LLNL-SM-866718

Lawrence Livermore National Laboratory

MTINV version 4. Interface Control Document for Greens Functions

7/16/24

Gene Ichinose

Disclaimers

Auspices

Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

This Ground-based Nuclear Detonation Detection (GNDD) research was funded by the National Nuclear Security Administration, Defense Nuclear Nonproliferation Research and Development (NNSA DNN R&D).

This Low Yield Nuclear Monitoring (LYNM) research was funded by the National Nuclear Security Administration, Defense Nuclear Nonproliferation Research and Development (NNSA DNN R&D). The authors acknowledge important interdisciplinary collaboration with scientists and engineers from LANL, LLNL, NNSS, PNNL, and SNL.

Section 1

Introduction

The purpose of this Interface Control Document (ICD) is to instruct the reader on how to modify or create codes writing output to store Greens functions (GFs) compatible with the moment tensor inversion toolkit (MTINV) version 4. MTINV uses an integrated frequency-wavenumber (*f*κ) code written by Yuehua Zeng[(1)] which is one of the best features of the toolkit making it easy to estimate moment tensors from regional long-period filtered waves. However, as scientific and computational advances are made on 3D earth models, there is now a need to load GFs from other *f*κ and simulation codes.

We provide applications and examples to load GFs from Robert B. Herrmann's (RBH) Computer Programs in Seismology (CPS) *f*κ-code named hspec96[(2)] and Lawrence Livermore National Laboratory's (LLNL) finite-difference simulation application **`sw4`** [(3,4)]. The **`sw4`** GF loader allows additional capability for scaling and polarity switchers in case there are any future coordinate or formatting convention changes, for example, this may be extended to SPECFEM[(5)]. The final section is a short description and example script to convert and load GFs using an alternative moment tensor formulation. The appendix includes some of the conventions and coordinate systems adopted by the MTINV toolkit.

Footnotes:

(1) Zeng, Y. and J. G. Anderson (1995). A Method for Direct Computation of the Differential Seismogram with Respect to the Velocity Change in a Layered Elastic Solid, Bull. Seism. Soc. Am. 85(1), 300-307, https://doi.org/10.1785/BSSA0850010300

(2) Herrmann, R. (2013). Computer Programs in Seismology: An Evolving Tool for Instruction and Research, Seismological Research Letters 84 (6), 1081-1088. https://doi.org/10.1785/0220110096

(3) LLNL **`sw4`** https://computing.llnl.gov/projects/serpentine-wave-propagation/software

(4) **`sw4`** GitHub https://github.com/geodynamics/sw4

(5) SPECFEM https://specfem.org

Section 2

Description of Greens Data Structure

There are 4 parts of the `grnlib` or "`*.glib`" dot G-lib file. The first part is a header with depth information since the GF library is a function of depth. The second part is a data structure named "Greens", described in the C header file "`include/mt.h`", that contains, for example, metadata including source and receiver coordinates, number of time samples, and sampling rate as well as some specific information about the *f*κ calculation. The third part is the velocity model used, see "VelMod" data structure in "`include/mt.h`". This data structure is designed for 1D models and therefore with 3D models the user creates and attaches an object initialized with zeros or some specified dummy model. The final fourth part contains the actual GFs as ten arrays of floats with fixed size array of 4096 samples. If the GFs sizes are less than 4096, then the rest of the array is initially set to zero but if the GF sizes is greater than 4096, then the GFs will be truncated to 4096. All GF components are scaled by a base moment of M_w 0 (Mo = 1.2445×10^{16} dyne*cm). RBH GFs are scaled to M_w 2.6 (M_0 = 1.0×10^{20} dyne*cm) where the difference is δM_0 = 8035.26. Most other *f*κ codes and **`sw4`** use scalar seismic base moment of 1.0×10^{20} dyne*cm.

The first section of the "`*.glib`" file format consists of a header with the GF source depth information. The first value is an integer for the number of depths, followed by a vector of floats containing the depths in units of kilometers (see first 2 rows in Table 1). The rest of details are described in Figure 1 and Table 1. It is important to note that the Greens part of the file repeats depending on the number of depths specified. The following is C-code to read GFs file.

```
/*** loop over all stations ***/
for( ista = 0; ista < nsta; ista++ )
{
  fread( &(z[ista].nz), sizeof(int), 1, fp );  /* read nz */

   /*** allocate depth array memory ***/
  z[ista].z = (float *)calloc( z[ista].nz, sizeof(float) );

    /*** read depth array ***/
  fread( &(z[ista].z[0]), z[ista].nz * sizeof(float), 1, fp );

    /*** allocate memory for GF array size of nz ***/
  grn[ista] = (Greens *)malloc( z[ista].nz * sizeof(Greens) );

  for( iz = 0; iz < z[ista].nz; iz++ ) /*** loop over depths to read all GFs ***/
  {
      fread( &(grn[ista][iz]), sizeof(Greens), 1, fp );
```

Figure 1. Code snippets from the file include/mt.h defining parts of the GF file format

```
/** In mt.h **/

static float base_moment = 1.2445146117713818e+16;

typedef struct {
                                  /*sta=stnm*/
        char filename[256], net[8], stnm[8], loc[8];
        float stla, stlo, stel, evla, evlo, evdp;
        float rdist, az, baz;
        float t0, dt, twin, fmax, damp, eps, smin, rigidity;
        float redv, ts0, tstart, tend;

/** newly added 2010/11/27 G. Ichinose see rayp_subs.c ***/

        float Ptakeoff, Prayparameter, Pttime, Praybottom;

        int kmax, nt;
        VelMod v;
        Greens_Function g;
        float *rad, *tra, *ver;
} Greens;

#define MAX_MODEL_LAYERS 1024
typedef struct {
        char modfile[256], modpath[256];
        int nlay;
        int maxlay;
        float thick[MAX_MODEL_LAYERS];
        float  ztop[MAX_MODEL_LAYERS];
        float    vp[MAX_MODEL_LAYERS];
        float    vs[MAX_MODEL_LAYERS];
        float    qa[MAX_MODEL_LAYERS];
        float    qb[MAX_MODEL_LAYERS];
        float   rho[MAX_MODEL_LAYERS];
        float sigma[MAX_MODEL_LAYERS];
} VelMod;

typedef struct {

        /*** the original 3-fundamental faulting orientations + 1-isotropic Gf ***/
        /*** 3-component displacement r-radial z-vertical t-transverse ***/

        float rss[4096], rds[4096], rdd[4096], rep[4096];
        float zss[4096], zds[4096], zdd[4096], zep[4096];
        float tss[4096], tds[4096];  /*** tdd=0 and tep=0 ***/

        /*** the 3-fundamental faulting orientations + 1-isotropic Gf ***/
        /*** 3 components of rotation w3-rotation about vertical axes, ***/
        /***      w1-rot about E-W, w2-rot about N-S axis ***/

        float w3ss[4096], w3ds[4096], w3dd[4096], w3ex[4096];
        float w2ss[4096], w2ds[4096], w2dd[4096], w2ex[4096];
        float w1ss[4096], w1ds[4096], w1dd[4096], w1ex[4096];

} Greens_Function;
```

Table 1. Greens function library file format structure (see mtinv.version/include/mt.h)

Depth Header					
N_z = (int) Number of depths					
Z_1 = (float) depth #1 in km	Z_2 = (float) depth #2 in km	Z_3 = (float) depth #3 in km	(…)	Z_N = (float) depth N_z in km	
This section repeats as function of depth					
Filename = (char [256]) name of grnlib file(1)	Net = (char [8]) network code	stnm = (char [8]) station code	Loc = (char [8]) location code		
stla = (float) station latitude in decimal degrees	stlo = (float) station longitude in decimal degrees	stel = (float) station elevation in decimal degrees	evla = (float) event latitude in decimal degrees	evlo = (float) event longitude in decimal degrees	evdp = (float) event depth km
rdist = (float) epicenter distance in km	az = (float) source-to-receiver azimuth in degrees	baz = (float) receiver to source azimuth in degrees			
t0 = (float) start time in seconds	dt = (float) sampling rate in seconds per sample	twin = (float) time length of greens function in seconds	fmax = (float) f-k computational cut-off frequency in Hz	damp = (float) f-k damping factor	eps = (float) f-k error tolerance
smin = (float) source minimum	rigidity = (float) $\rho*V_s*V_s$ at the source layer				
redv = (float) reduction velocity	ts0 = (float) time shift in seconds	tstart = (float) start time in seconds	tend = (float) end time in seconds		
ptakeoff = (float) P-wave takeoff angle in degrees	prayparam… = (float) P-wave ray parameter in sec/km	pttime = (float) P-wave travel time in sec	praybottom = (float) bottoming depth of P-wave in km		
kmax = (int) maximum wavenumber	nt = (int) number of samples				
VelMod data structure (see Table 2.)					
Greens_Function data structure (see Table 3.)					
Rad = (float *) temporary	tra = (float *) temporary	ver = (float *) temporary			

storage for radial component synthetics	storage for transverse component synthetics	storage for vertical component synthetics			
EOF or GF for next depth increment					

(1) The filename format is fixed: {network}.{station}.{location}.model.glib
(2) Gray shaded are optional parts of the file, or table headings

Table 2. Velmod data structure: 1-D layered earth velocity, attenuation, and density model

modfile = (char [256]) model filename
modpath = (char [256]) path to model file
nlay = (int) number of layers
maxlay = (int) maximum number of layers set to #define MAX_MODEL_LAYERS 1024
thick = (float [1024]) layer thickness in km
ztop = (float [1024]) depth to top of layer in km
Vp = (float [1024]) P-wave velocity in km/sec
Vs = (float [1024]) S-wave velocity in km/sec
Qa = (float [1024]) P-wave quality factor unitless
Qb = (float [1024]) S-wave quality factor unitless
rho = (float [1024]) density grams/cubic cm
sigma = (float [1024]) Poisson's Ratio unitless

Table 3. Greens_Function data structure: 10 fundamental faults and rotational cmps

rss = (float [4096]) radial cmp strike-slip	rds = (float [4096]) radial cmp vertical-dip slip	rdd = (float [4096]) radial cmp dip-slip	rep = (float [4096]) radial cmp isotropic
zss = (float [4096]) vertical cmp strike-slip	zds = (float [4096]) vertical cmp slip	zdd = (float [4096]) vertical cmp dip-slip	zep = (float [4096]) vertical cmp isotropic
tss = (float [4096]) transverse cmp strike-slip	tds = (float [4096]) transverse cmp vertical-dip slip		
w3ss = (float [4096]) rotational cmp for strike-slip	w3ds = (float [4096]) rotational cmp for vertical-dip slip	w3dd = (float [4096]) rotational cmp for dip slip	w3ex = (float [4096]) rotational cmp for isotropic
w2ss = (float [4096]) rotational cmp for strike-slip	w2ds = (float [4096]) rotational cmp for vertical-dip slip	w2dd = (float [4096]) rotational cmp for dip slip	w2ex = (float [4096]) rotational cmp for isotropic
w1ss = (float [4096]) rotational cmp for strike-slip	w1ds = (float [4096]) rotational cmp for vertical-dip slip	w1dd = (float [4096]) rotational cmp for dip slip	w1ex = (float [4096]) rotational cmp for isotropic

Gray shaded are optional parts of the file

Section 3

Application examples: `hspec96_to_grnlib`

The current version of MTINV has been integrated with `hspec96`. We added a `-hspec96` option to the helper applications `setupMT` which is run first to setup the GF calculations.

```
setupMT  --hspec96 \
         -z 2  \
         -ev 41.7392 +73.377  \
         -mod wus \
         -ot "2009-12-22T05:54:35" \
         -com "Kambrata, Kyrgyzs" \
         -fil 0.033 0.10 \
         -DataDir ../Data \
         -RespDir ../Resp \
         ../Data/*Z.?.SAC ../Data/*Z.SAC
```

Since the information needed to generate GFs is already provided, then it made sense to add the option at this stage. This generates a directory `hspec96_GFs` in the current working directory with subdirectories named by network.station.location format. Each subdirectory contains a C-shell script for that station (e.g.,) `./hspec96_GFs/KN.AML./KN.AML..wus.hspec.csh`.

The completely autogenerated script is designed to have `hspec96` generate the exact same GFs as the integrated *f*k-code provided by Yueha Zeng. Each script contains a single distance for the station, similar reduction velocity to shift the synthetic time window start time. There are 25 GF depths from 1 to 25 km in 1 km increments, the same as that specified in the `makeglib.csh` file (see `src/setupMT.c` for how depth is set). The model used is western US (`wus`) specified to `setupMT`. The `hspec96` results are run through `hpulse96` and `f96tosac` to output Seismic Analysis Code (SAC) binary files. The script finally provides a renaming of the SAC files to one that `hspec96_to_grnlib` can ingest and for the user to easily understand (i.e.,) network.station.location.model.depth.GF_component.SAC.

In addition to all the autogenerated scripts for generating `hspec96` GFs, the `setupMT` helper application also autogenerates the script named `runall_hspec96_to_glib.csh` shown in the following Figures. This script will run all the individual scripts, so the user does not have to run each manually. Note that this will take a few hours since `hspec96` does not exit out once `fmax`, a specific maximum frequency limit, has been reached as well as `kmax`, a maximum wavenumber limit is reached based on an accuracy tolerance. The final step is to read all the GFs components stored in SAC files by `hspec96` and merge them into a single `grnlib` format file for `glib2iv and mtinv`. The application `hspec96_to_grnlib` reads a par file with additional information not stored in the SAC files and outputs `grnlib` or "`*.glib`" files. The user can just copy over the "`*.glib`" files to the original working directory (..) and continue to perform the MT inversion by using the autogenerated "`run.csh`" script as usual.

```
#!/bin/csh
#########################################################
### Autogenerated C-shell script by setupMT.c        ###
#########################################################
###
### format: DIST DT NPTS T0 VRED  ## Net.Sta.Loc
###
cat >! distfile << EOF
50.8234   0.05 2048 -1.00 18.00 ## net=KN sta=AML loc=() R=50.8234 Az=30.9387 KN.AML.      nchan=1 nseg=1 (BHZ)
EOF
###
### depth file
###
cat >! depthfile << EOF
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
EOF
###
### Create Velocity Model file
###
cat >! wus.model << EOF
MODEL.01
TEST MODEL
ISOTROPIC
KGS
FLAT EARTH
1-D
CONSTANT VELOCITY
LINE08
LINE09
LINE10
LINE11
H       VP     VS   RHO  QP     QS    ETAP ETAS  FREFP FREFS
    4.00     4.52     2.61     2.39     500.0     250.0   0.0   0.0   1.0   1.0
   28.00     6.21     3.59     2.76     500.0     250.0   0.0   0.0   1.0   1.0
   20.00     7.73     4.34     3.22    1000.0     500.0   0.0   0.0   1.0   1.0
    0.00     7.64     4.29     3.19    1000.0     500.0   0.0   0.0   1.0   1.0
EOF
###
### generate hspec96.dat read by hspec96, options are -ALL -EQEX -EXF
###
# hprep96 -M wus.model -d distfile -HS 2 -HR 0 -EQEX -NDEC 1
# hprep96 -M wus.model -d distfile -FHS depthfile -HR 0 -EQEX -NDEC 1
hprep96 -M wus.model -d distfile -FHS depthfile -HR 0 -EQEX -NDEC 1 -XF 1
###
### run hspec96 and send output to hspec96.out
###
hspec96 > hspec96.out
###
### read hspec96 output greens functions hspec96.grn and write sac binary files
###
hpulse96 -i -D | f96tosac -B
###
### Rename the output file (e.g., copy B0101ZDD.sac -> NM.CGM3.00.cus.ZDD.SAC)
###
/bin/cp B0101ZDD.sac KN.AML..wus.1.ZDD.SAC
/bin/cp B0102RDD.sac KN.AML..wus.1.RDD.SAC
/bin/cp B0103ZDS.sac KN.AML..wus.1.ZDS.SAC
/bin/cp B0104RDS.sac KN.AML..wus.1.RDS.SAC

[cut short, see actual file for all copy commands]
```

File ./hspec96_GFs/KN.AML./KN.AML..wus.hspec.csh

File ./hspec96_GFs/runall_hspec96_to_glib.csh

```
#!/bin/csh


###
### do the hspec96 fk computations here
###
foreach nsl ( KN.AML. KR.ARLS. KN.EKS2. KN.UCH. KN.KBK. KN.CHM. KN.USP. KN.TKM2.
KN.ULHL. )
echo ${nsl}
cd ${nsl}
pwd
csh ${nsl}.wus.hspec.csh
cd ..
end


###
### create glib files from hspec96 output
###
cat >! hspec96_to_grnlib.par << EOF
velmod=wus
modeldb=/Users/ichinose1/Work/mtinv.v4.0.1/data/modeldb
stadb=../../Data/rdseed.stations
zrange=1,1,25
evla=41.7392
evlo=73.377
EOF

hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=AML loc="" ./KN.AML./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KR sta=ARLS loc="" ./KR.ARLS./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=EKS2 loc="" ./KN.EKS2./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=UCH loc="" ./KN.UCH./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=KBK loc="" ./KN.KBK./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=CHM loc="" ./KN.CHM./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=USP loc="" ./KN.USP./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=TKM2 loc="" ./KN.TKM2./*.SAC
hspec96_to_grnlib par=hspec96_to_grnlib.par net=KN sta=ULHL loc="" ./KN.ULHL./*.SAC
```

```
total 173248
drwxr-xr-x  509 ichinose1  50064    16288 Jul 10 16:03 KN.AML./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.AML..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:39 KN.CHM./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.CHM..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:23 KN.EKS2./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.EKS2..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:34 KN.KBK./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.KBK..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:51 KN.TKM2./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.TKM2..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:28 KN.UCH./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.UCH..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:58 KN.ULHL./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:59 KN.ULHL..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:45 KN.USP./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KN.USP..wus.glib
drwxr-xr-x  509 ichinose1  50064    16288 Jul  9 17:18 KR.ARLS./
-rw-r--r--    1 ichinose1  50064  9853904 Jul  9 17:58 KR.ARLS..wus.glib
-rw-r--r--    1 ichinose1  50064      139 Jul  9 17:58 hspec96_to_grnlib.par
-rwxrwxrwx    1 ichinose1  50064     1192 Jul  9 14:34 runall_hspec96_to_glib.csh*
ichinose1 (mtinv4_sampledata/2009-12-22T055435_Kyrgyzstan/dev/hspec96_GFs)->
```

Screen shot of the hspec96_GFs directory after running everything.

Section 4

Application examples: `sw4_to_grnlib`

The application `sw4_to_grnlib` provided in the current version of the MTINV toolkit reads SAC files output from **`sw4`** for the 10 fundamental faulting GF components using an agreed upon directory and file naming convention (conversation with Rongmao Zhou). **`sw4`** simulation was performed on the 2021 Tennessee mine collapse event (see Appendix B) and provided as an example here. The following script was created to demonstrate the application, but the user may encounter different setups, so the script provided here is just a guide that can be cloned.

The user can first reuse the `rdseed.stations`, station information file in `rdseed` format and `cus.mod`, velocity model file (**`sw4`** doesn't care about this and a dummy model is needed for filling in the file format). The par file is needed for running `sw4_to_grnlib` and this is where the user makes reference to `stadb=rdseed.stations` and `velmod=cus.mod`. The par file needs the event latitude, longitude and depth in decimal degrees and km. The parameter names are similar those used by `mkgrnlib`. The next parameters are specific to **`sw4`**. The first is the number of components `ncmp=10`. The next are the components "`cmpXX=`" where `XX=01, 02, ..., 10` which can be placed in any order. The first line "cmp01=" is comma delimited values where the first field contains the vertical component strike-slip GF named "`zss`". The next field of `"cus.sw4output.ZSS"` is the name of the directory containing the **`sw4`** output GF for this component. The next 3 fields contain the comma delimited `y,x,z` which states the e,n,z notation (e=east, n=north, z=vertical). In the case of **`sw4`**, the east-west axis is the y axis which is different from typical `x,y,z` -> n,e,z coordinate system used by many *f*κ codes or waveform simulation codes. The last 3 comma delimited fields contains the multiplication scaling factors for the GF components (in this case `y,x,z`). Making the scaling factors negative will flip the polarity.

Finally, the last few lines in the par file contain processing options. The parameter "`rotate`" rotates the horizontal components to the great-circle-path "`gcp`" or any user specified azimuth `az=` in degrees. Leaving out the rotate option in the par file or `norotate` will result in the **`sw4`** output to not be rotated which is not desirable because `mtinv` expects the GFs horizontal components to be rotated into the source-to-receiver azimuth. The option "`integrate`" will convert ground motion velocity to displacement as `mtinv` expects displacements output by most *f*κ-codes. Adding `nointegrate` or leaving out the option in the par file will default to no integration. The last option `base_moment_scaling` is provided to account for the difference between RBH and **`sw4`** use of 1.0e+20 dyne-cm compared to my use of 1.2445e+16 dyne*cm. The default is no base moment scaling and assumes all amplitudes are in units of cm displacement with base moment of Mw 0. Finally, after all the parameters are set in the par file, then the application is run with the usage: `sw4_to_grnlib par= net= sta= loc=` (see script).

```
#!/bin/csh

cat >! rdseed.stations << EOF
T50A N4 37.0204 -84.8384 302 "HHZ00 "
U49A N4 36.5129 -85.7796 234 "HHZ00 "
V48A N4 35.74 -86.8219 278 "HHZ00 "
V53A N4 35.6694 -82.8124 681 "HHZ00 "
W50A N4 35.2002 -85.3119 587 "HHZ00 "
W52A N4 35.0935 -83.9277 519 "HHZ00 "
X51A N4 34.5658 -84.8574 214 "HHZ00 "
TZTN US 36.5439 -83.549 394 "BHZ00 "
TKL IM 35.658 -83.774 351 "BHZ "
EOF

cat >! cus.mod << EOF
### Central US model
### R. B. Herrmann, H. Benz, C. J. Ammon;
### Monitoring the Earthquake Source Process in North America.
### Bulletin of the Seismological Society of America ; 101 (6): 2609–2625.
### doi: https://doi.org/10.1785/0120110095
###
# thick  vp     qp     vs       qb   rho
   1.0  5.00   581.0  2.89   258.0   2.50
   9.0  6.10   625.0  3.52   275.5   2.73
  10.0  6.40   671.1  3.70   275.5   2.82
  20.0  6.70   671.1  3.87   297.6   2.90
 700.0  8.15   515.4  4.70   232.0   3.36
EOF

cat >! sw4_to_grnlib.par << EOF
noverbose
stadb=./rdseed.stations
#
evla=35.8767
evlo=-84.898
evdp=1.0
#
# ot=2021-08-13T11:57:35
#
velmod=cus
modeldb=./
#
## cmp info, and amplitude scaling
#
ncmp=10
######code,directory,ew,ns,z,amp scaling
cmp01=zss,cus.sw4output.ZSS,y,x,z,1,1,1
cmp02=zds,cus.sw4output.ZDS,y,x,z,1,1,1
cmp03=zdd,cus.sw4output.ZDD,y,x,z,1,1,1
cmp04=zex,cus.sw4output.ZEX,y,x,z,1,1,1
cmp05=rss,cus.sw4output.ZSS,y,x,z,1,1,1
cmp06=rds,cus.sw4output.ZDS,y,x,z,1,1,1
cmp07=rdd,cus.sw4output.ZDD,y,x,z,1,1,1
cmp08=rep,cus.sw4output.ZEX,y,x,z,1,1,1
cmp09=tss,cus.sw4output.TSS,y,x,z,1,1,1
cmp10=tds,cus.sw4output.TDS,y,x,z,1,1,1
##
## processing, rotate to great-circle-path
##             integrate from velocity to displacement
##             scale amplitudes using reference or base moment
##
rotate gcp # or az=XXX
integrate
##
## no scaling, comparing SW4 output processes with SAC with itself processed using this code
##
#base_moment_scaling rbh_to_yz
nobase_moment_scaling
EOF
```

[script contents continues on next page]

```
# [script contents continued from previous page]

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=T50A loc=00
grnlib2sac z=1 glib=N4.T50A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=U49A loc=00
grnlib2sac z=1 glib=N4.U49A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=V48A loc=00
grnlib2sac z=1 glib=N4.V48A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=V53A loc=00
grnlib2sac z=1 glib=N4.V53A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=W50A loc=00
grnlib2sac z=1 glib=N4.W50A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=W52A loc=00
grnlib2sac z=1 glib=N4.W52A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=N4 sta=X51A loc=00
grnlib2sac z=1 glib=N4.X51A.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=US sta=TZTN loc=00
grnlib2sac z=1 glib=US.TZTN.00.cus.glib dumpgrn

sw4_to_grnlib par=sw4_to_grnlib.par net=IM sta=TKL loc=""
grnlib2sac z=1 glib=IM.TKL..cus.glib dumpgrn
```

File: mtinv4_sampledata/mtinv_sw4/run_sw4_to_grnlib.csh

Section 5

Moment tensor-based Greens function formulations

Application examples: `grn2Mxy` *and special load options for* `glib2inv` *and* `mtinv`

Moment tensor estimation is ideally formulated to be a linear inverse problem between the GFs and the moment tensor elements. Helmberger and Langston (H&L1975) formatted an approach that is based on the 3 fundamental faults (see Appendix A: SS, DS, DD and isotropic EX). This results in 10 components (cmps) because ground motions are recorded in 3D coordinate system (Z, R, T). There are 10 cmps because (3 faults + 1 isotropic) * 3 cmps = 12 but the DD and EX GF components for the transverse cmp are zero and left out leaving just 10 components.

The alternative approach is to output the Greens functions using the six moment tensor elements (Mxx, Myy, Mzz, Mxy, Mxz, Myz). This results in 3 cmps * 6 MT elements = 18 cmps; therefore, it was probably decided back in 1975 that it was easier to deal with 10 instead of 18 cmps. The flip side is that the H&L1975 convention is not unique and there are 2 other known conventions (e.g., Kikuchi and Kanamori; C. Ammon). Mistakes with mixing the conventions or coordinate systems can be made and so the moment tensor approach for GFs is a way around this as long as the orientation of the moment tensor is consistent with the inversion code. In the case of `mtinv`, the moment tensor is in cartesian coordinate system with the y-axis pointing positive north and x-axis positive east (also known as “NED” see Appendix A for fault and slip vector orientations).

To accommodate this approach, we provide a code `grn2Mxy` to convert the GFs in `grnlib` format to moment tensor Mij format. Additionally, the codes `glib2inv` and `mtinv` have options to load this formatted GFs. The following script follows an example provided in the sample data distribution for the 2021-11-18 New Madrid earthquake.

The `grn2Mxy` application requires the GF file `glib=` and one of the `z=` computed depths. The application also requires a parameter `mtdegfree=5` (default). Selecting `5` for the deviatoric moment tensor option, which is the default option, uses the xx and yy GFs to generate the zz component, so those components are zero traces (i.e., Mzz=-(Mxx+Myy)). Selecting `mtdegfree=6` gives the full moment tensor option, and, in this case, all 18 components are computed. The output is placed in a directory based on the station-code name so be careful when using stations with different location codes. Then the applications `glib2inv` requires the `test_special` option added to the `run.csh` script. The `run.csh` script also requires the `special` option added. All other parts of the `run.csh` script is unchanged. Currently the application of this approach is limited to just one depth. In the future, if this option becomes popular, then I would be willing to add full depth library support for this GF format.

```
#!/bin/csh

################################################################################
#select events from wilbur and download from iris
################################################################################

unpack.csh 2021-11-18-new-madrid-missouri-region.tar

################################################################################
# deviatoric-MT solution
################################################################################
#cd dev
mkdir dev_grnMxy_test
cd dev_grnMxy_test

## uncomment approriate line or add new event information
# vi setupMT.csh
# setupMT.csh

setupMT -z 16 -ev 36.9077 -90.543 -mod cus -ot "2021-11-18T02:53:04" -com "New Madrid, MO"
-fil 0.075 0.150 ../Data/*Z.?.SAC ../Data/*Z.SAC

################################################################################
## after setupMT.csh run the following to compute the greens functions and
## run the moment tensor inversion
################################################################################

makeglib.csh

run.csh

################################################################################
###   write the greens function out into different format
### Create test Greens functions using glib files for depth of 14 km
###  this depth is the best fit given an origin time of 2021-11-18T02:53:04
################################################################################

foreach i ( *.glib )
 grn2Mxy mtdegfree=5 z=14 verbose glib=$i
end

### add these two line in file run.csh replacing old ones
###
#
# glib2inv par=mtinv.par noverbose parallel test_special
#
# mtinv special ts0=0 mtdegfree=5 par=mtinv.par  verbose AutoAuth gmt5 use_snr minsnr=3 shift
ctol=0.85 maxshift=10 >> mtinv.out
# mtinv special ts0=0 mtdegfree=6 par=mtinv.par  verbose AutoAuth gmt5 use_snr minsnr=3 shift
ctol=0.85 maxshift=10 >> mtinv.out

cd ..
cp -Rp dev_grnMxy_test full_grnMxy_test
cd full_grnMxy_test

/bin/rm -r ??? ????
foreach i ( *.glib )
 grn2Mxy mtdegfree=6 z=14 verbose glib=$i
end

### add this line in file run.csh replacing old one
###
# mtinv special ts0=0 mtdegfree=6 par=mtinv.par  verbose AutoAuth gmt5 use_snr minsnr=3 shift
ctol=0.85 maxshift=10 >> mtinv.out
```

Appendix A

Table A1. Greens function convention for the 10 fundamental fault orientations. These values were verified using fk-to-fk comparisons at multiple depths and 1D earth velocity models. Az is the source-to-receiver azimuth in degrees.

COMP	STR	DIP	RAK	MXX	MYY	MZZ	MXY	MXZ	MYZ	AZ
ZSS, RSS	0	90	0	0	0	0	+1	0	0	45
TSS	0	90	0	0	0	0	+1	0	0	90
ZDS, RDS	0	90	-90	0	0	0	0	0	+1	90
TDS	0	90	+90	0	0	0	0	0	-1	0
ZDD, RDD	0	45	+90	0	-1	+1	0	0	0	45
ZEX, REX	N/A	N/A	N/A	+1	+1	+1	0	0	0	any

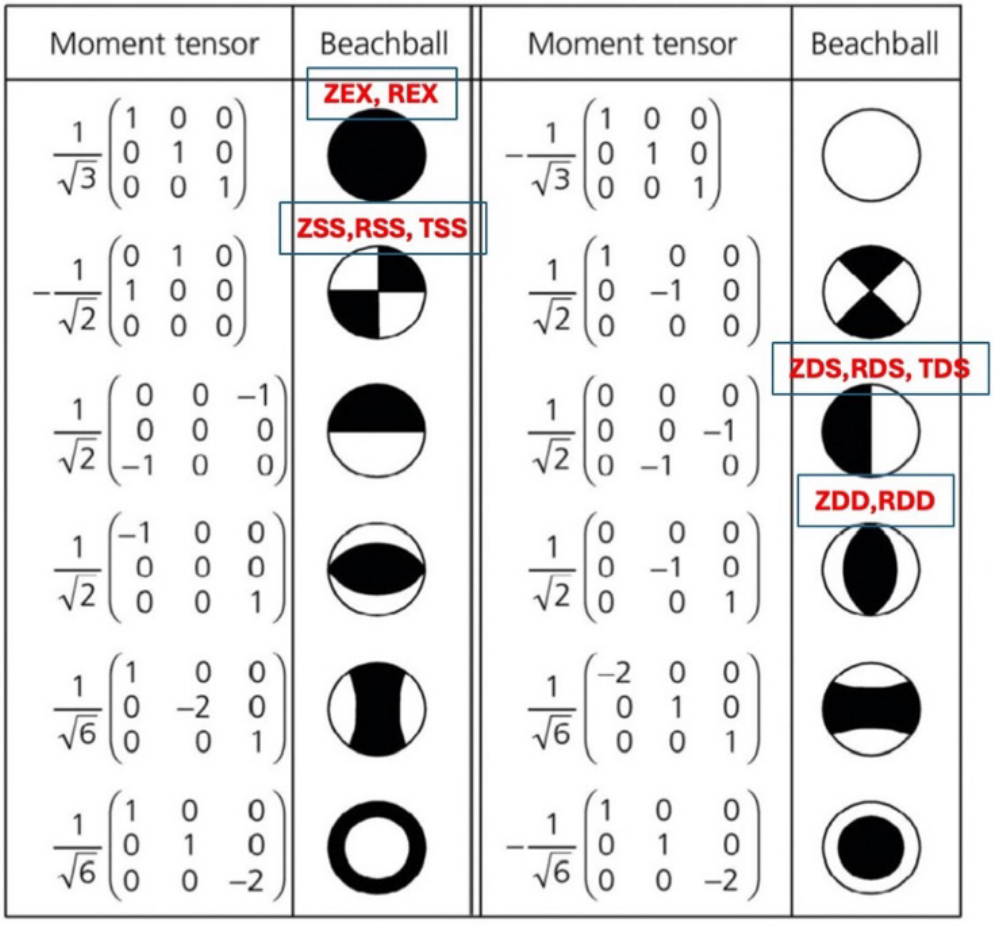


Figure A1. From Seth Stein (Mechanisms and Moment Tensors with various rotations of the eigenvector orientations).
Row 1 EXP (Mxx=+1 Myy=+1 Mzz=+1) on the left column,
Row 2 SS (Mxy=+1) on the left column,
Row 3 DS (Myz=-1) on the right column,
Row 4 DD (Myy=-1 Mzz=+1) on the right column

Figure A2. Comparisons between RBH vs Y.Zeng GFs at 2 km depth. Depth not at layer interface, depth=2 km is inside layer 2 of cus model GFs components comparisons are good within a few percent. The information required to recreate this figure is shown in appendix B.

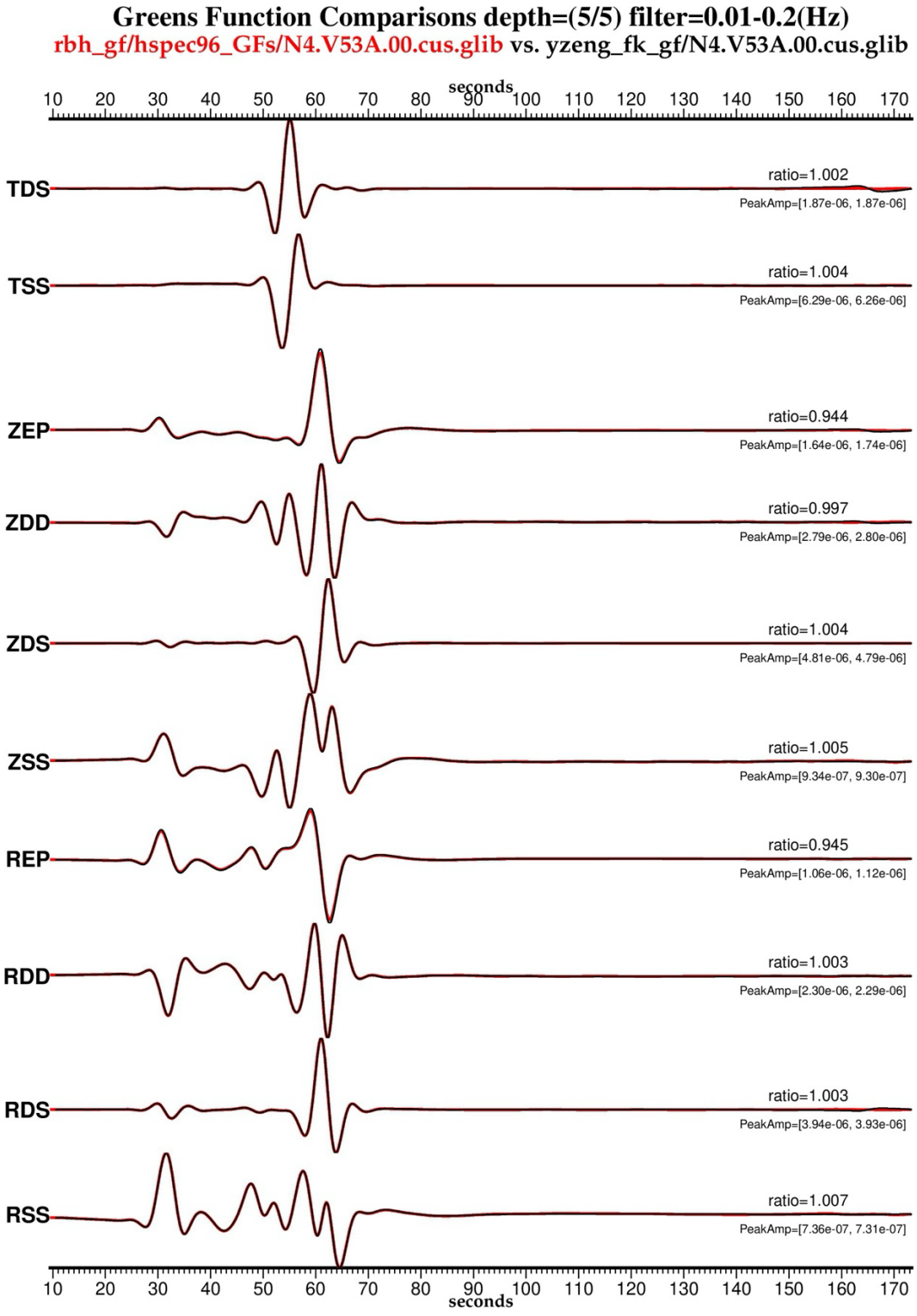


Figure A3. Comparisons between RBH vs Y.Zeng GFs at 5 km depth. Depth not at layer interface, depth=5 km is inside layer 2 of cus model GFs components comparisons are good within a few percent. The information required to recreate this figure is shown in appendix B.

Appendix B

The following information is required to recreate the appendix figures A2 and A3.

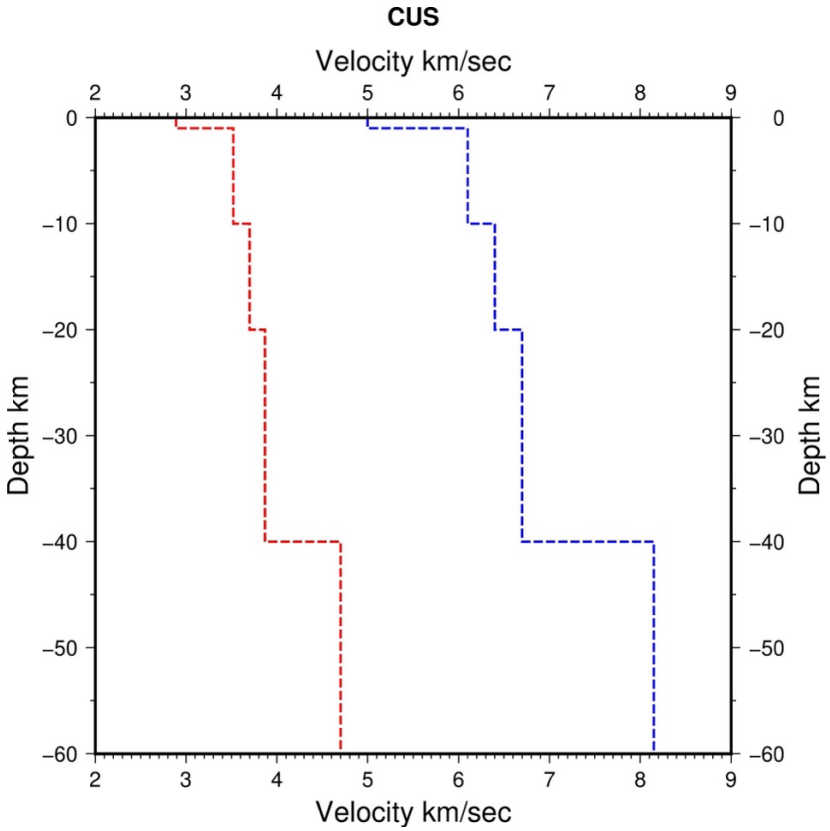
Event Location

Event: Franklin Mine, TN (mine collapse) usgs link event_id=se6013353
Origin-Time: 2021-08-13T11:57:35
Location: Lat,Lon,depth: 35.8767, -84.898, 2km (based on collapse crater in Google Earth photos)

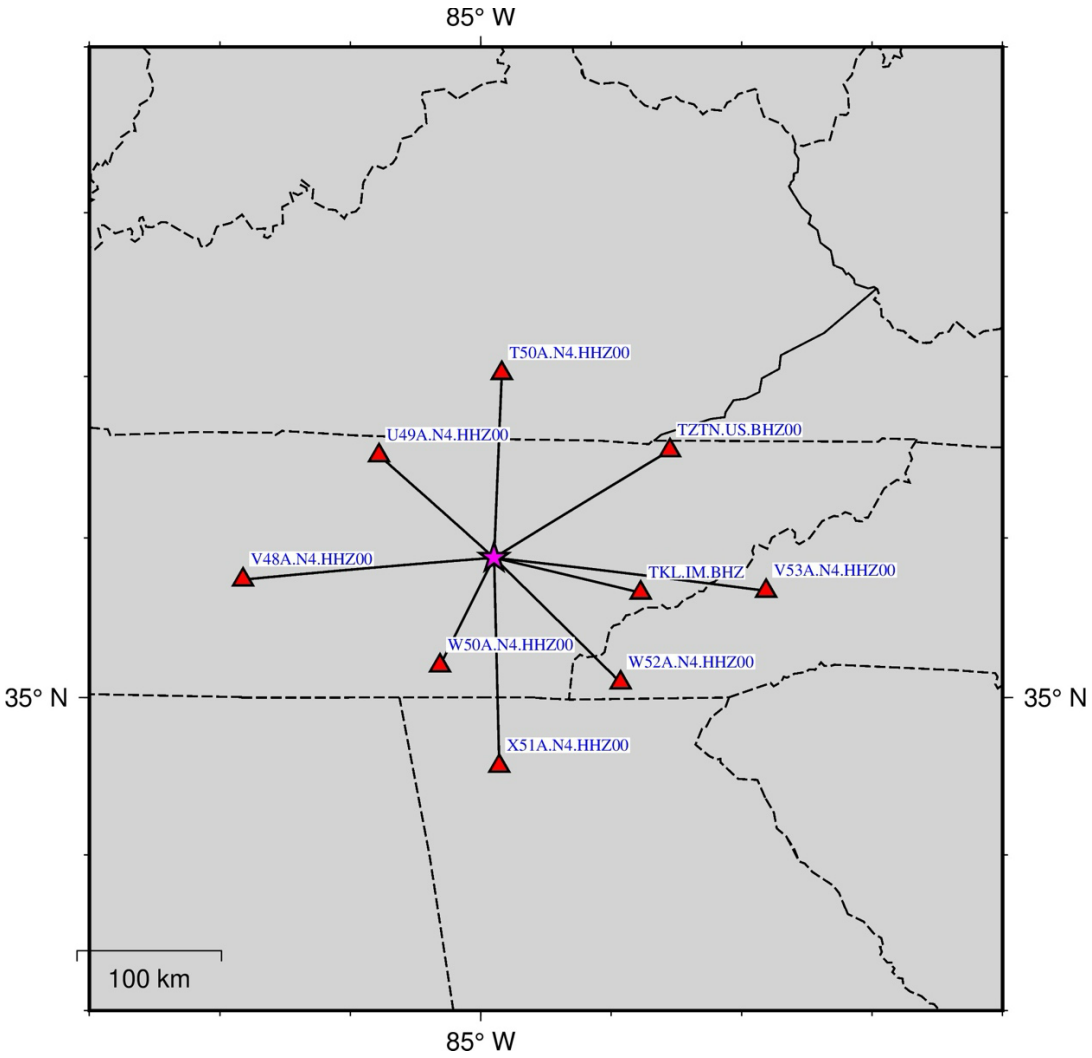
1D Earth Velocity Model

CUS model (central US)

thickness km	depth_to_top km	Vp km/s	Qp	Vs km/s	Qs	Density g/cc	sigma
1.00	0.00	5.00	581.00	2.89	258.00	2.50	0.25
9.00	1.00	6.10	625.00	3.52	275.50	2.73	0.25
10.00	10.00	6.40	671.10	3.70	275.50	2.82	0.25
20.00	20.00	6.70	671.10	3.87	297.60	2.90	0.25
700.00	40.00	8.15	515.40	4.70	232.00	3.36	0.25



Station Locations



net	sta	lat	long	Elev (m)	channels	distance (km)	azimuth
N4	T50A	37.0204	-84.8384	302	HH100, HH200, HHZ00	127.192	2.38932
N4	U49A	36.5129	-85.7796	234	HH100, HHZ00, HH200	106.037	312.07
N4	V48A	35.74	-86.8219	278	HH100, HH200, HHZ00	174.004	265.557
N4	V53A	35.6694	-82.8124	681	HH100, HH200, HHZ00	189.429	96.3723
N4	W50A	35.2002	-85.3119	587	HH100, HH200, HHZ00	83.9557	206.579
N4	W52A	35.0935	-83.9277	519	HH100, HH200, HHZ00	123.616	134.46
N4	X51A	34.5658	-84.8574	214	HH100, HH200, HHZ00	145.697	178.533
US	TZTN	36.5439	-83.549	394	BH100, BH200, BHZ00	141.862	58.0971
N4	U49A	36.5129	-85.7796	234	HH100, HHZ00, HH200	106.037	312.07
IM	TKL	35.658	-83.774	351	BHE, BHN, BHZ	104.219	103.152

Below is a quick reference to show how the A-matrix is generated (see src/make_amatrix.c for full version). MTINV is based on this original Helmberger and Langston (1975) GF formulation which was updated by Herrmann and Hutchenson (1993) and Minson and Dreger (2008).

```
/*****************************************************************************/
/*** this version of make_amatrix uses the formaution of Herrmann and Hutchenson ***/
/*** (1993) and can be used for both deviatoric and full moment tensors          ***/
/***  not Jost&Herrmann(1989) - see Minson and Dreger (2008) GJI                 ***/
/***                                                                             ***/
/*** Note that: synthetics = A_matrix * MomentTensor                             ***/
/***                                                                             ***/
/*** fi is short for phi, the source-to-receiver azimuth                         ***/
/*****************************************************************************/

/*** Transverse components ****/

a_matrix[irow][1] = (  half * sin(2*fi) * tss[it] );  /* Mxx */
a_matrix[irow][2] = ( -half * sin(2*fi) * tss[it] );  /* Myy */
a_matrix[irow][3] = (        -cos(2*fi) * tss[it] );  /* Mxy */
a_matrix[irow][4] = (           sin(fi) * tds[it] );  /* Mxz */
a_matrix[irow][5] = (          -cos(fi) * tds[it] );  /* Myz */
a_matrix[irow][6] = 0;                                /* Mzz */

/*** radial component ***/

a_matrix[irow][1] = ( +half * cos(2*fi) * rss[it] -sixth*rdd[it] + third*rep[it] ); /* Mxx */
a_matrix[irow][2] = ( -half * cos(2*fi) * rss[it] -sixth*rdd[it] + third*rep[it] ); /* Myy */
a_matrix[irow][3] = (         sin(2*fi) * rss[it] );                                /* Mxy */
a_matrix[irow][4] = (           cos(fi) * rds[it] );                                /* Mxz */
a_matrix[irow][5] = (           sin(fi) * rds[it] );                                /* Myz */
a_matrix[irow][6] = ( third * rdd[it] + third * rep[it] );                          /* Mzz */

/*** vertical component ***/

a_matrix[irow][1] = ( +half * cos(2*fi) * zss[it] -sixth*zdd[it] + third*zep[it] ); /* Mxx */
a_matrix[irow][2] = ( -half * cos(2*fi) * zss[it] -sixth*zdd[it] + third*zep[it] ); /* Myy */
a_matrix[irow][3] = (         sin(2*fi) * zss[it] );                                /* Mxy */
a_matrix[irow][4] = (           cos(fi) * zds[it] );                                /* Mxz */
a_matrix[irow][5] = (           sin(fi) * zds[it] );                                /* Myz */
a_matrix[irow][6] = ( third * zdd[it] + third * zep[it] );                          /* Mzz */
```