SeismoVLAB: A Parallel Object-Oriented Virtual Laboratory for Mesoscale Seismic Wave Propagation Simulations

National Earthquake Conference
March 5th, 2020, San Diego

Domniki Asimaki
Elnaz E. Seylabi
Albert Kottke
Danilo S. Kusanovic
The Presentation Outline.

➢ Motivation.
➢ The Software Design.
➢ External Packages.
➢ SeismoVLab.
   I. Coding Style.
   II. Code Performance.
   III. Code Extensions.
➢ Conclusions.
Motivation.

The FEM is the preferred methodology to approximate the linear or nonlinear responses of structure in structural and geotechnical engineering.

Several FEM Software are available: MSC Nastran, ANSYS, Abaqus, LS-DYNA, ETABS, OpenSEES, FreeFEM++, FEniCS.

However,

Most of these softwares lack features such as DRM, PML, soil material models. In general, they are closed sources or too difficult to modify.

A code that allows to explore full fidelity mesoscale models, and in particular SSSI simulations.

https://tenmilesquare.com/when-is-the-right-time-to-innovate-your-business/
**Motivation.**

**SeismoVLab** is a simple, fast, and extendable C++ multi-platform finite element software (suitable for laptop, desktop, servers), provided with the state-of-the-art solvers and integrators.

**SeismoVLab** is designed to optimize ~km scale (mesoscale) simulations of dynamic, nonlinear wave propagation problems.

---

SeismoVLAB User Guide

[Danilo S. Kusanovic](dkusanov@caltech.edu)
[http://liantze.penguinattack.org](http://liantze.penguinattack.org)

[Elnaz Esfahani](elnaze@unr.edu)
[https://www.elnaz-esmaeilzadeh.com](https://www.elnaz-esmaeilzadeh.com)

[Domniki Asimaki](domniki@caltech.edu)
[https://www.asimaki.caltech.edu](https://www.asimaki.caltech.edu)

[https://bitbucket.org/liantze/aautorating-calculator](https://bitbucket.org/liantze/aautorating-calculator)

February 29, 2020

Available for Linux, MacOSX, Windows (Serial Version) and Linux (Parallel Version) [multi-platform]
Seismo-VLab is implemented using classes: [extendable]

Developers can provide their own subclasses to personalize applications.

Partition is done by minimizing load-imbalance and is distributed to each processor using MPI.
Eigen C++ Template Library: [simple]

- It minimizes the debugging process since the structure of matrices and vector operations are similar to those in Matlab and Python.
- It is fully optimized, compared to the Intel Math Kernel Library (MKL).

MUMPS: [fast]

- Solution of large linear systems with symmetric positive definite matrices, general symmetric matrices, and general unsymmetric matrices.
- Interfaces to MUMPS: Fortran, C, Matlab and Scilab.

PETSC: [fast]

- Parallel vectors/matrix operations.
- Scalable parallel preconditioners.
- Krylov subspace methods.

https://eigen.tuxfamily.org/
http://mumps.enseeiht.fr/
https://www.mcs.anl.gov/petsc/
The Coding Style:

[simple, extendable, maintainable]

```cpp
// Compute the stiffness matrix of the element using gauss-integration.
Eigen::MatrixXd lin3DHexa8::ComputeStiffnessMatrix(void)
{
    // Element stiffness matrix definition:
    Eigen::MatrixXd StiffnessMatrix(24, 24);
    StiffnessMatrix.fill(0.0);

    // Gets the quadrature information.
    Eigen::VectorXd wi = QuadraturePoints->GetWeights();
    Eigen::MatrixXd xi = QuadraturePoints->GetPoints();
    unsigned int nPoints = QuadraturePoints->GetNumberOfQuadraturePoints();

    // Compute element stiffness matrix as in Equation (1.18).
    for(unsigned int i = 0; i < nPoints; i++)
    {
        // Jacobian matrix.
        Eigen::MatrixXd Jij = ComputeJacobianMatrix(xi(i, 0), xi(i, 1), xi(i, 2));

        // Compute Strain-Displacement Matrix at Gauss Point.
        Eigen::MatrixXd Bij = ComputeStrainDisplacementMatrix(xi(i, 0), xi(i, 1), xi(i, 2), Jij);

        // Gets material tangent matrix at Gauss point.
        Eigen::MatrixXd Cij = theMaterial[i]->GetTangentStiffness();

        // Numerical integration in Equation (1.20).
        StiffnessMatrix += wi(i)*fabs(Jij.determinant())*Bij.transpose() * Cij * Bij;
    }

    return StiffnessMatrix;
}
```
SeismoVLab Features.

[Domain Decomposition]

[2D/3D Domain Reduction Method]

[2D/3D Perfectly Matched Layers]

[Solution output compatible with PARAVIEW]

[Nonlinear Materials]
Application in Earth Dam.

The Code Performance:

Example I: *2D man-made topography problem*

The finite element model has

- **57770** degree of freedom
- **28885** nodes
- **28847** elements,
- **203** boundary nodal forces
- **1501** time steps.
Application in Earth Dam.

The Code Performance:

Example I: 2D *man-made topography problem*

![Graph showing speed-up vs number of processors]

<table>
<thead>
<tr>
<th>Software</th>
<th>Time [min]</th>
<th>RAM [MB]</th>
<th>Leaks [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeismoVLab</td>
<td>23.0</td>
<td>220.1</td>
<td>0.0</td>
</tr>
<tr>
<td>OpenSEES</td>
<td>283.5</td>
<td>825.1</td>
<td>0.9</td>
</tr>
<tr>
<td>Performance</td>
<td>~ 12×</td>
<td>~ 4×</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.1: Performance comparison between Seismo-VLab and OpenSEES. Speed-up, maximum memory usage, and memory leaks for a single core execution.

[scalable] [fast]
Application in 3D SSI Problem.

The Code Performance:

Example II: *3D fully-coupled SSI problem*

The finite element model has

- **380000** degree of freedom
- **121944** nodes
- **105825** elements
  - 2164 `lin3DFrame2`
  - 7388 `lin3DShell4`, and
  - 96273 `lin3DHexa8`, `PML3DHexa8`
- **7** different materials
- **19** different sections
- **26247** Constraint.

[Link to Map: goo.gl/maps/m4YgMumThCM2]
Application in 3D SSI Problem.

The Code Performance:

Example II: **3D fully-couple SSI problem**

<table>
<thead>
<tr>
<th>np</th>
<th>Assembler [s]</th>
<th>Solver [s]</th>
<th>Recorder [s]</th>
<th>Total Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.78</td>
<td>348.75</td>
<td>0.95</td>
<td>365.06</td>
</tr>
<tr>
<td>2</td>
<td>5.08</td>
<td>184.59</td>
<td>0.50</td>
<td>196.17</td>
</tr>
<tr>
<td>4</td>
<td>2.62</td>
<td>95.22</td>
<td>0.25</td>
<td>99.64</td>
</tr>
<tr>
<td>8</td>
<td>1.45</td>
<td>80.98</td>
<td>0.13</td>
<td>83.57</td>
</tr>
<tr>
<td>16</td>
<td>0.78</td>
<td>56.98</td>
<td>0.07</td>
<td>58.78</td>
</tr>
</tbody>
</table>

[faster]

<table>
<thead>
<tr>
<th>np</th>
<th>Assembler [s]</th>
<th>Solver [s]</th>
<th>Recorder [s]</th>
<th>Total Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>5.03</td>
<td>22.19</td>
<td>0.56</td>
<td>30.81</td>
</tr>
<tr>
<td>4</td>
<td>2.41</td>
<td>14.40</td>
<td>0.25</td>
<td>18.40</td>
</tr>
<tr>
<td>8</td>
<td>1.43</td>
<td>9.31</td>
<td>0.13</td>
<td>11.55</td>
</tr>
<tr>
<td>16</td>
<td>0.76</td>
<td>7.45</td>
<td>0.08</td>
<td>8.84</td>
</tr>
</tbody>
</table>

It can be reduced up to:  < 0.01 sec, for np = 10
Application in Random Media Problems.

SeismoVLab Extensions:

I. Spatial variability of soil properties.
Application in Inverse Problems.

SeismoVLab Extensions:

II. Soil parameter estimation for ROM.
Conclusions.

1) **SeismoVLab**: a **simple**, **fast**, and **extendable** C++ FEM for solving mesoscale problems.

2) **SeismoVLab** allows new elements, materials, integrators, and algorithms to be incorporated in a simple manner.

3) The **Eigen C++ library** allows simple manipulations at the element and material level, and this contributes to **SeismoVLab**’s extendability.

4) Iterative solvers (Krylov-subspace) are well-suited for 3D problems, and are faster than direct solvers.

5) Direct solvers are well-suited for 2D problems and 3D problems if the Cholesky decomposition needs to be performed once.

6) As expected, **SeismoVLab** scales well as long as the solver does.
Acknowledgement.

PG&E for the financial support to the authors for conducting doctoral studies/Post doctoral research at The California Institute of Technology.

Professor Dennis M. Kochmann who has motivated the development of SeismoVLab at its initial stage.

Professor Laurent Stainier for helping with the integrator/algorith interaction for material nonlinear and large-deformation problems.

Professor Louie L. Yaw for helping with the co-rotational formulation of truss and beam elements and for the implementation of the generalized displacement control algorithm.

Paolo Celli PhD for helping with the code structure, organization and formulation of the large deformation solid elements.

Justin Phalen (PE, GE) and Barry Zheng (PhD) from SLATE GEOTECHNICAL CONSULTANTS for testing and validating SiesmoVLab in real Earth dam problems, and providing useful comments in writing the user’s manual.
SeismoVLAB Developer Team.

Dominiki Asimaki.  
Professor  
California Institute of Technology.

Elnaz E. Seylabi.  
Assistant Professor  
University of Neveda Reno.

Albert Kottke.  
Pacific Gas and Electric  
San Mateo.  
California.

Peyman Ayoubi.  
PhD (c) Caltech  
Pasadena.

Danilo S. Kusanovic.  
PhD (c) Caltech  
Pasadena.

THANK YOU FOR YOUR ATTENTION.