Location Workflow (**LOC-FLOW**) CookBook

Miao Zhang (Miao.Zhang@dal.ca)
Ruijia Wang (Ruijia.Wang@ualberta.ca)

August 2021
Version 1.0

LOC-FLOW is a "hands-free" earthquake location workflow to process continuous seismic records: from raw waveforms to well located earthquakes with magnitude calculations. The package assembles several popular routines for sequential earthquake location refinements, suitable for catalog building ranging from local to regional scales (see Liu et al., 2020 for broadband station application and Wang et al., 2020 for a nodal array application).

The LOC-FLOW is released and maintained at https://github.com/Dal-mzhang/LOC-FLOW.

Please download and install the packages first (**PhaseNet**[contains **Obspy**], **REAL**, **VELEST**, **HYPOINVERSE**, **hypoDD**, **GrowClust**, **FDTCC**, **Match&Locate**; see [STEP 0]). Questions related to the original packages should be addressed to the corresponding authors. All other credits to *Miao Zhang* (also author of REAL and Match&Locate), *Min Liu* (also author of FDTCC), and *Tian Feng*, who integrated these packages and made the I/O codes publicly available. If you find any part of the workflow useful, please cite the corresponding publications of the packages and/or our work.

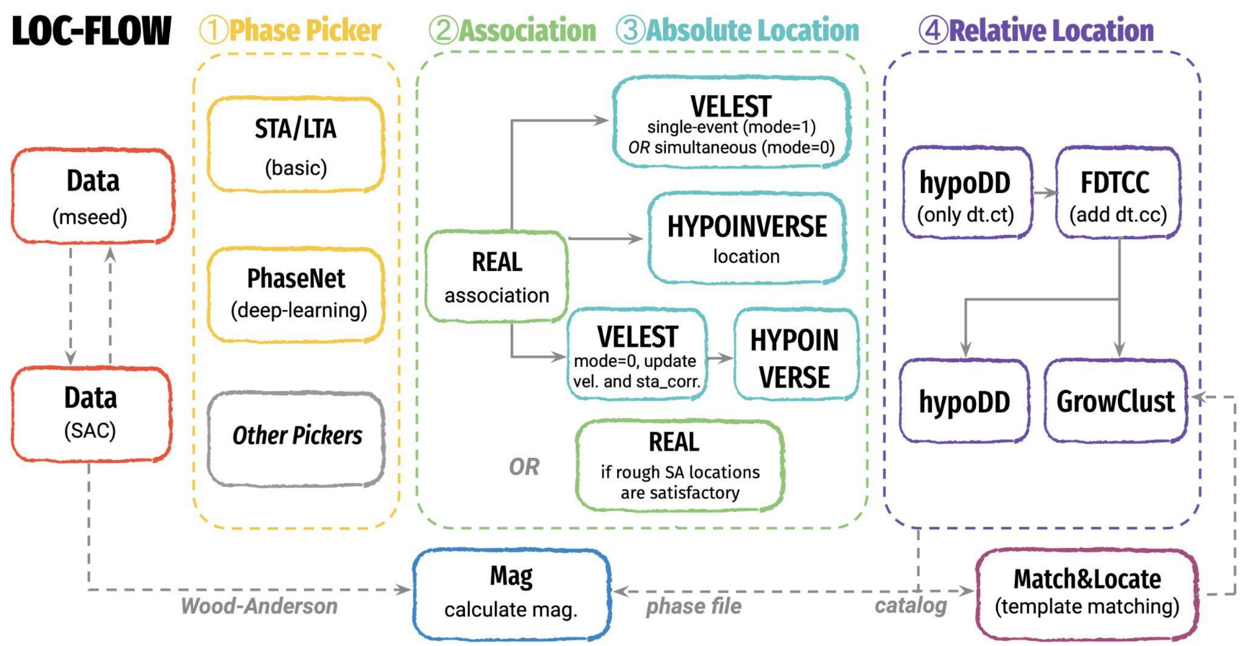

Figure1. Overview of the LOC-FLOW and packages involved.

Disclaimer:

While under continuous effort of improvements, the workflow has been tested by multiple research groups, but is made available without warranty. Users are free to make modifications to the programs to meet their particular needs, but are discouraged from distributing modified code to others without notification of the authors. Feedbacks, potential improvements and bug fixes should be addressed to the corresponding authors of these packages.

The views, thoughts, and opinions expressed in the text belong solely to the author(s), and not necessarily to the authors' employer, organization, committee or other group or individual.

Color and Text Style Definition:

codes to run (italic & bold) in corresponding colors in Figure 1

To run codes:

python CODENAME***.py***

perl CODENAME***.pl***

bash CODENAME***.sh***

CODENAME***.m*** in MATLAB

set up files for codes (may need to open and edit)

inputs/outputs (names may contain wildcards like *, [YYYYMMDD], STATION)

Tips and suggestions will be in blue boxes like this.

Installation

[STEP 0]   time: usually tens of minutes

- Download and install software packages

software_download.py download and manually install PhaseNet, REAL, FDTCC, hypoDD, GrowClust and Match&Locate (optional). A modified VELEST is included in the current version of REAL. PhaseNet installation environment includes ObsPy, which provides functions for data downloading and processing, as well as the STA/LTA picking.

Download and install HYPOINVERSE separately:
https://www.usgs.gov/software/hypoinverse-earthquake-location

run_install.sh move all commands to "bin" directory, manually add the "bin" path into your environment (~/.bash_profile or ~/.bashrc)

- Download data

waveform_download_mseed.py requests continuous data from FDSN providers. Store both raw mseed and response free SAC files. SAC files will be used for local magnitude estimation (see 6.Optional Utilities) and dt.cc calculation by FDTCC.

waveform_download.py alternative python script to download response free SAC data (need to provide station list: station_all.dat)

- Convert data and select station

phasenet_input.py prepares data inputs for PhaseNet to run, and station list (station.dat) for next steps

[STEP 1a or 1b] could be time consuming.
Suggest starting with [STEP 2] by assuming we have the data and picks .
Keep in mind that [STEP 5] and later still need the continuous SAC data.

The whole workflow was tested on MacBook Pro (2.9 Ghz, 6-core intel core i9, 32G memory), including the estimated running time.

Want to run LOC-FLOW seamlessly? See run_all.sh

1. Phase Picker

1.1. STA/LTA

[STEP 1a] **input:** **../../Data/waveform_sac; ../../Data/station.dat**
output: **./[YYMMDD]/NET.STATION.P[S].txt**
time: minutes

trigger_p_amp.py and ***trigger_s_amp.py*** run recursive_sta_lta and create REAL required P or S pick files. Phase amplitudes (for later magnitude calculation) are roughly measured by convolving wood-anderson response.

1.2. Phase Net [Zhu & Beroza, 2019]

[STEP 1b] **input:** **../../Data/waveform_phasenet; ../../Data/fname.csv**
output: **./[YYMMDD]/NET.STATION.P[S].txt**
time: minutes

Type ***conda activate phasenet*** in the command line to activate the proper environment for PhaseNet

runphasenet.py key code to run PhaseNet detection. Middle part of the code separates the P and S picks and then uses *pick2real* (included in REAL) to prepare the picks ready for REAL. Phase amplitudes (for later magnitude calculation) are roughly measured based on PhaseNet's PGV amplitude output.

2. Association

2.1. REAL [Zhang et al., 2019]

[STEP 2a] **input:** **./REAL/tt_db/mymodel.nd (velocity model, TauP format)**
output: **./REAL/tt_db/ttdb.txt**
time: minutes [depends on grid size]

tt_db/taup_tt.py [only need to run once], generate travel time table for REAL and FDTCC using Obspy's TauP and mymodel.nd

[STEP 2b] **input:** **./Pick/PhaseNet/[YYYYMMDD]/NET.STATION.P[S].txt**
./REAL/tt_db/ttdb.txt
./Data/station.dat
output: **./REAL/[YYYYMMDD].catalog_sel.txt**
./REAL/[YYYYMMDD].phase_sel.txt
./REAL/[YYYYMMDD].hypolocSA.dat (for next step)
./REAL/[YYYYMMDD].hypophase.dat (for next step)
./REAL/*allday.txt (merged file with all days)
./REAL/phase_best_allday.txt (merged file with all days, high quality events for later velocity model updating in VELEST, optional)
time: depends on grid size (most), number of stations and picks

runREAL.pl key code to run REAL detection by day. Please read the REAL manual for detailed parameter set ups.

Be careful with parameters！Extreme values significantly slow down speed.

i. start with a rough grid (e.g., 10*10*5) for faster tests and use a finer grid later (but still recommend <50*50*20). Scale the grid proportionally to your research region. Don't try to use a smaller grid size to improve locations. Later steps do a better job and much faster.

ii. More picks/stations, more strict thresholds. STA/LTA picks usually require more strict thresholds.

iii. *nrt* and *drt* should be scaled with the grid size properly. More tips are included at the beginning of ***runREAL.pl***.

3. Accurate Location

3.1. VELEST [Kissling et al., 1995]

run_velest.sh assembles three parts:

[STEP 3a] **input:** **../../Data/station.dat**
../../REAL/tt_db/mymodel.nd
../../REAL/phase_best_allday.txt [for mode=0 only]
../../REAL/phase_allday.txt
output: **velest.pha; velest.mod; initial.cat; velest.sta; velest.cmn**
time: seconds

convertformat.pl prepares inputs for VELEST

[STEP 3b] **input:** **velest.pha; velest.mod; velest.sta, regionsnamen.dat, regionskoord.dat**
output: **final.CNV; main.OUT; out.Check; sta.COR (Sta. Corr.; mode=0); velour.mod (updated vel., mode=0)**
time: seconds (mode=1), minutes (mode=0)

velest key code to run VELEST (for a given input set, one could modify parameters required by **velest.cmn** (in ***convertformat.pl***) and repeat this step only to test and optimize parameters). Two modes: mode=1- single event mode (update location only, one by one); mode=0 - simultaneous mode (use high quality events to update velocity, location, station correlation then re-run velest to relocate all events, only for large dataset with enough ray coverage)

[STEP 3c] **input:** **final.CNV**
output: **new.cat; dele.cat**
time: seconds

convertoutput.pl filters VELEST output based on travel time residual and station gap, save qualified events to new.cat.

Need to open and modify parameters！

i.in ***runvelest.sh*** the lat and lon should be region-specific (i.e., center of study region)

ii.in ***convertformat.pl*** a default **velest.cmn** is generated but is expected to be modified according to the VELEST manual.

3.2. HYPOINVERSE [Klein, 2002]

run_hypoinverse.sh assembles four parts:

[STEP 3a] **input:** **../../REAL/tt_db/mymodel.nd**
output: **vel_model_P.crh; vel_model_S.crh**
time: seconds

mk_velmodel.py [only need to run once] generates the velocity model for HYPOINVERSE format. The code doesn't like any two layers with same velocities or any low velocity layer, please modify it if your model has such.

[STEP 3b] **input:** **../../REAL/*.phase_sel.txt**
../../REAL/tt_db/mymodel.nd
../../Data/station.dat
output: **hypoinput.arc; station.dat**
time: seconds

mk_inputfile.py prepares phase and station inputs for HYPOINVERSE.

[STEP 3c] **input:** **hypoinput.arc; vel_model_P.crh; vel_model_S.crh; hyp.command**
output: **hypoOut.arc; prtOut.prt; catOut.sum**
time: seconds

hyp1.40 key code to run HYPOINVERSE (for a given input set, one could modify **hyp.command** and repeat this step only to test and optimize parameters).

[STEP 3d] **input:** **hypoOut.arc**
output: **new.cat; dele.cat**
time: seconds

cat hypoOut.arc | gawk filters HYPOINVERSE output based on location uncertainty, travel time residual and station gap, save qualified events to new.cat.

3.3. VELEST+HYPOINVERSE correction

run_hypoinverse_corr.sh assembles four parts:

[STEP 3a] **input:** **../VELEST/velest.mod [renamed from velout.mod in VELEST]**
output: vel_model_P.crh; vel_model_S.crh
time: seconds

mk_vel_velest2hypoinverse.py [only need to run once] generates the updated velocity model from VELEST (mode=0) for HYPOINVERSE format. Be aware that the code doesn't like any two layers with same velocities or any low velocity layer.

[STEP 3b] **input:** **../../REAL/*.phase_sel.txt**
../../Data/station.dat
output: **hypoinput.arc; station.dat**
time: seconds

mk_inputfile.py prepares phase and station inputs for HYPOINVERSE.

input: **.../VELEST/velest.sta [renamed from sta.COR in VELEST]**
../../Data/station.dat
output: P.del; S.del
time: seconds

mk_stacorr.py prepares station delay time files for HYPOINVERSE.

[STEP 3c] **input:** **hypoinput.arc;vel_model_P.crh; vel_model_S.crh; P.del; S.del;**
hyp.command
output: **hypoOut.arc; prtOut.prt; catOut.sum; new.cat**
time: seconds

hyp1.40 key code to run HYPOINVERSE (for a given input set, one could modify **hyp.command** and repeat this step only to test and optimize parameters, here the **hyp.command** is different from step 3.2).

[STEP 3d] **input:** hypoOut.arc
output: new.cat; dele.cat
time: seconds

cat hypoOut.arc | gawk filters HYPOINVERSE output based on location uncertainty, travel time residual and station gap, save qualified events to new.cat.

4. Relative Location

4.1. hypoDD [Waldhauser & Ellsworth, 2000]

run_hypoDD_dtct.sh assembles three parts:

[STEP 4a] **input:** **../REAL/phaseSA_allday.txt;** or **../location/VELEST/final.CNV;** or **../location/hypoinverse/hypoOut.arc;** or **../location/hypoinverse_corr/hypoOut.arc**
output: **hypoDD.pha**
time: seconds

velest2hypoDD.py or ***hypoinverse2hypoDD.py*** converts the outputs from [STEP3] for hypoDD.

[STEP 4b] **input:** **hypoDD.pha; station.dat**
output: **event.sel; event.dat; dt.ct**
time: seconds

ph2dt pairs the event and calculates the differential times based on criterions defined in **ph2dt.inp**.

[STEP 4c] **input:** **event.sel; station.dat; dt.ct**
output: **hypoDD.***
time: tens of seconds to minutes

hypoDD key code to run hypoDD relocation using only differential times (dt.ct).

Need to open and modify parameters！

i. in ***run_hypoDD_dtct.sh*** default **ph2dt.inp** and **hypoDD.inp** are generated but are expected to be modified according to the hypoDD manual.

ii. Note that hypoDD cannot locate events above sea level and will quit when the “air-quake” number becomes higher than 1000. A (imperfect) solution is to shift the input event depth (and velocity model) for 1-2 km (i.e., from seal-level to max regional elevation).

iii. The previous steps assume your velocity model is relative to the average station elevation (NOT sea level).

The above procedures should lead to a reasonably good catalog. Below are extra steps for relocation using cross-correlation (CC). We recommend finalizing results from the previous steps before moving forward.

5. Relative Location (CC)

5.1. hypoDD [Waldhauser & Ellsworth, 2000]

May hit RAM limit!

Go to HYPODD/include and change the ph2dt.inc for maximum event number (MAXEV) and other parameters (see hypoDD.inc). Make sure these parameters are 1) larger than your problem 2) suitable for your RAM.

run_hypoDD_dtcc.sh assembles three parts:

[STEP 5a] **input:** **../hypoDD_dtct/hypoDD.pha; ../hypoDD_dtct/hypoDD.reloc**
output: **hypoDD.pha**
time: seconds

updatephase.pl updates locations in the phase file (../hypoDD_dtct/hypoDD.pha) using available relocated events (../hypoDD_dtct/hypoDD.reloc) and generate a new hypoDD.pha

[STEP 5b] **input:** **hypoDD.pha; station.dat**
output: **event.sel; event.dat; dt.ct**
time: seconds

ph2dt pairs the event and calculates the differential times based on criterions defined in **ph2dt.inp**.

input: **../Data/station.dat; ../REAL/tt_db/ttdb.txt; ../Data/waveform_sac; event.sel; dt.ct; hypoDD.pha**
output: **dt.cc**
time: tens of seconds to minutes

FDTCC calculates differential times using cross-correlation (C-based code that runs fast; written by Min Liu and Miao Zhang, see FDTCC readme).

[STEP 5c] **input:** **event.sel; station.dat; dt.cc; hypoDD_cconly.inp**
output: **hypoDD.***
time: tens of seconds to minutes

hypoDD key code to run hypoDD relocation using cross-correlation differential times (dt.cc).

Need to open and modify parameters！

i. modify **ph2dt.inp** and **hypoDD_cconly.inp** according to the hypoDD manual.

ii.modify lines in ***run_hypoDD_dtcc.sh*** to personalize your cross-correlation calculation for FDTCC.

5.2. GrowClust [Trugman & Shearer, 2017]

run_growclust.sh assembles two parts:

[STEP 5a] **input:** **../../hypoDD_dtct/hypoDD.pha**
../../hypoDD_dtct/hypoDD.reloc
../../REAL/tt_db/mymodel.nd
../../Data/station.dat
output: **./IN/evllist.txt; ./IN/stlist.txt; ./IN/dt.cc; ./IN/vzmodel.txt**
time: minutes (depends on number of events, dominated by FDTCC)

gen_input.pl prepares inputs for GrowClust

FDTCC calculates differential times using cross-correlation (C-based code that runs fast; written by Min Liu and Miao Zhang, see FDTCC readme).

[STEP 5b] **input:** **growclust.inp; ./IN/***
output: **./OUT/out.growclust_cat; ./OUT/out.growclust_clust**
time: seconds or minutes (depends on number of events)

growclust key code to run GrowClust relocation

Need to open and modify parameters！

i. modify ./IN/**ph2dt.inp** according to the hypoDD manual and **growclust.inp** according to the GrowClust manual.

ii.modify lines in ./IN/gen_input.pl to personalize your cross-correlation calculation for FDTCC.

6. Optional Utilities

6.1. Plotting

[Opt] **input:** **REAL outputs (see [STEP2], REAL directory)**
output: *.pdf
time: seconds

t_dist.m plots P and S travel time vs. distance curves.

eventverify_pick.py and ***eventverify_all.py*** plots associated picks/waveforms of events (or at all stations) with specific ID (use SAC to zoom in and view specific phases for careful verification)

[Opt] **input:** **[any catalogs from any above steps, Plot directory]**
output: *.jpg, *.pdf (optional)
time: seconds

plot_3dmatlab.m plots the earthquake catalogs harvested from above steps (MATLAB code).

plot_3dgmt.sh plots the earthquake catalogs harvested from above steps (bash script uses GMT 6).

6.2. Magnitude calculation

[Opt] **input:** **[any catalogs and phases from any above steps, hypoDD phase format; magnitude directory];**
../Data/waveform_sac (response free continuous SAC data)
output: ./catalog_mag.txt
time: <1 second per event

calc_mag.py calculates local magnitude (ML) of the events. The magnitude formula may be modified as needed.

6.3. Match&Locate [Zhang & Wen, 2015]

[Opt] **input:** **Template catalog, velocity model, etc (MatchLocate directory)**
output: **MultipleTemplate/DetectedFinal.dat**
GrowClust/OUT/out.growclust_cat
time: varies

run_matchlocate.sh uses template events to search and locate for repeating/nearby events via waveform cross-correlation (recommend use matched-filter mode to save time; i.e., search reage -R=0/0/0). We refer readers to the Match&Locate manual for further information. The outputs will be fed to GrowClust (i.e., [Step 5a]) again for further location refinement. Complimentary codes are provided for the user's interest.

REFERENCES:

Zhang, M., Liu, M., Wang, R., Feng, T. & Zhu, W. LOC-FLOW: an end-to-end high precision earthquake location workflow. In prep.

Su, J., Liu, M., Zhang, Y., Wang, W., Li, H., Yang, J., Li, X. & Zhang, M. (2021). High resolution earthquake catalog building for the 21 May 2021 Yangbi, Yunnan, Ms 6.4 earthquake sequence using deep-learning phase picker, 64(8), 2647-2656, https://doi.org/10.6038/cjg2021O0530

Liu, M., Zhang, M., Zhu, W., Ellsworth, W. L., & Li, H. (2020). Rapid characterization of the July 2019 Ridgecrest, California, earthquake sequence from raw seismic data using machine‐learning phase picker. Geophysical Research Letters, 47(4), e2019GL086189, https://doi.org/10.1029/2019GL086189

Wang, R., Schmandt, B., Zhang, M., Glasgow, M., Kiser, E., Rysanek, S., & Stairs, R. (2020). Injection‐induced earthquakes on complex fault zones of the Raton Basin illuminated by machine‐learning phase picker and dense nodal array. Geophysical Research Letters, 46, e2020GL088168. https://doi.org/10.1029/2020GL088168

Zhang, M., Ellsworth, W. L., & Beroza, G. C. (2019). Rapid earthquake association and location. Seismological Research Letters, 90(6),2276–2284. https://doi.org/10.1785/0220190052

Zhang, M., & Wen L. An effective method for small event detection: match and locate (M&L). Geophysical Journal International, 200 (3), 1523-1537, 2015. https://doi.org/10.1093/gji/ggu466

Zhu, W., & Beroza, G. C. (2019). PhaseNet: A deep‐neural‐network‐based seismic arrival‐time picking method. Geophysical Journal International, 216(1), 261–273. https://doi.org/10.1093/gji/ggy423

Kissling, E., Kradolfer, U., & Maurer, H. (1995). Program VELEST user's Guide‐Short Introduction. Institute of Geophysics, ETH Zurich.

Klein, F. W. (2002). User's guide to HYPOINVERSE-2000, a Fortran program to solve for earthquake locations and magnitudes (No. 2002-171). US Geological Survey.

Waldhauser, F., & Ellsworth, W. L. (2000). A double‐difference earthquake location algorithm: Method and application to the Northern Hayward Fault, California. Bulletin of the Seismological Society of America, 90, 1353–1368. https://doi.org/10.1785/0120000006

Trugman, D. T., & Shearer, P. M. (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.https://doi.org/10.1785/0220160188