{\mathcal{J}} = \vec{v}$

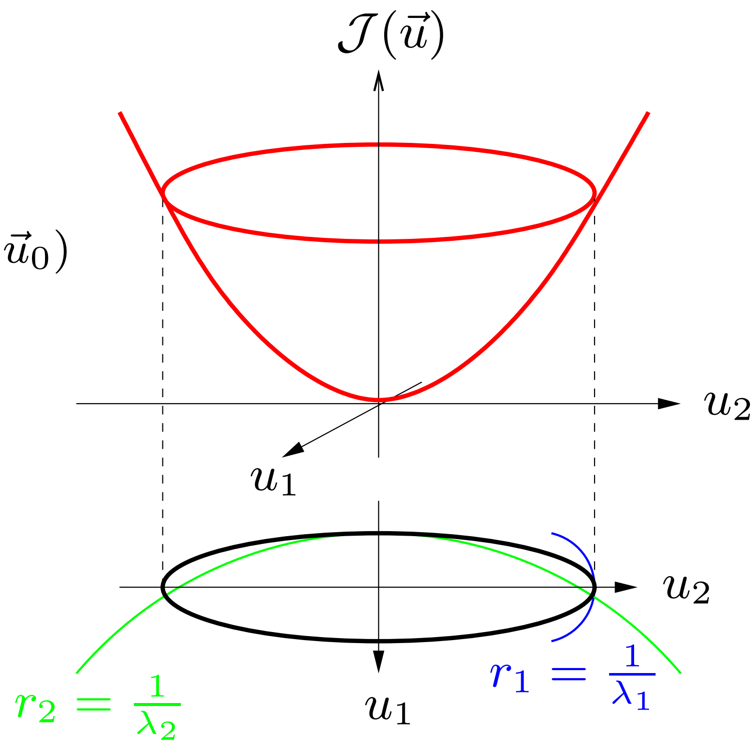
e.g. for SVD calculation, source-sink estimation

$$d_v \vec{\mathcal{J}}^T = \mathrm{Id}, \qquad d_u \vec{\mathcal{J}}^T = M^T \cdot d_v \vec{\mathcal{J}}^T = M^T$$

Uncertainties, ill-conditioning, 2nd derivative (Hessian)

- Consider linear approx. of cost function

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



- 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:

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

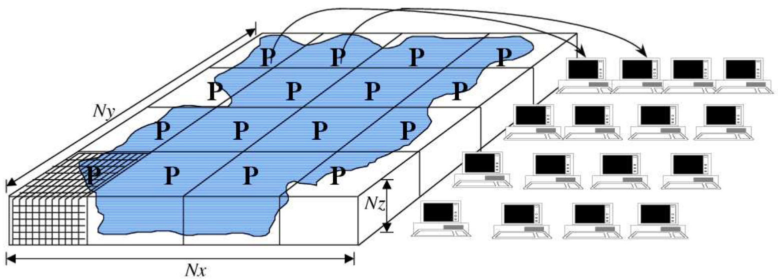
- Eigenvalues of H: principal curvatures

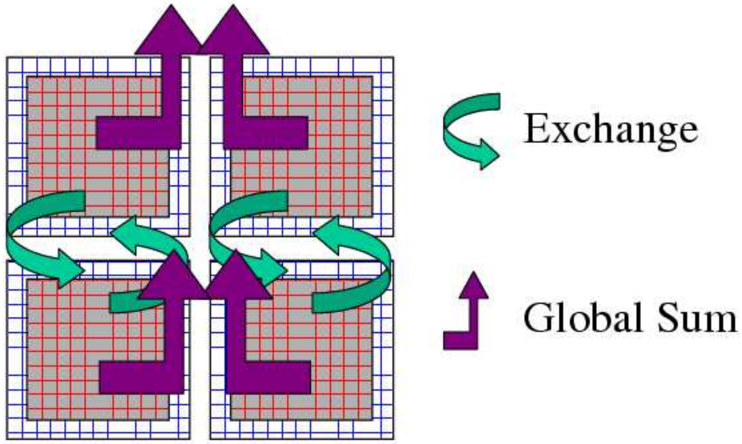
$$\frac{\text{largest EV}}{\text{smallest EV}} = \text{conditioning number}$$

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

AD challenges: (II) Scalability

- domain decomposition (tiles) & overlaps (halos)
- split into extensive on-processor and global phase





Global communication/arithmetic op.'s supported by MITgcm's intermediate layer (WRAPPER) which need hand-written adjoint forms

	operation/primitive	forward		reverse
•	communication (MPI,...):	send	⟷	receive
•	arithmetic (global sum,...):	gather	⟷	scatter
•	active parallel I/O:	read	⟷	write

AD challenges: (III) Parameterization schemes

- Parameterization schemes required for
  - turbulence closure for nonlinear Navier-Stokes equations
  - subgrid-scale processes
- Main issues:
  - nonlinear expressions
    (momentum advection, nonlinear equation of state, ...)
  - State-dependent conditional statements (thresholds and jumps)
  - small numbers `(SQRT(.), 1/x, ...)`; non-differentiable points
  - combinations thereof
- e.g.: many forms of mixing processes
  (shear instability, convection, diffusion, ...)
- Require signifi cant attention to obtain stable adjoint solution!

N.B.: Currently, common method to circumvene these problems is to exclude exact adjoint of parameterization schemes

Example: Limits on boundary layer depth in KPP

- for neutral stratifi cation: H_{bl} should be smaller than both h_e and L
- generally: H_{bl} should be larger than some minimal value

```
CADJ STORE hbl, bfsfc TO TAPE
do i = 1, Nx * Ny
  if (bfsfc(i).gt.0.0) then
     hekman  =  cekman * ustar(i) / max(abs(Coriol(i)), eps)
     hmonob  =  cmonob * ustar(i) * ustar(i) * ustar(i)
&                / vonk / bfsfc(i)
     hlimit  =
&                stable(i) * min(hekman, hmonob)
&                 + (stable(i) - 1.) * zgrid(Nr)

     hbl(i)  =  min(hbl(i), hlimit)
  end if
  hbl(i)  =  max(hbl(i), minKPPhbl)
end do
```

$h_e = 0.7 u^* / f$

$L = {u^*}^3 / (\kappa B_f)$

h_{bl} *limit for:*

– stable case

– unstable case

apply upper limit

apply lower limit

Example: Limits on boundary layer depth in KPP (cont'd)

```
CADJ RESTORE hbl_1   FROM TAPE
CADJ RESTORE bfsfc   FROM TAPE

do i = 1, Nx*Ny
  adhbl(i) = adhbl(i)*(0.5+sign(0.5d0,hbl_2(i)-minkpphbl))
  if (bfsfc(i) .gt. 0.) then
!      recompute hekman, hmonob, hlimit
    adhlimit = adhlimit+adhbl(i)*(0.5-sign(0.5d0,hlimit-hbl_1(i)))
    adhbl(i) = adhbl(i)*(0.5+sign(0.5d0,hlimit-hbl_1(i)))
    adhekman = adhekman+adhlimit*stable(i)*(0.5+sign(0.5d0,hmonob-hekman))
    adhmonob = adhmonob+adhlimit*stable(i)*(0.5-sign(0.5d0,hmonob-hekman))
    adstable(i) = adstable(i)+adhlimit*(zgrid(nr)+min(hekman,hmonob))
    adhlimit = 0.d0
    ...
  endif
end do
```

ECCO state estimation: Control variables

- Initial values (temperature, salinity, passive tracer)
- time-dependent surface forcing
  (either air-sea fluxes or atmospheric fi elds plus bulk formulae)
  - heat flux (or surface air temperature)
  - freshwater flux (or atmos. humidity)
  - zonal/meridional windstresses (or surface wind speeds)
- time-dependent open-boundary values
- background mixing coeffi cient
- Eliassen-Palm fluxes
- bottom topography (in progress)
- ...

ECCO state estimation: observational elements

$\mathcal{J} = (\bar{\eta} - \bar{\eta}_{TP})^t \, \mathbf{W_{geoid}} \, (\bar{\eta} - \bar{\eta}_{TP})$	TOPEX absolute SSH
$+ \left(\eta - \eta'_{TP}\right)^t \mathbf{W_{TP}} \left(\eta - \eta'_{TP}\right)$	TOPEX SSH anomalies
$+ \left(\eta - \eta'_{ERS}\right)^t \mathbf{W_{ERS}} \left(\eta - \eta'_{ERS}\right)$	ERS SSH anomalies
$+ \left(\bar{T}_{surf} - \bar{T}_{Reyn}\right)^t \mathbf{W_{SST}} \left(\bar{T}_{surf} - \bar{T}_{Reyn}\right)$	Reynolds SST
$+ \left(\bar{T} - \bar{T}_{Lev}\right)^t \mathbf{W_{T_{Lev}}} \left(\bar{T} - \bar{T}_{Lev}\right)$	Levitus clim.
$+ \left(\bar{S} - \bar{S}_{Lev}\right)^t \mathbf{W_{S_{Lev}}} \left(\bar{S} - \bar{S}_{Lev}\right)$	Levitus clim.
$+ \left(\tau_x - \tau_{x,NCEP}\right)^t \mathbf{W}_{\tau_\mathbf{x}} \left(\tau_x - \tau_{x,NCEP}\right)$	zonal wind stress
$+ \left(\tau_y - \tau_{y,NCEP}\right)^t \mathbf{W}_{\tau_\mathbf{y}} \left(\tau_y - \tau_{y,NCEP}\right)$	merid. wind stress
$+ \left(H_Q - H_{Q,NCEP}\right)^t \mathbf{W_{H_Q}} \left(H_Q - H_{Q,NCEP}\right)$	NCEP heat flux
$+ \left(H_F - H_{F,NCEP}\right)^t \mathbf{W_{H_F}} \left(H_F - H_{F,NCEP}\right)$	NCEP freshwater flux

Currently added:

- Jason-1 altimetry (sea surface height)
- WOCE hydrography, XBT, TAO buoys
- PALACE/ARGO tracer profiles and drift velocities
- surface drifter velocities
- NSCAT/QuickScat surface wind stress fields
- TRMM/TMI tropical surface temperature fields

ECCO state estimation: problem size

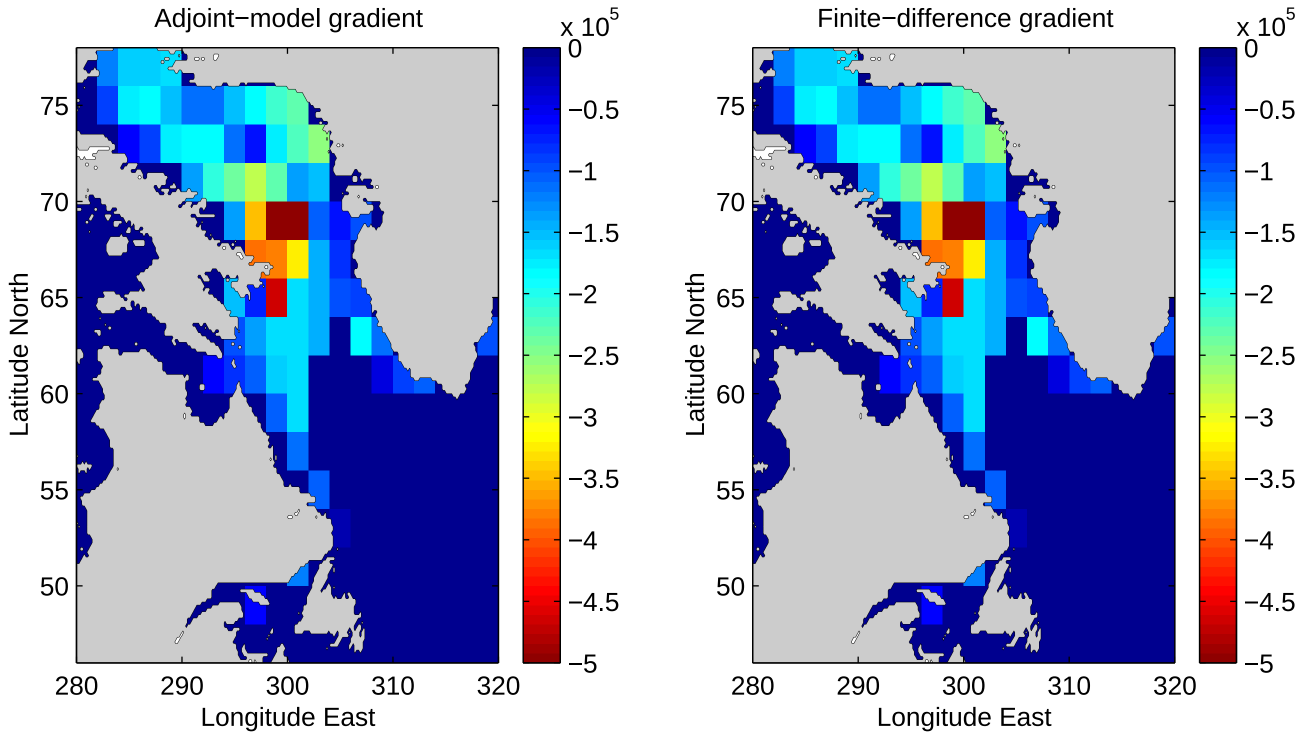
- Dimensionality:
  - grid @ $1^\circ \times 1^\circ$ resolution: $n_x \cdot n_y \cdot n_z$ = 360 · 160 · 23 $1,324,800$
  - model state: 17 3D + 2 2D fi elds $\sim 2 \cdot 10^7$
  - timesteps: 10 years @ 1-hour time step $87,600$
  - control vector $\sim 1 \cdot 10^8$
    - initial temperature (T), salinity (S)
    - time-dependent surface forcing (every 2 days)
  - cost function: observational elements: $\sim 1 \cdot 10^8$
- Computational size:
  - 60 processors (15 nodes) @ 512MB per proc.
  - I/O: 10 GB input, 35GB output
  - time: 59 hours per iteration @ 60 processors
- What we would ideally want:
  - $1/10^\circ \times 1/10^\circ$ resol., 1000 years, full model error covariance ...

Sea-ice model description

- Sea-ice model based on *Hibler (1979,1980)* has been added to MIT/ECCO adjoint-model estimation infrastructure.
- Snow is simulated as per *Zhang et al. (1998)*.
- Dynamics are governed by viscous-plastic rheology and solved with numerical method of *Zhang and Rothrock (2000)*.
- Relatively simple 2-category sea-ice model is chosen to to simplify adjoint-model development and to reduce computational cost.
- References:

  - W. D. Hibler, III, 1979. A dynamic thermodynamic sea ice model. J. Phys. Oceanogr., 9:815.
  - W. D. Hibler, III, 1980. Modeling a variable thickness sea ice cover. Mon. Wea. Rev., 1:1943.
  - J. Zhang, W. D. Hibler, III, M. Steele, and D. A. Rothrock, 1998. Arctic ice-ocean modeling with and without climate restoring. J. Phys. Oceanogr., 28:191.
  - J. Zhang and D. A. Rothrock, 2000. Modeling arctic sea ice with an effi cient plastic solution. J. Geophys. Res., 105:3325.

Coupled ocean-sea-ice model: adjoint sensitivity experiment



- Preliminary test of the coupled model (sea-ice thermodynamics only).
- Sensitivity of sea-ice volume in the Labrador Sea to surface atmospheric temperature perturbations over a 4 hour integration period in units of $m^3/^\circ C$.
- Left panel: adjoint-derived gradient,
  right panel: perturbation of surface atmospheric temperature at each location.
- The small difference between the two panels, less than one part in 10^5, demonstrates the accuracy of the adjoint-model solution.

Coupled ocean-sea-ice model: work in progress

- The coupled ocean-sea-ice adjoint model provides accurate results in the small test domain only for up to 10-day integrations.

  For longer integrations the forward-model gradient is ill-defi ned. Therefore it cannot be computed using the adjoint method.

  Work is underway to simplify the sea-ice adjoint model in order to permit longer integrations.

- Accuracy of dynamic solver needs to be increased and computational cost decreased for adjoint-model computations.

- Projection operators for sea ice data need to be written, and corresponding a priori errors determined.

Other issues

- impact of *nonlinearities, discontinuities* need further analyses
  What are limits of applicability for high-resolution, long-term integrations?
- cost regularization, gradient preconditioning, and model error analysis
  $\longrightarrow$ compute 2nd derivative (Hessian)
- *bulk formulae:* flux controls vs. atmospheric state controls
- *atmospheric setup:*
  - dynamic setup (Held-Suarez like) is adjointed
  - cubed-sphere needs update of adjoint components of WRAPPER
  - adjoint of physics packages will require (quite) some work
  - coupling
- *ESMF / PRISMA:* adjoint correspondents of coupler primitives

Conclusions & Outlook – Science aspects

- Global ocean state estimation via adjoint method is feasible,
  - yields a dynamical consistent model trajectory and parameter estimates
  - uses data in an 'optimal' way
  - yields a quantitative measure of model vs. data misfit
- State estimation at the eddy-resolving scale remains subject to research/discussion
- Incorporation of different data sets can reveil new insights into model vs. data incompatibility/inconsistency (and its causes)
- Careful analysis of 'adjoint state' yields complementary info on model behaviour (assertions on which can be tested)
- Inclusion of model error remains untackled problem
- Sea ice state estimation has just started; despite remaining problems with parameterizations, first results look promising; is expected to greatly enhance current state estimation system.

Conclusions & Outlook – Adjoint code generation

- *Exact scalable adjoint* code generation is feasible
- AD tools are indispensable for *evolving code* environment
- Nevertheless, *challenges remain* despite AD
  efficient adjoint code generation is **semi-automatic** so far
- Code development should have AD "in mind"
- Libraries/couplers need derivative forms (ESMF, PRISM)
- Room for AD tool improvement has led to
  NSF Information Technology Research (ITR) project:
  Adjoint Compiler Technology & Standard (ACTS)

  common platform that should facilitate AD tool
  development/improvement by larger community