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

This is the new version (tag `v2`) of the software for Downward Continuation (DC) of marine MCS field data. The improvements with respect to the previous version (tag `v1`) are:

- **Automated data preparation:** The code now fully automates the preparation of input data. It converts SGY files to SU format and splits the main SU file into smaller parts to ensure compatibility with Fortran. This process is handled by running the code with option `DC: 0`. Input is accepted directly in SGY or SU format. For standard SGY data (big endian) and common machines (little endian), both formats are supported. In rare cases of little-endian SGY data, the user must convert the file to SU format manually and set the data and machine endianness parameter properly.
- **Automated output generation:** The creation of the SU and SGY final files after each step is now automated.
- **One-step MCS DC execution:** The DC process for MCS data, which consists of two steps, can now be performed in a single run using `DC: 3`. This option combines step 0 (data preparation), step 1 (receiver redatuming), and step 2 (source redatuming). Due to potential I/O latency, especially when step 2 depends on outputs from step 1, users may prefer to run each step separately to ensure stable execution.
- **Updated input parameter file:** The input parameter file structure has been revised and improved. Details are provided in 5.1.
- **New case:** When geometry information for sources and receivers is completely absent, the code handles this by creating a default regular geometry (setting parameter `reg_grid`: 1). This generation relies on the supplied experimental parameters, specifically the number of receivers (`NumRec`), and the interdistance between receivers (`drec`) and between shot gathers (`dshots`). Details are provided in 5.1 and 5.3.
- **Test dataset included:** A dataset has been uploaded to Zenodo for testing and validation of the software. Available at zenodo [2], at doi: https://doi.org/10.5281/zenodo.15857062.

1 Introduction

One of the most widely used methods for extracting refraction information from Multichannel Seismic (MCS) data is the downward continuation technique. It is designed to redatum streamer field data to the seafloor. In this virtual configuration, early refractions are transformed into first arrivals that become visible from nearly zero offset, greatly facilitating their identification and use in travel-time tomography.

We present a user-friendly, open-source HPC software package for redatuming 2D streamer field data to the seafloor, adaptable to any bathymetric relief. The core of the method is the acoustic wave equation applied backward in time. The process consists of three main steps, which can either be executed independently — data preparation (`DC: 0`), receiver redatuming (`DC: 1`), and source redatuming (`DC: 2`) — or combined into a single execution (`DC: 3`).

The required inputs include a single SGY or SU file containing the shot gathers, and an ASCII file providing bathymetric information for each shot gather. Additional parameters must be supplied in a user-defined input file, which is read at execution and remains valid across all the steps (`DC: 0`, `1`, `2`, or `3`).

By default, a constant P-wave velocity model is assumed for the water column. However, users can optionally provide XBT data to build a more detailed velocity model. Still, as demonstrated in [1], using a homogeneous water model yields comparable results even in deep-water cases, and is therefore recommended for simplicity and robustness.

For more information on the physical basis of the implementation, refer to the reference preprint [1]. This source also discusses which types of streamer data are suitable for downward continuation, and outlines the validity limits of the method.

2 Software requirements

- The DC software is open-source and developed in Fortran 90 for high-performance computing (HPC) environments using OpenMPI.
- Parallel compilation and execution are required. The system must have the `mpif90` compiler for comilation and the `mpirun` command for execution.
- The Seismic Unix (SU) software package [3] must be installed. A basic understanding of SU is recommended, as it is essential for handling, converting, and visualizing seismic binary data.
- Due to the computational cost in big MCS data sets, it is recommended to run the software on a computing cluster or HPC environment.

3 Installation

- To compile the code, open a terminal and navigate to the `src/` directory. Then run:

  make

- Alternatively, manual compilation can be performed as follows:
  - First, compile the modules:

    mpif90 -c modules.f90
  - Then, compile all source files and generate the executable `DC_MCS_run`:

    mpif90 *.f90 -o DC_MCS_run

- After successful compilation, add the `src/` directory to your `$PATH` by appending the following line to your `.bashrc` file:

  export PATH="/home/user/path/to/src:$PATH"

4 Quick start

- Fill in all the required parameters in the input file, as described in Section 5.1.
- Prepare the input folder with the following:
  - Shot gather data in SGY or SU format (see Section 5.2).
  - Bathymetry information for each shot gather in a single ASCII file. If navigation is not included in the SU headers, it must be provided in this file as well (see Section 5.3).
- To execute the program, two command-line arguments are required:
  1. The name of the input parameter file (`parfile`).
  2. A number indicating the processing step: `0`, `1`, `2`, or `3`. These correspond to:

  - Step 0 (data preparation):
    mpirun -np numtasks DC_MCS_run parfile 0
  - Step 1 (receiver redatuming):
    mpirun -np numtasks DC_MCS_run parfile 1
  - Step 2 (source redatuming):
    mpirun -np numtasks DC_MCS_run parfile 2
  - Step 3 (full process: steps 0+1+2 in a single execution):
    mpirun -np numtasks DC_MCS_run parfile 3

- In these commands, `numtasks` refers to the number of parallel processes (cores) to use. The optimal number depends on the step:
  - For step `DC: 0`, only one task is needed.
  - For step `DC: 1`, the most efficient value is the number of shot gathers.
  - For step `DC: 2`, the most efficient value is the number of point gathers.
- The number of point gathers (`NumPGs`) can be estimated with the following formula:

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

  where `SL` is the streamer length, `dshots` is the distance between shot gathers, and `dmodel` is the spatial grid resolution (`dmodel` $<$ `dshots`).
- When running step `DC: 3`, it is safe to select the maximum number of available tasks on the machine. Unused tasks will simply remain idle.

5 Input data

Three types of input are required to run the DC software:

- Section 5.1 describes the parameters to include in the input parameter file.
- Section 5.2 details the requirements for the shot gather input data, also located in the input folder.
- Section 5.3 explains how to provide the bathymetry (and optionally the navigation) for each shot gather, located in the input folder.

It is understood that in some cases, a few shot gathers or navigation entries may be missing or not recorded. These cases are handled by the code. However, it is a strict requirement that the first and last shot gathers listed in the bathymetry file (Section 5.3) and in the SGY/SU data (Section 5.2) must coincide.

5.1 Input parameter file

The input parameter file is a plain ASCII text file consisting of lines with the following format:

```
parameter_name:  value
```

Example:

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

The file can have any name and remains valid for all processing steps (`DC: 0`, `1`, `2`, or `3`).

It is recommended that the user be familiar with the SU/SGY header structure, as several parameters are extracted from header fields. Below is a list of the available parameters and their usage. Not all parameters are required in every case. The list of parameters are described in the following lines (all of them might not be necessary):

- `input_folder` (string):
  Path to the folder containing the input files.
  Example: `/home/user/DCtest/data/input`

- `output_folder` (string):
  Path where the output files will be stored.The folder is automatically created at the given path if it does not exist yet.
  Default: `output`.

- `data_file` (string):
  Name of the SGY or SU binary file containing the recorded shot gathers.
  Must be located inside input folder.

- `data_format` (integer):
  Automatically inferred by the code. If detection fails, set manually:
  `1` = SGY format, `2` = SU format.
  Optional. (Only if auto-detection fails.)

- `nav_file` (string):
  Name of the ASCII file containing bathymetry and optionally navigation information.
  Must be located inside input folder.
  See Section 5.3 for details.

- `byte_shotnumber` (integer):
  Byte position in the SU header where the shot ID is stored.
  Example: `9` (for `fldr` header).
  Default: `9`.

- `sx_sy_header` (integer):
  Set to `1` to read shot positions from SU headers (`sx`, `sy`); in this case, `scalco` is also read.

If set to `0`, shot positions must be provided in the nav_file (see Section 5.3).
Default: `0`.

- `offset_header` (integer):
  Set to `1` to read receiver offsets from the SU header field `offset`.
  If set to `0`, receiver positions are calculated assuming regular spacing using `drec` and `streamer_depth`.
  Default: `0`.

- `offset_unit` (integer):
  Scaling factor for the offset if `offset_header` is set to `1`.
  Default: `1`.
  Offset is computed as:

$$\text{offset} = \begin{cases} \text{offset}^{(\text{header})} \cdot \text{offset\_unit} & \text{if offset\_unit} > 0 \\ \text{offset}^{(\text{header})} / |\text{offset\_unit}| & \text{if offset\_unit} < 0 \end{cases}$$

- `TWT_option` (integer):
  Set to `1` if bathymetry is given as Two-Way Travel Time (in seconds).
  If set to `0`, bathymetry must be in meters.
  Default: `0`. See Section 5.3.

- `reg_grid` (integer):
  This is a needed parameter in case of no having geometry for sources and receivers. The code will build a regular geometry using the parameters drec, NumRec and dshots.
  If set to `1`, the nav_file should only contain two columns, shotID and bathymetry (see section 5.3).
  Default: `0` (it is not activated by default).

- `reverse_streamer` (integer):
  Set to `1` if channel number 1 corresponds to the furthest receiver.
  Default: `0` (channel 1 is the closest receiver).
  Note: Output data from all DC steps is standardized to `reverse_streamer: 0`.

- `nt` (integer):
  Number of time samples.

- `dt` (real):
  Time sampling interval of the shot gathers (in seconds).

- `shot_init` (integer):
  Shot ID of the first shot from SGY/SU file.

- `shot_fin` (integer):
  Shot ID of the last shot from SGY/SU file.

- `shot_depth` (real):
  Depth of the shot sources (in meters).

- `dshots` (real):
  Distance between consecutive shots (in meters).

- `near_offset` (real):
  Distance between shot position and nearest receiver (in meters).

- `drec` (real):
  Distance between receivers (in meters).

- `NumRec` (integer):
  Number of receivers per shot.

- `streamer_depth` (real):
  Depth of the receivers (in meters).

- `dmodel` (real):
  Spatial sampling interval of the velocity model (in meters).
  Default: equal to `drec`.
  Must fulfill the CFL stability condition:

$$\frac{\texttt{dmodel}}{\texttt{dt}} \geq \sqrt{2} \cdot \max(V_p)$$

- `water_velocity` (real):
  Constant water velocity (in meters/second).
  Default: `1500`.

- `vp_file` (string):
  Name of the file containing XBT data for building a variable water velocity model.
  Optional. If not provided, `water_velocity` is used.
  Format example:

  ```
  1008    vp_1.dat
  5400    vp_2.dat
  9821    vp_3.dat
  ```

  Each file (`vp_n.dat`) must have two columns: depth (m) and velocity (m/s).
  See [1] and [4] for details.

- `save_gmt` (integer):
  Set to `1` to save shot gathers in GMT-compatible ASCII format:
  `X(1:NumRec), Y(1:nt), Z=shot gather`.
  Default: `0` (not activated).

- `save_matlab` (integer):
  Set to `1` to save shot gathers in MATLAB-compatible ASCII format:
  `shot_gather(nt, NumRec)`.
  Default: `0` (not activated).

- `shot_step_txt` (integer):
  Shot interval for exporting ASCII files when using `save_gmt` and/or `save_matlab`.
  Default: `1`.
  Example: If `shot_init: 1000` and `shot_fin: 2000`, a value of `100` will save shots 1000, 1100, 1200, ..., 2000.

5.2 Shot gathers

The shot gathers recorded during a marine seismic survey are typically stored in a binary SGY-formatted file. This software accepts an SGY file as the initial input. It is automatically converted to SU format and then split into smaller files (each less than 2 GB) to ensure compatibility with Fortran memory handling. It is also accepted a single SU file directly.

In rare cases where the SGY data are stored in little-endian format (e.g., vintage datasets), the automatic conversion may fail. In such cases, the user must manually convert the SGY file to SU format before using the software. The SU file would then be provided directly as input.

To manually convert an SGY file into SU format, you can use the segyread command from the Seismic Unix (SU) toolkit and set the different options properly. After conversion, make sure to check that headers (e.g., `fldr`, `sx`, `sy`, `offset`) contain the correct information, as these might be needed by the DC software.

5.3 Bathymetry and geometry/navigation information

Bathymetry information must be provided in an ASCII file `nav_file`. Depending on the parameters `sx_sy_header` and `TWT_option`, navigation data may also need to be included in the same file.

There are different possible configurations for the structure of the bathymetry/navigation file.

- **Case 1:** `sx_sy_header: 0` and `TWT_option: 0`
  The navigation must be provided in the ASCII file. The file must contain four columns:

  shotID_1 X_1 (UTM, m) Y_1 (UTM, m) Z_1 (depth, m)
  shotID_2 X_2 (UTM, m) Y_2 (UTM, m) Z_2 (depth, m)
  $\vdots$
  shotID_n X_n (UTM, m) Y_n (UTM, m) Z_n (depth, m)

- **Case 2:** `sx_sy_header: 0` and `TWT_option: 1`
  The file must still have four columns, but the bathymetry is expressed in TWT (seconds):

  shotID_1 X_1 (UTM, m) Y_1 (UTM, m) TWT_1 (s)
  shotID_2 X_2 (UTM, m) Y_2 (UTM, m) TWT_2 (s)
  $\vdots$
  shotID_n X_n (UTM, m) Y_n (UTM, m) TWT_n (s)

- **Case 3:** `sx_sy_header: 1` and `TWT_option: 0`
  Navigation is read from the SU file headers. The ASCII file must contain two columns with shotID and seafloor depth (in meters):

  shotID_1 Z_1 (depth, m)
  shotID_2 Z_2 (depth, m)
  $\vdots$
  shotID_n Z_n (depth, m)

- **Case 4:** `sx_sy_header: 1` and `TWT_option: 1`
  Navigation is read from the SU file headers. The ASCII file must contain two columns with shotID and TWT (in seconds):

  shotID_1 TWT_1 (s)
  shotID_2 TWT_2 (s)

$\vdots$

shotID$_n$ TWT$_n$ (s)

- **Case: no geometry available at all** When source and receiver geometry is unavailable, set `reg_grid` to 1. In this scenario, the `nav_file` should be structured like cases 3 or 4, and no header information is expected or required in headers of your seismic file.

Notes:

- Coordinates X and Y are given in Universal Transverse Mercator (UTM), in meters.
- The shotID must be a positive integer: shotID$_i > 0$. It usually matches the `fldr` header value in SU files.
- Seafloor depth Z$_i$ can be provided as a positive or negative number. Internally, the absolute value $|Z_i|$ is used.
- Two-Way Traveltime values TWT$_i$ must always be positive: TWT$_i > 0$.

6 Output data

- After execution of the different DC steps, the output files are located in the subfolders `DATA/` and `DATA/PARTS` inside the specified output folder. The main SU files generated are:
  - **DC: 0 – DC0 shot gathers (SU format)**
    These shot gathers are generated after executing the code with argument `0` or `3`. The input shot gathers are checked, rewritten, and reorganized (in case of reversed geometry) in this step. File names follow the pattern:
    Final file: `su_DC0.su`.
    Splitted in parts in output folder: 'DATA/PARTS': `su_DC0_i` where `i = 0,...,N-1`.
    These files are used as input for steps DC: 1.
  - **DC: 1 – DC1 shot gathers (SU format)**
    Final file: `su_DC1.su`.
    Splitted in parts in output folder: 'DATA/PARTS': `su_DC1_i` where `i = 0,...,N-1`.
    These files are used as input for steps DC: 2.
  - **DC: 2 – DC2 shot gathers (SU format)**
    Final file: `su_DC2.su`.
    Splitted in parts in output folder: 'DATA/PARTS': `su_DC2_i` where `i = 0,...,N-1`.

  - **DC: 3 – Full execution (DC0 + DC1 + DC2)**
    When argument `3` is used, all three steps are executed consecutively, generating all above outputs.
  - *Note:* Regardless of the original geometry, the output shot gathers from steps 0, 1, and 2 are stored in a standardized layout: channel number 1 corresponds to the receiver closest to the shot position. If your input data have reversed geometry (i.e., the furthest receiver is channel 1), make sure to set `reverse_streamer: 1` in the input parameter file.

- Additional SU files generated after step DC:2 in output folder: 'DATA/PARTS', include:
  - **PG1 – Point gathers from DC1 (SU format)**
    File names: `su_PG1_i`, where `i = 0,...,N-1`.
  - **PG2 – Point gathers from DC2 (SU format)**
    File names: `su_PG2_i`, where `i = 0,...,N-1`.

- Common files generated after DC steps 1, 2 (or 3) include:
  - **Bathymetry files in `BAT/` folder:**
    * `bat_model.dat` — Interpolated bathymetry on the model grid. Two columns: X position (m) and depth (m).
    * `bat_shots.dat` — Bathymetry at shot positions interpolated to the model grid. Two columns: ShotID and depth.
    * `bat_recs.shot.`*i*`.dat` — Receiver bathymetry per shot. Two columns: receiver number and depth.
  - **Geometry files in `GEOM/` folder:**
    * `geom_shots.dat` — Shot geometry interpolated to the model grid. Two columns: ShotID and X position (m).
    * `geom_recs.shot.`*i*`.dat` — Receiver geometry per shot. Two columns: receiver number and X position (m). The X-axis is adjusted so that the furthest receiver from the first shot is located at `0 m`.
  - `Vp_model.txt` (ASCII):
    2D P-wave velocity model used for the water column (only generated when `vp_file` is provided).

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] Jimenez Tejero, C. E. (2025). Data sample for testing the code: Downward Continuation (DC) for MCS data [Data set].
Zenodo. https://doi.org/10.5281/zenodo.15857062.

[3] 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.

[4] 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.