

# **SeidarT 1.0 Documentation**

**version 1.0**

**Steven Bernsen  
Christopher Gerbi  
Ian Nesbitt**

June 02, 2021



# Contents

<b>SeidarT 1.0</b>	<b>1</b>
About	1
Statement of Purpose	1
Known issues	1
Installation	1
Auto-installation	2
Manual installation instructions	2
Folder structure	4
Getting Started	4
General workflow	4
Files to generate or edit	5
Examples	7
Shapes model: create a 2.5D animation of wave propagation	7
Fill model: create a single shot seismogram and wiggleplot	8
Fill model: create a common offset radar profile	9
Routines and wrappers	9
Routines	10
prjbuild.py	10
sourcefunction.py	10
prjrun.py	10
im2anim.py	10
arraybuild.py	10
rcxdisplay.py	10
orientation_tensor.py	10
Wrappers	10
common_offset.sh	10
common_midpoint.sh	11
prjbuild	11
sourcefunction	11
prjrun	12
common_offset	12
common_midpoint	13
orientation_tensor	13
arraybuild	14
rcxdisplay	14
wiggleplot	15
im2anim	16
vtkbuild	16
implot	16
imvector	17

vectoranim	17
Creating a png file	18
Building documentation	18
Currently available materials	18
Units	19
References	19
Search documentation	19

# SeidarT 1.0

<https://github.com/sbernsen/SeidarT>

## Multi-layer Fortran-based geophysical wave modeling

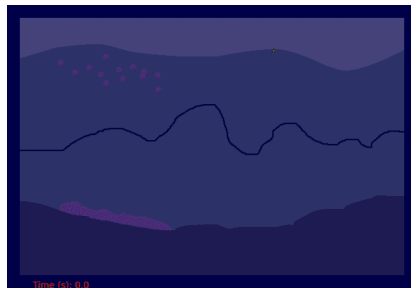
by Steven Bernsen and Christopher Gerbi

The Seismic and Radar Toolbox (SeidarT) is a collaboration between researchers at the Universities of Maine and Washington to provide an open source platform for forward modeling mechanical and electromagnetic wave propagation. The major objective of the project is to easily and quickly implement isotropic and anisotropic complex geometries and/or velocity structures to develop prior constraints for - not limited to - investigating, estimating, and imaging englacial ice structure, sub-glacial boundary conditions on the sub-regional scale. Larger problems would require the curvature of the Earth to be taken into consideration, but many glacier seismic and radar experiments do not expand into regional parameter estimation and velocity modeling.

Much of this code has been adopted from the SEISMIC\_CPML software provided by [Computational Infrastructure for Geophysics \(CIG\)](#). Further details to the backend numerical code can be found in the References section.

## About

SeidarT is a powerful numerical modeling tool to calculate englacial seismic and electromagnetic wave propagation. Though it grew out of a desire to simulate field surveys in glaciers and ice sheets, the code is general enough for any Earth material.



*An example of model output calculated by SeidarT, showing wave propagation through multiple material types along the X-Z plane and complex geometry between the layers.*

%%%%%%%%%% **This animation should be redone without background noise** %%%%%%%%%%%

## Statement of Purpose

## Known issues

- The volume around the source rings in the electromagnetic modeling.

## Installation

The dynamical programming language of Python3 is used as a front end to run the more computationally extensive modeling schemes in Fortran. There are a number of ways to install this software depending on your desired level of control over the process. Most users should be fine with the "Auto-installation" in the section below.

If you are an advanced user and looking to take more in-depth control of the installation process, you can use the Manual installation instructions.

Below the installation instructions, we describe the Folder structure.

**Note**

**Mac OS X users:** you will have to install [Homebrew](#) prior to installing this software if you do not already have it.

**Auto-installation**

First, some background:

This method will install SeidarT on most Unix-based systems using a combination of Anaconda and Fortran compilation. It's not necessary to know or do much more than execute a few command line entries. However, it will help to know that the installer will create what's called a "virtual environment" or more specifically an "Anaconda environment". Without getting into details, this conda environment is basically a way to create a virtual Python configuration that will let this software run but not affect your system's Python configuration, which is a fancy way of saying that conda will avoid causing cascading bad effects on your computer. In short: conda is nice because it plays nicely with various systems. Note well that you will have to activate this conda environment whenever you wish to run SeidarT. This is not hard but it is critical. Read on for instructions on how to both install and activate your SeidarT conda environment.

**1. First, make sure you have the proper prerequisites**

On Mac OS X systems with [Homebrew](#), copy/paste/enter the following two lines:

```
brew update
brew install gcc git
```

**or** on Debian/Ubuntu systems that use the [aptitude](#) package manager, do the following to get those prerequisites:

```
sudo apt update
sudo apt install gcc-9 gfortran git
```

**2. Get the software**

Change to the directory where you'd like SeidarT to go, clone it from GitHub, and change into the SeidarT directory created by Git:

```
cd /path/to/parent/directory
git clone https://github.com/sbernsen/SeidarT
cd SeidarT
```

**3. Install**

Run the auto-installer script and follow the prompts:

```
bash install.sh
```

This script will first check if Anaconda or Miniconda already exists on your system. It will download and install Miniconda if necessary. It will also add conda to your `$PATH`, which is to say that it will tell your computer where to look to use the conda software. Then, it will use conda to create a SeidarT environment and install the rest of the requirements into that environment. Finally, it will compile the Fortran code in this repository.

You should now be able to move on to the Getting Started section.

**Manual installation instructions**

This is the more involved and less sure-fire installation method, probably best left for intermediate level Unix users and up. The following system dependencies are required:

**Python3, Fortran95, GCC, pip, git, dos2unix, ghostscript, imageMagick**

and additionally, these Python dependencies are also required: *numpy*, *scipy*, *matplotlib*, *mplstereonet* (optional for viewing fabric distributions).

### 1. Get system prerequisites

First, install what you will need to compile the Fortran code. This can be with Homebrew (on OS X) using the commands:

```
brew update
brew install gcc git dos2unix ghostscript imagemagick numpy vtk python pip
```

and via **apt** (on Linux) with:

```
sudo apt update
sudo apt install gcc-10 git dos2unix ghostscript imagemagick \
python3.8 python3-numpy python3-vtk python3-pip
```

### 2. Install Miniconda

[Anaconda](#) will work as well, but miniconda is a smaller initial installation, and will only install what you need.

### 3. Get Python prerequisites

From a Terminal window in which the `conda` command is accessible, run the following commands:

```
conda create -n seidart python=3 pip git ghostscript imagemagick \
numpy matplotlib scipy pyvtk vtk
conda activate seidart
pip install mplstereonet
```

### 4. Get the software

```
cd /path/to/parent/directory
git clone git@github.com:sbernSEN/SeidarT.git
cd SeidarT
```

### 5. Run the installer

```
bash manual_install.sh
```

### 6. Update PATH

When the compilation is finished, we can add the folder to the path directory and the python path directory. Currently, this software is supported with bash so append the following lines to the `~/.bashrc` file if using Ubuntu:

```
export PATH=$PATH:/path/to/SeidarT/bin
export PYTHONPATH=$PYTHONPATH:/path/to/SeidarT/bin
```

and if Python 2 is the default version, create an alias by adding this line to your aliases (either in `~/.bashrc` or `~/.bash_aliases`)

```
alias python=python3
```

## Note

Notes for Macintosh users:

Depending on the OS release (El Capitan, High Sierra, Mojave, etc.) and whether you have Anaconda installed appending a path might be different. Anaconda may set aliases so troubleshooting on a Mac can be cumbersome. Before editing the `/etc/path`, `.bash_profile`, `.profile`, or `.bashrc` files, it is a good idea to create a backup especially if you are not familiar with either or any of those files. To do this copy the original to a new name. For example,

```
cp <location/of/path/definitions> <location/of/path/definitions>_original
```

that way you can always revert back to the working script.

There are a variety of ways to edit the documents but for simplicity change directories to the home folder:

```
cd ~
```

and input into the command line:

```
sudo nano .bashrc
```

and append the `export PATH=...` lines at the bottom. Save and close the file (`CTRL+X`, then `Y` and enter) then check to make sure it is included in the path:

```
. ~/.bashrc  
echo $PATH  
echo $PYTHONPATH
```

## Folder structure

Here we describe the folders you may need to use while working with the software.

- **bin**  
Contains the active Python and Fortran codes used in calculating and displaying the wave propagation.
  - **docs**  
Repository for html documentation.
  - **EXAMPLES**  
Hosts images and other files used in the tutorial. Also contains a shell script that can help with bookkeeping.
  - **materials**  
Library for definitions and subroutines.
- 

- `genindex`
- `search`

## Getting Started

### General workflow

Using SeidarT follows a relatively simple workflow.

1. You need two or three files to start:
  - A 2D image saved in *png* format.
  - A *csv* file listing the X,Y,Z coordinates of receivers for your survey
  - If your material is anisotropic, you need a file in the format delimited file specifying the Euler angles for a number of crystals, with one triplet per line. See an example orientation file and/or generate one using the `orientation_tensor` function.
1. Generate a project file (using `prjbuild`) and edit that text file to set up your survey.
2. Create files describing the radar or seismic source (`sourcefunction`).
3. Choose the style of survey you want to do [single shot, common offset, common midpoint, or (in development) polarimetric] and run the calculations.



4. For single shot, you can create an animation of the wave propagation (im2anim for 2D or vtkbuild for 2.5D).
5. Display your results as radar- or seismograms, or wiggle plots. You can also save the timeseries-receiver data in a csv file for further processing in different software.

Output from the seismic model is m/s and from the radar model is

## Files to generate or edit

- *PNG image (.png)*

This defines the geometry of your model system. A good starting size is 50 to 500 pixels for each direction. Each RGB color represents a different material, so **the file must be saved with no antialiasing**. Typically each pixel represents the same distance in x and z (in meters). To get started on a new project create a new folder and save the image to the folder. From the command line, change directories to the project folder then enter the following:

```
prjbuild -i /path/to/geometry/image.png -p project_filename.prj
```

Below, we describe the *prj* file structure and how to edit it.

- *receiver locations (text file, commonly receivers.xyz)*

A comma separated list of X,Y,Z coordinates (one set per line, with X,Y,Z as the first line) for receiver locations. Can use pixels, but more typically meters as the units.

- *project file (.prj)*

This file is the heart of the software. It defines domain values, material properties, and survey conditions for electromagnetic and seismic runs. Here, we identify what each line means and which to edit. All lines with # are comments. Bold text indicates a line the user should edit.

**Table: .prj File lines and their meanings**

Line	Description
I,fill2.png	The image file associated with this .prj file.
D,dim,2	<b>Choose either 2D or 2.5D.</b> 2.5D is the 2D image extruded in the y-direction.
D,nx,240	Read from the image file. Do not change.
D,ny,1	<b>Number of pixels in the extruded direction if using 2.5D.</b>
D,nz,50	Read from the image file. Do not change.
D,dx,1	<b>Number of meters each pixel represents in the x direction.</b>
D,dy,1	<b>Number of meters each pixel represents in the y direction.</b>
D,dz,1	<b>Number of meters each pixel represents in the z direction.</b>
D,cpml,20	<b>Thickness of absorbing boundary layer. A typical value is 20.</b>
D,nmats,3	Read from the image file. Do not change.
D,tfile,	An attenuation processing value that is not yet implemented.

Line	Description
M,1,icelh,98/19 7/178, -10,2,910,0,0,T RUE, test.ang	One comma-separated-values line per material (per color in the model image). <b>User should change/add the material name (see list), temperature (in degrees Celsius), density, porosity, water content, whether the material is anisotropic (TRUE or FALSE), and if anisotropic, the name of the anisotropy file. Use a dummy value of 2 for attenuation, recognizing that attenuation is not yet incorporated in the calculations.</b> User should not change the material ID or R/G/B values. Note: Since large density gradients cause numerical instabilities, the density for air must be increased. A value of 400.0 works until a better formulation of the air-rock interface is implemented.
S,dt,	Timestep will be calculated automatically.
S,time_steps,50 0	<b>Decide how many timesteps you want the model to run.</b>
S,x,100	<b>x-coordinate of the seismic source</b>
S,y,0	<b>y-coordinate of the seismic source</b>
S,z,0	<b>z-coordinate of the seismic source</b>
S,f0,60	<b>Frequency of the seismic source</b>
S,theta,0	<b>Inclination of the seismic source (+ is down)</b>
S,phi,0	<b>Angle of seismic source from x-axis in the x-y plane (+ is counterclockwise when viewed from above)</b>
C,0.0,	Stiffness tensor for each material. User <i>can</i> enter or change this manually if desired. If blank, calculated from materials information in the earlier section.
E,dt,	Timestep will be calculated automatically.
E,time_steps,50 0	<b>Number of timesteps to run the model.</b>
E,x,100	<b>x-coordinate of the radar source</b>
E,y,0	<b>y-coordinate of the radar source</b>
E,z,0	<b>z-coordinate of the radar source</b>
E,f0,1e8	<b>Frequency of the radar source. 10-100MHz is a good range to start.</b>
E,theta,0	<b>Inclination of the radar source (+ is down)</b>
E,phi,0	<b>Angle of radar source from x-axis in the x-y plane (+ is counterclockwise when viewed from above)</b>
P,0.0,	Permittivity tensor for each material. User <i>can</i> enter or change this manually if desired. If blank, calculated from materials information in the earlier section.

- *orientation file*

A delimited file of one entry of Bunge notation Euler angles per line. A typical number of entries is 500 to ensure a smooth data field.

## Examples

The following are small, simple models to introduce the routines.

### ***Shapes model: create a 2.5D animation of wave propagation***



*The model domain for the “shapes” example.*

1. Open a terminal and change directories into the EXAMPLES>shapes folder. Input into the command line:

```
prjbuild -i shapes.png -o shapes.prj
```

2. Using a text editor to modify shapes.prj, add values to the appropriate lines below the header to match the following:

```
I,shapes.png
D,dim,2.5
D,nx,300
D,ny,3
D,nz,100
D,dx,1
D,dy,1
D,dz,1
D,cpml,20
D,nmats,3
D,tfile,10

# Material definitions:
# number, id, R/G/B, Temperature, Attenuation, Density,
#           Porosity, Water_Content, Anisotropic, ANG_File
M,0,ice1h,120/152/76,-20,2,917,0,0,FALSE,
M,1,water,220/220/220,-20,2,1000,0,0,FALSE,
M,2,granite,255/75/75,-20,2,2700,0,0,FALSE,

# The source parameters for the seismic model
S,dt,
S,time_steps,400
S,x,150
S,y,1
S,z,0
S,f0,60
S,theta,0
S,phi,0

# id, C11, C12, C13, C22, C23, C33, C44, C55, C66, rho
C,0,,,,,,,,,
C,1,,,,,,,,,
C,2,,,,,,,,,

# The source parameters for the electromagnetic model
E,dt,
E,time_steps,400
```

```
E,x,150
E,y,1
E,z,0
E,f0,1e8
E,theta,0
E,phi,0

# id, e11, e22, e33, s11, s22, s33
P,0,,,,,,,,,
P,1,,,,,,,,,
P,2,,,,,,,,,
```

All lines that start with a # are commented lines. For reference, you can use the RGB values to identify the materials.

After filling in the domain and material values, save the .prj file, then run the command:

```
prjrun shapes.prj -m n
```

If you look at the prj file now, you will see that the dt lines and the stiffness and permittivity tensors are filled in. Even though we have all the required fields entered, the model didn't perform any wave propagation calculations because we used the -m n option. The -m n step is required to prepare the file to calculate the sourcefunction (next step).

Before we can do the wave propagation calculations, we need to generate a source function:

```
sourcefunction -p shapes.prj -S gauss0 -m b -a 10
```

Now you can run the model (both seismic and electromagnetic):

```
prjrun shapes.prj -m s
prjrun shapes.prj -m e
```

When that is finished let's build the vti files to view the wavefield. Starting with the seismic wavefield, enter:

```
vtkbuild -p shapes.prj -c Ex -n 20
```

This should produce a set of vti files identical to the ones in the folder "shapes\_ex1\_vti". This is a short animation, but the steps for a larger one are the same.

When that is finished let's build the GIF to view the wavefield. Starting with the seismic wavefield, enter:

```
im2anim dipping_bed.prj -c -n 10 Vx -f 30
```

then repeat when the command line returns but change the '-c' option to 'Vz', 'Ex', or 'Ez'. Alternatively, you could change the frame rate (-f) to a higher or smaller number to adjust the speed of the GIF. We specified the 'D,write' parameter to 10 which is a little undersampled to view seismograms or radargrams, but creating the GIF takes some time and we don't need that much resolution. If you would like to create the seismograms decrease the write parameter between 2-4.

### ***Fill model: create a single shot seismogram and wiggleplot***



*The model domain for the "fill" example.*

#. Open a terminal and change directories into the EXAMPLES>fill folder. We have already created and completed a .prj file for this example ('fill.prj'). As you can see from the materials, we have set it up to be a granite bedrock (blue) overlain by a subglacial (teal for ice) lake (red for water). We have also created a file listing the locations of the source and receiver (spaced every 2 m along the surface) from x=50 to x=190 meters.

1. Run the model:

```
prjrun -p fill.prj -m s
```

2. Build the csv file containing the receiver timeseries:

```
arraybuild -p fill.prj -c Vx -r receivers.xyz
```

3. Display the results:

```
rcxdisplay -p fill.prj -f receiver_array.csv -g 700 -e 0.02
```

4. Play with the gain and exaggeration to vary the image to your liking.

5. To make a wiggleplot:

```
wiggleplot -p fill.prj -g 5 -r receiver_array.csv -c Vx -x 2 -d 1 -n 5
```

### ***Fill model: create a common offset radar profile***

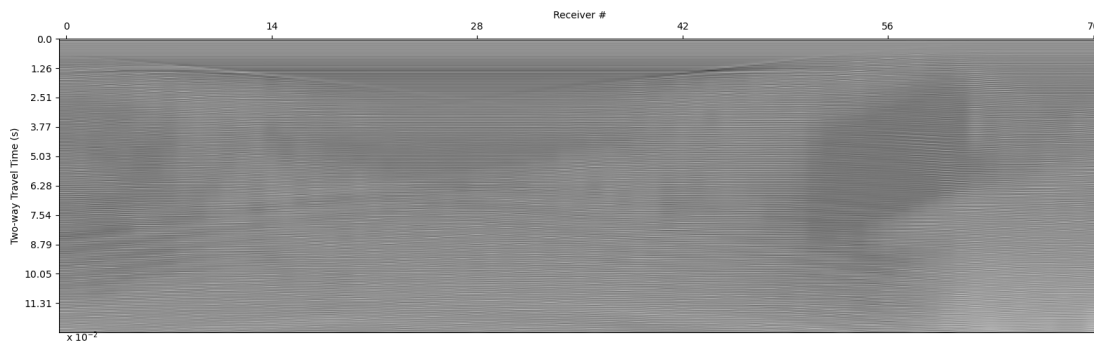
1. Using the same model as before, run it, this time with common offset:

```
common_offset -p fill.prj -o 1 0 0 -r receivers.xyz
```

2. The receiver timeseries are automatically generated, so you can directly display the results:

```
rcxdisplay -p fill.prj -f Ex.co.csv -g 100 -e 0.02
```

You should get something like this:



*A common offset survey across the 'fill' example.*

Play with the gain and exaggeration to vary the image to your liking.

## **Routines and wrappers**

The following is a brief overview of the commands available in the SeidarT environment. More information is available in the Command reference.

## ***Routines***

### ***prjbuild.py***

Constructs a template and assigns default values from a PNG image. See prjbuild.

### ***sourcefunction.py***

Creates .dat files needed to define the source impulse. Does not need to be run if source function files are already present. See sourcefunction.

### ***prjrun.py***

Reads the project file assigns coefficients given that all the required fields are satisfied then runs the specified 2D forward model. You can suppress modeling and edit the stiffness and/or permittivity and conductivity coefficients. Once they are provided in the project file, they won't be computed or overwritten from the material values. If you would like to change the material values and recompute the tensor coefficients, you need to delete the existing tensor coefficients if included in the project file. See prjrun.

### ***im2anim.py***

Create a animation from the model outputs. Currently, this takes some time to run which you can speed up by increasing the 'write' value in the project file. This only takes 2D models, and there are bugs with matplotlib that cause red/blue flashing. See im2anim.

### ***arraybuild.py***

Build the basis for plotting the seismograms or radargrams for the wide angle survey. You can suppress plotting which will return a .csv file. An auto-controlled gain function can be called for better visualization. The receiver locations are given by a text file with the header X,Y,Z. These locations can be given in meters relative to (0,0,0) or in indices. (0,0,0) is top left when viewing the image. See arraybuild.

### ***rcxdisplay.py***

#### **Note**

Originally "codisplay" in legacy code

Display the outputs of the common offset survey. This is also called to display the common midpoint survey. Similar to arrayplot.py, the gain function can be called. See rcxdisplay.

### ***orientation\_tensor.py***

Compute the Euler angles and orientation tensor for a fabric defined by it's trend and plunge angles. The orientation tensor isn't required by the program but it provides useful quantitative information describing the orientation of the fabric. See orientation\_tensor.

## ***Wrappers***

### ***common\_offset.sh***

This is a wrapper that simulates a common offset survey. The receiver .xyz file is input to give the points of the survey and the source is offset from this location given the offsets for the x, y, and z directions. See common\_offset.

## ***common\_midpoint.sh***

This is similar to the common offset survey but it shifts the source and receiver away from a common midpoint. The midpoint is specified by the source location in the project file. By default the source will be to the viewer's right of the midpoint but to flip the location of the source and receiver, set the midpoint x-value to negative. See `common_midpoint`.

### **Note**

The aspect ratio for the common offset and common midpoint surveys determines the axis exaggeration. This will be updated in the future to be easier to adjust but to change this value edit the line `ax.set_aspect(aspect=??)` in `arrayplot.py` and `codisplay.py` then run the plotting scripts individually not the wrapper scripts.

## **prjbuild**

*generates project text file template*

### **Usage**

```
prjbuild -i IMAGEFILE -p PROJECTFILE
```

### **Inputs**

- -i: .png file of cross sectional view of study site
  - No antialiasing
  - Use pixels dimensions proportional to unit dimensions
  - Start with small dimensions (150x150, 200x500), as larger dimensions quickly become more time intensive
- -p: filename, including .prj extension

### **Outputs**

- Project file to fill in based on site characteristics

## **sourcefunction**

*Creates .dat files needed to define the source impulse. Does not need to be run if files are already present.*

### **Usage**

```
sourcefunction [-h] -p PRJFILE -S SOURCETYPE -m [s e b] -a AMPLITUDE
```

### **Inputs**

- -p: PRJFILE  
Path to the .prj file, which must have dt and permittivity and stiffness tensors completed (usually via the -m n option of `prjrun`).
- -S: Source type  
Specify the source type. Available wavelets are: `gaus0`, `gaus1`, `gaus2` (gaussian n-th derivative), `chirp`, `chirplet`. (Default = `gaus0`)
- -m: Model type  
Specify whether to construct the source for an electromagnetic or seismic model. `e`: electromagnetic, `s`: seismic, `b`: both
- -a: Amplitude

Input the scalar actor for source amplification. (Default = 1.0)

- -h, --help  
show this help message and exit

### Outputs

.dat files in x, y, z for each model type

## prjrun

*Performs single-shot wave propagation with receiver locations specified later in postprocessing.*

### Usage

```
prjrun -p PROJECTFILE -m [n e s] -a [0 1]
```

### Inputs

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The file path for the project file, completely filled in for the model type used except for permittivity and stiffness coefficients, and dt
- -m [n e s] Which model to run
  - n calculates only timesteps and material tensors, necessary before running sourcefunction
  - e electromagnetic propagation
  - s seismic wave propagation
- -a [0 1] (optional)  
Append/recompute the coefficients to the permittivity and stiffness matrices; 1 = yes, 0 = no; default = 1. Do not recompute if you have made manual changes to the matrices in the prj file.

### Outputs

.dat files equal in number to the number of time steps specified in the .prj file.

## common\_offset

*A wrapper script to create a common offset survey. The survey is along the x-direction, but can be extended to other directions.*

### Usage

```
common_offset -p PROJECTFILE -o X Y Z -r RCXFILE -s [OPTIONAL] -c VALUE [optional]
```

### Inputs

- -p PROJECTFILE, --project PROJECTFILE  
Project file path
- -o X Y Z, --offset X Y Z  
Source-receiver offset distance for all three directions (meters). Even in 2D calculations, you must enter three space-separated X Y Z values.
- -r RCXFILE, --receivers RCXFILE  
The coordinate locations of every survey point (meters). The file is comma-delimited in each row, with the first row required to be 'X,Y,Z'
- -s, --seismic  
(OPTIONAL) Specifier to run seismic common offset. Default is electromagnetic.



- -c, --cores

For parallel computation, specify the number of cores.

### Outputs

Three CSV files, containing Ex, Ey, and Ez (or Vx, Vy, Vz) values, with time series in columns of the source locations.

## common\_midpoint

*creates receiver setup by defining final and middle locations, and receiver spacing across surface*  
*UNDER CONSTRUCTION*

### Usage

```
common_midpoint -p PROJECTFILE -t [X] -o [X] \
-d [distance between each receiver] [-s] [-p]
```

### Inputs

- -p PROJECTFILE, --prjfile PROJECTFILE  
The project file path
- -t VALUE, --total VALUE  
The terminal distance between the source and receiver
- -o VALUE, --offset VALUE  
The initial source and receiver offset from the midpoint given in (+/- meters). A negative value means that the source is left of the midpoint. The total source and receiver distance is  $2 \times \text{offset}$ .
- -d VALUE, --delta VALUE  
Source and receiver step length (meters); total distance between the source and receiver is  $2 \times \text{delta} \times i + 2 \times \text{offset}$ .
- -s, --seismic  
(OPTIONAL) Specifier to run seismic common offset
- -p, --plot  
(OPTIONAL) Show plot. Default is none

### Outputs

## orientation\_tensor

*Wrapper to generate the Euler angles for the plunge and trend then plot the results*

### Usage

```
orientation_tensor -o ANGFILE -n VALUE -P VALUE -t VALUE -a VALUE -A VALUE -S
```

### Inputs

- -o, --outputfile  
Specify the file to save the outputs
- -n VALUE, --npts VALUE  
Total number of grains in synthetic sample
- -P VALUE, --plunge VALUE  
Plunge angle in degrees for center of single pole
- -t VALUE, --trend VALUE

Trend angle in degrees for center of single pole

- -a VALUE, --anglemin VALUE  
Minimum angle deviation from center of single pole
- -A VALUE, --anglemax VALUE  
Maximum angle deviation from center of single pole
- -S, --suppress\_plotting  
(OPTIONAL) Suppress all plotting

### Outputs

- -o FILE, --outputfile FILE  
This will produce a delimited file, the name of which you can enter on the materials line(s) in the prj file.

## arraybuild

*generates a csv file (receiver\_array.csv) with the timeseries in columns by receiver*

### Usage

```
arraybuild [-h] -p PROJECTFILE -r RCXFILE [-i INDEX] -c CHANNEL
```

### Inputs

- -h, --help  
show this help message and exit
- -p PROJECTFILE, --prjfile PROJECTFILE  
The project file path
- -r RCXFILE, --rcxfile RCXFILE  
The file path for the text file of receiver locations
- -i INDEX, --index INDEX  
Indicate whether the receiver file contains coordinate indices or if these are the locations in meters. Default (0 - meters)
- -c CHANNEL, --channel CHANNEL  
The channel to query.  
For radar: Ex, Ey, or Ez  
For seismic: Vx, Vy, Vz, S1, S2, S3, S4, S5, S6  
S# represents the stress tensor:
  - S1 - sigma\_11
  - S2 - sigma\_22
  - S3 - sigma\_33
  - S4 - sigma\_23
  - S5 - sigma\_13
  - S6 - sigma\_12

### Outputs

CSV of amplitude values for all pixels surveyed (*receiver\_array.csv*)

## rcxdisplay

*Plot the amplitude timeseries based on a csv matrix*

**Usage**

```
rcxdisplay -p PROJECTFILE -f DATAFILE -g VALUE -e VALUE -s [OPTIONAL]
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The file path for the project file.
- -f DATAFILE, --file DATAFILE  
Path to the csv file with receiver timeseries data, commonly receiver\_array.csv or [Vx Vy Vz Ex Ey Ez].co.csv. See arraybuild for how to generate this file for single shots.
- -g VALUE, --gain VALUE  
Gain (smoothing length). Possible values are 1 through the number of timesteps.
- -e VALUE, --exaggeration VALUE  
Vertical exaggeration. Set the aspect ratio between the x and z axes for plotting. Default is 0.5
- -s VALUE, --seismic VALUE  
(OPTIONAL) Whether the model is seismic rather than electromagnetic
  - A nonzero value sets this flag to True (i.e. mark as a seismic model)
  - A value of 0 sets this (explicitly) to False (i.e. mark as an electromagnetic model, this is the default)

**Outputs**

Plot of amplitude across area surveyed.

**wigggleplot**

*Plot single, and select handful of amplitudes at specified X locations and all Z locations*

**Usage**

```
wigggleplot -p PROJECTFILE -r DATAFILE -g VALUE -x VALUE -d VALUE -n VALUE \
-c [Ex Ey Ez Vx Vy Vz]
```

**Input**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The file path for the project file.
- -r receiver\_array.csv  
The file path for the receiver array data, typically receiver\_array.csv
- -g VALUE, --gain VALUE  
The linear horizontal exaggeration of the amplitude values from the receiver array file.
- -x VALUE, --receiver\_spacing  
The horizontal distance between receivers, in meters, to label the x-axis properly.
- -d, --columns  
The frequency at which columns are pulled for plotting from the csv file
- -n VALUE, --single\_plot\_dist  
The receiver number (column) indicating amplitude values will be plotted in a single-wiggle plot
- -c [Ex Ey Ez Vx Vy Vz], --channel [Ex Ey Ez Vx Vy Vz]  
The channel to query (Vx for the seismic x direction, for example)

**Output**

- Plot of the amplitude value of each pixel in a column
- Plot of the amplitude value of every pixel in every nth column aligned across a graph

**im2anim**

Create animation of the propagation of waves across a 2-dimensional study area

**Usage**

```
im2anim -p PROJECTFILE -c [Ex Ez Vx Vz] -n [steps in time series] \
-d [delay between frames]
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The full file path for the project file
- -c [Ex Ez Vx Vz], --channel [Ex Ez Vx Vz]  
Whether to plot the seismic or radar model in the X or Z direction
- -n VALUE, --num\_steps VALUE  
Interval between time series steps used to create the animation
- -d VALUE, --delay VALUE OPTIONAL  
Delay between frames, default=1
- -a VALUE, --alpha VALUE OPTIONAL  
Change the transparency of the model plotted in the background; default = 0.3. Zero is full transparency, 1 is opaque.

**Outputs**

A GIF animation of the wave propagation.

**vtkbuild**

*Creates a series of 3-dimensional vti files that can be imported into Paraview to generate an animation. (Directly building the animation is planned for a future release.)*

**Usage**

```
vtkbuild [-h] -p PROJECTFILE -c [Ex Ey Ez Vx Vy Vz] -n NUM_STEPS
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The full file path for the project file.
- -c [Ex Ey Ez Vx Vy Vz], --channel [Ex Ey Ez Vx Vy Vz]  
Whether to plot the seismic or radar model in the X, Y, or Z direction
- -n VALUE, --num\_steps VALUE  
The time step interval between the images that are going to be used. n=20 means that 1 out of every 20 images will be used, thus significantly reducing how long it takes to compile.

**implot**

Plot a snapshot of the wavefield in 2D

**Usage**

```
implot -p PROJECTFILE -v VELOCITYFILE
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The full file path for the project file
- -v VELOCITYFILE, --velocity VELOCITYFILE  
The .dat file that corresponds to the velocity in either the x-direction or z-direction (e.g. Vx000400.dat). The corresponding orthogonal velocity file will be loaded as well.

**Outputs**

A png image of seismic or radar return of the selected timestep overlain on the model geometry

## imvector

Plot a snapshot of the vector wavefield in 2D

**Usage**

```
imvector -p PROJECTFILE -v VELOCITYFILE
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The full file path for the project file
- -v VELOCITYFILE, --velocity VELOCITYFILE  
The .dat file that corresponds to the velocity in either the x-direction or z-direction (e.g. Vx000400.dat). The corresponding orthogonal velocity file will be loaded as well.

**Outputs**

A png image of seismic or radar vector field of the selected timestep overlain on the model geometry

## vectoranim

This program builds a gif from the set of image output of the FDTD modeling. The images can be in csv or unformatted Fortran binary, however, the program runs faster to use the latter. To use, ensure the project file is in the same directory as the output files.

**Usage**

```
vectoranim -p PROJECTFILE -n VALUE
```

**Inputs**

- -p PROJECTFILE, --prjfile PROJECTFILE .prj file  
The full file path for the project file
- -n VALUE, --num\_steps VALUE  
The time step interval between the images that are going to be used. Every time step is written to file which means that we can take any equally spaced images to create the gif with an appropriate resolution, time to compute, and file size. For example, n=20 means that every 20 images will be used thus significantly reducing how long it takes to compile the gif.

**Outputs**

A GIF animation of the wave propagation.

## Creating a png file

Geometries for the model domain within SeidarT are initiated with a PNG image. The program identifies unique RGB values, setting material properties for each. For example, if you wanted to define a geometry with ice overlying bedrock, you would create a .png image that is one color for the ice and another for the rock below. Everyone has their preferences to generate images but [GIMP](https://www.gimp.org/downloads/install_help.html) or [Inkscape](http://wiki.inkscape.org/wiki/index.php/Installing_Inkscape) provide free and open software that are more than sufficient.

### Note

When creating a PNG, anti-aliasing must be turned off to avoid color boundary gradients.

Building images in Inkscape has some advantages other than being free. Saving a .svg to pdf allows the user to change the number of pixels and the spatial resolution of the image quite easily. With ghostscript, the command

```
gs -q -dBATCH -dNOPAUSE -sDEVICE=png16m -sOutputFile=<file> -r96 <input_file>
```

will generate a PNG file from a PDF. The resolution `-r` can be varied to change the pixels. In Inkscape, the image pixels can be set in Document Properties. When saving the SVG as PDF, you will be prompted with options, and the value for Resolution for rasterization (dpi): will determine - in order to get the same pixel setting that you set in Inkscape - the value for the `-r` (resolution) option above. If you want to double the resolution, just double this number (i.e. `-r192`).

## Building documentation

This documentation is built with Sphinx 3.5.4 and rst2pdf 0.98.

To build the documentation, ensure you are in the SeidarT conda environment, then install the requirements:

```
pip install sphinx==3.5.4 sphinx_rtd_theme rst2pdf
```

Then, change directory into docsrc/:

```
cd SeidarT/docsrc
```

Then, use the make command to direct Sphinx to build the documentation:

```
make github
```

This will build documentation (both HTML and PDF) and move these items to the folder where GitHub will look to render them (docs/). Additionally, you can specify which documentation is built by using the `make html` or `make pdf` in order to preview these. The outputs of these intermediary commands will be in docsrc/\_build/html or docsrc/\_build/pdf.

Additional information can be found [here](#).

### Note

Note that Sphinx 4.0 and above depreciates a function that rst2pdf relies on to build the PDF file. At this time it is necessary to downgrade to Sphinx 3.5.4 in order to build docs in PDF format.

## Currently available materials

- air
- ice1h

## Units

- soil
- water
- oil
- dry\_sand
- wet\_sand
- granite
- gneiss
- basalt
- limestone
- anhydrite
- coal
- salt

## Units

Units for seismic output are meters per second.

Units for electromagnetic output are volts per meter.

## References

< references will go here >

< a reference list may look like the following >

1. Bernsen, Steven (2021). *SeidarT: seismic and radar toolbox* (unpublished). University of Washington.
2. Gerbi, Christopher (2021). *SeidarT: seismic and radar toolbox* (unpublished). University of Maine.

## Search documentation

Need to look something up?

- **search**