# **GrowClust**: A Computer Program for the Relative Relocation of Earthquake Hypocenters

## User Guide, Version 1.1

Daniel Trugman
Scripps Institution of Oceanography
Institute of Geophysics and Planetary Physics
UC San Diego, La Jolla, CA, USA
dtrugman@ucsd.edu

February 25, 2017

# Contents

# 1 Overview

Earthquake location is the most fundamental task in seismology, and accurate earthquake locations are essential for improving seismic hazard analyses, studying earthquake triggering processes, and imaging fault structures. However, the catalog positions obtained through routine processing of phase arrival data are often highly scattered and imprecise, detracting immensely from the utility of a location data set. GrowClust is a computer program that applies the method of *Trugman and Shearer* (2017) to relocate earthquake hypocenters using differential times obtained from waveform cross-correlation of pairs of events observed at a set of common seismic stations. Building on previous work in this field (e.g., *Poupinet et al.*, 1984; *Got et al.*, 1994; *Shearer*, 1997; *Waldhauser*, 2000; *Richards-Dinger and Shearer*, 2000; *Lin et al.*, 2007), GrowClust is robust, computationally efficient, and multi-scale in its capacity to handle data sets small and large. The method also includes an internal mechanism for assessing location uncertainties, providing users valuable information about the resolution power of the relocation results. This main focus of this user guide is the use and application of the open-source GrowClust computing program, Version 1.0. I provide here only a brief overview of the key concepts underlying the GrowClust algorithm, while directing users to *Trugman and Shearer* (2017) for a more complete discussion of the algorithm and its implementation.

Locating an earthquake using phase arrival data can be posed as an inverse problem where the objective is to minimize some norm of the residuals $r_k$ between the observed phase arrival times $tt_k$ and predicted arrival times $\hat{tt}_k(m_i)$:

$$minimize \qquad ||r_k|| = ||tt_k - \hat{tt}_k(m_i)||. \tag{1}$$

from the source $(m_i)$ to a set $(k)$ of seismic stations. The predicted arrival time $\hat{tt}_k(m_i)$ depends both on the in-situ velocity structure and hypocentral location of the earthquake $(i)$, which we parameterize here in terms of its hypocentral position $(x_i, y_i, z_i)$ and origin time $T_i$:

$$m_i = (x_i, y_i, z_i, T_i). \tag{2}$$

The location algorithms routinely implemented by seismic networks typically consider each event in isolation, solving an independent inverse problem of the form (1) for hypocentral models $m_i$ of each event $i$.

Earthquake location accuracy is fundamentally limited by a number of factors, including unmodeled perturbations in the in-situ velocity structure, limitations in station coverage, and lack of precision in phase arrival data, which are often picked by necessity from noisy or otherwise emergent waveforms. However, recent advances in computing power have made it practical to apply waveform cross-correlation techniques to extract precise differential travel times,

$$dtt_{ij,k} = tt_{j,k} - tt_{i,k}, \tag{3}$$

for pairs of events $(i, j)$ observed at a common station $(k)$. Given a set of differential travel-time observations, one can recast equation (1) as a new inverse problem, larger in scale, where the new objective is to minimize a chosen norm of the residuals between the observed and predicted *differential* travel times,

$$minimize \qquad ||dr_k|| = ||dtt_{ij,k} - \hat{dtt}_k(m_i, m_j)||. \tag{4}$$

By linking event pairs through differential times in this way, it is possible to jointly invert for the relative

hypocentral locations of larger data sets while mitigating the effects of common-mode errors of unmodeled perturbations in velocity structure along common raypaths. Further improving the relative relocation accuracy is the fact that differential travel times obtained through waveform cross-correlation can be orders of magnitude more precise than the absolute travel times derived from operator phase picks.

The $L2$-norm ("least-squares") solution to the inverse problem posed in equation (4) can be obtained using sparse matrix inversion techniques (*Paige and Saunders*, 1982), though care must be taken to optimize a proper weighting scheme to ensure that the outliers pervasive in seismological data do not have undue influence on the solution and that the matrix inversion routine maintains stability for large-scale or poorly-conditioned problems (*Waldhauser*, 2000). GrowClust takes an alternative approach that requires no explicit matrix inversion, and optimizes against the $L1$-norm of the differential travel-time residuals (Equation 4), improving its robustness to data outliers (*Shearer*, 1997). Users input into GrowClust cross-correlation results – differential times and cross-correlation values – of event pairs observed at common stations. The GrowClust algorithm (detailed in (*Trugman and Shearer*, 2017)) then applies a hybrid, hierarchical clustering algorithm (*Kaufman and Rousseeuw*, 2005) that simultaneously groups events into clusters based on waveform similarity, and relocates each event with respect to its cluster. GrowClust also implements a nonparametric resampling technique (*Efron and Tibshirani*, 1994) to estimate location uncertainties, allowing users to directly examine the resolving power of the output relocation results.

The following sections of this User Guide outline the various procedures necessary to interact with the GrowClust program. The guide assumes that users have ample familiarity with both the relevant seismic data sets and command line operation, though not necessarily with programming in Fortran or any other language. Section 2 of the guide describes the steps needed to compile and run the GrowClust program, including how to set up the algorithm control (.inp) file that specifies GrowClust run parameters. Section 3 provides an overview of the inputs needed to run GrowClust, including the input event list (Section 3.1), station list (Section 3.2), cross-correlation data (Section 3.3), and 1-D velocity model (Section 3.4). Section 4 lists the various file types and formats in which GrowClust outputs relocation results. Section 5 demonstrates the use of GrowClust on the example problem included the source distribution. Finally, Section 6 describes some of the planned future improvements for later versions of GrowClust.

GrowClust is an open-source software package available under the GNU General Public License v.3. The GrowClust source distribution can be downloaded at `http://igppweb.ucsd.edu/~dtrugman`, and I always welcome user suggestions (`dtrugman@ucsd.edu`). If you do chose to use GrowClust in your research, simply cite *Trugman and Shearer* (2017) to direct your audience to a complete description of the algorithm.

# 2 Compiling and Running GrowClust

The GrowClust program is packaged in a source distribution that includes (i) the source codes and a Makefile to ease the compilation process, (ii) a set of example input and output files to run to test your setup, (iii) licensing information (GNU GPLv3), and (iv) this user guide.

## 2.1 Program compilation

GrowClust is written in the programming language Fortran90, and users will need a Fortran compiler to run it. The source code distribution includes a Makefile to simplify the compilation progress. To compile and

link the various source codes, simply open a terminal in the GrowClust directory (or wherever you unpacked the distribution) and navigate to the *SRC/* directory. Then type

» make

or

» make all

This command should create the necessary object files, modules, and program executable *growclust*. Users may need to adjust the compiler options (the FC variable in the Makefile) depending on their computation environment. Also, when testing out personal edits to the source code, it is safest to remove the optimization flag -O from the CFLAGS variable, and replace it with various debugging compiler flags (e.g., -fcheck=bounds). After testing, replace these flags with -O for optimal program efficiency.

## 2.2 Algorithm control file and parameters

The algorithm control file is read by GrowClust upon program initiation. It specifies the various input files, program options, and algorithm parameters required to run GrowClust. The control file must contain all of the following 20 lines, in order, but may be interspersed with comment lines (which are flagged by including a "*" as the first non-blank character). All input lines that contain one or more numerical fields are read in free format. An example control file is provided in the *EXAMPLE/* directory of the source distribution.

### 2.2.1 Line-by-line structure of the control file

L1. **evlist_fmt**
- evlist_fmt: input event list format (see Section 3.1)

L2. **fin_evlist**
- fin_evlist: (input) file name for the event list

L3. **stlist_fmt**
- stlist_fmt: input station list format (see Section 3.2)

L4. **fin_stlist**
- fin_stlist: (input) file name for the station list

L5. **xcordat_fmt**, **tdif_fmt**
- xcordat_fmt: cross-correlation data format (see Section 3.3)
- tdif_fmt: differential time sign convention for event pairs; 12 (for $\text{event}_1 - \text{event}_2$) or 21 (for $\text{event}_2 - \text{event}_1$)

L6. **fin_xcordat**
- fin_xcordat: (input) file name for the cross-correlation data

L7. **fin_vzmdl**
- fin_vzmdl: (input) file name for the velocity model ($Z, V_p, V_s$ format, see Section 3.4)

L8. **fout_vzfine**

– fout_vzfine: (output) file name for the interpolated velocity model (see Section 3.4)

L9. **fout_pTT**

– fout_pTT: (output) file name for the P-phase travel time table (see Section 3.4)

L10. **fout_sTT**

– fout_sTT: (output) file name for the S-phase travel time table (see Section 3.4)

L11. **vpvs_factor**, **rayparam_min**

– vpvs_factor: default $Vp/Vs$ ratio (if $Vs$ is unlisted in velocity model)

– rayparam_min: minimum ray parameter ($P$-phase) for the travel time computations. Negative values default to 1/7.5=0.133 s/km, which should remove the Pn phase for most velocity models.

L12. **tt_dep0**, **tt_dep1**, **tt_ddep**

– minimum, maximum, and spacing for depths (Z-distances, km) in the travel time tables

L13. **tt_del0**, **tt_del1**, **tt_ddel**

– minimum, maximum, and spacing for ranges (X-distances, km) in the travel time tables

L14. **rmin, delmax, rmsmax**

– rmin: minimum cross-correlation coefficient (r) used for the computation of the GrowClust event-pair similarity coefficient (rfactor)

– delmax: maximum station distance used for the computation of the GrowClust event-pair similarity coefficient (rfactor)

– rmsmax: maximum root-mean-square (rms) differential time residual for a proposed cluster merger to be allowed during the relocation algorithm

L15. **rpsavgmin, rmincut, ngoodmin, iponly**

– rpsavgmin: minimum desired average cross-correlation coefficient to keep event pair (set to 0 to keep all in the input cross-correlation data file)

– rmincut: minimum desired cross-correlation coefficient to keep a differential time observation (set to 0 to keep all in the input cross-correlation data file)

– ngoodmin: minimum number of differential time observations with cross-correlation coefficients r $\geq$ rmin for to keep an event pair (set to 0 to keep all in the input cross-correlation data file)

– iponly: (0 or 2) keep both $P$- and $S$-phase differential time, or (1) keep only $P$-phase

L16. **nboot, nbranch_min**

– nboot: number of bootstrap iterations desired for event location uncertainty analysis

– nbranch_min: minimum nbranch (number of events/cluster) to be output in cluster file (L18)

L17. **fout_cat**

– fout_cat: (output) file name for the relocated event catalog (Section 4.1)

L18. **fout_clust**

– fout_clust: (output) file name for the cluster-format relocated event catalog (Section 4.2)

L19. **fout_log**

– fout_log: (output) file name for the program computation log (Section 4.3)

L20. **fout_boot**

– fout_boot: (output) file name for the full bootstrap distribution of event locations (Section 4.4)

### 2.2.2 Other control file notes

- Input file formats (evlist_fmt, stlist_fmt, xcordat_fmt) are detailed in Sections 3.1 through 3.3.
- Details on the velocity model format and travel time table computation are listed in Section 3.4.
- The parameters (rmin, delmax) are used in the computation of the similarity coefficients (rfactor) that rank event pairs based on waveform similarity. They are not, however, applied to remove or otherwise limit observations from the cross-correlation data set in any way.
- In contrast, the parameters (rpsavgmin, rmincut, ngoodmin, iponly) are specific cutoff or threshold values that may be applied by the user to limit the cross-correlation data set. GrowClust will only keep event-pairs and differential time observations that meet these criteria. To apply no thresholding or cutoffs to the input data, set rpsavgmin=rmincut=ngoodmin=iponly=0.
- Details on the output file formats (fout_cat, fout_clust, fout_log, fout_boot) are detailed in Sections 4.1 through 4.4.
- If output files are written to specified directories, those directories must exist prior to computation (GrowClust will not make the directory, only the file).
- The output files (fout_clust and fout_boot) are both optional: set fout_clust=none and/or fout_boot=none to prevent their output.

## 2.3 The grow_params module and auxiliary run parameters

The *grow_params.f90* file in the source distribution is a Fortran90 module that defines the default values of a number of auxiliary GrowClust program parameters that should be suitable for most problems. Advanced users may consider modifying them to suite more specific needs. These auxiliary parameters are of four basic types:

- default array size parameters for GrowClust computations
- auxiliary GrowClust algorithm control parameters
- auxiliary parameters for relative relocation subroutines
- auxiliary parameters controlling bootstrap resampling

The comments included in the file *grow_params.f90* details how each of these parameters work. If you chose to modify any of these parameters, be sure to recompile GrowClust (*make*) so that these changes are reflected in associated module grow_params.mod used by the *growclust* executable.

### 2.4 Running GrowClust

Once the program is compiled and the control file is set to the proper specifications, running GrowClust is relatively simple. All you need to do is call the *growclust* executable from the command line, with the algorithm control file as the first argument. For example, if the executable *growclust* lives in a directory named *SRC/* and the control file *growclust.inp* in the present working directory, (i.e., *./*), then type:

» SRC/growclust  growclust.inp

into the terminal to get the program to run.

When finished, the program will output to the files specified in the control file for further user analysis. During computation, the program will also periodically write computation status reports and runtime statistics to standard (terminal) output. These progress reports are useful to check to see if the program is running correctly, and for debugging purposes, it is sometimes useful to pipe this output to a text file rather than standard output.

## 3 Input Files and Accepted Formats

In addition to the algorithm control file (Section 2.2), GrowClust requires four input files:

- an event list file that uniquely identifies each event to be relocated with an ID number (e.g., CUSPID or EVID), and an initial (e.g., catalog) location
- a station list file that uniquely identifies each station used in the cross-correlation data set with a station name/ID and its latitude and longitude
- a file containing cross-correlation data (differential times and cross-correlation coefficients) for pairs of events observed at common stations
- a file specifying a 1D model of seismic wave velocity as a function of depth, which is used by GrowClust to compute predicted differential times for comparison with the observations

GrowClust is relatively flexible in the accepted formats for these four files, which are outlined in the following subsections.

### 3.1 Event list

This file lists the event IDs, starting (e.g., catalog) locations, and origin times for each event in the cross-correlation data set. Note that for robustness, the GrowClust program will return an error if the cross-correlation data set includes an event that is unlisted in this file. GrowClust currently accepts event lists in four formats, denoted by evlist_fmt = 0, 1, 2 or 3 (line 1 of the algorithm control file). Note that in all cases, GrowClust only explicitly uses the ID numbers, event order, and initial positions (latitude, longitude, and depth) in its internal algorithm. Origin times are adjusted in the output files based on the relocation results, and magnitude information is preserved (though not altered) in these output files. For catalog formats that include event or magnitude types, or catalog RMS or location errors, this information is ignored internally by the algorithm.

#### 3.1.0 efslist format

This format is designed to be backwards-compatible with existing cross-correlation results from the data stored in the rapid I/O event based binary filing system (EFS) used here locally at Scripps Institution of Oceanography. Event lines consist of the following fields:

**eID yr mon day hr min sec evtype mag magtype lat lon dep**

where

- **eID**: event ID number
- **yr, mon, day, hr, min, sec**: catalog origin time
- **evtype, mag, magtype**: event type, magnitude, magnitude type
- **lat, lon, dep**: catalog latitude (decimal degree), longitude (decimal degree) and depth (km)

Lines are read in the fixed format:
(10x, i9, x, i4, x, i2, x, i2, x, i2, x, i2, x, f6.3, x, a2, x, f5.2, x, a1, x, f9.5, x, f10.5, x, f7.3).

#### 3.1.1 phase format

This format is designed to emulate the format commonly used for event lines in text files that list phase arrival data. Each event line consists of the following fields (read in free format):

**yr mon day hr min sec lat lon dep mag eh ez rms eID**

Note that the location error fields (eh, ez, rms) are distinct from the equivalent fields used internally within the GrowClust program, and are ignore upon input.

#### 3.1.2 GrowClust catalog format

This format is identical to that of GrowClust's output catalog, see Section 4.1 for details.

#### 3.1.3 HypoInverse output format

This format is designed to emulate output catalog files created by the HypoInverse program (*Klein*, 2002) commonly used to perform absolute relocation of earthquake hypocenters based on phase arrival data. Each event line consists of the following fields (read in free format):

**eID yr mon day hr min sec lat lon dep rms eh ez mag**

### 3.2 Station list

This file lists the station names and locations (latitude and longitude) for each station used in cross-correlation data set. Station names must be unique, 5-character strings that are consistent with the cross-correlation data set; for robustness, the GrowClust program will return an error if the the cross-correlation data set includes a station that is unlisted in this file. GrowClust accepts several different station list formats, some of which can include network and channel information. However, GrowClust only preserves the station name in its internal subroutines. This means that users may wish to pre-process their cross-correlation data to remove redundant cross-correlation observations from multi-channel instruments. Otherwise, GrowClust will treat cross-correlation results from different channels at the same station as independent observations.

GrowClust currently accepts station lists in two formats, denoted by stlist_fmt = 0 or 1 (line 3 of the algorithm control file).

#### 3.2.0 channel (SEED) format

This format follows the Standard for the Exchange of Earthquake Data (SEED) conventions for naming waveform channels. Each line in the station list corresponds to a distinct station (or channel), with the format:

**netwk stname loc chan lat lon**

where the individual fields denote:

- **netwk**: network name (2-char; not used by GrowClust)
- **stname**: station name (5-char, including blanks)
- **loc**: location code (2-char; not used by GrowClust)
- **chan**: channel name (5-char, including blanks; not used by GrowClust)
- **slat**: station latitude (decimal degrees, f10.5)
- **slon**: station longitude (decimal degrees, f11.5)

Each line is read in the fixed format (a2, x, a5, x, a2, x, a5, x, f10.5, x, f11.5).

#### 3.2.1 station name format

This format is a simplified version of the above that includes only the fields needed by GrowClust:

**stname lat lon**

Lines are read in free format, but users should take care to ensure station names are between 3 and 5 characters in length (excluding blanks).

### 3.3 Cross-correlation data

The fundamental input data needed to run GrowClust are results obtained from waveform cross-correlation of pairs of events observed at common stations. The GrowClust program does not itself perform the requisite cross-correlation operations, though any of a number of established time-domain or frequency-domain approaches (e.g., *Poupinet et al.*, 1984; *Ito*, 1985; *Fremont and Malone*, 1987; *Nadeau et al.*, 1995; *Phillips et al.*, 1997; *Rowe et al.*, 2002; *Schaff and Waldhauser*, 2005) are sufficient. GrowClust accepts several different formats, but all meet the following three requirements:

- Each event within a given event pair is identified with an ID number, and these ID numbers must be consistent with the ones listed in the event list (Section 3.1).
- All cross-correlation data are identified with a station name or ID consistent with those listed in the station list (Section 3.2).
- All cross-correlation data must include both differential travel times and cross-correlation values that quantify the strength (or relative weight) to be applied to the observation.

With regard to this last item, the most straightforward way to run GrowClust is simply use the cross-correlation coefficient (peak value of the normalized cross-correlation function) as the weight value. In principal, however, more advanced users can customize this input value by choosing an alternative weighting scheme tailored to their data set. For example, if one desires to down-weight observations at certain stations or from certain phases, one can modify these weighting values accordingly prior to running GrowClust (which treats all input values as if they were cross-correlation coefficients, regardless of their source).

The algorithm control file parameter tdif_fmt (Section 2.2) specifies the assumed sign convention of the input differential times. Set tdif_fmt = 21 for (event 2 - event 1), or tdif_fmt = 12 for (event 1 - event 2), respectively. Users familiar with the HypoDD program (*Waldhauser*, 2000) should note that the assumed sign convention for that program corresponds to tdif_fmt=12.

GrowClust currently accepts cross-correlation data in two formats, denoted by xcordat_fmt = 0 or 1 (line 5 of the algorithm control file).

### 3.3.0 binary (xcorbin) format

This format is a custom, binary format used here locally at Scripps Institution of Oceanography to efficiently store waveform cross-correlation results. Its format is designed to match the internal structure and data organization of GrowClust's Fortran source codes. (Note that since these files are created using auxiliary Fortran90 scripts, it is important to keep in mind that Fortran90 by default places 4-byte integer pads between write statements within a given binary file). The format is as follows:

- **npair, ndif** (file header 1):
  – (4-byte integer pad)
  – **npair** (4-byte integer): total number of event pairs in file
  – **ndif** (4-byte integer): total number of differential times in file
  – (4-byte integer pad)

- **rminchan, rmingood, delmax, rpsavgmin, iponly, ngoodmin, rminsave** (file header 2 – parameters not used by GrowClust but saved for completeness)
  – (4-byte integer pad)
  – **rminchan** (4-byte real): minimum cross-correlation coefficient used by the external subroutine that selects an optimal coefficient across channels of a given station (used to help produce the binary file)
  – **rmingood** (4-byte real): threshold cross-correlation coefficient for a differential time observation to be classified as "good" by the external subroutine that produced the binary file
  – **delmax** (4-byte real): maximum station distance from an event pair for differential time observation to be classified as good by the external subroutine that produced the binary file
  – **rpsavgmin** (4-byte real): minimum average cross-correlation coefficient for an event pair to be saved in the binary file
  – **iponly** (4-byte integer): a value of 1 implies that only *P*-phase observations are saved in the binary file, while 0 (or 2) implies that both *P*- and *S*-phases are saved
  – **ngoodmin** (4-byte integer): minimum number of differential time observations classified as good for auxiliary subroutine to save an event pair in the binary file
  – **rminsave** (4-byte real): minimum cross-correlation coefficient of differential time observations saved

in the binary file
– (4-byte integer pad)

- **qID1, qID2, eID1, eID2** (event pair arrays, part 1)
  – (4-byte integer pad)
  – **qID1** (4-byte integer array, length npair): serial event number (ranging from 1 to nq) corresponding to the first event in the event pair arrays
  – **qID2** (4-byte integer array, length npair): serial event number (ranging from 1 to nq) corresponding to the second event in the event pair arrays
  – **eID1** (4-byte integer array, length npair): event ID number (not sequential, e.g. a CUSPID or EVID) corresponding to the first event in the event pair arrays
  – **eID2** (4-byte integer array, length npair): event ID number (not sequential, e.g. a CUSPID or EVID) corresponding to the second event in the event pair arrays
  – (4-byte integer pad)

- **index1, index2** (event pair arrays, part 2)
  – (4-byte integer pad) – **index1** (4-byte integer array, length npair): indexing number mapping each event pair to its corresponding starting index within in the phase arrays (see below)
  – **index2** (4-byte integer array, length npair): indexing number mapping each event pair to its corresponding ending index within in the phase arrays (see below)
  – (4-byte integer pad)

- **stname, ipp, tdif, rxcor, dist** (phase arrays: store differential time and cross-correlation data for all event pairs; the event pair arrays index1 and index2 above map event pairs to data in these arrays).
  – (4-byte integer pad)
  – **stname** (12-char array, length ndif): array of station ID names for each observation (note that only the 5-char station name is used by GrowClust)
  – **ipp** (4-byte integer array, length ndif): array identifying the phase of each observation (1 for P, 2 for S)
  – **tdif** (4-byte real array, length ndif): array of differential times (event 2 – event 1) for each observation
  – **rxcor** (4-byte real array, length ndif): array of cross-correlation coefficients (or relative weight) for each observation
  – **dist** (4-byte real array, length ndif): distance in km of each observation (station) from the event pair centroid
  – (4-byte integer pad)

### 3.3.1 text file (dt.cc) format

A text file format identical to the dt.cc format used by the hypoDD relocation program (*Waldhauser*, 2000). Note that the origin time correction field (OTC) field is not used (as catalog differential times are not used by GrowClust), and that GrowClust users can specify the differential time sign convention using the control file parameter tdif_fmt (see Section 2.2).

In this format, differential time observations for each event pair follow event pair header lines. Event pair header lines consist of:

**# eID1 eID2**

followed by a series of phase observation lines:

**stname tdif rxcor phase**

where the above parameter names denote:

- **'#'**: event pair flag (1-char)
- **eID1**: ID number for first event in the pair
- **eID2**: ID number for second event in the pair
- **stname**: station name (5-char, excluding two leading blanks)
- **tdif**: differential travel time for event pair (s)
- **rxcor**: cross-correlation coefficient or relative weight of observation
- **phase**: phase identification string (1-char, 'P' or 'S')

## 3.4 Velocity model

GrowClust uses the velocity model listed in this file to compute the predicted travel times for source-receiver pairs, differencing travel-time predictions for pairs of events (at a common station) to obtain predicted differential times. To do this, GrowClust uses ray-tracing subroutines to construct travel time tables for both $P$ and $S$ phases on an evenly spaced grid in depth and range ($Z$ and $X$), defined by parameters (tt_dep0, tt_dep1, tt_ddep) and (tt_del0, tt_del1, tt_ddel) specified in the control file. The use of these precomputed tables allow for efficient travel time computations during the GrowClust algorithm, where trial event locations are repeatedly adjusted. The travel time table parameters (tt_dep0, tt_dep1) and (tt_del0, tt_del1) should span the range of possible event depths and source-station distances, respectively.

Each line of the velocity model input file corresponds to a depth must have three columns (read in free format):

**DEPTH(km) Vp(km/s) Vs(km/s)**

If the S-phase velocity Vs is unknown, set that field to 0.0 in the input file, and GrowClust will infer the appropriate Vs from Vp and the parameter vpvs_factor specified in the control file. In generating the travel time tables, GrowClust performs linear interpolation to subsample the input velocity model at regular but finely spaced intervals between the input depth points. To preserve "layer-cake" structure, one must craft the input file accordingly. As a simple example, a 3-layer model with Vp of 4.0, 5.0, and 6.0 km/s at layer (top) depths of 0.0, 5.0, and 10.0 km should be input as:

```
0.0   3.0   0.0
5.0   3.0   0.0
5.0   4.0   0.0
10.0  4.0   0.0
10.0  5.0   0.0
```

The first two lines correspond to the top layer, the next two to the second layer, and the last one to the bottom layer (a halfspace). In contrast, a model of

| | | |
|---|---|---|
| 0.0 | 3.0 | 0.0 |
| 5.0 | 4.0 | 0.0 |
| 10.0 | 5.0 | 0.0 |

implies a linear velocity gradient between the depth points 0.0, 5.0, and 10.0 km. In both cases, note that setting the last column to 0.0 means that Vs will be inferred automatically from from Vp (column 2) and the control file parameter vpvs_factor. If the maximum depth in the velocity model is less than tt_dep1 (the maximum depth in the travel time table), a constant layer velocity is assumed to apply in this depth range.

For stations at large distances from the source (beyond the $Pn/Sn$ crossover point) the first arriving phases will be the Moho-diffracted $Pn$ and $Sn$ phases, rather than the direct arrivals, $Pg$ and $Sg$. If unaccounted for, this can present a problem in instances where the observed differential times are from the direct arrivals, $Pg$ and $Sg$, but the predicted first arrivals are from $Pn$ and $Sn$. The most straightforward way to circumvent this issue is to only select stations closer than the $Pn/Sn$ crossover point, where cross-correlation results are typically most reliable. In instances where this is not possible, such as in regions of sparser station coverage, users can define a minimum ($P$-phase) ray parameter, rayparam_min, in the algorithm control file to ensure that predicted arrival times do not include Moho-diffracted phases. Entering a negative number in this field selects a default value appropriate for most velocity models ($1/7.5 \sim 0.133$ s/km), while entering 0.0 prevents this option from being applied. In all cases, corresponding minimum ray parameter for the $S$-phase is computed from rayparam_min and Vp/Vs ratio parameter vpvs_factor.

# 4 Program Output

GrowClust produces four output files that contain different forms of information about the relocation results and program run statistics. The catalog file (Section 4.1) contains the relocated earthquake catalog, including location uncertainties and other event statistics. This file also contains the cluster ID numbers for each event. The cluster file (Section 4.2, optional) contains similar information as the catalog file, but with events grouped by cluster rather than in input event order. This file also contains $(x, y, z)$ coordinates of each event relative to its cluster centroid. The program log file (Section 4.3) documents initial run parameters, tracks cluster mergers during the relocation procedure, and lists summary statistics for the program run. The bootstrap file (Section 4.4, optional) stores the full bootstrap distribution of event locations produced by the nonparametric uncertainty estimation scheme. GrowClust also writes program status updates to the terminal (standard output), which can be useful in tracking the progress of the program as it runs, and diagnosing potential problems in the input or control files (Section 4.0).

## 4.0 Terminal (standard) output

When the GrowClust program first starts up, it reads the algorithm control (.inp) file specified by the user. As it reads parameter and input file names, it checks for potential errors, printing results to the screen for user inspection. If no obvious errors are found, the algorithm proceeds to read the input event list, station list, and cross-correlation data into memory. The first few events, stations, and cross-correlation event pairs are printed to the screen to help the user ensure that these files have been successfully parsed by the program. Next, GrowClust uses the input velocity to derive travel time tables for both *P*- and *S*-phases, which are both stored internally and memory and written to text output files for future use. In the last step of the program's start-up procedure, GrowClust performs a test relocation of the event pair with the highest similarity coefficient, printing results to the screen.

Upon completion of the initial input stage, GrowClust begins its hybrid, hierarchical clustering algorithm that groups and relocates events within similar event clusters. As described in *Trugman and Shearer* (2017), the algorithm first uses the number and strength of the cross-correlation data associated with each event pair to compute event-pair similarity coefficients, storing the results in an internal array of coefficients, rfactor. It then sorts by rfactor, relocating the most similar event pairs first, and processing each event pair in descending order of event pair similarity. As the algorithm proceeds, the program periodically outputs results to the screen, showing the current pair number (as a fraction of total pairs), the two event numbers, the temporary cluster number of this pair (ntree), the number of events in previous clusters of both events (nbranch), and the similarity coefficient of the event pair (rfactor).

When the hybrid clustering and relocation procedure is first completed, the program performs travel time residual computations and saves the relocation results for later output. At this time, the program log file (Section 4.3) is finalized for user inspection. If the user has set the control file parameter nboot to 0, then the program also writes the other output files at this stage. If nboot $> 0$, then the cross-correlation data set is resampled as described in *Trugman and Shearer* (2017), and the program begins its nonparametric uncertainty analyses. As was the case for the initial (unresampled) data set, relocation status updates are printed to the terminal, along with algorithm timing metrics that allow users to track the run-time statistics for each bootstrap resampling iteration.

## 4.1 Relocated catalog file

The relocated catalog file provides an event list with the relocated event positions and origin time, along with other relevant event information. The catalog contains one line per event (nq total), and each line has the following 25 columns:

- **yr, mon, day, hr, min, sec**: relocated origin time (columns 1–6)
- **eID**: event ID (column 7)
- **latR, lonR, depR**: relocated latitude, longitude and depth (decimal degrees and km; columns 8–10)
- **mag** event magnitude (column 11)
- **qID, cID, nbranch**: event serial ID number, cluster serial ID number, total number of events in this cluster (columns 12–14)

- **qnpair, qndiffP, qndiffS**: number of event pairs, *P*-phase differential times, and *S*-phase differential times used to relocate this event (columns 15–17)
- **rmsP, rmsS**: RMS residual differential times for this event for *P*- and *S*-phases (s; columns 18–19)
- **eh, ez, et**: estimated location errors in horizontal (km), vertical (km), and origin time (s; columns 20–22)
- **latC, lonC, depC**: initial (catalog) latitude, longitude and depth (decimal degrees and km; columns 23–25)

Lines are written in fixed format:
(i4, 4i3, f7.3, i10, f10.5, f11.5, f8.3, f6.2, 3i8, 3i6, 2f6.2, 3f8.3, 2x, f10.5, f11.5, f8.3).

All events that appear within the input event list are also listed in the relocated catalog (in the same order), even those that are not relocated by the GrowClust algorithm (e.g., due to insufficient waveform similarity). These events can be easily spotted within the relocated catalog, as they have have values nbranch = 1 and eh= ez = et = -1.000 (the default flag for unrelocated event errors).

## 4.2 Relocated cluster file

The relocated cluster file is an optional output file (set fout_clust="none" to prevent output) that groups events by cluster, and contains both event-specific and cluster-specific information about the relocation results. The cluster file is written in a block format, with each block corresponding to a different cluster of events. Each cluster block contains a cluster header line (one per cluster/block) and a sequence of event lines (one per event within the cluster). Clusters are listed by cluster ID number (ascending order), with lower cluster numbers corresponding to larger clusters (e.g., the cluster with the most events is cluster 1). Clusters with fewer events than the nbranch_min parameter specified in control file (Section 2.2) are not included in the cluster file.

Each cluster/block header has the following 6 fields:

1. **clID**: cluster serial ID number
2. **nbranch**: total number of events in this cluster
3. **latCEN**: latitude of cluster centroid (degrees)
4. **lonCEN**: longitude of cluster centroid (degrees)
5. **depCEN**: longitude of cluster centroid (km)
6. **toffCEN**: origin time offset of cluster centroid (s)

Header lines are written in fixed format (i8, i8, f10.5, f11.5, f8.3, f8.3).

Each event line has the following fields:

- **clID, qID, eID**: cluster serial ID number, event serial ID number, event ID (non-serial, e.g. a CUSPID or EVID)

- **yr, mon, day, hr, min, sec**: relocated event origin time
- **latR, lonR, depR**: relocated event latitude, longitude and depth (decimal degrees and km)
- **xR, yR, zR**: relocated event position, relative to the cluster centroid (km)
- **eh, ez, et**: estimated location errors in horizontal (km), vertical (km), and origin time (s)
- **latC, lonC, depC**: initial (catalog) event latitude, longitude and depth (decimal degrees and km)

Event lines written in fixed format (i8, i9, i10, f6.2, i5, 4i3, f7.3, f10.5, f11.5, f8.3, 3f10.4, 3f8.3, 2x, f10.5, f11.5, f8.3).

Note that because each event belongs to one (and only one) cluster, events are listed at most once in the cluster file. If the parameter nbranch_min $\geq 2$, this file will only contain events that were successfully relocated, and thus there may be events missing from this file that appear in the input event list and relocated catalog file.

## 4.3 Computation log file

The computation log file serves to document the input parameters and provide summary statistics for each GrowClust program run. The log file contains three distinct sections. The first section lists the various input file names, travel time table names, and output file names for the current run, along with the other run parameters specified in the algorithm control file (Section 2.2). Also listed are select auxiliary run parameters stored in *grow_params.mod* (Section 2.3), though these should remain consistent from run to run (unless the user chooses to modify and recompile the module).

The second section of the log file, the GrowLog, documents each cluster-join operation performed over the course of the clustering algorithm (*Trugman and Shearer*, 2017). These join (or merge) operations must meet the various algorithm control criteria listed at the top of the log file, which are specific to each run of the program. Each row of the GrowLog describes corresponds to an event pair whose previously distinct clusters are now merged into a single cluster. Each row contains 14 fields:

1. **CID1**: cluster serial ID number for event 1 in pair (prior to cluster merger)
2. **CID2**: cluster serial ID number for event 2 in pair (prior to cluster merger)
3. **NB1**: total number of events in cluster CID1 (prior to cluster merger)
4. **NB2**: total number of events in cluster CID2 (prior to cluster merger)
5. **CID**: cluster serial ID number for newly merged CID1/CID2 cluster (equal to CID1)
6. **NB**: total number of events in newly merged CID1/CID2 cluster (equal to NB1 + NB2)
7. **NPR**: number of event pairs linking clusters CID1/CID2 (maximum of 10, the best of which are used to relocate)
8. **NDT**: number of differential time observations for these event pairs
9. **dLATC**: difference in latitude between relocated cluster centroids (decimal degrees)

10. **dLONC** difference in longitude between relocated cluster centroids (decimal degrees)
11. **dDEPC**: difference in depth between relocated cluster centroids (km)
12. **dDISTC**: distance between relocated cluster centroids (km)
13. **RMSC**: root-mean-square (RMS) differential travel time residual (s) for the newly merged cluster
14. **RMEDC**: median absolute differential travel time residual (s) for the newly merged cluster

Note that the cluster ID numbers listed here are not the same as the final cluster ID numbers, which are redefined at the end of the program such that the largest cluster has an ID number of 1, the second largest has an ID number of 2, and so on. Following completion of the hybrid relocation and clustering algorithm, summary statistics for the GrowClust run are computed and documented the Run Summary. This final section of the log file of the contains information about the input data set, how it was used, and how well the relocation results fit the observed differential times. Output statistics here include:

1. the total number of events in the initial, input catalog
2. the number of successfully relocated events (a subset of the total input)
3. the total number of input event pairs
4. the number of event pairs used (i.e., linked) in the relocation algorithm (a subset of the total input)
5. the total number of cross correlation data (differential time observations) used: *P*- and *S*-phases
6. the total number of cross correlation data used: *P*-phase only
7. the root-mean-square (RMS) differential travel time residual (observed – predicted, s) for these *P*-phase cross-correlation data
8. the mean differential travel time residual (observed – predicted, s) for these *P*-phase cross-correlation data
9. the total number of cross correlation data used: *S*-phase only
10. the root-mean-square (RMS) differential travel time residual (observed – predicted, s) for these *S*-phase cross-correlation data
11. the mean differential travel time residual (observed – predicted, s) for these *S*-phase cross-correlation data

The final section of the log file also provides a listing of the number of clusters with greater than 2, 5, 10, 20, 50, and 100 events. This information can be easily derived from the catalog and cluster output files, but is also documented in log file for easy user reference.

## 4.4 Bootstrap distribution file

The bootstrap distribution file is an optional output (set fout_boot="none" to prevent output), and is designed to allow users to perform more advanced statistical analyses of the bootstrap distribution of relocated positions for each event. The first line of the file contains two fields: the total number of events

in the input event list (nq), and the total number of bootstrap iterations performed (nboot). Following this file header line, information about the bootstrap distribution of event locations are output in a sequence of discrete event blocks (nq blocks in total). Each event block consists of an event header line (one per event) and a sequence of bootstrap iteration lines (nboot per event).

The event header lines (nq total within the file) contain the following fields:

- **qID, eID**: event serial ID number (range: 1 to nq), event ID (non-serial, e.g. a CUSPID or EVID)
- **yr, mon, day, hr, min, sec**: relocated event origin time
- **latR, lonR, depR**: relocated event latitude, longitude and depth (decimal degrees and km)
- **cID, nbranchR**: cluster serial ID number, total number of events in this cluster
- **nbranchB_mean, nbranchB_min, nbranchB_max**: mean, minimum, and maximum nbranch (events/cluster) for the bootstrap distribution of this event
- **latB_mean, lonB_mean, depB_mean**: mean relocated latitude, longitude and depth (decimal degrees and km) from the bootstrap distribution of this event
- **eh, ez, et**: estimated location errors in horizontal (km), vertical (km), and origin time (s)
    - ∗ These error estimates are the same as those that appear in the catalog file, and correspond to median absolute deviations of the bootstrap distribution of event locations.
- **serrh, serrz, serrt**: estimated location errors in horizontal (km), vertical (km), and origin time (s)
    - ∗ These error estimates differ from those that appear in the catalog file, and correspond to standard errors (i.e., standard deviations) of the bootstrap distribution of event locations.

Lines are written in fixed format: (i8, i10, f10.5, f11.5, f8.3, 2i8, f7.2, 2i8, f10.5, f11.5, f8.3, 6f8.3).

The bootstrap iteration lines (nboot × nq total within the file) contain 5 fields:

1. **latB**: relocated event latitude for each bootstrap iteration (degrees)
2. **lonB**: relocated event longitude for each bootstrap iteration (degrees)
3. **depB**: relocated event depth for each bootstrap iteration (degrees)
4. **cIDB**: cluster serial ID number for each for the current bootstrap iteration
5. **nbranchB**: total number events in this cluster for each bootstrap iteration

Lines are written in fixed format: (f10.5, x, f11.5, x, f8.5, x, i8, x, i8).

# 5 Tutorial Example

The purpose of this section is to walk users through an example run of the GrowClust program. All files referenced in this section can be found in the *EXAMPLE/* directory within the source distribution. The data set used in this tutorial is from the 2012–2015 Spanish Springs, Nevada sequence; see *Trugman and Shearer* (2017) for details. Occurring over a length scale of $\sim$ 3 km, the sequence is not large in its spatial

scale, but the dense seismicity and good station coverage allows for an expansive cross-correlation data set. It should be noted that many of the earthquakes in this sequence are quite small ($M_L \sim 0.0$), so we should not anticipate from the outset to be able to find adequate signal-to-noise (and hence, waveform similarity) to be able to relocate all events.

## 5.1 Data and input files

There are three subdirectories within the *EXAMPLE/* folder: *IN/*, *OUT/*, and *TT/* (users should consider saving a copy of *OUT/* to compare their final results with the output included in the source distribution). The *IN/* sudirectory contains four input files required to run GrowClust:

- events.txt: the event list in "phase" format (format 1, see Section 3.1)
- stations.txt: the station list in "station name" format (format 1, see Section 3.2)
- xcordat.txt: cross-correlation data in the text file format (format 1, see Section 3.3)
- vzmodel.txt: the input velocity model (see Section 3.4)

We can see from the input files that there are a total of 1616 events in the sequence, and the cross-correlation data contain 51 distinct stations. The 11th through 13th columns in the event list (eh, ez, rms) are set to values of 0.000, which is a simple way of denoting that these values are unknown or otherwise unavailable. Event magnitudes may be treated in a similar manner, if unknown, but one may not wish to use 0.000 as the placeholder flag to avoid confusion with true M0 events. One may also notice that the velocity model, vzmodel.txt, has a layered structure, and the S-wave velocity is not explicitly listed (the third column is set to zero).

## 5.2 Setting up the algorithm control file

To run GrowClust, we need to set up an algorithm control file that specifies the names of these four input files, their formats, and a number of other algorithm control parameters (see Section 2.2). An example control file, *ssprings.inp*, is provided in the EXAMPLE/ directory. Lines that begin with a "*" character are interpreted as comments, and are not read by the GrowClust program. Though optional, including lines like these improve the readability of the file and help document the control file parameters for future reference.

The first 6 lines in the control file (excluding comments) specify the file formats and names for the event list, station list, and cross-correlation data. Setting the parameters evlist_fmt=1, stlist_fmt=1, and xcordat_fmt=1 denote use of a "phase" format event list, a "station name" format station list, and "text" format cross-correlation data file, respectively. The parameter tdif_fmt specifies the sign convention of the differential times, and its value of 12 implies that convention in the cross-correlation data file is the differential travel time from event 1 (to a common station) minus the travel time for event 2, as opposed to event 2 minus event 1.

Lines 7–10 lists the various file names associated with the velocity model. The first of these (Line 7, fin_vzmodel), is an input file, while the latter 3 (fout_vzfine, fout_pTT, fout_sTT) are output by the GrowClust program. These files are used internally by GrowClust to computed predicted travel times

(interpolating between adjacent table entries as necessary), but the files are output in text format for user reference.

Lines 11–13 specify the parameters needed to assemble the travel time table, given the velocity model. In line 11, vpvs_factor=1.732 ($\sim \sqrt{3}$) gives the assumed Vp/Vs ratio, if Vs is unlisted in the velocity model file. Setting rayparam_min equal to a negative value invokes GrowClust default value of 0.133 ($\sim 1/7.5$ s/km), which is appropriate to remove Moho-diffracted phases for most typical velocity models. Note that while selecting this parameter is good practice, it is not necessary if the input cross-correlation data set contains only differential time data within the $Pn/Sn$ crossover distance, as is typical. Lines 12 and 13 control the depth and horizontal distance spacing in the travel time tables. The spacings (tt_ddep and tt_ddel) do not need to be set to be extremely fine, as GrowClust uses an interpolation scheme for source/receiver distances that fall between adjacent table entries. The travel time computations are, however, most stable if no extrapolation is required. It is therefore best to ensure that the minimum and maximum depth and distance encompass the input event and station data: tt_dep0 should be less than the minimum source depth and tt_dep1 greater than the maximum source depth, while all stations should lie between a minimum station range of tt_del0 and maximum range of tt_del1.

Line 14 sets parameters used within the GrowClust program to control algorithm execution. Specifically, the parameters rmin and delmax (line 14) control GrowClust's computation of event-pair similarity coefficients, which are used in GrowClust's hybrid clustering and relocation algorithm (*Trugman and Shearer*, 2017), to sort event pairs by waveform similarity. The values in the example input file are nominal (rmin=0.6, delmax=80), but users should select values appropriate to their data set, with the guideline being that rmin and delmax should approximate the minimum cross-correlation coefficient (or data weight) and maximum station distance (in km) of reliable differential time estimates. The parameter rmsmax (line 14) specifies the maximum root-mean square (rms) differential travel time residual permitted for a proposed cluster merger, with values of order 0.2s to 0.3 s being appropriate for most data sets. This parameter ensures that the clustering algorithm does not apply cluster mergers that are not required by the data.

The parameters rpsavgmin, rmincut, ngoodmin, and iponly (line 15) allow users to limit the input cross-correlation data set to only such observations that meet criteria specified by each parameter. In the example control file (*ssprings.inp*), all of these values are set to zero to indicate that no such thresholding should performed. This is a common situation, as often it is desirable to perform external quality control (or other pre-processing) procedures on the cross-correlation data set prior to running GrowClust. The use of these parameters is discussed more thoroughly in section 2.2.

Finally, lines 16–20 set the output file options and file names. The parameter nboot (field one of line 16) specifies the number of bootstrap iterations to use in GrowClust's nonparametric uncertainty estimation algorithm (*Trugman and Shearer*, 2017). It is best to initially set nboot equal to 0 (or another low value) and run the codes as a test, examining the log files and terminal output to ensure everything is set to user specifications. Once satisfied that this is the case, users can then increase nboot to larger value (e.g. 50 or 100), scaling the computation time accordingly, but allowing users to obtain a measure of location uncertainty. The nboot value of 10 used in the example control file is sufficient for demonstration purposes, but should be later increased by the user before undertaking any rigorous uncertainty analyses. Lines 17–20 list the file names for the four output files described in Sections 4.1 through 4.4. The parameter nbranch_min (the second field in line 16) allows the user to specify the minimum number of events per cluster to output in the cluster-format file (line 18, Section 4.2).

## 5.3 Running GrowClust and interpreting file output

Once the control file is set up and the input files are assembled, you are ready to run the GrowClust program. To do so, open a terminal and navigate to the *EXAMPLE/* directory. Then type the path of the *growclust* executable and the control file into the command line, e.g.

» ../SRC/growclust ssprings.inp

One can make this step easier for future work by exporting an environmental variable, e.g., *GROWCLUST*, in one's terminal shell configuration file that specifies the full path to the growclust executable. Then the command syntax simplifies to:

» GROWCLUST ssprings.inp

As described in Section 4.0, the GrowClust program regularly outputs status updates to the terminal during computation. These updates can be useful to track down possible problems in the input or control file, and help the user track the program's progress during repeated bootstrap iterations. The log file (Section 4.3) contains a initial header listing input files used and output files created, along with the set of input and auxiliary algorithm control parameters. At end of the log file, summary statistics for the run are compiled and listed for user inspection. The RMS and mean absolute travel-time residuals (which here are auspiciously small due to the spatial density of the input data set) provide the user a measure of how well the relocation results fit the observations. One can also see, for example, that about half of the total number of events are successfully relocated (793/1616). (Though not explicitly shown in the log file, it turns out that about 95% of the M0.5 and greater events are relocated, including all of the M1.0 and greater events). Also of note is that there is only one major event cluster in the data set, which is typical for a compact sequence such as Spanish Springs.

The relocated catalog output file (along with the optional cluster and bootstrap output files) can used for a number of post-processing purposes, including:

- comparing initial and relocated event positions
- examining location uncertainties as a function of space and time
- grouping events into clusters based on waveform similarity
- tracking magnitude statistics of the relocated events, for comparison with the input data set

The specific formats for these files are discussed in Section 4. Figures 1 and 2 provide map view and cross-section comparisons of initial (left) and GrowClust-relocated (right) event positions. As is apparent in these figures, the relocation results provide a significant sharpening in seismicity, resolving distinct fault structures that are not apparent in the input catalog. Median relative location uncertainties are of order tens of meters.

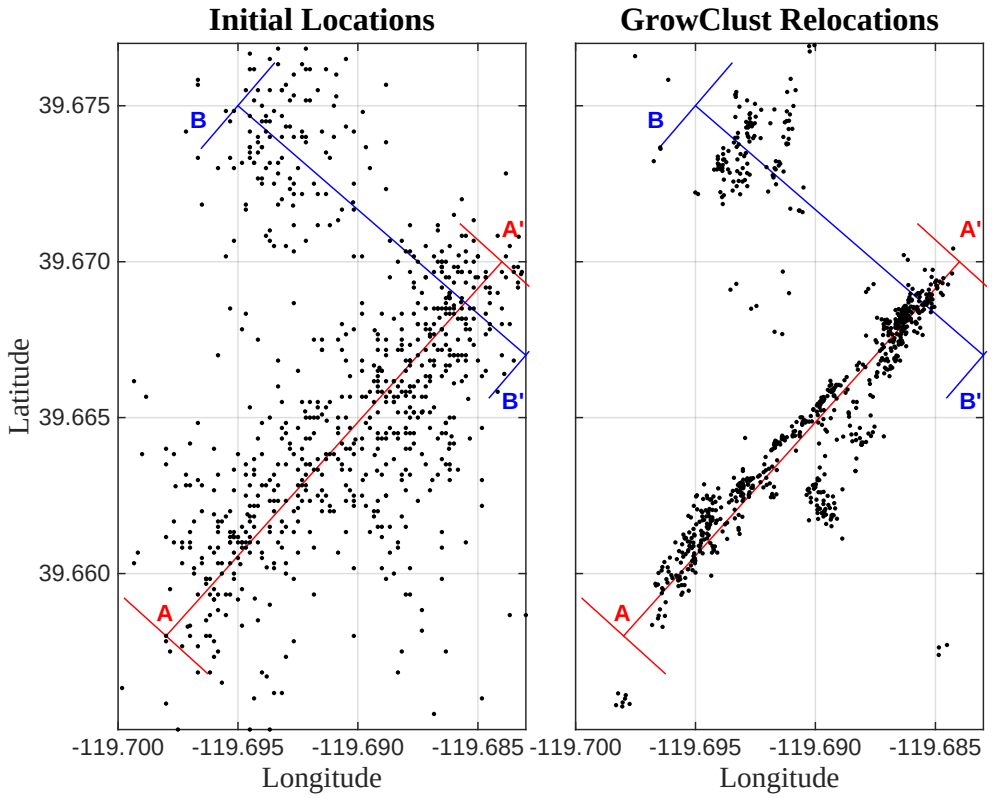


Figure 1: Map view comparison of initial locations (NSL/ANSS catalog, left) and GrowClust relocations (right) for the example data set (the 2012–2015 Spanish Springs sequence, see *Trugman and Shearer* (2017)).

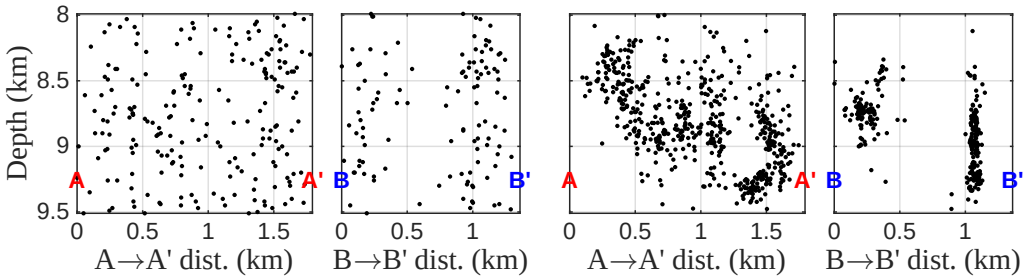


Figure 2: Cross-section view comparison of initial locations (NSL/ANSS catalog, left) and GrowClust relocations (right) for the example data set (the 2012–2015 Spanish Springs sequence, see *Trugman and Shearer* (2017)). Fault-parallel ($A \rightarrow A'$, red) and fault-perpendicular cross-sections ($B \rightarrow B'$, blue) are defined in map view in Figure 1.

# 6 Planned Future Work

The contents of this User Guide describe the GrowClust program in its initial release version (1.0). Because the GrowClust program is open source, I anticipate and welcome user comments and feedback, and intend to incorporate as many of the suggested improvements as is reasonable into future releases of GrowClust. In addition to the anticipated user suggestions, there are several other potential improvements I aim to address in future releases.

First, as currently written, GrowClust 1.0 can only handle 1D velocity models (though variations in wave speed are not limited to "layer-cake" models, and can be an arbitrary function of depth). However, the computational framework underlying GrowClust is well-suited to be generalized to include velocity models with 3D variations in structure, and I hope to incorporate this flexibility in future releases of the codes. Second, I intend in future releases to explore ways of boosting the computational efficiency of GrowClust's nonparametric algorithm for estimating location uncertainties. The bootstrapping procedure is in principle fully parallelizable, and significant performance gains may be obtained by exploiting this fact, especially in computing environments with a large number of available processors.

# Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program (NSFGRFP) under grant number DGE-1144086. Additional support was provided by the United States Geological Survey - National Earthquake Hazards Reduction Program (USGS-NEHRP) grant number G15AS00037-2016-0072.

I first and foremost wish to credit and thank P. Shearer, whose initial work and vision formed the foundation of the GrowClust algorithm upon which I have subsequently built. I also thank to R. Matoza, A. Borsa, Y. Fialko, and D. Kilb for stimulating discussions that helped improve the algorithm's design, and to K. Smith, R. Hatch, C. Ruhl, and R. Abercrombie for their guidance in its application to Nevada seismicity.

# References

Efron, B., and R. J. Tibshirani (1994), *An Introduction to the Bootstrap*, CRC press, Boca Raton, FL.

Fremont, M.-J., and S. D. Malone (1987), High precision relative locations of earthquakes at Mount St. Helens, Washington, *Journal of Geophysical Research: Solid Earth*, *92*(B10), 10,223–10,236, doi: 10.1029/JB092iB10p10223.

Got, J.-L., J. Fréchet, and F. W. Klein (1994), Deep fault plane geometry inferred from multiplet relative relocation beneath the south flank of Kilauea, *Journal of Geophysical Research: Solid Earth*, *99*(B8), 15,375–15,386, doi:10.1029/94JB00577.

Ito, A. (1985), High Resolution Relative Hypocenters of Similar Earthquakes by Cross-Spectral Analysis Method, *Journal of Physics of the Earth*, *33*(4), 279–294, doi:10.4294/jpe1952.33.279.

Kaufman, L., and P. J. Rousseeuw (2005), *Finding groups in data: an introduction to cluster analysis*, Wiley series in probability and mathematical statistics, Wiley, Hoboken, N.J.

Klein, F. W. (2002), User's guide to HYPOINVERSE-2000, a Fortran program to solve for earthquake locations and magnitudes, *Report 2002-171*, U.S Dept. of the Interior, Geological Survey, Reston, VA.

Lin, G., P. M. Shearer, and E. Hauksson (2007), Applying a three-dimensional velocity model, waveform cross correlation, and cluster analysis to locate southern California seismicity from 1981 to 2005, *Journal of Geophysical Research*, *112*(B12), doi:10.1029/2007JB004986.

Nadeau, R. M., W. Foxall, and T. V. McEvilly (1995), Clustering and Periodic Recurrence of Microearthquakes on the San Andreas Fault at Parkfield, California, *Science*, *267*(5197), 503–507, doi:10.1126/science.267.5197.503.

Paige, C. C., and M. A. Saunders (1982), LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares, *ACM Trans. Math. Softw.*, *8*(1), 43–71, doi:10.1145/355984.355989.

Phillips, W. S., L. S. House, and M. C. Fehler (1997), Detailed joint structure in a geothermal reservoir from studies of induced microearthquake clusters, *Journal of Geophysical Research: Solid Earth*, *102*(B6), 11,745–11,763, doi:10.1029/97JB00762.

Poupinet, G., W. L. Ellsworth, and J. Frechet (1984), Monitoring velocity variations in the crust using earthquake doublets: An application to the Calaveras Fault, California, *Journal of Geophysical Research: Solid Earth*, *89*(B7), 5719–5731, doi:10.1029/JB089iB07p05719.

Richards-Dinger, K. B., and P. M. Shearer (2000), Earthquake locations in southern California obtained using source-specific station terms, *Journal of Geophysical Research: Solid Earth*, *105*(B5), 10,939–10,960, doi:10.1029/2000JB900014.

Rowe, C. A., R. C. Aster, W. S. Phillips, R. H. Jones, B. Borchers, and M. C. Fehler (2002), Using Automated, High-precision Repicking to Improve Delineation of Microseismic Structures at the Soultz Geothermal Reservoir, *Pure and Applied Geophysics*, *159*(1-3), 563–596, doi:10.1007/PL00001265.

Schaff, D. P., and F. Waldhauser (2005), Waveform Cross-Correlation-Based Differential Travel-Time Measurements at the Northern California Seismic Network, *Bulletin of the Seismological Society of America*, *95*(6), 2446–2461, doi:10.1785/0120040221.

Shearer, P. M. (1997), Improving local earthquake locations using the L1 norm and waveform cross correlation: Application to the Whittier Narrows, California, aftershock sequence, *Journal of Geophysical Research: Solid Earth*, *102*(B4), 8269–8283, doi:10.1029/96JB03228.

Trugman, D. T., and P. M. Shearer (2017), GrowClust: A Hierarchical Clustering Algorithm for Relative Earthquake Relocation, with Application to the Spanish Springs and Sheldon, Nevada, Earthquake Sequences, *Seismological Research Letters*, *88*(2A), 379–391, doi:10.1785/0220160188.

Waldhauser, F. (2000), A Double-Difference Earthquake Location Algorithm: Method and Application to the Northern Hayward Fault, California, *Bulletin of the Seismological Society of America*, *90*(6), 1353–1368, doi:10.1785/0120000006.