User guide

Downward Continuation to the seafloor of streamer data

Author: Clara Estela Jiménez Tejero.

Developed at BCSI group at ICM-CSIC (Barcelona, Spain).

Abstract

The most widely used tool to extract refraction information from Multichannel Sesimic (MCS) data is the so-called downward continuation technique, which is designed for redatuming streamer field data to the seafloor. In this new virtual configuration, the early refractions transform to first arrivals becoming visible from nearly zero offset, which facilitates identification and use in travel-time tomography.

We present a user friendly open source HPC software for redatuming 2D streamer field data to the sea bottom for any seafloor relief. The main ingredient is the acoustic wave equation used backward in time, allowing first the redatuming of the receivers (or DC_1 step), and after, the redatuming of the sources (or DC_2 step).

As inputs, the shot gathers are required in a series of SU files of less than 2 GB, and the bathymetry at each shot gather in a ascii file. Also, a series of parameters have to be provided through an input file which is read in the execution line. By default, the p-wave velocity model for the column water is considered constant but also XBT data can be provided to build a specific Vp water model. For more details on the physics implemented in the software, see the reference pre-print [1]. This reference is also revealing to know which type of streamer data is a good candidate for downward continuation, and also to understand the validity limits of the DC results.

1 Software requirements

- The DC software presented here is an open source code developed under fortran 90 and HPC architecture built with open MPI.
- Parallel compilation is needed using MPI (mpif90 command installed for compilation and mpirun command installed for execution).
- Seismic Unix tool [2] installed and a minimum of knowledge working with it. It is a open source software necessary to work/convert/visualize seismic binary data files.
- For time consuming reasons, the main requirement is to run it on a cluster environment.

2 Installation

- Type in the terminal, inside the 'src/' folder:
  make

- Also, it can be installed manually:
  - Compilation of the modules first:
    mpif90 -c modules.f90

  - Compilation of all and generation of executable DC_MCS_run:
    mpif90 *.f90 -o DC_MCS_run

- After it is properly installed, include the path in your .bashrc, so that the executable DC_MCS_run (and also de scripts DC_input.sh and DC_output.sh) are accesible from any location:
  export PATH="/home/user/path/to/src:$PATH"

3 Quick start

- Give a value to all the parameters in the input files for the step 1 and step 2 of the DC process, respectively, as specified in section 4.1.
- Provide in input folder:
  - The shot gathers (see section 4.3). It is necessary to convert the data file from SGY to SU, and split the SU file in different partitions of less than 2 GB. To do so, run this script in the input folder and follow the instructions:
    DC_input.sh

  - The bathymetry information at each shot gather in a single ascii file. Navigation has to be included too, if it is not specified in SU headers (see section 4.2).

- For executing the program, two arguments are needed. First one, the name of the parfile where all the parameters are described, and the second one, the number 1 or number 2, for the steap 1 or 2 of downward continuation, respectively:

  - Step 1:
    mpirun -np numtasks DC_MCS_run parfile 1
    (Here 'numtasks' refers to the number of cores which will be used to parallelize the calculation. The most efficient and maximum value for 'numtasks' in this first step is equal to t he number of shot gathers.)
    * As an optional step, before doing step 1, one might try step 0, which consist only on the reading of input data. For running step 0, use number 0 as second argument. This is useful to check if all input parameters and data are given correctly. In output folder one can check results.
    * Step 0 (optional):
      mpirun -np numtasks DC_MCS_run parfile 0

  - Step 2:
    mpirun -np numtasks DC_MCS_run parfile 2
    In this second step, the most efficient and maximum value for 'numtasks', is equal to the number of point gathers. To estimate the number of point gathers:

$$\text{NumPGs} = 1 + \frac{\text{SL} + (\text{NumShots} - 1) \cdot \text{dshots}}{\text{dmodel}} \ , \tag{1}$$

    where 'SL' is the streamer length, 'dshots' is the distance between shot gathers and 'dmodel' the grid resolution (dmodel<dshots).

- If wanting the output shot gathers in a unique file and/or to transform it to SGY format, run this script in the output folder and follow the instructions:
  DC_output.sh

4 Input data

There are three different types of inputs:

- Section 4.1 lists the parameters to include in the input parfiles.

- Section 4.2 shows how to introduce the bathymetry (and perhaps navigation) measured at each shot gather (inside input folder).

- Section 4.3 shows the requirements for the shot gather input data (inside input folder).

We understand that sometimes few shot gathers are missing or not recorded and the same can happen for the bathymetry/navigation file. All these issues have been taken into account, however it is a requirement that the first and last shot gathers specified in bathymetry file (4.2) and in SU data (4.3) are coincident.

4.1 Input parameter file

The input parameter file is a ascii file and its structure consist on different lines, each line containing the specific parameter name, followed by ':' and at least one spacebar followed by the correspondent parameter value:

parameter_1: value_1
parameter_2: value_2
...
parameter_n: value_n

The file can be named as the user want. Also, it is convenient to be familiar with the header structure of the SU files, as information is obtained from the headers, if this is specified in the parameters file (see https://wiki.seismic-unix.org/sudoc:su_data_format). The list of parameters are described in the following lines (all of them might not be necessary):

1. endianness_machine: integer variable
   - It should be set to 0 (little endian) or 1 (big endian).
   - If not included, the value by default is little endian, 'endianness_machine: 0'.
2. endianness_data: integer variable
   - It should be set to 0 (little endian) or 1 (big endian).
   - If not included, the value by default is big endian, 'endianness_data: 1'.
3. input_folder: character variable
   - Path to the folder where the input files are located.
   - Example, folder_input: '/home/user/DCtest/data/input'.
   - Notice that input and output folder can be the same directory. It is the user choice to separate input and output data or place all data in same directory.
4. output_folder: character variable
   - Path to the folder where the output files will be stored.

- Example, folder_output: '/home/user/DCtest/data/output'.
- Notice that input and output folder can be the same directory. It is the user choice to separate input and output data or place all data in same directory.

5. su_file: character variable
   - Name of the binary SU file (or files) which contain the recorded shot gathers.
   - Example: if 'su_file: name_', and 'split_parts: 3', then the program expects to find in the input folder, three files named: name_0, name_1, name_2.
6. split_parts: integer variable
   - Number of partitions in which the SU file is splitted.
   - Each partition should be smaller than 2 GB.
   - To properly split the SU file, use the script 'DC_input.sh' (see section 4.3).
7. byte_shotnumber: Integer variable
   - Number of byte in SU header where the shotID must be read.
   - Example: "byte_shotnumber: 9" if the shotID is read in fldr header.
   - Default value: 5 (reading shotID from tracr).
8. reg_grid: Integer variable
   - It should be set to 0 (non regular grid geometry) or 1 (regular grid geometry).
   - Default value: 1.
9. nav_file: character variable
   - Name of the bathymetry ascii file, which contain the bathymetry information and also might contain the navigation (position of shot gathers).
   - More information in section 4.2.
10. sx_sy_header: integer variable
    - If 'sx_sy_header: 1', the position of the shot gathers are read from the headers of the SU files (referred as 'sx' and 'sy' in the header). Also in this case, the parameter 'scalco' is automatically read in the headers to correctly obtain sx and sy parameters (UTM coordinates, in meters).
    - If not included, the value by default is 'sx_sy_header: 0'. In this case, the position of each shot gather is read from the ascii file (explained in section 4.2).
11. offset_header: integer variable

- If 'offset_header: 1', the position of each receiver for each shot gather is read from the headers of the SU files (referred as 'offset' in the header).
- If not included, the value by default is 'offset_header: 0'. The position of the receivers for each shotagher are calculated as a regular grid using the parameters drec and streamer_depth.

12. offset_unit: integer variable

    - This parameter needs to be included if 'offset_header: 1' and if the offset information in headers is not expressed in meters.
    - If this parameter is not included when 'offset_header: 1', the software understands that header parameter 'offset' is already in meters. By default, its value is 'offset_unit: 1'.
    - It works similar to parameter 'scalco' in the SU headers:
        - If offset_unit > 0, the offset is obtained as: $\text{offset} = \text{offset}^{(\text{header})} \cdot \text{offset\_unit}$
        - If offset_unit < 0 , the offset is obtained as: $\text{offset} = \text{offset}^{(\text{header})} / |\text{offset\_unit}|$

13. TWT_option: integer variable

    - If 'TWT_option: 1', the bathymetry information has to be given as the Two-Way-Traveltime (TWT) in the input file (in seconds).
    - If not included, the default value is 'TWT_option: 0'. In this case, the bathymetry information must be given in meters in the input file.
    - For more details, check section 4.2.

14. reverse_streamer: integer variable

    - If not specified, it is considered by default, 'reverse_streamer: 0'.
    - Choose 'reverse_streamer: 1', if the closest receiver to the shot gathers is the last channel.

15. dt: real variable.
    Time sampling of the shot gathers (seconds).

16. nt: integer variable
    Number of time steps.

17. dshots: real variable

    - Distance between shots (meters).
    - Compulsory if reg_grid: 1.

18. drec: real variable
    Distance between receivers (meters).

19. NumRec: integer variable
    Number of receivers in the streamer line.

20. streamer_depth: real variable
    Depth of the receivers (meters).

21. shot_init: integer variable
    shotID of the first shot provided in the su_files.

22. shot_fin: integer variable
    shotID of the last shot provided in the su_files.

23. shot_depth: real variable
    Depth of the shot gathers (meters).

24. near_offset: real variable
    Distance between the shot position and the closest receiver (meters).

25. dmodel: real variable

    - Space sampling of the p-wave velocity model (meters).
    - If not specified, it is considered by default, 'dmodel: drec'.
    - If specified, it should be smaller than drec and fulfilling the inequality of the Courant-Friedrichs-Lewy stability condition to avoid instabilities in the propagation: $\frac{dmodel}{dt} \geq \sqrt{2}\ \max(\mathrm{V}_p)$.

26. water_velocity: real variable

    - Water velocity model (meters/second).
    - If not included, it is considered by default, water_velocity: 1500.

27. vp_file: character variable

    - Name of the file which contains the XBT data available at each specific shot gather.
    - It needs to be included if user wants to better describe the water column with a realistic velocity model.
    - If 'vp_file: vp.dat', this is an example of a valid content for the file vp.dat:

      1008 vp_1.dat
      5400 vp_2.dat

9821 vp_3.dat

In this example, there is data available for three different shot gathers with shotID: 1008, 5400 and 9821. Each of the files, vp_1.dat, vp_2.txt and vp_3.dat, must contain 2 columns; the first column indicates the depth (meters), and the second column indicates the p-wave velocity at each available depth.

- There are different ways to calculate the p-wave velocity from XBT data (temperature and salinity). As an example, the p-wave velocity is calculated in [1] using the Mackenzie empirical equation [3].
- The software interpolates the given data into the resolution required for the model (dmodel). In the case of not providing data up to the real depth at each shot gather, the software uses the deepest value provided.
- Nevertheless, building a realistic velocity model with XBT data is not a requirement to properly redatume our data from the surface to the bottom of the sea. We show in [1] that using a realistic homogeneous value for the water column, the results are very similar even for deep waters.
- If 'vp_file' parameter is not included, the water column is considered homogeneous using the velocity value specified in the parameter 'water_velocity'.

28. save_gmt: integer variable

    - Activate this parameter as 'save_gmt: 1', to save the shot gathers in ascii gmt format: X(1:NumRec), Y(1:nt), Z=shot gather.
    - If not included, this parameter is not activated by default, 'save_gnuplot_txt: 0'.

29. save_matlab: integer variable

    - Activate this parameter as 'save_matlab: 1', to save shot gathers in ascii matlab format: shot gather(nt,NumRec).
    - If not included, this parameter is not activated by default, 'save_matlab: 0'.

30. shot_step_txt: integer variable

    - Shot periodicity to be saved in ascii files, if 'save_gmt: 1' and/or 'save_matlab: 1'.
    - If not included, the value by default is 'shot_step_txt: 100'. This means, that if 'shot_init: 1000' and 'shot_fin: 2000', the shot gathers which will be kept in ascii format will be written every 100 shots: 1000, 1100, 1200, ..., 2000.

4.2 Bathymetry and navigation information

The bathymetry information of the shot gathers must be given in an ascii file. But depending on the options selected in the parameter input files, the navigation information has also to be provided together with the bathymetry in the same ascii file.

For regular geometry, there are two possibilities:

- If 'TWT_option: 0', the bathymetry information is introduced with the seafloor depth (in meters) measured at each shot gather.
- If 'TWT_option: 1', the bathymetry information is introduced using the Two-Way-traveltime (in seconds) measured at each shot gather.

Therefore there are two options to build the bathymetry/navigation file:

- If 'TWT_option: 0'.

  $\text{shotID}_1 \quad \text{Z}_1(\text{meters})$
  $\text{shotID}_2 \quad \text{Z}_2(\text{meters})$
  ...
  $\text{shotID}_n \quad \text{Z}_n(\text{meters})$

- If 'TWT_option: 1'.

  $\text{shotID}_1 \quad \text{TWT}_1(\text{seconds})$
  $\text{shotID}_2 \quad \text{TWT}_2(\text{seconds})$
  ...
  $\text{shotID}_n \quad \text{TWT}_n(\text{seconds})$

For non regular geometry, there are different situations, depending on the activation or not of the parameters 'sx_sy_header' and 'TWT_option'.

- If 'sx_sy_header: 0', the navigation (in UTM coordinates) has to be included in the ascii file.
- If 'sx_sy_header: 1', the navigation (in UTM coordinates) is extracted from the SU headers and don't need to be included here.
- If 'TWT_option: 0', the bathymetry information is introduced with the seafloor depth (in meters) measured at each shot gather.

- If 'TWT_option: 1', the bathymetry information is introduced using the Two-Way-traveltime (in seconds) measured at each shot gather.

Therefore there are four options to build the bathymetry/navigation file:

- If 'sx_sy_header: 0', the file must be a 4-column structure:
  - If 'TWT_option: 0'.

    shotID_1 X_1(UTM, meters) Y_1(UTM, meters) Z_1(meters)
    shotID_2 X_2(UTM, meters) Y_2(UTM, meters) Z_2(meters)
    ...
    shotID_n X_n(UTM, meters) Y_n(UTM, meters) Z_n(meters)

  - If 'TWT_option: 1'.

    shotID_1 X_1(UTM, meters) Y_1(UTM, meters) TWT_1(seconds)
    shotID_2 X_2(UTM, meters) Y_2(UTM, meters) TWT_2(seconds)
    ...
    shotID_n X_n(UTM, meters) Y_n(UTM, meters) TWT_n(seconds)

- If 'sx_sy_header: 1'. The file must be 2-column structure:
  - If 'TWT_option: 0':

    shotID_1 Z_1(meters)
    shotID_2 Z_2(meters)
    ...
    shotID_n Z_n(meters)

  - If 'TWT_option: 1':

    shotID_1 TWT_1(seconds)
    shotID_2 TWT_2(seconds)
    ...
    shotID_n TWT_n(seconds)

Notes:

- The units for each shot gather position, UTM, refer to the 'Universal Transverse Mercator' coordinates (in meters).
- The parameters, $\text{shotID}_i > 0$, is the shot number, normally same value than for the parameter 'fldr' in SU header.
- The depth of the seafloor at each shot gather position, Z_i, in meters, can be expressed as positive or negative numbers, the easiest for the user. It is always used as $|Z_i|$.
- The Two-Way-Traveltime (in seconds), it is always a positive number, $\text{TWT}_i > 0$.

4.3 Shot gathers

The shot gathers of a marine survey experiment are recorded in a binary SGY formatted file. The SGY file have to be converted into a SU file first using Seismic unix. A SU file consist on a certain number of shot gathers, where each shot consist on a certain number of channels or traces measured at the receivers. Each trace begins with a header of $60 \cdot 4$ bytes containing the information of experimental parameters, and it is followed by the recorded values at the different time samples (nt). Therefore, the total number of bytes for a SU file is $\text{NumShots} \cdot \text{NumRec} \cdot (60 + \text{nt}) \cdot 4$.

Due to the fact that a realistic seismic line commonly consists of thousands of shot gathers, the SU file must be divided in parts to avoid stack overflow at the time of reading the file in Fortran. This partition does not affect the redatuming results, it is only a requirement so that the complete SU data file can be read in the code. The maximum size allowed for each partition is 2 GB and it should contain a whole number of shot gathers. That is, taking into account that float numbers occupy 4 bytes, the number of bytes occupied by one shot stored in SU format is $\text{Bytes}_{\text{shot}} = \text{NumRec} \cdot (240 + 4 \cdot \text{nt})$. In this way, the maximum number of shot gathers stored in each SU file partition of, for example, 1.9 GB is $1.9 \cdot (1.07 \cdot 10^9)/\text{Bytes}_{\text{shot}}$.

The conversion of a file from SGY to SU format and also the partition of the SU file can be done by the user but we recommend running the script provided at the location 'src/DC_input.sh'. It is important to mention that the partitions are numbered from 0. As an example, if a SU file named as 'su_file' is splitted in 4 parts, the parts must be numbered from 0 to 3: su_file_0, su_file_1, su_file_2 and su_file_3.

5 Output data

- The files located in the output folder after the calculation of step 1 are:
  - DC0 shot gathers (SU format). These shot gathers are obtained after running the code with argument '0' or '1' of execution line. The input shot gathers get tested, rewritten and reorganized (in case of using a reverse streamer) in the

output folder. These SU file names are specified as 'su_DC0_part' and followed by the partition number, from 0 to split_parts-1. These shot gathers are read as input for the calculation of the steps 1 and 2.

- DC1 shot gathers (SU format) [1].
  Shot gather results after step DC: 1. The SU file names are specified as 'su_DC1_part' and followed by the partition number, from 0 to split_parts-1.
- bathymetry_meters.txt (ascii file).
  This file contains the bathymetry interpolated to the grid of the model. It contains 2 columns: x-axis (grid model in meters) and y-axis (bathymetry).
- Vp_model.txt (ascii file).
  This file contains the 2D-Vp water model in case of using XBT data to characterize the water column (in case of using, 'vp_file: 1').

- The files located in the output folder after the calculation of step 2, are:
  - PG1 point gathers (SU format) [1].
    Point gathers obtained from shot gathers 'su_DC1_part'. The SU file names are specified as 'su_PG1_part' and followed by the partition number, from 0 to split_parts-1.
  - PG2 point gathers (SU format) [1].
    Point gathers results after step DC: 2. The SU file names are specified as 'su_PG2_part' and followed by the partition file, from 0 to split_parts-1.
  - DC2 shot gathers (SU format) [1].
    Shot gather results after step DC: 2. The SU file names are specified as 'su_DC2_part' and followed by the partition file, from 0 to split_parts-1. If 'phase_correction: 1", then the SU file names are specified as 'su_DC2_corrected_part', also followed by the partition file, from 0 to split_parts-1.

- The shot gathers in the output folder after step 1 and 2, are obtained being the number 1 the closest receiver to the shot gather, even if the original shot gathers in the input folder are reversed (when the furthest receiver from the shot gather is the number 1). To indicate that the input data are reversed, please select 'reverse: 1' at the input parameter file.

References

[1] Clara Estela Jimenez Tejero, Cesar R. Ranero, Valenti Sallares and Claudia Gras. 'Open source downward continuation to the seafloor of streamer data', arXiv, physics.geo-ph, 2106.00646, 2021.https://arxiv.org/abs/2106.00646.

[2] Murillo, Alejandro E. and J. Bell. "Distributed Seismic Unix: a tool for seismic data processing." Concurrency and Computation: Practice and Experience 11 (1999): 169-187. Seismic Unix tool: https://wiki.seismic-unix.org/doku.php.

[3] K.V. Mackenzie, Nine-term equation for the sound speed in the oceans (1981) J. Acoust. Soc. Am. 70(3), 807-812. https://doi.org/10.1121/1.386920.