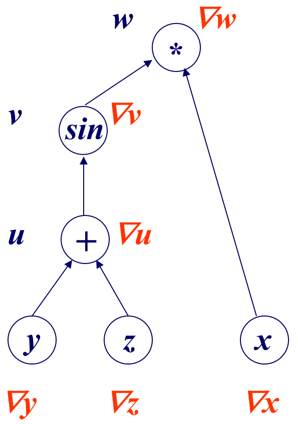
Forward Mode – FM

Example code f***:*** $w = x * \sin(y+z)$

Original code and derivative statements:



$\nabla u = \nabla y + \nabla z$

$u = y + z$

$\nabla v = \cos(u) * \nabla u$

$v = \sin(u)$

$\nabla w = \nabla v * x + v * \nabla x$

$w = v * x$

If $(\nabla x, \nabla y, \nabla z) = (1, 0, 0)$

$\rightarrow$ ∇w ***computes*** $\partial w / \partial x$

Each variable t ***is associated with a derivative object*** ∇t

84

Example

X	Y	Z	
1	0	0	-> dw/dx
0	1	0	-> dw/dy
0	0	1	-> dw/dz

vector * J

1	1	1

dw/dx + dw/dy + dw/dz

vector * J

85

86

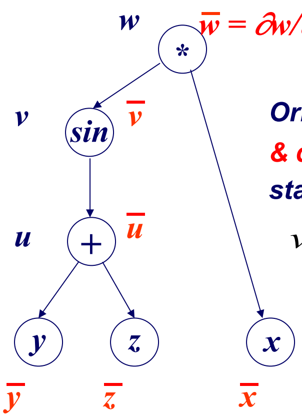
Forward Mode Facts

- Computes $\partial f/\partial t$*S (S is called "seed matrix" – [∇x, ∇y, ∇z]) by propagating sensitivities of *intermediate values* with respect to *input values*
- For *p* input values of interest, ***runtime*** and ***memory*** scale approximately with *p*. May be much less e.g. for **sparse Jacobians**
- FM is appropriate for moderate *p*'s

87

Reverse Mode - RM

The same example code f: $w = x * \sin(y+z)$



Original code & derivative statements:

$u = y + z$
$v = \sin(u)$
$w = v * x$
$\overline{x} \mathrel{+}= \overline{w} * v$
$\overline{v} \mathrel{+}= \overline{w} * x$
$\overline{u} \mathrel{+}= \overline{v} * \cos(u)$
$\overline{z} \mathrel{+}= \overline{u}$
$\overline{y} \mathrel{+}= \overline{u}$

$$v_k = f_k(\ldots, v_j, \ldots)$$
$$\overline{v_j} = \overline{v_k} \cdot \sum_k \frac{\partial f_k}{\partial v_j}$$

If $\overline{w} = 1 \rightarrow (\overline{x}, \overline{y}, \overline{z})$ contains
$\nabla w = (\partial w/\partial x, \partial w/\partial y, \partial w/\partial z)$

Each variable t is associated with an adjoint object $\overline{t}$, the differentiated expressions are accumulated in the reverse order

Reverse Mode Facts

- Computes $W^T * \partial f / \partial t$ by propagating sensitivities of *output values* with respect to *intermediate values*
- For q output values of interest, the ***runtime*** scales with q. ***Memory requirements*** are harder to predict → greatly depend on the structure & implementation of program/AD-tool
- RM is great for computing „**long gradients**" – small q's and big p's

Forward & Reverse Mode Example

given function $f(x,y,z) = (xy + \cos z)(x^2 + 2y^2 + 3z^2)$, the partial derivatives are

$$\frac{\partial f}{\partial x} = y \cdot (x^2 + 2y^2 + 3z^2) + (xy + \cos z) \cdot 2x = 3x^2 y + 2y^3 + 3yz^2 + 2x\cos z,$$

$$\frac{\partial f}{\partial y} = x \cdot (x^2 + 2y^2 + 3z^2) + (xy + \cos z) \cdot 4y = x^3 + 6xy^2 + 3xz^2 + 4y\cos z,$$

$$\frac{\partial f}{\partial z} = -\sin z \cdot (x^2 + 2y^2 + 3z^2) + (xy + \cos z) \cdot 6z$$

$$= -x^2 \sin z - 2y^2 \sin z - 3z^2 \sin z + 6xyz + 6z\cos z.$$

$$\nabla f = \left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \quad \frac{\partial f}{\partial z} \right]^T$$

Forward Mode

Code list + Gradient entries

Code list	Gradient entries
$u_1 = x,$	$\nabla u_1 = [1,0,0],$
$u_2 = y,$	$\nabla u_2 = [0,1,0],$
$u_3 = z,$	$\nabla u_3 = [0,0,1],$
$u_4 = u_1 u_2,$	$\nabla u_4 = u_1 \nabla u_2 + u_2 \nabla u_1 = [0,u_1,0]+[u_2,0,0]=[u_2,u_1,0],$
$u_5 = \cos u_3,$	$\nabla u_5 = (-\sin u_3)\nabla u_3 = [0,0,-\sin u_3],$
$u_6 = u_4 + u_5,$	$\nabla u_6 = \nabla u_4 + \nabla u_5 = [u_2,u_1,-\sin u_3],$
$u_7 = u_1^2,$	$\nabla u_7 = 2u_1 \nabla u_1 = [2u_1,0,0],$
$u_8 = 2u_2^2,$	$\nabla u_8 = 4u_2 \nabla u_2 = [0,4u_2,0],$
$u_9 = 3u_3^2,$	$\nabla u_9 = 6u_3 \nabla u_3 = [0,0,6u_3],$
$u_{10} = u_7 + u_8 + u_9,$	$\nabla u_{10} = \nabla u_7 + \nabla u_8 + \nabla u_9 = [2u_1, 4u_2, 6u_3],$
$u_{11} = u_6 u_{10},$	$\nabla u_{11} = u_6 \nabla u_{10} + u_{10} \nabla u_6 = [2u_6 u_1 + u_{10} u_2, 4u_6 u_2 + u_{10} u_1, 6u_6 u_3 - u_{10} \sin u_3].$

$$\boxed{\begin{aligned}\nabla f(x,y,z) = \nabla u_{11} = [&3x^2 y + 2x\cos z + 2y^3 + 3yz^2,\\ &6xy^2 + 4y\cos z + x^3 + 3xz^2,\\ &6xyz + 6z\cos z - x^2 \sin z - 2y^2 \sin z - 3z^2 \sin z].\end{aligned}}$$

Correct!

Reverse Mode

Code list + Adjoints

Code list	Adjoints
$u_1 = x,$	$\frac{\partial u_{11}}{\partial u_{11}} = 1, \ \frac{\partial u_{11}}{\partial u_{10}} = u_6, \ \frac{\partial u_{11}}{\partial u_9} = \frac{\partial u_{11}}{\partial u_{10}}\frac{\partial u_{10}}{\partial u_9} = u_6,$
$u_2 = y,$	$\frac{\partial u_{11}}{\partial u_8} = \frac{\partial u_{11}}{\partial u_{10}}\frac{\partial u_{10}}{\partial u_8} = u_6, \ \frac{\partial u_{11}}{\partial u_7} = \frac{\partial u_{11}}{\partial u_{10}}\frac{\partial u_{10}}{\partial u_7} = u_6,$
$u_3 = z,$	$\frac{\partial u_{11}}{\partial u_6} = u_{10}, \ \frac{\partial u_{11}}{\partial u_5} = \frac{\partial u_{11}}{\partial u_6}\frac{\partial u_6}{\partial u_5} = u_{10}, \ \frac{\partial u_{11}}{\partial u_4} = \frac{\partial u_{11}}{\partial u_6}\frac{\partial u_6}{\partial u_4} = u_{10},$
$u_4 = u_1 u_2,$	$\frac{\partial u_{11}}{\partial u_3} = \frac{\partial u_{11}}{\partial u_9}\frac{\partial u_9}{\partial u_3} + \frac{\partial u_{11}}{\partial u_5}\frac{\partial u_5}{\partial u_3} = 6u_6 u_3 - u_{10}\sin u_3,$
$u_5 = \cos u_3,$	$\frac{\partial u_{11}}{\partial u_2} = \frac{\partial u_{11}}{\partial u_4}\frac{\partial u_4}{\partial u_2} + \frac{\partial u_{11}}{\partial u_8}\frac{\partial u_8}{\partial u_2} = u_{10} u_1 + 4u_6 u_2,$
$u_6 = u_4 + u_5,$	$\frac{\partial u_{11}}{\partial u_1} = \frac{\partial u_{11}}{\partial u_4}\frac{\partial u_4}{\partial u_1} + \frac{\partial u_{11}}{\partial u_7}\frac{\partial u_7}{\partial u_1} = u_{10} u_2 + 2u_6 u_1.$
$u_7 = u_1^2,$	
$u_8 = 2u_2^2,$	
$u_9 = 3u_3^2,$	
$u_{10} = u_7 + u_8 + u_9,$	
$u_{11} = u_6 u_{10},$	

$$\boxed{\begin{aligned}\nabla f(x,y,z) = [\frac{\partial u_{11}}{\partial u_1}, \frac{\partial u_{11}}{\partial u_2}, \frac{\partial u_{11}}{\partial u_3}] = [&3x^2 y + 2x\cos z + 2y^3 + 3yz^2,\\ &6xy^2 + 4y\cos z + x^3 + 3xz^2,\\ &6xyz + 6z\cos z - x^2 \sin z - 2y^2 \sin z - 3z^2 \sin z].\end{aligned}}$$

Correct!

Divided Differences

First order differentiation

Forward differentiation: $\frac{\partial f}{\partial x_m} = \frac{f(x_1,\dots,x_m+h,\dots,x_n) - f(x_1,\dots,x_m,\dots,x_n)}{h} + O(h)$

Backward differentiation: $\frac{\partial f}{\partial x_m} = \frac{f(x_1,\dots,x_m,\dots,x_n) - f(x_1,\dots,x_m-h,\dots,x_n)}{h} + O(h)$

Centered differentiation: $\frac{\partial f}{\partial x_m} = \frac{f(x_1,\dots,x_m+h,\dots,x_n) - f(x_1,\dots,x_m-h,\dots,x_n)}{2h} + O(h^2)$

Second order differentiation

$$\frac{\partial^2 f}{\partial x_m^2} = \frac{f(x_1,\dots,x_m+h,\dots,x_n) - 2f(x_1,\dots,x_m,\dots,x_n) + f(x_1,\dots,x_m-h,\dots,x_n)}{h^2} + O(h^2)$$

92

Which mode to use?

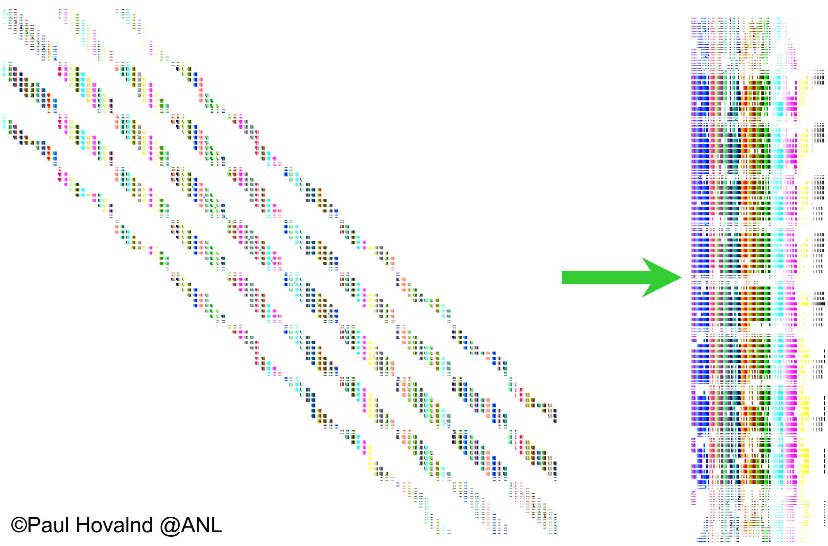
- Use **forward mode** when
  - \# **independents** is very small
  - Only a directional derivative **Jv** is needed
  - Reverse mode is not tractable
- Use **reverse mode** when
  - \# **dependents** is very small
  - Only $\mathbf{J}^T\mathbf{v}$ is needed

93

Case Study – Matrix Coloring

- Jacobian matrices are often sparse
- The forward mode of AD computes J × S, where S is usually an identity matrix or a vector
- One can "**compress**" Jacobian by choosing S such that *structurally orthogonal* columns are combined
- A set of columns are structurally orthogonal if no two of them have nonzeros in the same row
- Equivalent problem: color the graph whose adjacency matrix is J^TJ
- Equivalent problem: distance-2 color the bipartite graph of J

Compressed Jacobian



96

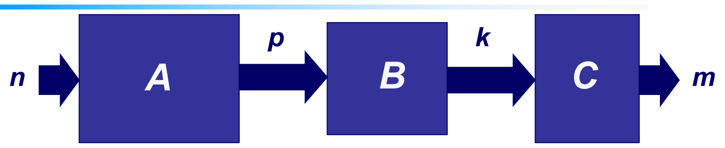
What is feasible & practical

- Key point:
  - Forward mode computes J · S at cost proportional to number of columns in S
  - Reverse mode computes J^T · W at cost proportional to number of columns in W
- Jacobians of functions
  - Small number (1—1000) of **independent** variables (FM)
  - Small number (1—100) of **dependent** variables (RM)
- Extremely large, but (very) sparse Jacobians and Hessians
- Jacobian-vector products (forward mode)
- Transposed-Jacobian-vector products (adjoint mode)
- Hessian-vector products (forward + adjoint modes)

96

97

Scenarios



- n small: use FM on full computation
- m small: use RM on full computation
- m & n large, p small: use RM on A, FM on B&C
- m & n large, k small: use RM on A&B, FM on C
- n, p, k, m large, Jacobians of A, B, C sparse: compressed FM
- n, p, k, m large, Jacobians of A, B, C low rank: scarce FM
- n, p, k, m large: Jacobians of A, B, C dense: what to do?

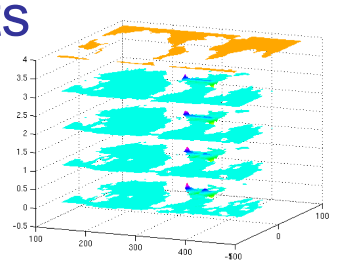
97

Issues with Black Box Differentiation

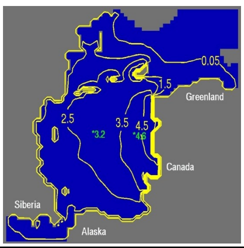
- Source code may not be available or may be difficult to work with
- Simulation may not be (chain rule) differentiable
  - Feedback due to adaptive algorithms
  - Non-differentiable functions
  - Noisy functions
  - Convergence rates
  - Etc.
- Accurate derivatives may not be needed – FD might be cheaper
- Differentiation and discretization do not commute

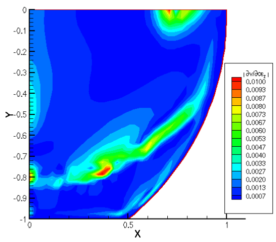
98

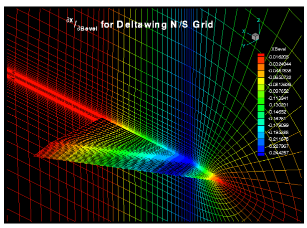
Application highlights



- Atmospheric chemistry
- Breast cancer biostatistical analysis
- CFD: CFL3D, NSC2KE, Fluent 4.52
- Chemical kinetics
- Climate and weather: MITGCM, MM5, CICE
- Semiconductor device simulation
- Water reservoir simulation

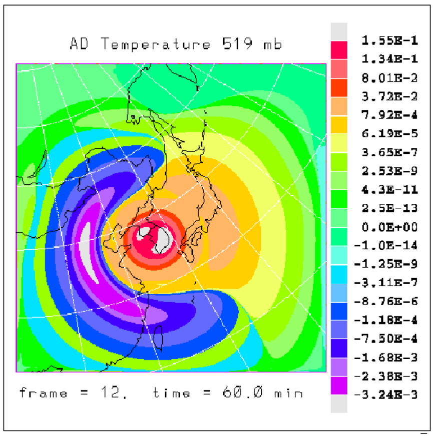


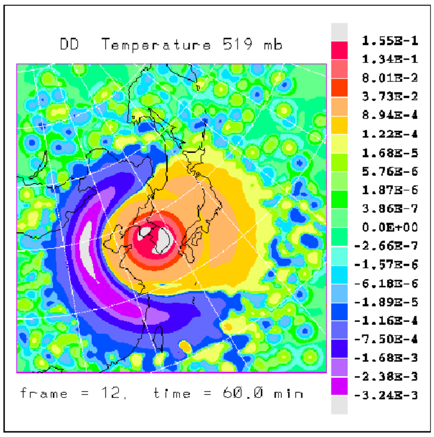




99

Sensitivity Analysis: Mesoscale Weather Modeling





100

AD Conclusions & Future Work

- Automatic differentiation research involves a wide range of combinatorial problems
- AD is a powerful tool for scientific computing
- Modern automatic differentiation tools are robust and produce efficient code for complex simulation codes
  - Requires an industrial-strength compiler infrastructure
  - Efficiency requires sophisticated compiler analysis
- Effective use of automatic differentiation depends on insight into problem structure
- Future Work
  - Further develop and test techniques for computing Jacobians that are effectively sparse or effectively low rank
  - Develop techniques to automatically generate complex and adaptive checkpointing strategies

101