An efficient exact adjoint of the parallel MIT general circulation model, generated via automatic differentiation

Patrick Heimbach[1], Chris Hill[1] and Ralf Giering[2]

[1] MIT / EAPS, Cambridge (MA), USA — E-mail: {heimbach, hill}@mit.edu — URL: http://mitgcm.org
[2] FastOpt, Hamburg, Germany — E-mail: ralf.giering@fastopt.de — URL: http://www.fastopt.de

1 Introduction

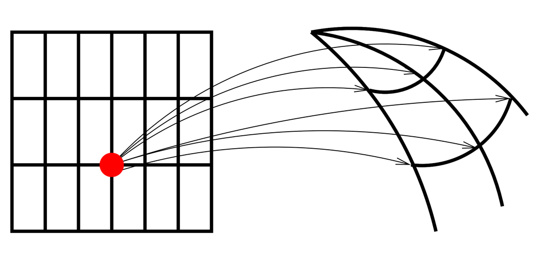
(I) Ocean State Estimation [1, 2]

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

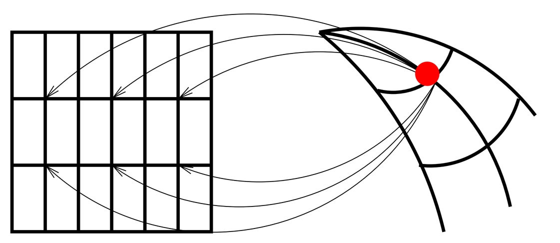
$$\vec{u}^{n+1} = \vec{u}^n + \Delta\vec{u}$$

(II) Sensitivity analysis [3]

- Finite difference approach:
  - Take a "guessed" anomaly (e.g. $\mathrm{SST}(x,y,t)$) and determine its impact on some model output (e.g. MOC)
  - Perturb each input element ($\mathrm{SST}(i,j)$) to determine its impact on output (MOC).
- Reverse/adjoint approach:
  - Calculates "full" sensitivity field $$\frac{\partial\,\mathrm{MOC}}{\partial\,\mathrm{SST}(x,y,t)}$$
  - Approach: $\mathcal{J} = \mathrm{MOC}$, $\vec{u} = \mathrm{SST}$ $\longrightarrow \vec{\nabla}_u \mathcal{J}(\vec{u}) = \frac{\partial\,\mathrm{MOC}}{\partial\,\mathrm{SST}(x,y,t)}$



finite difference / forward approach



adjoint / reverse approach

2 The Adjoint Method

- Consider least-square model vs. data misfit cost function:

$$\mathcal{J}(\mathcal{M}(\vec{u})) = \left(\vec{\mathcal{H}}(\vec{v}) - \vec{d}\right)^T W^{obs} \left(\vec{\mathcal{H}}(\vec{v}) - \vec{d}\right) + \left(\vec{u} - \vec{u}^{(0)}\right)^T W^u \left(\vec{u} - \vec{u}^{(0)}\right)$$

- $\vec{v} = \mathcal{M}(\vec{u})$ nonlinear model
- M, M^T tangent linear (TLM) / adjoint (ADM)
- $\delta\vec{u}$, $\delta\vec{u}^*$ perturbation / dual (or sensitivity)
- $\mathcal{H}$, H^T data projection operator / adjoint thereof
- W^{obs}, W^u inverse error weights (metric)

- Need gradient of $\mathcal{J} \in I\!R^1$ w.r.t. control variables $\vec{u} \in I\!R^m$

$$\min_u \mathcal{J}(\vec{u}) \quad\Rightarrow\quad \vec{\nabla}_C \mathcal{J}(\vec{u}, \mathcal{M}(\vec{u}))\Big|_{\vec{u}=\vec{u}^{opt}} = 0$$

- Use the adjoint method to compute gradient

$\mathcal{J}$:	$\vec{u}$	$\longmapsto$	$\vec{v} = \mathcal{M}(\vec{u})$	$\longmapsto$	$\mathcal{J}(\mathcal{M}(\vec{u}))$
TLM :	$\delta\vec{u}$	$\longmapsto$	$\delta\vec{v} = M \cdot \delta\vec{u}$	$\longmapsto$	$\delta\mathcal{J} = \vec{\nabla}_u\mathcal{J} \cdot \delta\vec{u}$
ADM :	$\delta\vec{u}^* = \vec{\nabla}_u\mathcal{J}^T$	$\longleftarrow$	$\delta\vec{v}^*$	$\longleftarrow$	$\delta\mathcal{J}$

$$\vec{\nabla}_u\mathcal{J}^T|_{\vec{u}} = 2\,M^T \cdot H^T \cdot W \cdot \left(\mathcal{H}(\vec{v}) - \vec{d}\right) \sim (\text{adjoint model}) \cdot (\text{model} - \text{data})$$

TLM : $\mathbf{m}$ $(\sim n_x n_y n_z)$ integrations @ $1\cdot$ (#forward)
ADM : 1 integration @ $\gamma\cdot$ (#forward)

3 Automatic Differentiation (AD)

Model code	Adjoint code
$\vec{v} = \mathcal{M}_\Lambda(\mathcal{M}_{\Lambda-1}(\ldots\ldots(\mathcal{M}_0(\vec{u}))))$	$\delta\vec{u}^* = M_0^T \cdot M_1^T \cdot \ldots\ldots \cdot M_\Lambda^T \cdot \delta\vec{v}^*$

- 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 [4]

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

$\xrightarrow{\text{TAMC/TAF}}$

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

AD and *tangent* linearity

Consider the nonlinear function

$$f(x,y,z) = 3x + a\,xy^2 + b\,z$$

Its differential will be of the form

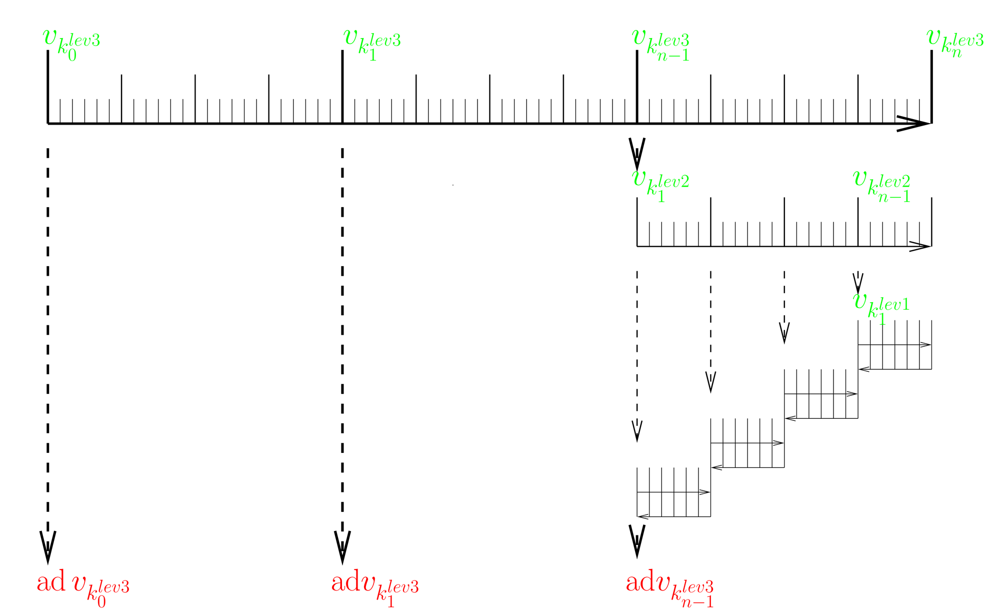
$$\mathrm{d}f = \left(3 + a\,y^2|_{y_0}\right)\mathrm{d}x + \left(2a\,x|_{x_0}\,y|_{y_0}\right)\mathrm{d}y + b\,\mathrm{d}z$$

$\Longrightarrow$ state at $(x = x_0, y = y_0)$ has to be known to evaluate $\mathrm{d}f$.

$\Longrightarrow$ evaluation of derivative is tangent *at each point along nonlinear trajectory*

5 Flow reversal and Checkpointing

- *Adjoint = transpose of TLM*
  $\rightarrow$ evaluated in reverse order
  $\rightarrow$ all state stored or recomputed
- *Solution: Checkpointing*
  e.g. Griewank, 1992
  $\rightarrow$ balances storing vs. recomputation



- **e.g. 3-level checkpointing:**

$$n_T = n_1 \cdot n_2 \cdot n_3$$

$\rightarrow$ *Storing:* reduced from

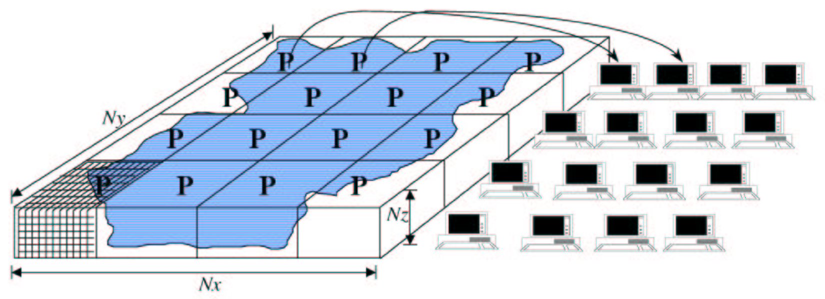
$$n_1 \cdot n_2 \cdot n_3 \quad \text{to} \quad \begin{cases} n_2 + n_3 & \text{to disk} \\ n_1 & \text{to memory} \end{cases}$$

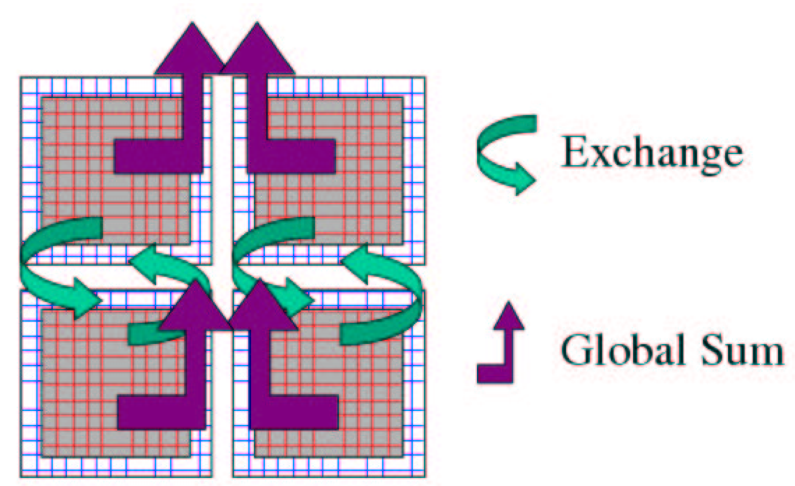
$\rightarrow$ *CPU:* $3 \cdot \mathrm{forward} + 1 \cdot \mathrm{adjoint} \approx 5.5 \cdot \mathrm{forward}$

- **adjoint dump & restart closely related!**

6 Adjoint Code 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	$\longleftrightarrow$	receive
arithmetic (global sum,...):	gather	$\longleftrightarrow$	scatter
active parallel I/O:	read	$\longleftrightarrow$	write

7 Example: Boundary layer limits in KPP

Implement following criteria:

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

Model code:

```
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
```

$h_e = 0.7u^*/f$
$L = u^{*3}/(\kappa B_f)$

h_{bl} limit for:
– stable case
– unstable case
apply upper limit

apply lower limit

Adjoint code:

```
  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
c ------> 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
  enddo
```

hbl_2 NOT available

hbl_1 available
hbl_1 available

8 Conclusions:

- Scalable adjoint code generation is feasible for fully-fledged models [5, 6]
- AD tools indispensable for ongoing code development
- Nevertheless, challenges remain despite AD:
  Efficient adjoint code generation is at best **semi-automatic** so far!
- Code development should have AD "in mind"
- Libraries/couplers need derivative forms (e.g. the ESMF, PRISMA model coupling projects)
- AD tools need to be improved

ACTS: Adjoint Compiler Technology & Standard
AD tools ought to be

- open-source,
- easier to use,
- transparent,
- common platform that facilitates AD tool development by many

References

[1] Stammer, D., Wunsch, C., Giering, R., Eckert, C., Heimbach, P., Marotzke, J., Adcroft, A., Hill, C., Marshall, J.: The global ocean circulation and transports during 1992 – 1997, estimated from ocean observations and a general circulation model. J. Geophys. Res. **107(C9)** (2002a) 3118

[2] Stammer, D., Wunsch, C., Giering, R., Eckert, C., Heimbach, P., Marotzke, J., Adcroft, A., Hill, C., Marshall, J.: 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. (2002b) in press.

[3] Marotzke, J., Giering, R., Zhang, K., Stammer, D., Hill, C., Lee, T.: Construction of the adjoint MIT ocean general circulation model and application to Atlantic heat transport variability. J. Geophys. Res. **104, C12** (1999) 29,529–29,547

[4] Giering, R., Kaminski, T.: Recipes for adjoint code construction. ACM Transactions on Mathematical Software **24** (1998) 437–474

[5] Heimbach, P., Hill, C., Giering, R.: Automatic generation of efficient adjoint code for a parallel navier-stokes solver. In J.J. Dongarra, P.M.A. Sloot and C.J.K. Tan, ed.: Computational Science – ICCS 2002. Volume 2331 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (Germany) (2002a) 1019–1028

[6] Heimbach, P., Hill, C., Giering, R.: An efficient exact adjoint of the parallel mit general circulation model, generated via automatic differentiation. Future Generation Computer Systems (FGCS) (2002b) submitted.

Acknowledgements

This is a contribution to the ECCO project, supported by NOPP, and with funding from NASA, NSF and ONR. Many members of the ECCO Consortium under the lead of D. Stammer have contributed.