Paracousti

2D/3D modeling of underwater acoustics

Tutorial 2: Pre-Processing for a 2D Model

Sandia National Laboratories

Tutorial Objectives and Assumptions

- **Objectives**
  - Introduce users to Paracousti
  - Provide users a step-by-step guide for any necessary pre-processing to produce input files when provided a collected data set for a natural bathymetry
    - Note that this Tutorial will not cover how to put together the model using these input files, but will provide an example run script at the end
- **Assumptions**
  - Users have an understanding of acoustics and underwater acoustics
  - Users have a familiarity with and access to MATLAB
    - Users can follow along and perform pre-/post-process in most computer languages, but this tutorial uses MATLAB
    - Python scripts are forthcoming
  - **Users have completed Tutorial 1**

Tutorial Outline

- **Introduction**
- **Definitions**
- **Paracousti Workflow**
  - Pre-Processing
    - Data Collection
    - Data Processing
    - Domain construction
- **2D Example: Newport, OR Coastline**
  - Data Interpolation Options
  - 3D or 2D bathymetry options
- **Best Practices**
- **More Information**

Sandia National Laboratories

Brief Introduction to Paracousti

- **Paracousti**
  - 3D, time-domain, underwater acoustic propagation simulator which solves linearization of Cauchy equations of motion through coupled finite difference solution
    - 4th order spatial, 2nd order temporal
  - Records time-varying pressure and particle velocities
    - Volumetrically: desired timesteps for full 3D space, extremely high storage cost
    - Planar slice(s): desired timesteps, moderate to high storage cost
    - Coordinate(s): instantaneous collection for length of simulation at a singular point or over a grid, low storage cost

$$\frac{\partial \boldsymbol{v}^*}{\partial t} + \frac{1}{\rho^\circ} \nabla p^* = \frac{1}{\rho^\circ}\left[\boldsymbol{F} + \nabla \boldsymbol{m}^{\mathrm{dev}}\right]$$

$$\frac{\partial p^*}{\partial t} + \rho^\circ (c^\circ)^2 \nabla \cdot \boldsymbol{v}^* = \frac{-1}{3}\frac{\partial \boldsymbol{m}^{\mathrm{iso}}}{\partial t}$$

Definitions

- **Acoustic Sound Speed [m/s]**
  - Medium sound speed as a function of space over the entire 3-D model domain
  - Allowed to vary spatially
  - Can be calculated based on environmental conditions
- **Convolution Perfectly Matched Layer - CPML**
  - Boundary condition that absorbs energy on a domain face to prevent reflections back into domain
- **Density [kg/m^3]**
  - Medium mass density as a function of space over the entire 3-D model domain
  - Allowed to vary spatially

Definitions

- **Earth Model**
  - A reference to the model domain and grid spacing defined at the start of every simulation and required for the Paracousti input files
- **NetCDF – Network Common Data Form**
  - An open standard for the binary storage of arrays of scientific data
  - The data storage mechanism for Paracousti input and output files
  - https://www.unidata.ucar.edu/software/netcdf/
- **Receiver**
  - Location and parameters associated with a point in space where trace data is to be recorded

Definitions

- **Sound Pressure Level [dB] – SPL**
  - A normalization of the root mean squared pressure or sound intensity, measured in decibels
  - Specified relative to a reference pressure [Pa]
    - 1 µPa for underwater acoustics

$$SPL = 20log_{10}\left(\frac{P_{rms}}{P_{ref}}\right)$$

- **Source**
  - A time-varying pressure profile referenced to 1 meter from the source location of any amplitude
  - Recommended to be normalized to an amplitude of $\pm$ 1 Pa and scaled by a scalar amplitude during the model run
  - The source profile **is not** used by Paracousti (see Source Time Function)

Sandia National Laboratories

Definitions

- **Source Time Function – STF**
  - The 1st or 2nd integral, with respect to time, of the source pressure profile for a directional or monopole source, respectively
  - <u>This</u> is the input profile used by Paracousti to define the source
- **Slice**
  - A planar output of particle velocity and/or pressure from Paracousti
  - Recorded at desired time(s)
  - Aligned with the Cartesian grid defining the model
- **Trace**
  - A pressure and/or particle velocity output from Paracousti at a single point
  - Continuous in time
  - Defaults to cubic interpolation if between grid points

Definitions

- **Transmission Loss (or Propagation Loss) [dB] – TL**
  - A measure of the reduction in sound intensity or pressure
  - Similar to SPL, but the reference pressure is that of the source as measured 1 m away

$$TL = 20log_{10}\left(\frac{P_{rms}}{P_{source_{ref\ 1m}}}\right)$$

- **Volume Output**
  - Full velocity or pressure data output on the entire simulation 3D grid as a function of time
  - Allows for full view of the evolving wavefield through time
  - Incredibly large files

Paracousti Workflow

- **Tutorial 2 assumes a prior run of Paracousti and a familiarization with the program**
  - This tutorial deals with further analysis of required files for the earth model and how collected data may need to be formatted for Paracousti. All discussion is separate from the Pekeris Example shown in Tutorial 1
- **MATLAB is the presently supported pre-/post-processor**
  - However, many of the functions used in this tutorial exist in or can be quickly converted to Python using the NumPy and matplotlib libraries
- **The files for this tutorial and other examples include:**
  - Bathymetry files for Newport, OR coastline
  - The MATLAB scripts to for the Newport Bay, OR earth model construction and an example model run script
  - These can be found at: https://github.com/SNL-WaterPower/Paracousti

Workflow: Pre-Processing

- **Pre-processing is the step that defines the model domain, the type of source(s), and how you would like to store any output data**
  - Paracousti requires
    - An environmental domain (earth model) as a NetCDF file
    - At least one text file detailing the time history of a noise source
    - Command-line flags indicating additional boundary, source, and output criteria
- **Many datasets required for constructing an earth model does not provide data at the location or fidelity required by Paracousti**
  - MATLAB provides many tools to quickly manage arrays of environmental parameters, interpolate measured data into a model domain, and create the input NetCDF file
  - Any collected data will likely need to be interpolated onto a set grid

Workflow: Pre-processing

- **Paracousti requires a bathymetry, density, and sound speed parameters for an earth model, which can be collected experimentally or approximated**
  - Generally bathymetric data is collected on a grid or via boat where the grid points may vary location to location
    - Model domain grid spacing (dx) is defined by the source parameters and sound speeds
    - Interpolation is required
  - Density, temperature, and salinity values may be collected, but are often assumed for particular environments and sediment layers. Any experimental data is generally collected from a limited number of locations in the area of interest
    - Interpolation over the entire area of interest is required
      - **Values may be interpolated over the entire domain, with depth, or averaged values may be assigned to a particular fluid type (water, sediment layers)**
    - Sound speed may be recorded or calculated from environmental parameters

Sandia National Laboratories

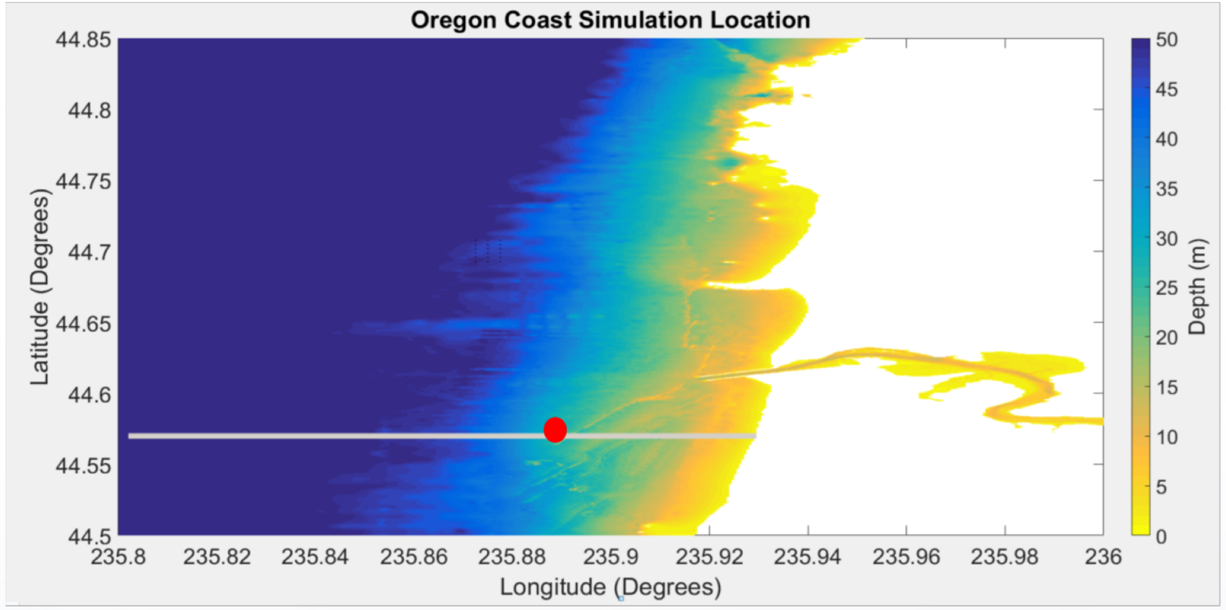
Workflow: Pre-processing

- **Data processing of the bathymetry is required to fully define the density and sound speed associated with each grid point within the domain**
  - A domain size is chose based on a model area of interest with the grid spacing determined by the source frequency and maximum sound speed in the model
  - Simulation types:
    - 3D simulation: The full bathymetry is interpolated and used to define the transition between water and sediment layers for the x, y, and z dimensions of the domain.
    - 2D simulation: A cross section of interest is selected from the 3D bathymetry and interpolated onto the domain grid to define the transition between water and sediment layers in the x and z dimensions of the domain
  - Fluid properties are assigned based on grid cell location post interpolation
- **Domain construction may change based on bathymetric interpolation, repeating the process**

2D Example: Newport, OR

- **The problem**
  - Newport OR coastline bathymetry
    - Data points approximately every 40 m
    - 2D X-section (grey), source location (red)
  - Source: continuous, sinusoidal function
    - 100 Hz @ 1 Pa
  - Constant densities and sound speeds
  - Set up the earth model
- **Initial domain setup**
  - Required step size: dx <= 1.5; dt <= 0.0004 sec

Note: the domain size is similar to that of Tutorial 1



Water: $\rho = 1000\ kg/m^3$, $c = 1500\ m/s$
Sediment: $\rho = 1800\ kg/m^3$, $c = 1800\ m/s$

```
>> x = 0:1.5:7500
>> y = -30:1.5:30
>> z = -6:1.5:90
>> t = 0:0.0002:8;
```

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Determining the bathymetry cross-section**
  - Import Bathymetry and re-name the bathymetry data variable for convenience

```
>> load bathy_contours_NETS
>> bathy = bathy_interp_cgrid;
```

  - Define a single array of row, column designations for any data point in the bathymetry for any data point with a depth < 0
    - Set all values in bathy at speicified locations equal to NaN

```
>> il = find(bathy<0);
>> bathy(il) = NaN;
```

The water surface starts at *z=0* and gets larger in depth, so anything less than 0 m is above the water surface

  - Select the cross section of interest: Longitude (235.87˚); row 141 in the data set
    - Call the cross section slicexz as it is in the xz dimensions of the model domain

```
>> slicexz = bathy(141,:);
```

Note: since we are only interested in a 2D cross section of the bathymetric data, or slice, we can select all columns of data associated with a particular row or longitude. The slice will then extend all the way across the given bathymetry and include any associated latitudes

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Truncating the bathymetry**
  - Remove all data (previously defined as NaN) were the depth is <0 in slicexz

```
>> NaNi = find(isnan(slicexz));
>> slicexz(NaNi) = [];
```

    - **Note that the same process may be used to select a slice at one latitude with all data selected for a singular column, not row**
  - Where slicexz is an array of depth values associated with longitudinal positions along the cross section selected
  - Longitudinal positions can be calculated based on the spacing between each bathymetric point originally collected

At this point, there is a matrix with x locations for each depth along the slice. These x locations must be calculated and converted to proper units (m)

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Converting bathymetry units**
  - Create an array of x locations (range) for each slicexz data point in meters
    - Define the conversion metric; ***n*** meters per degree for Lat = 44.57˚

```
>> n = 79434.3163923549
```

    - Create an array of the longitudinal values scaled by the first value; ***lonC*** is a stored variable which contains an array of all longitudinal values for the bathymetry grid

```
>> R = (lonC - lonC(1,1))
```

    - Define the step size, ***i***, in the new array, ***R***

```
>> i = R(2,1)-R(1,1)
```

    - Convert the array of scaled longitudes to meters and rename it ***Range***

```
>> Range = n.*R;
```

    - Remove all values of cells which are at the same array designation as slicexz

```
>> Range(NaNi) = [];
```

As the range values are set to the x-dimension locations for the water depths recorded in slicexz, any information that was removed in one must be removed in the other. MATLAB requires that the two arrays be the same length

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Finalizing the bathymetry definition**
  - Add a zero value on the minimum side of slicexz to define the shoreline

```
>> slicexz = [slicexz 0]
```

  - Flip the direction of the array slicexz so that it matches the direction of the created range; increasing in values right to left

```
>> slicexz = fliplr(slicexz)
```

  - Calculate a last point on the end of Range so the length of the Range array matches slicexz

```
>> Range = [Range; i*n+Range(find(Range,1,'last'),1)
```

Note that this calculation just adds one last point based on the step, i. As the distances are scaled and only referenced with respect to slicexz, it is fine that we tack on one additional point.

  - Save the range and slicexz as a .mat file in MATLAB to call in the future as variables

```
>> save('Newport_slice.mat','slicexz','Range');
```

    - **Any number of variables may be saved in a .mat file, limited only by memory space**

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Interpolating the bathymetry**
  - Now we may interpolate the values of the created range and slicexz arrays onto the domain boundaries already defined
  - Use the MATLAB function `>> interpl(x,v,xq,’method’)`
    - Where x is the data set to be interpolated, v is the corresponding values, xq is the data set you are interpolating on to, and ‘method’ defines the method of interpolation
      - **Methods: (‘linear’), nearest neighbor (‘nearest’), cubic (‘pchip’)**

```
>> sliceint = interp1(Range,slicexz,x,'pchip');
```

  - Either ‘linear’ or ‘pchip’ are recommended for the interpolation scheme
    - Different interpolation methods may affect your model result as is affects the uncertainty in the problem

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Environmental properties**
  - Define a single array for both the density (`rho`) and sound speed (`vp`) of the domains
    - For a constant water layer on top of sediment, define arrays of values based on the grid location where the location of the sediment layer changes based on water depth (sliceint)

```
>> for i=1:length(x)
     for j=1:length(z)
         zu(i)=sliceint(1,i);
         if z(j)>zu(i)
                vp(i,:,j)=1800; rho(i,:,j)=1800;
         else
                vp(i,:,j)=1500; rho(i,:,j)=1000;
         end
     end
end
```

The water surface starts at *z=0* and gets larger in depth, so anything less than the value of ***sliceint*** at that location is within the water column

These arrays represent the variability of the parameters in the 3D model domain, but we are only changing values here based upon the depth and range `(i,:,j)` since the bottom changes with location in range

Sandia National Laboratories

2D Newport, OR Waveguide: Pre-Processing and Setup

- **Domain setup: Writing the domain input file**
  - This defaults to a 3D model, unless specified otherwise
  - Uses the previously defined spatial and time vectors, `x`, `y`, `z`, and `t` and the `rho` and `vp` arrays that define the density and sound speed for every grid point within the domain
  - The `‘vp’` and `‘rho’` flags designate the model data following, while the data variable name can be changed to reference the previously defined array

```
>> writeSgfdModel(‘newport.cdf‘, x, y, z, t, 'vp', vp, 'rho', rho)
```

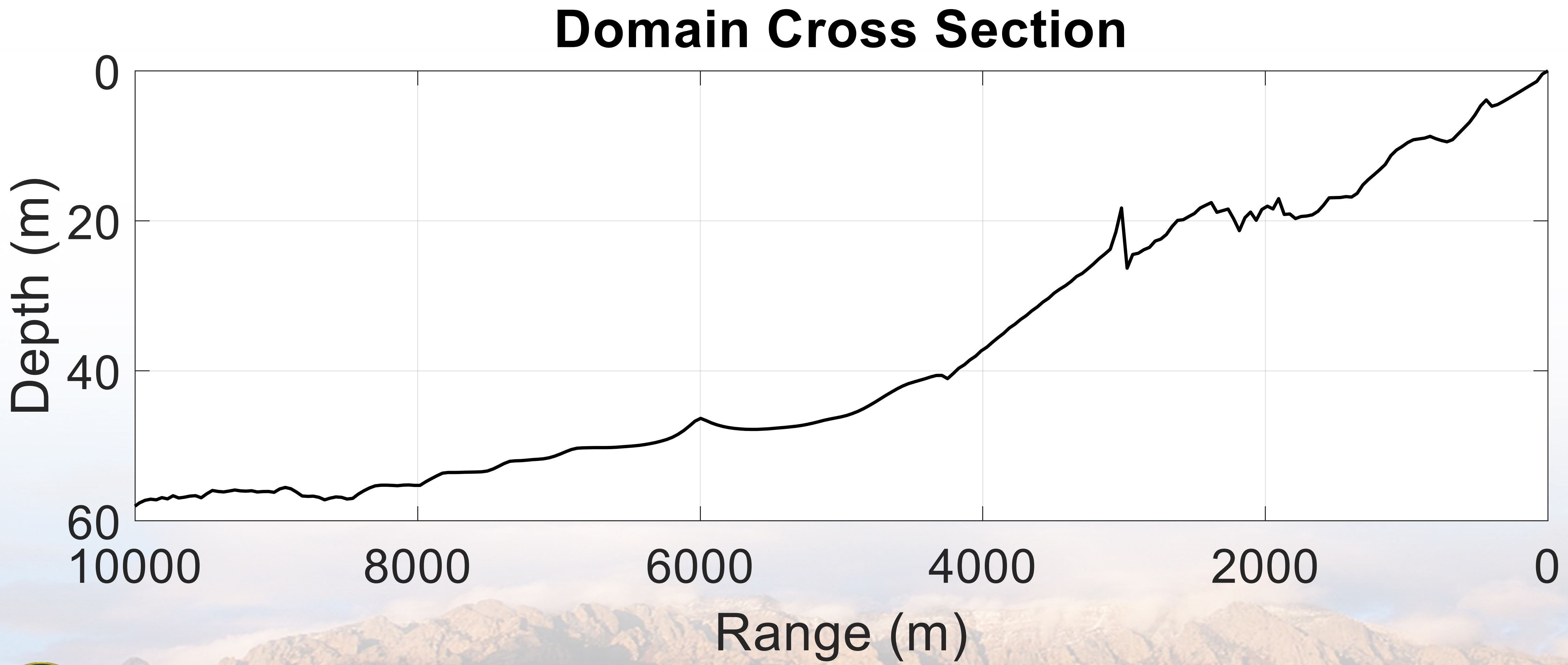
2D Newport, OR Waveguide: *Pre-Processing and Setup*

- **The MATLAB output script defines the run parameters by adding flags when executing Paracousti; below is a suggested script for this problem for convenience**
  - This includes defining the boundary conditions, source, and simulation outputs
  - Newport, OR Example run script:

    -
      ```
      mpirun -np 4 ParAcousti_RHEL6 newport.cdf -p 1 1 3 -bF -bpc6 10 1e-6 314
      1 10 1e-6 314 1 10 1e-6 314 1 10 1e-6 314 1 2 1 314 1 10 1e-6 314 1 -Sw
      source.txt -Se 3600 0 20 1 -Rg Pressure 5:100:7405 0:0 10:10:80 -Ro
      newport.trace.cdf -En 500 Pressure XZ 0 -Eo newport.slice.cdf
      ```
  - A grid of receivers (-Rg) was established at locations; 5:100:7405 0:0 10:10:80 for x, y, and z, respectively
  - 500 slices (-Eo) were requested in the XZ plane at y=0 over the total simulation run time
  - All output data will be available in either the new *.trace.cdf and *.slice.cdf files

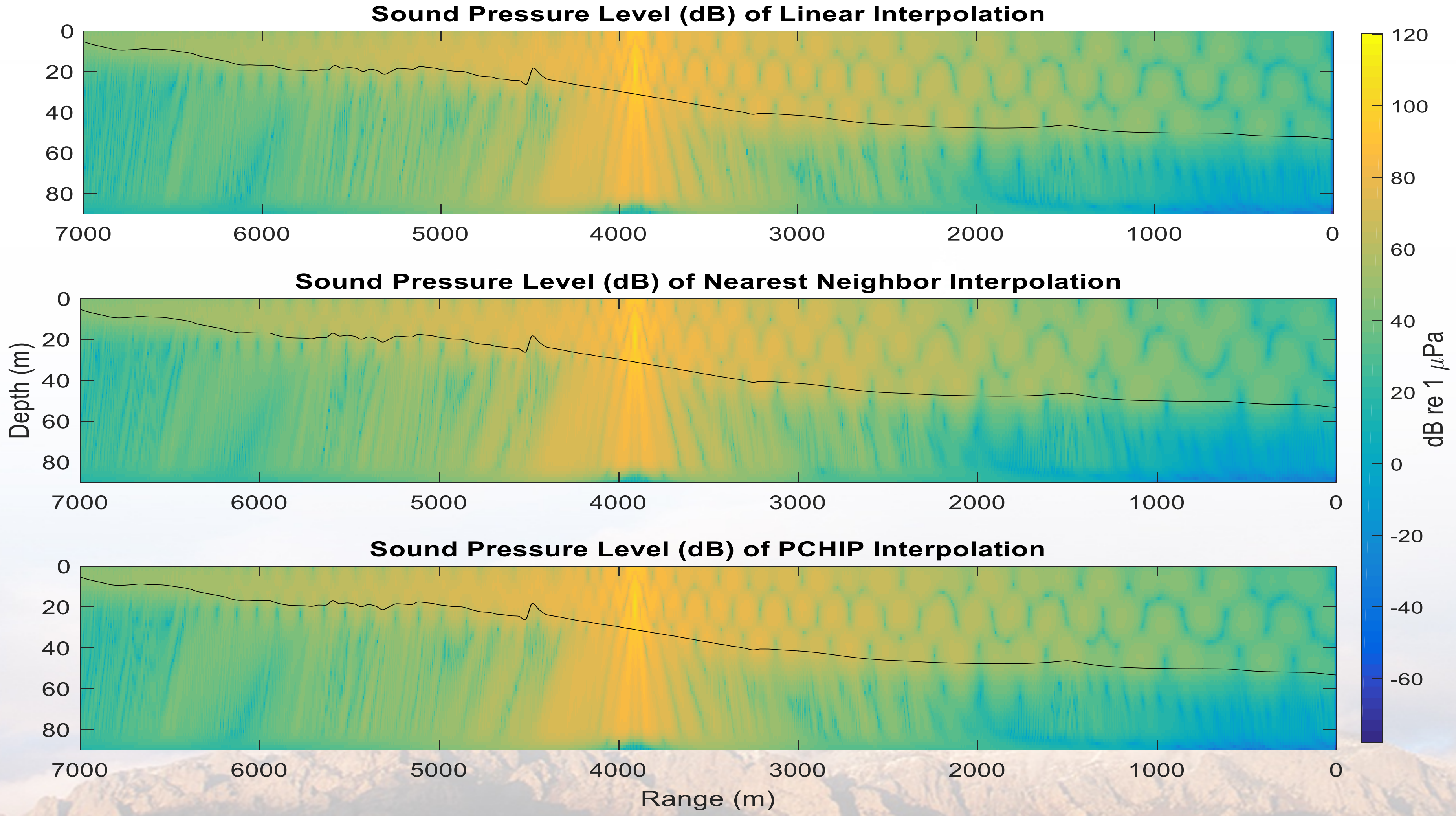
Sandia National Laboratories

2D Newport, OR Waveguide



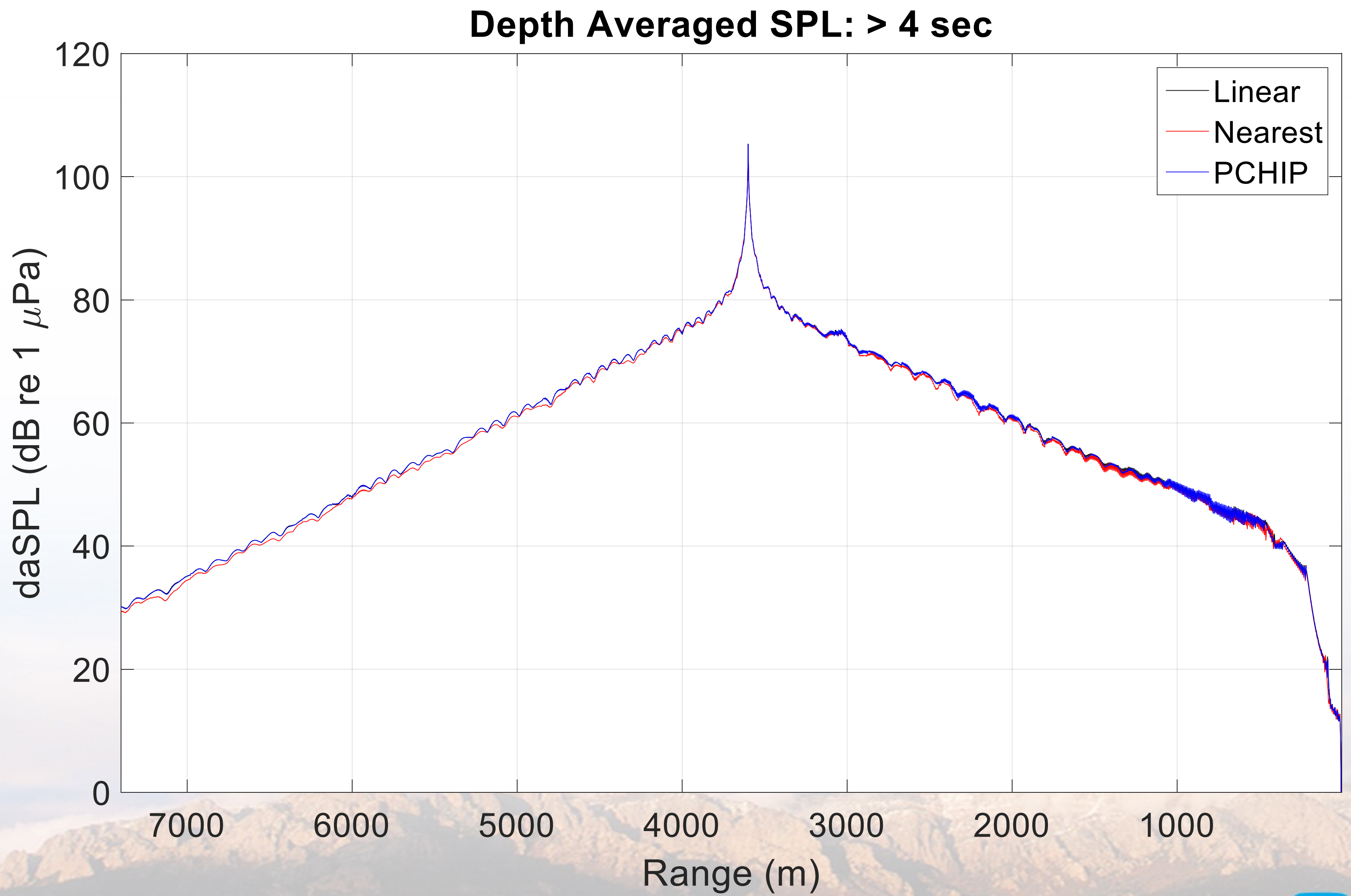
Sandia National Laboratories

2D Newport, OR: Example Output



Sandia National Laboratories

2D Newport, OR: Example Output



Best Practices: Data Interpolation

- **For any bathymetry which is not provided via a grid, use the MATLAB function**

```
>> F = scatteredInterpolant(bathyx,bathyy,bathyz)
```

  - `‘scatteredInterpolant’` provides an array of all values from the bathymetry in the x, y, and z directions from which you may query any value
    - Input arrays are treated as scattered data
    - Values are interpolated standard using linear interpolation
    - For example, to call a value; Value = F(x, y, z) for location x, y, z
- **3D and 2D bathymetric interpolation is specific to the original data set**
  - For all bathymetries, the same process will apply, but how you orient and organize the cross-section will be dependent on the environment you are modeling
- **For post-processing information see Tutorial 2**

Sandia National Laboratories

More Information

- **More information, user manual, and example files can be found at:**
  - https://snl-waterpower.github.io/Paracousti/
- **Source code and executables can be found at:**
  - https://github.com/SNL-WaterPower/Paracousti/
- **Future documentation:**
  - Development of additional tutorials and example cases
  - Additional pre- and post-processing options with Python
  - Other documentation:
    - Preston, L. "TDAAPS2: Acoustic wave propagation in attenuative moving media," Sandia National Laboratory, Alberquerque, Technical Report, pp. 158, 2016
    - Hafla, E., Johnson, E., Johnson, C.N., Preston, L., Aldridge, D., and Robert, J.D. "Modeling underwater noise propagation from marine hydrokinetic power devices through a time-domain, velocity-pressure system," J. of Acoust. Soc. Of Am., 143(3242), pp. 12, 2018

Sandia National Laboratories

Contact Information

- **Sandia National Laboratories**
  - Program Lead

    Jesse Roberts

    Water Power Technologies Dept.

    jdrober@sandia.gov
  - Lead Developer

    Leiph Preston

    Geophysics and Atmospheric Science Dept.

    lpresto@sandia.gov

- **Montana State University**
  - Application Lead

    Erick Johnson

    Mechanical Engineering Dept.

    erick.johnson@montana.edu
  - Graduate Researcher

    Erin Hafla

    Ph.D. Candidate

    erinhafla@gmail.com