The Unified Community Velocity Model Software Framework

Patrick Small[a,1,*], David Gil[b,**], Phil J. Maechling[b,**], Ricardo Taborda[c], Thomas H. Jordan[b,d], Geoffrey P. Ely[e], Other Authors TBD

[a]*Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA*
[b]*Southern California Earthquake Center, 3651 Trousdale Parkway, Suite 169, Los Angeles, CA 90089, USA*
[c]*Center for Earthquake Research and Information, and Department of Civil Engineering, University of Memphis, Memphis, TN 38152, USA*
[d]*Department of Earth Sciences, University of Southern California, Los Angeles, CA 90089, USA*
[e]*Leadership Computing Facility, Argonne National Laboratory, Argonne, IL 60439, USA*

Abstract

This paper presents the Unified Community Velocity Model (UCVM) software framework developed by the Southern California Earthquake Center. UCVM is a collection of software tools and application programming interfaces designed to provide standard access to multiple seismic velocity models used in seismology and geophysics research. Seismic velocity models are key components of current research efforts dedicated to advancing our knowledge of the Earth's crustal structure and its influence on the ground response during earthquakes, including both the regional deep geology and local effects due to the geometry, spatial distribution, and material composition of sediments in basins and valleys. In general, the UCVM software framework is designed to facilitate all research activities involving the use of seismic velocity models, but its development has been particularly driven by applications for deterministic physics-based earthquake ground-motion simulation and seismic hazard analysis. The UCVM has been extensively used in modeling and simulation activities in southern California. Here we describe the background that led to the development of the UCVM and its various software components, including advanced high-performance computer tools available within UCVM, and its capabilities in terms of product deliverables and performance. We also present examples of recent applications that use UCVM tools or output datasets.

Keywords:
seismic velocity models, earthquake simulation, high-performance computing, meshing

1. Introduction

The quantitative understanding of the physical world is an essential goal of geoscience research. We use mathematical abstractions to represent the behavior of systems under static and dynamic conditions; and properties such as density and elastic moduli to characterize the capacity of materials to absorbe or transmit forces in stationary and transient processes. In seismology and geophysics, our understanding of physical phenomena associated to earthquakes, their genesis, and effects, depends in a good measure on our knowledge and accurate representation of the geometry and material properties of the Earth's structure, as well as on our capacity to represent the mechanical characteristics of the rupture process that takes place when a seismic fault breaks and the subsequent seismic wave propagation problem. For the former case, we use stress conditions and dynamic rupture models to describe the faulting process. On the other hand, for the latter case, we use seismic velocity and attenuation models to describe the geologic structure of the mantle and crust, and the mechanical properties of the materials in the different geologic units that compose these structures to describe the propagation of seismic waves through basins and sedimentary deposits, and thus determine the characteristics of the ground motion.

Source models and seismic velocity models are therefore the basic input to earthquake simulation. We are interested on how seismic velocity models are built and made available to geoscientists, and in particular, on how these models can help advance physics-based earthquake simulation. We call physics-based earthquake simulation, the modeling approaches that use deterministic numerical techniques—such as the finite element, finite difference, or spectral element methods—to simulate the ground motion in ways that incorporate the physics of earthquake processes explicitly. That is, methods that explicitly solve the associated wave propagation problem. The use of physics-based earthquake simulation has increased considerably over the last two decades thanks to the growth—in capacity and availability—of high-performance computing (HPC) facilities and applications (e.g., Aagaard et al., 2008; Olsen et al., 2009; Bielak et al., 2010; Cui et al., 2010). These simulations have specific applications of great impact in seismology and earthquake engineering in aspects such as the assessment of regional seismic hazard (e.g., Graves et al., 2011).

*Principal corresponding author
**Corresponding author
Email addresses: `patrices@usc.edu` (Patrick Small), `davidgil@usc.edu` (David Gil), `maechlin@usc.edu` (Phil J. Maechling)
[1]This is only a tentative author line-up. Final line-up to be agreed upon by the whole group

Recent simulations have highlighted the importance of velocity models in the accuracy of simulation results (e.g., Taborda and Bielak, 2014). Numerous seismic velocity models have been built for specific regional or local structures and used in particular simulations over the years (e.g., Frankel and Vidale, 1992; Brocher, 2008; Graves, 2008). The need for these models in simulation gave way to the conception of the community velocity models (CVMs). CVMs are seismic velocity models that have been developed, maintained, advanced and used by a community of interested investigators. Some examples of CVMs for the regions of southern and northern California, Utah, and the central United States are those models developed by Kohler et al. (2003); Brocher et al. (2006); Magistrale et al. (2006) and Ramírez-Guzmán et al. (2012).

CVMs have been typically distributed in the form of datasets or collections of files, or in the form of computer programs that can dynamically operate on these datasets and files to provide information about the geometry and material properties of the crust in a particular region. However, these datasets and computer programs have not always been thought carefully from a computational perspective. In addition, recent advances in earthquake simulations, powered by the increasing capability of supercomputers, have increased significantly the computational demand placed on CVMs as input to these simulations.

This paper presents the Unified Community Velocity Model (UCVM), a software framework designed to provide standardized and computationally efficient access to seismic velocity models, developed and maintained by the Southern California Earthquake Center (SCEC). UCVM is a collection of software tools and application programming interfaces (APIs) that facilitate the access to the material properties stored in CVMs. Although UCVM was conceived as a tool to aid physics-based earthquake ground-motion simulation and regional seismic hazard assessment, it can be used in other geosciences and engineering applications. Here, we describe the development of UCVM and its various software components, including features for use in high-performance parallel computers, and present examples of recent applications of UCVM tools in earthquake research.

2. The UCVM Software Framework

The primary functionality provided by UCVM is the ability to query a wide array of CVMs for material properties in standardized formats, independently of the particularities of each dataset or CVM. UCVM achieves this by registering datasets and velocity models into the framework. Registration of a velocity model or dataset consists of creating the appropriate programming application interface (API) to facilitate the communication between the framework utilities and tools, and the velocity models and datasets. Once a velocity model or dataset has been registered with UCVM, a client can use the framework utilities to retrieve information from the models at any geographic point within the coverage region of the model. A client can be either a user or another software. The primary data-point typically retrieved by a client consists of a float triplet with the seismic velocities (V_P and V_S), and the material's density (ρ). At times we refer to this triple as the payload. The UCVM can then be used to produce standardized output in the form of three-dimensional (3D) volumetric datasets, two-dimensional (2D) vertical cross-sections and horizontal slices, and individual data-points. A client can also use other UCVM utilities for plotting and transforming models and datasets.

In order to facilitate access to the models, UCVM conceals each model's local coordinate system behind a generic querying interface. Data points are queried through this interface by geographic latitude and longitude, and a vertical z-coordinate. The framework allows defining the z-axis as either depth below the free surface (in meters, positive downward) or elevation relative to mean sea level (where zero is at sea level, positive upward and negative downward). The framework further extends the standardized interface by allowing multiple velocity models to be aggregated into a single composite model. Composition is accomplished by tiling two or more velocity models in three dimensions according to a user-specified priority ordering. To support this flexible query mechanism consistently across all models, UCVM includes a high-resolution digital elevation model (DEM). The DEM is synthesized from the USGS National Elevation Dataset (citation needed) and the ETOPO1 Global Relief Model (citation needed). An additional advantage to providing the built-in DEM is that the client can retrieve the surface elevation at any query point in addition to the default data-point payload (V_P, V_S), ρ).

With the exception of the Wasatch Front (Utah) CVM, currently the primary focus of UCVM has been on models available for the State of California (and portions of neighboring States). However, the framework has been designed to be easily modified to cover any arbitrary region of the Earth's surface, provided adequate resolution velocity and elevation models exist. Additional details about the models available through UCVM are given in the following section on Community Velocity Models. Subsequent sections provide further information on the main UCVM utilities and APIs. However, due to space limitations, not all UCVM utilities and options can described here. General and advanced users should refer to on-line manuals and documentation. Table 1 provide URL addresses linking to supporting material. The last section of the paper is dedicated to additional aspects on the computational performance of the UCVM framework and two recent case applications.

3. Supported Velocity Models and Datasets

Velocity models vary considerably in terms of area of coverage, depth extent, composition, and resolution. The UCVM framework is flexible in its support for such variability and has been designed to integrate different velocity models, as well as to interact with their individual features seamlessly. UCVM has built-in support for a number of standard CVMs, which the platform utilities identify through a series of corresponding string labels. Table 2 lists the models currently supported in the UCVM framework and provides additional details and references, along with their string labels. While all these models are supported by UCVM, only a fraction of them are included in the automated installation package, the remaining ones need

Table 1: Electronic addresses to UCVM on-line documentation.

Description	URL Address
General Documentation	http://scec.usc.edu/scecpedia/UCVM
General User Guide	http://scec.usc.edu/scecpedia/UCVM_User_Guide
Advanced User Guide (v14.3)	http://scec.usc.edu/scecpedia/UCVM_14.3.0_Advanced_User_Guide
Tutorial (v14.3)	http://scec.usc.edu/scecpedia/UCVM_14.3.0_Tutorial
Visualization Tools	http://scec.usc.edu/scecpedia/UCVM_Visualization
Model Integration	http://scec.usc.edu/scecpedia/UCVM_Model_Integration_Guide
Release Schedule	http://scec.usc.edu/scecpedia/UCVM_Release_Schedule

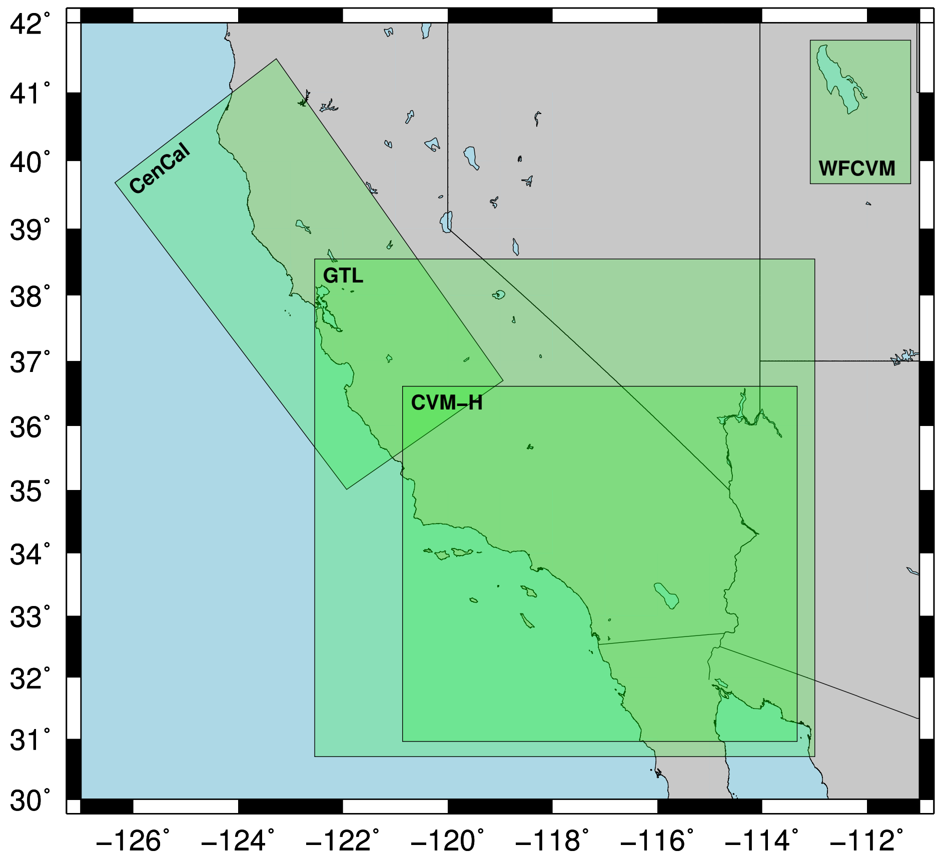

Figure 1: Temporary mock-up figure done in GMT to be improved later once we have all the boxes in Table 2. Surface horizontal projection of the areas covered by the various standard velocity models supported by the UCVM platform.

to be installed manually before they can be accessed through the platform. Additional details are given in Section 4. We indicate in the table which of the models are included in the automated installation and which require manual installation. Figure 1 shows a map with the coverage areas of various CVMs in Table 2.

Velocity models need to be enabled at installation time. UCVM is distributed with an easy installation method which runs a Python script (`ucvm_setup.py`) that prompts the user about which of the automated models are to be installed (see Section 4. If the user wants to enable other velocity models at a later time, the installation process needs to be repeated to enable the desired additional models. Advanced users can also customize the installation to include other models, including user-defined velocity models, which can be added in the form of an *etree* (Tu et al., 2003) database or as a *patch* (rasterized) model. These advanced features are described in detail in the UCVM Advanced User guide (see Table 1).

In addition to the standard velocity models, UCVM includes two geotechnical layer (GTL) models, also listed in Table 2. These models are intended to supplement (replace) the near-surface information in the original velocity models for a

Table 2: Question marks indicate fields that need to be completed. This list needs to be checked carefully. List of velocity models currently supported by the UCVM platform.

Model Name	Region	String Label	Installation	Coverage Coordinates	References
SCEC CVM-H	Southern California	cvmh	Automated	-120.8620, 30.9565 -113.3329, 30.9565 -113.3329, 36.6129 -120.8620, 36.6129	Plesch et al. (2011a) Plesch et al. (2011b) ?
SCEC CVM-S	Southern California	cvms	Automated	?	Magistrale et al. (1996) Magistrale et al. (2000) Kohler et al. (2003)
SCEC CVM-S4.26	Southern California	?	Automated	?	?
SCEC CVM-S5	Southern California	?	Automated	?	?
Hadley-Kanamori 1D	?	1d	Automated	?	?
Carl Tape SoCal	Southern California	tape	Manual	?	?
Broadband 1D	Whittier Narrows	?	Automated	?	?
Graves	Cape Mendocino	cmrg	Manual	?	?
USGS CenCalVM	Central California	cencal	Automated	-126.3532, 39.6806 -123.2732, 41.4849 -118.9445, 36.7022 -121.9309, 35.0090	Brocher (2005) Brocher et al. (2006)
SCEC CVM-NCI	?	cvmnci	Manual	?	?
Lin-Thurber	California Statewide	lt	Manual	?	?
USGS WFCVM	Wasatch Front, Utah	wfcvm	Manual	?	Magistrale et al. (2006)
Ely GTL	Southern California	elygtl	Automated	-122.5410, 30.7046 -112.9958, 30.7046 -112.9958, 38.5460 -122.5410, 38.5460	Ely et al. (2010)
1D GTL	Southern California	1dgtl	Automated	?	?
V_{S30} Maps	Southern California	ucvm, yong		?	Wills and Clahan (2006) Wald and Allen (2007) Yong et al. (2012)

smoother transition from the softer near-surface soil deposits to the stiffer bedrock basement. The first of the two GTL models implements a V_{S30}-based interpolation from the free surface down to a given depth z, following Ely et al. (2010). The second GTL model is a generic one-dimensional (1D) model identical the 1D crustal model. Two interpolation schemes can be used to smooth GTL material properties with the underlying crustal model material properties: a linear interpolation, or the interpolation relationship used by Ely et al. (2010). As in the case of the CVMs, the GTLs and the interpolation schemes are identified with string labels, `elygtl` and `1dgtl` for the V_{S30}-based and the 1D models, respectively. Similarly, the interpolation schemes are identified with the labels `ely` and `linear`. The Ely GTL and interpolation models are used in the CVM-H model by default, with a reference depth of $z = 350$ m. This option can be turned on or off by setting the model flag `USE_GTL = true/false`.

To support the V_{S30}-based GTL model, UCVM has two built-in standard maps for California at 1 arcsec resolution (following USGS NED standards). These maps contain elevation data and V_{S30} data for the region and V_{S30} values are assigned following one of two possible options. Option one implements the procedure by Wills and Clahan (2006) and Wald and Allen (2007). Option two follows Yong et al. (2012). These options are identified by the string labels `ucvm`, which is the default option, and `yong`. These maps are stored as etree databases, and their details are included in Table 2 and Figure 1.

The UCVM framework also provides a background 1D model to support queries at points outside and below the domains covered by the standard CVMs. This option is inactive by default and can be controlled manually by setting the model flag `USE_1D_BKG = true/false`.

4. Download and Installation

The latest version of the UCVM platform is available at: http://hypocenter.usc.edu/research/ucvm/14.3.0/ucvm-14.3.0.tar.gz. We must use a non-versioned URL that always points to the latest version. After successfully downloading this package, the user can follow one of two possible options: the *easy method* or the *custom method*. The easy method uses a Python wrapper to facilitate the installation process that prompts the user about installation paths and default models to be enabled. Figure 2 shows the sequence of terminal commands used during the easy installation process in a Linux workstation.

Alternatively, one can also customize the installation to use advanced features. Some of these features include: option A, option B, option C. As an example, Figure 3 shows the sequence of commands an advanced user would need to manually install UCVM in a Cray supercomputer system with default PGI

```
> tar zxvf ucvm-14.3.0.tar.gz
> cd ./UCVM
> ./ucvm_setup.py
  It looks like you are installing UCVM for the first time.
  Where would you like UCVM to be installed?
  Enter path or blank to use default:

  <user path entry>

  Would you like to download and install CVM-S4?
  Enter yes or no:

  <yes,no>

  [...]
```

Figure 2: Easy installation procedure. Shell commands are indicated by the symbol >. This example needs to be completed with a full installation sequence.

compilers with static build and IO buffering, enabling the Proj4 and Etree libraries and the model CVM-S. Note that UCVM requires GNU GCC compiler 4.3+; therefore, in the example in Figure 3, the user needs to switch to the GNU compilers. When installing UCVM manually, three flags need to be provided for each model to be properly configured, namely a `lib-path`, a `model-path`, and an `include-path`. Similarly, each library needs a `lib-path` and an `include-path`. The last command in the example in Figure 3, `make check` will test the installation to make sure that UCVM was deployed correctly.

5. Using UCVM

The UCVM platform offers a seriers of commands that can be used in sequence or in parallel. This section details the most relevant of these commands. The single-core commands can be run in any system where the platform has been successfully installed. The parallel commands, however, require a system where the standard Message Passing Interface (MPI) library and compilers are available. Single-core commands are useful to most users interested in exploring the properties of the regions covered by the models supported by the platform. On the other hand, advanced users needing to build large-scale (regional) materialized velocity models for earthquake modeling and simulation are the more likely to use the MPI commands.

In either case, before calling on UCVM commands, the user needs to make sure the main package configuration file is in place and has the desired setup. This file is used by UCVM to identify which components will be used when executing the commands. That is, is the file where the paths to all configured models and maps are specified, as well as any model flags are defined. The UCVM installer sets up this file automatically at `$UCVM_DIR/conf/ucvm.conf`. However, there are a number of situations where a user will want to modify this file (e.g., when adding a new model manually). Figure 4 shows an example of a configuration file in which the models CVM-S and CVM-H are enabled, along with the 1D and GTL models used in the background of CVM-H. The following sections expand on how

```
> module swap PrgEnv-pgi PrgEnv-gnu
> module load iobuf
  [...]
> UCVM_DIR=<user defined path>
> UCVM_VERSION=14.3.0
> ROOT_URL=http://hypocenter.usc.edu/research/ucvm
> LIBS_URL=$ROOT_URL/$UCVM_VERSION/libraries
> MODELS_URL=$ROOT_URL/$UCVM_VERSION/models
  [...]
> cd $UCVM_DIR
> wget $LIBS_URL/proj-4.8.0.tar.gz
> wget $LIBS_URL/euclid3-1.3.tar.gz
> wget $MODELS_URL/cvms4.tar.gz
  [...]
> tar xzvf proj-4.8.0.tar.gz
> cd proj-4.8.0
> ./configure --prefix=$UCVM_DIR/lib/proj-4 --with-jni=no
> make
> make install
  [...]
> cd $UCVM_DIR
> tar xvzf euclid3-1.3.tar.gz
> cd euclid3-1.3
> ./configure --prefix=$UCVM_DIR/lib/euclid3
> make; make install
  [...]
> cd $UCVM_DIR
> tar xzvf cvms4.tar.gz
> cd CVM-S
> ./configure --prefix=$UCVM_DIR/model/cvms4
> make
> make install
  [...]
> cd $UCVM_DIR
> tar xzvf ucvm-14.3.0.tar.gz
> cd UCVM
> ./configure \
  --prefix=$UCVM_DIR \
  --enable-iobuf \
  --enable-static \
  --with-etree-include-path=$UCVM_DIR/lib/euclid3/include \
  --with-etree-lib-path=$UCVM_DIR/lib/euclid3/lib \
  --with-proj4-include-path=$UCVM_DIR/lib/proj4/include \
  --with-proj4-lib-path=$UCVM_DIR/lib/proj4/lib \
  --enable-model-cvms \
  --with-cvms-lib-path=$UCVM_DIR/model/cvms4/lib \
  --with-cvms-model-path=$UCVM_DIR/model/cvms4/src \
  --with-cvms-include-path=$UCVM_DIR/model/cvms4/src
> make
> make install
> make check
  [...]
```

Figure 3: Advanced installation procedure showing the commands a user will follow when working on a Cray supercomputer system with default PGI compilers and Lustre filesystem. Note that UCVM requires GNU GCC compilers, version 4.3+. This particular set of instructions installs the model CVM-S (version 4) with the Proj and Etree libraries using static build and enabling IO-buffering. Shell commands are indicated by the symbol >. This example needs to be checked.

5.1. *Single-Core Commands*

We include here a short description of the single-core commands that are most used, or that we believe are most useful to the majority of users. A complete description of all the single-core commands available in UCVM can be found in the documentation links provided in Table 1.

```
# UCVM config file

# UCVM model path
ucvm_interface=map_etree
ucvm_mappath=/path/to/ucvm-14.3.0/model/ucvm/ucvm.e

# SCEC CVM-S
cvms_modelpath=/path/to/ucvm-14.3.0/model/cvms4/src

# SCEC CVM-H
cvmh_modelpath=/path/to/ucvm-14.3.0/model/cvmh1191/model

# 1D
1d_modelpath=/path/to/ucvm-14.3.0/model/1d/1d.conf

# 1D GTL
1dgtl_modelpath=/path/to/ucvm-14.3.0/model/1d/1d.conf

# Model flags
cvmh_param=USE_1D_BKG,True
cvmh_param=USE_GTL,True
```

Figure 4: UCVM configuration file example. Before running UCVM commands, the user needs to prepare a configuration file. This file is used by UCVM to identify what models and libraries are being used by the client. This example corresponds to a UCVM installation where only the models CVM-S and CVM-H were enabled.

5.1.1. *The* `ucvm_query` *command*

This is the core tool of UCVM for querying the models supported by UCVM. Any set of crustal and GTL velocity models may be selected and queried in order of preference. Points may be queried by longitude, latitude and depth or longitude, latitude and elevation. These coordinate conversions for a particular model are handled transparently (Not sure of what this means, we need to check the wording here to be as clear as possible.). Here is an example of how to run this command.

```
> ./ucvm_query -f $UCVM_DIR/conf/ucvm.conf -m cvms ????
```

Note that the location of the package configuration file needs to be passed to the command using the flag "`-f`" and that the model to be used is set with the "`-m`" flag.

5.1.2. *The* `basin_query` *command*

The command `basin_query` allows you to retrieve the depth at which a given V_S-threshold is first crossed. By default, the threshold value is set to be $V_S = 1,000$ m/s, but that can easily be changed with the "`-v`" flag. Here is an example of this command for a V_S-threshold of 2,500 m/s. we need to also put print the output of this query.:

```
> ./basin_query $UCVM_DIR/conf/ucvm.conf -m cvms -v 2500
```

5.1.3. *The* `ucvm2mesh` *command*

This command generates a mesh or grid in either IJK-12, IJK-20, IJK-32, or SORD formats. The formats IJK-12, IJK-20, IJK-32, for instance, are used in discrete finite difference models for wave propagation simulations by the code AWP-ODC (?) (Cui et al., 2010). SORD, on the other hand, is a finite element code used to simulate dynamic rupture processes. The `ucvm2mesh` takes as input a configuration file that is passed on as an argument using the flag "`-f`". This file specifies the model to be used, the region size, and the resolution of the mesh or grid. The following example shows how to execute this command.

```
> ./ucvm2mesh -f ./ucvm2mesh_example.conf
```

5.1.4. *The* `ucvm2etree` *command*

The command `ucvm2etree` builds a materialized model in the form of an etree database, which uses an octree unstructured mesh-like format that adjust octant sizes to a specific resolution based on the material properties in the CVM being used at every point in a given volume. Resulting etrees built using this command are stand-alone discrete representations of a given velocity mode, and can be used as input models for other operations using UCVM. They can also be queried separately using APIs distributed with the etree library. In wave propagation problems, etrees are used by the code Hercules (**??**) in earthquake ground motion simulations. As in the previous case, this command takes an argument input configuration file. The following example shows how to execute this command.

```
> ./ucvm2etree -f ./ucvm2etree_example.conf
```

Note that `ucvm2etree` is the basic serial version of other parallel (MPI) commands used to build large materialized models in high performance computer systems. Parallel tools for building etrees are useful because regional size simulations require models that can be of the order of hundreds of gigabytes to terabytes and can take significant time and resources to build. Thus, for large etrees, we strongly recommend using the `ucvm2etree-extract-MPI`, `ucvm2etree-sort-MPI`, and `ucvm2etree-merge-MPI` commands explained below.

5.1.5. *Other mesh and etree tools*

UCVM also provides other single-core serial commands to manipulate meshes and etrees. The additional mesh-related commands are: `mesh-check`, `mesh-op`, and `mesh-strip-ijk`. The command `mesh-check` does a basic quality assurance check of a mesh file. It checks that each record in the file is of the correct size makes sure that each value is not `NaN`, infinity, or negative. The command `mesh-op` subtracts a mesh from another mesh and outputs the difference. And the command `mesh-strip-ijk` converts an IJK-20 or IJK-32 mesh to an IJK-12 formatted mesh. Usage examples of each of these commands are shown next.

```
> ./mesh-check new_mesh.mesh IJK-12
> ./mesh-op diff ./inmesh1 ./inmesh2 IJK-12 ./outmesh
> ./mesh-strip-ijk ijk20_mesh IJK-20 output_mesh
```

Additional etree manipulation commands include: `ecoalesce`, `ecompact` and `grd2etree`. The command `ecoalesce` aggregates octants in homogeneous regions. It helps compact the physical representation of an etree by replacing eight adjacent octants of the same size and properties with a single octant twice that preserves the same material properties (i.e., it replaces eight identical leaf octants at level n by their parent octant at level $N - 1$). The command `ecompact`, on the other hand, helps compact an etree by removing empty unused space in the etree data structure on disk, effectively reducing the file's size. The `ecompact` is usually run following the `ecoalsce`. The command `grd2etree` extracts an etree map (a one octant-thick etree with mapping information) from a set of DEM and V_{S30} grid files in ArcGIS Gridfloat format. This command is accompanied by the command `grd_query`, which can be used to query data from ArcGIS grid files in GridFloat format. The following are usage examples of these commands.

```
> ./ecoalesce original.etree coalesced.etree
> ./ecompact original.etree compacted.etree
> ./grd2etree -f ./grd2float_sample.conf
```

5.2. MPI Commands

As mentioned before, UCVM provides additional tools for accessing and manipulating CVMs using parallel applications that can be run in high-performance computer systems. These MPI commands provide the same basic operations of their equivalent single-core commands but provide the accelerated performance of parallelism. In general, these are all embarrassingly parallel processes and all the operations perform by each core are independent of each other. Interprocessor communications are limited to load distribution coordination at launch and collection of results at the end.

5.2.1. The `basin_query_mpi` command

This command is the parallel version of `basin_query`. It creates a binary float file listing the depths at which V_S crosses the user-defined threshold. The utility works by partitioning the points across the available processors, querying each point in parallel, and then amalgamating the results into one file. The command takes arguments preceded by flags for the output file, the UCVM configuration file, the velocity model to be used, the vertical step size (in meters), the threshold depth, the origin coordinates (in longitude and latitude), the horizontal gridding size (in degrees), and the number of grid points to the east (x) and to the north (y). The following example outputs the data to `out.file` using the model `cvms` defined in the configuration file `ucvm.conf`. It queries each point at 20 m increments on the z-axis until reaching $V_S = 1000$ m/s starting at longitude and latitude coordinates $-118°, 34°$, and then goes by 0.01°in both directions for 101 points to reach $-117°, 35°$.

```
> mpirun -np 4 ./basin_query_mpi -b out.file -f ucvm.conf \
  -m cvms -i 20 -v 1000 -l 34,-118 -s 0.01 -x 101 -y 101
```

Note that in the example above, the call to `basin_query_mpi` is preceded by a call to `mpirun` and a request for 4 cores.

5.2.2. The `ucvm2mesh-mpi` command

This tools is the parallel equivalent of `ucvm2mesh`, and can be used to generate meshes in either IJK-12, IJK-20, IJK-32, or SORD formats. In this case, unlike with `basin_query_mpi`, the arguments are passed on in the run configuration file.

```
> mpirun -np 768 ucvm2mesh-mpi -f ./ucvm2mesh_example.conf
```

5.2.3. The etree MPI commands

The operations equivalent to the single-core command `ucvm2etree` are done by a set of three commands, `ucvm2etree-extract-MPI`, `ucvm2etree-sort-MPI` and `ucvm2etree-merge-MPI`. All three take as input the same run configuration-file and are executed in this order as shown in the following example:

```
> mpirun -np 1025 ucvm2etree-extract-MPI \
  -f ./ucvm2etree_example.conf
> mpirun -np 1024 ucvm2etree-sort-MPI    \
  -f ./ucvm2etree_example.conf
> mpirun -np 1024 ucvm2etree-merge-MPI   \
  -f ./ucvm2etree_example.conf
```

Here, the first operation, `ucvm2etree-extract-MPI`, divides the etree region into a user-defined set of columns which are distributed to all processors. The rank-0 processor works as a dispatcher and farms out each column to all the other processors in a pool of N cores. Each core queries the chosen CVM independently in an embarrassingly parallel process. Since the rank-0 processor does not participate of the extraction, `ucvm2etree-extract-MPI` requires $N + 1$ processors. The output of this first step is a set of N sub-etrees. Subsequently, `ucvm2etree-sort-MPI` sorts these sub-etrees so that each file is in local pre-order (following a z-order). Again, the is an embarrassingly parallel operation in which each rank reads in one of the sub-etrees produced by the previous operation and outputs its own sorted etree. Finally, `ucvm2etree-merge-MPI` merges N locally sorted sub-etrees into a final, compacted etree.

5.3. Configuration Files

In the sections above we made mention two two types of configuration files used by UCVM. The first of these files is the UCVM configuration file, which is automatically built in the easy installation process or should be prepared by the user in the advanced installation mode. This file is used by UCVM at runtime to identify which models have been pre-compiled and are ready to be used. This configuration file is used explicitly and must be passed as an argument to the commands `ucvm_query` and `basin_query`. An example of the file's format and content is shown in Figure 4. The mesh and etreee commands, on the other hand, use the UCVM configuration file implicitly and, instead, require a different type of configuration file to be passed as an argument, as it was shown in the examples above. In particular, `ucvm2mesh`, `ucvm2etree`, `./grd2etree`, and their equivalent MPI commands require the composition of this additional file. An example of this second configuration file is shown in Figure 5. Other examples are available at: give a valid URL, maybe reference Table 1 again.

6. Additional Utilities and Tools

6.1. Small-Scale Heterogeneities Tools

The last decade has seem a tremendous growth on high performance computing and applications dedicated to earthquake ground motion simulations which is leading the charge toward performing deterministic simulations at high frequencies of engineering interest ($f_{\max}$ 0–10 Hz). With increasing simulation

```
# RUNNING config file




coordinates

resolution

need example




# end
```

Figure 5: Example of a configuration file as needed for the command...

frequencies, seismic velocity models must not only be accurate representations of a region's crustal structure at the geologic scale and represent the geotechnical characteristics of basins and deposits, but should also provide for the adequate representation of the scattering characteristics of the typical heterogeneities observed in geomaterials. To this end, UCVM implements an algorithm generates models that incorporate small-scale heterogeneities to any underlying velocity model. The command `ssh_generate` generates a binary float file of heterogeneities on a regular grid space with a normalized amplitude. Then, the command `ssh_merge` adds the heterogeneities multiplied by some scaling factor to the model. The introduction of the heterogeneities is actually done by applying the scaled perturbation to the slowness of the underlying mesh velocities, and then converted back to the seismic velocity. The following example generates and merges a set of small scale heterogeneities to ... describe example.

6.2. *Plotting Utilities*

This section briefly describes a collection of utilities provided with the UCVM platform. These utilities are written in Python scripts that operate as wrappers around the basic single-core commands explained above, and work to produce specific output data commonly used for plotting. Because of brevity, we cannot fully describe here all the options and parameters that each of these commands can take as input and will only illustrate their operability. Additional details can be found at the UCVM User Guide (see Table 1).

6.2.1. *The* `cross_section.py` *utility*

This utility plots a cross section of a model given two bounding latitude and longitude points, the depth of the cross section, and a few other parameters. The output of this command is either an image (PNG?) or a text file (ascii?) containing the data from the underlying velocity model. It can be run in interactive mode. The following example, for instance, plots a cross section from zero depth down to 10 km at horizontal and vertical grid-sizes of 100 m for the values of V_S in the CVM-S model using a discretized color scale between the origin point at coordinates $-34, 118$ and the ending point at coordinates $-35. -117$, and outputs the image to the file `image.png`. The result is shown in Figure 6a

```
> ./cross_section.py -s 0 -e 10000 -h 100 -v 100 -d vs \
  -c cvms -a d -o 34,-118 -f 35,-117 -i image.png
```

6.3. *The* `horizontal_slice.py` *utility*

This utility plots a horizontal slice of a model given two bounding latitude and longitude points, the depth at which the slice is to be taken, and other additional parameters. The output can be either an image or a text file containing the extracted data points from the model. The following example plots a horizontal slice at zero depth (surface) with... give an explanation that matches the figure. The resulting plot is shown in Figure 6b.

```
> ./horizontal_slice.py -b 34,-118 -u 35,-117 -s 0.01 \
  -e 0 -d vs -c cvms -a d -i image.png
```

6.4. *The* `vs30.py` *utility*

This utility generates a V_{S30} map (i.e., the average V_S over the top 30 m) of a model given two bounding latitude and longitude points, along with a few other settings. As with the previous utilities, the output of this command can be either an image or a text file. The following example plots the V_{S30}map obtained from... give an explanation that matches the figure. Figure 6c shows the result.

```
> ./vs30.py -b 34,-118 -u 35,-117 -s 0.01 -z 0.1 -c cvms
  -a d -i image.png
```

6.5. *The* `z10.py` *and* `z25.py` *utilities*

Both these utilities are intended to plot the depth of basins in a given model. The `z10.py` utility generates an isosurface map for the depths at which $V_S = 1000$ m/s. The equivalent `z25.py` utility perfomrs the same operation but for the depth at which $V_S = 2500$ m/s is reached. The output in either case, as with the previous plotting utilities, can be an image or a text file with the corresponding data points within a box defined by two bounding latitude and longitude coordinates. The following example plots the depth of the... describe example, and the resulting image is shown in Figure 6d.

7. Recent UCVM Applications

7.1. *Chino Hills – Hercules results using CVM-S and CVM-H*

7.2. *Chino Hills – AWP results using SSH*

7.3. *Something involving inversion and CVM-S4.26?*

8. Final Remarks

References

Aagaard, B.T., Brocher, T.M., Dolenc, D., Dreger, D., Graves, R.W., Harmsen, S., Hartzell, S., Larsen, S., McCandless, K., Nilsson, S., Petersson, N.A., Rodgers, A., Sjogreen, B., Zoback, M.L., 2008. Ground-motion modeling of the 1906 San Francisco earthquake, Part II: Ground-motion estimates for the 1906 earthquake and scenario events 98, 1012–1046. URL: http://www.bssaonline.org/cgi/content/abstract/98/2/1012.

PENDING

(a) Cross section example of CVM-S showing V_S profiles in the Los Angeles basin produced with the `cross_section.py` utility.

PENDING

(b) Horizontal slice of CVM-S showing surface V_S on the Los Angeles basin produced with the `horizontal_slice.py` utility.

PENDING

(c) Surface V_{S30} map of the Los Angeles produced with the `vs30.py` utility.

PENDING

(d) Depth of the Los Angeles basin at $V_S = 1000$ m/s produced with the `z10.py` utility.

Figure 6: Examples of the images obtained using the plotting utilities available in UCVM

Bielak, J., Graves, R.W., Olsen, K.B., Taborda, R., Ramírez-Guzmán, L., Day, S.M., Ely, G.P., Roten, D., Jordan, T.H., Maechling, P.J., Urbanic, J., Cui, Y., Juve, G., 2010. The ShakeOut earthquake scenario: Verification of three simulation sets 180, 375–404. URL: http://dx.doi.org/10.1111/j.1365-246X.2009.04417.x, doi:10.1111/j.1365-246X.2009.04417.x.

Brocher, T.M., 2005. Compressional and shear wave velocity versus depth in the San Francisco Bay Area, California: Rules for USGS Bay Area Velocity Model 05.0.0. Technical Report OFR-2005-1317. U.S. Geological Survey. URL: http://pubs.usgs.gov/of/2005/1317/.

Brocher, T.M., 2008. Compressional and shear-wave velocity versus depth relations for common rock types in northern California 98, 950–968. URL: http://www.bssaonline.org/cgi/content/abstract/98/2/950, doi:10.1785/0120060403.

Brocher, T.M., Aagaard, B.T., Simpson, R.W., Jachens, R.C., 2006. The USGS 3D seismic velocity model for northern California 87, Fall Meet. Suppl., Abstr. S51B–1266.

Cui, Y., Olsen, K., Jordan, T., Lee, K., Zhou, J., Small, P., Roten, D., Ely, G., Panda, D., Chourasia, A., Levesque, J., Day, S., Maechling, P., 2010. Scalable earthquake simulation on petascale supercomputers, in: SC '10: Proc. of the 2010 ACM/IEEE Int. Conf. for High Performance Computing, Networking, Storage and Analysis, pp. 1–20. doi:10.1109/SC.2010.45.

Ely, G.P., Jordan, T.H., Small, P., Maechling, P.J., 2010. A Vs30-derived near-surface seismic velocity model, in: Abstr. AGU Fall Meet., San Francisco, California, December 13–17. URL: http://web.alcf.anl.gov/~gely/pub/Ely2010-AGU-Vs30-GTL.pdf.

Frankel, A., Vidale, J., 1992. A three-dimensional simulation of seismic waves in the Santa Clara Valley, California, from a Loma Prieta aftershock 82, 2045–2074. URL: http://www.bssaonline.org/cgi/content/abstract/82/5/2045.

Graves, R., Jordan, T., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K., Okaya, D., Small, P., Vahi, K., 2011. CyberShake: A physics-based seismic hazard model for Southern California 168, 367–381. doi:10.1007/s00024-010-0161-6.

Graves, R.W., 2008. The seismic response of the San Bernardino basin region during the 2001 Big Bear lake earthquake 98, 241–252. URL: http://www.bssaonline.org/cgi/content/abstract/98/1/241, doi:10.1785/0120070013.

Kohler, M.D., Magistrale, H., Clayton, R.W., 2003. Mantle heterogeneities and the SCEC reference three-dimensional seismic velocity model version 3 93, 757–774. URL: http://www.bssaonline.org/cgi/content/abstract/93/2/757, doi:10.1785/0120020017.

Magistrale, H., Day, S., Clayton, R.W., Graves, R., 2000. The SCEC southern California reference three-dimensional seismic velocity model version 2 90, S65–S76. URL: http://www.bssaonline.org/cgi/content/abstract/90/6B/S65, doi:10.1785/0120000510.

Magistrale, H., McLaughlin, K., Day, S., 1996. A geology-based 3D velocity model of the Los Angeles basin sediments 86, 1161–1166. URL: http://www.bssaonline.org/cgi/content/abstract/86/4/1161.

Magistrale, H., Olsen, K.B., Pechmann, J.C., 2006. Construction and Verification of a Wasatch Front Community Velocity Model. Technical Report 06HQGR0012. U.S. Geological Survey. URL: http://earthquake.usgs.gov/research/external/reports/06HQGR0012.pdf.

Olsen, K.B., Day, S.M., Dalaguer, L.A., Mayhew, J., Cui, Y., Zhu, J., Cruz-Atienza, V.M., Roten, D., Maechling, P., Jordan, T.H., Okaya, D., Chourasia, A., 2009. ShakeOut-D: Ground motion estimates using an ensemble of large earthquakes on the southern San Andreas fault with spontaneous rupture propagation. Geophys. Res. Lett. 36, L04303. doi:10.1029/2008GL036832.

Plesch, A., Tape, C., Graves, R., Shaw, J., Small, P., Ely, G., 2011a. Updates for the CVM-H including new representations of the offshore Santa Maria and San Bernardino basin and a new Moho surface, in: Proc. SCEC Annu. Meet.

Plesch, A., Tape, C., Shaw, J., Small, P., Ely, G., Jordan, T., 2011b. User Guide for the Southern California Earthquake Center Community Velocity Model: SCEC CVM-H 11.9.0. Harvard University and University of Southern California. URL: http://scec.usc.edu/scecwiki/images/5/51/Cvmh_manual.pdf.

Ramírez-Guzmán, L., Boyd, O.S., Hartzell, S., Williams, R.A., 2012. Seismic velocity model of the Central United States (Version 1): Description and simulation of the 18 April 2008 Mt. Carmel, Illinois, earthquake 102, 2622–2645. doi:10.1785/0120110303.

Taborda, R., Bielak, J., 2014. Ground-motion simulation and validation of the 2008 Chino Hills, California, earthquake using different velocity models 104, in press.

Tu, T., López, J., O'Hallaron, D., 2003. The Etree Library: A System for Manipulating Large Octrees on Disk. Technical Report CMU-CS-03-174. School of Computer Science, Carnegie Mellon University. Pittsburgh, Pennsylvania. URL: http://www.cs.cmu.edu/~euclid/.

Wald, D.J., Allen, T.I., 2007. Topographic slope as a proxy for seismic site conditions and amplification 97, 1379–1395. doi:10.1785/0120060267.

Wills, C.J., Clahan, K.B., 2006. Developing a map of geologically defined site-condition categories for California 96, 1483–1501. doi:10.1785/0120050179.

Yong, A., Hough, S., Iwahashi, J., Braverman, A., 2012. (2012), a terrain-based site conditions map of california with implications for the contiguous united states 102, 114–128. doi:10.1785/0120100262.