Simulating seismic wave propagation with **SW4**

N. Anders Petersson
Center for Applied Scientific Computing
Computations Directorate

CIG webinar, November 12, 2015

Lawrence Livermore National Laboratory

LLNL-PRES-677035
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Outline

- What is implemented in sw4 (seismic waves, 4^{th} order)?
- Setting up the simulation
  - Sources
  - How do I pick the grid size?
  - Topography
  - Material model, visco-elastic modeling
  - Output options
- Running sw4
- Practical suggestions
  - Workflow

The SW4 code solves the time dependent 3D visco-elastic wave equation

- Summation-by-parts finite difference method
  - 4^{th} order accurate in space+time, energy stable
- Cartesian geometry (earth's curvature neglected)
  - Projection from geographical to Cartesian coordinates
- Elastic or visco-elastic material model
  - Isotropic material model ($\mu > 0$, $\lambda > 0$, $\rho > 0$)
  - Quality factors Q_P and Q_S, approx. constant in frequency band
  - Anisotropic materials (21 parameter model)
- Curvilinear coordinates
  - Flat or realistic topography; Mesh generated by SW4
  - Absorbing far-field super-grid layers
- Moment tensor sources or point forces
  - Many pre-defined time functions, or user-specified
  - Complex ruptures: many sources or SRF file (v. 1)
- Output options:
  - Time-series at receivers, solution on 2-D cross-sections, GMT scripts
- Local mesh refinement (coming soon)

Summation by parts discretization mimic integration by parts for finite differences

- Elastic wave equation:

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \mathbf{L}\mathbf{u} + \mathbf{f}(\mathbf{x}, t)$$

$$\mathbf{L}\mathbf{u} := \nabla \cdot \mathcal{T}$$

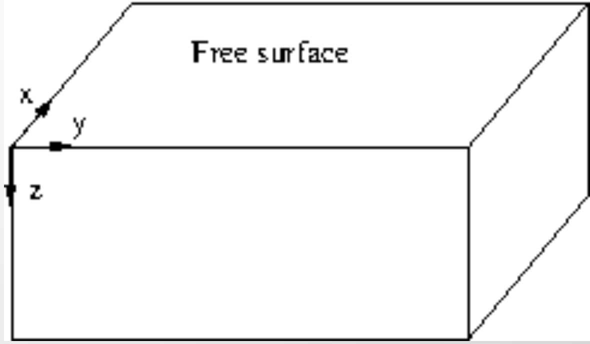
- Discretized in space:

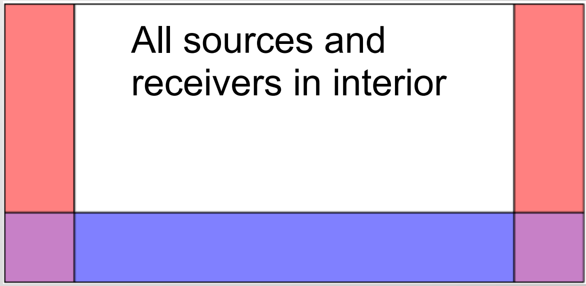
$$\rho \frac{d^2 \mathbf{v}}{dt^2} = \mathbf{L}_h \mathbf{v} + \mathbf{f}(\mathbf{x}, t)$$

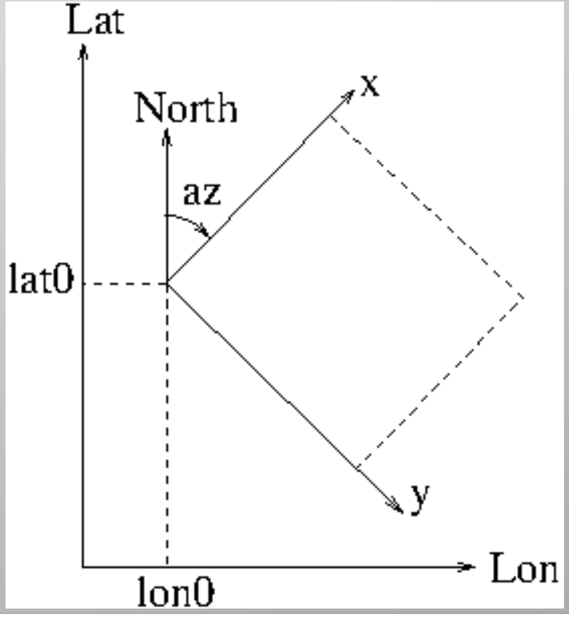
- Symmetry and negative definite properties of ‘**L**’ preserved by summation by parts discretization ‘$\mathbf{L}_h$’
- Numerical scheme is energy stable
- Theory and papers on our web page: **computation.llnl.gov/project/serpentine/**

sw4 uses SI-units (MKS), angles in degrees

- Box-shaped computational domain
  - Right-handed system, z positive downwards
- Top surface optionally follows topography
- Free surface boundary condition on top surface
- Super-grid absorbing layers on all other sides
  - SG layers included in computational domain
  - Default thickness: 30 grid points
  - Smallest Cartesian grid: 60x60x30
- Locations in Cartesian or geographic coordinates
- Geographic coordinates are mapped:
  - Default: spheroidal projection
  - Proj.4 library for better accuracy
- Rotate grid with azimuth angle (az)
  - az=0: x-axis = North, y-axis = East







The sw4 simulation is specified by the command file

- cat seismic1.in

```
fileio path=lamb-res
grid x=10e3 y=10e3 z=5e3 h=50
time t=8.0
block vp=1.7320508076e+03 vs=1000 rho=1500
source type=C6SmoothBump x=5e3 y=5e3 z=0 fz=1e13 freq=1 t0=0
# Time history of solution (comments or blank lines ignored)
rec x=6e3 y=8e3 z=0 file=v1s
# Snapshot of solution every 0.5 seconds
image mode=uz z=0 file=lamb timeInterval=0.5
```

- mpirun –np 8 sw4 seismic1.in

SW4 implements two types of kinematic source terms

- Point force:

$$g(t)\mathcal{F}\delta(\mathbf{x}-\mathbf{x}_*)$$

- Moment tensor source:

$$g(t)\mathcal{M}\nabla\delta(\mathbf{x}-\mathbf{x}_*)$$

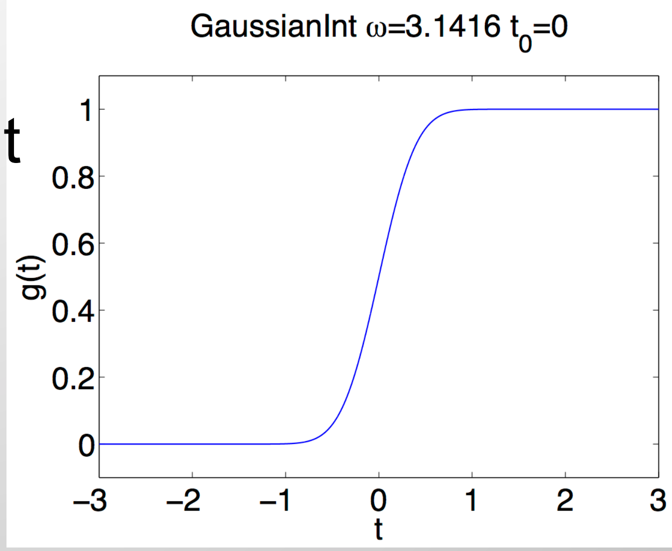
- We discretize the Dirac distribution δ(x) by imposing moment conditions
  - 6x6x6 point stencil in 3-D
- 4^{th} order accuracy away from the source
- The sources can be located independently of the grid

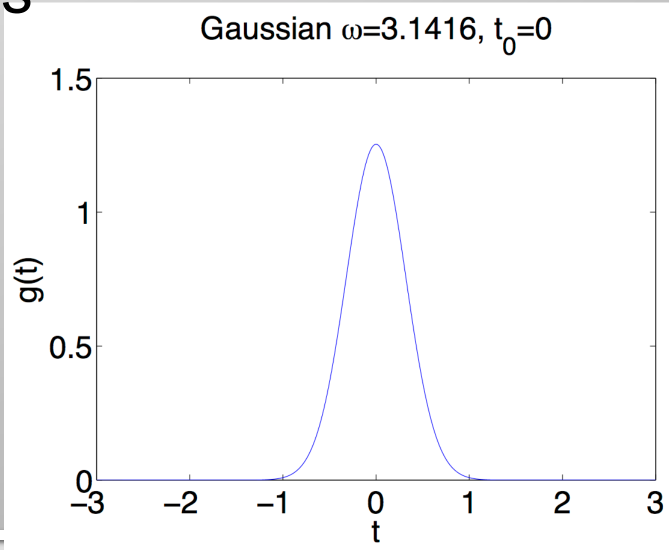
sw4 provides several options for specifying sources and time-functions

- Any number of point forces or moment tensor sources
  - Forces defined by 3-component vector $\mathbf{F}_i$
  - Moment tensor (symmetric) defined by 6 components M_{ij}, or
  - Seismic moment (M_0) and strike, dip, rake angles
- 14 pre-defined time functions (Gaussian, Ricker, Liu, …)
  - Frequency parameter and start/center time
- Dirac delta-distribution: triggers all frequencies on mesh
  - Discrete Green's functions. Motion must be filtered for accuracy
- User defined discrete time function (interpolated by spline)
- Complex ruptures can be specified by using many point sources or through an SRF file (version 1 format)

There are several ways of computing displacements, velocities, or acceleration

- Displacements correspond to source time-functions that tends to a constant (e.g. gaussianInt)
  - Displacements obtained directly
  - Velocities by differentiating once
  - Acceleration by differentiating twice
- Velocities correspond to time-functions that tend to zero but have non-zero integral (e.g. gaussian)
  - Velocities obtained directly
  - Displacements by integrating once
  - Acceleration by differentiating once



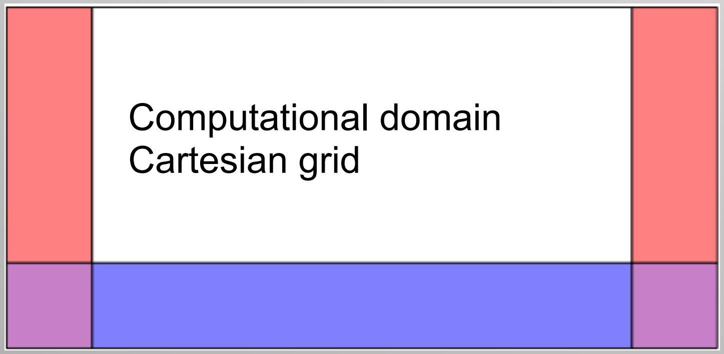


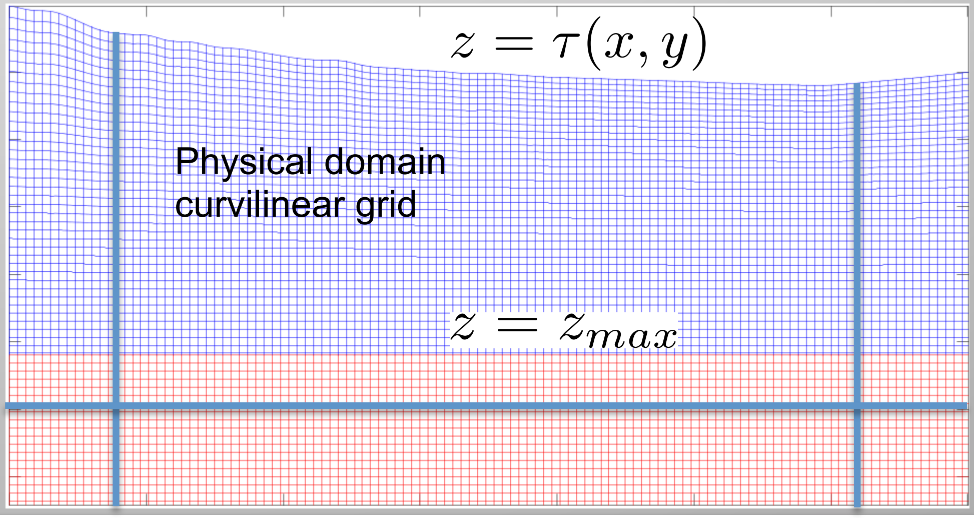
The highest frequency and the lowest wave speed determine the grid size

- Shortest wave length L_{min}=min C_S/f_{max}
- Grid points per shortest wave length $P_{min} = L_{min}/h$
- Good accuracy when P_{min} > 6 to 10 (depending on distance)
- Several options:
  - Tune f_{max} in source time function to not trigger unresolved waves
  - Use the `prefilter` command to pre-process all source time fcns
  - Calculate f_{max} = min $C_S/(P_{min}$ h), remove unresolved frequencies afterwards
- Relation between f_{max} and frequency parameter (`freq`) is different for each time function (see UG 4.4)
- `matlab/octave` scripts for plotting time function and Fourier transform in …sw4/tools: `fcnplot`, `ftfcnplot`

Curvilinear coordinates are used to treat general topography

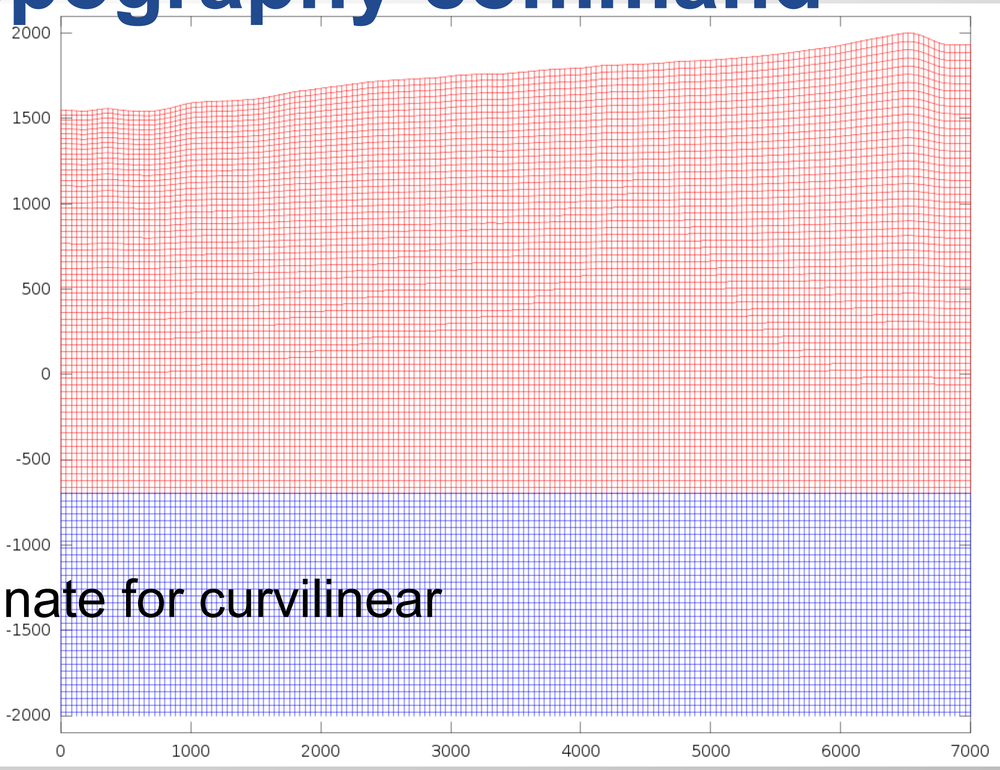
- Polynomial stretching in vertical direction
  - $$z = \tau(x,y)\,(1 - P(\zeta)) + z_{max} P(\zeta)$$
- Transform elastic wave equation to curvilinear coordinates
  - Metric terms similar to anisotropic material properties
- Also used for super-grid absorbing far-field layers





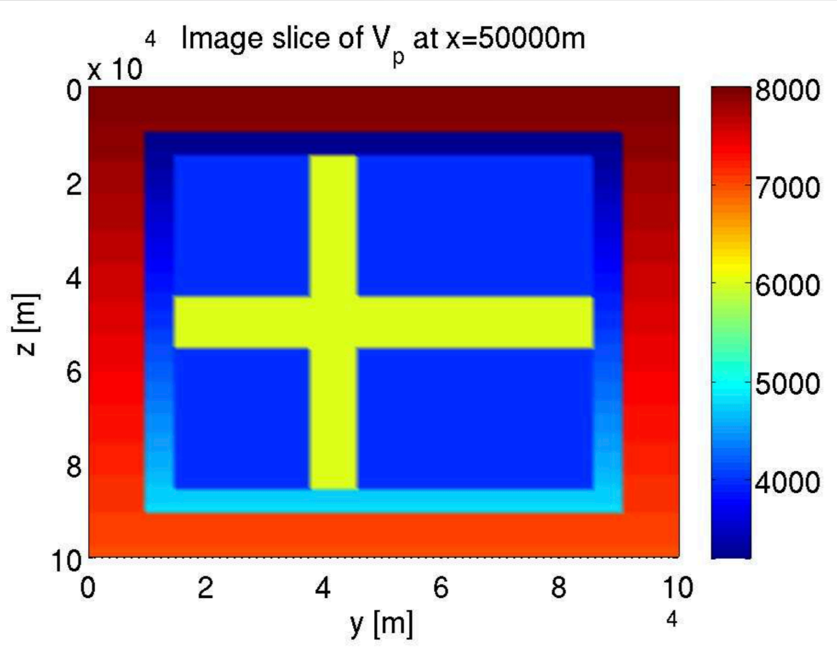
The curvilinear mesh is constructed by sw4 based on the topography command

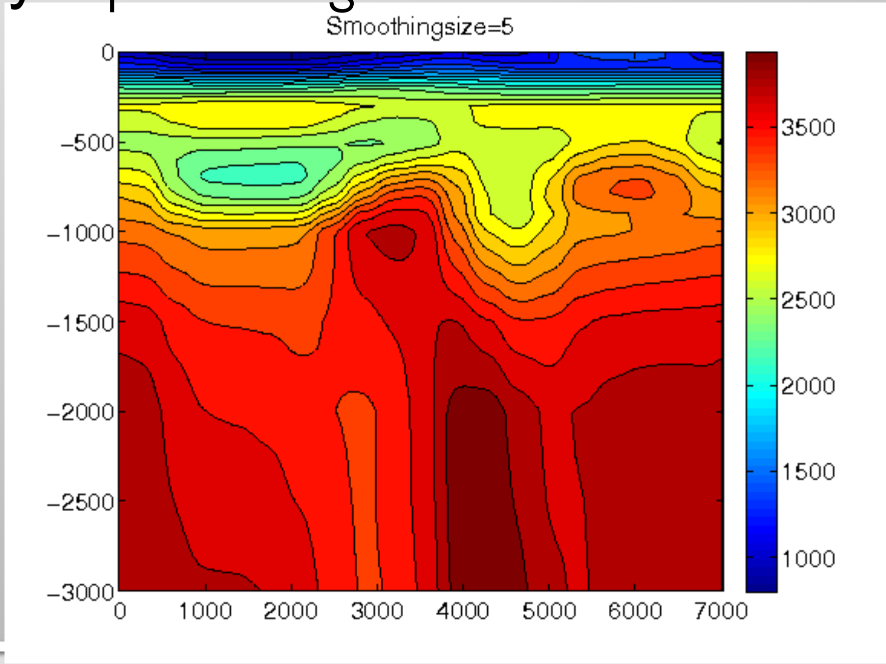
- 4 ways to specify topography
  - Gaussian hill (testing)
  - Grid file (regular lat-lon lattice)
  - rfile raster file
  - Etree database (SF bay area)
- z=0 is mean sea level
- User must pick bottom z-coordinate for curvilinear grid: zmax
- Rule of thumb: example:
  - 1500 < elevation < 2500 m (positive above sea level)
  - zmax >= - (1500 – 2*(2500-1500)) = +500 (positive below sea level)
- topography input=geographic file=AltaRockSim.topo zmax=700 order=3



The material model (ρ, C_P, C_S) must be defined at all points in the grid

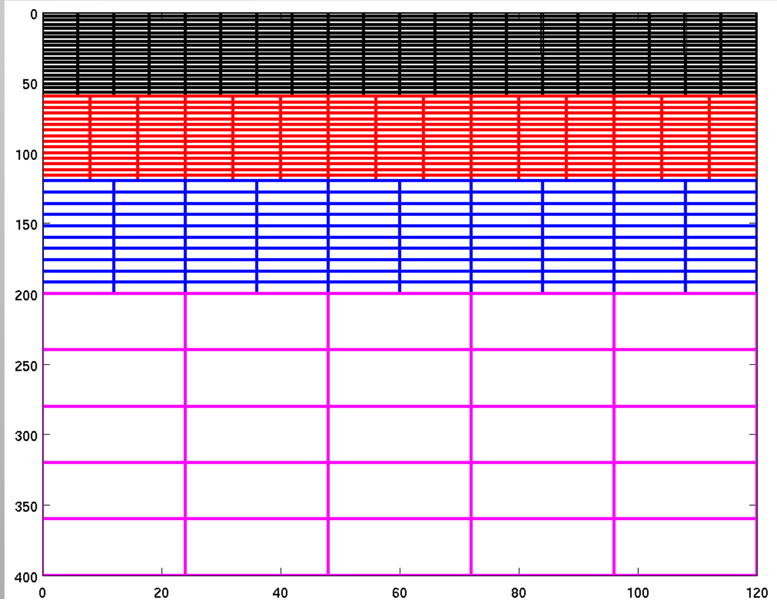
- One or several material commands can be combined
  - Block, efile, pfile, ifile, rfile
- The order matters (later commands take precedence)
- Visco-elastic modeling triggered by the **attenuation** command: Also need to specify Q_P and Q_S

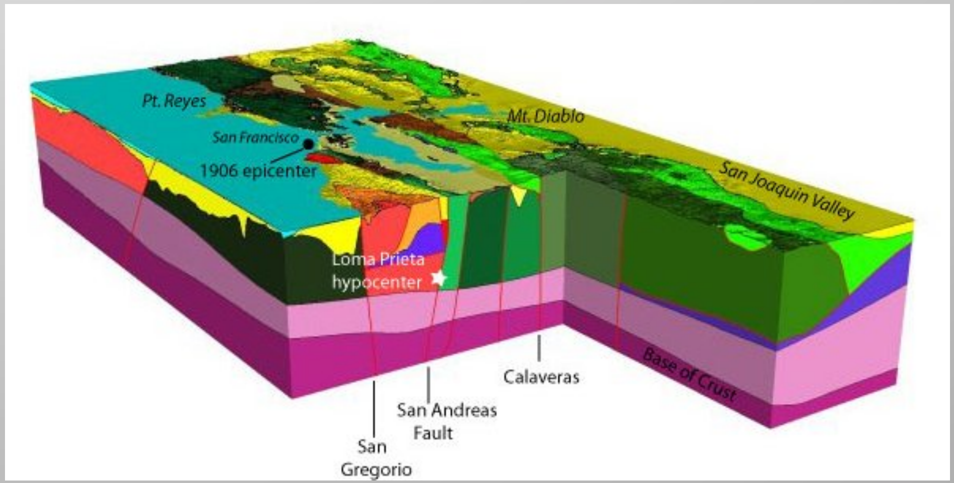




We recommend the binary rfile format for heterogeneous models with topography

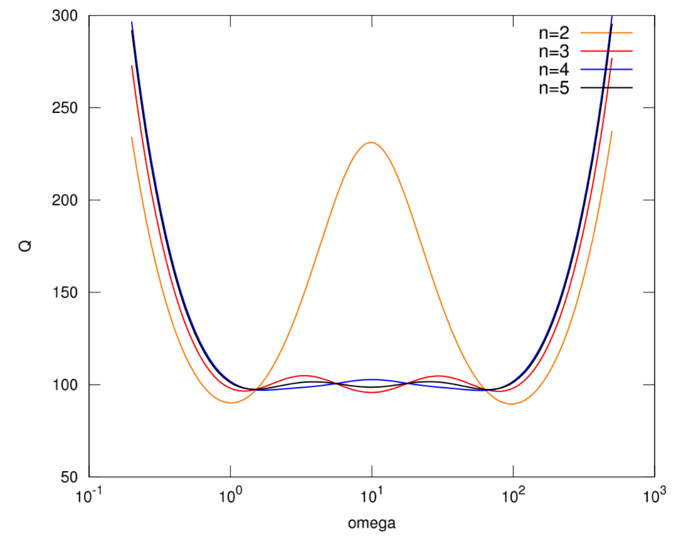
- Header + several rectangular blocks of data
- Byte-swapping for mixing Intel/IBM byte ordering
- Parallel I/O allows fast access
- Each core only reads what it needs

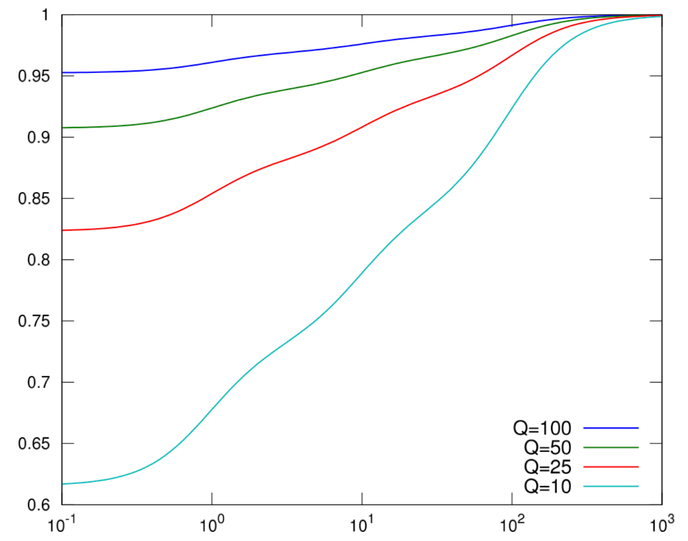




Standard linear solid (SLS) mechanisms are used to model anelastic attenuation

- The quality factors (Q_P and Q_S) are approximately constant in frequency band [0.01 f_{max}, f_{max}]
  - User must specify upper frequency: f_{max}
  - More mechanisms: less variation in Q, but more calculations. Default `nmech=3`
- The visco-elastic material is dispersive: phase velocity depends on frequency
  - User must specify `phasefreq` parameter: frequency where C_P and C_S are accurate.





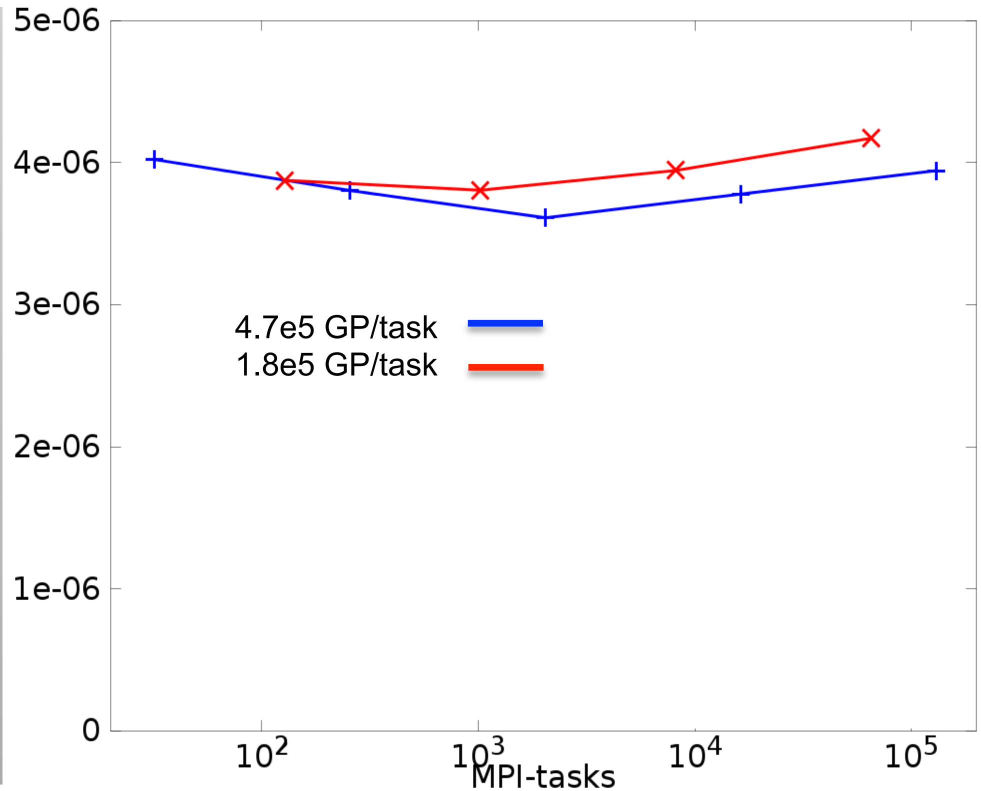
Installing and running sw4

- sw4 can be built on Linux/OSX/Unix machines
  - cmake or make
  - See user’s guide appendix A for instructions
- Command file specifies the simulation
  - `srun -ppdebug -n128 sw4 my-input.in`
- Syntax for running MPI-jobs varies:
  - Macs with openmpi: `openmpirun -np 4 sw4 input.in`
  - Linux box with mpich2: `mpiexec -np 8 sw4 input.in`
  - Livermore computing (LC): `srun -n4096 sw4 my-big-job.in`

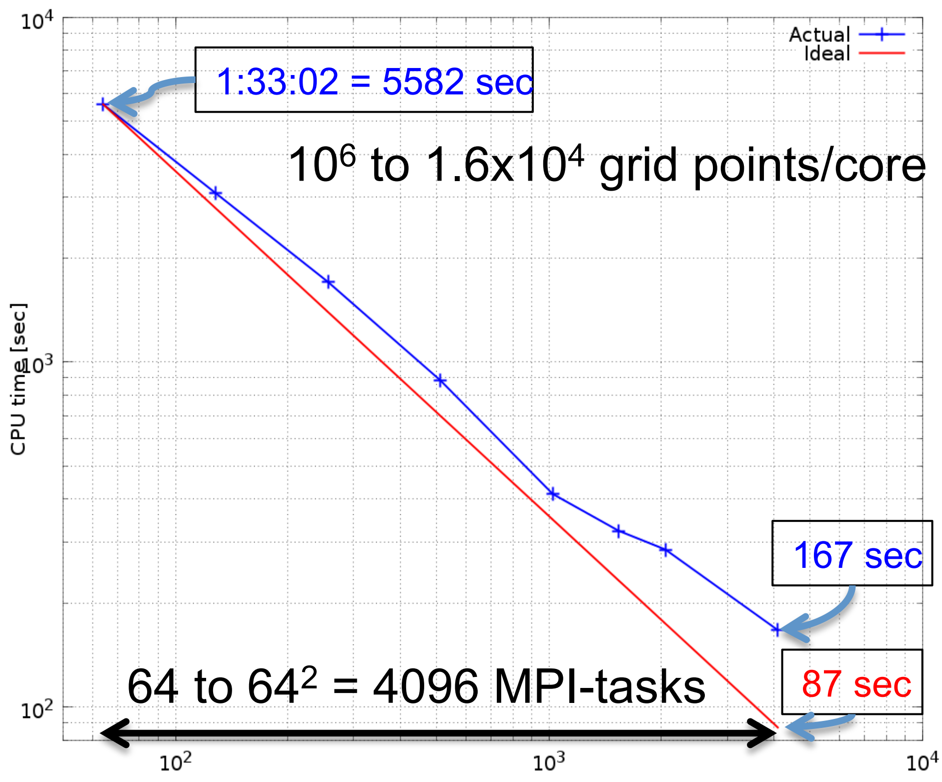
How many cores do I need for running SW4 on a parallel machine?

- The calculation needs to fit in memory (see Appendix C)
- We often use 1e5 to 1e6 grid points / core

Weak scaling: 32 to 131,072 cores
Fixed # grid points per core



Strong scaling: 64 to 4096 cores
Fixed problem on more cores



Most SW4 output files can be read with matlab / octave scripts in …/sw4/tools

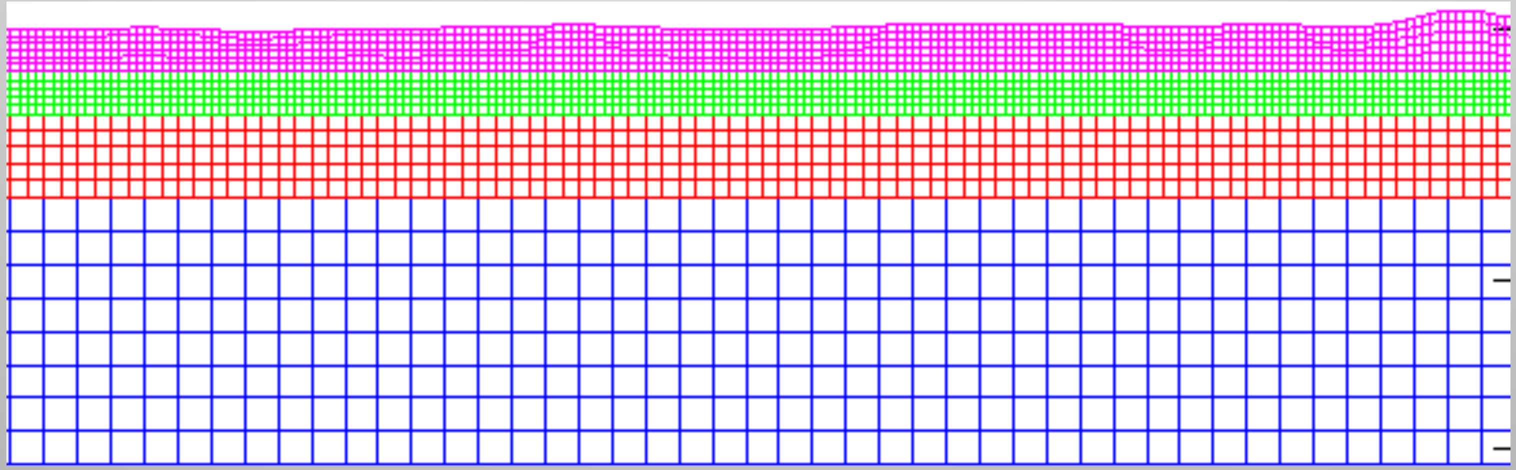
- Synthetic seismograms (sac or usgs txt files): sac or matlab / octave
- 2-D cross-sections (image files): octave / matlab
- Overview on a map (gmt script): gmt programs

General guidelines and scaling

- Check setup on a coarse grid
- Check a few images of the material model
- Check receivers on a map: gmt
- Extrapolate computational resources from coarse run
- Doubling frequency = Halving the grid size = doubling # grid points / dimension
- Doubling # grid points/dimension: 8x grid points, 2x time steps
  - On same number of cores: 8x memory, 16x CPU time
  - On 8x more cores: same memory, 2x CPU time (weak scaling)
  - On 16x cores: ½ x memory, about the same CPU time
- Several examples in .../sw4/examples
- Many matlab/octave scripts in .../sw4/tools
- SW4 documentation in .../sw4/doc and from CIG

Summary and future directions

- SW4 is a documented open source code available from CIG
  - Version 1.1 as tar-ball (Oct-2014)
  - Bleeding edge on github
- Anisotropic materials and curvilinear meshes described in
  - N.A Petersson and B. Sjogreen, J. Comput. Phys. v. 299 pp. 820-841 (2015)
- Local mesh refinement will soon be available



- Currently looking at options for heterogeneous GPU systems

Lawrence Livermore
National Laboratory