Towards Extreme Scale Seismic Wave Propagation Simulation for Earth

Václav Hapla, Martin van Driel, Michael Afanasiev, Christian Boehm, Lion Krischer, Andreas Fichtner
Towards Extreme Scale Seismic Wave Propagation Simulation for Earth Mars

Václav Hapla, Martin van Driel, Michael Afanasiev, Christian Boehm, Lion Krischer, Andreas Fichtner
Full Waveform Inversion Across Scales

non-destructive testing

medical imaging
Salvus

- software suite designed for large-scale, time-domain, full waveform modelling and inversion
- Spectral Element Method for forward problems
- Full Waveform Inversion (FWI) using adjoints and nonlinear optimization methods
- focus on both flexibility and performance
- modular set of packages: SalvusWave, SalvusOpt, SalvusMesher, SalvusFlow
- computation kernels in templated C++ with dependencies of PETSc and Eigen
- Python-based interface
Forward modelling: Spectral Element Method (SEM)

- **SEM** = basically FEM with high order shape function and particular quadrature rules
- **shape functions** = high order (4) piecewise polynomials
  - tensor-product Lagrange interpolants within each element
  - nodes are placed at the zeros of Legendre polynomials (Gauss-Lobatto points) mapped from reference domain to each element
- **Gauss-Lobatto quadrature**
  - efficiency is achieved by using Gauss-Lobatto quadrature for evaluating elemental integrals
  - quadrature points coincide with the nodal points
  - fast tensor-product techniques can be used for iterative matrix solution methods
  - results naturally in purely diagonal mass matrix
SEM consequences

- relatively large number of DOFs per element (e.g. 125 for hex with degree=4)
- fewer elements for the same accuracy
- good for modelling rapidly oscillating phenomena (waves, turbulations)
- diagonal mass matrix, trivial to invert
- matrix-free global stiffness matrix
  - elemental matrices assembled using Eigen in Salvus
Modular design through C++ template mixins

```cpp
template class WavefieldIO<
    AbsorbingBoundary<
        Acoustic<
            TensorGll3D<
                Hexahedra<HexP1>,
                PolynomialOrder>>>
```

- reduced code duplication
- no performance regression
- increase in compile time and file size
- optimized for waveform inversion, particularly in seismology
Example: stiffness computation in visco-elastic media

\[ \int_{\Omega} \varepsilon(v)(\mathbf{x}, t) \cdot (C(\mathbf{x}, t - \tau) : \varepsilon(u)(\mathbf{x}, t)) \, d\mathbf{x} \]

\[ \text{quad} \quad \text{isotropic} \quad \text{quad} \\
\text{hex} \quad \text{transversely isotropic} \quad \text{hex} \\
\text{tri} \quad \text{anisotropic} \quad \text{tri} \\
\text{tet} \quad \text{anisotropic} \quad \text{tet} \]

template <typename Element>
RealArray Visco<Element>::computeInternalForces(
    RealArray &u, RealArray &accel)
{
    mStrain = Element::computeGradient(u);
    mStress = computeStress(mStrain) - mMemoryVariables;
    accel = Element::applyGradTestAndIntegrate(mStress);
    updateMemoryVariables(mStrain);
}
PETSc DMPIlex

- PETSc's unstructured mesh API
- abstract mesh connectivity
- Directed Acyclic Graph (DAG) representation
  - Hasse diagram
- dimension-oblivious
- topology separated from discretisation
- PetscSection
  - describes irregular data arrays (CSR)
  - mapping DAG points to DOFs

[Knepley and Karpeev 2009; Logg 2009]
Combining elastic and acoustic elements

F0 - elastic
F1 - acoustic
E3 - coupling edge with 2 different physics
Waveform propagation in coupled media (movie)
Alternating acoustic and elastic cells (movie)
Our current challenge: NASA InSight mission 2018

- highly sensitive broadband seismometer will be placed on Mars
  - the spaceship already left in May
  - estimated arrival in November
- investigate deep interior structure of Mars
- elastic wave propagation simulations crucial to interpret data
- at the highest frequencies, the resulting system has trillions of spatial DOFs and requires hundreds of thousands of time steps
- number of sources being much greater than number of receivers (=1)
  - exploit wave equation reciprocity
  - swap the roles of sources and receivers
“Measuring the Pulse of Mars“

SEIS Instrument
(covered with Wind & Thermal Shield)

https://mars.nasa.gov/insight/mission/instruments/seis/
• hypothetical Mars earthquake that can be explored once seismometer is deployed
• 3D crust and topography effects taken into account
• period 10 seconds, polynomial degree 4
• 6.5M elements, 300,000 time steps, $10^{10}$ DOFs
• CSCS Piz Daint supercomputer used for computation
• 7,200 cores – almost in realtime
• part of official NASA press materials at the mission launch broadcast
  • the only material used there from outside NASA
• fancy visualization, but too low resolution to do science…
  • needed mesh sizes will be 10-50 times bigger (higher frequencies)!
  • mesh I/O becomes a bottleneck
Mesh datafile format – current state

- **PETSc**
  - input vs output format asymmetry – much more input than output formats
  - only HDF5 can be used for both input and output
  - 2 redundant topologies stored: “PETSc” (DMPLex serialization) and “visualization” topology
    - sequential HDF5 reader just for “PETSc” topology
    - parallel HDF5 writer for both topologies
  - only Salome MED can be loaded in parallel; other formats loaded sequentially onto rank 0

- **Salvus**
  - mesh loaded from Exodus (DMPLexCreateExodusFromFile())
  - model data loaded in a custom way (Exodus API functions called directly)
  - uses sequential Exodus loader = bottleneck for Salvus
    - load the whole mesh onto rank 0
    - distribute mesh from rank 0 onto all remaining ranks
New format – selected requirements

- parallel I/O
- able to specify material parameters (per element, node, integration points)
- the same format for input and output
  - flexible workflows
  - e.g. produce mesh — partition — enhance the mesh — save mesh — visualize — manual edit — use as input again
- separate mesh topology (connectivity) and geometry (coordinates)
XDMF (eXtensible Data Model and Format)

- separates *light* and *heavy* data
- heavy data
  - raw binary data holding just "values"
  - stored using HDF5 with all its advantages
  - read and write data in large, contiguous chunks to achieve optimal I/O performance
- light data
  - metadata such as number type, precision, location, dimensions
  - give “meaning” to the “values”, and specify relations
  - stored using XML
  - both machine- and human-readable without any special tool
Current PETSc support for XDMF

- XMDF file produced by a Python script run after the PETSc app
- gives meaning to “visualization” topology and coordinates
- one-way
- meant just for ParaView visualization
Principal decision

- new format for Salvus can be XDMF!
- but current PETSc support is insufficient

- improve XDMF support in PETSc instead of implementing a new format from scratch in Salvus
  - take Salvus as an example of a complex simulation package
  - community use and/or further contributions
  - other FEM codes using PETSc can benefit as well
    - they can even grasp it as their native mesh format
Requirements met for XDMF?

- parallel I/O
  - yes
- able to specify material parameters (per element, node, integration points)
- equivalent feature to the exodus side-sets
  - both yes – using `<Set>` + `<Attribute>`
  - applicable to nodes, edges, faces or cells
- the same format for input and output
  - yes
- separate mesh topology (connectivity) and geometry (coordinates)
  - yes
Pitfalls

- no replacement for "PETSc native" HDF5
  - still limited set of element shapes whereas DMPlEx allows any possible shape
- mixed cell shapes
  - XDMF supports it
  - multiple <Topology> or one <Topology TopologyType="Mixed">
- Salvus custom additions
  - e.g. model and checkpointing data
  - using <Information> or only in the HDF5 file
- HDF5 criticism
  - see e.g. https://cyrille.rossant.net/moving-away-hdf5/
Parallel HDF5 reader

1) load element-disjoint portions of the mesh file directly onto different ranks
   - PETSc HDF5 interface allows easy loading of Vecs and ISs from HDF5
2) get a naively distributed DMplex representation right from beginning
   - DMplexCreateFromCellListParallel()
3) redistribute with DMplexDistribute()
   - load balancing using ParMETIS or PT-Scotch PetscPartitionerPartition()
   - redistribution (with parallel communication) DMplexMigrate()

- now in PETSc master branch
Parallel HDF5 reader – preliminary results

- test outside Salvus, only PETSc, to get an idea of the speedup
- 10.3M elements, 10.5M vertices (1.5e10 DOFs – already over 32-bit limit)
- 10 nodes of Piz Daint (120 cores) – finished today around 1AM 😊
- downside: does not compare quality of distribution (i.e. impact on solver time)

<table>
<thead>
<tr>
<th>mesh format</th>
<th>interpolate*</th>
<th>partitioner</th>
<th>load t</th>
<th>interpolate* t</th>
<th>distribute t</th>
<th>total t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 seq exo</td>
<td>0 before distribution</td>
<td>0 parmetis</td>
<td>14</td>
<td>151</td>
<td>135</td>
<td>300</td>
</tr>
<tr>
<td>1 par hdf5</td>
<td>0 after distribution</td>
<td>1 ptscotch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>151</td>
<td>121</td>
<td>286</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>2</td>
<td>166</td>
<td>182</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>50</td>
<td>66</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>14</td>
<td></td>
<td>fail (PETSc bug)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>fail (PETSc bug)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>33</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

* interpolate = compute edges & faces
TODOs

- XDMF file output
  - using XDMF API, some general XML IO API, or just plain ASCII output
- face sets using <Attribute>
- integration points
  - not directly supported
  - maybe <Attribute AttributeType="Vector"> ?
- minimal model storage
Thank you for your attention!

Double thanks to

Platform for Advanced Scientific Computing

for this conference and for paying me 😊