

Code Specifications and Performance Report for First Milestone:
Virtual_California Serial Codes

JPL GEM Project

July 12, 2002

John B. Rundle
Jorge S.S. Martins
Paul B. Rundle

University of Colorado
Boulder, CO 80309

Contents:

	Topic	Page
1.	General Description	2
2.	References	2
3.	Code Physics	2
4.	Running Virtual California	3
5.	Example	5
6.	Performance Benchmark Results	6

1. General Description:

This is the Monte Carlo code that generates simulated, realistic earthquakes on an arbitrary fault surface mesh. The topology of the fault mesh is defined by the user. The stress Green's functions are then computed in codes VC_STRESS_GREEN.f and VC_SG_COMPRESS.f and then used as input to the Virtual_California family of codes. The stress Green's functions are then used, together with a user-defined friction model, as input to an initializer code VC_INIT_SER.f. The initializer code is run several times until the user sees all initial transient effects have ceased. Finally, this code, VC_SER.f is used to generate simulated earthquakes.

2. References:

The basic ideas behind these codes are described in the following series of papers:

Rundle, JB, A physical model for earthquakes, 2. Application to Southern California, J. Geophys. Res., 93, 6255 (1988)

Rundle, JB, Linear pattern dynamics in nonlinear threshold systems, Phys. Rev. E, 61, 2418 (2001)

Rundle, PB, JB Rundle, KF Tiampo, JSS Martins, S McGinnis, and W Klein, Phys. Rev. Lett., 87, 148501 (2001)

Rundle, JB, PB Rundle, W Klein, JSS Martins, KF Tiampo, A Donnellan and LH Kellogg, GEM plate boundary simulations for the plate boundary observatory: Understanding the physics of earthquakes on complex fault systems, Pure. Appl. Geophys., in press (2002)

3. Code Physics:

Virtual California uses topologically realistic networks of independent fault segments that are mediated by elastic interactions. (Note that earlier versions had viscoelastic interactions as well, but such are not included in the present code). VC is a "backslip" model, inasmuch as the plate tectonic stress increases is produced by means of applying a negative ("backslip") velocity to each segment whose magnitude is that of the long-term rate of slip on the segment. Since "positive slip" reduces the stress on a fault segment, "negative slip" due to the backslip increases the stress. On each time step, all faults are checked to determine whether the shear stress has reached the failure threshold. Once at least one segment reaches the threshold, the "long time steps" stop, and "short (failure) time steps" (a.k.a. Monte Carlo Sweeps, or mcs) begins. An mcs begins with a check of each site to determine whether it has failed, followed by a parallel updating of each segment. An update of a segment consists of increasing the sudden seismic slip on each segment so that the stress of the segment, considered in isolation, drops to a residual value, plus or minus a random overshoot/undershoot. The elastic stress on all segments is then recalculated, and another mcs is carried out. This iterative process repeats until all segments are below the failure threshold, at which time the mcs time steps cease and the long plate tectonic time steps begin again. Note that

VC also includes a stress-dependent "precursory slip", or "stress leakage" of the type that has been observed in laboratory experiments by TE Tullis (1996) and S Karner and C Marone (2001). The physics of this process is that as the stress on a segment increases, a small amount of stable sliding occurs that is proportional to the level of the stress above the residual. Lab experiments and field data suggest that the parameter called "alpha" (in Rundle et al. 2001) is of the order of a few percent. This parameter is called `facsdp()` or `facrat()` or `facsd()` in this code.

In addition, as described in Rundle (1988), the fault system topology + elasticity + failure law may lead to an unstable, "runaway" dynamics due to the presence of positive eigenvalues, so we introduce a nonlinear, "self-adapting" or "self-correcting" term. The VC initializer code estimates the magnitude of these terms for each segment using a simple but unrealistic dynamics. They are then readjusted by running the VC code with real dynamics until all startup transients (unrealistically large stresses) disappear. The time scale for the transients to disappear depends on the complexity of the problem. Physically, one can imagine that the system evolves on a rough energy landscape, and on average sits at the bottom of a free energy well. Large earthquakes may tend to displace the system from the well allowing non-stationary evolution to occur before the system settles into a new well. The configuration of the well is determined by the magnitude of the positive eigenvalues discussed above. The eigenvalues for each segment are proportional to the derivative of the shear stress with respect to the slip on each fault segment.

4. Running Virtual_California:

The following data files are needed:

1. A fault topology data file -- sample: `VC_FAULTS_1999.d`
2. A fault friction file -- sample: `VC_FRICTION_1999.d`

These can be created in various ways. Note that the fault friction file gets overwritten when you run the actual fault simulator (called `VC_INIT_SER.F`) on an initial run, so you want to make a copy of this and run the codes with the copy.

The basic fortran source files you will need to compute earthquake histories are:

1. `VC_STRESS_GREEN.F` -- computes the stress Green's functions
2. `VC_SG_COMPRESS.F` -- when using viscoelasticity, this code compresses the viscoelastic Green's function using a collocation, or spectral representation
3. `VC_INIT_SER.F` -- this code uses a phony dynamics (allowing both forward and backward slips on a segment) to produce a dynamics in which all the eigenvalues are negative, corresponding to a stable well on the dynamical energy landscape
4. `VC_SER.F` -- this is the actual earthquake simulator code, with the real dynamics in it. It includes the stress smoothing (alpha effect) as well as the real nonlinear dynamics, the Coulomb Failure Functions, and so forth.

Note that you also need a file containing array dimensions, included here as `EQPARAM_4.FOR`

The order for running the codes is:

1. Run VC_STRESS_GREEN using fault topology data file, result is a fault output file with the stress Green's functions

2. Run VC_SG_COMPRESS on the fault output file from 1., to get the final form of the stress Green's function: the viscoelastic Green's functions are compressed

3. Using the stress Green's function file from 2., along with a fault friction file, run VC_INIT_SER to initialize the dynamical variables. The output file from this run should then be repeatedly and iteratively fed back into the VC_Init code several times to refine the dynamical parameters until the number of fault segments failing on any given time step, in both forward and backward slip events, is approximately constant. For the sample fault topology file, this certainly happens after several thousand (~ 5000) time steps. At the moment, the way things are set up, you have to run VC_INIT_SER maybe twice (the code writes an output file that is read as an input file on the next running of VC_INIT_SER)

4. Using the data file that eventually comes out of 3., finally run VC_SER. When you first run this, you will observe another instability develop. You will have to run VC_SER several times, until all transients are gone, and you see that no fault has remained stuck for a time period of 10,000 years or longer (watch output to screen); the number of fault segment failures basically stabilizes, but with fluctuations. Typical time steps would be 1 year, or perhaps .2 year. If needed, the time steps can be increased to 10 years or larger to help eliminate unwanted transient effects more quickly. Another thing to look at is the values of the Coulomb Failure Stress, as well as the fraction of unstable slips, and the values of the cumulative slips themselves. When all segments seem to have turned on and slipping at reasonably regular intervals, you have reached a statistically steady state, in which the dynamics has found a new local free energy minimum, and the point in phase space corresponding to the system state is fluctuating near the bottom of this new potential well.

5. When an output data file is felt to have achieved a statistically steady state, the output data can be viewed with the VC_VISUALIZE code, which is a series of scripts that runs on an IDL base.

To compute and plot the horizontal surface deformation corresponding to a give earthquake data file, you need to do the following. Note that you also need a file, included here, or array dimensions, DEPARM.FOR

1. First run VC_DEF_GREEN.f to compute the deformation, or kinematic Green's functions. These are computed at a square grid of points centered on the x-y midpoint of the fault system.

2. Second, run VC_DEFORM.f to compute the deformation difference between two instants of time. This code can be kept running at an interactive level while plotting is going on.

3. Third, use VC_VISUALIZE to make .ps plots of surface deformation, either as arrows, or as InSAR fringes.

The code VC_VISUALIZE.pro is a set of scripts written on an IDL base (<http://www.rsinc.com/idl/index.asp>) that includes the following capabilities and procedures:

1. An x-y (km) map of the model.
Uses procedure: model.pro
2. A seismicity time-distance plot.
Uses procedure: timdis.pro
3. A plot of slip as a function of fault location.
Uses procedure: slipdis.pro
4. A map of the event superposed on the x-y map of the model.
Uses procedure: event_map.pro
5. A 3-dimensional map view of the model segments.
Uses procedure: model_3d.pro
6. A 3-d map of event slip superposed on the fault map.
Uses procedure: slipmap_3d.pro
7. A 3-d color-coded fault friction map.
Uses procedure: frictionmap_3d.pro
8. A map of horizontal deformation arrows superposed on fault map.
Uses procedure: displ_arrow.pro
9. An image of wrapped InSAR fringes superposed on fault map.
Uses procedure: insarfringe_wrapped.pro

5. Example:

As an example, the data files VC_SIMEQ.d and VC_SA_1999.out are included. The data file VC_SA_1999.out was made by using VC_FAULTS_1999.d with VC_STRESS_GREEN.f and VC_SG_COMPRESS.f. The files VC_SA_1999.out and VC_FRICTION_1999.d, were used as input to VC_INIT_SER.f. Note that the overall factor entered to multiply the nominal time intervals was .44. This factor gives recurrence intervals for great earthquakes on the big bend of the San Andreas similar to those observed in nature, see PB Rundle et al. (2001).

The code VC_INIT_SER.f was then used twice, first with 6000 time steps of 1 yr, then with 5000 time steps of 1 yr. The data file produced by these was then used as input for a number of runs of VC_SER.f, each for 5000 iterations with crude time steps as large as 40 years. These large time steps were used in order to speed the convergence of the system to a statistically steady state (with small fluctuations), having no locked fault segments. This procedure corresponds to annealing the system into a stable potential well on the dynamical energy landscape, through a process of self-adaptation. Finally, a run of VC_SER.f, with 5000 time steps of 1 yr., produced the data file VC_SIMEQ.d. This file can either be plotted using the visualization codes, or used as input for another run of VC_SER.f.

Performance Results of Benchmarking Tests for Major Code Elements of *Virtual_California* Codes

This family of codes perform realistic earthquake simulations on topologically realistic fault systems subject to realistic evolutionary dynamics based on quasi-static elastic fault interactions, together with friction laws observed in laboratory experimentation.

Introduction: We benchmarked two codes on two workstation-class serial machines. Descriptions of the machines are:

- I. A workstation running Slakware Linux, “**Boltzmann**”
 - CPU -- 696.98 MHz Pentium III (Coppermine)
 - Memory -- 256 MB
 - Memory Speed -- 133 MHz

- II. A workstation running Slakware Linux, “**Julia**”
 - CPU -- 997.5 MHz Pentium III (Coppermine)
 - Memory -- 256 MB
 - Memory Speed -- 256 MHz

We ran two of the Virtual California codes for purposes of baseline timing:

1. **VC_Stress_Green.f** Code computes the $(N^2 + N)/2$ elastic interaction coefficients for the fault network specified. We timed the main loop over all main 215 fault segments in the 1999 southern California fault system model.

2. **VC_Ser.f** Code is the main earthquake simulation code, uses interactions, together with friction and failure dynamics to evolve the fault system stress field through time and compute simulated earthquakes. We timed the main time-stepping loop for 10,000 1-yr time steps for the 215 fault segments in the 1999 southern California fault system model.

Results: The results we obtained from benchmarking the codes on the two systems are shown in the table below.

System →	Boltzmann	Julia
Code ↓		
VC_Stress_Green.f	10 m 1.64 s	5 m 37.74 s
VC_Ser.f	29 m 22.16 s	15 m 48.33 s