OPM Flow – overview and demonstration

Atgeirr Flø Rasmussen

SINTEF Digital, Mathematics and Cybernetics

MSO4SC workshop
23rd May 2017

Overview of talk

Reservoir simulation

Mathematical formulation

Implementation with automatic differentiation

What next?

Overview of talk

Reservoir simulation

Mathematical formulation

Implementation with automatic differentiation

What next?

What is reservoir simulation

Simulation of *porous medium* flow in *subsurface reservoirs*.

Examples of use:

- Energy industry
  - Forecasting and optimizing oil and gas production.
  - Forecasting and optimizing geothermal energy production.
- Environmental management
  - Groundwater flows and pollutants
  - CO_2storage

Reservoir simulators solve *systems of nonlinear PDEs* that are coupled to *well/facility models*.

Why is it hard?

- Porous medium is strongly heterogeneous and anisotropic.

Why is it hard?

- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.

Why is it hard?

- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.

Why is it hard?

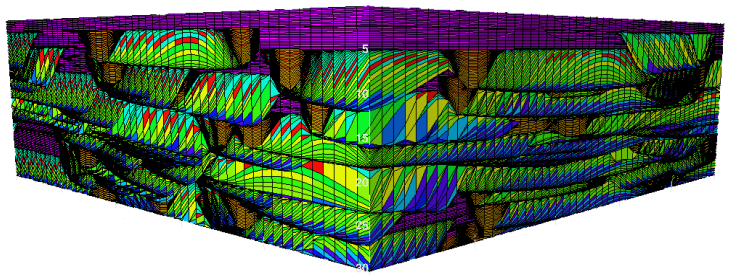
- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.
- Coupling to wells can connect regions that are far away from each other.

Why is it hard?

- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.
- Coupling to wells can connect regions that are far away from each other.
- The models are highly nonlinear.

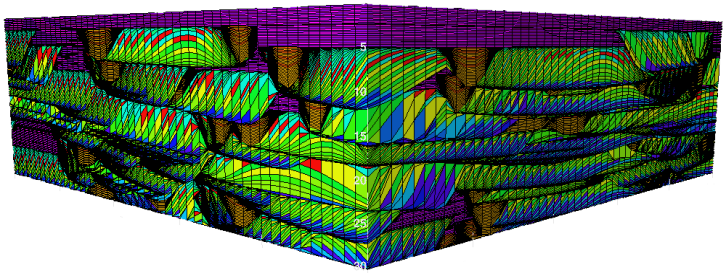
Why is it hard?

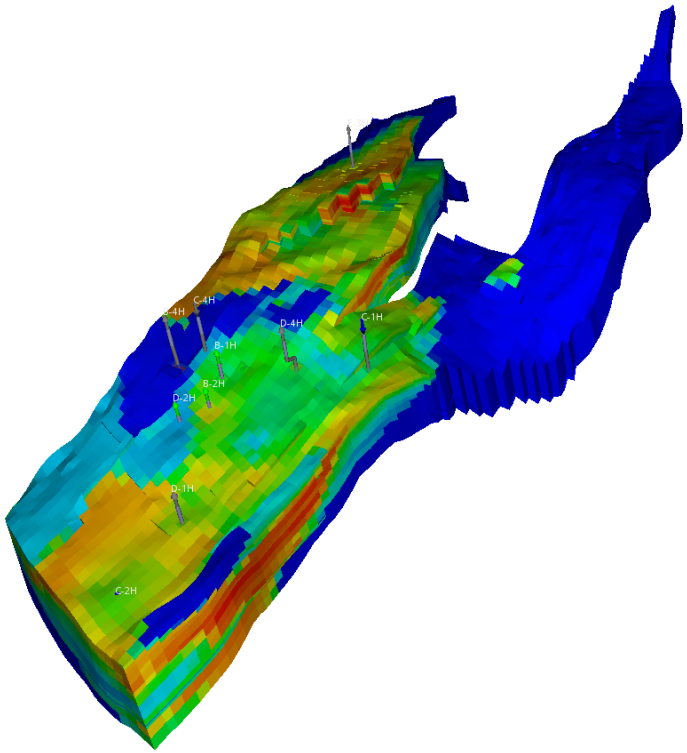
- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.
- Coupling to wells can connect regions that are far away from each other.
- The models are highly nonlinear.



Why is it hard?

- Porous medium is strongly heterogeneous and anisotropic.
- Grids with high aspect ratio, fully unstructured, polygonal cells.
- Nontrivial phase behaviour. Phases can appear and disappear as fluid components dissolve or vaporize.
- Coupling to wells can connect regions that are far away from each other.
- The models are highly nonlinear.





Why does it require HPC/Cloud

- Model sizes increasing (Saudi-Aramco record: 1e12 cells)
- Model complexity increasing (well/facility models, fluid models)
- New mechanisms (polymer, CO_2) require better resolved fronts
- Large ensembles for history matching, optimization, uncertainty quantification

The market situation

Commercial reservoir simulators (expensive but comprehensive):

- ECLIPSE, Intersect (Schlumberger)
- IMEX, GEM, STARS (CMG)
- Nexus (Landmark)
- tNavigator (Rock Flow Dynamics)
- ...

In-house simulators (unavailable to outsiders):

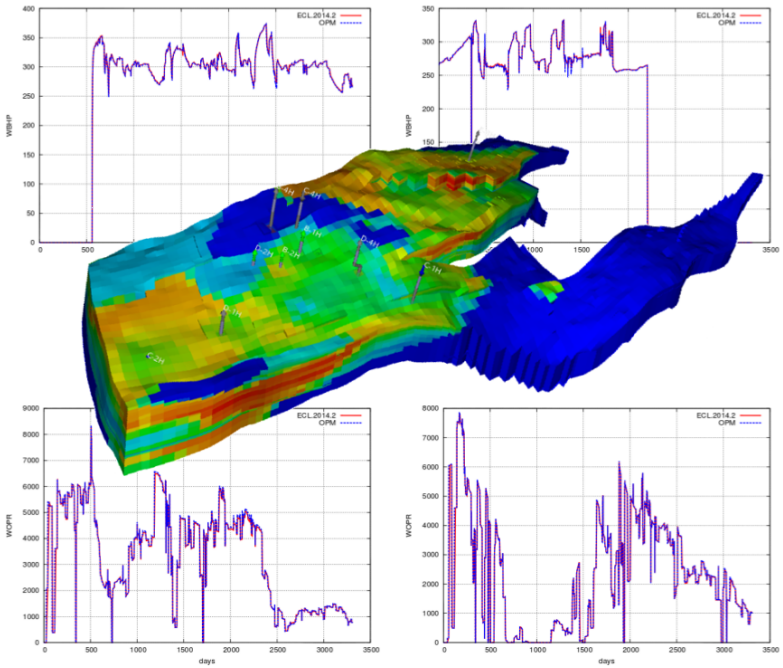
- (Tera)POWERS (Saudi-Aramco)
- MoReS (Shell)
- GPRS (Stanford University)
- ...

Open source simulators:

- OPM Flow
- MRST (Sintef)

OPM Flow at a glance

- Open source
- Handling cases of full industrial complexity (wells, properties)
- Competitive performance
- Currently used to study (for example):
  - Oil production: history matching and prediction of field performance
  - Enhanced oil recovery: CO2, polymer
  - CO2 sequestration
- Automatic differentiation enables rapid development of fluid models.



Ambition: to be a strong base for both industrial development and academic research

Overview of talk

Reservoir simulation

Mathematical formulation

Implementation with automatic differentiation

What next?

Physical laws and behaviours

- Conservation of mass
  - for each fluid pseudocomponent (oil, gas, water)
  - for injected EOR fluids (polymer, CO_2, surfactants)
  - (for ion species)
  - $\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$

Physical laws and behaviours

- Conservation of mass
  - for each fluid pseudocomponent (oil, gas, water)
  - for injected EOR fluids (polymer, CO_2, surfactants)
  - (for ion species)
  - $\frac{\partial}{\partial t}(\phi A_\alpha) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$
- Darcy's law (single phase)
  - $\mathbf{v} = -(1/\mu)\mathbf{K}(\nabla p - \rho \mathbf{g})$

Physical laws and behaviours

- Conservation of mass
  - for each fluid pseudocomponent (oil, gas, water)
  - for injected EOR fluids (polymer, CO_2, surfactants)
  - (for ion species)
  - $\frac{\partial}{\partial t}(\phi A_\alpha) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$
- Darcy's law (single phase)
  - $\mathbf{v} = -(1/\mu)\mathbf{K}(\nabla p - \rho \mathbf{g})$
- Multiphase flow: relative permeability and capillary pressure
  - flow rate reduced by k_r, function of fluid saturations
  - pressure difference between phases
  - $\mathbf{v}_\alpha = -(k_{r,\alpha}/\mu_\alpha)\mathbf{K}(\nabla p_\alpha - \rho_\alpha \mathbf{g})$

The black-oil model

- "Black-oil" model assumptions
  - Lump HC species into two (pseudo)components (oil, gas)
  - Allow oleic *phase* to contain both oil and gas *components*
    - Dissolved gas ratio, r_S
  - Allow gaseous *phase* to contain both oil and gas *components*
    - Vaporized oil ratio, r_V
  - Assumed always at thermodynamic equilibrium
  - Simple enough PVT relations to use table lookup

The black-oil model

- "Black-oil" model assumptions
  - Lump HC species into two (pseudo)components (oil, gas)
  - Allow oleic *phase* to contain both oil and gas *components*
    - Dissolved gas ratio, r_S
  - Allow gaseous *phase* to contain both oil and gas *components*
    - Vaporized oil ratio, r_V
  - Assumed always at thermodynamic equilibrium
  - Simple enough PVT relations to use table lookup
- Consequence: *phase* and *component* confusion!

The black-oil model

- "Black-oil" model assumptions
  - Lump HC species into two (pseudo)components (oil, gas)
  - Allow oleic *phase* to contain both oil and gas *components*
    - Dissolved gas ratio, r_S
  - Allow gaseous *phase* to contain both oil and gas *components*
    - Vaporized oil ratio, r_V
  - Assumed always at thermodynamic equilibrium
  - Simple enough PVT relations to use table lookup
- Consequence: *phase* and *component* confusion!
- Consequence: can have three different states:
  - fully saturated (all three phases present)
  - undersaturated oil (no gaseous phase)
  - undersaturated gas (no oleic phase)

The black-oil model

- "Black-oil" model assumptions
  - Lump HC species into two (pseudo)components (oil, gas)
  - Allow oleic *phase* to contain both oil and gas *components*
    - Dissolved gas ratio, r_S
  - Allow gaseous *phase* to contain both oil and gas *components*
    - Vaporized oil ratio, r_V
  - Assumed always at thermodynamic equilibrium
  - Simple enough PVT relations to use table lookup
- Consequence: *phase* and *component* confusion!
- Consequence: can have three different states:
  - fully saturated (all three phases present)
  - undersaturated oil (no gaseous phase)
  - undersaturated gas (no oleic phase)
- (Subset of more general *compositional* model, must solve equation of state, such as Peng-Robinson)

System of equations

The continuous equations form a system of partial differential equations, one for each pseudo-component α:

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha \tag{1}$$

where

$$A_w = m_\phi b_w s_w, \qquad \mathbf{u}_w = b_w \mathbf{v}_w, \tag{2}$$

$$A_o = m_\phi (b_o s_o + r_V b_g s_g), \qquad \mathbf{u}_o = b_o \mathbf{v}_o + r_V b_g \mathbf{v}_g, \tag{3}$$

$$A_g = m_\phi (b_g s_g + r_S b_o s_o), \qquad \mathbf{u}_g = b_g \mathbf{v}_g + r_S b_o \mathbf{v}_o. \tag{4}$$

and these additional relations should hold:

$$s_w + s_o + s_g = 1 \tag{5}$$

$$p_{cow} = p_o - p_w \tag{6}$$

$$p_{cog} = p_o - p_g. \tag{7}$$

The phase fluxes are given by Darcy's law:

$$\mathbf{v}_\alpha = -\lambda_\alpha \mathbf{K}(\nabla p_\alpha - \rho_\alpha \mathbf{g}). \tag{8}$$

Coupling to well model

What is the source term, anyway?

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

Coupling to well model

What is the source term, anyway?

$$\frac{\partial}{\partial t}\left(\phi A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha = Q_\alpha$$

Well rates!

- Computed using separate well model(s).
- Must be solved *simultaneously* with reservoir equations.

Discretization

The system is discretized using:

- First-order implicit Euler in time
- First-order finite volumes in space
  - two-point flux approximation (!)
  - phase-based upwinding of properties

Why such "primitive methods" (low order, inconsistent)?

- Historically, finite difference methods
- Need systems that linear solvers can deal with well
- Sufficient for practical purposes
- Discretization errors not significant compared to inherent uncertainty

Quite a bit *has* been done by various groups (not mainstream yet):

- Consistent discretizations (MPFA, mimetic)
- Higher-order FV or DG methods
- Higher-order time discretizations

Discrete equations (I)

The discretized equations and residuals are, for each pseudo-component α and cell i:

$$R_{\alpha,i} = \frac{\phi_{0,i} V_i}{\Delta t} \left(A_{\alpha,i} - A^0_{\alpha,i} \right) + \sum_{j \in C(i)} u_{\alpha,ij} + Q_{\alpha,i} = 0 \tag{9}$$

where

$$A_w = m_\phi b_w s_w, \qquad u_w = b_w v_w, \tag{10}$$

$$A_o = m_\phi (b_o s_o + r_{og} b_g s_g), \qquad u_o = b_o v_o + r_{og} b_g v_g, \tag{11}$$

$$A_g = m_\phi (b_g s_g + r_{go} b_o s_o), \qquad u_g = b_g v_g + r_{go} b_o v_o. \tag{12}$$

The relations (5), (6) and (7) hold for each cell i.

Discrete equations (II)

The fluxes are given for each connection ij by:

$$(b_\alpha v_\alpha)_{ij} = (b_\alpha \lambda_\alpha m_T)_{U(\alpha,ij)} T_{ij} \Delta H_{\alpha,ij} \tag{13}$$

$$(r_{\beta\alpha} b_\alpha v_\alpha)_{ij} = (r_{\beta\alpha} b_\alpha \lambda_\alpha m_T)_{U(\alpha,ij)} T_{ij} \Delta H_{\alpha,ij} \tag{14}$$

$$\Delta H_{\alpha,ij} = p_{\alpha,i} - p_{\alpha,j} - g\rho_{\alpha,ij}(z_i - z_j) \tag{15}$$

$$\rho_{\alpha,ij} = (\rho_{\alpha,i} + \rho_{\alpha,j})/2 \tag{16}$$

$$U(\alpha, ij) = \begin{cases} i & \Delta H_{\alpha,ij} \geq 0 \\ j & \Delta H_{\alpha,ij} < 0 \end{cases} \tag{17}$$

Solving the discrete equations

Main method: Newton-Raphson

- solve large heterogenous linear systems
- challenging to precondition (CPR + AMG best?)
- must modify updates for phase changes
- must handle convergence failures (timestep cuts)

Overview of talk

Reservoir simulation

Mathematical formulation

Implementation with automatic differentiation

What next?

What does AD provide

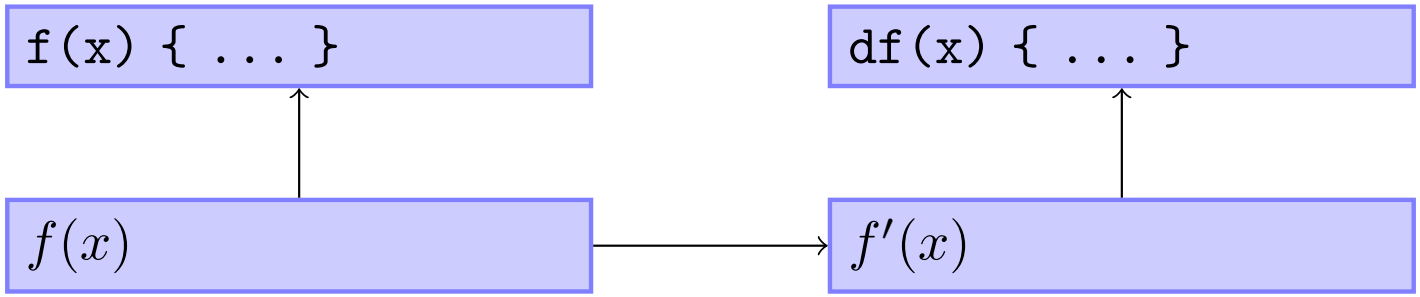
```
f(x) { ... }
```

```
df(x) { ... }
```

$f(x)$

$f'(x)$

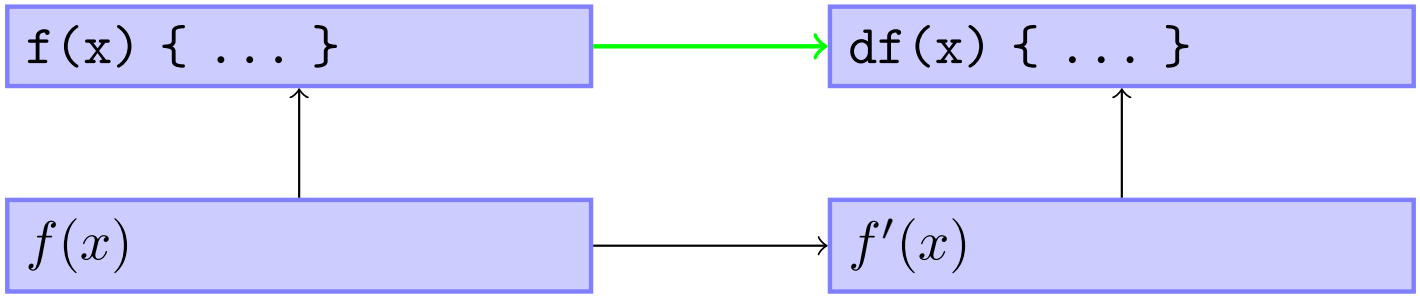
What does AD provide



Traditional Process

- Human implements code to evaluate $f(x)$
- Manual or symbolic calculation to derive $f'(x)$
- Human implements code to evaluate $f'(x)$

What does AD provide



Traditional Process

- Human implements code to evaluate $f(x)$
- Manual or symbolic calculation to derive $f'(x)$
- Human implements code to evaluate $f'(x)$

Automatic Differentiation (AD)

- Human implements code to evaluate $f(x)$
- Computer code to evaluate $f'(x)$ is automatically generated

Benefits of using AD

AD makes it easier to create simulators:

- only specify nonlinear residual equation
- automatically evaluates Jacobian
- sparsity structure of Jacobian automatically generated

Note that AD is *not* the same as finite differencing!

- no need to define a 'small' epsilon
- as precise as hand-made Jacobian
- ... but much less work!

Performance (of equation assembly) will usually be somewhat slower than a *good* hand-made Jacobian implementation.

Basic idea

A numeric computation $y = f(x)$ can be written (D = derivative)

$$
\begin{aligned}
y_1 &= f_1(x) & \frac{dy_1}{dx}(x) &= Df_1(x) \\
y_2 &= f_2(y_1) & \frac{dy_2}{dx}(x) &= Df_2(y_1) \cdot Df_1(x) \\
&\vdots \\
y &= f_n(y_{n-1}) & \frac{dy}{dx}(x) &= Df_n(y_{n-1}) \cdot Df_{n-1}(y_{n-2}) \cdots Df_1(x)
\end{aligned}
$$

Automatic Differentiation:

- make each line an elementary operation
- compute right derivative values as we go using chain rule

Implementation approaches

Two main methods:

Operator overloading

- requires operator overloading in programming language
- syntax (more or less) like before (non-AD)
- efficiency can vary a lot, depends on usage scenario
- easy to implement and experiment with
- Examples: OPM, Sacado (Trilinos), ADOL-C

Source transformation with AD tool

- can be implemented for almost any language
- may restrict language syntax or features used
- efficiency can be high (depends on AD tool)
- Examples: TAPENADE, OpenAD

Types of AD

Two different approaches.
(We compute $f(x)$, u is some intermediate variable.)

Forward Mode

Carry derivatives with respect to independent variables:

$$(u, \frac{du}{dx})$$

Reverse Mode

Carry derivatives with respect to dependent variables (adjoints):

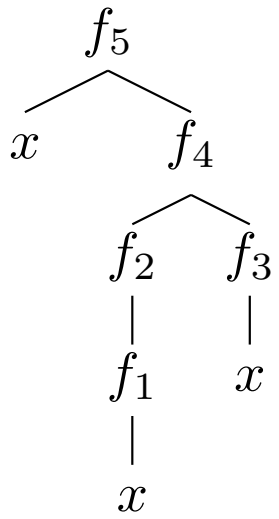
$$(u, \frac{df}{du})$$

Forward AD example (1)

Example function: $f(x) = x(sin(x^2) + 3x)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

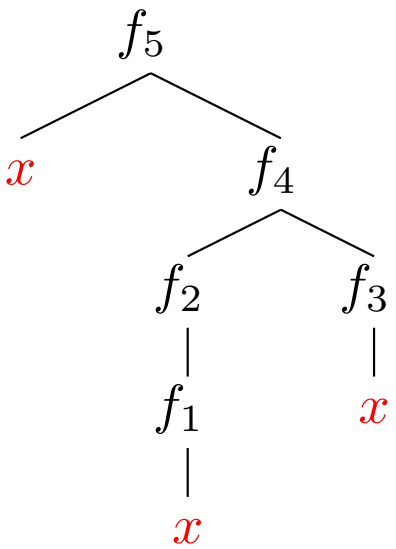
Rewritten:
$f(x) = f_5(x, f_4(f_2(f_1(x)), f_3(x)))$

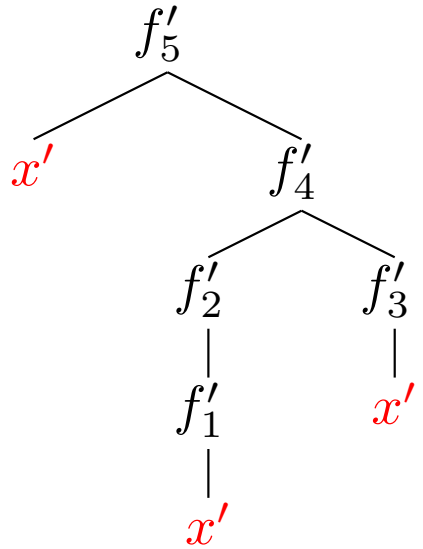


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

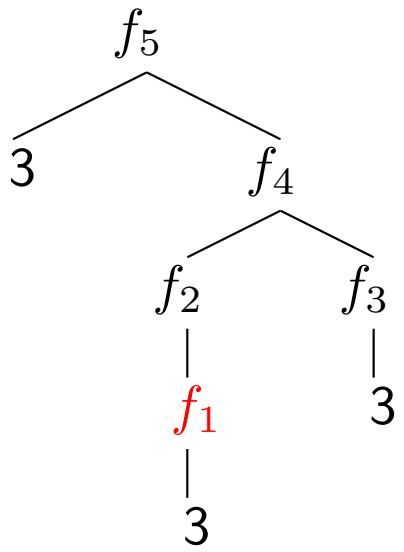


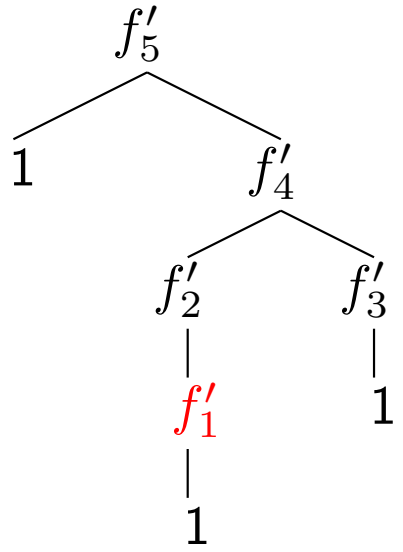


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

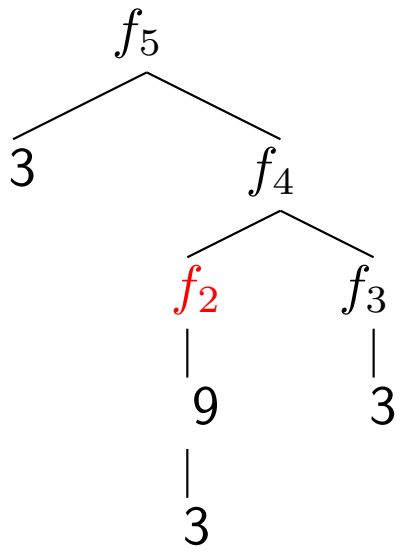


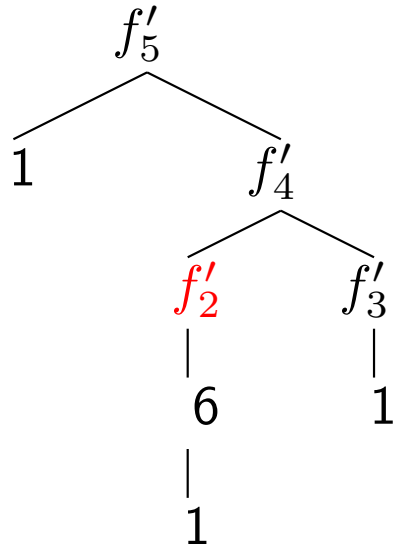


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

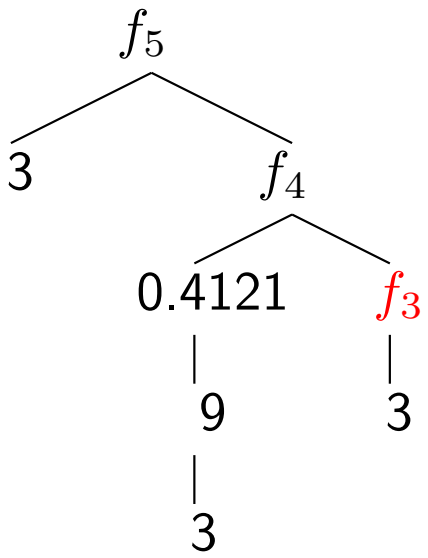


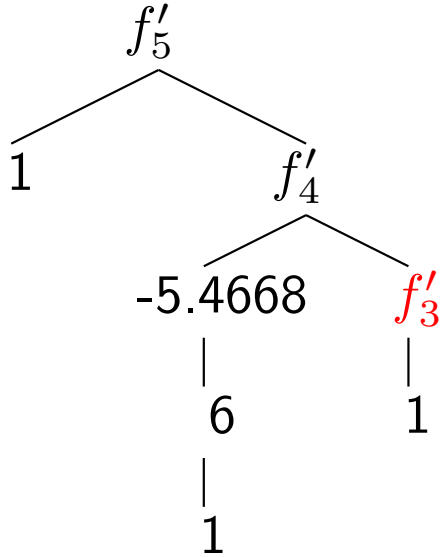


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

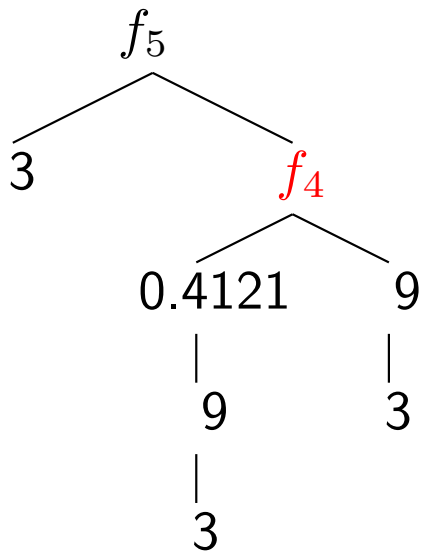


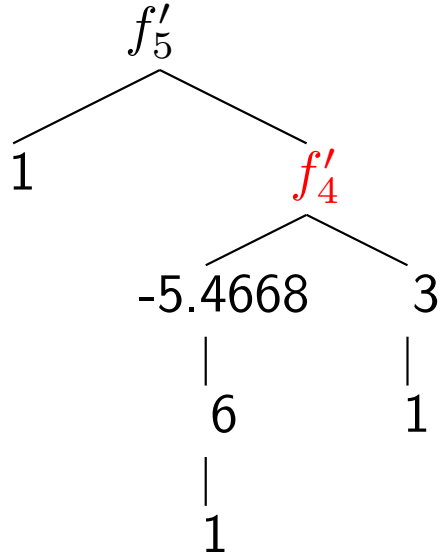


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

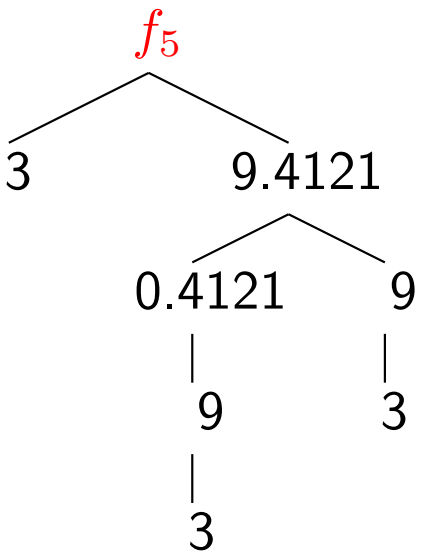


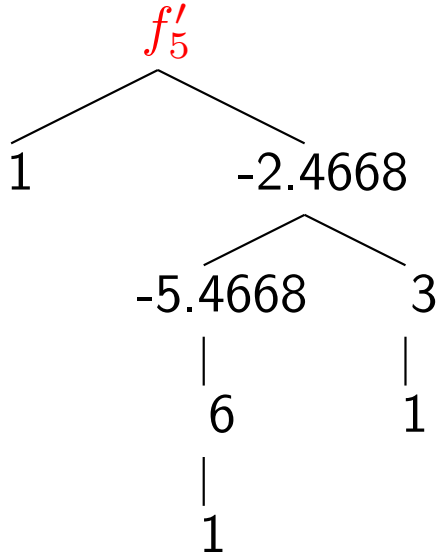


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$

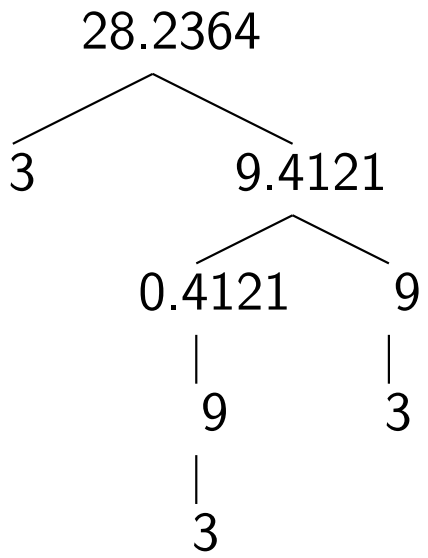


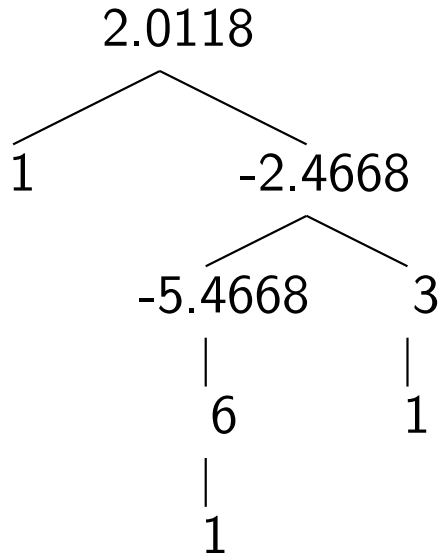


Forward AD example (2)

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & f_1'(u) &= 2uu' \\
f_2(u) &= \sin(u) & f_2'(u) &= \cos(u)u' \\
f_3(u) &= 3u & f_3'(u) &= 3u' \\
f_4(u,v) &= u + v & f_4'(u,v) &= u' + v' \\
f_5(u,v) &= u \cdot v & f_5'(u,v) &= u'v + uv'
\end{aligned}
$$





Properties of forward AD

- Easy to implement with operator overloading
- Storage required (scalar): $2\times$ normal (value, derivative).
- Storage required ($f : R^m \to R^n$): $(n+1)\times$ normal (value, derivative vector), unless sparse.

Reverse Mode AD

Recall: Reverse Mode

Carry derivatives with respect to dependent variables (adjoints):

$$(u, \frac{df}{du})$$

We will use the chain rule again, but in the opposite direction:

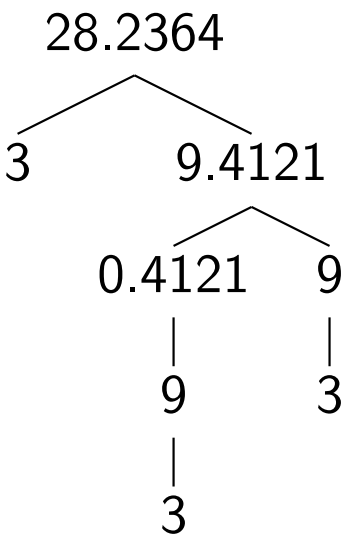
$$\mathrm{adj}(u) = \mathrm{adj}(f_i)\frac{\partial f_i}{\partial u}.$$

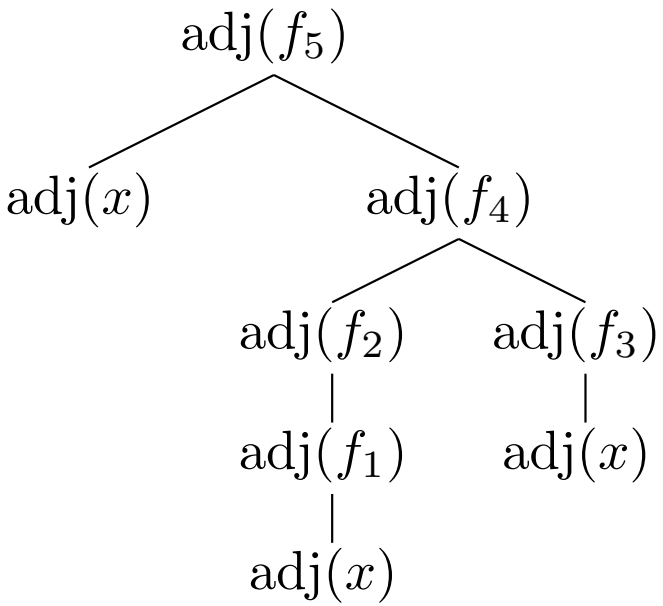
Using $\mathrm{adj}(u)$ to mean the adjoint $\frac{df}{du}$.
(So $\mathrm{adj}(x)$ is our goal.)

Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

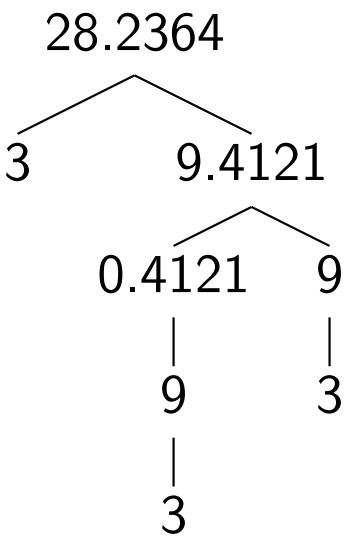


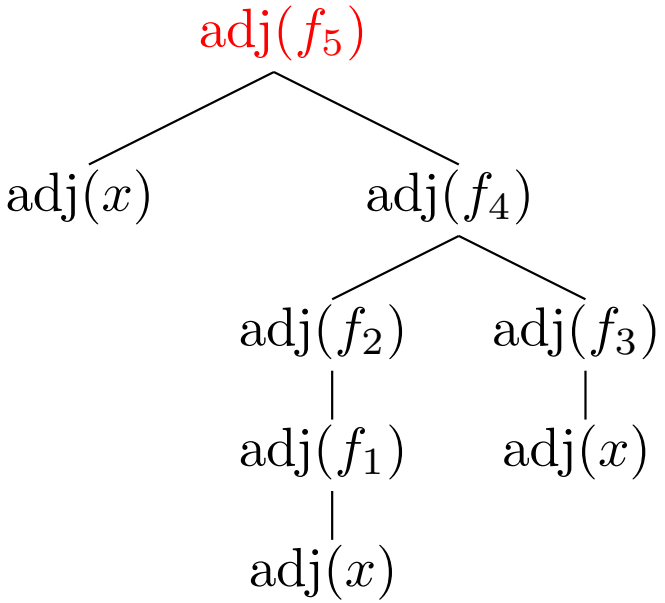


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

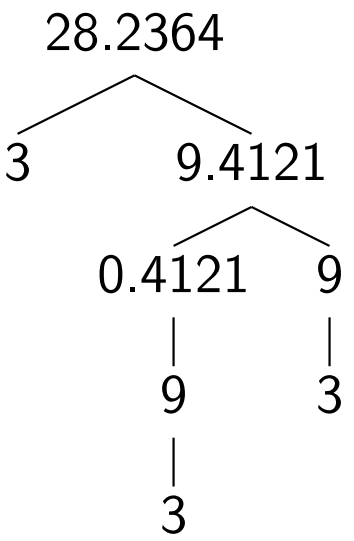


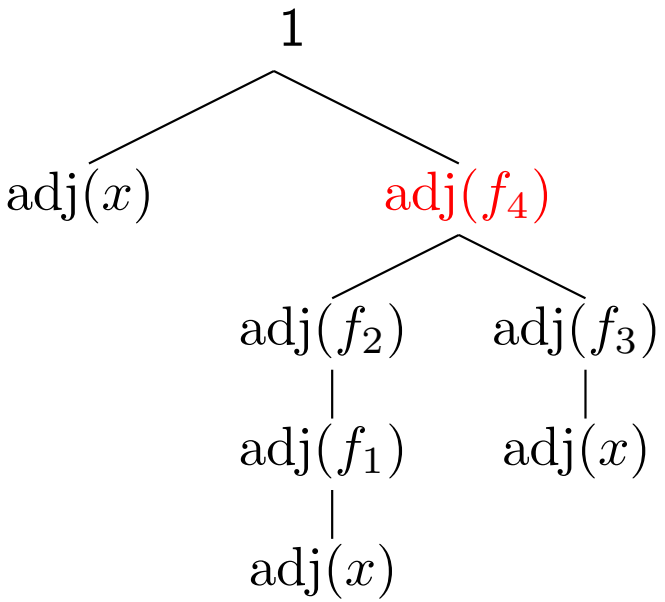


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

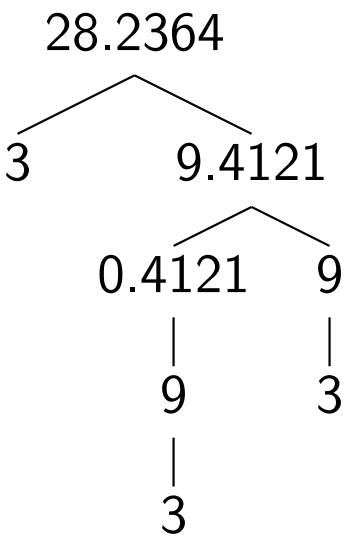


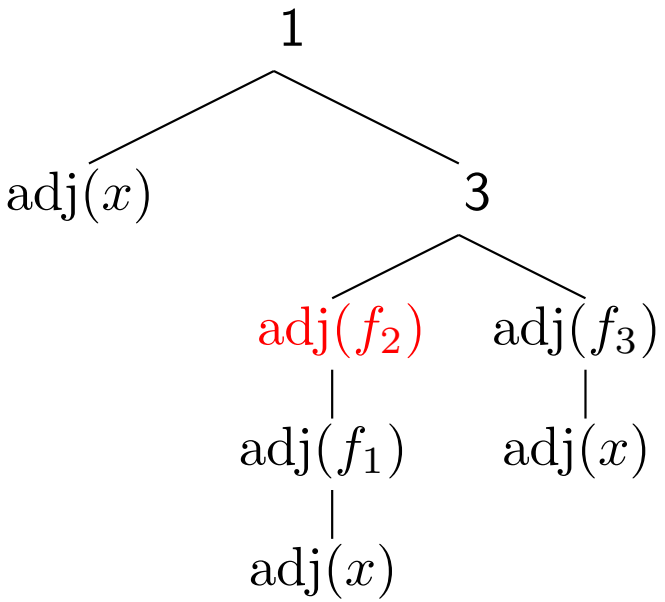


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

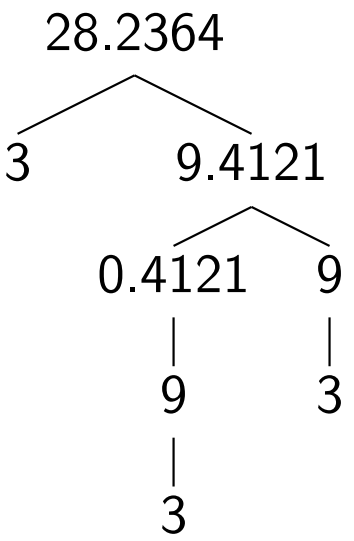


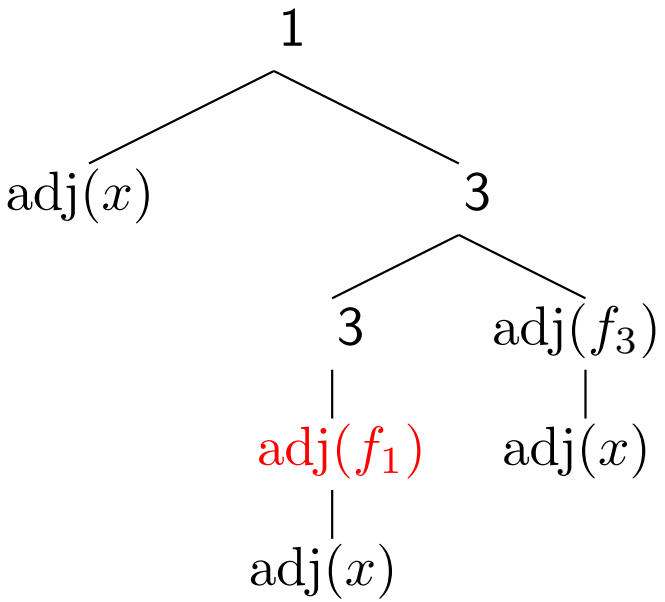


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

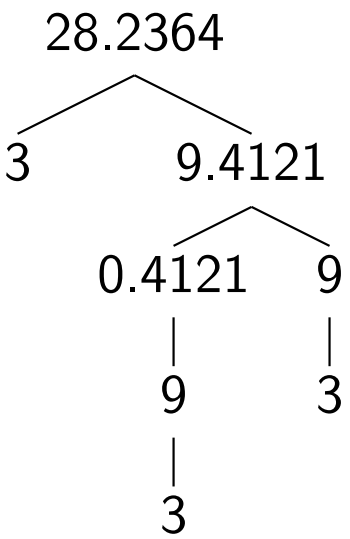


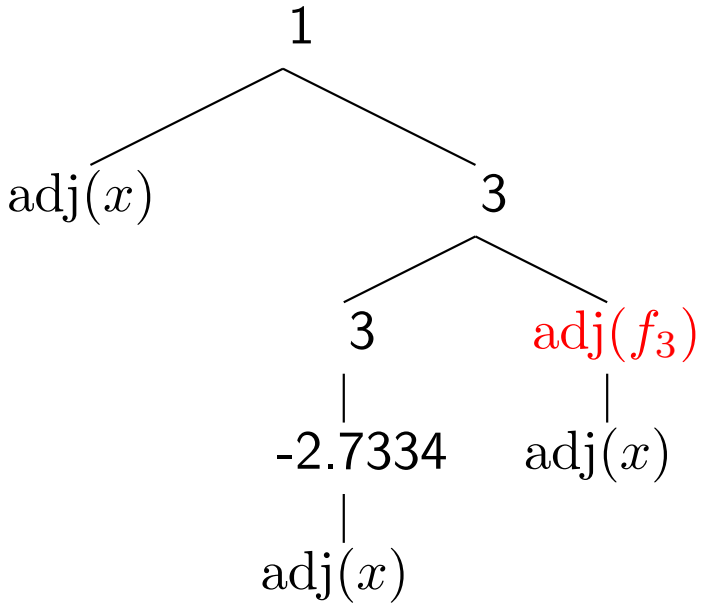


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 && \mathrm{adj}(u) = \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) && \mathrm{adj}(u) = \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u && \mathrm{adj}(u) = \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v && \mathrm{adj}(u) = \mathrm{adj}(f_4), && \mathrm{adj}(v) = \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v && \mathrm{adj}(u) = \mathrm{adj}(f_5) \cdot v, && \mathrm{adj}(v) = \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

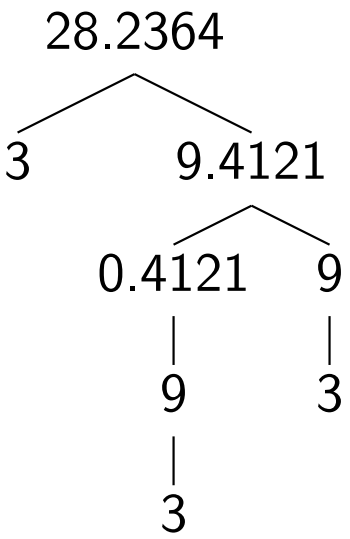


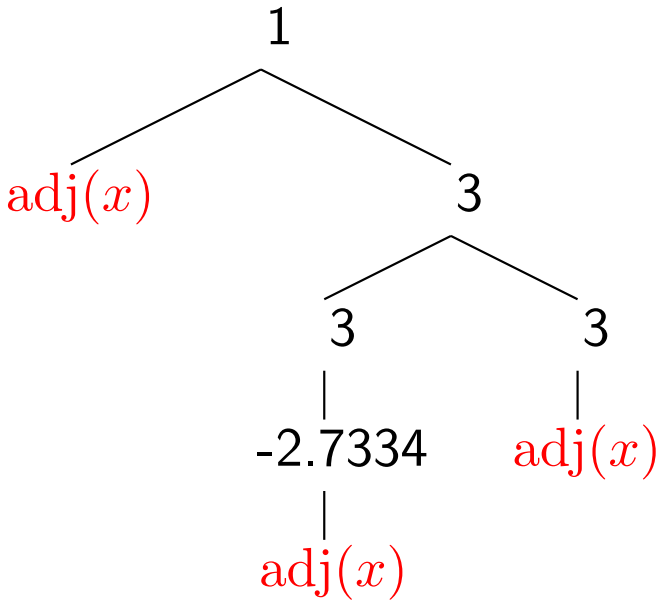


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

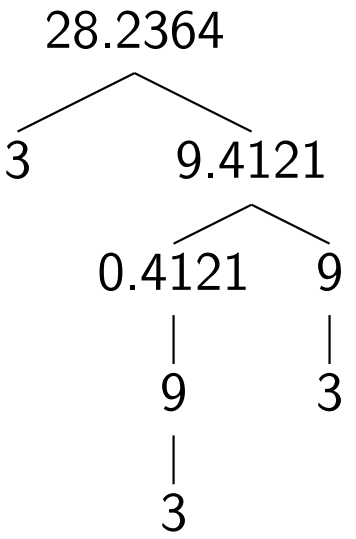


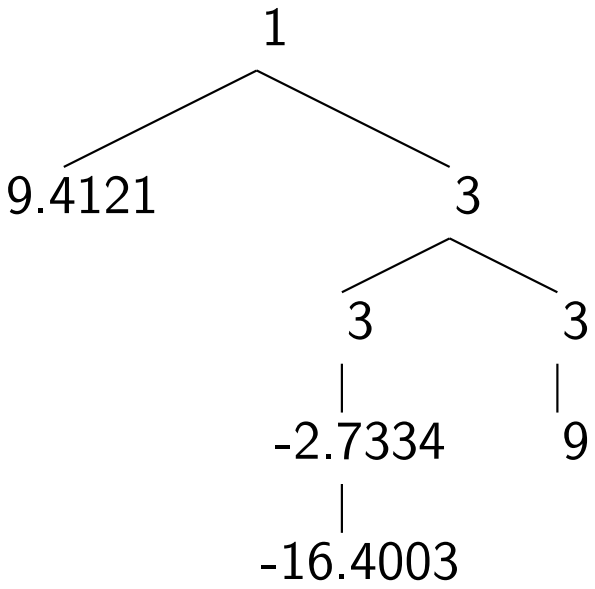


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$

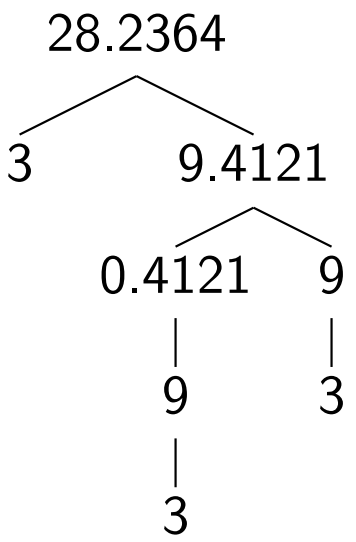


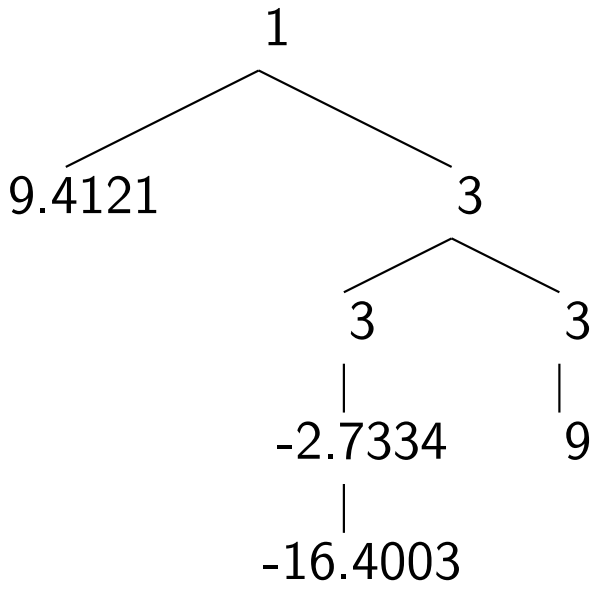


Reverse AD example

Example function: $f(x) = x(sin(x^2) + 3x)$. Computing $f(3), f'(3)$.
Sequence of elementary functions:

$$
\begin{aligned}
f_1(u) &= u^2 & \mathrm{adj}(u) &= \mathrm{adj}(f_1) \cdot 2u \\
f_2(u) &= \sin(u) & \mathrm{adj}(u) &= \mathrm{adj}(f_2) \cdot \cos(u) \\
f_3(u) &= 3u & \mathrm{adj}(u) &= \mathrm{adj}(f_3) \cdot 3 \\
f_4(u,v) &= u + v & \mathrm{adj}(u) &= \mathrm{adj}(f_4), & \mathrm{adj}(v) &= \mathrm{adj}(f_4) \\
f_5(u,v) &= u \cdot v & \mathrm{adj}(u) &= \mathrm{adj}(f_5) \cdot v, & \mathrm{adj}(v) &= \mathrm{adj}(f_5) \cdot u
\end{aligned}
$$





Must sum contributions:
$f'(3) = 9.4121 - 16.4003 + 9 = 2.0118.$

Automatic Differentiation: OPM implementations

AutoDiffBlock (legacy)

- class implementing *forward AD*
- deals with *vectors of values* at a time
- derivatives are *sparse matrices*
- implemented with operator overloading
- based on Eigen library for basic types and operands
- helper library provides discrete div, grad etc.

Evaluation (new effort)

- class implementing *forward AD*
- deals with *a single scalar value* at a time
- derivatives are *compile-time-size vectors*
- implemented with operator overloading
- discrete div, grad must be implemented "manually"

Overview of talk

Reservoir simulation

Mathematical formulation

Implementation with automatic differentiation

What next?

We keep pursuing...

- Performance
  - Improving preconditioners and linear solvers
  - New discretizations, nonlinear preconditioning
  - Parallel scaling
- Usability
  - Reference manual started!
  - More output/logging options
  - Better error messages
- Deployability
  - Container usage
  - MSO4SC Portal and web integration
  - Better error messages

Further ahead: some possibilities

- New fluid models for CO_2, EOR options.
- Scriptable? Python? Cling-able?
- More flexible boundary conditions
- Higher order DG methods?
- Consistent discretizations?
- Fully compositional fluid model?
- New I/O system for parallel scalability?
- Preprocessing tools?

Thank you for listening!