lsforce

Release v1.0.0

Kate E. Allstadt and Liam Toney

Nov 05, 2020

README

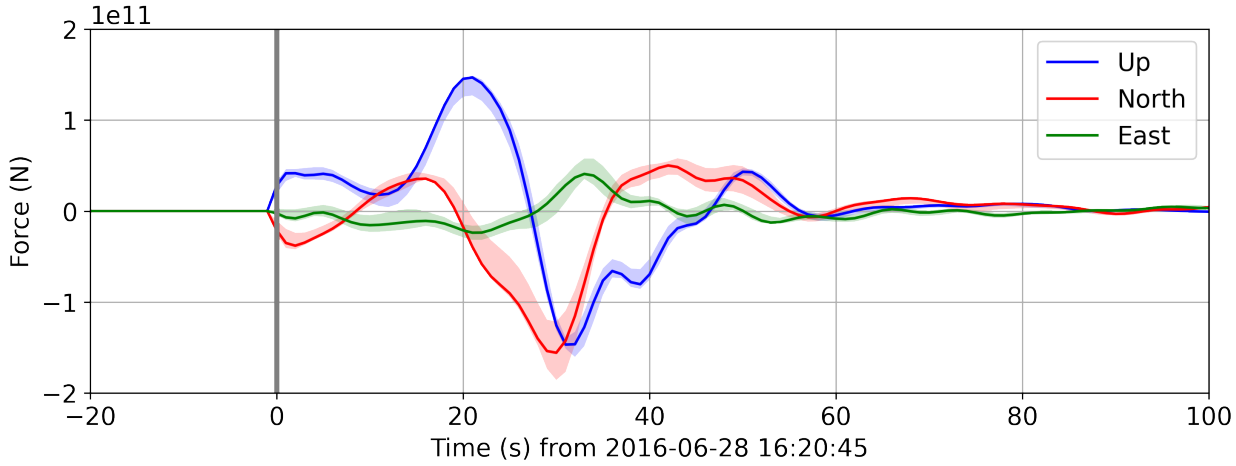
A Python-based seismic force inversion framework for massive landslides

CHAPTER

ONE

LSFORCE

lsforce is a Python-based seismic force inversion framework for massive landslides. The codes can also be applied to other seismic sources well-approximated as a single force (e.g., volcanic eruptions, some glacial events). The library can be used to invert long period (tens to hundreds of seconds) seismic waveforms to estimate a time series vector of single forces that represents the equivalent forces exerted on the Earth by the landslide (see example output figure below).



1.1 Installation

The following has only been tested on macOS Mojave.

Clone this repo and run the installation script, which creates a conda environment named `lsforce` and installs the *lsforce* package into the environment:

```
git clone https://code.usgs.gov/ghsc/lhp/lsforce.git
cd lsforce
bash install.sh  # Or `bash install.sh 1` if you want developer tools as well
```

By default, the Green's functions used by *lsforce* come from the Synthetics Engine (Syngine) hosted by IRIS Data Services. The user can choose from a fixed set of 1D Earth models.

Alternatively, if users prefer to compute Green's functions using a custom model, they can optionally install Computer Programs in Seismology (CPS) via the following:

1. Install GCC with e.g. Homebrew:

```
brew install gcc
```

2. Complete the CPS license form, download the resulting archive, and unzip
3. Move the directory `PROGRAMS.330` to where you'd like to install, then:

```
cd PROGRAMS.330
./Setup OSX40
./C
```

4. Add the executables to `PATH` by adding the following line to e.g. `~/.bash_profile`:

```
export PATH="$PATH:/path/to/PROGRAMS.330/bin"
```

1.2 Documentation

Usage examples for the two currently-supported parameterization methods are given in the three Jupyter Notebooks `example_full.ipynb`, `example_triangle.ipynb`, and `example_lamplugh.ipynb`, which are located in the `notebooks` directory. To open the notebooks, run:

```
conda activate lsforce
jupyter notebook notebooks
```

This will start a Jupyter Notebook server and open a new window or tab in your browser with the interactive notebooks displayed.

To build the documentation, first ensure that you installed the developer tools (`bash install.sh 1`), which are required for documentation building. Then:

```
conda activate lsforce
cd doc
make html
open _build/html/index.html  # macOS command to open file in browser
```

1.3 Testing

Tests are located in the `tests` directory. To run the tests, first ensure that you installed the developer tools (`bash install.sh 1`), which are required for testing. Then:

```
conda activate lsforce
pytest
```

1.4 Citation

Allstadt, K. E., & Toney, L. (2020). lsforce (Version 1.0.0) [Source code]. U.S. Geological Survey Software Release. https://doi.org/10.5066/P9CR20KW

CHAPTER

TWO

LSFORCE PACKAGE

2.1 Submodules

2.1.1 lsforce.lsdata module

class lsforce.lsdata.**LSData**(*st*, *source_lat*, *source_lon*, *remove_response=True*, *skip_zne_rotation=False*)

Bases: object

Class for force inversion data that is an extension of an ObsPy Stream.

st_orig
Original input Stream *st*.

Type Stream

st_proc
Stream rotated into RTZ (radial, transverse, vertical) relative to *source_lat*, *source_lon*.

Type Stream

source_lat
Latitude in decimal degrees of centroid of landslide source location.

Type float

source_lon
Longitude in decimal degrees of centroid of landslide source location.

Type float

Create an LSData object.

Parameters

- **st** (Stream) – Stream object with tr.stats.latitude and tr.stats.longitude defined and station response info attached to each trace in the Stream.
- **source_lat** (*float*) – Latitude in decimal degrees of centroid of landslide source location
- **source_lon** (*float*) – Longitude in decimal degrees of centroid of landslide source location
- **remove_response** (*bool*) – Correct for station response to displacement units. Set to *False* to handle response removal manually at an earlier step.
- **skip_zne_rotation** (*bool*) – If *True*, then the ->ZNE rotation step is skipped. This is a necessary flag if the stations used do not have metadata on IRIS (e.g., for synthetic cases)

plot_data(*equal_scale=True*, *period_range=None*)
Create a record section plot of waveforms in *st_proc*.

Parameters

- **equal_scale** (*bool*) – If *True*, all plots will share the same y-axis scale
- **period_range** (*list or tuple*) – If not *None*, filter the data between *period_range[0]* and *period_range[1]*, given in seconds

Returns Output figure handle

Return type Figure

plot_stations(*region=None*, *label_stations=False*, *gshhs_scale='auto'*)
Create a map showing stations and event location.

Parameters

- **region** (*list or tuple*) – Array of the form [lonmin, lonmax, latmin, latmax] specifying the desired map region in decimal degrees. If *None*, we automatically pick a region that includes the event and stations
- **label_stations** (*bool*) – If *True*, label stations with their codes
- **gshhs_scale** (*str*) – Resolution for coastlines; one of *'auto'*, *'coarse'*, *'low'*, *'intermediate'*, *'high'*, or *'full'*

Returns Output figure handle

Return type Figure

2.1.2 lsforce.lsforce module

class lsforce.lsforce.**LSForce**(*data*, *data_sampling_rate*, *main_folder=None*, *method='full'*)
Bases: object

Class for performing force inversions.

gf_dir
Directory containing Green's functions

Type str

gf_computed
Whether or not Green's functions have been computed for this object

Type bool

filtered_gf_st
Stream containing filtered Green's functions

Type Stream

inversion_complete
Whether or not the inversion has been run

Type bool

filter
Dictionary with keys `'freqmin'`, `'freqmax'`, `'zerophase'`, `'periodmin'`, `'periodmax'`, and `'order'` specifying filter parameters

Type dict

data_length
Length in samples of each data trace

Type int

force_sampling_rate
[Hz] The sampling rate of the force-time function

Type int or float

W
Weight matrix

Type 2D array

Wvec
Weight vector

Type 1D array

jackknife
Dictionary with keys `'Z'`, `'N'`, `'E'`, `'VR_all'`, `'alphas'`, `'num_iter'`, and `'frac_delete'` containing jackknife parameters and results

Type `AttribDict`

angle_magnitude
Dictionary with keys `'magnitude'`, `'magnitude_upper'`, `'magnitude_lower'`, `'vertical_angle'`, and `'horizontal_angle'` containing inversion angle and magnitude information

Type `AttribDict`

G
Design matrix

Type 2D array

d
Data vector

Type 1D array

model
Model vector of concatenated components (n x 1) of solution

Z
[N] Vertical force time series extracted from model (positive up)

N
[N] North force time series extracted from model (positive north)

E
[N] East force time series extracted from model (positive east)

tvec
[s] Time vector for forces, referenced using *zero_time* (if specified)

VR
[%] Variance reduction. Rule of thumb: This should be ~50–80%, if ~100%, solution is fitting data exactly and results are suspect. If ~5%, model may be wrong or something else may be wrong with setup

dtorig
Original data vector

dtnew
Modeled data vector (Gm-d)

alpha
Regularization parameter that was used

alphafit
Dictionary with keys `'alphas'`, `'fit'`, and `'size'` specifying regularization parameters tested

Type dict

Create an LSForce object.

Parameters

- **data** (*LSData*) – LSData object, corrected for station response but not filtered
- **data_sampling_rate** (*int or float*) – [Hz] Samples per second to use in inversion. All data will be resampled to this rate, and Green's functions will be created with this rate
- **main_folder** (*str*) – If *None*, will use current folder
- **method** (*str*) – How to parameterize the force-time function. One of *'full'* — full inversion using Tikhonov regularization (L2 norm minimization) or *'triangle'* — inversion parameterized using overlapping triangles, variation on method of Ekström & Stark (2013)

invert(*zero_time=None*, *impose_zero_start=False*, *add_to_zero=False*, *duration=None*, *jackknife=False*, *num_iter=200*, *frac_delete=0.5*, *alpha=None*, *zero_scaler=2.0*, *zero_start_taper_length=0*, *tikhonov_ratios=1.0, 0.0, 0.0*, *jk_refine_alpha=False*, *save_matrices=False*)
Performs single-force inversion using Tikhonov regularization.

Parameters

- **zero_time** (*int or float*) – [s] Optional estimated start time of real (landslide-related) part of signal, in seconds from start time of seismic data. Useful for making figures showing selected start time and also for the *impose_zero_start* option
- **impose_zero_start** (*bool*) – Adds weighting matrix to suggest that forces tend towards zero prior to *zero_time* (*zero_time* must be defined)
- **add_to_zero** (*bool*) – Adds weighting matrix to suggest that all components of force integrate to zero
- **duration** (*int or float*) – Maximum duration allowed for the event, starting at *zero_time* if defined, otherwise starting from the beginning of the seismic data. Forces after this will tend towards zero. This helps tamp down artifacts due to edge effects, etc.
- **jackknife** (*bool*) – If *True*, perform *num_iter* additional iterations of the model while randomly discarding *frac_delete* of the data
- **num_iter** (*int*) – Number of jackknife iterations to perform
- **frac_delete** (*int or float*) – Fraction (out of 1) of data to discard for each iteration, if frac_delete=1, will do leave a one out error analysis
- **alpha** (*int or float*) – Set regularization parameter. If *None*, will search for best alpha using the L-curve method
- **zero_scaler** (*int or float*) – Relative strength of zero constraint for *impose_zero_start* and *duration* options. Ranges from 0 to 10. The lower the number, the weaker the constraint. Values up to 30 are technically allowed but discouraged because

high *zero_scaler* values risk the addition of high frequency oscillations due to the sudden release of the constraint

- **zero_start_taper_length** (*int or float*) – [s] Length of taper for *impose_zero_start* option
- **tikhonov_ratios** (*list or tuple*) – Proportion each regularization method contributes to the overall regularization effect, where values correspond to [0th order, 1st order, 2nd order]. Must sum to 1
- **jk_refine_alpha** (*bool*) – Refine the alpha parameter used for each jackknife iteration by searching over order of magnitude around the best alpha for the full solution. If *False*, each jackknife iteration will use the same alpha as the main solution (note that this is much faster but can result in some jackknife iterations having depressed amplitudes)
- **save_matrices** (*bool*) – If True, will save the inverted matrices as part of the object (Ghat, dhat, I, L1, L2) in case user wants to do additional alpha searching

plot_angle_magnitude(*xlim=None*, *ylim=None*)

Plot angles and magnitudes of inversion result.

Parameters

- **xlim** (*list or tuple*) – x-axis limits
- **ylim** (*list or tuple*) – y-axis limits

Returns Output figure handle

Return type Figure

plot_fits(*equal_scale=True*, *xlim=None*)

Create a plot showing the model-produced waveform fit to the data.

Parameters

- **equal_scale** (*bool*) – If *True*, all plots will share the same y-axis scale
- **xlim** (*list or tuple*) – [s] Array (length two) of x-axis limits (time relative to zero time)

Returns Output figure handle

Return type Figure

plot_forces(*subplots=False*, *xlim=None*, *ylim=None*, *same_y=True*, *highf_tr=None*, *hfshift=0.0*, *hfylabel=None*, *infra_tr=None*, *infra_shift=0*, *jackshowall=False*)

Plot inversion result.

Parameters

- **subplots** (*bool*) – If *True*, make subplots for components, otherwise plot all on one plot
- **xlim** (*list or tuple*) – x-axis limits
- **ylim** (*list or tuple*) – y-axis limits
- **same_y** (*bool*) – If *True*, use same y-axis limits for all plots
- **highf_tr** (Trace) – Seismic trace with start time identical to start time of the data used in the inversion
- **hfshift** (*int or float*) – [s] Time shift for seismic trace
- **hfylabel** (*str*) – Label used for seismic trace. If not defined, will use station name

- **infra_tr** (*Trace*) – Infrasound trace with start time identical to start time of the data used in the inversion
- **infra_shift** (*int or float*) – [s] Time shift for infrasound trace
- **jackshowall** (*bool*) – If *True* and jackknife was run, will show all individual runs (changes *subplots* to *True*)

Returns Output figure handle

Return type *Figure*

saverun(*prefix*, *filepath=None*, *timestamp=False*, *figs2save=None*, *figs2save_names=None*, *light=True*, *filetype='png'*)
Save a force inversion run for later use.

Parameters

- **prefix** (*str*) – Run name to prepend to all output files
- **filepath** (*str*) – Full path to directory where all files should be saved. If *None*, will use *self.main_folder*
- **timestamp** (*bool*) – Name results with current time to avoid overwriting previous results
- **figs2save** (*list or tuple*) – Figure handles to save
- **figs2save_names** (*list or tuple*) – Names of figures (appends to end)
- **light** (*bool*) – If *True*, does not save seismic data with object to save size
- **filetype** (*str*) – Filetype given as extension, e.g. *'png'*

setup(*period_range*, *syngine_model=None*, *cps_model=None*, *triangle_half_width=None*, *source_depth=0*, *weights=None*, *noise_window_dur=None*, *filter_order=2*, *zerophase=False*)
Downloads/computes Green's functions (GFs) and creates all matrices.

Parameters

- **period_range** (*list or tuple*) – [s] Bandpass filter corners
- **syngine_model** (*str*) – Name of Syngine model to use. If this is not None, then we calculate GFs using Syngine (preferred)
- **cps_model** (*str*) – Filename of CPS model to use. If this is not None, then we calculate GFs using CPS
- **triangle_half_width** (*int or float*) – [s] Half-width of triangles; only used if the triangle method is being used
- **source_depth** (*int or float*) – [m] Source depth in meters
- **weights** (*list or tuple or str*) – If *None*, no weighting is applied. An array of floats with length `st.count()` and in the order of the *st* applies manual weighting. If *'prenoise'*, uses standard deviation of a noise window defined by *noise_window_dur* to weight. If *'distance'*, weights by 1/distance
- **noise_window_dur** (*int or float*) – [s] Length of noise window for *'prenoise'* weighting scheme (if not *None*, *weights* is set to *'prenoise'*)
- **filter_order** (*int*) – Order of filter applied over period_range
- **zerophase** (*bool*) – If *True*, zero-phase filtering will be used

lsforce.lsforce.**Lcurve**(*fit1*, *size1*, *alphas*, *bestalpha=None*)
Plot an L-curve.

Parameters

- **fit1** (*1D array*) – List of residuals
- **size1** (*1D array*) – List of model norms
- **alphas** (*1D array*) – List of alphas tried
- **bestalpha** (*float*) – The alpha value chosen

Returns figure handle

lsforce.lsforce.**find_alpha**(*Ghat*, *dhat*, *I*, *L1=0*, *L2=0*, *tikhonov_ratios=1.0, 0.0, 0.0*, *rough=False*, *range_rough=None*, *int_rough=0.75*, *plot_Lcurve=True*)

Finds best regularization (trade-off) parameter alpha.

Computes model with many values of alpha, plots L-curve, and finds point of steepest curvature where slope is negative.

Parameters

- **Ghat** (*array*) – (m x n) matrix
- **dhat** (*array*) – (1 x n) array of weighted data
- **I** (*array*) – Identity matrix
- **L1** (*array*) – First order roughening matrix. If *0*, will use only 0th-order Tikhonov regularization
- **L2** (*array*) – Second order roughening matrix. If *0*, will use only 0th-order Tikhonov regularization
- **tikhonov_ratios** (*list or tuple*) – Proportion each regularization method contributes to the overall regularization effect, where values correspond to [0th order, 1st order, 2nd order]. Must sum to 1
- **rough** (*bool*) – If *False*, will do two iterations to fine tune the alpha parameter. The second iteration searches over +/- one order of magnitude from the best alpha found from the first round. If *True*, time will be saved because it will only do one round of searching.
- **range_rough** (*tuple*) – Lower and upper bound of range to search over in log units. If *None*, the program will choose a range based on the norm of `Ghat`
- **int_rough** (*float*) – Interval, in log units, to use for rough alpha search
- **plot_Lcurve** (*bool*) – Toggle showing the L-curve plot

Returns

Tuple containing:

- **bestalpha** (float) – The optimal alpha
- **fit1** (1D array) – List of residuals
- **size1** (1D array) – List of model norms
- **alphas** (1D array) – List of alphas tried

Return type tuple

lsforce.lsforce.**readrun**(*filename*)

Read in a saved LSForce object.

Parameters **filename** (*str*) – File path to LSForce object saved using *saverun()*

Returns Saved LSForce object

Return type *LSForce*

2.1.3 lsforce.lstrajectory module

class lsforce.lstrajectory.**LSTrajectory**(*force*, *mass=None*, *target_length=None*, *duration=None*, *detrend_velocity=None*, *zeroacc=None*)

Bases: object

Class for force inversion derived trajectories.

force
Inversion results used to compute this trajectory

Type *LSForce*

mass_requested
[kg] Mass specified

Type int or float

mass_actual
[kg] Mass used (same as *mass_requested* if *target_length* is not specified)

Type int

target_length
[km] Center-of-mass runout length of event, *None* if not specified

Type int or float

jackknife
Jackknifed trajectory results

Type AttribDict

acceleration
[m^2/s] Computed acceleration with Z, E, N components as attributes

Type AttribDict

velocity
[m/s] Computed velocity with Z, E, N components as attributes

Type AttribDict

displacement
[m] Computed displacement with Z, E, N components as attributes

Type AttribDict

horizontal_distance
[m] Computed horizontal distance

Type ndarray

traj_tvec
[s] Time array for all trajectory arrays

Type ndarray

Create an LSTrajectory object.

Parameters

- **force** (*LSForce*) – Completed force inversion
- **mass** (*int or float*) – [kg] Mass of event. If *None*, the mass is computed using *target_length*, which must be specified
- **target_length** (*int or float*) – [km] Center-of-mass runout length of event. If *None*, *mass* must be specified
- **duration** (*int or float*) – [s] If not *None*, only use the force time series from 0–*duration* seconds in the trajectory calculation
- **detrend_velocity** – [s] If provided, require the velocity to linearly go to zero at this time; if *None*, don't detrend
- **zeroacc** – [s] If provided, require the acceleration to be zero after this time, usually when forces are indistinguishable from zero, to reduce noise at end of trajectory. This is best used with the detrend_velocity option to avoid allowing the landslide to slide forever.

plot_trajectory(*elevation_profile=False*, *plot_jackknife=False*, *image=None*, *dem=None*, *reference_point=None*)

Plot trajectory results with context.

Parameters

- **elevation_profile** (*bool*) – If *True*, plot vertical displacement versus horizontal runout distance (H vs. L) instead of a map view
- **plot_jackknife** (*bool*) – Toggle plotting jackknifed displacements as well (if available)
- **image** (DataArray) – An image with coordinates defined in km with the origin (0, 0) being the start location of the trajectory
- **dem** (*str*) – A UTM-projected DEM GeoTIFF to slice thru for elevation profile plot
- **reference_point** (*int or float or list*) – If not *None*, plot a dot on trajectory, and line on colorbar, at this specified time(s) for reference

Returns Output figure handle

Return type Figure

2.2 Module contents

PYTHON MODULE INDEX

l

INDEX

R

S

T

V

W

Z