`CMT3D_FLEXWIN` Manual

Qinya Liu
Department of Physics
University of Toronto

June 12, 2013

`CMT3D_FLEXWIN` is re-written from my old `CMT3D` program to reflect the gist of the program, which is to use Gaussian elimination to solve the source inverse problem, In particular,

- directly use the output windows file from Alessia's `FLEXWIN` package, which selects time windows with decent match between data and synthetics.
- eliminate bandpass filter as it should have been applied before entering `FLEXWIN`.
- by default, inversions are done in local coordinates {`dep, lon, lat`}. However, with `global_coord=.true.`, it also allows inversions to be done in global coordinates {`X, Y, Z`}, which may come in handy for future multiple source inversions. The bottom lines is that the code should run exactly the same as before (in local coordinates) without even specifying this logical.
- incorporate centroid time shift and half duration as inversion parameters (npar=10 and 11), although derivatives for hdur and tshift have to be precomputed by python scripts (which one?)

What still remains to be added is the possiblity of `CMT3D` inversion for multiple subevents, which may be useful if N_{event} is relatively small, and may be handy for efficiency comparisons to `CMT3D_CG`.

1 Setup

1.1 Setup derivative cmt solutions

According to the theory in the Appendix, we need to compute the derivative synthetics by brute force difference formula

$$\frac{\partial s}{\partial m} \sim \frac{s(m+\Delta m) - s(m)}{\Delta m} \tag{1}$$

for all source parameters involved in the inversion. Depending on how many parameters are chosen (`npar`), we need to run at least $6-9$ SEM simulations to gather all the derivatives. Each SEM simulation requires a corresponding `CMTSOLUTION` for $m + \Delta m$, and the Δm should be chosen carefully to avoid either numerical inaccuracy or instability. In the local coordinates {`dep, lon, lat`}, for the global code (`SPECFEM3D_GLOBE`), a choice may be

$$\Delta M = 1 \times 10^{24} \text{ dyne.cm}, \quad \Delta X = 2 \text{ km (depth)}, \quad \Delta\delta = 0.02 \text{ deg (lat/lon)} \tag{2}$$

and for the basin code (`SPECFEM3D`), a choice may be

$$\Delta M = 1 \times 10^{22}\ \text{dyne.cm}, \quad \Delta X = 0.5\ \text{km}, \quad \Delta\delta = 0.01 \tag{3}$$

where ΔX refers to depth perturbations and $\Delta\delta$ refers to perturbation in latitude and longitude.

Depending on what coordinate system you choose to calculate derivatives and run `CMT3D` inversions, different extensions are expected for derivative `CMTSOLUTION`s as well as derivative synthetics (generated by `gen_der_cmt()` python script):

- local (depth, lon, lat) coordinates (`global_coord=.false.`):
  `ext=['Mrr','Mtt','Mpp','Mrt','Mrp','Mtp','dep','lon','lat']`
  This system should used as default for most applications.

- global code, global (X,Y,Z) coordinates (`global_coord=.true.`):
  `ext=['Mxx','Myy','Mzz','Mxy','Mxz','Myz','xxx','yyy','zzz']`
  This coordinate system may only be used to accommodate multiple event inversions in the future. For derivations of rotation matrices, please refer to the Appendix.

Note:

1. Python script and modules are provided in `scripts` directory to facilitate this process, and allow full compatibility with `CMT3D` code:

   ```
   gen_cmt_der('CMTSOLUTION',npar=9,dmoment=1.e24,
                 ddepth=2,dlocation=0.02,utm=0)
   ```

   where $dmoment$, $ddepth$ and $dlocation$ are given in dyne·cm, km and degrees ($\Delta X = 111.12\Delta\delta$), and $utm = 0$ refers to local coordinates (depth, lon, lat).

   If $utm = -1$, it refers to global coordinates (X, Y, Z), and `ddepth=dX`, `dlocation=dY=dZ` (may need to modified in future). Note that `gen_cmt_der` does allow utm to be 1-60, and map small regions into a cartesian geometry (depth, E, N). But that is NOT yet used for the `cmt3d` package.

2. Also as discussed in the theory section, derivatives for centroid time and half duration simply depend on the original synthetics, and no extra simulation is necessary. how to compute?

3. After simulations for these derivative `CMTSOLUTION`s are finished, collect synthetic seismograms, convert them to sac files and add proper extension to the sac files according to the choice of coordinate system.

4. Note synthetics for the original `CMTSOLUTION` do not need to be obtained through an extra simulation, a simple linear combination of moment tensor derivatives can be realized by:

   ```
   xadd_frechet_derivatives s syn_list cmt moment_scale(1e24)
   ```

1.2 Pre-processing data and synthetics

After gather raw data from IRIS (for global earthquakes) or SCEDC (for socal earthquakes), and collect synthetics as well derivative synthetics from SEM simulations, you need to go through the pre-processing step to present properly cut and filtered data and synthetics to the `FLEXWIN` package. I have come a long way in writing such scripts, and right now I am settled on a recent one in python:

```
 process_ds.py  \
  -d data,sac -s syn,  -p CMTSOLUTION,STATIONS\
  -t 6,30 -w -D [-r]
```

For detailed usage, simple type `process_ds.py`. The outcomes of this script are

- filtered data and (derivative) synthetics in `data_6_30` and `syn_6_30`.
- an input file to `FLEXWIN` named `INPUT_flexwin_6_30` will be generated.

1.3 Executing FLEXWIN

After choosing the right parameters in `PAR_FILE` and user functions for flexwin package (please refer to `FLEXWIN` manual for details), simply run

```
./flexwin < INPUT_flexwin_6_30 > OUTPUT_flexwin_6_30
```

After running `FLEXWIN` through data and synthetics filtered at various period bands (e.g., `3-30` and `6-30` secs), all individual windows files in `MEASURE_t1_t2` directories can be cat into an input for `cmt3d_flexwin` package.

```
write_flexwin_out.py MEASURE_6_30 MEASURE_20_50  \
                     INPUT_cmt3d_flexwin_noweight
```

Now it is possible to pad time shift and weight information to the composite output windows file (after the time range of the window), either to facilitate a direct evaluation of misfit function, or to eliminate the weight calculations in `CMT3D` package, which can be rather cumbersome and inflexible.

```
write_flexwin_shift_weight.py -c 2/2/1 -d 1.15/0.55/0.78 -a 0.2 -r 2 -t \
      -D data -s MEASURE  INPUT_cmt3d_flexwin_noweight \
      INPUT_cmt3d_flexwin_withweight
```

Obviously you can not just blindly use this package without checking. Here are two auxilliary scripts:

```
plot_flexwin.py MEASURE
plot_data_and_syn.pl -d data_dir,data_ext -s syn_dir,syn_ext -m cmt \
   -M OUTPUT_flexwin -A (Z/R/T) 0.05/0.05/0.08
```

The first script gives `flexwin_seismo.pdf` and the second one gives `ds_all.pdf`.

2 CMT3D package

2.1 Parameter file

For structures of the main program and subroutine calls, refer to `call-graph`. The name of the parameter file is hard-wired in the main program as `INVERSION.PAR`.

```
CMTSOLUTION
CMTSOLUTION_NEW
6  .false.                      -- npar[, global_coord]
0.01 1.0  1.0e22                -- dlocation,ddepth,dmoment
flexwin.out
.true. .false.                 -- weigh_data_files[, read_weight]
2 2 1  0  1.15 0.55 0.78       -- weights of data comp,az,dist
.true.                         -- station_correction
.true.  .true. 0.0             -- zero_trace,double_couple,damping
.false.                        -- write_new_syn (psuedo-synthetics)
```

The first line gives the old cmt solution file name. This is also the cmt solution you have used to generate derivative cmts and compute the synthetics from linear combinations. The second line gives name of the new cmt solution file output after the inversion is done.
File `flexwin.out` holds the start and end time of the windows outputed from FLEXWIN package, in the following format:

```
nfiles
data-file-1
syn-file-1
 nwins-for-this-data-and-syn
 tstart(1) tend(1) [tshift(1) weight(1)]
 ...
 tstart(nwin) tend(nwin) [tshift(nwin) weight(nwin)]
data-file-2
syn-file-2
...
```

Only data points inside the windows from `flexwin.out` file will be used to perform time integration and assemble the inversion matrices. Note the values in [] are recent additions where we allow weights to be specified in the input file together with an indicative logical `read_weight`. For now, `tshift` is still a dummy variable not used for aligning data and synthetics (a separate xcorr subroutine calculates it in the code, but future merge may be possible).

All other parameters are pretty self-explanatory:

- Number of parameters to be inverted can only be 6(moment-only), 7(moment+depth), 9(moment+locations). `global_coord` indicates if the input derivative synthetics are in terms of (X, Y, Z) or $(depth, lon, lat)$.

- For local coordinates, the derivative synthetics should have been calculated based on the `dlocation` (in degrees), `ddepth` (in km), `dmoment` (in dyne·cm) values. However, notice that strictly speaking, only the ones for moment are actual derivative synthetics; the ones for depth, longitude, and latitude (or R, S, E depending on coordinate system or utm) are actual synthetics, from which the code subtracts the old synthetics and divides it by ddepth or ddelta (refer to Appendix for more details).
- Weighting subroutine `compute_data_weights()` needs to be tweaked if you choose to `weigh_data_files`, and not importing weights directly from the input file by `read_weight`. The idea is that you utilize the `dist_km`, `dist_deg`, `azimuth`, and other file information from the sac headers to come up with a sensible weighting scheme for your particular scale/region.
- Station correction is used to align the data with the synthetics to accommondate 3D path effects that may bias your source inversion results.
- Different constraints on the moment-tensor are allowed, including zero-trace and double-couple. To improve the stability of the matrix inverse (when only very few windows are used), damping is also allowed before Gaussian elimination.
- Finally synthetics for the new cmt solution can be written, however, notice that these are obtained from 'linearized' approximation, not from an actual SEM simulation.

Some more words on weighting:

- If you want to invert either only surface waves or body waves, make sure you set the optimal parameters in `FLEXWIN` to get the correct sets of windows.
- How do the weights work in `compute_data_weights()`:

```
data_weights(nwint) =  cmp_weight(nwint) &
     * (dist/REF_DIST) ** dist_exp_weight(nwint) &
     * corr(nwint) ** corr_exp_weight(nwint) &
     / ( naz(k) ** az_exp_weight)
```

  where component weights are applied linearly, distance and azimuth weights are applied exponentially (such that 0 means no weight and 1 means linear weight). `naz(k)` is the number of traces available in a particular azimuth bin ($+1$). One can set `az_exp_weight` to be 0 to mute out the azimuthal effect, the same for distance weights.

2.2 Synthetic data tests

This package has been fully tested for $6, 7, 9, 10$ parameter inversions based on synthetic data.

A CMT3D Theory

A.1 Moment-tensor Inversions

We seek to minimize the difference between data $\{d_i(t), i = 1, \ldots, N\}$ and corresponding synthetics $\{s_i(\mathbf{m}, t), i = 1, \ldots, N\}$ computed for a given model parameters $\mathbf{m} = \{m_j, j = 1, \ldots, M\}$ after applying some bandpass filter:

$$\phi(\mathbf{m}) = \frac{1}{2}\sum_{rq}\int W_{rq}(t)\left[\mathbf{s}^b(\mathbf{m}, \mathbf{x}_r, t) - \mathbf{d}_i^b(\mathbf{x}_r, t)\right]^2 dt \tag{4}$$

where $W_{rq}(t)$ combines the taper function with weighting factor for the q'th window at r'th receiver. $\mathbf{s}^b$ and $\mathbf{d}^b$ denote filtered data and synthetics.

If we take the first and second order derivatives of this misfit function at some given reference model $\mathbf{m}^0$, we obtain

$$\frac{\partial\phi(\mathbf{m}^0)}{\partial m_j} = \sum_{i=rq}\int W_{rq}(t)\left[\mathbf{s}^b(\mathbf{m}^0, \mathbf{x}_r, t) - \mathbf{d}^b(\mathbf{x}_r, t)\right]\frac{\partial s_i(\mathbf{m}^0, \mathbf{x}_r, t)}{\partial m_j}\,dt \tag{5}$$

$$\frac{\partial^2\phi(\mathbf{m}^0)}{\partial m_j \partial m_k} \sim \sum_{i=rq}\int W_{rq}(t)\frac{\partial\mathbf{s}^b(\mathbf{m}^0, \mathbf{x}_r, t)}{\partial m_j}\frac{\partial\mathbf{s}^b(\mathbf{m}^0, \mathbf{x}_r, t)}{\partial m_k}\,dt \tag{6}$$

where for the second-order derivatives, we have omitted the term related to second-order synthetics derivatives, i.e. approximate Hessian

By assuming that at the optimal model parameter $\mathbf{m}$, we have $\frac{\partial\phi}{\partial\mathbf{m}} = 0$, which translates to the quasi-Newton's method:

$$\frac{\partial\phi(\mathbf{m}^0)}{\partial m_j} + \frac{\partial^2\phi(\mathbf{m}^0)}{\partial m_j \partial m_k}(m_k - m_k^0) = 0 \tag{7}$$

Therefore, we just need to compute the first and second-order derivatives according to equations 6, which further points to the computation of derivative synthetic seismograms $\frac{\partial s_i(\mathbf{m}^0)}{\partial m_j}$.

A.2 Derivative Synthetics

We know in general, the synthetics for model $\mathbf{m}$ can be expanded with respect to a reference model $\mathbf{m}^0$:

$$s(\mathbf{m}, t) = s(\mathbf{m}^0, t) + \sum_{j=1}^{M}\frac{\partial s_i(\mathbf{m}^0, t)}{\partial m_j}(m_j - m_j^0) \tag{8}$$

If source parameters only include moment-tensor elements, then the linear dependences can be written as:

$$s(\mathbf{m}, t, t) = \sum_{j=1}^{M} \frac{\partial s_i(\mathbf{m}^0, t)}{\partial m_j} m_j \tag{9}$$

Therefore derivatives for moment-tensor elements can be computed by

$$\frac{\partial s_i(\mathbf{m}^0, t)}{\partial M_j} = s(M_j^0, t)/M_j^0 \tag{10}$$

where $s(M_j^0, t)$ is the synthetics calculated by setting all the moment tensor elements other than the jth component to 0.

For other source parameters, including event location and depth, a finite-difference formula has to be used:

$$\frac{\partial s_i(\mathbf{m}^0, t)}{\partial m_j} = \frac{s(m_j^0 + \Delta m_j^0, t) - s(m_j^0, t)}{\Delta m_j} \tag{11}$$

where $s(m_j^0 + \Delta m_j^0, t)$ refers to synthetics computed from only perturbing the j'th component of the source parameters. Δm_j needs to be selected with care to avoid numerical noise. Typically $2 - 5$ km for global earthquakes and 1 km for local earthquakes.

Derivatives for time shift and half duration can be derived from the original synthetics itself. Given a Gaussian source time function

$$g(t) = \frac{1}{\sqrt{\pi} t_h} e^{-\left(\frac{t - t_s}{t_h}\right)^2} \quad \text{or} \quad g(\omega) = e^{-\frac{\omega^2 t_h^2}{4}} e^{-i\omega t_s} \tag{12}$$

under the Fourier transformation pairs:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt, \quad f(t + a) \Leftrightarrow F(\omega) e^{i\omega a}, \quad \dot{f}(t) \Leftrightarrow (i\omega) F(\omega) \tag{13}$$

Given Green's functions for a point source with delta moment function, we can express the synthetics for an earthquake as

$$\mathbf{s}(\mathbf{x}_r, \omega) = \frac{M}{i\omega} : \nabla G(\mathbf{x}_r, \mathbf{x}_s, \omega) g(\omega) \tag{14}$$

i.e., synthetics for error function as moment function. Therefore

$$\frac{\partial \mathbf{s}(\mathbf{x}_r, \omega)}{\partial t_{s,h}} = \frac{M}{i\omega} : \nabla G(\mathbf{x}_r, \mathbf{x}_s, \omega) \frac{\partial g(\omega)}{\partial t_{s,h}}. \tag{15}$$

Since

$$\begin{aligned}\frac{\partial g(\omega)}{\partial t_s} &= -(i\omega)g(\omega)\\ \frac{\partial g(\omega)}{\partial t_h} &= -\frac{1}{2}t_h\omega^2 g(\omega) = -\frac{1}{2}t_h(i\omega)\frac{\partial g(\omega)}{\partial t_s}\end{aligned} \tag{16}$$

we have

$$\begin{aligned}\frac{\partial s(\mathbf{m}_0,t)}{\partial t_s} &= -\dot{\mathbf{s}}(\mathbf{m}_0,t)\\ \frac{\partial s(\mathbf{m}_0,t)}{\partial t_h} &= -\frac{1}{2}t_h\partial_t[\frac{\partial s(\mathbf{m}_0,t)}{\partial t_s}]\end{aligned} \tag{17}$$

A.3 Rotation between Global and Local coordinates

A series rotation is needed to tranform local (r,θ,ϕ) coordinates (which is also what CMT moment tensor uses) into the global coordinates (X,Y,Z), where X is pointing to $0°$ longitude, and vice versa.

For a given point with $(\theta = 90 - Lat, \phi = Long)$ on a **unit** sphere, the coordinates in the cartesian global coordinate system is given by

$$X = \sin\theta\cos\phi, \quad Y = \sin\theta\sin\phi, \quad Z = \cos\theta \tag{18}$$

Conversely, given the global coordates (X,Y,Z),

$$\theta = acos(Z/r), \quad \phi = atan2(Y/\sin\theta, X/\sin\theta) \tag{19}$$

Note here if $\theta = 0$ (North pole), then ϕ is undetermined, and can be basically any number.

For rotation matrices, as an example, in 2-D, for the same point, the coordinates under new coordinate system is related to the old coordinates by

$$\begin{bmatrix}X_{new}\\ Y_{new}\end{bmatrix} = \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}X_{old}\\ Y_{old}\end{bmatrix} \tag{20}$$

After the first two simple rotations ($Z(\phi)$ and $Y(\theta)$) demonstrated by Figure 1, we have the relation between (X,Y,Z) and (θ,ϕ,r) systems:

$$\begin{bmatrix}x_\theta\\ x_\phi\\ x_r\end{bmatrix} = \begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\phi & \sin\phi & 0\\ -\sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} \tag{21}$$

This is not the end of story, as the usual order of local coordinate system is (r,θ,ϕ) instead of (θ,ϕ,r). Therefore two extra rotations ($Y(-90°)$ and $X(-90°)$) are needed, which give an extra

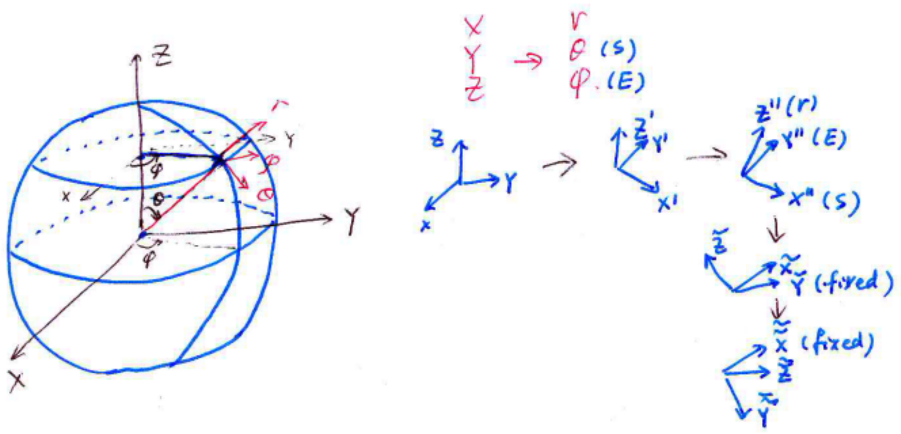

Figure 1: Rotation matrices

(left-applied) rotation matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-90^\circ) & \sin(-90^\circ) \\ 0 & -\sin(-90^\circ) & \cos(-90^\circ) \end{bmatrix} \begin{bmatrix} \cos(-90^\circ) & 0 & -\sin(-90^\circ) \\ 0 & 1 & 0 \\ \sin(-90^\circ) & 0 & \cos(-90^\circ) \end{bmatrix} \tag{22}$$

Multiply these two matrices together, we obtain the final rotation matrix from (X, Y, Z) to (r, θ, ϕ):

$$R^T = \begin{bmatrix} \sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \end{bmatrix} \tag{23}$$

which can be verified by

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = R^T \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix} \tag{24}$$

The rotation matrix from local (r, θ, ϕ) to global (X, Y, Z) is simply:

$$R = \begin{bmatrix} \sin\theta\cos\phi & \cos\theta\cos\phi & -\sin\phi \\ \sin\theta\sin\phi & \cos\theta\sin\phi & \cos\phi \\ \cos\theta & -\sin\theta & 0 \end{bmatrix} \tag{25}$$

B Testing (further update awaits)

Two types of datasets are provided to accurately test this inversion package, one with station correction, and one without. Synthetics and derivative synthetics for corresponding CMTSOLUTION are always in syn_T006_T030. No pre-processing are needed for these dataset as they have gone through all the procedures. Although to achieve inversion result that is accurate to several digits after the decimal point, several tweaks with the usual pre-processing procedures have to be done.

- (obsolete, needs fix) we should not use -h in the synthetics processing to ensure that the synthetics data and the derivative synthetics go through the exact same set of processing procedures. we also need to take out the rtrend and rmean line in the filtering steps of process_syn_new.pl and process_cal_data.pl (?). Since the data and synthetics presented in test_dataset have gone through the script:

  ```
  process_data_and_syn.pl  -d data,sac -s syn,  \
         -0 -m CMTSOLUTION -a station_file   \
         -1 -b -26 -2 -t 6/30
  ```

  which generated process_ds.csh script for all the commands, we can just tweak individual lines in process_ds.csh to achieve our test dataset. Also note that the -i none option for data processing and -h option for synthetics processing have to be taken out from the script to make sure that synthetics and data are processed in exactly the same way.

- The synthetic data for the 7 paramter case are generated by the following command:

  ```
  xadd_frechet_full s syn_file_list old_cmt true_cmt \
                    doment depth dx[dy]
  ```

  which is a fancier version of the original

  ```
  xadd_frechet_derivatives s syn_file_list \
                           cmt_file moment_scale
  ```

  Of course we can't really use the true synthetics for the true cmt solution in a 7-par case, since we know that the synthetics is a non-linear function of depth, and the inversion won't accurately recover depth. Linearized 'pseudo-synthetics' will ensure accurate results, which is the purpose of testing.

- It is impossible to design test dataset for 7 parameter inversion with time shift, since time shifts interfer with the effect of waveform change caused by depth variation, therefore, may not give exact solution, and sometimes may even produce the incorrect solution. This is part of the reason why I also wrote the grid search package GRID3D_FLEXWIN.

C Auxilliary Scripts and Programs

A list of the scripts that are used before and after the package:

- `process_data_and_syn.pl`, which includes `process_data_new.pl`(global/regional) or `process_cal_data.pl` (socal), `process_syn_new.pl` (global/regional) or `process_trinet_syn_new.pl`(socal), `pad_zeros.pl` (include `pad_zero_to_syn`), `rotate.pl`, `saclst`. Of course `sac` has to be present in the system.
- `prepare_meas_all.pl`
- `xadd_frechet_derivatives` and `xadd_frechet_full`