Documentation of Programs for Kinematic Rupture Modeling

prepared by Jan Schmedes and Jorge Crempien

Table of Contents

Broadband Simulation of Ground Motion

Introduction

The method to compute broadband seismograms for kinematic source models is described. Many of the details can be found in Liu et al. (2006). Only 1D velocity models are considered. A 3D finite difference code exists, but was not used and is thus not described further. This description covers all necessary steps:

- Computation of 1D Greens functions
- Computation of kinematic source models
- Computation of broadband seismograms
- Application of nonlinear site response if necessary
- Seismogram post processing

The post processing is done using various shell scripts and Mathematica to produce various plots. After the description of the single components an example will be discussed step by step.

Necessary information

The following information is necessary in order to proceed

- The 1D velocity structure
- Station locations
- Hypocenter location
- Geometry of fault and location of hypocenter on fault
- Seismic moment and corner frequency
- If site response should be computed Vs30 for each site

The steps necessary to compile the code as well as the packages for Mathematica is explained in Appendix.

Computation of Green's functions

To compute the Green's function **green_bankFar** using the 1D velocity model is necessary as input. Furthermore, one needs to know the depth range for which Green's functions need to be computed as well as the range of epicentral distances. Furthermore, based on the range of epicentral distances one needs to decide on the total duration of the Green's functions. They need to be long enough to not have artifacts due to wrap- around effects.

It is possible to define two ranges with different sampling distance in space for epicentral and depth sampling. For example, close to the fault one needs the Greens function sampled finer than for distances further away. The same is true for the depth sampling, which should be finer in the top few km. If the situation arises that one has a shallow source (say 100m below free surface) and needs to compute out to larger distances there can be problems with the FK method. To avoid artifacts related to this, one can also chose a distance after which the Greens functions for

shallow sources are replaced with Green's functions of a greater depth (say 300 m). One parameter that should never (!!!) be set to zero is the time to save before the first arrival. If this parameter is set to zero artifacts will occur! Furthermore, the minimal depth and the minimal epicentral distances can not be set to exactly zero, which likely yields Nan errors.

To test if the computed Greens functions have artifacts, for each depth the farthest Green's function is written to a file called *test"i"* where *i* is the depth index, i.e., for the most shallow source the file is called *test1*. The Mathematica notebook *checkGreen.nb* provided with this documentation can be used to plot the Greens function and see if the duration was chosen long enough.

The input file *GreeFar.in* contains all this information and is described in the Appendix. Once the Green's functions are computed the file *Green_Bank.inf* and a large binary file is created. Those files will be needed in the next steps. One thing that is important is that the file *Green_Bank.inf* contains the correct path to the large binary file containing the actual Greens functions. Because the binary file can be very large, only the file *Green_Bank.inf* will be copied to the directories in which the computations of the synthetics are performed. The file might have to be edited by hand.

Computation of Source, Liu et al. (2006) (LAH)

To compute the source using **output_LAH** the following information is necessary:

- 1D velocity model
- hypocenter depth, and position of hypocenter on fault (along strike and along dip)
- coordinates of epicenter in local Cartesian coordinate system
- dimension of fault plane (length and width)
- seismic moment and corner frequency
- strike, dip, and rake
- time stepping (dt) of Green's functions. They have to be the same

All this information is specified in the input file *output_LAH.inp* discussed in the Appendix.

Two source files are computed. One is just an ascii file with 10 columns containing x, y, z, moment, rupture time, rise time, peak slip rate, strike, dip, and rake of each subfault. This file is discussed in the Appendix and referred to as *LAH format* (for Liu, Archuleta, Hartzell). This file was used in the original ground motion simulation code of Liu et al. (2006), which should not be used because of some problems in the interpolation of the Green's functions. Note, that the 7th column containing the peak slip rate is not used in the computations. This file is still created because it is very useful for plotting the source model. The notebook *plotSource.nb* provides code and examples on how to do this. Furthermore, the notebook *plotGeometry.nb* provides code and examples on how to plot the stations and the fault plane in 2D and 3D.

The other output format is the so called 'standard-rupture-format', short *SRF format*, used in the SCEC broadband platform. It is explained in the Appendix. For each point the full slip rate function is specified. This makes the file larger and more tedious to plot than the *LAH format*. However, the notebook *plotSource.nb* shows examples on how to plot the source and also slip rate functions. The nice thing about this format is that it contains all information and can be interchanged between different researchers, as done in the broadband simulation platform.

Another file necessary for the computation of the seismograms in *source_SCEC.list,* which simply contains the names of the created source models.

Computation of Source, Schmedes (2009) (SAL)

This rupture model generator **KinModel** is based on the results obtained by analyzing a large number of dynamic models (Schmedes *et al.*, 2009). Its implementation and additional information can be found in Schmedes (2009). It is strongly recommended to read this chapter about the implementation.

SAL requires an input file *KinModel.inp* described n the Appendix and the file containing the velocity model. The required information is the same as for *LAH* except the corner frequency, which is chosen automatically.

It also outputs the *SRF format,* the files are called *modSRF_'i'*, where *i* is the index of the model. Then the file *source_SCEC.list* required to compute the seismograms. Furthermore, a format that is almost the same as the *LAH format* except the 7th column in which now instead of the peak slip rate the peak time is written. They are called *model_'i'.* Again, these files are good for plotting. In addition, they can be used to create an *SRF* formatted file using the slip rate function of Liu et al. (2006) instead of the one in Schmedes et al. (2009). This also means, that the peak time is not used (this parameter does not exists in *LAH*). The reason to do this is for a comparison between the slip rate functions and the influence of the additional free parameter in *SAL*, the peak time.

In addition, there is a file called l*og_'i',* which contains the four kinematic parameters that are the basis of the *SAL* rupture model generator: slip (not moment), rise time, peak time, local rupture velocity. It is described in the Appendix. The notebook *plotSource.nb* illustrates how to plot these.

Other files that are created are *mrspec_'i',* which contains the spectrum of the moment rate and a Brune spectrum. The three columns are frequency, Brune spectrum, moment rate spectrum. The notebook *plotSource.nb* illustrates how to plot them. And finally, a file *corr_'i'* is created. It contains the six off-diagonal components of the correlation matrix for the model. Because there is a fixed number of iterations, these correlations will always be not exactly the target correlation.

It is possible to influence the variation among models by changing the number of iterations. This can be done in one of the header files of the code, which also requires to re-compile the code. There is a special Appendix dedicated to the description of the code and various parameters than can be changed.

Computation of Linear Ground Motion

To compute seismograms not accounting for nonlinear response of the top few hundred meters the code **syn1D_LAH** is used. The following files are necessary:

- The file *Green_Bank.inf* pointing to the Green's function database
- The file containing the station coordinates in Cartesian coordinates. This file is described in the Appendix
- The file *source_SCEC.list*
- The *SRF format* files listed in *source_SCEC.list*

Then there is the input file *syn1D_LAH.inp described in the Appendix*. Besides listing the name of the station file and *source_SCEC.list*, it also contains additional information. First, the number of point sources for each subfault. There is an interpolation performed to ensure that there are no artifacts in the high frequencies, see Liu et al. (2006). Then there is a perturbation of azimuth, rake and dip. This is similar to Liu et al. (2006), but the perturbation of strike is exchanged with the perturbation of azimuth. This is done to mimic scattering which leads to a more isotropic radiation patter. Then two frequencies, between which there is a perturbation from deterministic to stochastic (in terms of radiation pattern). Again, for more info see Liu et al. (2006). Finally, which component to output and which format (binary format does not work). If the programs and scripts provided with this tutorial should be used these should be set to output velocity in txt format (both switches set to 2).

For each station three files are created, two for the horizontal orientation defined in the station file, and the vertical motion. External programs to compute displacement and acceleration, to filter the time series, and to compute response spectra and Fourier spectra exist and will be described in the Appendix.

Computation of Nonlinear Soil Response

I only have limited knowledge of the code NOAHW by Bonilla et al. (2005), I used the input files as they were provided by Pengcheng Liu. Thus, I will not decribe any input files for this code. A better understanding of this part of the simulation chain is up to the user.

Essentially, Pengcheng Liu provided for each site Class A, B, BC, D, DE, and E the input velocity structure for the nonlinear code. First, one needs to deconvolve the seismograms to the depth of the seismic basement. In the example provided later I use 300m. In the next step the stations get assigned a site class according to their Vs30 (Wills *et al.*, 2000). The following table shows the site classes and Vs30 ranges.

Site Class	B	BC	C	CD	D	E
Vs30 range (m/s)	762-1524	762-555	555-366	366-270	270-183	<183

Nothing is done for site class A, which is rock. For the other spoil classes the waves are propagated to the free surface using the nonlinear code NOAHW. Because it does not account for horizontal motion, like surface waves, I then stitch the linear and nonlinear seimograms together using w matching filter in the wavelet domain

provided by Pengcheng Liu . I chose a transition frequency of 1 Hz. This is similar to Liu et al. (2006), but they stitch the 3D and 1D seismograms using a frequency domain matching filter. I provide a script in the example section, which performs all these steps. The programs used in the various steps are shortly described in the Appendix.

Post Processing

Various codes described in the Appendix are available. The important ones are:

- Filter routines
- Computation of Response spectrum
- Computation of Fourier spectrum
- Convert txt to SAC format
- Compute response spectral and Fourier spectral bias of horizontal components (requires of course data against which the comparison is performed)

Furthermore, the notebook plotGroundMotion.nb explains how to prepare simple plots of seismograms and response spectra. The notebook GMstatistic.nb provides some tools for statistic of ground motion. Finally, there is a Mathematica package GroundMotion.m available, which is explained in docuGroundMotion.nb.

Some tools for lon lat conversion

Often one encounters the situation that one has stations and some fault coordinates in lon lat and needs a Cartesian coordinate system. There are some tools available, but they are developed for the SCEC Broadband platform and require some different input files. The file *faultGlobal.in* contains similar information as *output_LAH in* and *KinModel.in*. One important difference is, that it requires the top center of the fault in lon lat coordiantes. This will be taken as the coordinate origin. The tool to convert a lon lat station list into xy coordinates uses whatever is given there as origin. But in the created xy station file, the first line, which contains the epicenter in xy, will assume the top center was used. In case one wants to use the epicenter lon lat coordinates, one needs to edit the station file by hand to use 0 0 as origin. There are also tools to convert SRF files into lon lat and SRF lon lat into XY. In SCEC the srf files are defined to have lon lat and not XY. Here the top center Is again expected in the file *faulGlobal.in*.

Bonilla, L.F., Archuleta, R.J. & Lavallee, D., 2005. Hysteretic and Dilatant Behavior of Cohesionless Soils and Their Effects on Nonlinear Site Response: Field Data Observations and Modeling, *Bull. Seism. Soc. Am.,* 95**,** 2373-2395.

Liu, P., Archuleta, R.J. & Hartzell, S.H., 2006. Prediction of Broadband Ground-Motion Time Histories: Hybrid Low/High- Frequency Method with Correlated Random Source Parameters, *Bull. Seism. Soc. Am.,* 96**,** 2118-2130.

Schmedes, J., Archuleta, R.J. & Lavallee, D., 2009. Correlation of Earthquake Source Parameters Inferred from Dynamic Rupture Simulations, *J. Geophys. Res.,* submitted.

Wills, C.J., Petersen, M., Bryant, W.A., Reichle, M., Saucedo, G.J., Tan, S., Taylor, G. & Treiman, J., 2000. A Site-Conditions Map for California Based on Geology and Shear-Wave Velocity, *Bull. Seism. Soc. Am.,* 90**,** S187-208.

Appendix A

File formats

List below are the different file formats

Velocity model

1. line: numberLayers, placeholder [the number of layers (including halfspace) in the 1D model, an input that is of no importance for the 1D broadband modeling]
2. line … numberLayers-1 line: Vp, Vs, density, thickness, Qp, Qs [P-wave velocity, S-wave velocity, density, thickness of layer, quality factor for P-wave, quality factor for S-wave]
3. numberLayers line: : Vp, Vs, density, 0.0, Qp, Qs [P-wave velocity of halfspace, S-wave velocity of halfspace, density of halfspeace, for halfspace use thickness 0.0, quality factor for P-wave in halfspace, quality factor for S-wave in halfspace]

Example input:

```
8  1.0
1.2 0.3 1.7   0.1    27.0    18.0
1.6 0.5 1.8   0.2    45.0    30.0
1.9 1.0 2.1   0.2    90.0    60.0
4.0 2.0 2.4   1.0   420.0   280.0
4.7 2.7 2.6   2.5   567.0   378.0
6.3 3.6 2.8  23.0   864.0   576.0
6.8 3.9 2.9  13.0   936.0   624.0
7.8 4.5 3.3   0.0  1080.0   720.0
```

Station file

1. line: NStat, epiX, epiY, placeholder [number of stations, x-coordinate of epicenter, y-coordinate of hypocenter, some placholder that should always be 0.0]
2. line: stationName [name of station. This is the name that will be used to name the files containing the time series]
3. line: statX, statY, statZ, comp1, comp2 [x-ccordinate of station, y-coordinate of station, depth of station (should be 0), component 1 of horizontal ground motion. For example if 0.0 then the component will be along X. component 2 of horizontal ground motion]

   …

   Example input:

   ```
   4 -15782.580698 -2786.980552 0.000000
   scse
   -3517.273682 3144.834717 0.000 0.000000 90.000000
   ```

jeng
-3562.219238 1793.048462 0.000 0.000000 90.000000
jens
-3562.219238 1793.048462 0.000 0.000000 90.000000
rrs
-7012.928223 3338.997314 0.000 0.000000 90.000000

GreenFar.in

1. line: nameVelmod [name of file containing velocity model]
2. line: minDepth, dz1, Nz1, dz2, Nz2 [minimal depth for Greens functions, depth sampling increment for first Nz1 sources, Number of sources with dz1 sampling, depth sampling increment for Nz1+1…Nz1+Nz2 sources, Nz2 number of sources with dz2]
3. line: minEpi, dx1, Nx1, dx2, Nx2 [minimal eicentral distance for Greens functions, epicentral distance sampling increment for first Nx1 sources, Number of sources with dx1 sampling, epicentral distance sampling increment for Nx1+1…Nx1+Nx2 sources, Nx2 number of sources with dx2]
4. line: Nt, dt, tBefore [number of time steps, time increment, seconds to be saved before first arrival. This should never be set to 0 (because of wrap-around artifacts!!!]
5. line: nameGreenDB [name of the file containing the Greens function database]
6. line: minDepthFar, NFar [for sources with epicentral distance index NFar…Nx1+Nx2 every source that is more shallow than minDepthFar, the Greens Function will be replaced with a source that is at the closest but larger depth than minDepthFar. This is done, because for larger distances there can be a problem with too shallow sources.]

Below is an example. Note, that the last line (number 6) has no effect because minDepth>minDepthFar

Example Input:

velocity.soil2
5.0 0.3 15 0.5 25
0.05 0.5 30 1. 100
4000 0.01 3.0
Green_1d.soil
0.4 35

output_LAH.inp

1. line: switchSRF [slip rate function to use (use 4 which is the one defined in Liu et al. 2006)]
2. line: rupL, ddW [rupture length, down-dip width, i.e., dimensions of fault plane in km]
3. line: hypoStrike, hypoDip, depthHypo [position of hypocenter on fault along dip, position of hypocenter on fault along dip, depth of hypocenter in km]
4. line: xEpi, yEpi [x-coordinate of epicenter, y-coordinate of epicenter in km]

5. line: M0, fc [seismic moment of event, corner frequency of event]
6. line: strike, dip, rake (strike, dip, rake of event)
7. line: NSubStrike, NSubDip [number of subfaults along strike, number of subfaults along dip. Both have to be a power of 2!!!]
8. line: seed1, seed2, seed3 [random seeds]
9. line: index1, index2 [lower index of models, upper index of models. For example if 1,1 then 1 source is created, if 1,10 10 sources are created]
10. line: nameVelMod [name of file containing velocity model]
11. line: 0.0
12. line: switchSlip [moment (1), slip-area (2), slip(3)]
13. line: nameOutput [name for output files]
14. line: dt [time increment for slip rate function. Has to be the same as for Green's functions!]

Example Input:

```
4
20.0  25.0
16.0  19.4  17.5
-15.782 -2.7869
1.23e+19  0.15
122.  40.0  105.0
128 128
-5 -11 -9
1,1
velocity.soil2
0.0
1
NR_sources
0.01
```

KinModel.inp

1. line: rupL, ddW [rupture length, down-dip width, i.e., dimensions of fault plane in m]
2. line: hypoStrike, hypoDip [position of hypocenter on fault along dip, position of hypocenter on fault along dip, in m]
3. line: hypoX, hypoY, hypo [hypocenter coordinates in m]
4. line: M0, fc [seismic moment in Nm and corner frequency in Hz]
5. line: strike, dip, rake (strike, dip, rake of event)
6. line: dx, dt [grid spacing (m), time increment for slip rate function (has to be same as for Green's function!]
7. line: NSources [number of sources]
8. line: seed1, seed2, seed3 [random seeds]
9. line: nameVelMod [name of file containing velocity model]

Example Input:
20000 25000
16000 19400
-15782. -2786.9 17500.
1.23e+19 0.2
122. 40. 105.
200 0.01
20
12124224 12421 534234
velocity.soil2

syn1D_LAH.inp

1. line: subStrike, subDip [# point sources for each subfault (subfaults are interpolated)]
2. line: perturbAz, perturabRake, perturbDip [perturbation of azimuth, rake and dip for the high frequencies]
3. line: fDeterministic, fStochastic, kappa [until frequency fDeterministic radiation pattern is deterministic, above fStochastic it is stochastic. In betweem there is a linear transition, kappa value in s]
4. line: nameSources [name of file containing names of source model files]
5. line: nameStation [name of file containing station locations]
6. line: switchTimeSeries [1: displacement, 2: velocity, 3: acceleration. Note that the post processing programs work on velocity]
7. line switchFormat [1:SAC, 2: TXT. Post processing works on TXT]

Example input:

```
2 2              ! # of point source for each subfault
60.0,   30.0,   15.0   ! Perturbation on strike, rake, and dip
1.0,  3.0, 0.03
source_SCEC.list
stations25
2      ! 1 for Displacement, 2 for Vel., 3 for Acc
2      ! 1 for SAC;  2 for TXT;  3 for Binary
```

LAH rupture model format

1. line: switch1, NSub, switch2 [switch1 is for output, should be 1 for seismic moment, NSub is number of subfault, switch2 is for slip rate function, usually 4]
2. next NSub lines: X, Y, Z, slip, rupture time, rise time, variable, strike, dip, rake [x-ccordinate of subfault, y-coordinate of subfault, z-coordinate of subfault, slip, rise time, rupture time (start of rupture for subfault), variable is either

peak slip rate for LAH or peak time for SAL, strike of subfault, dip of subfault ,rake of subfault]

SAL log file

Each line: slip, rise time, peak time, rupture velocity, peak slip rate [see SAL09 for more info]

Standard rupture format (SRF)

This is documentation provided by Rob Graves. The only change I made is that instead of lon and lat a Cartesian coordinate system is used. Note that this format is very general. In the current implementation only one slip component is used, not three. This format was meant to be as general as possible for future use.

Robert Graves
06/30/2005

Introduction

The general fault surface is represented by a distribution of point sources. In principle, there is no inherent restriction on the geometry of the fault surface or on the spacing and distribution of the point sources used to describe that surface. Each point source description contains all the necessary information to compute the contribution of that point to the total response of the fault rupture. However, it is recognized that most current methods used to develop or model fault ruptures are based on descriptions that employ planar rectangular segments. To facilitate the exchange (and retention) of fault representations in this format, the description presented below allows for the (optional) inclusion of information that specifies the planar segments used to define the fault surface.

Aki & Richards convention for strike, dip and rake are used for all the source descriptions. Slip is specified in three orthogonal directions, two within the subfault surface, and the third in the direction of the outward normal to that surface. This allows for the possibility of time dependent rake and/or fault opening. Figure 1 illustrates the coordinate system conventions used in this description.

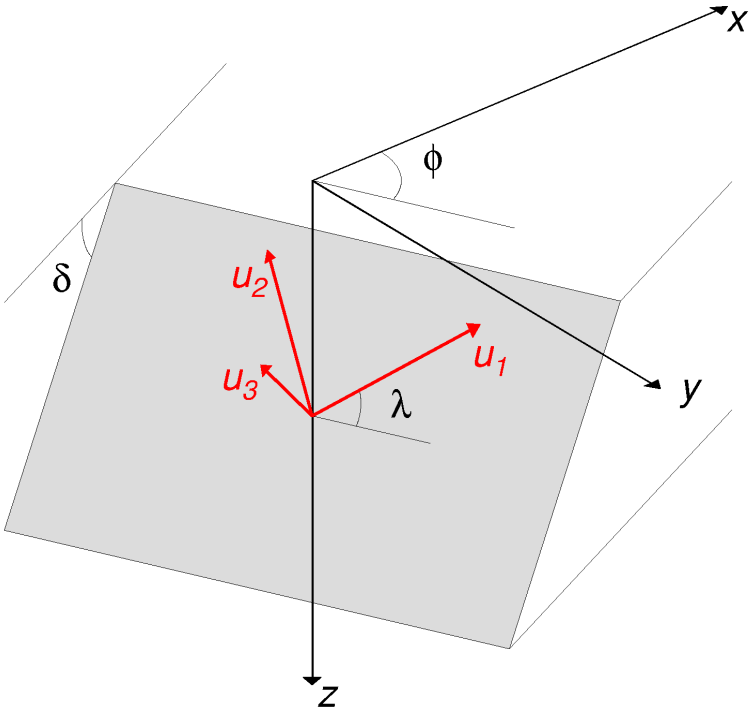

Figure1: Coordinate system conventions for an individual subfault surface. The system (x,y,z) is the global system (e.g., x = north, y = east, z = down). The system (u_1,u_2,u_3) is the local system in which the slip is specified. The axes u_1 and u_2 are tangent to the surface and u_3 is the outward normal to the surface. The strike (ϕ), and dip (δ) are the same as Aki & Richards. Strictly speaking, the rake (λ) defines the orientation of the u_1 axis (with u_2 at $\lambda+90^o$).

For a pure shear dislocation with constant rake, all slip will be in the u_1 direction, thus requiring convolution of only a single STF. Time variable rake needs two orthogonal components (u_1 and u_2) each with a unique STF, thus requiring convolution of two STFs. In this case λ can be specified such that u_1 and u_2 bracket the average rake direction.

Format Specification

The file type as described below is in ASCII format to allow for easy exchange and editing capabilities. The general file format is as follows:

VERSION
HEADER BLOCK (optional)
DATA BLOCK

where

VERSION	= version identifier (e.g., 1.0)
HEADER BLOCK	= general fault description (used for general description of the fault surfaces)
DATA BLOCK	= general point source information covering fault surface

HEADER BLOCK

The optional HEADER BLOCK consists of a series of lines describing the general features of the fault surfaces. Given below is a description for planar segments representing the fault. Additional descriptive formats can be developed and added as needed.

For the planar segment format, the first line is:

PLANE NSEG

where

PLANE = flag specifying that following lines describe planar fault
NSEG = number of segments in fault description

For each segment, the following two lines are needed:

EX EY NSTK NDIP LEN WID
STK DIP DTOP SHYP DHYP

where (for this segment)

EX = top center x-coordinate
EY = top center y-coordinate
NSTK = # of subfaults along strike
NDIP = # of subfaults down-dip
LEN = fault length
WID = fault width
STK = fault strike
DIP = fault dip
DTOP = depth to top of fault
SHYP = along strike location (from top center) of hypocenter
DHYP = down-dip location (from top edge) of hypocenter

The above description is repeated for each additional segment.

DATA BLOCK

Following the optional header lines described above are the required DATA BLOCK lines. The first line in the DATA BLOCK is:

POINTS NP

where

POINTS = flag specifying that following lines describe point sources
NP = number of point source to follow

The remaining lines in the DATA BLOCK contain information about each of the point sources. For each point source, the format is:

```
X       Y       DEP    STK    DIP    AREA   TINIT  DT
RAKE    SLIP1   NT1    SLIP2  NT2    SLIP3  NT3
Sv1[1]
Sv1[2]
Sv1[3]
 . . .
Sv1[NT1]
Sv2[1]
Sv2[2]
Sv2[3]
 . . .
Sv2[NT2]
Sv3[1]
Sv3[2]
Sv3[3]
 . . .
Sv3[NT3]
```

where

X = x-ccordinate of point source
Y = y-coordinate of point source
DEP = depth (km) of point source
STK = strike
DIP = dip
AREA = area of "point" source (cm*cm)
TINIT = initiation time (when rupture reaches subfault center)
DT = time step in slip velocity function
RAKE = direction of u_1 axis (rake direction)
SLIP1 = total slip (cm) in u_1 direction
NT1 = number of time points in slip velocity function for u_1 direction
SLIP2 = total slip (cm) in u_2 direction
NT2 = number of time points in slip velocity function for u_2 direction
SLIP3 = total slip (cm) in u_3 (surface normal) direction
NT3 = number of time points in slip velocity function for u_3 direction
Sv1[1],...,Sv1[NT1] = Slip velocity at each time step for u_1 direction
Sv2[1],...,Sv2[NT2] = Slip velocity at each time step for u_2 direction
Sv3[1],...,Sv3[NT3] = Slip velocity at each time step for u_3 direction

Although the general format given here allows for slip in all three directions (u_1,u_2,u_3), most cases will have only one (u_1) or two (u_1, u_2) non-zero slip components. In these situations, the remaining slip components, as well as the number of time points for the slip velocity, are specified as "0".

Time series format from syn1D_LAH

The naming convention is the following. For station with the entry in the stations file of:

scse
-3517.273682 3144.834717 0.000 0.000000 90.000000

the three created files are: scse.000.gm1D.001, scse.090.gm1D.001, scse.ver.gm1D.001.
The 001 in the end indicates the first source model. If multiple source models are computed the endings will be 002, 003,…
Here is the time series format:

1. line: NSamp, dt [number of time samples, time increment]

the other lines contain the selected ground motion (for example ground velocity) in cm/s .

Time series format from timeHist ('URS-format')

The naming convention is the following. For a ground velocity file with the name scse scse.ver.gm1D.001 (see previous section) three 'URS-format' files are created

the three created files are: scse.000 (velocity), scse.000.dis (displacement), scse.000.acc (acceleration). Note, that there is no model specification (here 001), so it is up to the user to make sure the files for different models are in different folders (see also example section for one way of doing this using simple shell scripts).
Here is the time series format:

1. line: station name, component
2. line: NSamp, dt [number of time samples, time increment]

the other lines contain the selected ground motion (for example ground velocity) in cm/s .

Appendix B

The KinModel rupture model generator

The kinematic rupture model generator is outlined in the thesis. It consists of a number of C source and header files. I programmed most code myself, but there is some code that is from a third party, like the Fourier transform or random number generation. This is indicated in the top lines of the files.

Some of the code is programmed pragmatic, meaning it is possible to speed the code up or make parts of it more elegant. It was developed at the end of my PhD and I did not have time to clean it up. The major required change has to do with the fact that I use a Fourier transform that requires a power of 2 as input. Thus, random fields larger than the actual fault are produced and used throughout the code. This of course makes the code slower. There are also a number of subroutines that are not used anymore or were replaced.

globalParam.h: changing parameters, correlations and marginal distributions

Besides the input file there is another file that sets all the basic parameters. It is called *globalParam.h* and changing these parameters requires to re-compile the code. I chose this because these parameters were intended to be fixed, but it turns out that in many applications it can be useful to change them. Again, a more elegant solution could be found, like set them as default and use getpar if the users wants to make changes. In the following I will describe the parameters inside this file, but there are also comment lines inside the file that explain them

- DKPATH: the path in which the marginal distributions and the covariance matrix is stored. This path needs to be changed to match the folder on the machine of the user.
- NBIN: number of distance bins. The upper bounds of each bin can be set in utlFault.h, the name of the array is distBin
- RISEBODY: the file body of the files containing the marginal distributions of rise time. for example if 'riseP2Kin', there need to be the files *riseP2Kin1.dat*, .., *riseP2Kin5.dat* (assuming NBIN is 5)
- PEAKBODY: same as RISEBODY but for peak time
- VRUPBODY: same as RISEBODY but for normalized rupture velocity (rupture velocity over local shear wave velocity)
- TAPER1: Taper for rupture velocity,i.e., the rupture velocity is at the boundaries of the fault tapered to zero over this distance (in m)
- TAPER2: taper for slip amplitude.
- SLIPFD: Fourier spectral decay for slip amplitude (times two to get power spectral decay)
- RISEFD: Fourier spectral decay for rise time (see Thesis on how related to SLIPFD)
- PEAKFD: Fourier spectral decay for peak time (see Thesis on how related to SLIPFD)

- VRUPFD: Fourier spectral decay for rupture velocity (see Thesis on how related to SLIPFD)
- SLIPA: position of mode for Cauchy law (will be renormalized depending on moment. See also Liu et al (2006) for description of Cauchy law or better ask Daniel Lavallée.
- SLIPB: parameter of Cauchy law that controls heaviness of tail
- X1: lower truncation for Cauchy law (will be rescaled to match average slip of event)
- X2: upper truncation for Cauchy law (will be rescaled to match average slip of event)
- EVENT: This parameter is either set to "SLOW" or "FAST". "SLOW" mean that the marginal distributions for the different distance bins are used If fast is used there is no distance dependency. The marginal distribution for the last bin will be used for all distances. If NBIN 5 and RISEBODY "riseP2Kin" than the name of the file used will be *riseP2Kin5.dat.*
- ITERCORR: how many iterations to use for the computation of the correlated fields. If a small number is used the correlations will differ among models. To always get the same correlations one can set a large number (like 15).
- ADDVEL: This should not be changed. One can add some value to the normalized rupture velocity to make the rupture faster (or use small number to make it slower).

Most of the parameters will not have to be changed. Only the marginal distribution for rupture velocity can be changed for dipping mod III) and vertical (mode II) ruptures. If different parameters should be changed to test their influence I recommend to also make a copy of the used *globalParam.h* file that to store the paramaters used in different scenarios. This avoids confusion and makes it easier to go back and check what was done.

If marginal distributions without distance dependency should be used I recommend to use NBIN=1 and EVENT="FAST".

Description of source files

Below are the different source file names. In mot cases there is a source and a header file. I also list who wrote the code in case there are questions (at least for my code one can email me). Note, that there are two Numerical recipe routines (for uniform and normal distributed random numbers) I use. Those can be easily replaced if necessary

- utlRupTime: contains FD code used to compute rupture times from local rupture velocity. Written by myself
- utl_filter: contains some 1D filter routines, written by F. Scherbaum (PITSA routines). Those are for time series (used for slip rate function)
- utlFilter: Contains routine to perform wavenumber filter that enforces power spectral decay. Also contains taper routine. Written by myself
- utlFault: Routines to set up fault plane, read in various files (like velocity model). Also contains routines to split fault in distance bins and combine distance bins to one fault. Written by myself

- utlProbability: Routines to create random distributions based on Cauchy or based on discrete probability density (from file). Also routines to compute correlation matrix and iterate correlation matrix. The uniform and Gauss random number generation routines are from Numerical Recipes, all other routines are written by myself
- utlKinModel: contains main routine that creates correlated stochastic fields. Also routines to read input and write SRF file. Written by myself
- utlSourceFunction: routine to create source time function. Written by myself
- utlMomentRate: Routiens to compute moment rate function. Also misfot function. This also contaisn a bunch of not used routines in which I was testing various things with respect to scaling rise and peak times. Written by myself
- downhillsimplex.c: routine for downhill simplex. From Carnegie Mellon University, slightly modified by myself to accept cost-function
- ranlib , com, linpack: routines to create 4d normal distribution. Downloaded from http://orion.math.iastate.edu/burkardt/c_src/ranlib/ranlib.html and written by Barry W. Brown
- fft*: Vrious CCMATH Fourier transform routines. Written by Daniel A. Atkinson
- csqrt, carith, pfac, complex: Various routines by CCMATH necessary for fft. Written by Daniel A. Atkinson
- DataStructs.h: used structs throughout code, written by myself
- globalParam.h: Key statistical parameters, see previous section, written by myself
- main.c: main program, written by myself

A few comments about peak and rise time scaling: potential problems and improvements

One of the most crucial choices in the validation is the caling for rise time and peak time. If just the distributions from the dynamic models would be chosen one would underpedict due to the long rise times. It is necessary to scale these accordingly.

I followed Liu et al. (2006), but I decided to fit the whole moment rate function to a Brune spectrum. The tricky part here is the cost-function. This function is the routine *misfit* in the file *MomentRate.c*. Furthermore, in *fitBrune* I added some penalties, for example of the peak time gets shorter than the rise time at points on the fault, or if the norm of the moment rate function divided by the seismic moment of the target is not very close to 1.

One can see that my misfit function looks rather arbitrary. First, I used just the sums of logarithmic differences (absolut), but this can lead to too much weight at the high frequencies (many points). Hence I decided to use the absolute misfit, which gets smaller as the frequencies get larger (because of omega-squared decay for frequencies above the corner). This gave too little weight to the high frequencies, hence I multiply with a power of the frequency smaller than 2. SO there is still a

decrease with increasing frequency, but it is not as fast. The slope was chosen by trial-and error. So if one encounters problems in prediction (too large or too small values of peak ground motion), then this part is probably the reason. In general, if one has a better idea then this part is certainly improvable.

The other part that goes hand in had with this is the corner frequency. I decided to compute it directly from the rupture times instead of have it provided by the user. Right now all is subshear, but if one would include supershear this approach can be dangerous because it can lead to larger corner frequencies and thus to very small peak and rise times.

Appendix C

Additional programs for post processing

Described here are some of the additional programs that can be used for post processing. A lot of this can be done with any program like matlab or Mathematica, but using these little command line tools in a shell script is faster and more convinient.

Low pass filter

In the folder *timeHist* is the program **lpfilt**. It performs a low pass filter using a zero phase butterworth bandpass. Note, that no taper is applied. It should only be used for synthetic time series. The subroutines used are the PITSA filter routines (by F. Scherbaum). It is possible to use it only forward, but this has to be changed in *lpfilt.c* (the variable zph has to be set to 0)The program will scan three line of entry. It works with the time series format which is written by **syn1D_LAH** using the switch TXT

1. name of input file
2. name of output file
3. corner of lowpass (Hz), number of poles

The usage is demonstrated in the shell script *spectra.csh* in the provided example, even though a bandpass (next section) is used there but the concept is the same.

Band pass filter

In the folder *timeHist* is the program **bpfilt**. It performs a low pass filter using a zero phase butterworth bandpass. No taper is applied (only for synthetic). The routine **bpfiltTap** woroks the same way, but uses a cosine taper. This routine can be used with observed data. The subroutines used are the PITSA filter routines (by F. Scherbaum). It is possible to use it only forward, but this has to be changed in *bpfilt.c* (the variable zph has to be set to 0)The program will scan three line of entry. It works with the time series format which is written by **syn1D_LAH** using the switch TXT

1. name of input file
2. name of output file
3. lower corner (Hz), upper corner (Hz), number of poles

The usage is demonstrated in the shell script *spectra.csh* in the provided example

Computing displacement velocity and acceleration in 'URS' format

Given the TXT ground velocity from **syn1D_LAH** it is possible to compute also acceleration and displacement. The format is very similar but slightly different. The velocity will also be written in the new format. It is called 'URS-format' (see Appendix). It scans the file name. The name of the program is **timeHist** and can be found in the folder with the same name. Again, for usage example see shell script *spectra.csh* in the provided example.

Conversion to SAC format (necessary for some of the post processing programs)

The program **toSAC2** converts any 'URS-fomat' file to SAC format. It scans the file name. Again, for usage example see shell script *spectra.csh* in the provided example.

Conversion in one 3 component file

The tool **conv3Comp2** converts the 3 single component files into a three component file. This tool is only used, because the 3 component files are used in the SCEC broadband platform. The header of the files contains some dummy line (for SCEC this information is filled in, but different input files are necessary)

Deconvolution

The tool **deconvBBP** can be used to deconvolve the time series to the seismic basement It reads the 3 component files (*.3comp) and requires the input file *deconvBBP.inp*

Stitch linear and nonlinear

The tool **stitchBBP** stitches linear (3 component files *.3comp) with nonlinear (3 single component files, *nl1D*). The stitching is done in the wavelet domain and is more accurate than frequency matching filter. Typically 1 Hz I used as matching frequency.

Appendix D

Supporting *Mathematica* notebooks

In the folder *Mathematica* are a number of simple Mathematica notebooks forl plotting various things. Note, that much more code exists (like for plotting contours of peak ground motion, doing statistics for multiple models,…). Here I only included the most commonly used routines. You can contact me if you need something more specific, I might have it and save you some work.

checkGreen.nb

To read in the files *test?* , which are created for each depth. Useful for checking if the Greens functions are long enough.

plotSource.nb

Read and plot the LAH and srf source models.

plotSpectra.nb

Routines for plotting Fourier and response spectra.

plotTimeSeries.nb

Tools for plotting and comparing time series. Also includes a template that creates for each station a plot containing time series and pseudo velocity on one page.

Appendix E

Example

In this appendix an example is discussed. Ground motion for a few stations for Northridge are computed using multiple source models. Post-processing routines are used. In addition, the nonlinear site response is applied. The example can be find in the subfolder *Tutorial*.

Compiling the code

I run the code on a MAC using the intel compilers, except **KinModel** for which is use gcc. Most of the code is written in fortran by Pengcheng Liu, the Greens function code is from (Zhu and Rivera, 2002). The code **KinMode***l* is developed by myself, but I use subroutines from other people. Each sub-folder in the folder *src* contains the code and a Makefile. Go into each folder and type 'make'. The make file might have to be edited to use a different compiler. **KinModel** only works with gcc because one of the external subroutines has some problem with ifort. I need to fix that but did not have time. gcc is free available, but icc is usually faster. I will send an updated version once this is fixed.

Computing the Green's functions

Go into the folder *Tutorial/Green*. The input file is *GreenFar.in*, furthermore the velocity file *velocity.soil2* is in this folder. In the file *GreenFar.in* the right path has to be entered (location of this folder on local machine).
If the code was compiled call **green_bankFar** *(*after copying it to the current location or setting a path to it). The Green's functions are computed. This can take a while (hrs), but they only have to be computed once for a given velocity model. After the computation is finished, copy the file *Green_Bank.inf* into the folder *Syn*. Note , that as mentioned in the description of **green_bankFar** in the main document, for each depth a file is written that can be plotted using *checkGreen.nb*.
Potential problems: The path set in GreenFar.in and also written into Green_Bank.inf might be too long. In that case the computation of the time series (later) will give an error message. So it is always good to check *GreenFar.in*. The time window is not long enough and wrap-around effects occur (check *test* files). The value of the time to save before the first arrival was set to 0 (should never be done!!)

Computing the source models using LAH

Go to the folder *Tutorial/Source*. Here the input file *output_LAH.inp* and the velocity model file *velocity.soil2* are located. Nine source models will be computed. Call **KinModel** *(*after copying it to the current location or setting a path to it). After the models are created, copy the following files into the folder *Example/Syn*:

- *source_SCEC.list*
- *the files *.srf*

Potential Problems: The parameters where not set right. For example, if one uses a hypocenter depth that is too small for the chosen position along dip, then the fault extends above the free surface. To check, one should never have negative values in the 3rd column of the LAH source model files (z-component is defined positive down). The top of the fault should also never be exactly at 0 (FK-method cannot handle this)

Computing the source models using SAL

Go to the folder *Tutorial/Sourc2e*. Here the input file *KinModel.inp* and the velocity model file *velocity.soil2* are located. Nine source models will be computed. Call **KinModel** *(*after copying it to the current location or setting a path to it). After the models are created, copy the following files into the folder *Example/Syn2*:

- *source_SCEC.list*
- *the files *.srf*

Potential Problems: The parameters where not set right. For example, if one uses a hypocenter depth that is too small for the chosen position along dip, then the fault extends above the free surface. To check, one should never have negative values in the 3rd column of the LAH source model files (z-component is defined positive down). The top of the fault should also never be exactly at 0 (FK-method cannot handle this)

Computing the time series, linear only

Go to the folder *Tutorial/Syn* or *Tutorial/Syn2*. Now the time series can be computed. The input file is *syn1D_LAH.inp*. Furthermore, there is a station file containing 4 stations named *stations.dat.* Finally, all the files from the previous two steps (*Green_Bank.inf, sopurce_SCEC.list, *.srf*). Call **syn1D_LAH** *(*after copying it to the current location or setting a path to it). The time series should be computed. The output will be in ascii and ground velocity. The subfolder *Tutorial/SynPrecompute2d/* contains pre-computed time series. The folder *Mathematica* contains a Mathematica notebook for quick plotting of the time series
Potential problems: Green's functions not computed for large enough distances or depths. Error in input files. Fault extends above free surface.

Filtering of time series, response and Fourier spectra

In the folder *Tutorial/Syn* is the script **mDirs.csh**. It is a simple shell script which can be used to create one sub-folder for each model and move the corresponding time series into the subfolder. This has to be called first. Then, the script **spectra.csh** can be called. Two arguments are necessary, the lower and upper corner of the bandpass filter that should be used. For example use
./spectra.csh 0.2 15.0

The script is documented. It filters the time series, computes acceleration and displacement. Furthermore, response spectra and Fourier spectra using the acceleration. It also demonstrates how one can convert the TXT files to SAC.
The folder for each model are called mod?, where ? is the number of the model. Each folder will have sub-folder called Filt which will contain the filtered time series and spectra. The notebook *plotSpectra.nb* shows how to plot spectra, *plotTimeSeries,nb* how to plot the time series.

Computing nonlinear site response

The folder *Tutorial/SynPrecompute* or *Tutorial/SynPrecompute2* contains the script **getSite.csh** which contains the steps to compute the nonlinear site response. The linear time series are already pre-computed and stored. Several steps are necessary. Because the scripts used were developed for the SCEC Broadband platform, in a first step it is necessary to create files, which contain all three components (one file per station instead of three). In the next step the time series are copied in the folder SiteIN that was copied in the necessary location. It contains further scripts and input files. First, the program **deconvBBP** is used to deconvolve the linear time series to the seismic basement (files called ****de1D****). Here I use 290 m. Then the script **doSite.csh** is called. It converts the resulting deconvolved files into SAC using **toSAC**. Finally, the code **noah_w** is called to propagate the deconvolved wave back to the free surface using nonlinear wave propagation (files called ****nl1D****). Because surface wave are not accounted for, the code **stitchBBP** is used to stitch the linear files (3 component files) with the nonlinear files using a wavelet matching filter. I use a matching frequency of 1 Hz. He resulting files are called ****gmBB*****. The file to compute the spectra and such is called **spectraBB.csh** and does the same as **spectra.csh** described earlier. The resulting spectra and time series are on the folders mod?/SiteIN/Filt/ where ? is the number of the source model.