# Paracousti

2D/3D modeling of underwater acoustics

Tutorial 3: Post-Processing for a Simple 2D Model



Sandia National Laboratories

# *Tutorial Objectives and Assumptions*

- **Objectives**
  - Introduce users to Paracousti
  - Provide users a step-by-step guide to analyzing the data output from a simulation modeling a simple sound environment and noise source in 2D
- **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 a familiarity with and access to Linux
  - **Users have completed Tutorial 1 and are familiar with the Pekeris Example**

# Tutorial Outline

- Introduction
- Definitions
- Paracousti Workflow
  - Post-Processing
  - MATLAB and NetCDF Files
- 2D Example: Pekeris Waveguide Output
  - Data Output Options
  - Output Visualization
    - Time Slices
    - Traces
- Best Practices
- More Information

# 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)

# 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
  - This 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 3 assumes a prior run of Paracousti and a set of designated output files**
  - This tutorial deals with further analysis of out put data files defined as the post-processing steps only from the Pekeris Example 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:**
  - The Pekeris MATLAB scripts to indicate parameters associated with output files
  - The NetCDF input cdf data files; output from Tutorial 1 provides the slice/trace files
  - The MATLAB scripts used to perform simple post-processing of the results
  - These can be found at: https://github.com/SNL-WaterPower/Paracousti

# Workflow: Pre- and Post-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 provides many options for data output depending on the requirements of the example
  - Various options will be discussed wherein
- **Post-processing is the step of taking and manipulating the output data that Paracousti creates to analyze a problem**
  - Trace data can be analyzed similarly to any hydrophone recording
  - Slice data provides an instantaneous snapshot of the sound field
  - Volume output provides

# Workflow: MATLAB and NetCDF Files

- **Because Paracousti requires an earth model written as a NetCDF file MATLAB provides many built in functions already to identify and access data in these files**
- `ncinfo`(filename.cdf)
  - Returns all of the information about the NetCDF data source and can be saved into a variable
- `ncread`(filename.cdf, variablename)
  - Read data from a variable in the NetCDF file
  - In addition to pre-defined variables, this will also include names for your output traces and slices
- `ncdisp`(filename.cdf)
  - Displays all the groups, dimensions, variable definitions, and all attributes in the NetCDF data source as text in the Command Window

# Workflow: MATLAB and NetCDF Files

- **The information returned from `ncinfo()` is stored as a structure and can be accessed by appending deeper levels**

```
>> finfo = ncinfo('baseline.cdf')
```

```
finfo =
  struct with fields:
      Filename: ..\baseline.cdf'
          Name: '/'
    Dimensions: [1×5 struct]
     Variables: [1×9 struct]
    Attributes: [1×2 struct]
        Groups: []
        Format: 'classic'
```

- **To see the variable names available**

```
>> finfo.Variables.Name
```

```
ans =
    'minima'
```

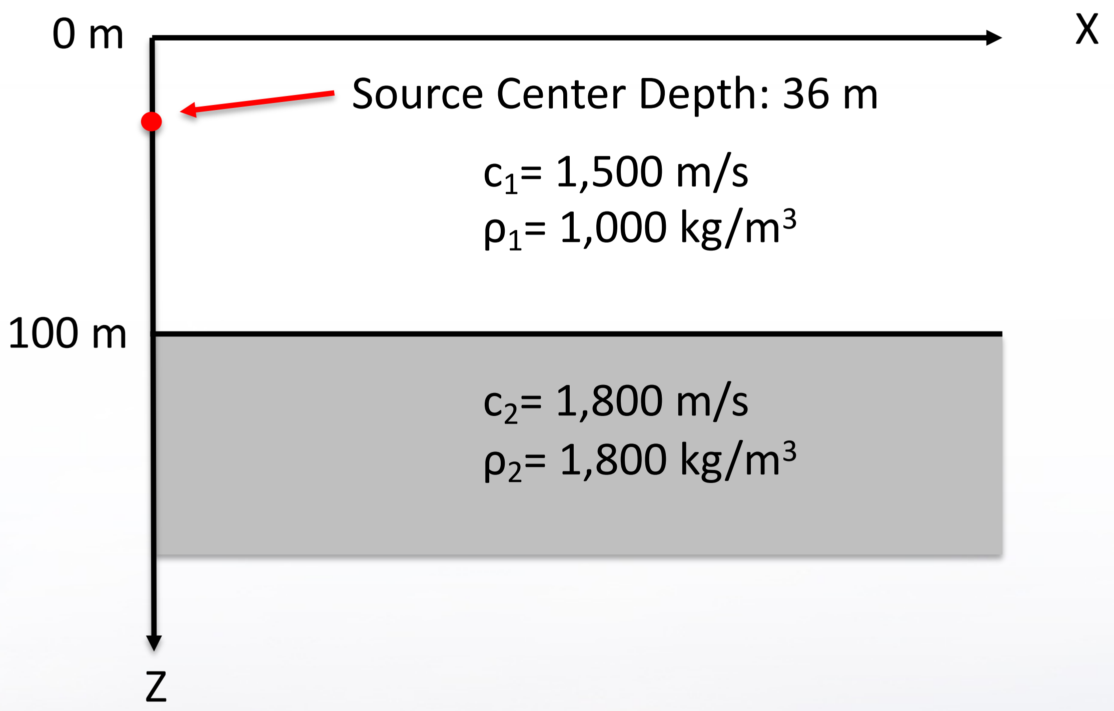
- **Which can then be used to store data from a variable**

```
>> fminima = ncread('baseline.cdf', 'minima')
```

```
fminima =
  4×1 single column vector
   -50
   -50
   -50
     0
```

# 2D Example: Pekeris Waveguide

- **The problem**
  - Pekeris waveguide with a continuous, sinusoidal source in a 3D simulation
  - See Tutorial 1 for full setup
- **Domain setup**
  - Depending on output parameters, some output types require that they are assigned to actual grid locations over interpolation
  - Initial time step parameters are required for temporal orientation of slices
  - Additional space required for boundary conditions are indicated by (-)




```
>> x = -160:1:4000
>> y = -160:1:160
>> z = -2:1:200
>> t = 0:0.0002:6
```

# 2D Pekeris Waveguide: Pre-Processing and Setup

- **The MATLAB output script defines the run parameters by adding flags when executing Paracousti and do no strictly require any MATLAB capabilities**
  - This includes defining the boundary conditions, source, and simulation outputs
  - Pekeris Example run script:
    - ```
      mpirun -np 4 ParAcousti_RHEL6 pekeris3D.cdf -p 1 1 3 -bF -bpc6 10 1e-6 62 1 10 1e-6 62 1 10 1e-6 62 1 10 1e-6 62 1 2 1 62 1 10 1e-6 62 1 -Sw source.txt -Se 0 0 36 1 -Rg Pressure 5:100:3905 0:0 10:5:200 -Ro pekeris3D.trace.cdf -En 1000 Pressure XZ 0 -Eo pekeris3D.slice.cdf
      ```
  - A grid of receivers (-Rg) was established at locations; 5:100:3905 0:0 10:5:200 for x, y, and z, respectively
  - 1000 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
      ```

# 2D Pekeris Waveguide: Pre-Processing and Setup

- **Available trace commands for output**
  - Add one receiver of *type* data collected at any domain location, x y z

    ```
    -R 'Type' x y z
    ```

  - Specify individual trace locations or automate multiple traces on a grid

    ```
    -Rg 'Type' rxmin:dxr:rxmax rymin:dyr:rymax rzmin:dzr:rzmax
    ```

  - Specify individual trace locations or multiple traces through text file

    ```
    -Rf3 'Type' filename.txt
    ```

    - The text file must specify the x, y, and z location of each reciever
  - Range of x, y, and z values indicate locations of receivers in domain. These do not need to match domain grid
    - **data is interpolated between grid cells and defaults to a cubic**
  - Receivers may not be located within space required for boundary conditions

# 2D Pekeris Waveguide: Pre-Processing and Setup

- **Available planar slices commands for output**
  - Defines total number of instantaneous snapshots in time can be collected on Cartesian planes over the entire simulation run time

    ```
    -En N 'Type' 'Plane' 'Position'
    ```

  - Define an output of insantaneious snapshots in time at times specified by a MATLAB vector

    ```
    -Et minT:Dt:maxT 'Type' 'Plane' 'Pos'
    ```

  - Slice output covers the entire defined domain dimensions, including boundary conditions
    - Data calculated in the area beyond or defined spatially for a boundary condition should be omitted from a figure as it is not directly part of the solution area

# 2D Pekeris Waveguide: Pre-Processing and Setup

- **Available commands for output**
  - The trace output file
    - Designates the file to collect the recorded data at each grid point defined by the receiver locations

      ```
      -Ro pekeris3D.trace.cdf
      ```

    - Only one trace output file may be defined per run
  - The slice output file
    - Designates the file to collect the recorded data

      ```
      -Eo pekeris3D.slice.cdf
      ```

    - Multiple slice output files may be requested or defined per run

# 2D Pekeris Waveguide: Post-Processing

- **Post-processing includes formatting and accessing any output files requested**
  - Slice and trace files are covered. Full volume output is omitted due to file size
  - Pressure output type is used as velocities do not require additional formatting
- **Determine the properties associated with any NetCDF output file; slice or trace files**
  - Instead of remembering how many slices we have, we can use `ncinfo()` is used to determine any slice file properties

```
>> slice_info = ncinfo('pekeris3D.slice.cdf')
>> [~,~,~,slice_length] = slice_info.Dimensions.Length
```

  - we can look at slice_info.Dimensions.Name to determine which column we want the length from (the 4th)
  - ncinfo may also be used to determine variable names as necessary within NetCDF files

# 2D Pekeris Waveguide: Post-Processing

- **Slices may be selected singularly or averaged together over a period of time**
  - A steady state solution for a domain requires that the slice files be averaged over the period of solution time once the model reaches steady state
  - We collect each pressure slice in order of time and store it in the 3D variable `P`, (Pa)

```
>> for i = slice_length;
     P(:,:,i)=squeeze(ncread('pekeris3D.slice.cdf','xzPressure',[1 1
         i],[inf inf 1]));
     end
```

- `P` is comprised of 2 spatial dimensions and the 3rd is for each time snapshot
- `squeeze()` reduces the spatial order of the data into a 2D array
- The storage variable names will be organized by the data type and orientation you requested when you ran Paracousti. In this case, `xzPressure`
- For a singular slice, i = slice number
- To determine the simulation time the slice was taken at: *time = (T/totalslice#)*i*

# 2D Pekeris Waveguide: Post-Processing

- For slices averaged over a range

```
>> for i = slice_min:slice_max;
    P(:,:,i-(slice_min – 1))=squeeze(ncread('pekeris3D.slice.cdf','xzPressure',[1 1
          i],[inf inf 1]));
    end
```

  - `slice_min` and `slice_max` indicate the min and max counters for a range of slices
- Data is now in a matrix of pressure values in a particular selected plane (XZ) and may be formatted into a SPL or TL
- From here, we can quickly calculate the root mean squared pressure

```
>> Prms=sqrt(mean(P.^2,3))
```

  - Note that slice output is already a Pressure value in Pa
- And then calculate the SPL or TL

```
>> SPL = 20.*log10(Prms./1e-6)
>> TL = 20.*log10(Prms./P_1m)
```

  - Where `P_1m` is the pressure 1 m from the source and may be obtained from the Trace files or source parameters

# 2D Pekeris Waveguide: Post-Processing

- Pressure data may additionally be averaged in water depth; for Pekeris:

```
>> for i = 1:xmax;
     Pavg(i)= average(Prms(i,1:zmax));
     end
```

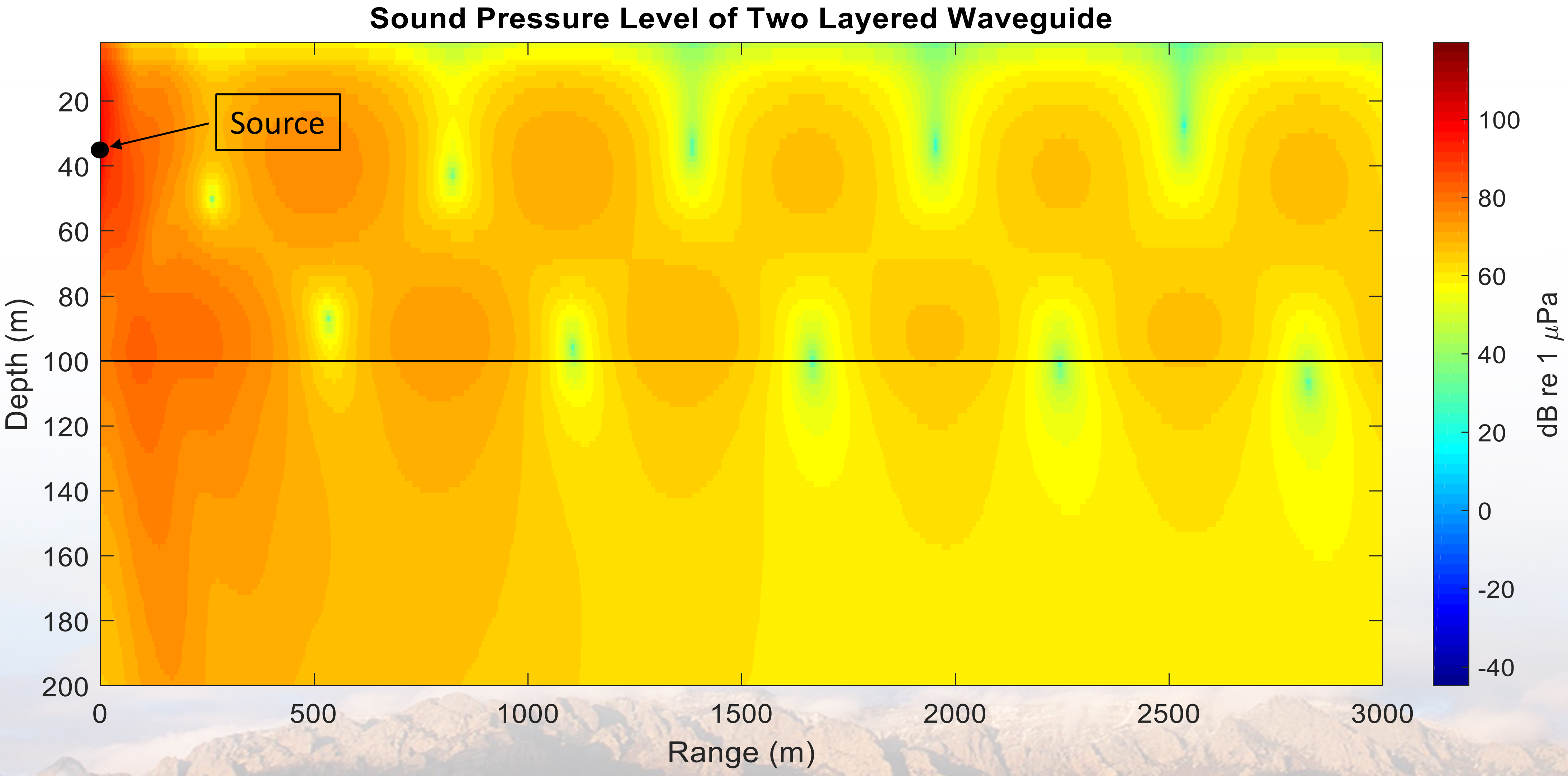
  - Where xmax indicates the range of values you would like to average over and zmax is the bottom depth
  - All depth averaged values may then be converted to SPL or TL as required

- MATLAB provides a lot of plotting options, but an easy way to display the full color representation of the SPL array is to use `imagesc()`

```
>> imagesc(x,z,SPL')
```

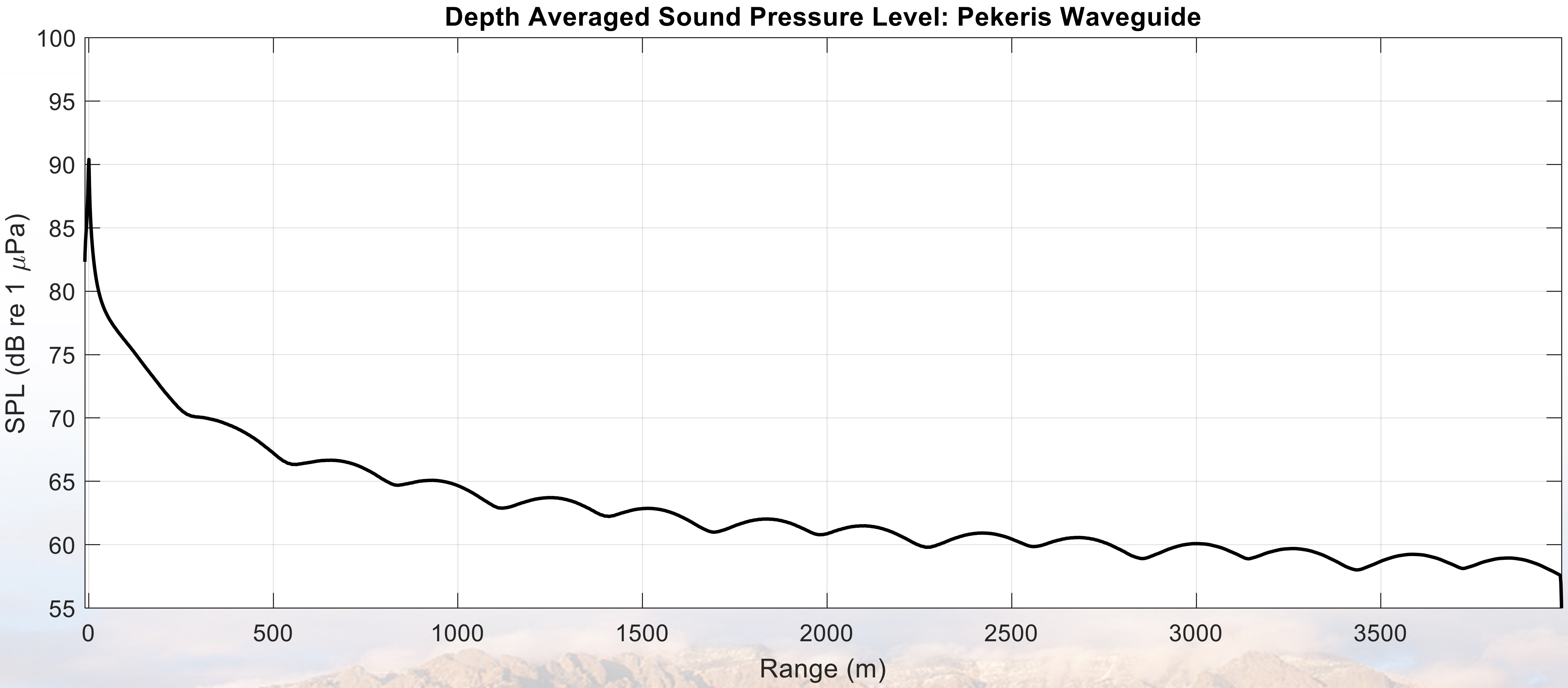
- For any 2-D plots, MATLAB's `plot()` is sufficient

# 2D Pekeris Waveguide: Post-Processing




Sandia National Laboratories

# 2D Pekeris Waveguide: Post-Processing



# 2D Pekeris Waveguide: Post-Processing

- **Trace data may be plotted for a single point in time; several plots may be overlaid**
  - Trace pressure data is output as Pa and each location may be plotted to easily check the validity of a solution
  - We collect each pressure trace for each location and store it in the 3D variable `P`
  - Each trace designation

```
>> for i=1:maxtrace
          P = ncread('pekeris3D.trace.cdf','receiverData',[1 i],[inf 1]);
          plot(time, P)
          pause
end
```

    - `max trace` is the total number of traces; i may also be equal to one value
    - `P` is comprised of 1 spatial location in the xyz and in time, so it is already in a 2D array
    - For some number of traces, the code will cycle through all of them and plot the output

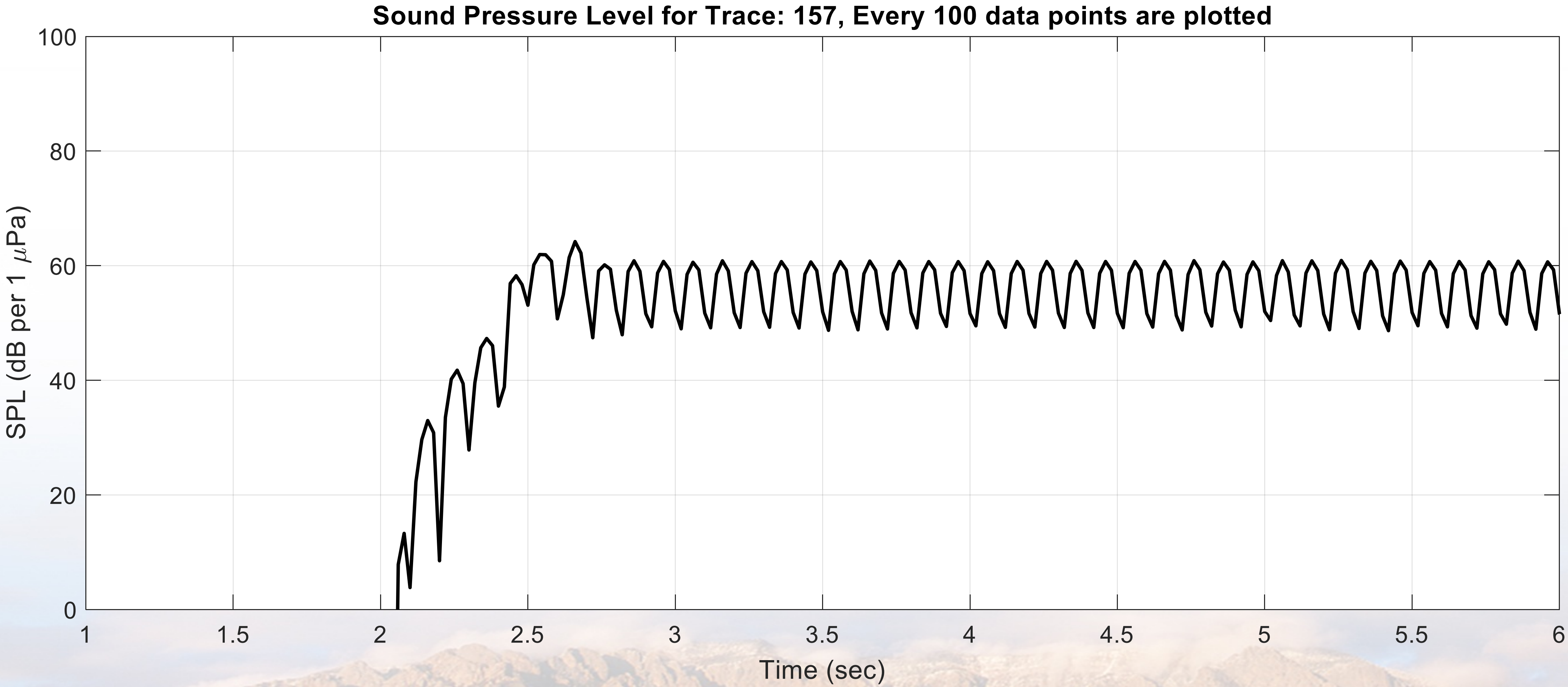
# 2D Pekeris Waveguide: Post-Processing

- **Traces are numbered by the x, y, z location designation. For traces in a grid, the identifier marches through each dimension consecutively**
  - For a Trace number 157, the location would be (105 0 25) for (x y z)
- **Trace data once brought in as a Pressure, may be converted to SPL or TL and plotted**
  - The root mean square of the pressure values (`Prms`) is not a necessary calculation as the data is for a singular location
  - Then calculate the SPL or TL

```
>> SPL = 20.*log10(Prms(i)./1e-6)
>> TL = 20.*log10(Prms(i)./P_1m)
```

    - Where `P_1m` is the pressure 1 m from the source and may be obtained from the Trace files or source parameters
    - The `P_1m` may be asked for by finding a specific value 1 m from the source from a particular Trace location
  - Traces may be plotted using the MATLAB command ‘`plot(x,’SPL or TL’)`’
    - Multiple trace plots may be overlaid on the same MATLAB plot to compare locations

# 2D Pekeris Waveguide: Post-Processing



# *Best Practices: Slices*

- **Determining the best grid spacing based on output locations**
  - Domain size is defined based on area of interest and may be expanded until memory requirements become limiting due to the number of cells
  - Note that along with any spacing requirements due to stability, it is also useful for the spacings to match those of the receivers for easy processing
- **Bathymetry input and grid orientation may not match so should be monitored to make sure all plots correlate with respect to the x, y, and z directions**
- **Asking for a total number of slices is recommended over a time vector**
- **For plotting slices at a particular time, it is best to average that slice data over 3λ**
  - Every slice is tied to a particular point in time. Averaging over 3 wavelengths smooths the output and removes any noise
  - The number of slices for three wavelengths depends on the source, total simulation time, and number of slices output

# Best Practices: Traces

- **Grid output is recommended and may be scaled to a very small number of traces**
  - Even for a singular area of interest, say for a sensor array location, trace data should be collected 1 m away in every direction to compare any energy disipation
  - Grid output also allows for a consistent output across the entire domain. This allows for a check to make sure that the model is producing reasonable data while pulling from small (in comparison to slices) data files
  - Traces should be collected 1 m from the source location to collect the data required for TL calculations and to monitor the source parameters

# 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

# 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