SAND2010-6787C

ExxonMobil Upstream Research

Verification of Gradient and Hessian Computation for Full Wavefield Inversion Using Automatic Differentiation

Lijian Tan, Valery Brytik, Anatoly Baumstein,
David Hinkley

ExxonMobil Upstream Research Company

Outline

ExxonMobil
Upstream Research

- Introduction
- Theory
- Numerical examples
- Summary

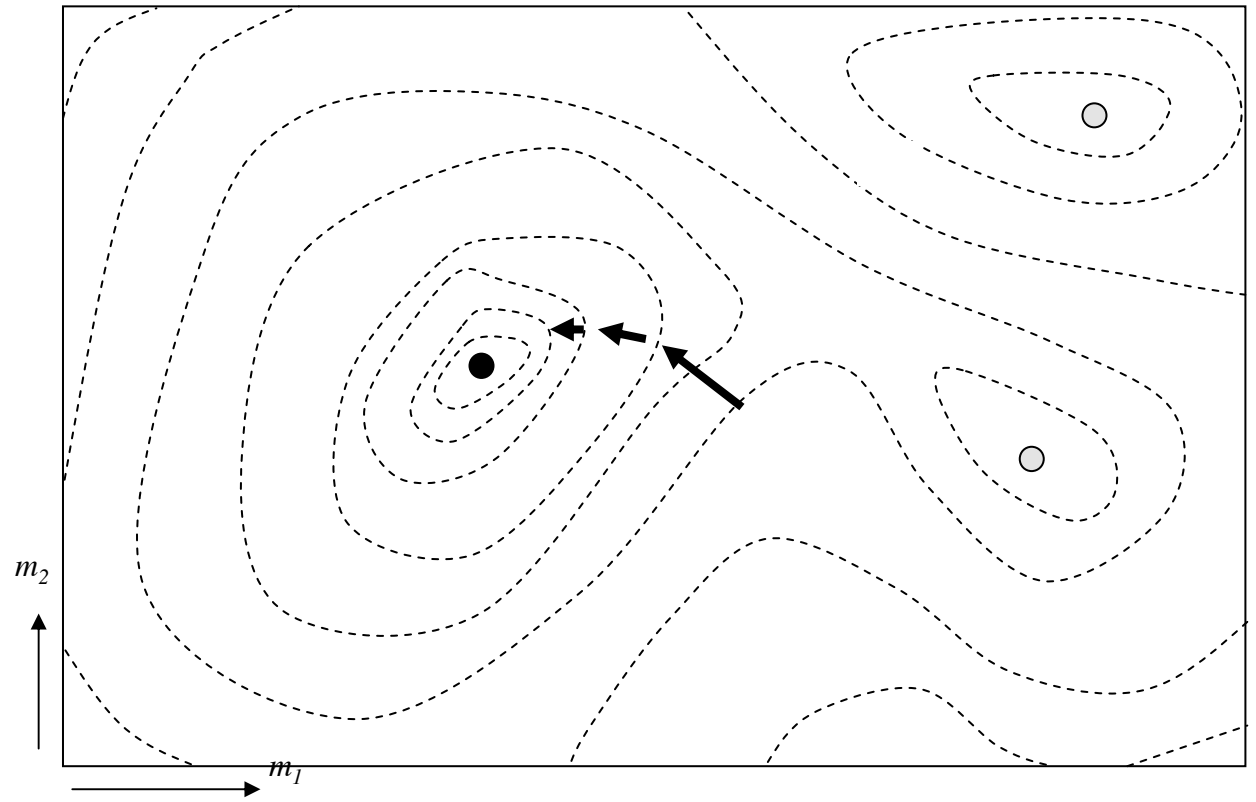
Motivation

ExxonMobil
Upstream Research

- Full waveform inversion
  - provides quantitative information about the subsurface structure which may otherwise not be found using conventional processing
  - is usually formulated as a minimization problem.
- Minimization problems are often solved using iterative gradient-based methods.
- Gradient-based methods benefit from accurate evaluation of the derivative of the objective function with respect to medium parameters.
- Gradient can be computed in various ways.
  - finite differences
  - adjoint state method
  - automatic differentiation

Gradient and its use

ExxonMobil
Upstream Research



-gradient direction ○ -local minima ● -global minimum

Schematic of Full Waveform Inversion (FWI) with the Adjoint-State Method

ExxonMobil
Upstream Research

- Fit the complete wavefield – optimize $|Gm - d|^2$

d=data

G=forward simulation

Physics – wave equation (acoustic, elastic, ...)

m=model

Update Earth Model Vp, Vs, ρ

Forward Modeling

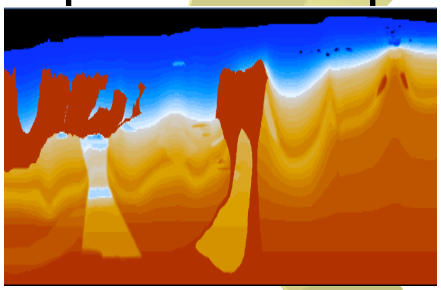
Simulated Data

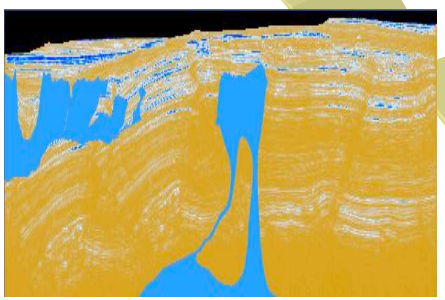
Seismic Data

Adjoint Modeling

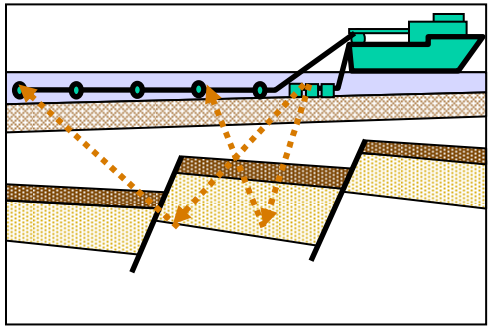
Residual

Data Fit Error

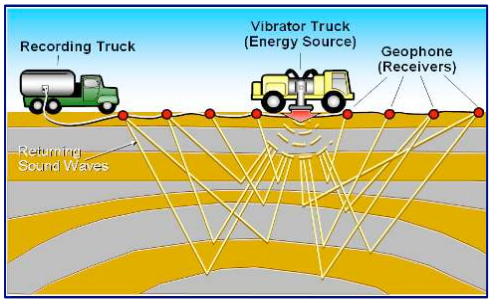




Marine Acquisition



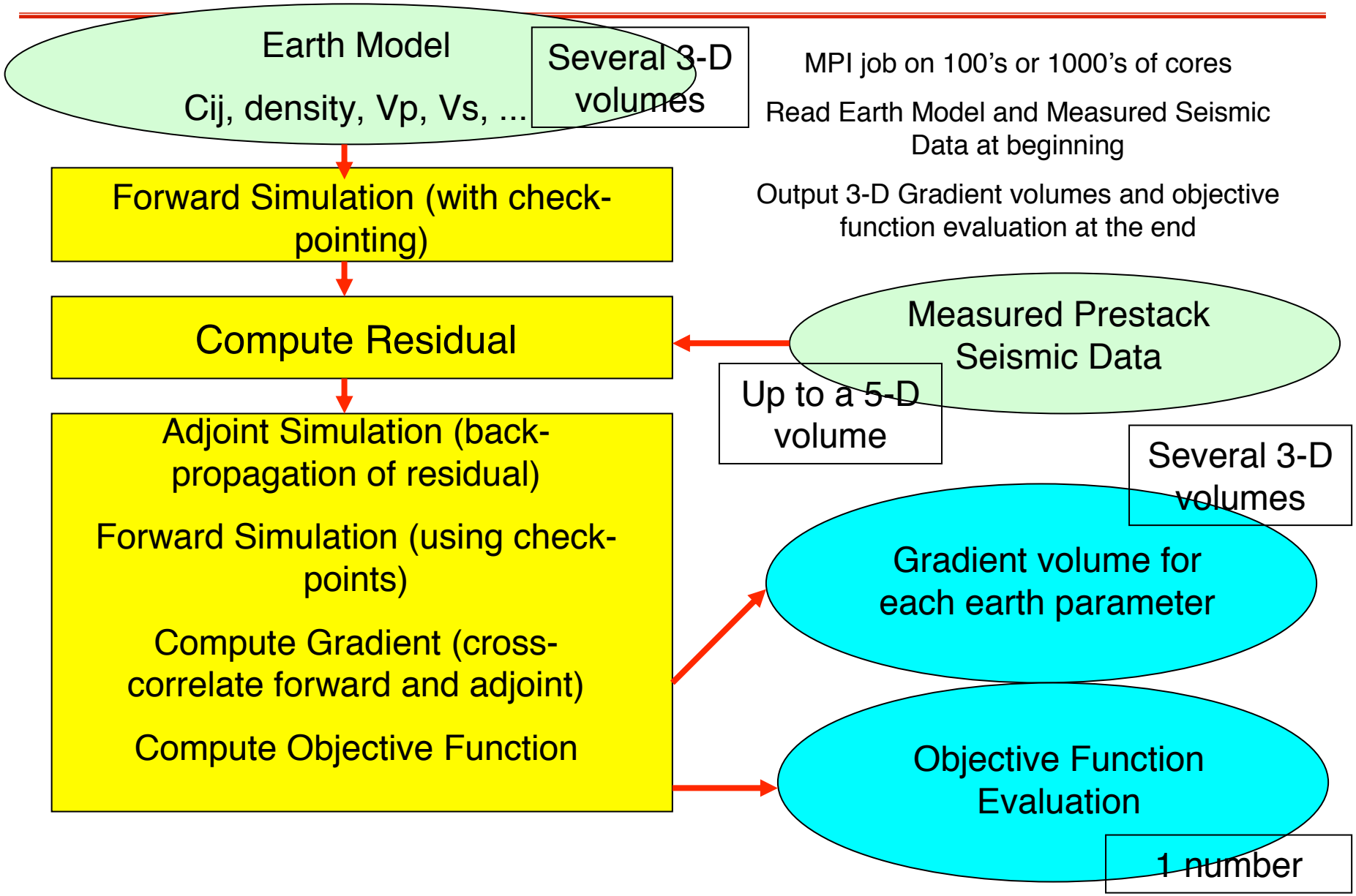
Land Acquisition



Key Advantage: Uses all of the data

Adjoint-State Method

ExxonMobil
Upstream Research



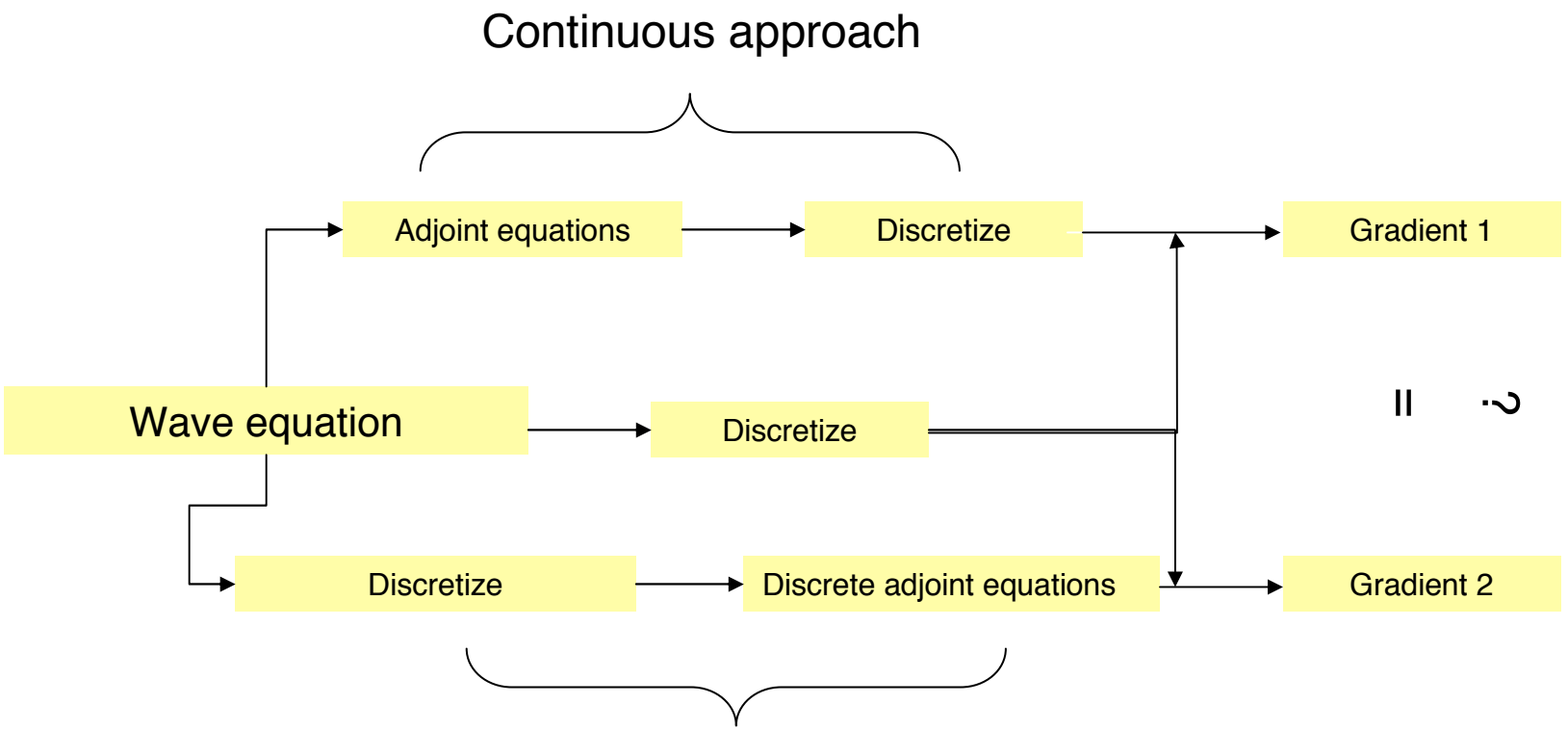
Comparison of gradient computation approaches

ExxonMobil
Upstream Research

- Finite differences
  - Advantage: simplicity of formulation and implementation
  - Disadvantage: very high computational cost.
- Automatic differentiation
  - Advantage: produces perfect adjoint
  - Disadvantage: relatively high cost: less than FD, but higher than ASM. Difficult to apply to complex algorithms.
- Adjoint-state method
  - Advantage: simplicity of formulation. Low cost.
  - Disadvantage: relatively difficult implementation.
  - Problem: "Discrete of adjoint is not always adjoint of the discrete”

Continuous vs. Discrete

ExxonMobil
Upstream Research



Use AD to verify the gradient obtained using the continuous approach

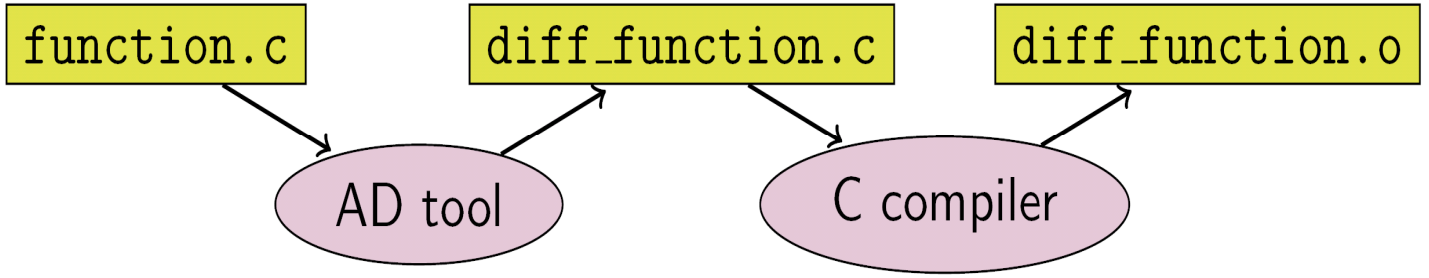
Continuous vs. Discrete Adjoints

ExxonMobil
Upstream Research

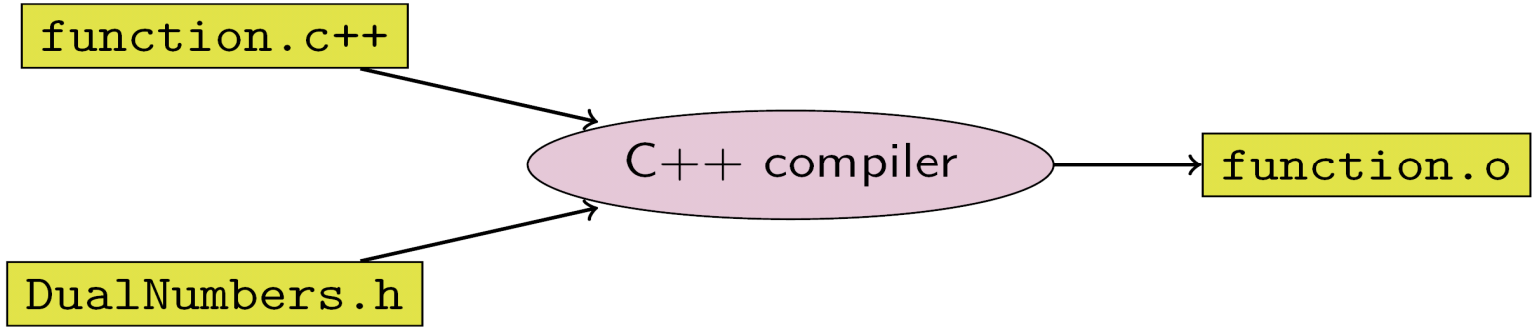
- Continuous adjoint
  - Derive adjoint, then discretize
  - Pros: Adjoint is easy to derive and is independent of the implementation (finite difference scheme)
  - Cons: Once both forward and adjoint equations are disretized (even using the same finite-difference scheme), the adjoint may not be exact.
- Discrete adjoint
  - Discretize, then derive adjoint
  - Pros: Adjoint is exact
  - Cons: More difficult to derive than in the continuous case; need to re-derive the adjoint if finite-difference scheme changes.

Two different approaches to AD

ExxonMobil
Upstream Research



Code transformation automatic differentiation



Operator overloading automatic differentiation

http://en.wikipedia.org/wiki/Automatic_differentiation

Automatic differentiation (AD) : ADOL-C

ExxonMobil
Upstream Research

- Facilitates evaluation of first and higher derivatives of vector functions that are defined by computer programs written in C or C++.
- Derivative evaluation routines may be called from C,C++, Fortran, or any other language that can be linked with C.
- Derivative calculations involve a possibly substantial but always predictable amounts of data.
- Since the data is accessed strictly sequentially it can be automatically paged out to external files.

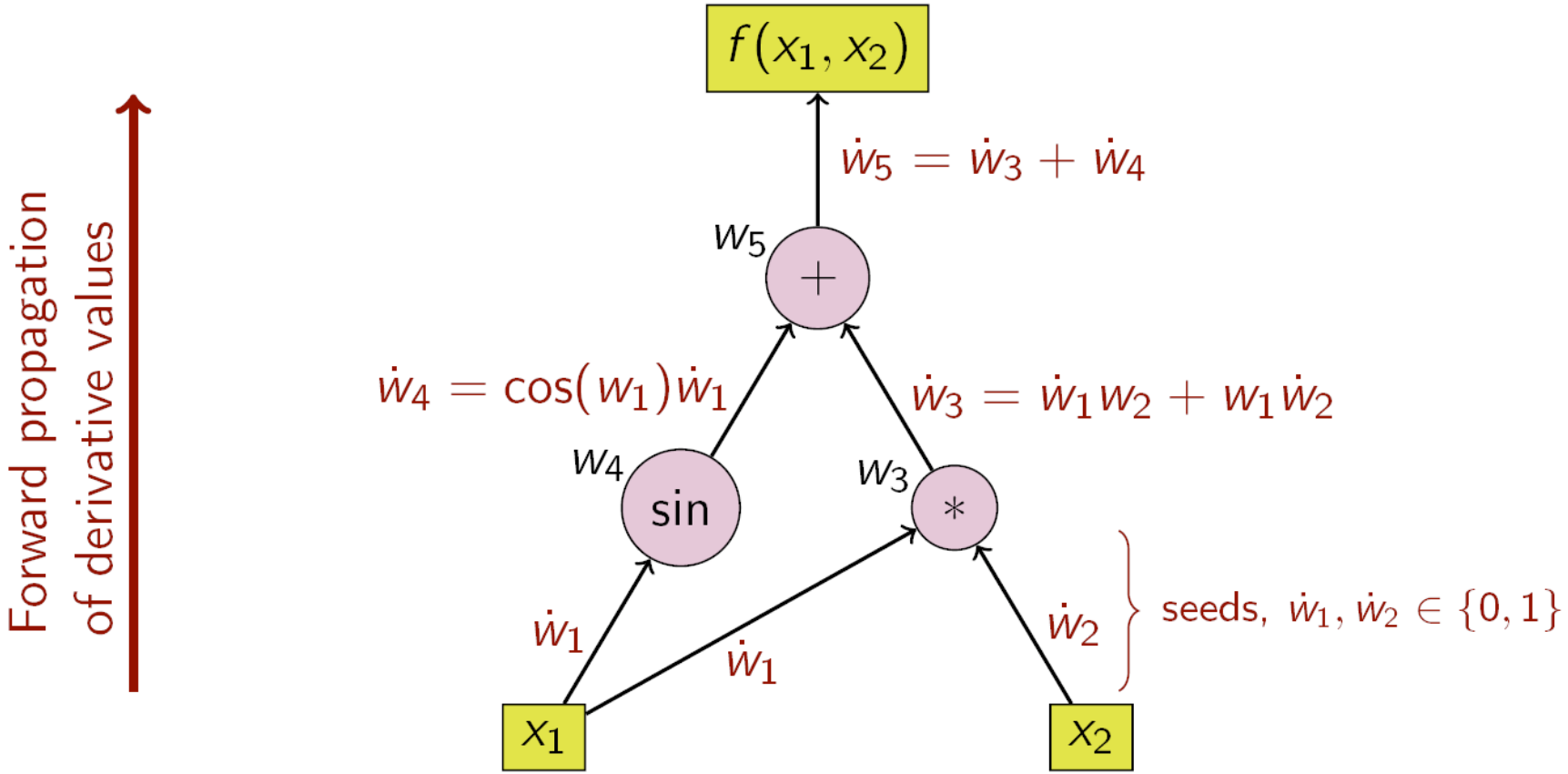
$$(f \circ g)'(x) = f'(g(x))g'(x)$$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

AD basics: Forward mode

ExxonMobil
Upstream Research

$$f(x_1, x_2) = x_1 \times x_2 + \sin(x_1)$$

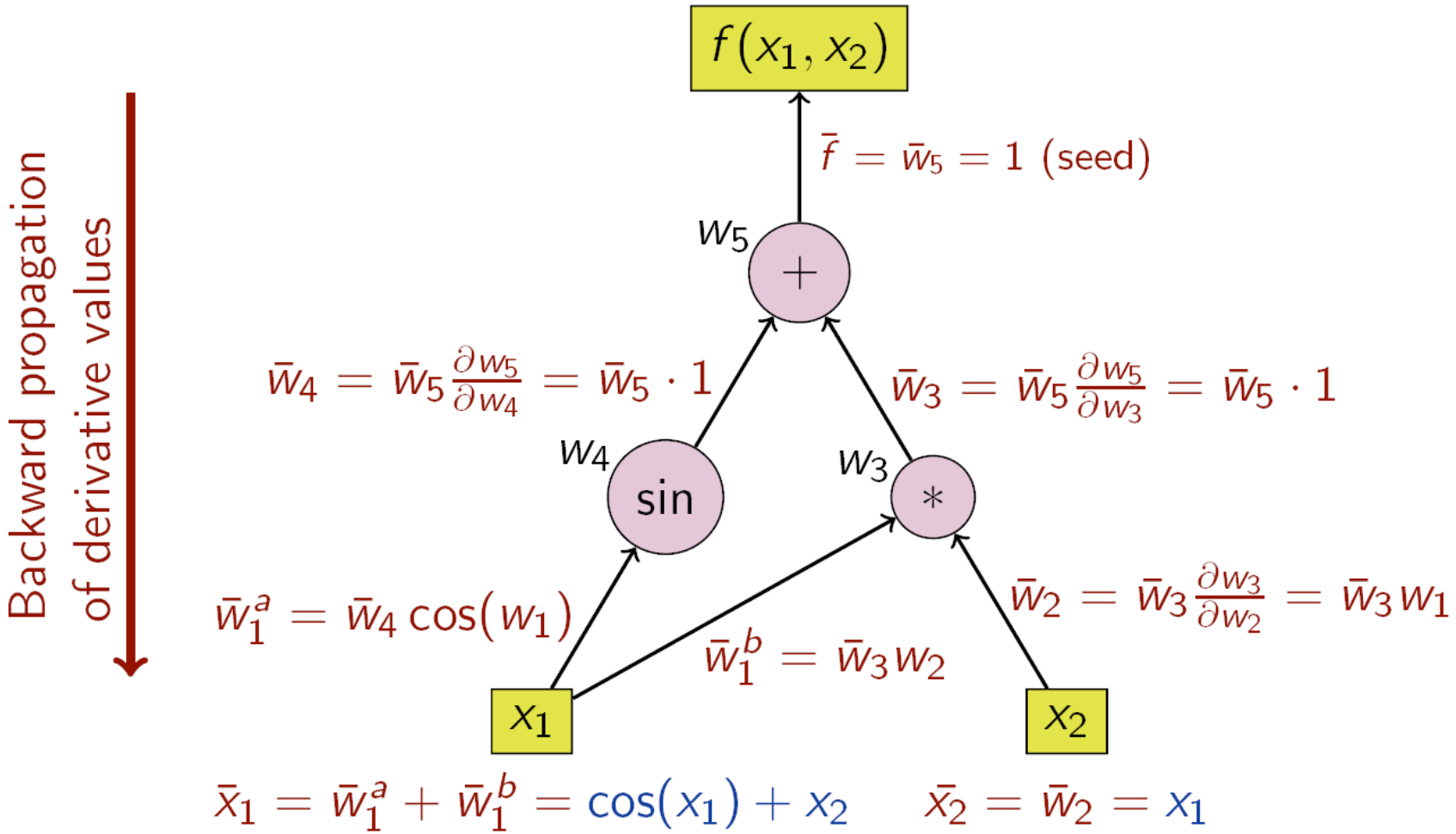


http://en.wikipedia.org/wiki/Automatic_differentiation

AD basics: Reverse mode

ExxonMobil Upstream Research

$$f(x_1, x_2) = x_1 \times x_2 + \sin(x_1)$$



http://en.wikipedia.org/wiki/Automatic_differentiation

AD basics: Forward Vs Reverse

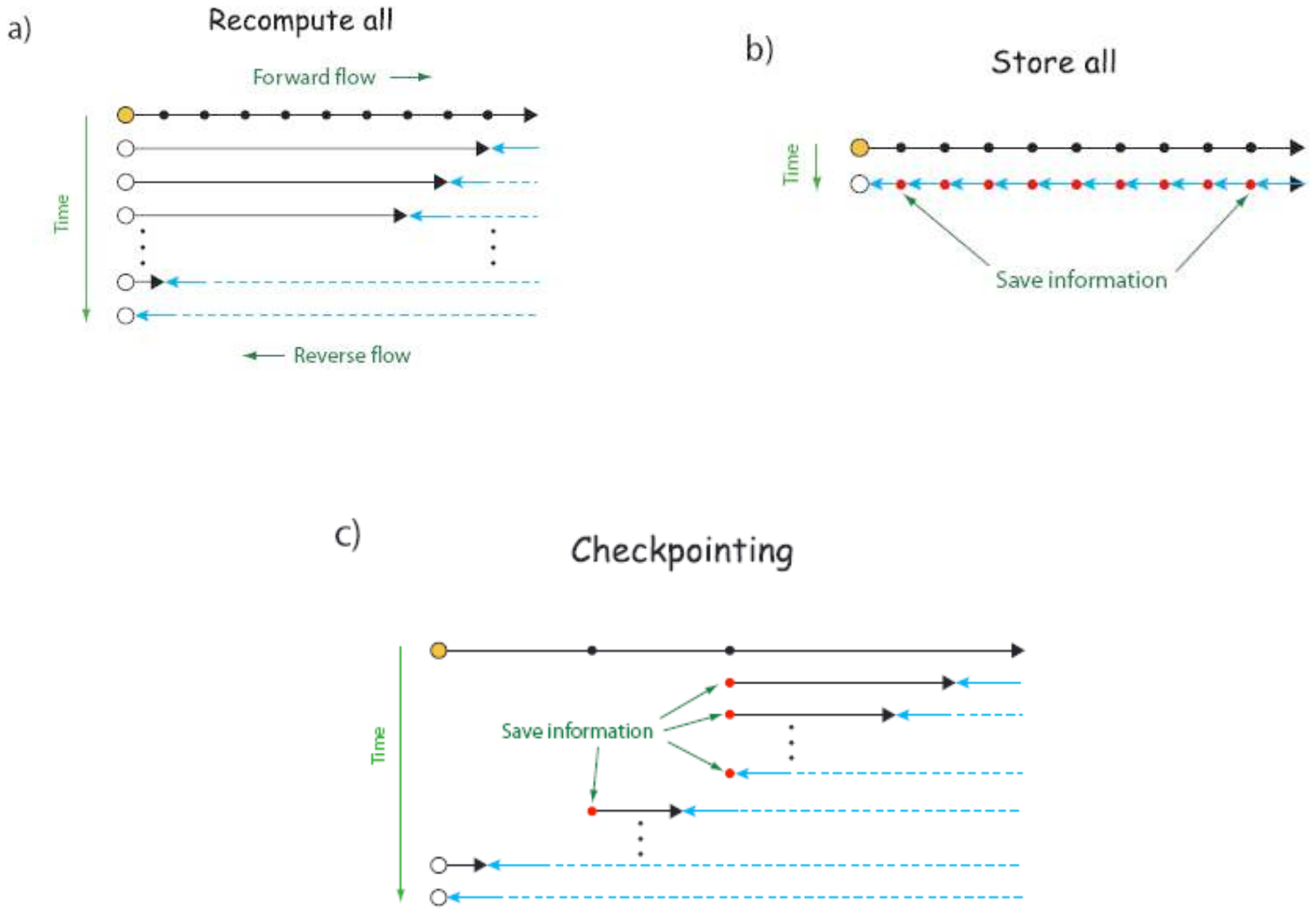
ExxonMobil
Upstream Research

- Forward mode
  - Has limited memory requirements. The state of all variables is updated as the source code proceeds.
  - Very efficient when number of dependent variables is significant.

- Reverse mode
  - At each stage of the reverse list, evaluation of the elemental Jacobians requires numerical values of the corresponding intermediate variables.
  - The entire state of the code, including the values of all variables, needs to be known at each time step.
  - Very efficient for gradient computations.

Reverse mode, using checkpointing.

ExxonMobil
Upstream Research



Gradient computations.

ExxonMobil
Upstream Research

Forward:

✓ Slow, restricted to small models.

Reverse. No checkpoining:

✓1.Fast. Slow if needs to use I/O

✓2.Could be reused for another velocity model

Reverse with checkpointing:

✓1.Resonably fast.

✓2. I/O could be reduced or eliminated.

→

WINNER:

Reverse with checkpointing

*Hessian*vector and Hessian computations in ADOL-C*

ExxonMobil
Upstream Research

- ADOL-C package contains "easy-to-use" drivers for the Hessian*vector and Hessian computations

- In the standard ADOL-C package checkpointing is not implemented for the functions used in the procedures computing "Hessian * vector" and as a consequence for "Hessian" computations.

The applicability of these drivers is limited to very small models due to the absence of checkpointing.

Resolving some limitations

ExxonMobil
Upstream Research

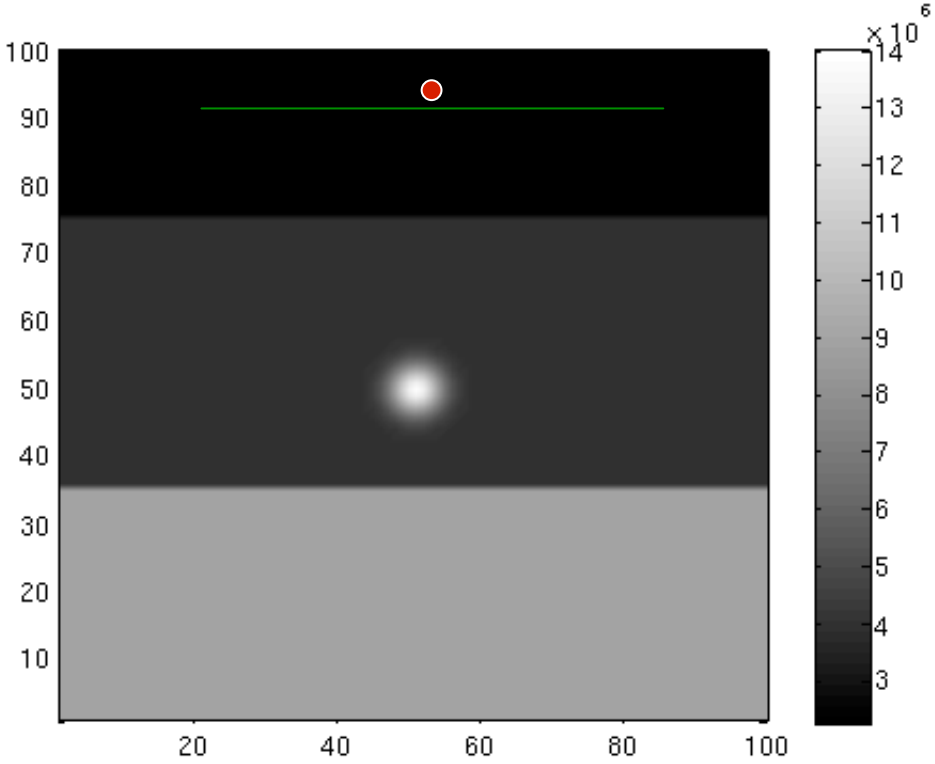
- ADOL-C Hessian computation can be implemented with checkpointing.

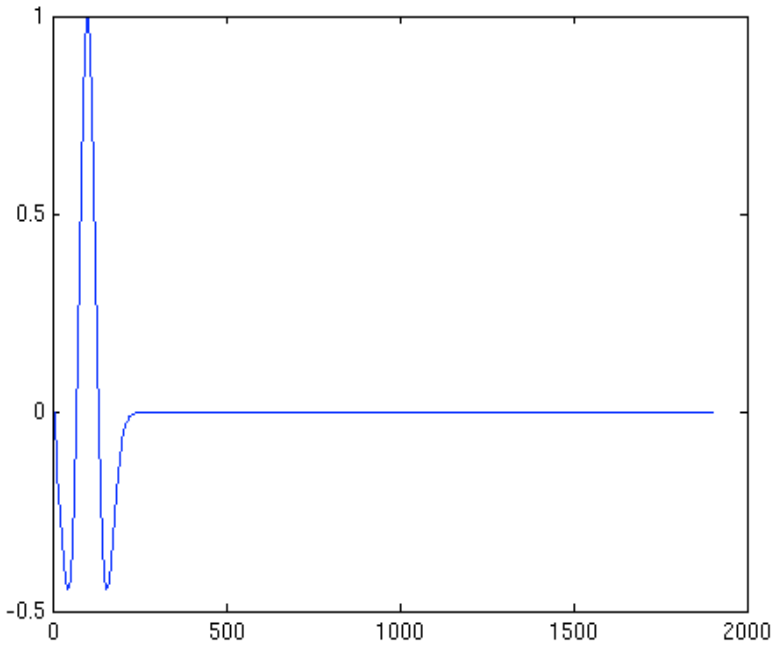
- Developed a Hessian computation algorithm for general objective function and acquisition geometry as appropriate for FWI.

Numerical examples

ExxonMobil
Upstream Research

100 by 100 mesh, with 20m spacing, rho=1
Total simulation time 1.9s; source at the red dot, 64 receivers along the green line.





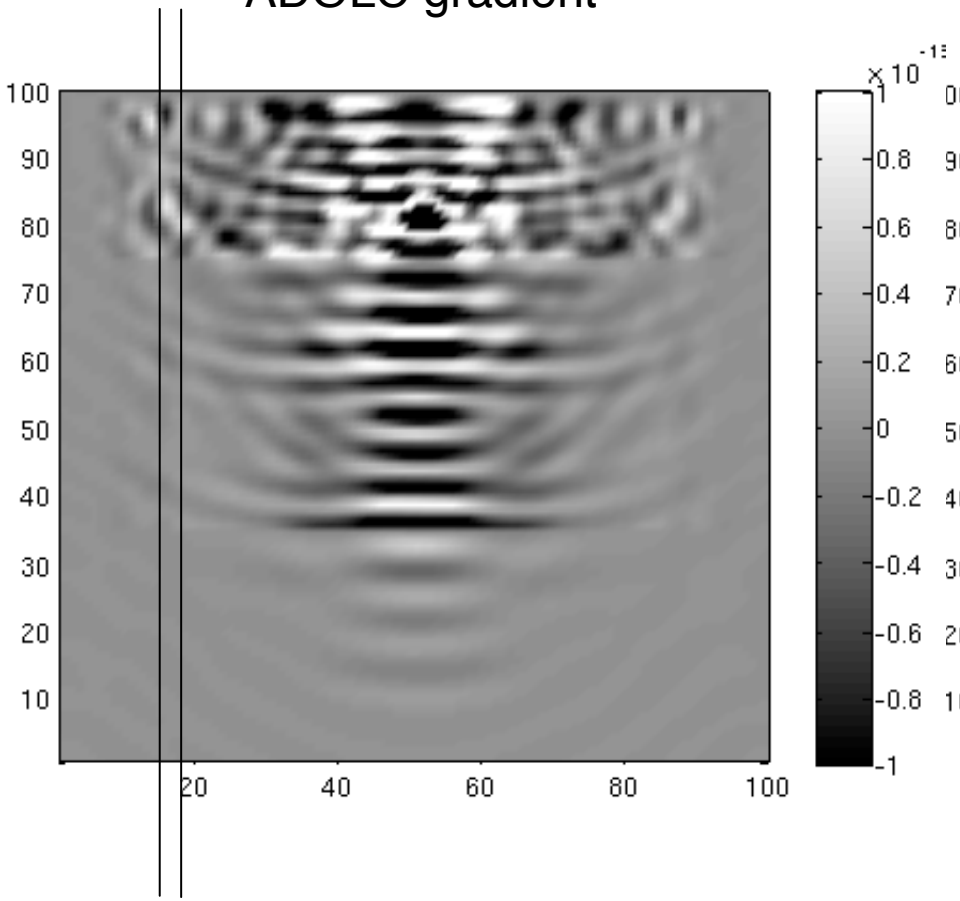
Source information

This “ball” doesn’t exist in the initial model

Numerical examples

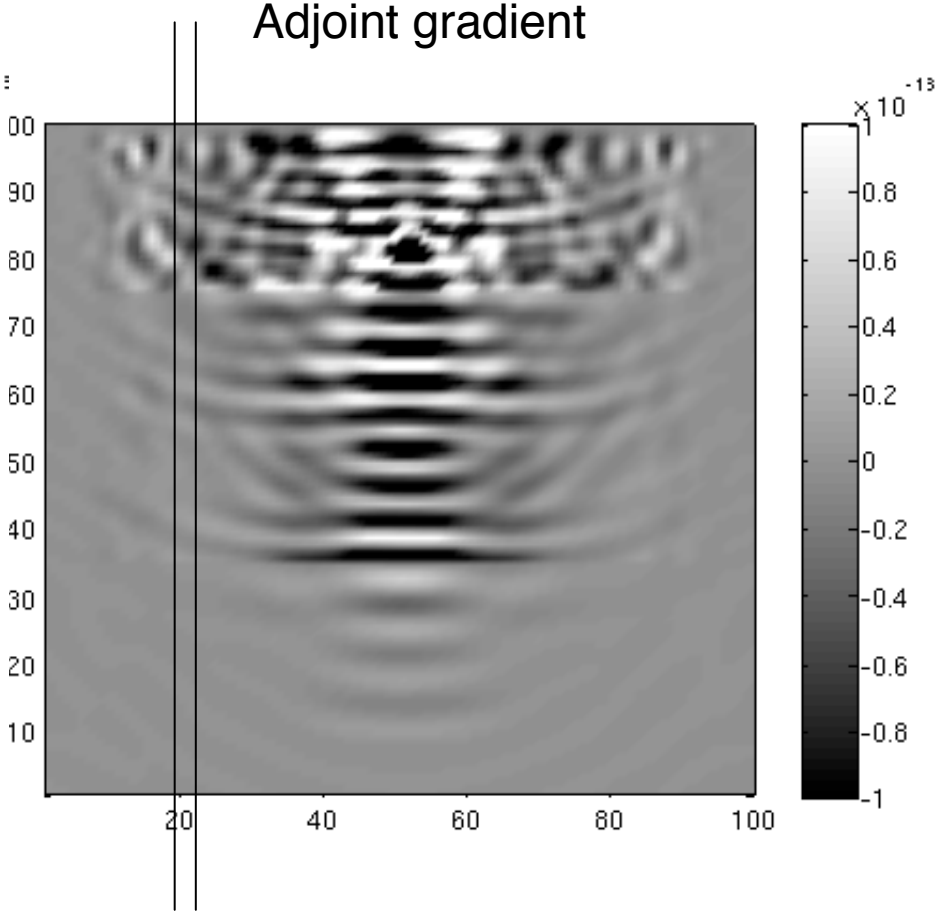
ExxonMobil
Upstream Research

ADOLC gradient



i=20,22

Adjoint gradient

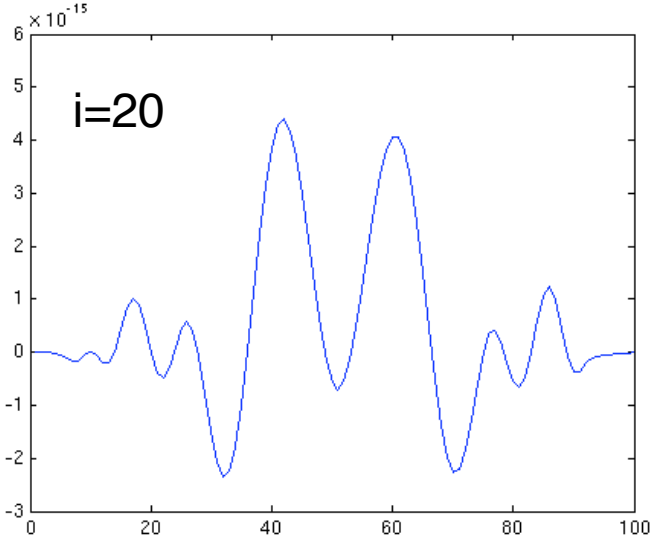


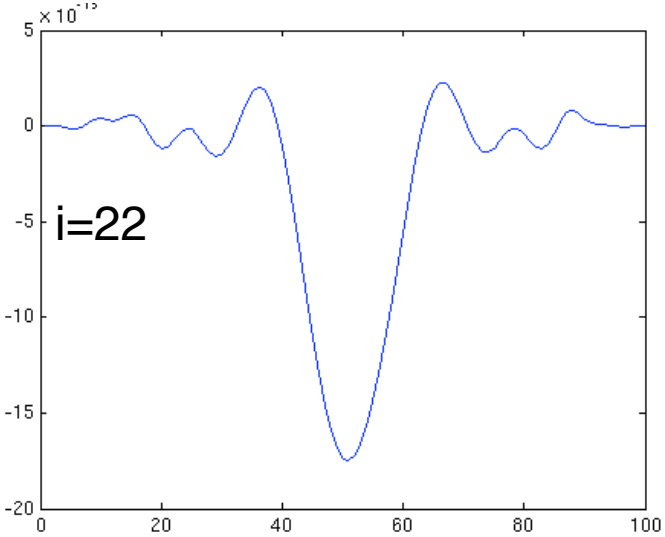
i=20,22

Numerical examples

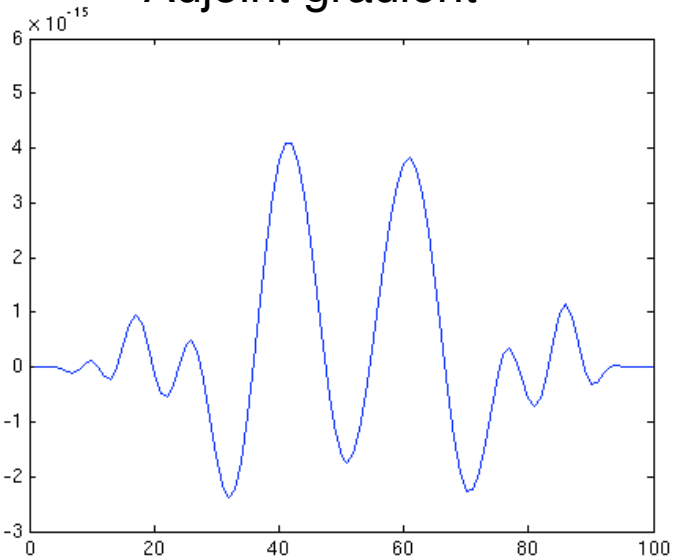
ExxonMobil
Upstream Research

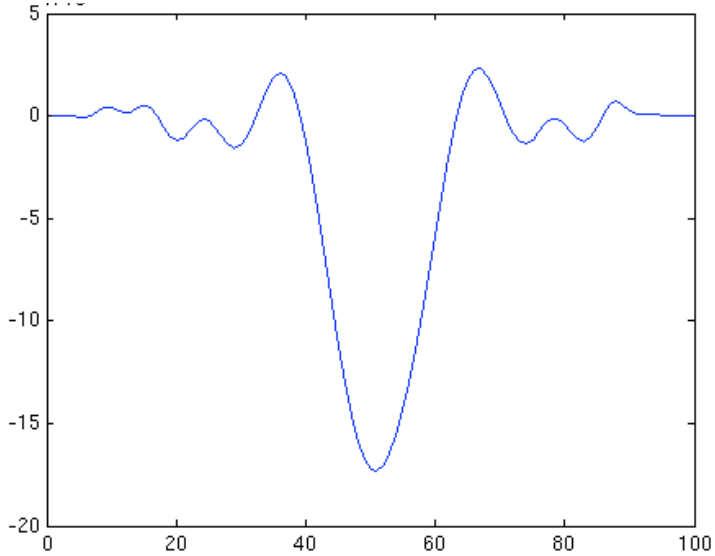
Adolc gradient





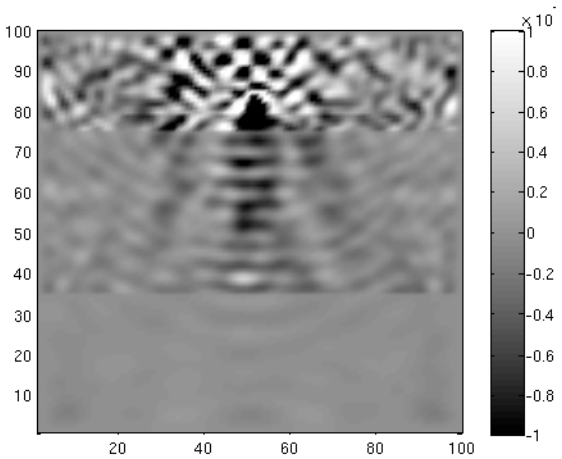
Adjoint gradient



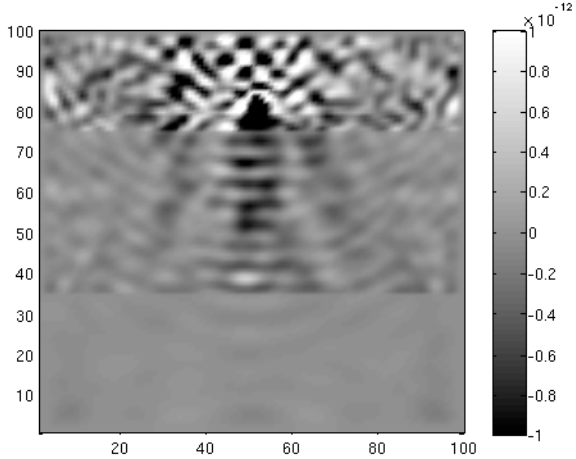


2nd-order algorithm without PML

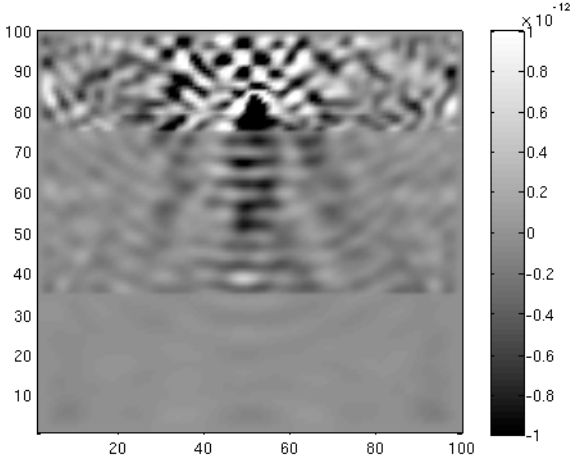
ExxonMobil
Upstream Research



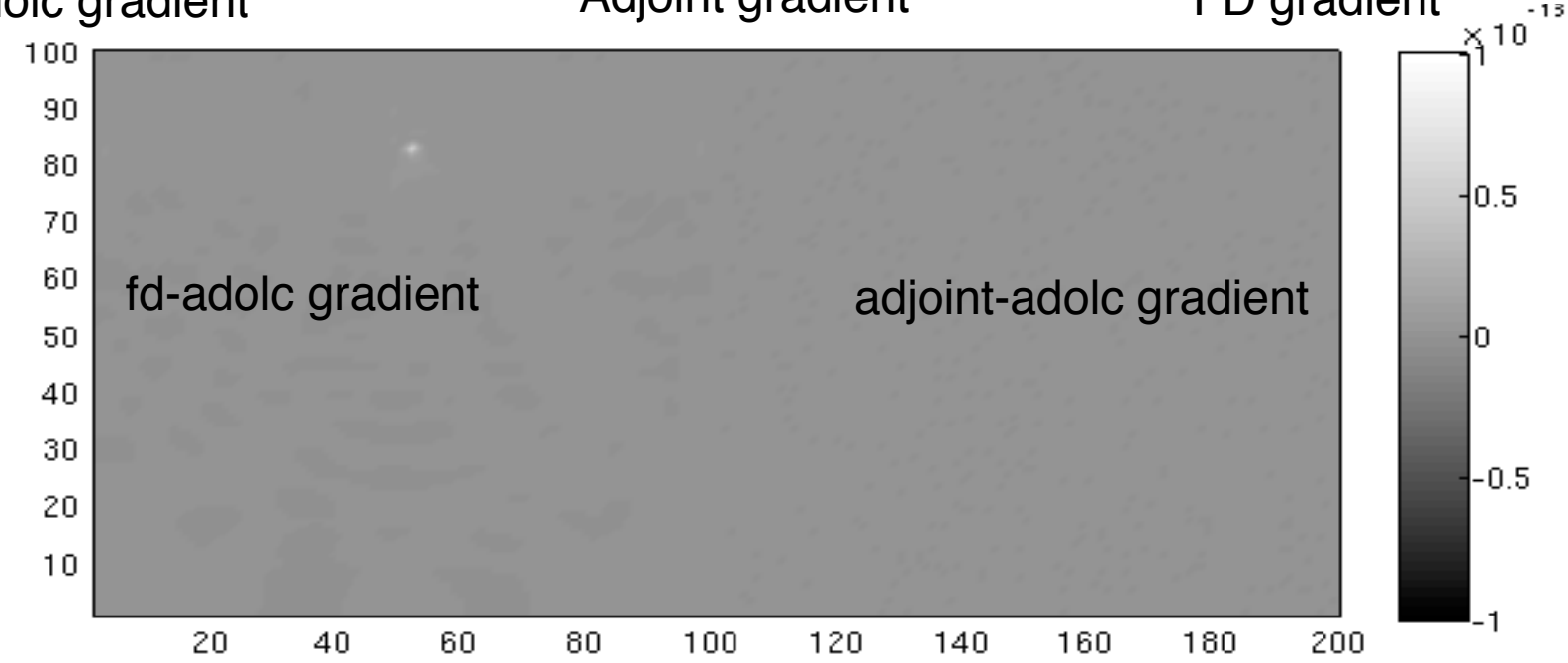
Adolc gradient



Adjoint gradient



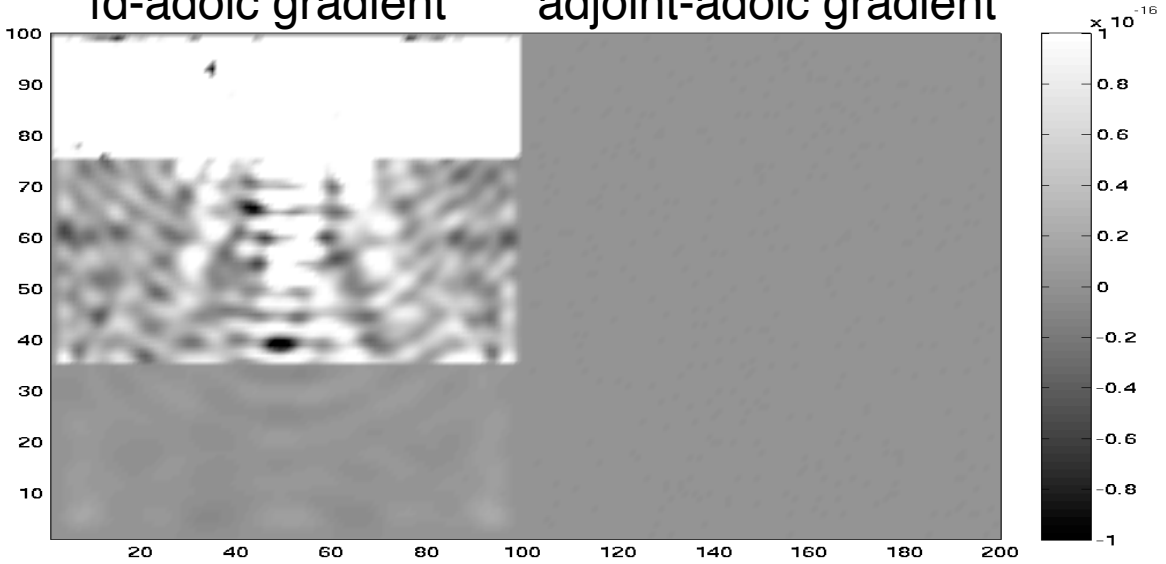
FD gradient



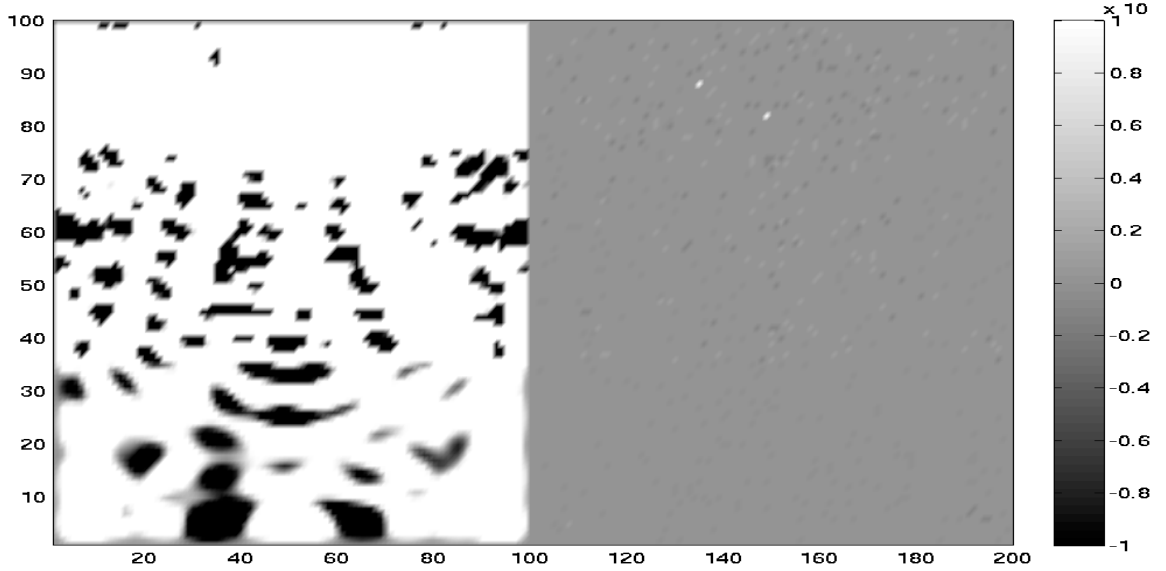
Differences, magnified

ExxonMobil
Upstream Research

fd-adolc gradient adjoint-adolc gradient



10^-16

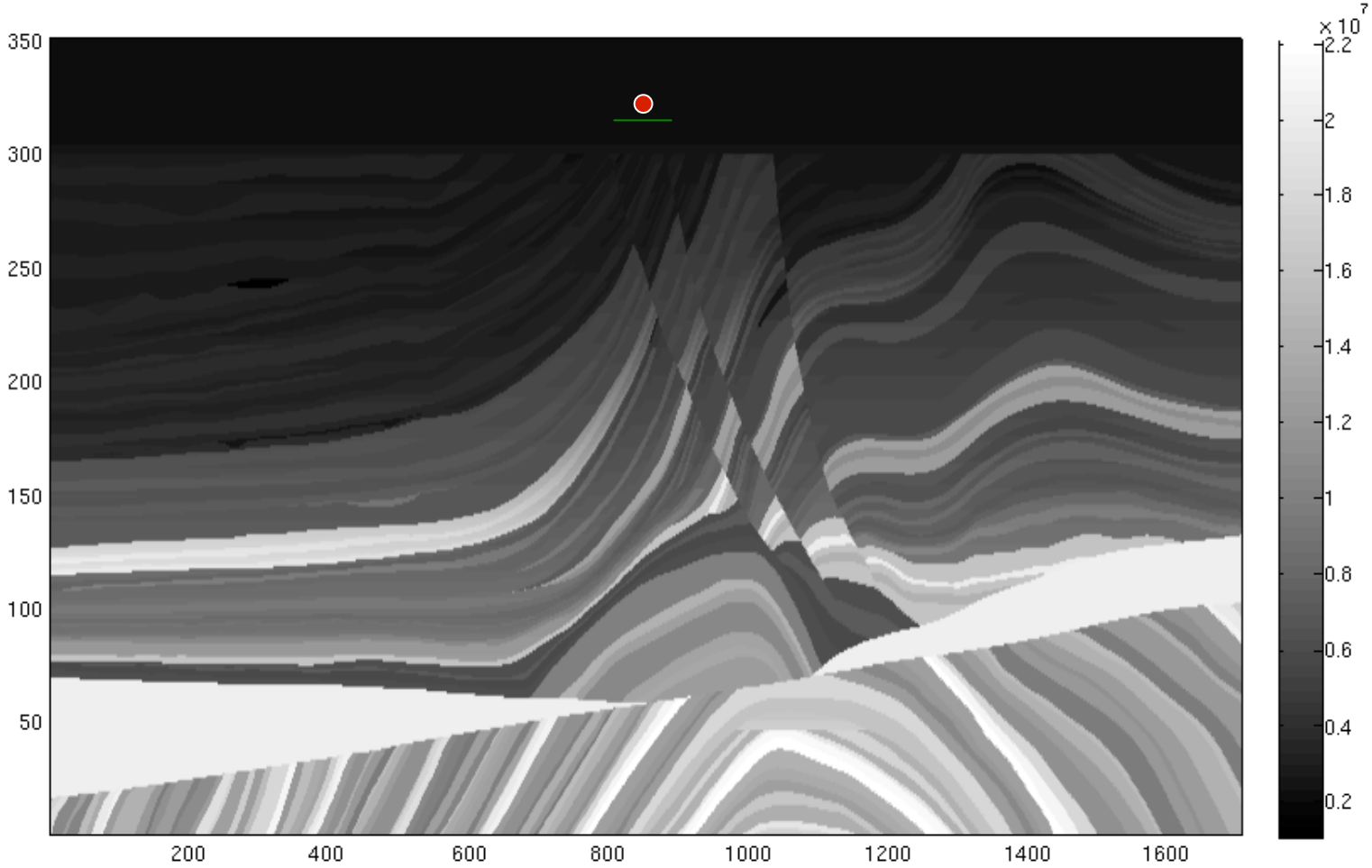


10^-18

Adolc for a relatively large model

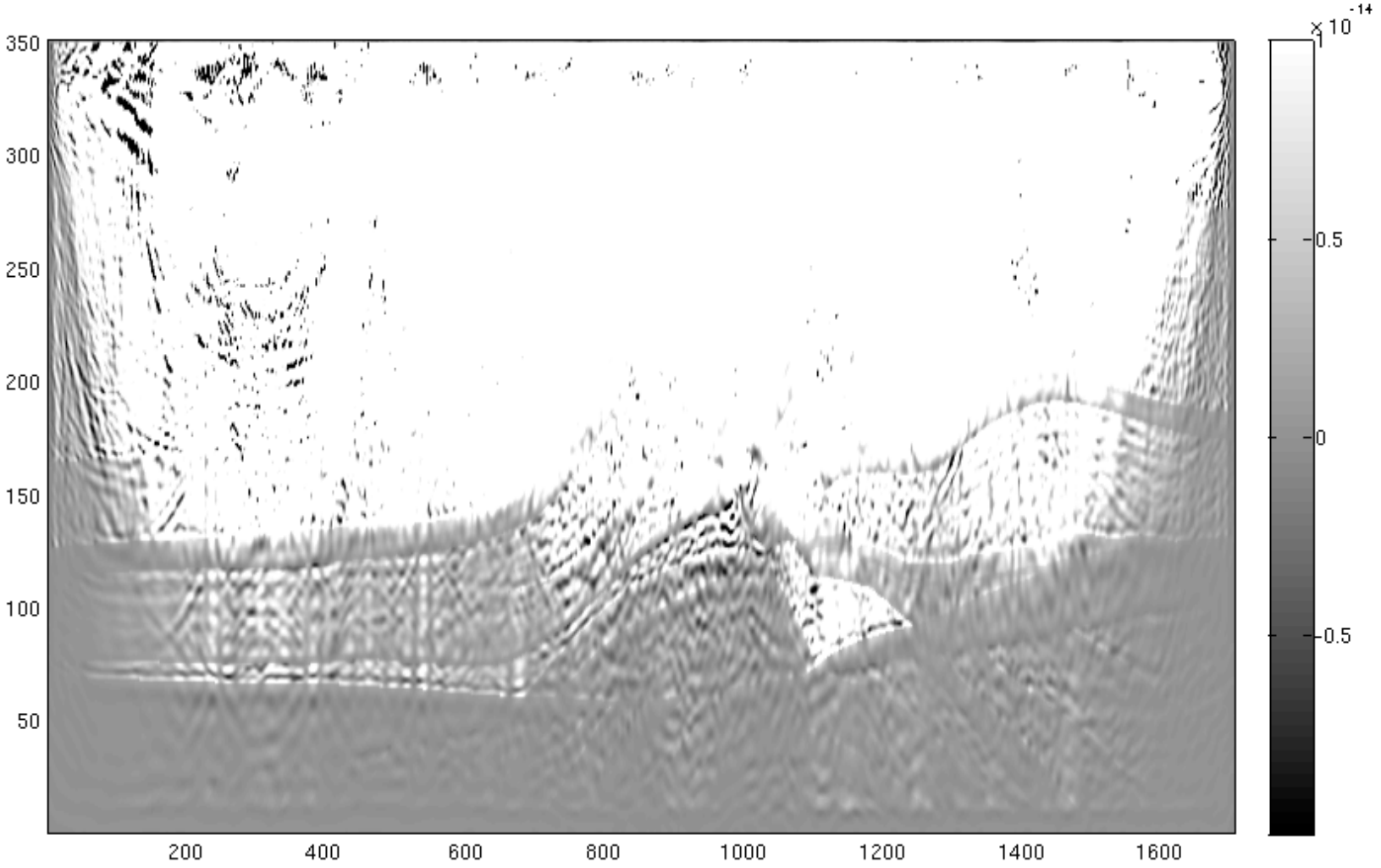
ExxonMobil
Upstream Research

Marmousi with grid size 1701*351;
Dirichlet boudary condition.
Initial model generated by smooth the true model (not shown here).



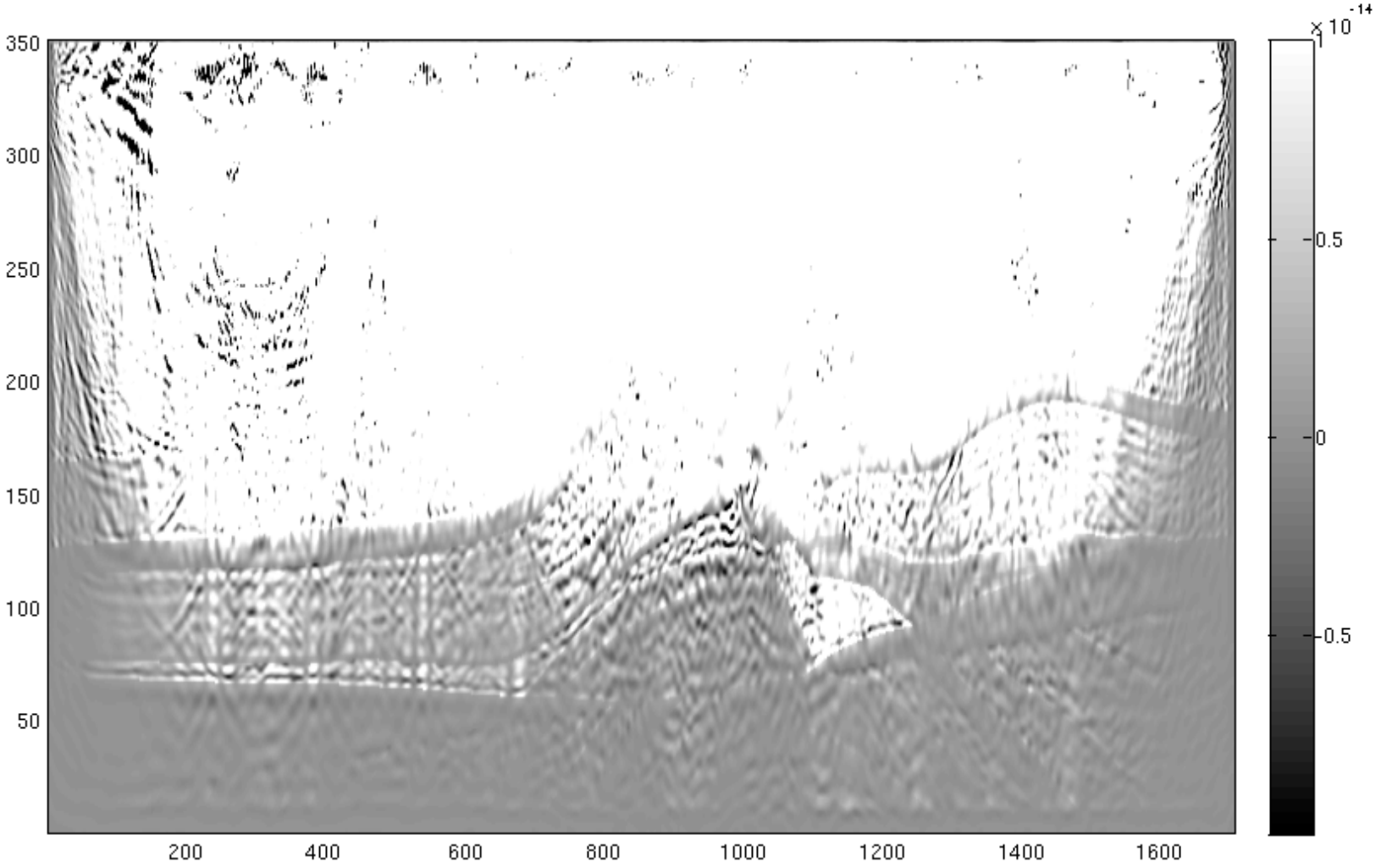
Adjoint Gradient

ExxonMobil
Upstream Research



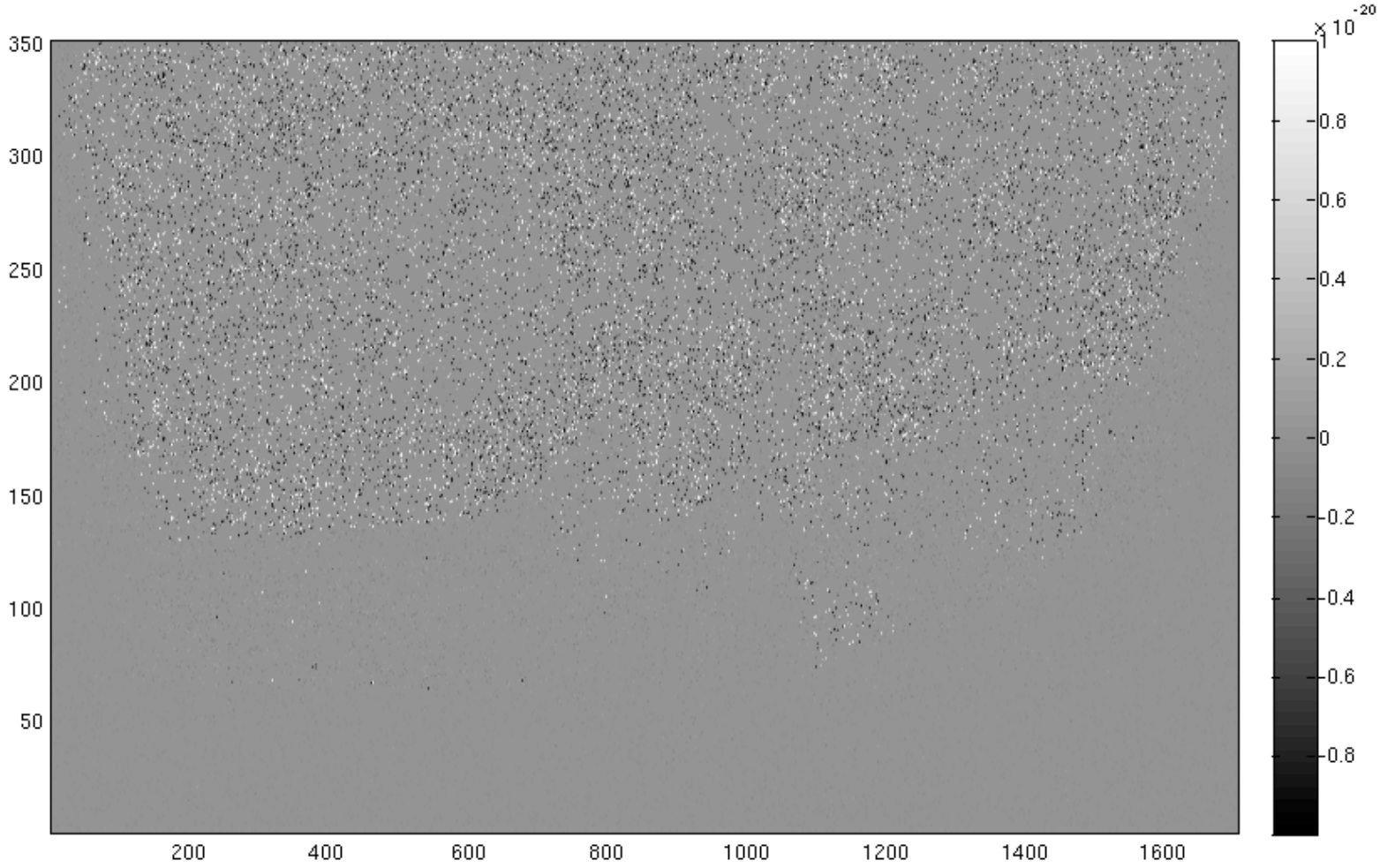
ADOLC Gradient

ExxonMobil
Upstream Research



ADOLC Gradient – Adjoint Gradient

ExxonMobil
Upstream Research



Discussion

ExxonMobil
Upstream Research

- Automatic differentiation is a valuable benchmarking tool for the validation of the current gradient computations.

- For the purpose of gradient and Hessian computations the reverse mode with checkpointing appeared to be the most efficient.

Backup

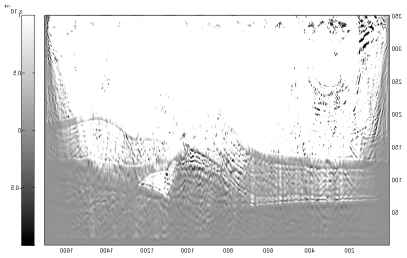
ExxonMobil
Upstream Research

Continuous vs. Discrete

ExxonMobil
Upstream Research

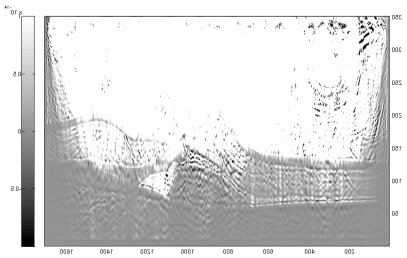
Continuous approach

Continuous equations → Adjoint equations

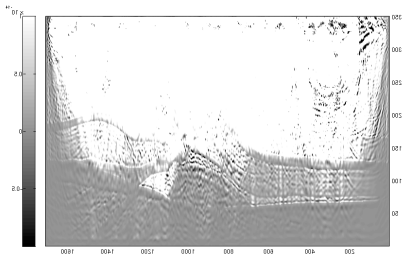


Direct equations E(c)= 0

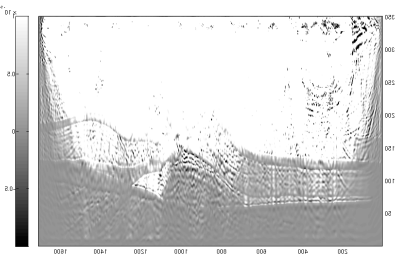
Discrete direct equations → Discrete adjoint equations



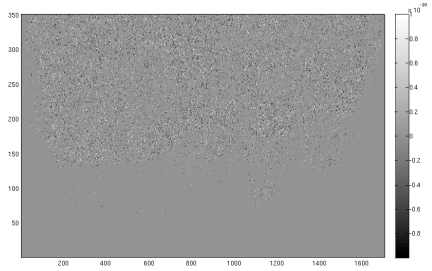
Discrete approach



-



=



Continuous Adjoint approach

ExxonMobil Upstream Research

Constrained minimization problem → adjoint field

$$j[\lambda,P,c] = E[P(c)] + \int dV \int dt \lambda \left[\left(-\frac{1}{c^2(x,y,z)} \partial_{tt} + \nabla^2 \right) P + s \right]$$

A sort of wave equation with source term = residual error

$$\frac{\delta j}{\delta P} = 0 \rightarrow \left[\frac{1}{c^2(x,y,z)} \partial_{tt} - \nabla^2 \right] \lambda(x,y,z,t) = S(x,y,z,t)$$

$$\frac{\delta j}{\delta \lambda} = 0 \rightarrow S(x,y,z,t) = [P(x,y,z,t) - Data(x,y,t)]\delta(z)$$

$$\lambda(x,y,z,T) = \partial_t \lambda(x,y,z,T) = 0$$

Back in time!

From P and λ evaluate the gradient

$$\frac{\delta j}{\delta c} = \frac{dE}{dc} = -\frac{2}{c^3} \int_0^T (\partial_t P)(\partial_t \lambda) dt$$

Continuous Adjoint approach

ExxonMobil Upstream Research

Constrained minimization problem → adjoint field

$$j[\lambda,P,c] = E[P(c)] + \int dV \int dt \lambda \left[\left(-\frac{1}{c^2(x,y,z)} \partial_{tt} + \nabla^2 \right) P + s \right]$$

A sort of wave equation with source term = residual error

$$\frac{\delta j}{\delta P} = 0 \rightarrow \left[\frac{1}{c^2(x,y,z)} \partial_{tt} - \nabla^2 \right] \lambda(x,y,z,t) = S(x,y,z,t)$$

$$\frac{\delta j}{\delta \lambda} = 0 \rightarrow S(x,y,z,t) = [P(x,y,z,t) - Data(x,y,t)]\delta(z)$$

$$\lambda(x,y,z,T) = \partial_t \lambda(x,y,z,T) = 0$$

Back in time!

From P and λ evaluate the gradient

$$\frac{\delta j}{\delta c} = \frac{dE}{dc} = -\frac{2}{c^3} \int_0^T (\partial_t P)(\partial_t \lambda) dt$$

Discrete vs. Continuous

ExxonMobil
Upstream Research

$$\frac{\partial P}{\partial t}(x,y,z,t) = \frac{1}{\Delta t}(P(x,y,z,t+\Delta t) - P(x,y,z,t))$$

$$\frac{\partial P}{\partial x}(x,y,z,t) = \frac{1}{\Delta x}(P(x+\Delta x,y,z,t) - P(x,y,z,t))$$

Tape mode

ExxonMobil
Upstream Research

- Development of the ADOLC-C software package is based on the decision to store all data necessary for derivative computation on “tapes”, where large applications require the tapes to be written out to the corresponding files. It means that the code starts to use I/O in the case of tape size exceeding size of the memory buffer.
  - Advantages:
    - Enables ADOLC-C to offer multiple functions
    - Very fast, if the buffer size is sufficient
    - Could be reused to compute gradients for multiple velocity models
  - Disadvantage:
    - Extensive use of I/O means a considerable drawback in terms of run time due to the excessive disk accesses. Could be used with the checkpointing procedure which gives us tradeoff between memory use and computational complexity.

AD basics: Overloading. Forward mode.

ExxonMobil Upstream Research

```
class doublet
{  double val;
   double dot;  }

doublet operator* (doublet a, doublet b)
{  doublet c;
   c.val = a.val*b.val;
   c.dot = a.dot*b.val+a.val*b.dot;
   return c;  }
```

In order to implement the forward approach to calculating a tangent vector, it is convenient to have a data type *doublet* that contains two floating point values, corresponding to the calculated values of a variable and the corresponding dotted variable, v_i and $\dot{v}_i$, respectively.

Hessian

ExxonMobil
Upstream Research