Time-Domain Moment Tensor INVerse Code (TDMT_INVC)
Release 1.1

with

Time-Domain Moment Tensor with ISOtropic INVerse Code (TDMTISO_INVC)
Release 1.1

by

Douglas S. Dreger

August 8, 2002 (TDMT_INVC)
May 8, 2007 (TDMTISO_INVC)

Time-Domain Moment Tensor INVerse Code (TDMT_INVC)

Introduction

This seismic moment tensor inverse software package has been in use at the University of California, Berkeley Seismological Laboratory (BSL) since 1993 and is employed to automatically investigate all M_L>3.5 events in northern California (e.g. www.seismo.berkeley.edu/~dreger/mtindex.html). The package has been successfully implemented at the Japan National Research Institute for Earth Science and Disaster Prevention (NIED; argent.geo.bosai.go.jp), and has been used by individual researchers in the United States, Europe and Asia. This distribution includes a set of programs for calculating a library of Green's functions, inverting broadband data for the seismic moment tensor, various data processing utilities and shell scripts, and code and scripts demonstrating how to automate the procedures. Complete source code is provided. The frequency-wavenumber integration program (FKRPROG) written by Chandan Saikia of URS is included with his permission. This software may be distributed only by means of its gzipped tar file. There is no warranty either expressed or implied, and the author is not responsible for damages due to misinterpretation of results and misuse of the software.

Finally, this software is freely distributed for non-commercial use. The reporting of seismic moment tensor solutions obtained using this software on the WWW and in the technical literature should include the following acknowledgement statement. "Moment tensors were computed using the mtpackagev1.1 package developed by Douglas Dreger of the Berkeley Seismological Laboratory, and Green's functions were computed using the FKRPROG software developed by Chandan Saikia of URS."

In the following the users manual documents how to build the software package, demonstrates usage through a series of examples, provides references to studies illustrating how suitable velocity models may be determined, and describes how the method may be automated.

Building the Program

Download the package compressed tar file via request from **http://www.seismo.berkeley.edu/~dreger**. Uncompressing and extracting the files will produce a directory structure with a root directory named MTPACKAGE and various subdirectories. The source code and Makefile are located in the MTCODE subdirectory. The software uses **Numerical Recipes for C**, and the **Seismic Analysis Code (SAC),** which must be obtained separately by the user. When building the numerical recipes library, nrcv2.a, it is necessary to use the standard C option as opposed to ANSI C.

Before compiling, copy the file makefile_dev to makefile and set the Numerical Recipies for C, "NRC", variable in the makefile with the directory path pointing to the nrcv2.a library file. Use the command "make all" to build the software package. The distribution compiles using the GNU c (gcc) and GNU fortran (g77) compilers on PCs running Linux and SUN workstations running Solaris. The ‘make cleanup’ command will remove all program files, libraries, and object code.

After the build, all of the executables are located in the MTPACKAGE/MTCODE/BIN subdirectory. The README file in that directory identifies shell scripts that **CANNOT** be rebuilt if removed.

Basic Methodology and Program Assumptions

The general representation of seismic sources is simplified by considering both a spatial and temporal point-source.

$$U_n(x,t) = M_{ij} \cdot G_{ni,j}(x,z,t)$$

U_n, is the observed nth component of displacement, $G_{ni,j}$ is the nth component Green’s function for specific force-couple orientations, and M_{ij} is the scalar seismic moment tensor, which describes the strength of the force-couples. The general force-couples for a deviatoric moment tensor may be represented by three fundamental-faults, namely a vertical strike-slip, a vertical dip-slip, and a 45^0 dip-slip. The indices i and *j* refer to geographical directions. The above equation is solved using linear least squares for a given source depth. In this distribution only the deviatoric seismic moment tensor is solved for, and the inversion yields the M_{ij} which is decomposed into the scalar seismic moment, a double-couple moment tensor and a compensated linear vector dipole moment tensor. The decomposition is represented as percent double-couple (Pdc) and percent CLVD (PCLVD). Percent isotropic (PISO) is always zero for this deviatoric application. The double-couple is further represented in terms of the strike, rake and dip of the two nodal planes. The basic methodology and the decomposition of the seismic moment tensor is described in Jost and Herrmann (1989).

Source depth is found iteratively by finding the solution that yields the largest variance reduction,

$$VR = \left[1 - \sum_i \sqrt{(data_i - synth_i)^2} \Big/ \sqrt{data_i^2}\right] * 100,$$

where *data*, and *synth* are the data and Green’s function time series, respectively, and the summation is performed for all stations and components.

Another measure that is useful for determining source depth in regions where explosive events are unlikely is the RES/Pdc, the variance divided by the percent double-couple where,

$$RES/Pdc = \sum_i \sqrt{(data_i - synth_i)^2} \,/\, Pdc$$

Dividing the variance by the percent double-couple tends to deepen the minimum.

It is assumed that the event location is well represented by the high frequency hypocentral location, and a low frequency centroid location is not determined. Second, the simplified representation above assumes that the source time history is synchronous for all of the moment tensor elements and that it may be approximated by a delta function. These assumptions are generally reasonable for $M_w<7.5$ events since long period waves (>10-20s) are used. It is noted however, that for larger events these point-source assumptions break down in the period range employed and alternative finite fault approaches (e.g. Dreger and Kaverina, 2000) or longer period waves and larger source-station distances (e.g. Fukuyama and Dreger, 2000) are required.

Finally, it is assumed that the crustal model is sufficiently well known to explain low frequency wave propagation. This software package will not work in a region if calibrated velocity models are not available. Calibrating velocity models to obtain a robust catalog of Green's functions is singly the most important step in successful seismic moment tensor applications.

In California, we have found that three 1D velocity models are adequate for the recovery of the seismic moment tensor. Different monitoring regions may require fewer or more crustal velocity models. Crustal velocity models that are sufficient for moment tensor analysis may be derived from models used to locate earthquakes, or by modeling of the broadband seismograms. There are numerous papers in the literature that describe how to model 3-component waveforms to constrain velocity structure and these (e.g. Dreger and Helmberger, 1990, 1993; Dreger and Romanowicz, 1994; Rodgers et al., 1999; Zhao and Helmberger, 1991; Song et al., 1996) are some examples for getting started. Two- and three-dimensional models may be used provided that the codes used to synthesize the Green's functions produces the full compliment of fundamental-fault responses.

In the following the usage of the software is demonstrated through a series of three examples.

Example 1

The objective of example 1 is to become familiar with computing Green's function files, performing post-processing filter operations, and using the seismic moment tensor inversion code by inverting synthetic waveform data. The files necessary for example 1 are found in MTPACKAGE/EXAMPLE_1. You should use this directory as your working directory. In this directory you will find the following files; tdmt.config.linux, tdmt.config.sun, MODEL.socal, run_parallel, run_fkrsort, run_filter, b2s.par, mt_inv.in, and testdata[1-3].

Before doing anything else set the system specific environment parameters. You will need to edit either the tdmt.config.linux or tdmt.config.sun files to set the environment variable MTPACKAGE to the full directory path of the software package. After making the changes use the source program to invoke the environment variables (i.e. source tdmt.config.linux). This command sets a number of environment variables such as paths to executables and Green's function files, which will be clarified later on.

Step 1: Compute Green’s functions

The file, MODEL.socal, contains a sample 1D velocity structure appropriate for the Sierra Nevada and Southern California regions. The contents of this file are listed below.

MODEL.socal:

```
.F.
    0   64
GREEN.X
    6.0      8.00       X  512 1024    0.500     5    Y
    1    1    1    1    1    1    1    1    1    1    0
 0.5500E+01 0.5500E+01 0.3180E+01 0.2400E+01   600.00    300.00
 0.2500E+01 0.6300E+01 0.3640E+01 0.2670E+01   600.00    300.00
 0.8000E+01 0.6300E+01 0.3640E+01 0.2670E+01   600.00    300.00
 0.1900E+02 0.6700E+01 0.3870E+01 0.2800E+01   600.00    300.00
 0.4000E+03 0.7800E+01 0.4500E+01 0.3300E+01   600.00    300.00
    3
  0.4000000E+03  1.500000E+00         0
    1  10000.0     30.0      2.9       2.5
100.00      0.0         10.0
```

The first line of MODEL.socal is a debugging flag. Keep it set to ".F.". The second line specifies the frequency range for debugging, do not change. GREEN.X is the name of the output file. The fourth line specifies alpha (small complex number for integration stability), source depth (km), the starting frequency indice (X), the total number of frequencies (a power of two), the total number of time points (2 times the number of frequencies), the sample rate (seconds), the total number of layers, and the number of processors the code will be run on (Y). The values of X and Y are set by the preprocessing script, run_parallel (usage is described below).

Line 5 is a series of flags that turn on the various fundamental fault calculations. Do not change this line. Lines 6-10 list the model parameters such as layer thickness (km), p-velocity (km/s), s-velocity (km/s), density (g/cc), Q-alpha, and Q-beta. Note that the source must be located at an artificial boundary where the velocities above and below are the same (layers 2 and 3 , lines 7 and 8, in the example). Line 11 gives the layer number below the source. Line 12 does not need to be changed. Line 13 specifies the number of stations to calculate (1 in this case), and the phase velocity window of the integration. There is no need to change the phase velocity values 10000.0 and 30.0 (km/s), but you will want to be sure that the other two are less than the Rayleigh wave velocity of the model. The lines that follow define the distance to the station, a delay time (not currently implemented), and a reduction velocity. It is recommended that you keep the reduction velocity set to 8 km/s. Green's functions produced by the code with these settings will begin at t=distance/(reduction velocity). The file as it is currently set up will generate Green's function responses for stations located at a distance of 100 km.

The first task of this example is to run the frequency-integration program (FKRPROG) through the run_parallel script to compute the Green’s functions used to invert the synthetic waveform data. **Very important note**; FKRPROG requires formatted input. Be careful not to change the column location of each field. Integer input is right justified, and floating point input should retain the position of the decimal point.

To generate the Green’s functions first run the run_parallel script. This script performs all of the steps required to generate the GREEN.* output files. The usage is:

```
run_parallel number_of_cpus name_of_inputfile
```

For example using the command “run_parallel 1 MODEL.socal” will first generate a new input file, MODEL1, which has the X and Y parameters each set to 1. Second, FKRPROG is launched generating an output file, GREEN.1 If the number of CPUs is larger than 1 (maximum value of 4) then multiple input files, MODEL[1-4], are created and FKRPROG is executed multiple times generating the corresponding GREEN.[1-4] files.

Step 2: Post-processing to create Green’s functions in usable ascii format

Once the FKRPROG process(es) have finished the next step is to create ascii format, time domain Green’s function files that will be used by the inversion code. Another script, run_fkrsort, is used for this purpose. This script first calls the wvint9 program, which reads the GREEN.* file(s) and performs an inverse FFT and writes all of the displacement (cm) Green’s functions into a headerless binary file, vec. Note that options for wvint9 are hard-coded in the script. It is possible to output velocity (cm/s) Green’s functions instead of displacement (cm) by changing the ‘d’ to ‘v’ in the standard input statement for wvint9. Subsequent to the wvint9 post-processing the specific time series are then extracted from the vec file and an eight component ascii file is constructed. The ordering of the components are as follows, and the same as in Jost and Herrmann (1989); transverse component vertical strike-slip (tss) and vertical dip-slip (tds) faults, radial component vertical strike-slip (xss), vertical dip-slip (xds) and 45-degree dip-slip (xdd) faults, and vertical component vertical strike-slip (zss), vertical dip-slip (zds), and 45-degree (zdd) faults.

The following is the usage for run_fkrsort.

```
run_fkrsort filename_prefix distance depth number_of_distances
```

In this example use ‘run_fkrsort socal 100 8 1’

This script will output the files with the naming convention, {filename_prefix}{distance}d{depth}.disp. If the number_of_distances is not equal to 1 then the script assumes that the distance given is the closest distance, and the following distances are for intervals of 5 km. Note if you chose to modify run_fkrsort so that it outputs velocity (see above) you should also change the file extension for the Green’s function files to .vel.

Step 3: Filter Green’s functions

Once the *.disp files have been created it is necessary to perform bandpass filtering using the run_filter script. The usage of run_filter is:

```
run_filter  infile_name outfile_name  high_pass low_pass
```

The infile_name is the output filename from the run_fkrsort script. The outfile_name can be anything, but typically the infile_name with the '.disp' truncated is used. The high_pass and low_pass filter corners are given in hertz. An acausal (two pass), 4th order butterworth filter is applied using SAC. The SAC commands are imbedded in the run_filter script.

In this example the Green's functions need to be bandpass filtered between 0.02 to 0.10 Hz. To do this use 'run_filter socal100d8.disp socal100d8 0.02 0.10'

Step 4: Run Moment Tensor Inversion

Once the filtered Green's function files have been computed you can use them to invert the test data for example 1. You will find three data files, testdata[1-3], in the working directory. These data files are listed in the provided ***tdmt_invc*** input file called mt_inv.in.

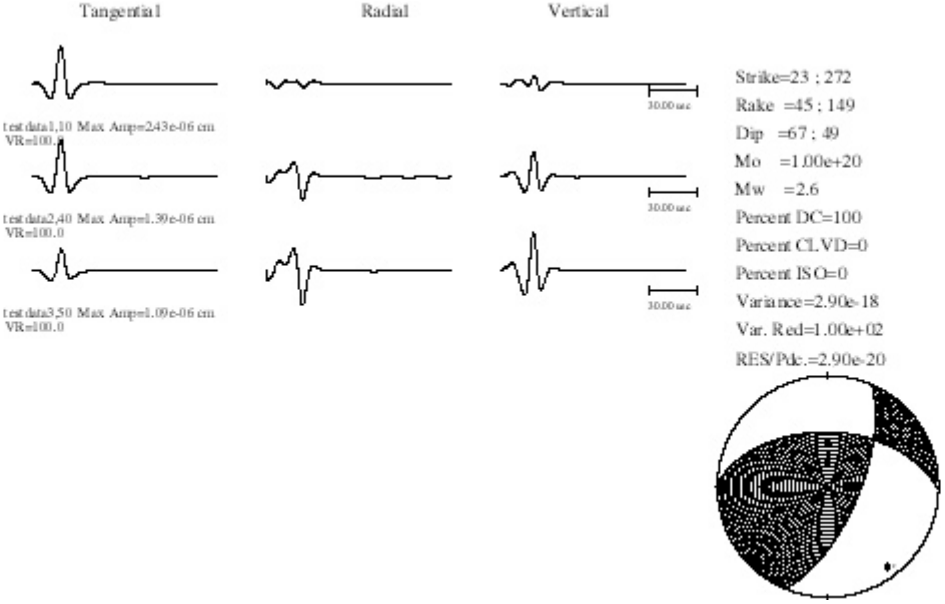
mt_inv.in has the following format

```
3  8 1 1                       <number of 3-component stations, source depth, distance weighting flag, plotting flag>
testdata1 100. 10. 0 120       <data_filename, distance (km), azimuth (deg from north), sample-offset (Zcor), number_of_samples>
testdata2 100. 40. 0 120       <ditto for station 2>
testdata3 100. 50. 0 120       <ditto for station 3>
socal100d8 0 120               <filtered GF_filename, zero-offset (always zero), number_of_samples (same as corresponding data>
socal100d8 0 120               <ditto for station 2>
socal100d8 0 120               <ditto for station 3>
```

To run the moment tensor code execute the command "$REDI_MT_BINDIR/tdmt_invc > plot", where plot is a plot file, that may be converted to postscript using "psigl < plot > plot.ps" When tdmt_invc is run the following information is output to the screen, and to a log file (mt_inv_redi.out).

```
Depth=8
Station Information
Station(0): testdata1  R=100.0km  AZI=10.0  W=1.000  Zcor=5
Station(1): testdata2  R=100.0km  AZI=40.0  W=1.000  Zcor=5
Station(2): testdata3  R=100.0km  AZI=50.0  W=1.000  Zcor=6
Mo=9.13262e+19
Mw=2.6
Strike=20 ; 275
Rake=39 ; 155
Dip=70; 54
Pdc=92
Pclvd=8
Piso=0
Station(0)=99.418358  2.93047e-11
Station(1)=99.254257  1.7449e-11
Station(2)=83.429100  1.33789e-11
VAR=2.34412e-15
VR=95.81  (UNWEIGHTED)
VR=95.81  (WEIGHTED)
Var/Pdc=2.553e-17
Quality=4
```

Most of the output information is self-explanatory. W is the applied inverse distance weight. Since in this case the distance to the three test stations is the same the weight is the same. If the distances differ the more distant stations are given a larger weight. Zcor is the sample offset that the code obtains from cross-correlating the data with the fundamental fault Green’s functions (tss, tds, etc.). Zcor is a very important parameter that is used to align the data with the Green’s functions prior to inverting the data. Because the cross-correlation is against fundamental fault Green’s functions its value should be checked. Typically testing values that are +- 3 samples from the value obtained automatically is sufficient for finding the optimal value, but sometimes when the data is noisy or the velocity model used to construct the Green’s functions is very approximate a larger search may be necessary. The optimal value is evaluated using the variance reduction for each station, where Station(0)=99.418358 is the variance reduction for the first station. Try changing the value of the offset by editing the sample_offset field (Zcor) in the mt_inv.in file and rerunning the tdmt_invc program. Figure 1 shows the best result that is obtainable.



```
Strike=23 ; 272
Rake  =45 ; 149
Dip   =67 ; 49
Mo    =1.00e+20
Mw    =2.6
Percent DC=100
Percent CLVD=0
Percent ISO=0
Variance=2.90e-18
Var. Red=1.00e+02
RES/Pdc.=2.90e-20
```

Figure 1. Example of ***tdmt_invc*** graphical output using Green’s functions for a source depth of 8 km. The data are shown as solid lines, the synthetics are dashed. For each station three-component data and synthetics are compared, and the azimuth (10, 40, and 50-degrees), the maximum three-component trace amplitude, and variance reduction (VR) are provided. Solution information includes the strike, rake and dip for the two possible double-couple planes, the scalar seismic moment, and Mw. Information about the moment tensor decomposition in terms of percent double-couple (DC), CLVD, and isotropic (ISO)

are also listed. Fitting parameters such as the variance, the variance reduction (Var. Red), and the variance modulated by the percent double-couple (RES/Pdc). To obtain this result it was necessary to manually shift the Green's functions for station 3 (testdata3) by one sample.

The output parameters VAR, VR, Var/Pdc and Quality are used to gauge the success of the inversion. VAR is the overall variance estimate, VR is the variance reduction (both unweighted and distance weighted estimates), Var/Pdc is the ratio of the variance to the percent double couple, quality is a subjective measure where 4 is the best and 1 is the worst. The higher the value of VR the better the solution. The Var/Pdc measure can also be very useful for areas where non-double-couple solutions are not expected.

To determine the source depth inversions are performed over a range of source depths taking the best solution as either the one with the greatest overall variance reduction or smallest Var/Pdc measure. Figure 2 illustrates the variance reduction plotted against source depth for this synthetic test example.

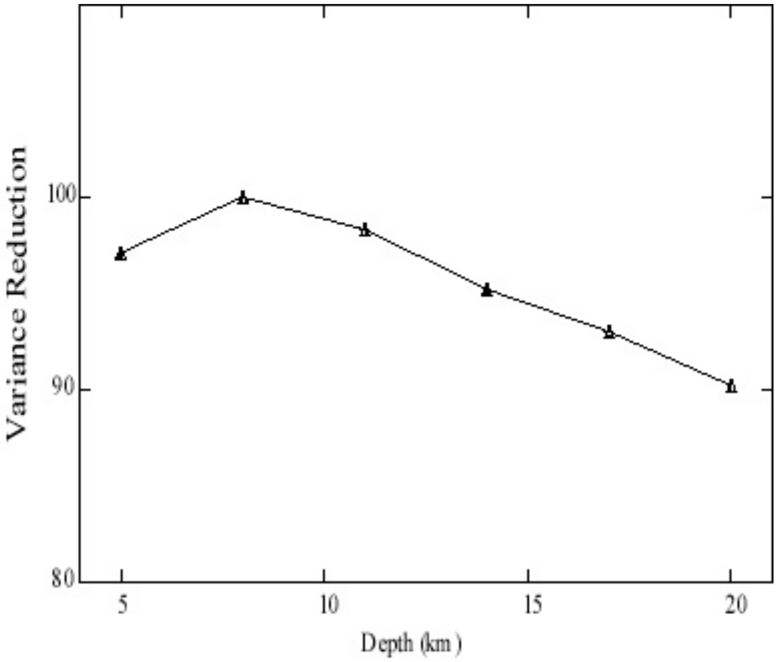


Figure 2. Variance reduction is plotted against source depth. For each inversion the Green's function alignment parameter has been optimized.

Example 2

The objective of this example is to apply the data processing scripts to broadband data for a real earthquake, generate suitable Green's functions, and invert the data for the seismic moment tensor. In the MTPACKAGE/EXAMPLE_2 subdirectory you will find the following files; tdmt.config.linux, tdmt.config.sun, mt_inv.in, run_parallel, run_fkrsort, run_filter, MODEL.gil7, b2s.par. In addition, there are two subdirectories, DATA_PC and DATA_SUN, which contain SAC binary files for PC and SUN platforms, respectively. The files have the same functionality as in Example 1. MODEL.gil7 is an input file for FKRPROG to compute Green’s functions that are suitable for the Coast Ranges of central and northern California.

Before doing anything else edit either the tdmt.config.linux or tdmt.config.sun files to set the environment variable MTPACKAGE to the full directory path of the software package, and then source the configuration file that is suitable for your operating environment.

To begin copy the binary data files for stations BKS, CMB, KCC and PKD from the appropriate data directory to your working directory. These stations will be used to illustrate the usage of the program. The following files should have been copied;

```
19980812141000.BKS.BHE 19980812141000.BKS.BHN 19980812141000.BKS.BHZ
19980812141000.CMB.BHE 19980812141000.CMB.BHN 19980812141000.CMB.BHZ
19980812141000.KCC.BHE 19980812141000.KCC.BHN 19980812141000.KCC.BHZ
19980812141000.PKD.BHE 19980812141000.PKD.BHN 19980812141000.PKD.BHZ
```

Use SAC to verify that the files are intact.

Step 1: Instrument correct and filter waveform data and write data to ascii files

The first step is to produce the ascii, three-component data files used by tdmt_invc. This step involves acquiring the pole-zero instrument response, using SAC to demean, deconvolve instrument response, integrate to displacement (cm), rotate to transverse and radial components, bandpass filter, resample to 1 sps, and finally write the ascii data files. All of this is done using a single script, tdmt_redi_prepdata, which is located in the MTPACKAGE/MTCODE/BIN directory. A detailed explanation of the processing script is not given, but users are encouraged to examine it to understand the SAC processing steps. Note that the instrument response database file, instr.resp, is read by the get_resp program which then writes the SAC instrument pole-zero files that are used to deconvolve the instrument response. The path to instr.resp is set using the REDI_MT_RESP environment variable in either the tdmt.config.linux or tdmt.config.sun configuration files. The program assumes that the pole-zero responses contained in instr.resp convert from digital units to velocity in m/s.

The usage of tdmt_redi_prepdata is shown below for BKS (the REDI_MT_BINDIR variable provides the path to the executable directory and is set by sourcing the configuration files). The execution will have to be repeated for each of the stations.

```
$REDI_MT_BINDIR/tdmt_redi_prepdata 19980812141000 BKS 1998 224 36.755 -121.464 0.02 0.05
```

The first command line argument is the filename prefix, which consists of the year (1998), month (08), day (12), hour (12), minute (14) and seconds (00) of the start of the record. This is followed by the station name (BKS), the year, the day of the year (224), the latitude (36.775) and longitude (-121.464) of the event, and the highpass (0.02) and lowpass (0.05) filter parameters. In this case we are using 0.02 to 0.05 Hz waves, which have been found to be effective in the regional distance range (e.g. Pasyanos et al., 1996; Fukuyama and Dreger, 2000). You can use any frequency passband you wish provided that both the data and Green's functions are processed using the same filter, and the velocity structure that is used to generate the Green's functions is a good representation of structure at the wave lengths that are being modeled. In practice we use a magnitude dependent frequency passband, where for M<4.0 the passband is 0.02 to 0.1 Hz, for 4.0 <= M < 5.0 the passband is 0.02 to 0.05 Hz, for M>= 5.0 the passband is 0.01 to 0.05 Hz. For very large events (M>7.5) a passband of 0.005 to 0.02 Hz is desirable (Fukuyama and Dreger, 2000).

After running tdmt_redi_prepdata for each station make a note of the azimuth and distance that is output to the screen. The values obtained for BKS are:

```
  az = 3.314721e+02  <degrees from north>
dist = 1.420443e+02  <distance in km>
```

After running the script for the three stations you will have created the files, BKS_f0.05.data, CMB_f0.05.data, KCC_f0.05.data, PKD_f0.05.data.

Step 2: Generate Green's functions for a suite of distances and source depths
The next step is to generate a suite of Green's functions over a range of source depth for each source-station distance. The file MODEL.gil7 has been setup to generate Green's functions for 16 distances. Typically the distances are rounded to the nearest interval of 5 km. You can generate the gil7 synthetics using the following sequence of commands;

```
run_parallel 1 MODEL.gil7
run_fkrsort gil7_ 125 8 16
run_filter gil7_*d*.disp gil7_*d* 0.02 0.05
```

Note that you will need to explicitly enter the distance and depth in the spaces denoted by asterisks when filtering the Green's functions, and that nested foreach loops will be useful. This method for computing Green's functions may be used to compute a catalog of prefiltered Green's functions that may be used when automating these procedures as described in the next section.

Using the calculated Green's functions, the processed waveform data and the procedure for using tdmt_invc outlined in Example 1 determine the best fitting moment tensor solution for this event. Figure 3 shows the automatic cross-correlation results using three stations (BKS, CMB and PKD), and Figure 4 shows the best result obtained when KCC is added and inversions were performed with Green's functions for a range of source depth.

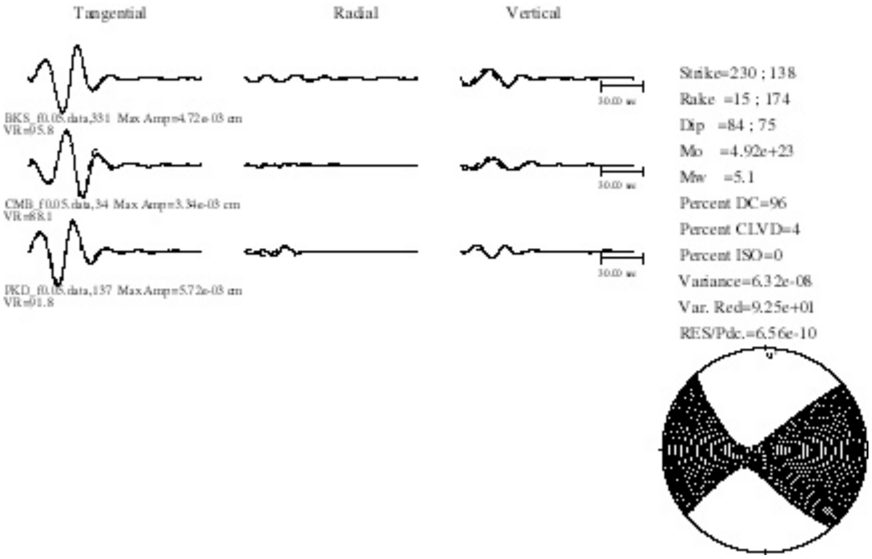


Figure 3 shows the result that should be obtained using Green's functions for a source depth of 8 km and the automatic cross-correlation feature of the code for the three stations BKS, CMB and PKD.

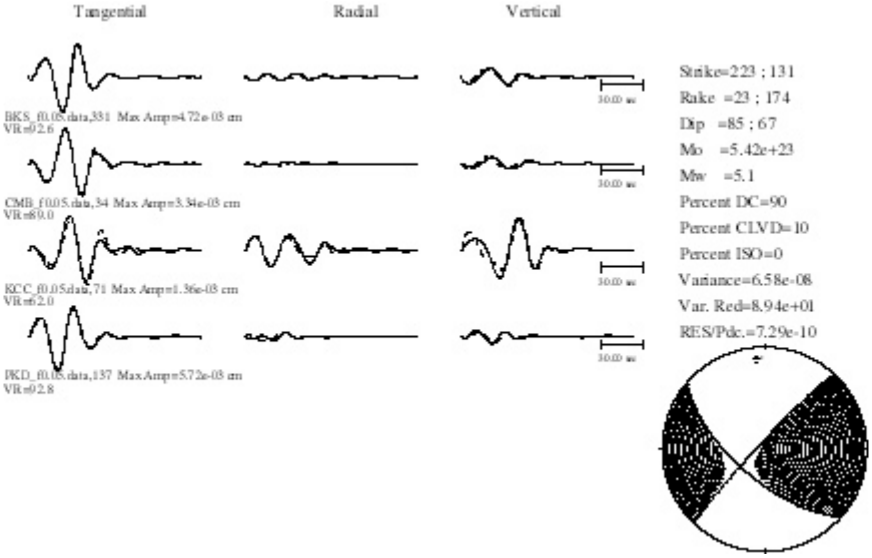


Figure 4 this is the best solution obtained testing source depths of 4.5, 8, 11, 14 km for stations BKS, CMB, PKD and KCC. The depth obtained is 11 km (though it is not well resolved in this particular case).

Realtime Processing

As discussed previously the runtime properties of the software are controlled by the "tdmt.config.*" configuration files, located in each of the example subdirectories. All of the environment variables are described below,

setenv REDI_MT_BINDIR	sets the executable directory
setenv REDI_MT_PROG_1	defines the launching program
setenv REDI_MT_DATDIR	sets the directory where individual event working directories are created
setenv REDI_MT_LOGDIR	sets the directory where results are logged.
setenv REDI_MT_SYNTHDIR	sets the directory where Green’s functions are located
setenv REDI_MT_EXTRACT_OPTIONS	defines runtime flags for qdata data extraction program
setenv REDI_MT_DEBUG_OPTION	turns on debugging output
setenv REDI_MT_STATLIST	defines the station coordinate file
setenv REDI_MT_RESP	defines the instrument response (pole-zero file)
setenv REDI_MT_DATASTREAM	defines data stream to be used (BH and LH streams are supported in this release)
setenv REDI_MT_PLOT	flag for plot output (1=yes; 0=no)
# Program specific variables	
setenv REDI_MT_PROG1_PAGE	flag for paging results (1=yes; 0=no)
setenv REDI_MT_PROG1_GFLOC	sets the directory where Green’s functions are located (same as REDI_MT_SYNTHDIR)
setenv REDI_MT_PROG1_STATMAX	number of REDI approved stations
`	
setenv REDI_MT_PROG1_GETLIST	list of REDI approved stations (stations not on this list will not be processed, may include stations that are not moment tensor processing approved)
setenv REDI_MT_PROG1_NSTAT	number of moment tensor approved stations
setenv REDI_MT_PROG1_STATLIST	list of moment tensor approved stations

Note that the REDI_MT_STATLIST and REDI_MT_PROG1_STATLIST environment variables are extremely important in defining the subset of available stations to use in the automatic inversion. The first environment variable defines the path to a file that contains a list of operational stations and their coordinates. By removing a problematic station from the station list the station is effectively removed from all seismic moment tensor processing. Although a station is operational, and included on the REDI_MT_STATLIST, it may not be suitable for moment tensor analysis. For example, stations that are located inside of buildings, have substantial site response, or whose source-station paths have complex local and regional structure may not perform well using 1D velocity model Green’s functions. Therefore a second environment variable, REDI_MT_PROG1_STATLIST, is used to define those stations that perform well under the assumptions of the methodology. In sum, REDI_MT_STATLIST is used for the extraction of data and preprocessing in the SAC data format, and can be used to communicate to the program which stations are available as network conditions fluctuate. Only stations on the REDI_MT_PROG1_STATLIST qualify for subsequent processing and are used by the inversion routine.

Automated Program Flow

There are two primary parts in the automation of these routines as illustrated in Figure 5. The first is the triggering based on some external event hypocenter information obtained via email or other method, the creation of the working directory, and creation of the input file for subsequent processing. The second part is executing a wrapper program that retrieves and processes data, copies the correct Green’s functions into the working directory, and executes the inversion routine keeping track of the best solution.

Step 1: Triggering

One way to trigger processing is to scan for incoming email messages. For example, the UNIX cron function below may be used.

```
0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48,51,54,57 * * * *
/data/10/dreger/AUTOMT/tdmt_trigger 1>/data/10/dreger/AUTOMT/NEWEVENT/trig.msgs
2>&1
```

This cron function will execute the tdmt_trigger script every 3 minutes and write the output to a ‘trig.msgs’ messages file, which can be analyzed for relevant event information. The tdmt_trigger script is in MTPACKAGE/MTCODE/BIN. This script initiates the UNIX mail reader program (mailx), scans for suitable messages (those coming from redi in this example), and saves the messages in a file called redi_trigger. The program, tdmt_msgs_scan, then opens the redi_trigger file, and scans for event email messages, retrieves the event information and creates an event file in a staging subdirectory, GUIDE. The path to the GUIDE subdirectory is defined by the REDI_MT_DATDIR environment variable in tdmt.config.*. If files are present in GUIDE a new event directory is created and a wrapper program input file (tdmt_redi_sched.in) is copied to the new event directory, and the wrapper program, tdmt_redi_sched, is executed.

This is just one example of how to automated this code, and it will be necessary to modify the above procedures for your specific operating environment.

Step 2: Scheduling (The Wrapper Program)

tdmt_redi_sched is a wrapper program that reads the tdmt_redi_sched.in file, and executes a series of shell scripts and codes (tdmt_redi_getdata, tdmt_redi_prepdata, tdmt_redi_prepsyn, tdmt_invc) (i.e. Figure 5). tdmt_redi_sched.c will need to be modified for your specific monitoring situation, to account for different crustal velocity models, different frequency passbands, changes to the Green’s function distance and depth ranges, and the specifics on how you want to report the results.

The input file for tdmt_redi_sched, namely tdmt_redi_sched.in, has the following format.

```
tdmt.config.linux 1998 224 14 10 0.0 36.755 -121.464 5.0 4.8 9999
```

The first field is the name of the run-time configuration file, the year (1998), julian day (224), hour (14), minute (10), and seconds (00) of the event, the latitude (36.755), longitude (-121.464), initial source depth (5.0), magnitude (4.8), and an event id number (9999). The origin time only needs to be approximate as the data are optimally aligned with the Green’s functions using a cross-correlation function. In practice the origin time is truncated to the preceding minute (seconds field is always zero). The hypocenter information in tdmt_redi_sched.in is externally derived and delivered to the working directory via email or other method.

The first script that is executed is tdmt_redi_getdata, which extracts waveform data from the archive and translates it to SAC format for subsequent processing. This script will need to be modified for the specific data retrieval method used on your system. In the BSL application we retrieve data in miniseed format and use the ms2sac program to translate miniseed to SAC. The BSL specific code has been commented out of the script and code that extracts the SAC files from MTPACKAGE/Example_2 has been added to facilitate Example 3 illustrating the usage of tdmt_redi_sched.

Next, based on the data availability (REDI_MT_STATLIST environment variable) and the list of stations that may be used to compute moment tensors (REDI_MT_PROG1_STATLIST) the data for the three closest stations in 50-400 km distance range is processed.

The tdmt_redi_prepdata script used in Examples 1 & 2 is called to create the ascii files containing the processed (instrument deconvolved, integrated to ground displacement (cm), bandpass filtered, and resampled to 1 sps) waveform data. All data processing is accomplished using SAC. This script requires an instrument response database whose location is set using the REDI_MT_RESP environment variable in the tdmt.config.linux file.

The tdmt_redi_prepsyn script uncompresses prefiltered Green's functions files for the relevant source-station distances, and all source depths into the event subdirectory. The code is currently designed to use the sc and gil7_ velocity models that are expected to be stored in compressed format in the LP10SYN (0.02 to 0.10 Hz), LP20SYN (0.02 to 0.05 Hz) and LP5020SYN (0.02 to 0.05 Hz) subdirectories. The addition of new velocity models will require modification of the tdmt_redi_sched program, and the computation of new Green's functions.

Finally, with the processed data files and Green’s function files in the working directory tdmt_redi_sched loops over source depth. For each source depth iteration a tdmt_invc input file, mt_inv.in is created, tdmt_invc is then executed, and the results are stored in a log file, mt_inv_redi.out. After all of the inversions have been performed this file is read to find the source depth that yielded the greatest variance reduction, and the corresponding source information is written to the pager.file, which is then emailed or paged to interested parties. Of course, this latter step of disseminating the results will need to be modified for your application. One final note, the REDI_MT_PAGE environment variable controls whether or not to perform this final dissemination stage.

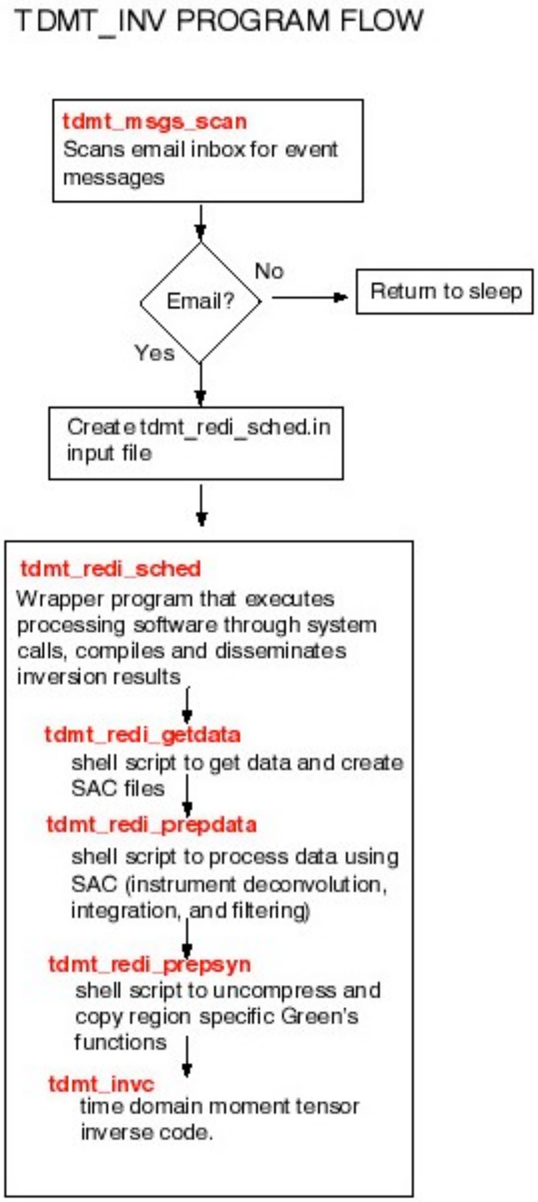

TDMT_INV PROGRAM FLOW
tdmt_msgs_scan
Scans email inbox for event messages
Email?
No
Return to sleep
Yes
Create tdmt_redi_sched.in input file
tdmt_redi_sched
Wrapper program that executes processing software through system calls, compiles and disseminates inversion results
tdmt_redi_getdata
shell script to get data and create SAC files
tdmt_redi_prepdata
shell script to process data using SAC (instrument deconvolution, integration, and filtering)
tdmt_redi_prepsyn
shell script to uncompress and copy region specific Green's functions
tdmt_invc
time domain moment tensor inverse code.

Figure 5. Automated moment tensor program flow. All of the processing scripts are in the $MTPACKAGE/MTCODE/BIN directory.

Example 3

This example will illustrate the usage of tdmt_redi_sched. Change directory to MTPACKAGE/Example_3. There should be four files, tdmt.config.linux, tdmt.config.sun, example3.stat and tdmt_redi_sched.in. tdmt_redi_sched.in is the input file used by tdmt_redi_sched, and example3.stat is a list of stations that tdmt_redi_sched will process (defined by the REDI_MT_STATLIST environment variable).

Prefiltered Green’s function files needed for this example have been placed in the LP10SYN, LP20SYN, and LP5020SYN subdirectories in MTPACKAGE for this exercise. This is an incomplete set and you will need to construct a complete set of Green’s functions for your study area before applying this routine to other earthquakes.

Before doing anything else edit either the tdmt.config.linux or tdmt.config.sun files to set the environment variable MTPACKAGE to the full directory path of the software package, and source the configuration file that is appropriate for your system.

The Only Step

In the working directory execute the tdmt_redi_sched program. I.e.,

```
$REDI_MT_BINDIR/tdmt_redi_sched
```

Code progress is written to the screen.

If everything runs you will find the following information in the pager.file that is created.

```
REDIMT:9999 5.1 19980812141000 36.76 -121.46 5.0 5.87E+23 49,-10,82  141,-172,80 14 VR94.1
```

This is the format of an email message that may be sent out if the REDI_MT_PAGE parameter is set to 1 in the tdmt.config.linux configuration file. It shows a REDIMT identifier, the event id number, the Mw, the year, month, day, hour, minute, seconds as a string, the latitude, the longitude, the initial depth, the scalar moment (dyne cm), strike,rake,dip for both planes, the depth determined from the moment tensor inversion, and the composite variance reduction.

Time-Domain Moment Tensor with ISOtropic INVerse Code (TDMTISO_INVC)

Introduction

This seismic moment tensor inverse software package is a newly developed methodology from Minson & Dreger (2007) after Herrman & Hutchensen (1993). This distribution includes a set of programs for calculating a library of Green's functions, inverting broadband data for the seismic moment tensor, and various data processing utilities and shell scripts. Complete source code is provided. The frequency-wavenumber integration program (FKRPROG) written by Chandan Saikia of URS is included with his permission. This software may be distributed only by means of its gzipped tar file. There is no warranty either expressed or implied, and the author is not responsible for damages due to misinterpretation of results and misuse of the software.

Finally, this software is freely distributed for non-commercial use. The reporting of seismic moment tensor solutions obtained using this software on the WWW and in the technical literature should include the following acknowledgement statement. "Moment tensors were computed using the MTpackageV2.1 package developed by Douglas Dreger of the Berkeley Seismological Laboratory, and Green's functions were computed using the FKRPROG software developed by Chandan Saikia of URS."

In the following the users manual documents how to build the software package and demonstrates usage through an example.

Building the Program

You must already have built the deviatoric code by running the makefile as above in the directory MTCODE. To compile the isotropic code copy the file makefile_full to makefile and use the make command. The distribution compiles using the C (cc) and fortran (f77) compilers on SUN workstations running Solaris. After the build, all of the executables are located in the MTPACKAGE/MTCODE/BIN subdirectory.

Basic Methodology and Program Assumptions

As opposed to the deviatoric solution computed above, the full solution requires an additional explosive fundamental fault component. Percent isotropic (PISO) is now computed and variance reduction is computed as above.

Finally, it is assumed that the crustal model is sufficiently well known to explain low frequency wave propagation. This software package will not work in a region if calibrated velocity models are not available. Calibrating velocity models to obtain a robust catalog of Green's functions is singly the most important step in successful seismic moment tensor applications.

In the following the usage of the software is demonstrated through an example that is similar to Example 1. The main difference between the deviatoric processing steps above and the ones listed here are that now the explosive fundamental fault componenet is extracted from the FKRPROG binary output.

Example 4

The objective of Example 4 is to become familiar with computing Green's function files, performing post-processing filter operations, and using the seismic moment tensor inversion code by inverting synthetic waveform data. The files necessary for example 4 are found in MTPACKAGE/EXAMPLE_4. You should use this directory as your working directory. In this directory you will find the following files; tdmt.config.sun, MODEL.song, run_parallel, run_isosort, run_filtiso, b2s.par, mt_inv.in, and testdata[1-8].

Before doing anything else set the system specific environment parameters. You will need to edit the tdmt.config.sun files to set the environment variable MTPACKAGE to the full directory path of the software package. After making the changes use the source program to invoke the environment variables (i.e. source tdmt.config.sun). This command sets the path to executables.

Step 1: Compute Green's functions

The file MODEL.song contains a sample 1D velocity structure appropriate for the Western Basin & Range region that is based on the model from Song et al. (1996). The format is similar to the file MODEL.socal above.

The first task of this example is to run the frequency-integration program (FKRPROG) through the run_parallel script to compute the Green's functions used to invert the synthetic waveform data. **Very important note**; FKRPROG requires formatted input. Be careful not to change the column location of each field. Integer input is right justified, and floating point input should retain the position of the decimal point.

To generate the Green's functions first run the run_parallel script. This script performs all of the steps required to generate the GREEN.* output files. The usage is:

```
run_parallel number_of_cpus name_of_inputfile
```

Step 2: Post-processing to create Green's functions in usable ascii format

Once the FKRPROG process(es) have finished the next step is to create ascii format, time domain Green's function files that will be used by the inversion code. Another script, run_isosort, is used for this purpose. This script first calls the wvint9 program, which reads the GREEN.* file(s) and performs an inverse FFT and writes all of the displacement (cm) Green's functions into a headerless binary file, vec. Note that options for wvint9 are hard-coded in the script. It is possible to output velocity (cm/s) Green's functions instead of displacement (cm) by changing the 'd' to 'v' in the standard input statement for wvint9. Subsequent to the wvint9 post-processing the specific time series are then extracted from the vec file and a ten component ascii file is constructed. The ordering of the components are as follows, and the same as in Jost and Herrmann (1989); transverse component vertical strike-slip (tss) and vertical dip-slip (tds) faults, radial component vertical strike-slip (xss), vertical dip-slip (xds) and 45-degree dip-slip (xdd) faults, and vertical component vertical strike-slip (zss), vertical dip-slip (zds), 45-degree (zdd) faults, and radial (rex) and vertical (zex) explosions.

The following is the usage for run_isosort.

```
run_isosort filename_prefix distance depth number_of_distances
```

In this example the MODEL.song file is built to produce 8 files from 100 - 800 km distance. So one would use ‘run_isosort song 100 1 8’

This script will output the files with the naming convention, {filename_prefix}{distance}d{depth}.disp. Since the number_of_distances is not equal to 1 then the script assumes that the distance given is the closest distance, and the following distances are for intervals of 100 km, so you will have files song100d1.disp, song200d1.disp, etc.

Step 3: Filter Green’s functions

Once the *.disp files have been created it is necessary to perform bandpass filtering using the run_filtiso script. The usage of run_filtiso is:

```
run_filtiso  infile_name outfile_name  high_pass low_pass
```

The infile_name is the output filename from the run_isosort script. The outfile_name can be anything, but typically the infile_name with the ‘.disp’ truncated is used. In this example we have included the file SCRIPT which will process the files with run_filtiso. The high_pass and low_pass filter corners are given in hertz. An acausal (two pass), 4th order butterworth filter is applied using SAC. The SAC commands are imbedded in the run_filtiso script.

Step 4: Run Moment Tensor Inversion

Once the filtered Green’s function files have been computed you can use them to invert the test data for Example 4. You will find three data files, testdata[1-8], in the working directory. These data files are listed in the provided ***tdmtiso_invc*** input file called mt_inv.in.

mt_inv.in has the following format

8 1 1 6 1 <number of 3-component stations, source depth, distance weighting flag, free parameters (5=deviatoric solution; 6=full solution), plotting flag>
testdata1 100.00 0.00 0 120 <data_filename, distance (km), azimuth (deg from north), sample-offset (Zcor), number_of_samples>
etc.
song100d1 0 120 <filtered GF_filename, zero-offset (always zero), number_of_samples (same as corresponding data>
etc.

To run the moment tensor code execute the command “$REDI_MT_BINDIR/tdmtiso_invc > plot”, where plot is a plot file, that may be converted to postscript using “psigl < plot > plot.ps” When tdmt_invc is run the following information is output to the screen, and to a log file (mt_inv_redi.out).

The output information is the same except for how the scalar moment is calculated. In the deviatoric code

$$M_0 = \frac{|m_1| + |m_3|}{2}$$

where m_1 and m_3 are the largest and smallest deviatoric eigenvalues, respectively. This equation is appropriate for a purely double-couple source. The full code calculates the scalar moment with

$$M_0 = \frac{\text{trace}(\mathbf{M})}{3} + m_1$$

where **M** is the moment tensor (Bowers & Hudson, 1989).

Figure 5 shows the best result that is obtainable.

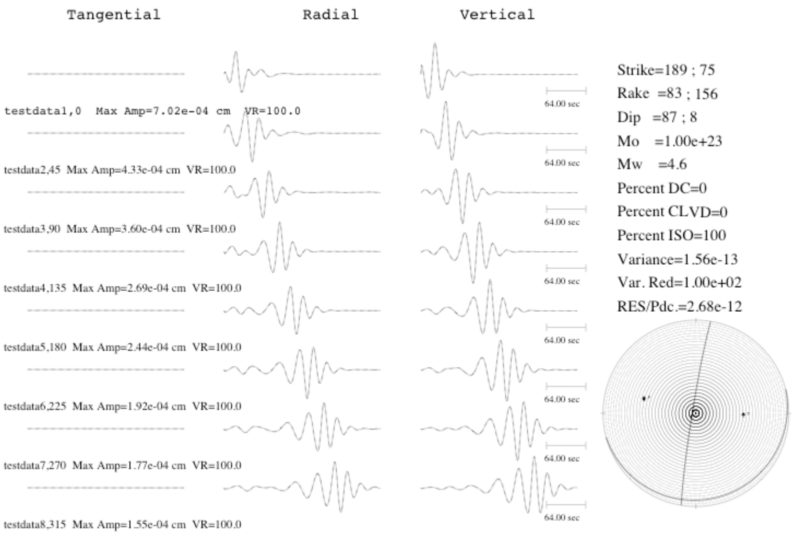


Figure 5. Example of ***tdmtiso_invc*** graphical output using Green’s functions for a source depth of 1 km. See Figure 1 for explanation. To obtain this result it was necessary to manually shift the Green’s functions by one sample.

Concluding Remarks

I hope that you find this seismic moment tensor package helpful for your seismic monitoring and scientific needs.

Send comments, bug reports and other inquiries to dreger@seismo.berkeley.edu.

Updates to this package will be posted to www.seismo.berkeley.edu/~dreger.

Please add the following acknowledgement to any WWW or technical journal publications that use moment tensor results obtained using this software. "Moment tensors were computed using the tdmt-invc package developed by Douglas Dreger of the Berkeley Seismological Laboratory, and Green's functions were computed using the FKRPROG software developed by Chandan Saikia with URS."

References and Supplemental Reading

Papers illustrating velocity model calibration:

Dreger, D. S., and D. V. Helmberger (1990). Broadband Modeling of Local Earthquakes, *Bull. Seism. Soc. Am.*, 80 1162-1179.

Dreger, D. S., and D. V. Helmberger (1993), Determination of Source Parameters at Regional Distances with Single Station or Sparse Network Data, *Journ. Geophys. Res.*, 98, 8107-8125.

Dreger, D. and B. Romanowicz (1994). Source Characteristics of Events in the San Francisco Bay Region, *USGS Open-file report*, 94-176, 301-309.

Helmberger, D. V., and G. Engen (1980). Modeling the long-period body waves from shallow earthquakes at regional distances, *Bull. Seism. Soc. Am.*, 70, 1699-1714.

Rodgers, A., W. Walter, R. Mellors, A. Al-Amri, Y. Zhang (1999). Lithospheric structure of the Arabian Shield and Platform from complete regional waveform modelling and surface wave group Velocities, *Geophys. Journ. Int.*, 138, 871-878.

Saikia, C. K. (1994). Modified frequency-wavenumber algorithm for regional seismograms using Filon's quadrature; modeling of L_g waves in eastern North America, *Geophys. Journ. Int.*, 118, 142-158.

Song, X., L. Zhao and D. V. Helmberger (1996). Broad-band modelling of regional seismograms; the Basin and Range crustal structure, *Geophys. Journ. Int.*, 125, 15-29.

Zhao, L., and D. V. Helmberger (1991). Broadband modelling along a regional shield path, *Geophys. Journ. Int.*, 105, 301-312.

Papers describing various moment tensor applications:

Dreger, D., and A. Kaverina (2000). Seismic remote sensing for the earthquake source process and near-source strong shaking: A case study of the October 16, 1999 Hector Mine earthquake, Geophys. Res. Lett., 27, 1941-1944.

Fukuyama, E., and D. Dreger (2000). Performance test of an automated moment tensor determination system for the future "Tokai" earthquake, Earth Planets Space, 52, 383-392.

Gee, L. S., D. S. Neuhauser, D. S. Dreger, M. Pasyanos, R. A. Uhrhammer, and B. Romanowicz (1996), Real-Time Seismology at UC Berkeley: The Rapid Earthquake Data Integration Project, *Bull. Seism. Soc. Am.*, 86, 936-945.

Jost, M. L., R. Herrmann (1989). A student's guide to and review of moment tensors, *Seism. Res. Lett.*, 60, 37-57.

Pasyanos, M. E., D. S. Dreger, and B. Romanowicz (1996), Towards Real-Time Determination of Regional Moment Tensors, *Bull. Seism. Soc. Am.*, 86, 1255-1269.

Romanowicz, B. D. Dreger, M . Pasyanos, and R. Urhammer (1993). Monitoring of Strain Release in Central and Northern California Using Broadband Data, *Geophys. Res. Let.*, 20, 1643-1646.