Simulating seismic wave propagation with sw4

CIG tutorial

N. Anders Petersson

June, 2016

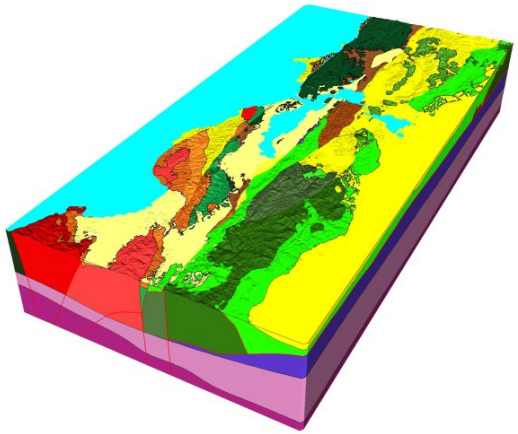
LLNL-PRES-695958
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

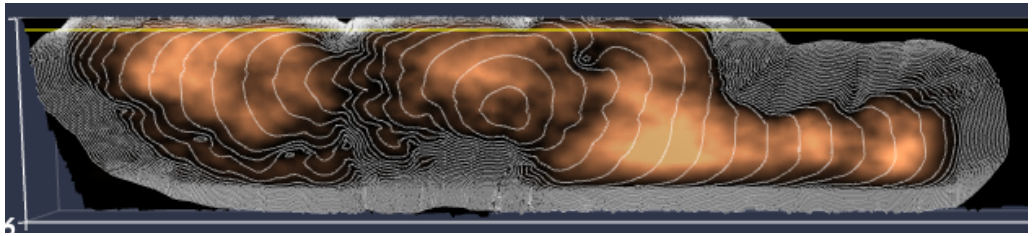
SW4 simulates regional earthquake ground motions on parallel computers

- Mathematical model of the Earth:
  - Navier's eqn of 3-D linear isotropic solid dynamics
  - 4th order summation-by-parts finite differences
  - 4th order time explicit stepping

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \mathcal{T} + \mathbf{f}$$



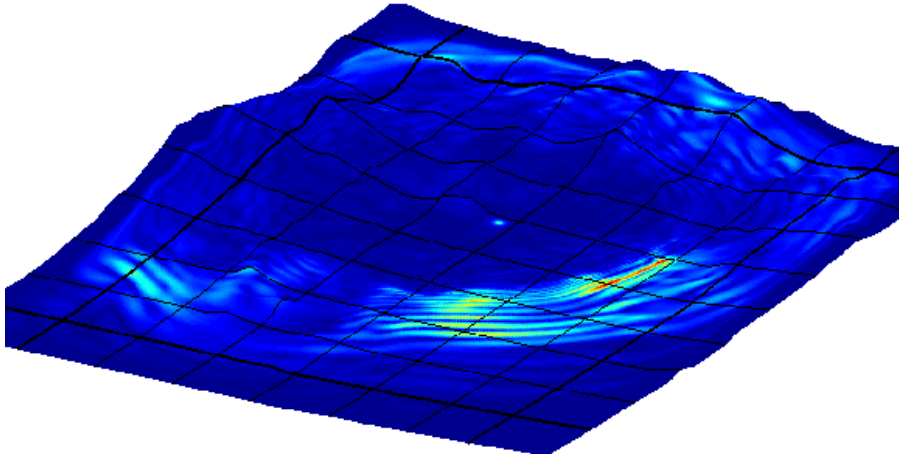
- Geophysical model
  - Isotropic heterogeneous elastic & visco-elastic material
  - Topography
- Kinematic source rupture model



- MPI-based distributed memory programming model
  - C++ with Fortran numerical kernels
- SW4 is open source, GNU GPL-v2

The SW4 and WPP codes solve the time dependent 3D seismic wave equation

- WPP is 2nd order accurate and SW4 is 4th order
- Summation by parts (SBP) finite difference methods
  - Fundamentally different from FDTD
  - Energy stability
  - Higher order accuracy
- Curvilinear coordinates
  - Realistic topography
  - Absorbing far-field layers
- Visco-elastic modeling
- Point forces and moment tensor sources
  - Grid independent location
- Anisotropic materials
- Local mesh refinement



Summation by parts discretization mimic integration by parts for finite differences

- Elastic wave equation:

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

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

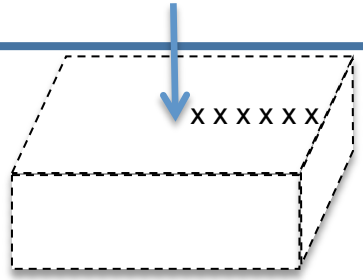
- Discretized in space:

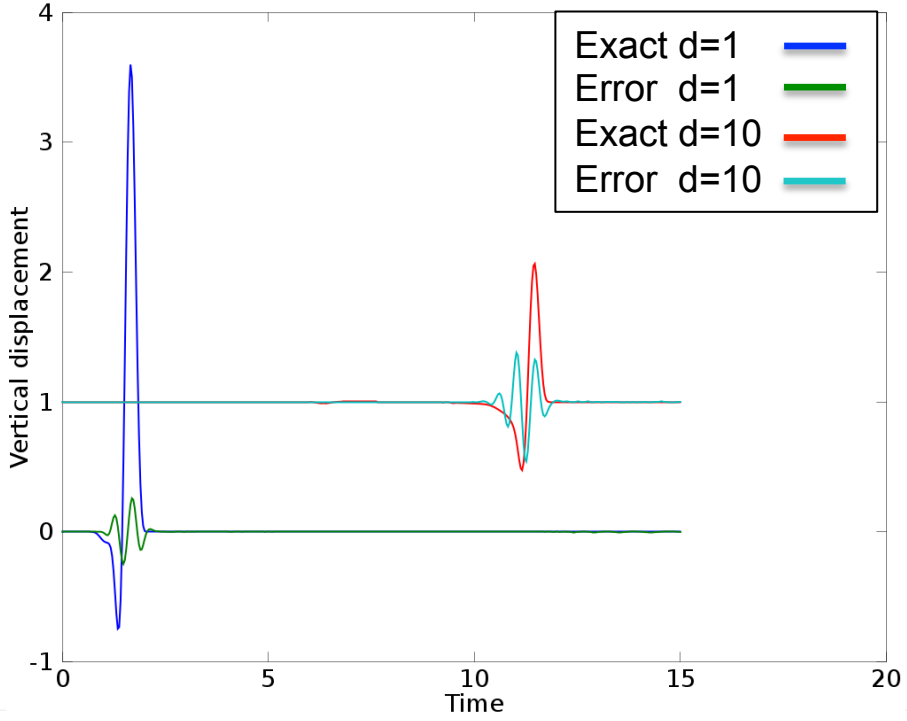
$$\rho \frac{d^2 \mathbf{u}}{dt^2} = \mathbf{L}_h \mathbf{u} + \mathbf{f}(\mathbf{x}, t)$$

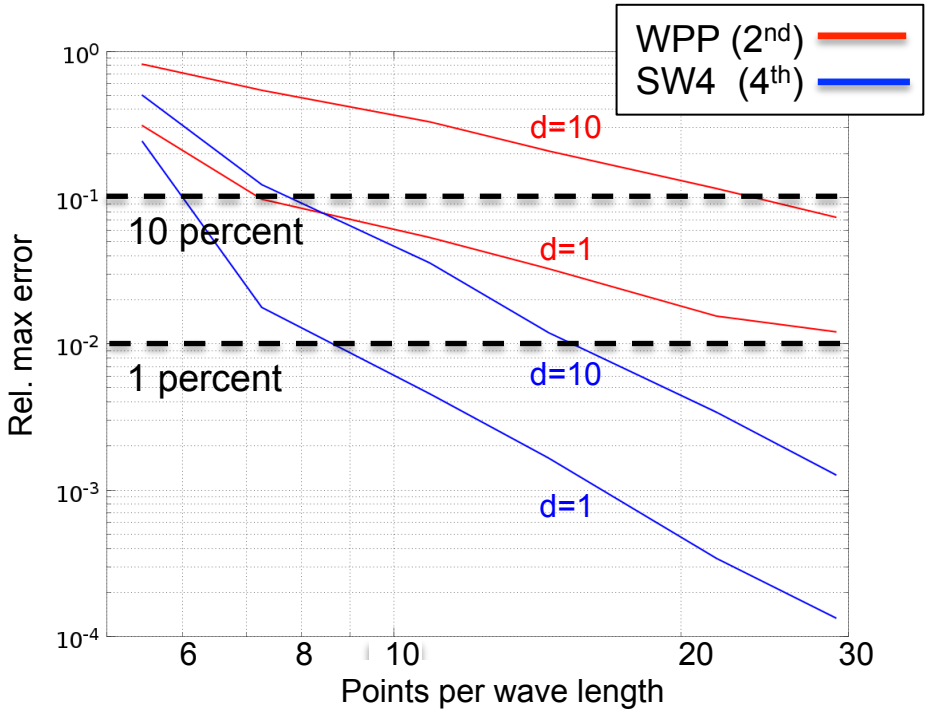
- Symmetry and negative definite properties of '**L**' preserved by summation by parts discretization '$\mathbf{L}_h$'
- Numerical scheme is energy stable

A 4th order method is more efficient since it needs fewer grid points per wavelength

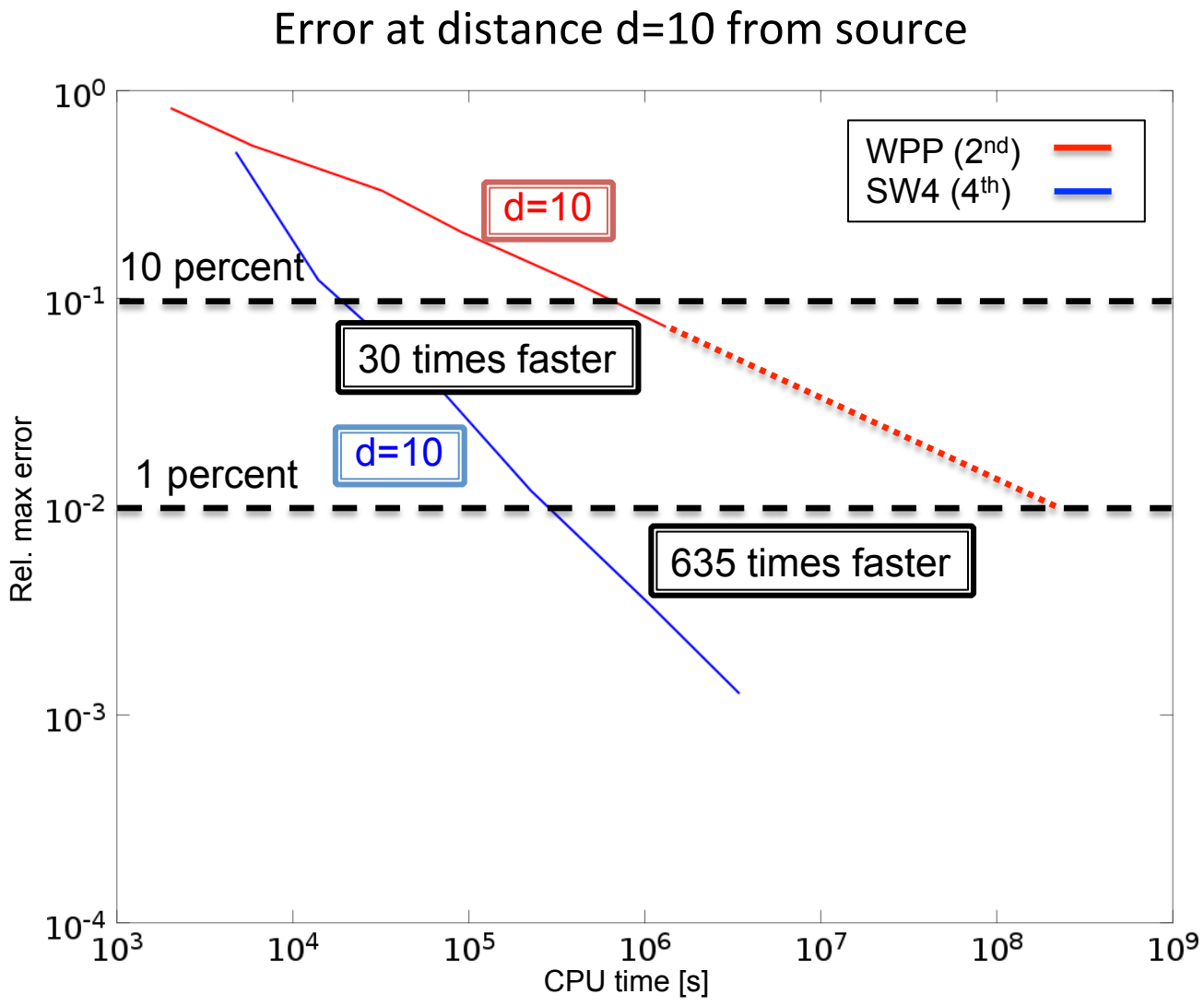
- Lamb's problem
- Solution at distance d=1 and d=10 [Mooney (1974)]
- Error: P^{-2} versus P^{-4} (P = grid points per wave length)
- Improved efficiency with 4th order accuracy





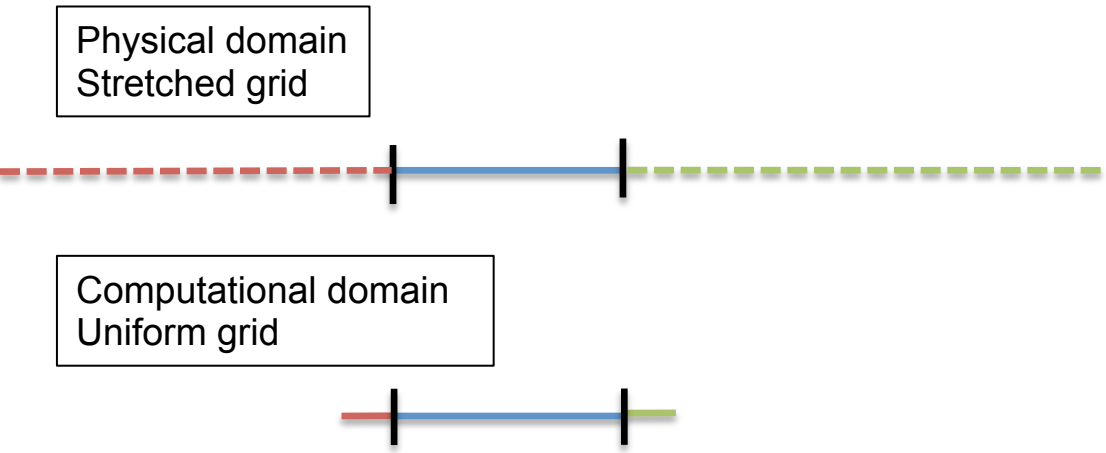


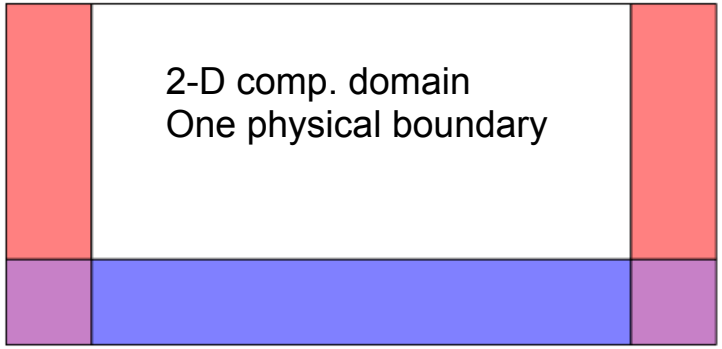
The 4th order method is much faster at larger distances and gives smaller errors



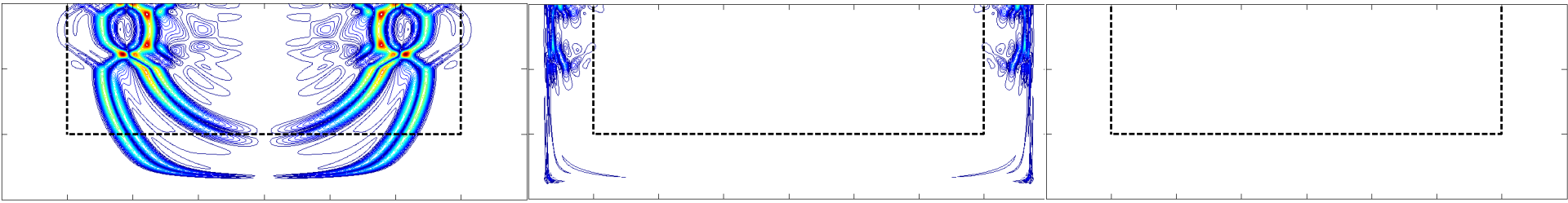
SW4 uses super-grid layers to suppress artificial reflections from far-field boundaries

- Perfectly matched layers (PML) can be unstable
- Super-grid shrinks very large physical domain
  - Slow down and compress waves in layers
  - Damp unresolved modes with high order dissipation
- Always stable
  - Accurate if layer is sufficiently wide

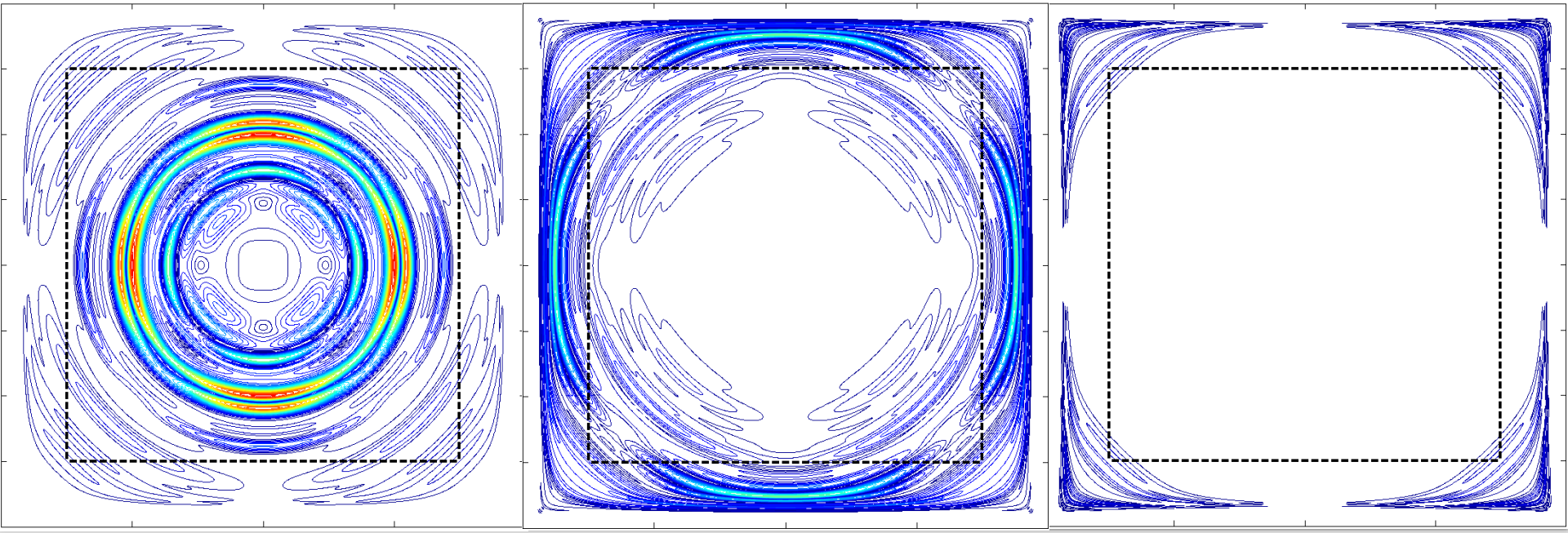




Super-grid test: 3-D elastic wave equation with layered material and a free surface

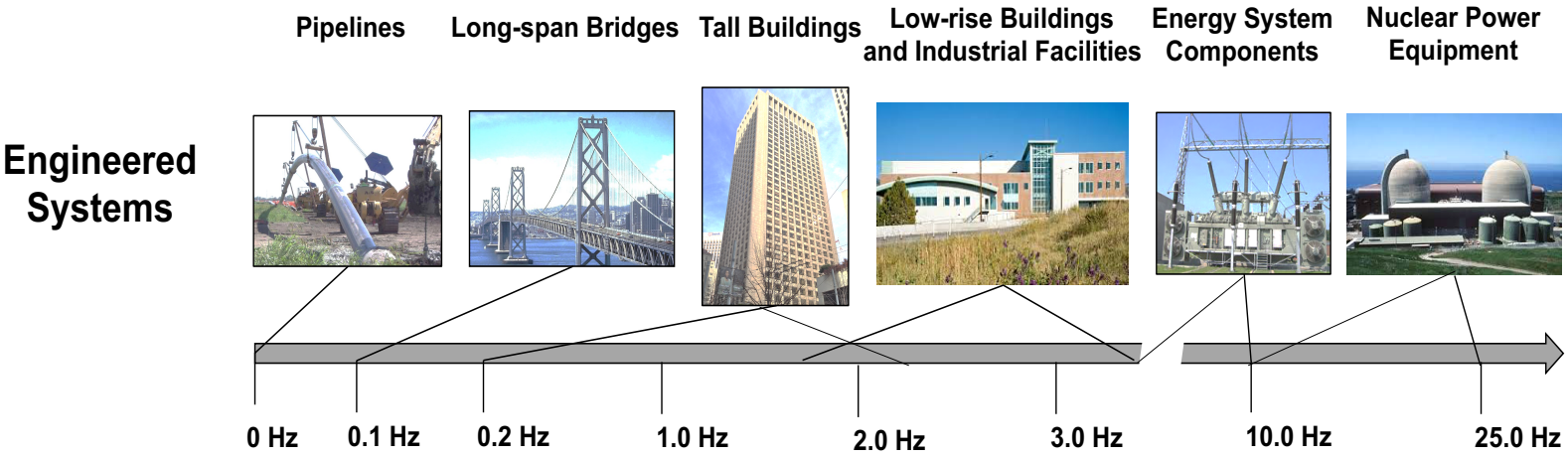


t=5 t=10 t=15



The computational requirements scale as (frequency)^4

- Smallest wave length $\lambda \approx V_s/f_{max}$
- Grid size: $h = \lambda/P$; Points per wavelength: P
- Double frequency -> 8 x grid points, 16 x CPU



Rel. Complexity:	0.0001	0.0016	1	16	81	10,000	390,625
Rel. Memory:	0.001	0.008	1	8	27	1,000	15,625

Hands-on sw4 tutorial

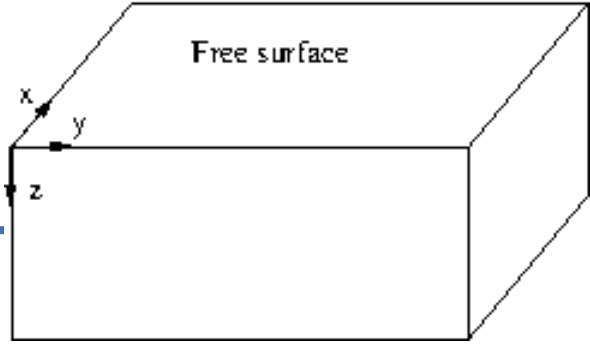
- How to run sw4
  - Smaller & larger jobs
- How to set up the simulation
  - Sources
  - How do I pick the grid size?
  - Topography
  - Material model, visco-elastic modeling
  - Output options
- Practical suggestions
  - Start coarse, check model, estimate resources, refine
  - Use the parallel file system
  - Initial analysis before moving large files
- Lots of info in the user's guide
  - VM: ~/Documents/SW4-UsersGuide.pdf

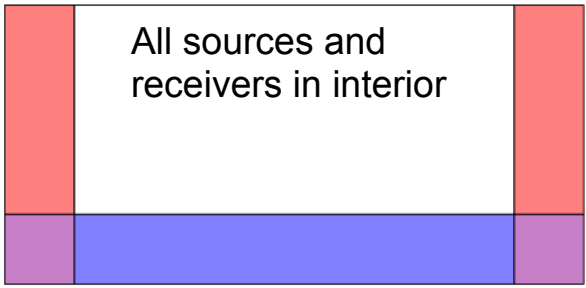
Building and running sw4

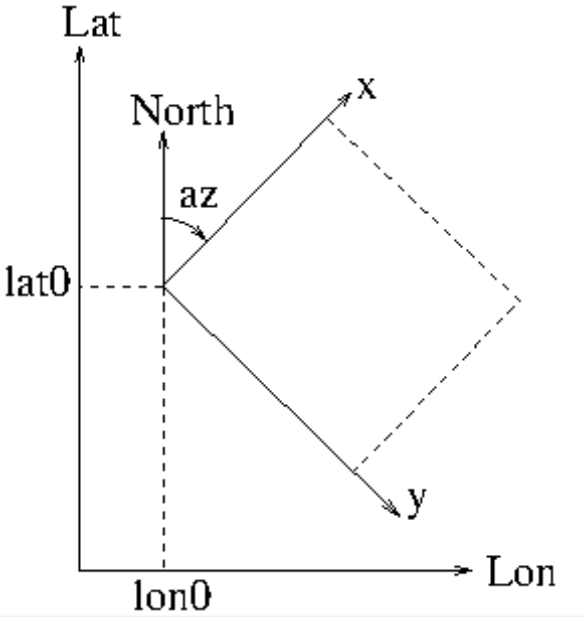
- sw4 can be built on Linux/OSX/Unix machines
  - ~/Documents/SW4-Installation.pdf
- Command file specifies the simulation
  - `mpirun -np 2 sw4 my-input.in`
- Syntax for running MPI-jobs varies:
  - Macs with openmpi: `openmpirun -np 4 sw4 input.in`
  - VM-ware mpirun: `mpirun -np 8 sw4 input.in`
  - Livermore computing (LC): `srun -n4096 sw4 my-big-job.in`
- Running on supercomputers (details may vary)
  - Smaller, shorter jobs on interactive debug partition. Start job from command prompt. No CPU-bank needed
  - Larger jobs use a batch system. Job described by a script. Usually needs a CPU-bank
  - For faster I/O, use a parallel file system if available

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







1st exercise: solve Lamb's problem

- Start VM virtualBox & import the CIG16-TutorialVM appliance
- Log on as sw4_user (No password)
- cd Desktop
- Simulation described in input file: seismic1.in

```
fileio path=seis-out
grid nx=151 x=10e3 y=10e3 z=5e3
time t=5.0
# supergrid thickness for h = 10000/150 = 1000/15, 30*h = 2000, 60*h = 4000
supergrid gp=60
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 on the surface
rec x=5e3 y=6e3 z=0 file=v1s sacformat=0 usgsformat=1
```

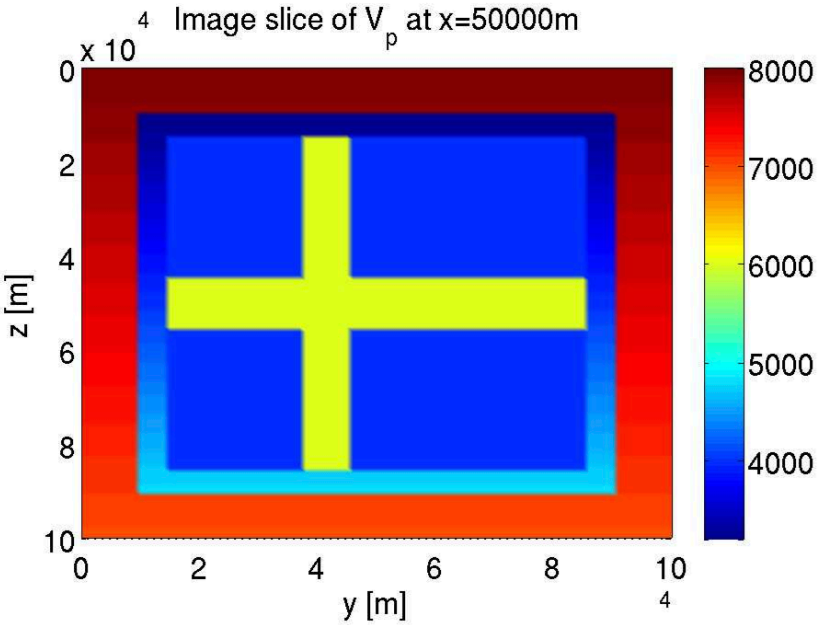
- mpirun –np 1 sw4 seismic1.in
- (1 min, 51 sec runtime on my Mac laptop)

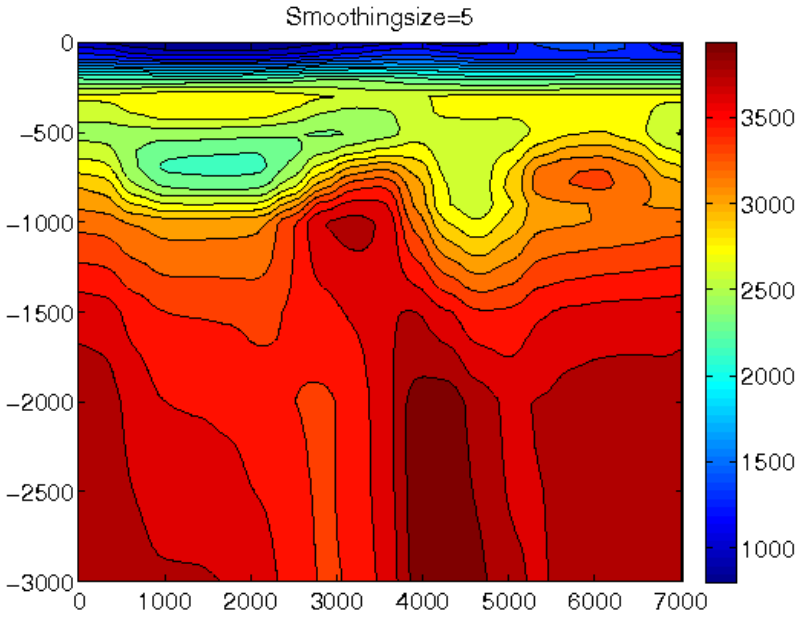
We use octave to view the results

- octave is open source software (similar to Matlab)
- Open octave from terminal window (in VM):
  - >octave
- Expand octave search path (put command in ~/.octaverc file)
- octave:1> addpath("/home/sw4_user/Desktop/tools")
- octave:2>cd Desktop/seis-out
- octave:3>ls
- octave:4>plotusgs("v1s.txt")

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

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



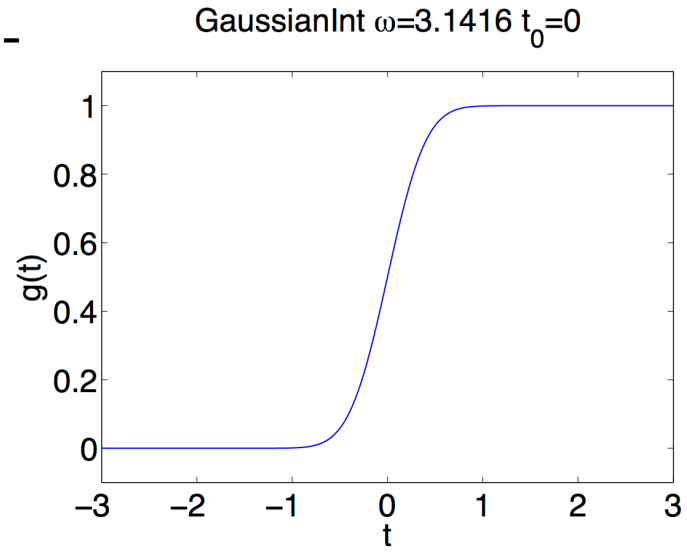


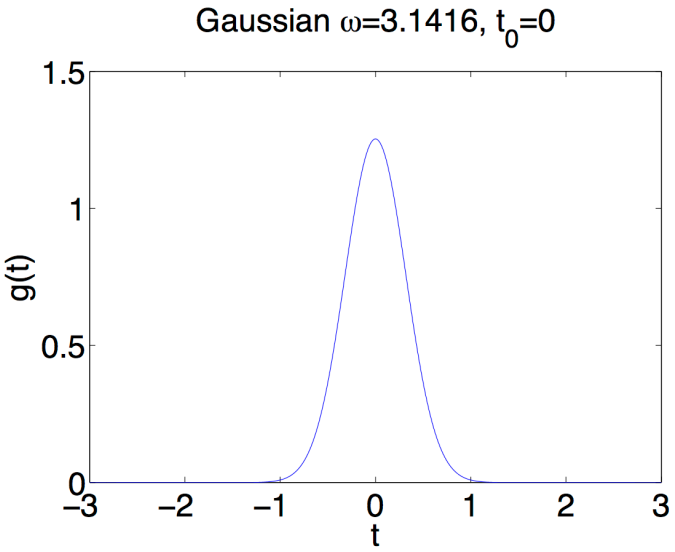
sw4 provides several options for specifying sources and time-functions

- Any number of point forces or moment tensor sources
  - Forces defined by 3-component vector
  - Moment tensor (symmetric) defined by 6 components M_{ij}, or
  - Seismic moment (M_0) and strike, dip, rake angles
- Many pre-defined time functions (Gaussian, Ricker, Liu, …)
  - Frequency parameter and start/center time
- Dirac delta-distribution: trigger 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 an SRF file

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)
- 3 approaches:
  - Tune freq in source time function to not trigger unresolved waves
  - Calculate $f_{max} = \min C_S/(P_{min}\ h)$, filter out unresolved frequencies afterwards (sac)
  - Use `prefilter` command to pre-process all sources (not discussed today)
- Relation between f_{max} and frequency parameter (`freq`) is different for each time function (see UG 4.4)
- octave scripts for plotting time function and Fourier transform: `fcnplot`, `ftfcnplot`

2nd exercise: solve a layer over half-space problem. Input file: loh1-slow.in

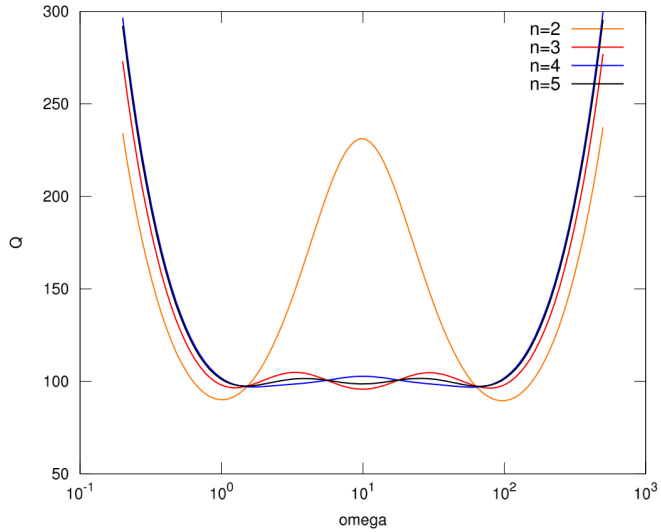
- The order of the block commands is important
  block vp=6000 vs=3464 rho=2700
  block vp=4000 vs=2000 rho=2600 z2=1000
- Gaussian time function gives the velocities directly
  source x=15000 y=15000 z=2000 mxy=1e18 t0=1.2 freq=5.0 type=Gaussian
- Time histories recorded @ the closest grid point
  rec x=15600 y=15800 z=0 file=sta01 usgsformat=1
  rec x=21000 y=23000 z=0 file=sta10 usgsformat=1
- Material images don't change, recorded initially
  image mode=s y=15e3 file=ver cycle=0
- Solution images can be recorded at fixed time intervals
  image mode=mag z=0 file=surf timeInterval=0.5

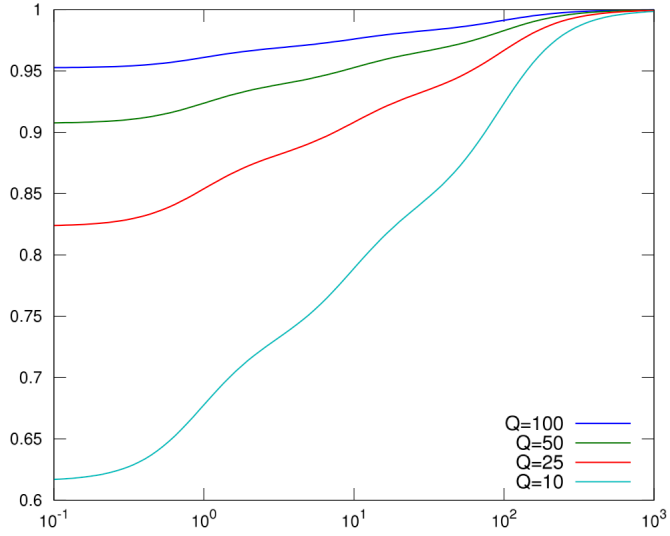
We can use octave to view image files

- octave>cd Desktop/loh1-slow
- octave>ls
- octave>help plotimage
- octave>plotimage("surf.cycle=170.z=0.mag.sw4img")
- octave>help imageinfo
- octave>imageinfo("surf.cycle=170.z=0.mag.sw4img",10,1)

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.





3rd exercise: solve an anelastic layer over half-space problem. Input file: loh3-slow.in

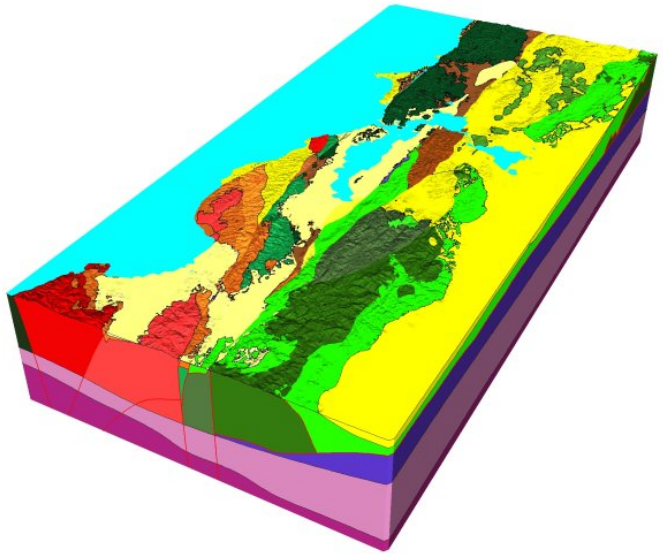
- Only differences from loh1-slow.in

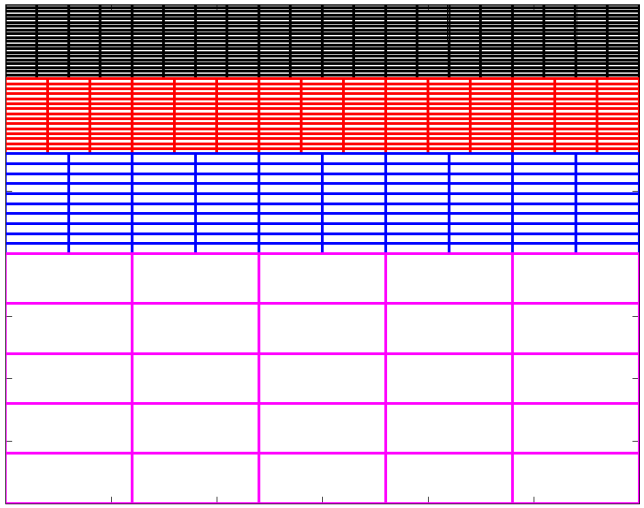
```
# different output directory
fileio path=loh3-slow
# must specify Qp and Qs
block vp=6000 vs=3464 rho=2700 Qs=69.3 Qp=155.9
block vp=4000 vs=2000 rho=2600 z2=1000 Qs=40 Qp=120
# enable attenuation modeling
attenuation phasefreq=2.5 nmech=3 maxfreq=15
```

- About 3x CPU time and 2.7x memory

Large complex material models can be described in an rfile model

- Variable resolution data structure
- 7.6 Gbyte file for regional SF bay area
- Vulcan (5 PFlops): 1 Gbyte memory per core
- Parallel I/O routines on parallel file systems





2897 x 1401 x 74 (6 Gb)

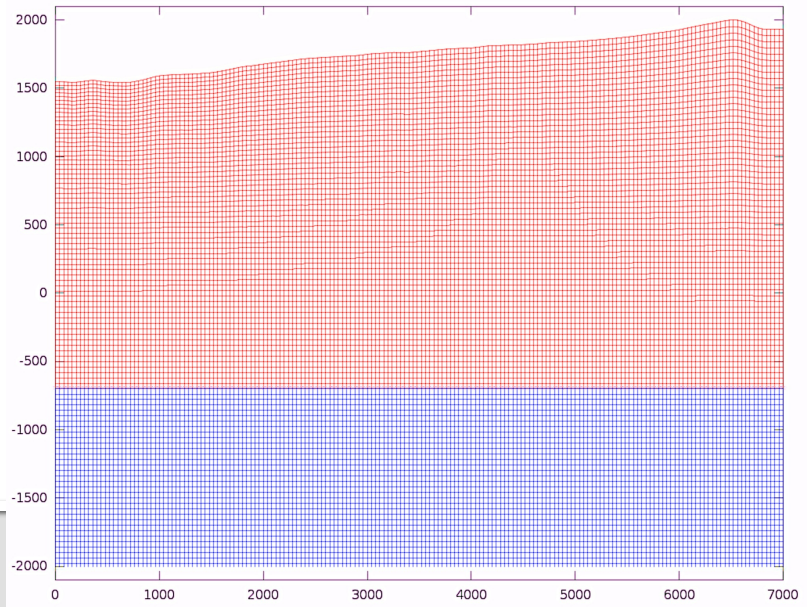
1449 x 701 x 57 (1.2 Gb)

725 x 351 x 33 (0.17 Gb)

363 x 176 x 194 (0.25 Gb)

Realistic topography is included in an rfile

- SW4 constructs a curvilinear grid on startup
- z=0 is mean sea level
- User must pick bottom z-coordinate for curvilinear grid: zmax
- Rule of thumb: zmax = twice the variation below the lowest elevation
  - 1500 < elevation < 2500 m (positive above sea level)
  - zmax >= - (1500 – 2*(2500-1500)) = +500 (positive below sea level)



4th exercise: Simulate earthquake near Berkeley. Input file: berkeley-small.in

- Proj.4 commands in grid command (az must agree with rfile header)

  grid x=12e3 y=12e3 z=5e3 nx=151 lat=37.93 lon=-122.25 az=143.638

  proj=tmerc datum=NAD83 lon_p=-123.0 lat_p=35.0 scale=0.9996

- Topography command

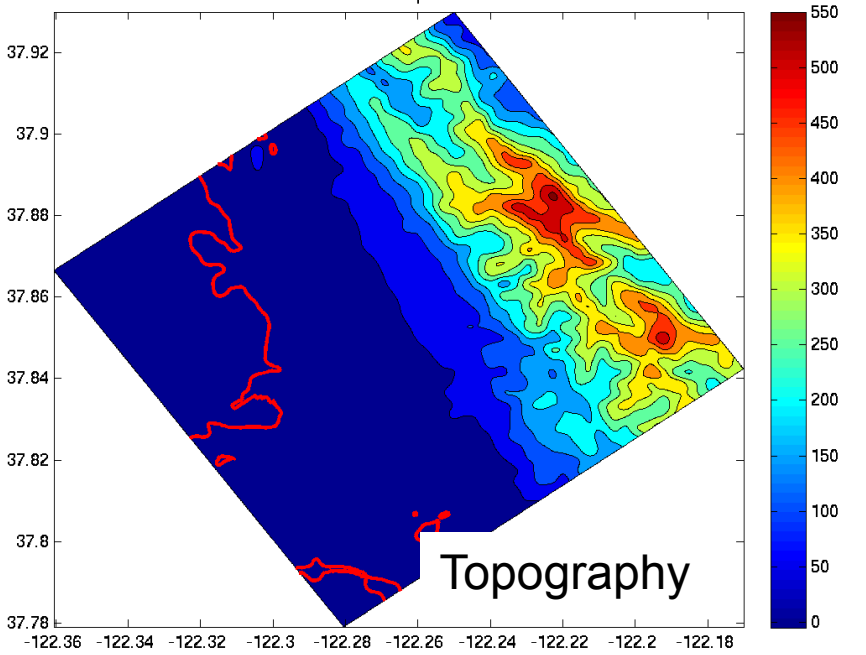
  topography input=rfile zmax=2e3 order=3 file=berkeley-r.rfile

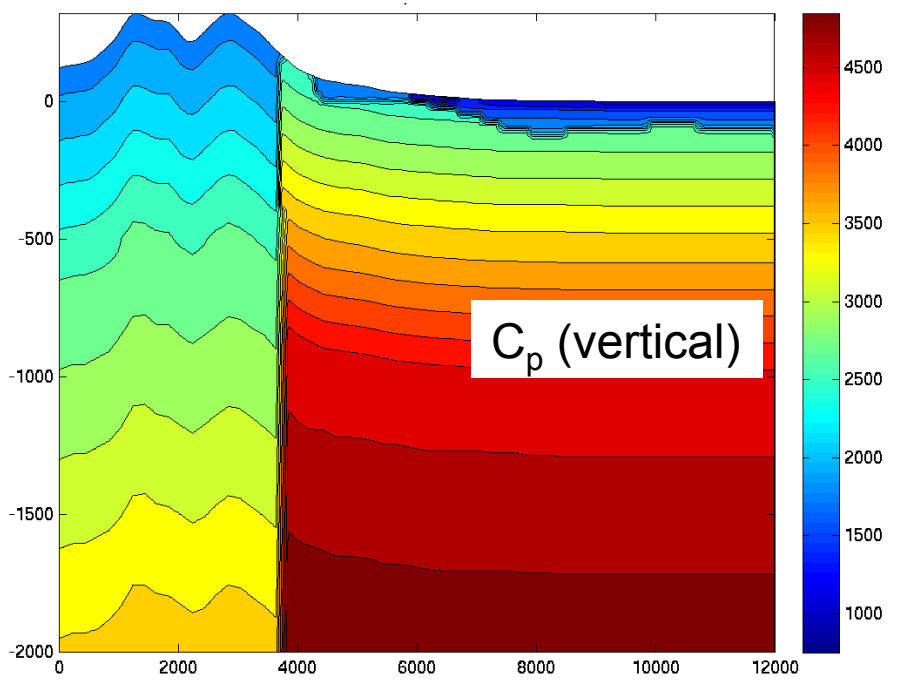
- Material model

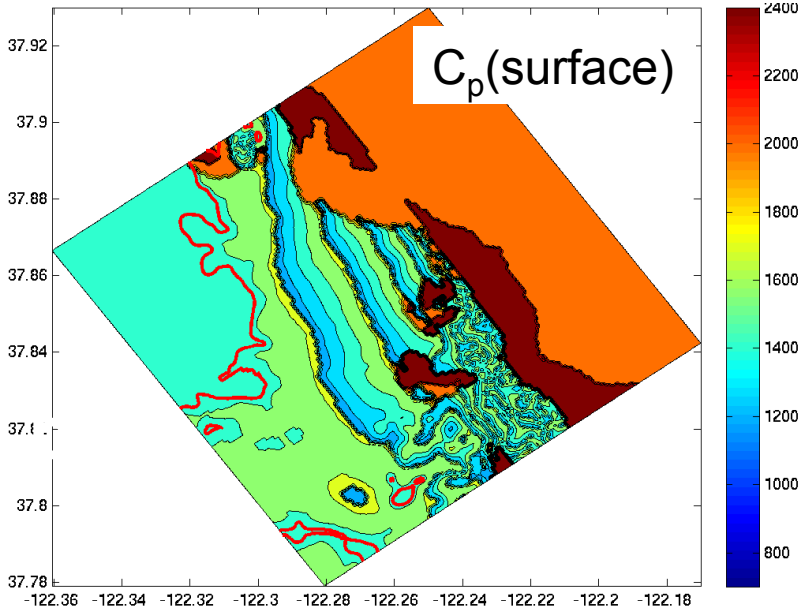
  rfile filename=berkeley-r.rfile

- Runtime 9 min, 50 sec

The case berkeley-small.in uses a small rfile model

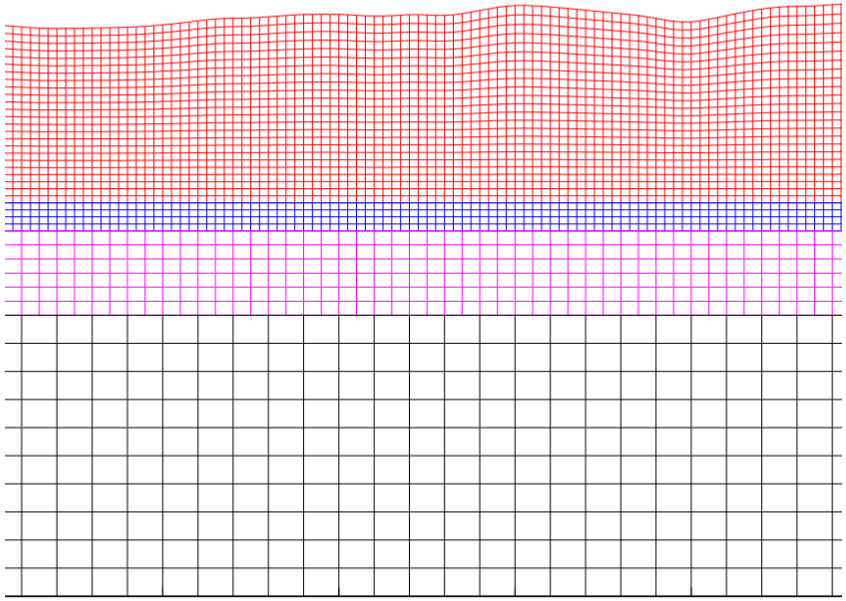






Mesh refinement can be used to save computational resources

- Keep number of grid points per wave length ~ constant
- Saves memory
- Allows larger time step
- Currently only elastic materials



5th exercise: Try mesh refinement. Input file meshref.in

- The grid command specifies the coarsest grid size

  grid h=350 x=30000 y=30000 z=17000

- User specifies z-level with the refinement command

  refinement zmax=2.5e3

- Each grid patch must have at least 12 grid points in the vertical direction.

- Runtime ~5 min

- Same problem w/o mesh refinement in nomeshref.in

General guidelines

- 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
- SW4 source distribution includes examples and octave scripts

What if it doesn't work?

- Check command syntax and rules in SW4 user's guide
- Update SW4 source code from github
- Ask questions on CIG-SEISMO mailing list

  cig-seismo@geodynamics.org
- Discuss problems in the github issue tracker

  https://github.com/geodynamics/sw4/issues
- Help provided as time permits


Lawrence Livermore
National Laboratory