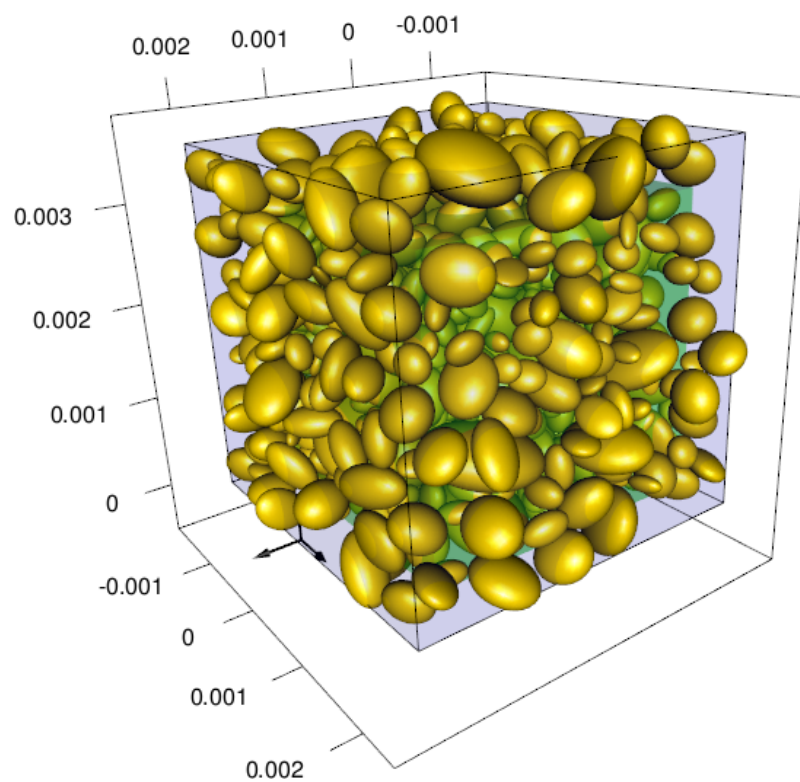


GranularSim Installation Guide

Version Version 18 .1

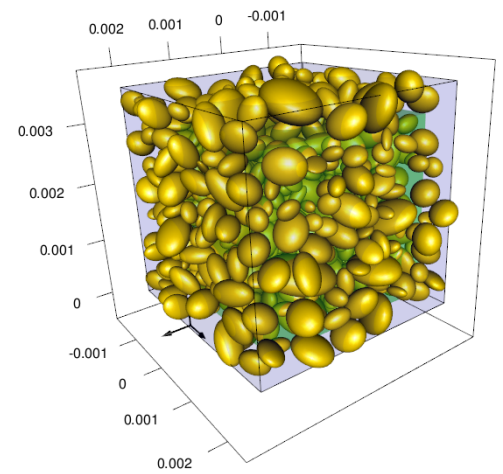
January 31, 2018



Biswajit Banerjee

Copyright © 2015-2018 Parresia Research Limited

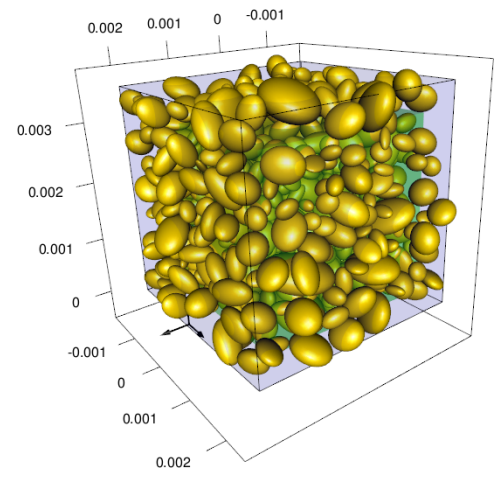
The contents of this manual can and will change significantly over time. Please make sure that all the information is up to date.



Contents

| | | |
|----------|---|-----------|
| 1 | Overview of GranularSim | 5 |
| 1.1 | Obtaining the Code | 5 |
| 1.2 | Developer version | 6 |
| 1.2.1 | Getting the latest version, adding files etc. | 6 |
| 2 | Prerequisites and library dependencies | 7 |
| 2.1 | Using a package manager | 7 |
| 2.1.1 | Compilers | 7 |
| 2.1.2 | Cmake | 7 |
| 2.1.3 | Boost | 7 |
| 2.1.4 | MPI libraries | 7 |
| 2.1.5 | OpenMP libraries | 8 |
| 2.1.6 | Eigen3 library | 8 |
| 2.1.7 | XML libraries | 8 |
| 2.1.8 | JSON libraries | 8 |
| 2.1.9 | Zlib compression library | 8 |
| 2.1.10 | GoogleTest library | 8 |
| 2.1.11 | VTK libraries | 8 |
| 2.1.12 | Qhull library | 8 |
| 2.2 | Building from source | 8 |
| 3 | Configuring and Building GranularSim | 9 |
| 3.1 | Basic instructions | 9 |
| 3.1.1 | The GranularSim repository | 9 |
| 3.1.2 | Out-of-source optimized build | 9 |
| 3.1.3 | Out-of-source debug build | 9 |
| 3.1.4 | Clang compiler: | 9 |
| 3.2 | Compiling the code: | 10 |
| 4 | Installing VisIt | 11 |
| 4.1 | Directory structure | 11 |

| | | |
|-------|--------------------------------|----|
| 4.2 | Using the VisIt install script | 11 |
| 4.3 | Using the VisIt build script | 11 |
| 4.3.1 | Remote visualization | 12 |
| 4.3.2 | Host Profile | 12 |



1 — Overview of GranularSim

GRANULARSIM is a rewritten and updated version of the parallel discrete element code PARALLELIP3D developed at the University of Colorado at Boulder, with the intent of having a more general code that can be improved and appended to more easily. The primary use case is the simulation of soils and other granular materials.

The main components of GRANULARSIM include a discrete element (DEM) code that can handle ellipsoidal particles, a peridynamics (PD) code that couples to DEM particles, and a smoothed particle hydrodynamics (SPH) code that also couples to DEM particles. In addition, the code is able to handle special boundary conditions that are needed for hierarchical upscaling.

GRANULARSIM provides outputs in VTK format and results can be visualized either with PARAVIEW or with VisIt (<https://wci.llnl.gov/simulation/computer-codes/visit/>).

1.1 Obtaining the Code

GRANULARSIM can be obtained either as a zipped archive (approx. 550 Mb) from

```
https://github.com/bbanerjee/ParSim/archive/master.zip
```

or cloned from Github using either HTTP

```
git clone --recursive https://github.com/bbanerjee/ParSim.git
```

or SSH

```
git clone --recursive git@github.com:bbanerjee/ParSim.git
```

The above commands check out the PARSIM source tree and installs it into a directory called **ParSim** in the user's home directory. The PARSIM code contains several research codes. The main GRANULARSIM code can be found in the **ParSim/GranularSim/src** directory. Manuals for GRANULARSIM are in **ParSim/GranularSim/Manuals**.

You may also need to install **git** on your machine if you want to clone the repository from **Github** :

```
sudo apt-get install git-core
```

The **--recursive** flag is needed in order to make sure that the submodules **googletest**, **json**, and **Others?** are also populated during the download process.

1.2 Developer version

If you are a developer, we will suggest that you use the following longer process.

1. Create a **GitHub** account

```
https://github.com/signup/free
```

2. Install **git** on your machine

```
sudo apt-get install git-core
```

3. Setup git configuration

```
git config --global user.email "your_email@your_address.com"
git config --global user.name "your_github_username"
```

4. Email your **github** username to the owner of PARSim .

5. Create a clone of the ParSim repository

```
git clone https://github.com/bbanerjee/ParSim
```

6. Check branches/master

```
git branch -a
```

7. Configure **ssh** for secure access (if needed)

```
cd ~/.ssh
mkdir backup
cp * backup/
rm id_rsa*
ssh-keygen -t rsa -C "your_email@your_address.com"
sudo apt-get install xclip
cd ~/ParSim/
xclip -sel clip < ~/.ssh/id_rsa.pub
ssh -T git@github.com
ssh-add
ssh -T git@github.com
```

8. For password-less access with **ssh** , use

```
git remote set-url origin git@github.com:bbanerjee/ParSim.git
```

9. To change the default message editor, do

```
git config --global core.editor "vim"
```

If you are new to GRANULARSIM, we strongly suggest that you fork the code from GitHub and submit your changes via pull requests.

1.2.1 Getting the latest version, adding files etc.

1. Check status

```
git status
git diff origin/master
```

2. Update local repository

```
git pull
```

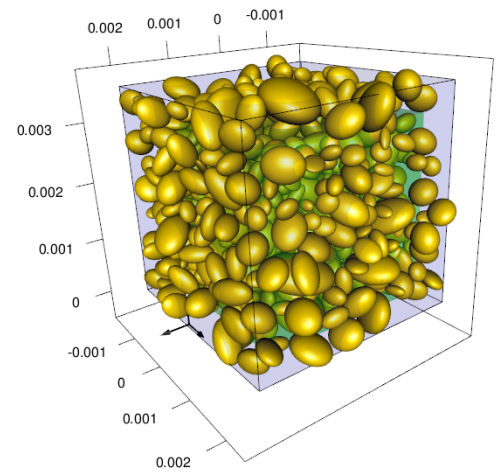
3. Add file to your local repository

```
git add README
git commit -m 'Added README file for git'
```

4. Update main repository with your changes

```
git push origin master
```

If you are new to GRANULARSIM, we strongly suggest that you fork the code from GitHub and submit your changes via pull requests.



2 — Prerequisites and library dependencies

GRANULARSIM depends on several tools and libraries that are commonly available or easily installed on various Linux or Unix like OS distributions.

The following instructions are tailored for installations on Ubuntu/Debian-like systems.

2.1 Using a package manager

If you are using a package manager to install the prerequisites, follow the instructions in this section. Otherwise, go to section [2.2](#).

2.1.1 Compilers

You will need a C++ compiler, either **g++** or **clang**:

```
sudo apt-get install g++
```

```
sudo apt-get install clang-3.8
```

You will also need a C Compiler such as **gcc**:

```
sudo apt-get install gcc
```

2.1.2 Cmake

You will probably need to install **Cmake** to control the software compilation process. To do that:

```
sudo apt-get install cmake
```

2.1.3 Boost

You will need **Boost** for parts of the code (and also some of the unit tests) to compile. GRANULARSIM uses the boost MPI and serialization libraries. It's easiest to install everything from Boost.

```
sudo apt-get install libboost-all-dev
```

2.1.4 MPI libraries

Install the **OpenMPI** libraries:

```
sudo apt-get install mpi-default-dev
```

2.1.5 OpenMP libraries

For threaded simulations you will need the **OpenMP** library. This library is included with both **gcc** and **clang**. All GRANULARSIM programs are compiled with **OpenMP** enabled.

2.1.6 Eigen3 library

Some parts of GRANULARSIM use the **Eigen3** library. You will have to install this library if you don't have it in your system:

```
sudo apt-get install libeigen3-dev
```

2.1.7 XML libraries

If you are reading data in VAANGO format into GRANULARSIM, you will need the **libxml2** library:

```
sudo apt-get install libxml2
sudo apt-get install libxml2-dev
```

GRANULARSIM reads and writes data in Ellip3D format. You will need **zenxml** to read that data. This header-only library is included with the source code.

2.1.8 JSON libraries

The **json** header-only library is also cloned into the repository when you use the **-recursive** flag. You don't need any further installation.

2.1.9 Zlib compression library

You will also need to install the development version of **zlib**.

```
sudo apt-get install zlib1g zlib1g-dev
```

2.1.10 GoogleTest library

The **googletest** libraries are cloned into the repository when you use the **-recursive** flag. You don't need any further installation.

2.1.11 VTK libraries

In order for output to be written in VTK format, you will also need the **VTK** libraries. Use

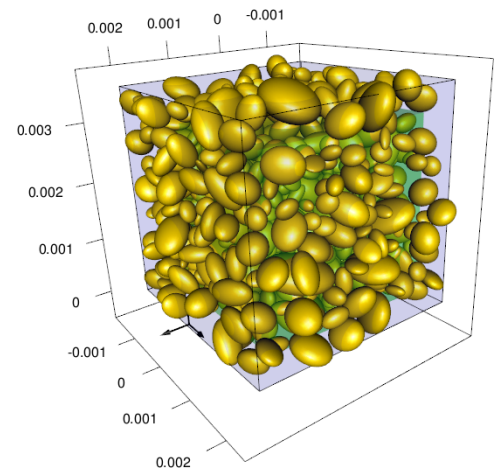
```
sudo apt-get install libvtk5-dev
sudo apt-get install python-vtk tcl-vtk libvtk-java libvtk5-qt4-dev
```

2.1.12 Qhull library

The **Qhull** source code is part of the repository. [Add instructions for Qhull.](#)

2.2 Building from source

Building on desktop is easier if you use a package manager. However, in some situations you may have to build from source. See <http://www.parresianz.com/dem/compiling-paraellip3d-for-a-cluster/> for an example.



3 — Configuring and Building GranularSim

3.1 Basic instructions

Once you are sure that all the prerequisites have been installed, you can build GRANULARSIM.

3.1.1 The GranularSim repository

After you get the code from GitHub, follow these steps:

- Go to the [GranularSim](#) directory:

```
cd ParSim/GranularSim
```

- The source code is in the directory [src](#).

3.1.2 Out-of-source optimized build

- For the optimized build, create a new directory called [opt](#) under [GranularSim](#):

```
mkdir opt
```

- followed by:

```
cd opt
```

- To create the make files do:

```
cmake ../src
```

3.1.3 Out-of-source debug build

- For the debug build, create a directory [dbg](#) under [GranularSim](#):

```
mkdir dbg  
cd dbg
```

- To create the makefiles for the debug build, use

```
cmake -DCMAKE_BUILD_TYPE=Debug ../src
```

3.1.4 Clang compiler:

If you want to use the [clang](#) compiler instead of [gcc](#):

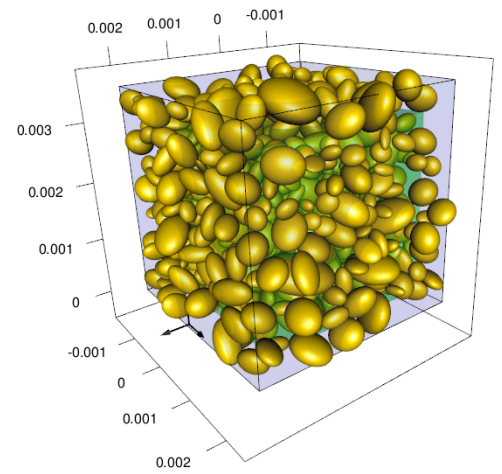
```
cmake ../src -DUSE_CLANG=1
```

3.2 Compiling the code:

Next you need to compile the files from `src`. So go to the `opt` directory and then type :

```
make -j4
```

After this compilation you have all the executable files in your `opt` directory. The same process can be used for the debug build.



4 — Installing VisIt

Visualization of GRANULARSIM data is currently possible using VisIt. The VisIt package from LLNL is general purpose visualization software that offers all of the usual capabilities for rendering scientific data. It is still developed and maintained by LLNL staff, and its interface to GRANULARSIM data is supported by the VTK team.

4.1 Directory structure

The following directory structure is suggested for building VisIt:

```
/myPath/VisIt
/myPath/VisIt/thirdparty
/myPath/VisIt/2.10.2
```

4.2 Using the VisIt install script

Download the build script from

http://portal.nersc.gov/project/visit/releases/2.10.2/visit-install2_10_2. Follow the instructions at http://portal.nersc.gov/project/visit/releases/2.10.2/INSTALL_NOTES. GRANULARSIM writes its output in the VTK file format. If VisIt does not recognize the VTK file format, you may have to follow the detailed instructions below.

4.3 Using the VisIt build script

1. Get the `build_visit` script from http://portal.nersc.gov/project/visit/releases/2.10.2/build_visit2_10_2.
2. Set the MPI compiler to be used (must be the same version used to build VTK):

```
export PAR_COMPILER='which mpic++'
export PAR_COMPILER_CXX='which mpicxx'
export PAR_INCLUDE="-I/usr/lib/openmpi/include/"
```

3. Use the `build_visit` script to download, build, and install the thirdparty libraries required by VisIt.
4. Go to the VisIt thirdparty directory:

```
cd /myPath/VisIt/thirdparty
```

5. Run `build_visit`:

```
/myPath/VisIt/2.10.2/build_visit --console --thirdparty-path /myPath/VisIt/
thirdparty --no-visit --parallel --fortran --uintah
```

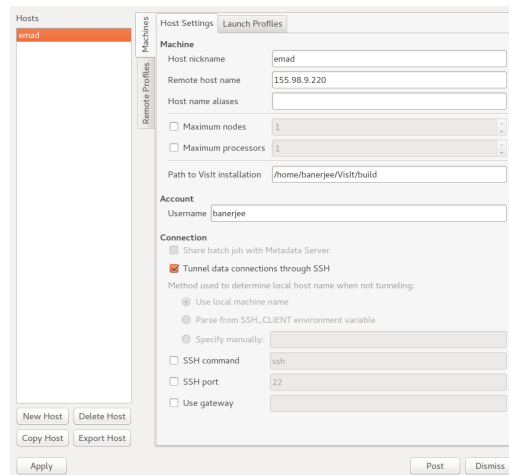


Figure 4.1: Setting up Host Profile

To add additional libraries to enable other database readers using the `build_visit` script you will have to add the appropriate flags when you run `build_visit`.

6. The `build_visit` script will take a while to finish. After the script finishes it will produce a machine specific `cmake` file called `<your_machine_name>.cmake`. This file will be used when compiling visit.
7. Move the `<your_machine_name>.cmake` file to the VisIt source `config-site` directory:

```
mv <your_machine_name>.cmake /myPath/VisIt/2.10.2/src/config-site
```

8. Go to the VisIt source directory:

```
cd /myPath/VisIt/2.10.2/src
```

9. Run `cmake` using the `cmake` that was built by the `build_visit` script

```
/myPath/VisIt/thirdparty/cmake/<version>/<OS-gcc_version>bin/cmake .
```

10. Finally make VisIt:

```
make -j 8
```

One can install VisIt or run it directly from the bin directory.

4.3.1 Remote visualization

In order to read data on a remote machine, you will need to have a version of VisIt and the VTK data format reader plugin installed on that machine. You will also need to make sure that the VisIt executable is in your path. You may need to edit your `.<shell>rc` file so that the path to VisIt is included in your shell's `PATH` variable.

The version of VisIt installed on your local desktop and the remote machine should be the same.

4.3.2 Host Profile

Next you will want to set up a Host Profile for your remote machine. Select `Host Profiles` from the `Options` menu and set up a new profile as shown in figure 4.1.

After filling in the remote machine information, check the `Tunnel data connections through SSH` option. Click on `Apply` and then do a `Save Settings` in the `Options` menu.

For the remote visualization option to work, you must have ports 5600 - 5609 open. You can try this by running the following on your local desktop (on Linux distributions),

```
traceroute -p 560[0-9] <remote machine>
```

If the tunneling option does not work, try the [Use local machine name](#) option.

Now you should be able to select [Open](#) from the VisIt [File](#) menu. After selecting your host from the host entry list, you will be prompted for a password on the remote machine (unless you have set up password-less ssh access).

Once the ssh login has completed, you should see the directory listing. You can then change directories to your data directory and load the data.