Salvus
IO, Adjoint & Sensitivity Kernels

MESS 2017
Modular Design through Template Mixins

- Wavefield I/O
- Boundary Conditions
- Additional Physics
- Basic Wave Equation
- Finite Elements

Finite Elements:
- Tetrahedra
- Hexahedra
- Quads
- Triangles

Additional Physics:
- Attenuation
- Anisotropy
- Coupling

Boundary Conditions:
- Absorbing
- Dirichlet

Advanced Physics:
- Acoustic
- Elastic
Modular Design through Template Mixins

```
template class WavefieldIO<
    IsotropicElastic<
        TensorGll3D<
            Hexahedra<HexP1>,
            ORDER>>;
```

- Wavefield I/O
- Boundary Conditions
- Additional Physics
- Basic Wave Equation
- Finite Elements
Modular Design through Template Mixins

- Wavefield I/O
- Boundary Conditions
- Additional Physics
- Basic Wave Equation
- Finite Elements

```cpp
template class WavefieldIO<
    AbsorbingBoundary<
        SolidToFluid<
            Scalar<
                TensorGll3D<
                    Hexahedra<HexP1>,
                    ORDER>>>;
```
Wavefield IO for Volume, Surface and Point Data
Wavefield IO for Volume, Surface and Point Data
Extensions

Time: 0.000000
Extensions

Time: 0.0000
**Forward Problem**
Given the material properties of the Earth, simulate the propagation of seismic waves through the subsurface.

\[ F(m) = d \]

**Inverse Problem**
Given a set of seismograms and a description of the seismic sources, determine the material parameters of the Earth.
Nonlinear Optimization Problem

Inverse Problem

\[ F(m) = d \]

- \( m \) model parameters
- \( d \) data
- \( F \) forward operator (the wave equation)

Reformulation as minimization problem

\[ \min_m \quad \frac{1}{2} \| F(m) - d \|^2 + \frac{1}{2} \| m - m_{\text{ref}} \|^2 \]

misfit
regularization
A Topography Map of Misfit Land

Iterative minimization based on **local** information (misfit, derivatives)

(i) Find a direction along which the misfit decreases.
(ii) Determine a step-length to update the model and ensure

\[ m^{k+1} = m^k + \alpha s \]
Full-Waveform Inversion

\[ \min_{\rho, C} \chi(u) = \frac{1}{2} \int_0^T (u(x_r, t) - u_{\text{obs}}(x_r, t))^2 \, dt \]

\[ \rho \frac{\partial^2}{\partial t^2} u - \nabla \cdot (C : u) = f \]

simplifying notation

\[ \min_m \chi(u) = \frac{1}{2} \| u - u_{\text{obs}} \|^2 \]

misfit

\[ L(m)u = f \]

wave equation
Deriving Adjoints without Physics

\[
\frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \frac{\partial}{\partial m} u(m) \cdot \delta m
\]
Deriving Adjoints without Physics

\[
\frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \frac{\partial}{\partial m} u(m) \cdot \delta m
\]

easy
Deriving Adjoint without Physics

\[
\frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \frac{\partial}{\partial m} u(m) \cdot \delta m
\]

\[
\frac{\partial}{\partial u} \left( \frac{1}{2} \| u - u_{\text{obs}} \|^2 \right) = (u - u_{\text{obs}})
\]
Deriving Adjoints without Physics

\[ \frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \frac{\partial}{\partial m} u(m) \cdot \delta m \]

easy

difficult
Deriving Adjoints without Physics

\[
\frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \delta u
\]

\[
L(m)\delta u = -L(\delta m)u(m)
\]

- easy
- difficult
Deriving Adjoint without Physics

\[ \frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \delta u \]

\[ L(m)\delta u = -L(\delta m)u(m) \]

Abstraction from basic linear algebra

\[ \mathbf{v}^T \mathbf{w}_i \quad \text{where} \quad \mathbf{A} \mathbf{w}_i = b_i, \quad i = 1, \ldots, N \]
Deriving Adjoints without Physics

\[ \frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \delta u \]

**Abstraction from basic linear algebra**

\[ L(m) \delta u = -L(\delta m)u(m) \]

\[ \mathbf{v}^T \mathbf{w}_i \quad \text{where} \quad \mathbf{A} \mathbf{w}_i = b_i, \quad i = 1, \ldots, N \]

- linear systems
- vector-vector products
Deriving Adjoints without Physics

\[ \frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \delta u \]

Abstraction from basic linear algebra

- \( N \) linear systems
- \( N \) vector-vector products

\[ L(m)\delta u = -L(\delta m)u(m) \]

\[ \mathbf{v}^T \mathbf{w}_i \quad \text{where} \quad \mathbf{A}\mathbf{w}_i = b_i, \quad i = 1, \ldots, N \]

\[ \mathbf{A}^T \mathbf{d} = \mathbf{v} \]

\[ \mathbf{v}^T \mathbf{w}_i = (\mathbf{A}^T \mathbf{d})^T \mathbf{w}_i = \mathbf{d}^T \mathbf{A}\mathbf{w}_i = \mathbf{d}^T b_i \]
Deriving Adjoints without Physics

\[
\frac{\partial}{\partial m} \chi(u(m)) \cdot \delta m = \frac{\partial}{\partial u} \chi(u(m)) \cdot \delta u
\]

\[v^T w_i\]

\[N\] linear systems

\[N\] vector-vector products

\[A w_i = b_i\]

\[L(m) \delta u = -L(\delta m) u(m)\]

Abstraction from basic linear algebra

\[v^T w_i \text{ where } A w_i = b_i, \quad i = 1, \ldots, N\]

\[A^T d = v\]

adjoint equation

\[1\] linear system

\[N\] vector-vector products

\[v^T w_i = (A^T d)^T w_i = d^T A w_i = d^T b_i\]
Back to the Wave Equation

- The adjoint of a linear operator is the generalization of the transpose of a matrix

\[
\int_0^T \int_G \rho(x) \frac{\partial^2}{\partial t^2} u(x, t) \cdot \varphi(x, t) \, dx \, dt \\
+ \int_0^T \int_G (C(x) : \varepsilon(u)(x, t)) : \varepsilon(\varphi)(x, t) \, dx \, dt
\]
The adjoint of a linear operator is the generalization of the transpose of a matrix

\[ \int_0^T \int_G \rho(x) \frac{\partial^2}{\partial t^2} u(x, t) \cdot \varphi(x, t) \, dx \, dt + \int_0^T \int_G (C(x) : \varepsilon(u)(x, t)) : \varepsilon(\varphi)(x, t) \, dx \, dt \]

= \int_0^T \int_G \rho(x) \frac{\partial^2}{\partial t^2} \varphi(x, t) \cdot u(x, t) \, dx \, dt

2x integration by parts in time
+ initial time conditions (forward)
+ final time conditions (adjoint)
Back to the Wave Equation

- The adjoint of a linear operator is the generalization of the transpose of a matrix

\[
\int_0^T \int_G \rho(x) \frac{\partial^2}{\partial t^2} u(x, t) \cdot \varphi(x, t) \, dx \, dt \\
+ \int_0^T \int_G (C(x) : \varepsilon(u)(x, t)) : \varepsilon(\varphi)(x, t) \, dx \, dt
\]

\[
= \int_0^T \int_G \rho(x) \frac{\partial^2}{\partial t^2} \varphi(x, t) \cdot u(x, t) \, dx \, dt \\
= \int_0^T \int_G (C(x) : \varepsilon(\varphi)(x, t)) : \varepsilon(u)(x, t) \, dx \, dt
\]
Sensitivity Kernels

\[ K_\rho(x) = \int_0^T \frac{\partial^2}{\partial t^2} u(x, t) \cdot u^\dagger(x, t) \, dt, \]

\[ K_\lambda(x) = \int_0^T (\nabla \cdot u(x, t)) \cdot (\nabla \cdot u^\dagger(x, t)) \, dt, \]

\[ K_\mu(x) = \int_0^T \varepsilon(u)(x, t) : \varepsilon(u^\dagger)(x, t) \, dt. \]

How does the misfit change (up to first order) if we increase the model parameters?
Sensitivity Kernels

- Forward wavefield
- Memory usage
- Adjoint wavefield
- Interaction field

Time: 0.0 s
Kernel Computations

\[ K_\lambda = \int_0^T \text{tr}(\varepsilon(u^\dagger)) \cdot \text{tr}(\varepsilon(u)) \, dt \]

for (auto i=0; i<mStoreLocalFieldData.size(); i++)
    mStoreLocalFieldData[i] = &(mEpsilonVoigt[i]);

```
template <typename Element>
void IsotropicElastic3D<Element>::updateKernel() {
    this->mGradient +=
        (this->mEpsilonVoigt[0]
        + this->mEpsilonVoigt[1]
        + this->mEpsilonVoigt[2]) *
        (this->mAdjEpsilonVoigt[3]
        + this->mAdjEpsilonVoigt[4]
        + this->mAdjEpsilonVoigt[5]);
}
```

```
template <typename Element>
void IsotropicElastic3D<Element>::writeKernel() {
    Element::sumFieldsInExodusModel(kernelFields,
        Element::MapIntPtsToVtx(
            Element::applyTestAndIntegrate(this->mGradient)));
}
Kernel Computations

\[ K_\rho \quad K_\lambda \quad K_\mu \]
Kernel Computations
Unit Tests for Sensitivity Kernels

Example:
Check for a consistently discretized adjoint equation and gradient by a finite-difference approximation

\[ \nabla_m \chi(m) \cdot \delta m \approx \frac{1}{h} (\chi(m + h\delta m) - \chi(m)) \]
Automation is possible if

- Kernels and model parameterization are consistent
- File I/O and data transfer are efficiently handled
- A workflow management tool allows for reproducibility and user interaction
Final Remarks

- Salvus is work in progress
- Watch [http://salvus.io](http://salvus.io) for updates
- Sign up to [salvus@googlegroups.com](mailto:salvus@googlegroups.com)
- Please report bugs and feature requests