Paracousti

2D/3D modeling of underwater acoustics

Tutorial 1: Building a Simple 2D Model

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under Contract No. DE- NA0003525.

Tutorial Objectives and Assumptions

- **Objectives**
  - Introduce users to Paracousti
  - Provide users a step-by-step guide to building, solving, and analyzing a simple sound environment and noise source

- **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

Tutorial Outline

- **Introduction**
- **Definitions**
- **Paracousti Workflow**
  - Pre- and Post-Processing
  - MATLAB and NetCDF Files
- **2D Example: Pekeris Waveguide**
  - Model Definition
  - Pre-Processing and Setup
  - Solving
  - Determining Sound Pressure Levels
- **Best Practices**
- **More Information**

Brief Introduction to Paracousti

- **Paracousti**
  - 3D, time-domain, underwater acoustic propagation simulator
  - Modification of SNL's TDAAPS (Time-domain Atmospheric Acoustic Propagation Suite)
  - Assumes non-moving ambient medium with no prior stress, ideal fluid, zero shear/bulk viscosity, adiabatic
  - Solves linearization of Cauchy equations of motion through coupled finite difference solution
    - 4th order spatial
    - 2nd order temporal

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

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

Sandia National Laboratories

Capabilities of Paracousti

- **Able to represent**
  - Range-independent domains in 2D and 3D
  - Spatial variation of water and bathymetric properties
  - *N* number of distinct noise sources
  - Allows monopole and higher-order noise sources
- **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

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 ± 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

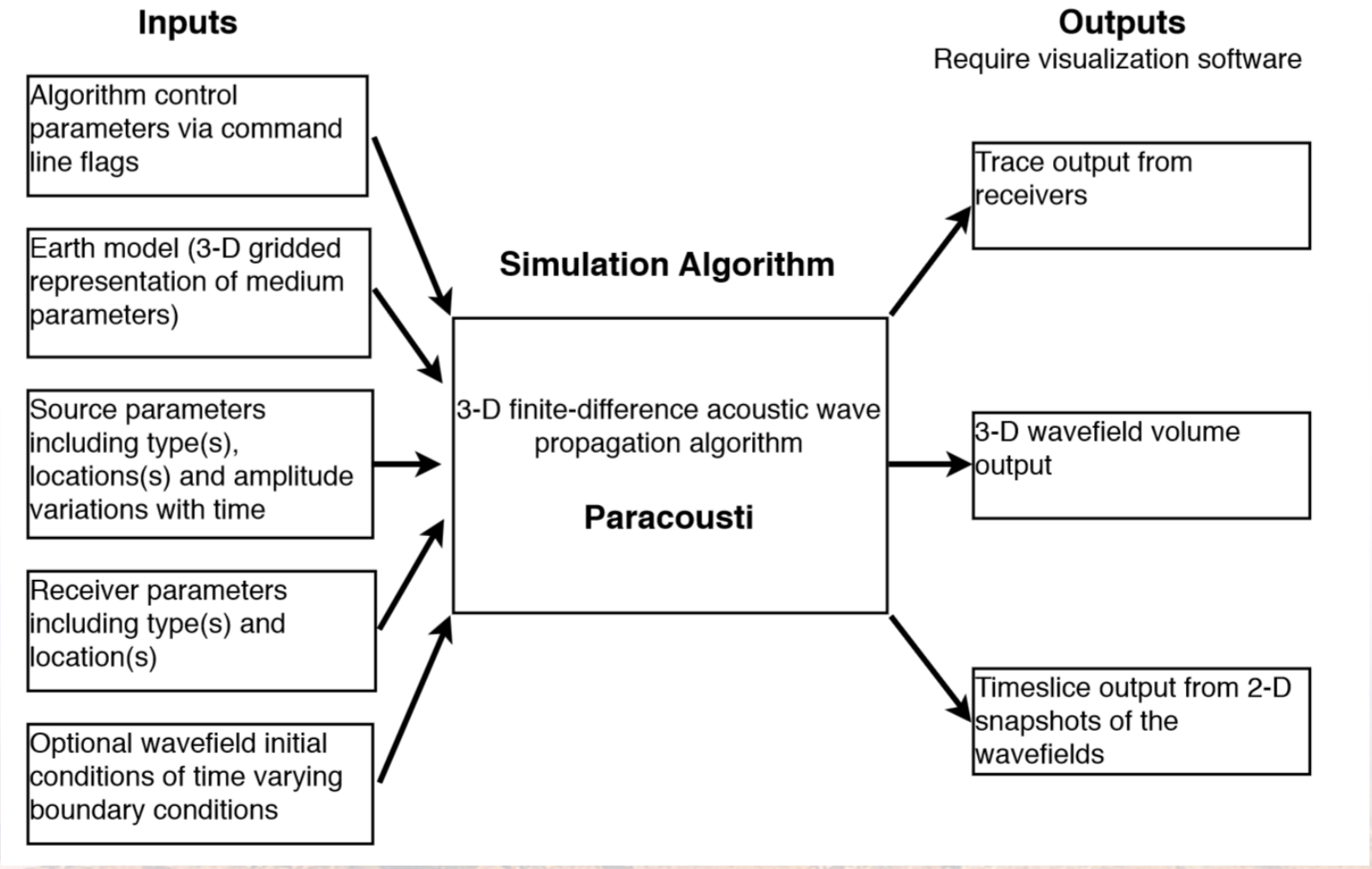
- **Source Time Function – STF**
  - The 1st or 2nd integral, with respect to time, of the source pressure profile for a directional or monopole source, respectively
  - This is the input profile used by Paracousti to define the source
- **Slice**
  - A planar output of particle velocity and/or pressure 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

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

Paracousti Workflow



Pre-processing → Solution → Post-processing

Paracousti Workflow

- **Running Paracousti refers only solves an underwater acoustics problem**
  - All of the problem setup and analysis is split into pre- and post-processing steps performed separately
- **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 MATLAB scripts to define the problem and create a NetCDF input file
  - The NetCDF input file
  - 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 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
  - MATLAB provides many tools to quickly manage arrays of environmental parameters, interpolate measured data into a model domain, and create the input NetCDF file
- **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

Workflow: MATLAB and NetCDF Files

- **Because Paracousti requires an earth model written as a NetCDF file MATLAB provides many built in functions already to write and access data in these files**
- **The provided `writeSgfdModel.m` function uses `nccreate()` and `ncwrite()` to build the full input file for Paracousti**
  - No need to develop this functionality separately
- **The provided `writeSubdomainSgfdModel.m` is similar and builds multiple input files when memory constraints restrict file size**
  - Generally used for large 3D simulation runs
  - Requires user to specify the number of splits to the domain
  - The maximum size for a single NetCDF file is ≈ 2.6 GB

Workflow: MATLAB and NetCDF 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

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**

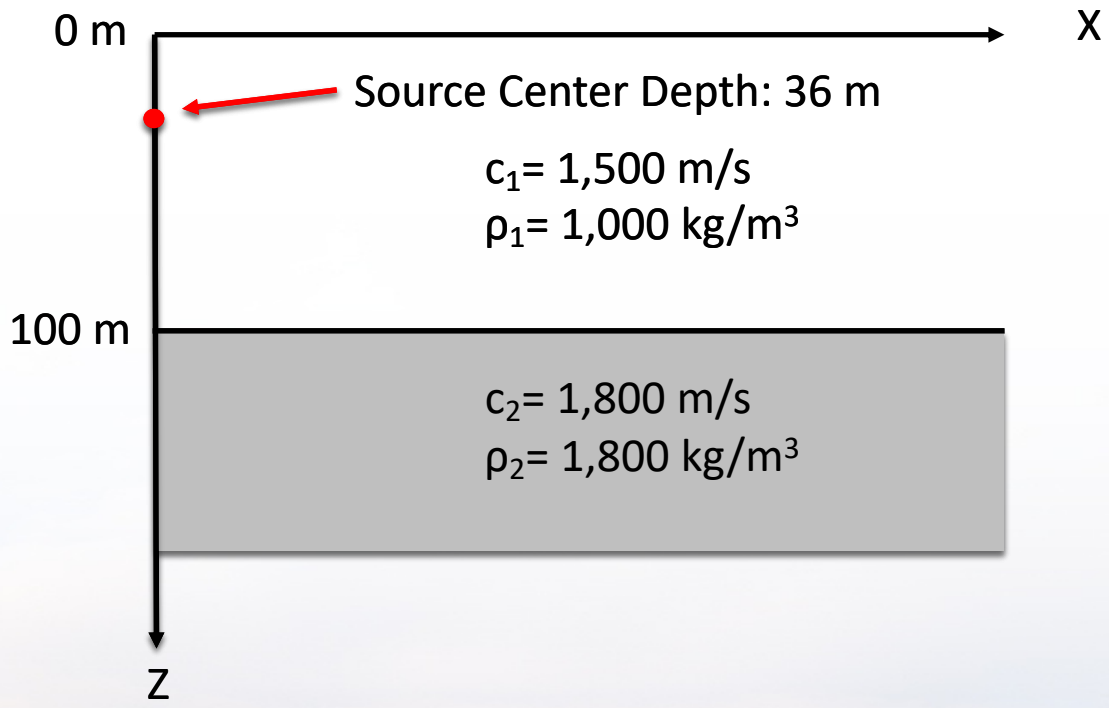
```
>> fminima = ncread('baseline.cdf', 'minima')
```

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

Sandia National Laboratories

2D Example: Pekeris Waveguide

- **The problem**
  - Determine the SPL in a Pekeris waveguide with a repeating source.
- **Model properties**
  - One monopole source
    - Oriented along the LHS of domain
    - ω = 20 Hz, amplitude = 1 Pa
  - 2-layer waveguide
  - Each domain has distinct and constant properties
  - Flat and smooth interface
  - Modeled as 3D
    - 1D is also possible for range-independent problems



2D Pekeris Waveguide: Pre-Processing and Setup

- **Domain setup**
  - Create vectors that define
    - The desired spatial discretization along their respective axes x, y, and z
    - The simulation's time history
    - These are of the form

```
>> x = min : step_size : max
```

  - While the domain extents are often sufficient to determine these sizes, ensure additional grid points are available when
    - Inserting a pressure-free boundary
    - Near an interface between changing materials
    - Sources and/or receivers are near a boundary

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

While we're only solving a "2D" problem, y has large enough dimensions to minimize low angle grazing from the very low frequency

2D Pekeris 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
    - Changes in values determine a difference in medium properties
      - **Such as defining the location of any changes in bathymetry**
    - For a constant water layer on top of sediment, define arrays of values based on the grid location

```
>> for i = 1:length(z)
        if z(i) <= 100
            vp(:,:,i) = 1500; rho(:,:,i) = 1000;
        else
            vp(:,:,i) = 1800; rho(:,:,i) = 1800;
        end;
 end
```

The water surface starts at $z=0$ and gets larger in depth, so anything less than 100 m 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 `(:,:,i)` since the bottom is constant and flat

2D Pekeris 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('pekeris3D.cdf', x, y, z, t, 'vp', vp, 'rho', rho)
```

  - This domain model can be reused to solve problems with different sources and boundary conditions

2D Pekeris Waveguide: Pre-Processing and Setup

- **Source setup: Defining a source profile**
  - Define the source parameters (single)
    - Desired regular frequency (`sf`) in [Hz]
    - Scaling amplitude (`amp`) in [-],
    - Cartesian location (`[sx,sy,sz]`) in [m]

```
sf = 20
amp = 1
[sx, sy, sz] = [0, 0, 36]
```

  - Create the source time function for Paracousti
    - For a repeating harmonic source defined by a pressure of $P = \cos(2\pi\omega t)$
    - The double integral for an explosive monopole source that can be differentiated into a smooth continuous function is then

```
>> stf = c^2/(pi()*sf^2)*(1-cos(2*pi()*sf*t))
```

`c` is the sound speed at the source location

    - We recommend creating a source time function with a unity amplitude and scaled by `amp`

this includes terms from the integration AND a required scaling factor

Sandia National Laboratories

2D Pekeris Waveguide: Pre-Processing and Setup

- **Source setup: Writing the input file**
  - Write a new input file for each source as a tab-delimited text file with no header information

```
>> sourceTable = table(t',stf')
>> writetable(sourceTable,'sourcepekeris.txt', ...
   'writeVariablenames',0,'delimiter','\t')
```

  - This source file is called later when running Paracousti

Sandia National Laboratories

2D Pekeris Waveguide: Pre-Processing and Setup

- **While building the domain and source time function are easily accomplished with MATLAB, the remainder of the setup is finished by adding flags when executing Paracousti and do no strictly require any MATLAB capabilities**
  - This includes defining the boundary conditions, source, and simulation outputs
  - However, we encourage the user to include these within their MATLAB scripts to
    - Conveniently repeat and change a simulation
    - Provide a record of what conditions were run
    - Avoid fat fingering a command
    - Automate performing a batch of simulations

2D Pekeris Waveguide: Pre-Processing and Setup

- **Defining the boundary conditions**
  - The CPML is defined by 4 parameters for each six boundaries and specified by

```
-bpc6 nXmin RXmin aXmin kXmin nXmax RXmax aXmax kXmax nYmin RYmin aYmin kYmin nYmax
RYmax aYmax kYmax nZmin RZmin aZmin kZmin nZmax RZmax aZmax kZmax
```

> n is the CMPL thickness and we recommend 10
> $R <= 0.001$
> $a = \pi\omega_{peak}$, where ω_{peak} is the dominant frequency of the source
> $k = 1$

  - For the 2D Pekeris Waveguide

```
-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
```

  - A pressure-free boundary is applied to the air/water interface at z = 0 with `-bF`
    - This overwrites the CPML on that boundary
    - We could alternatievly add sufficient grid points to the air domain with a density = 0, leaving the CPML boundary condition

2D Pekeris Waveguide: Pre-Processing and Setup

- **Defining the source**
  - Indicate the source profile is defined through the input file written previously
  - This source is a monopole (explosion) source at the already defined location with an additional scaling amplitude, amp

```
-Sw sourcepekeris.txt
-Se sx sy sz amp
```

  - Multiple sources can be added in sequence with individual locations and source text files as necessary

2D Pekeris Waveguide: Pre-Processing and Setup

Defining traces as outputs

- You can specify individual trace locations or automate multiple traces on a grid

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

  - Type = data collected, e.g. `pressure`
  - 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**
  - dxr, dyr, dzr are the step size between receiver locations
  - For the 2D Pekeris Waveguide

```
-Rg Pressure 5:100:3905 0:0 10:5:200
```

In this case no data is collected along the y-dimension

- 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
```

2D Pekeris Waveguide: Pre-Processing and Setup

Defining planar slices as outputs

- Instantaneous snapshots in time can be collected on Cartesian planes

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

  - N = number of snapshots in time evenly spaced over the total model run time
  - Type = data collected, e.g. particle velocity components `vx`, `vy`, `vz`
  - Plane = 2D plane data can be collected on each of the three planes, i.e. `XY`, `XZ`, `YZ`
  - Position = Location along the third axis the snapshot will occur.
  - N should divide evenly into the model run time
  - For the 2D Pekeris Waveguide

```
-En 1000 Pressure XZ 0
```

This collects 1000 pressure snapshots on the XZ-plane at y=0

- The slice output file
  - Designates the file to collect the recorded data

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

2D Pekeris Waveguide: Solving

- **Run Paracousti from the terminal by typing the whole model setup**

```
mpirun -np N Paracousti filename.cdf -p px py pz boundary, source, outputs
```

  - `-np N` denotes N number of processors to be used in the solution
  - `-p px py pz` determines the parallelization breakdown in each direction, where

```
N = px*py*pz+1
```

  - `boundary, source, outputs` consist of all of the additional flags required to define a model run

- The complete 2D Pekeris Waveguide is run with

```
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
```

- Note that after Paracousti has completed its run all output data will be available in either the new *.trace.cdf and *.slice.cdf files

2D Pekeris Waveguide: Post-Processing

- **To determine the sound pressure level from the model we need to access all of the pressure values we have recorded on the slices**
  - 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)

  - We collect each pressure slice in order of time and store it in the 3D variable `P`

```
>> 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`

2D Pekeris Waveguide: Post-Processing

- 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

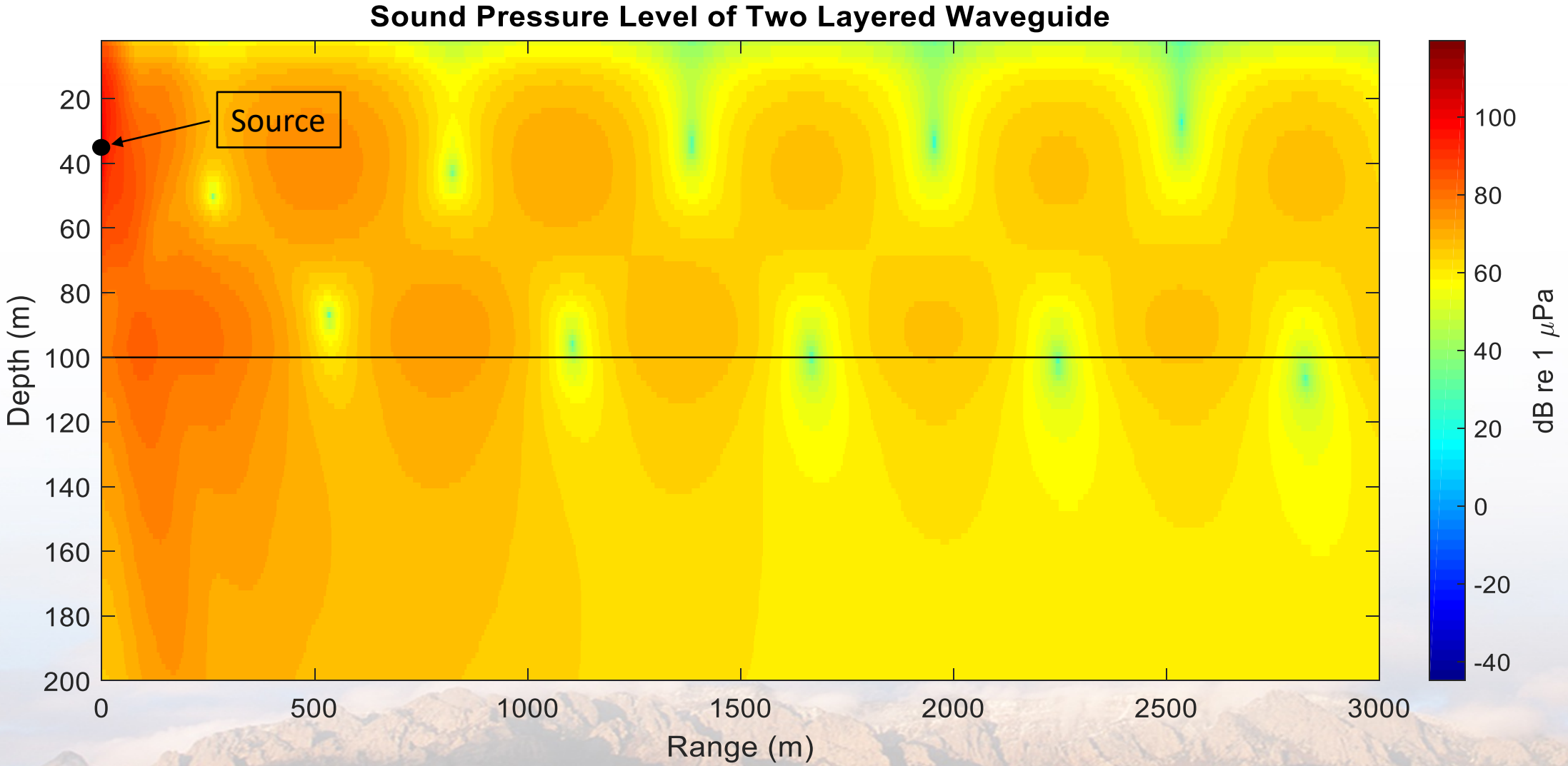
- And then calculate the SPL

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

- 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')
```

2D Pekeris Waveguide: Post-Processing



Best Practices: Grid Size

- **Determining the best grid spacing based on model parameters**
  - Domain size is defined based on area of interest and may be expanded until memory requirements become limiting due to the number of cells
  - Best if the grid step size is the same size in all directions, i.e. `dx=dy=dz`
  - We recommend that a minimum of 10 points per the smallest source wavelength is used
  - Ensure that the grid spacing is no larger than

```
>> dx=min(vp)/sf/10
```

Best Practices: Timestep

- **Determine the timestep based on model parameters**
  - With an explicit finite difference scheme, solution stability depends on the timestep size
    - This will give erroneous results
    - Resulting errors in the model can cause the system to crash
  - Ensure that the timestep is no larger than

```
>> dt=dx/max(vp)/2.04
```

    - The 2.04 is a product of the finite difference coefficients used
- **As this is a transient simulation, do not neglect having a sufficiently large solution time in order to capture the correct problem statistics**
  - That is, a steady state is not achieved in less than $\frac{domain\ length}{smallest\ sound\ speed}$

Best Practices: Boundary Conditions

- **The pressure free boundary condition**
  - Applied across the top of the model to simulate an air-water and/or air-earth interface
  - The interface is placed at `z = Zmin + 2*dz`
    - Top of water surface is `z = 0`, with positive `z` increasing in depth
    - The additional `2*dz` spacing is required from the finite difference scheme
    - This means `Zmin` should be defined at `-2*dz`
- **The CPML requires a buffer zone of a minimum `10*dx` or `10*dy`**
  - This absorbs sound leaving the domain and prevents reflections
  - High frequency sources may require larger buffer zones

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