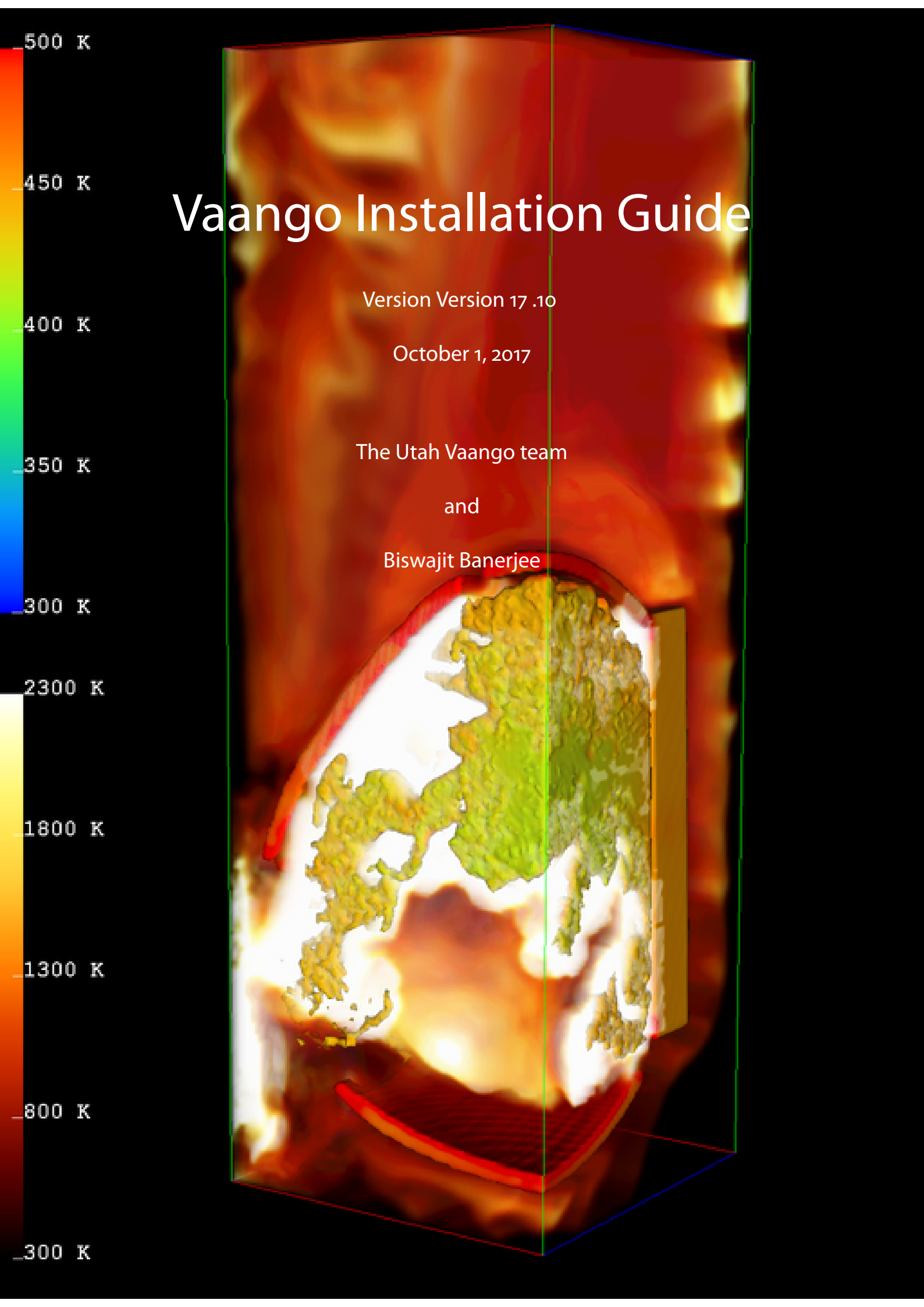


Vaango Installation Guide

Version Version 17 .10

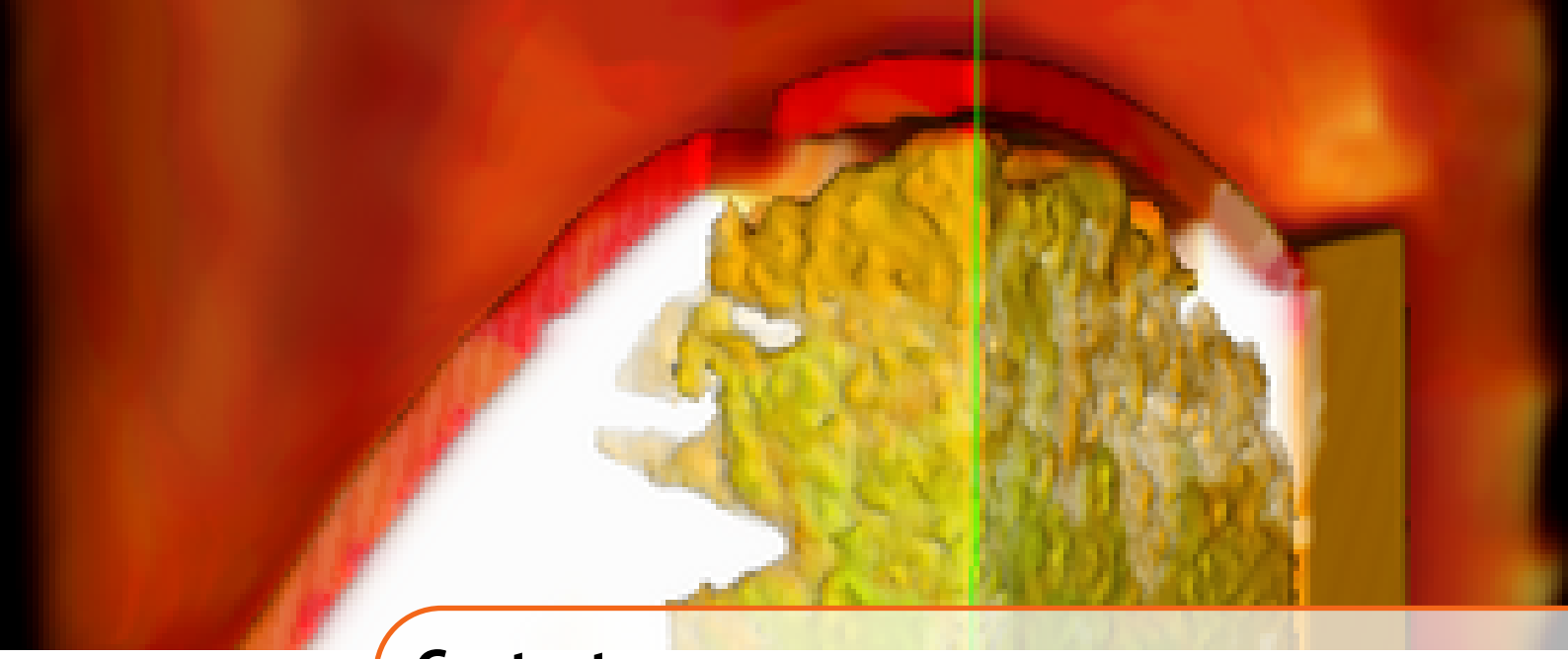
October 1, 2017

The Utah Vaango team

and

Biswajit Banerjee

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 Vaango** 5

1.1 Obtaining the Code 5

1.2 Developer version 6

1.2.1 Getting the latest version, adding files etc. 6

1.3 Installation Overview 7

1.3.1 Explicit time integration only 7

1.3.2 Implicit and explicit time integration 7

2 **Prerequisites and library dependencies** 9

2.1 Prerequsites for explicit time integration only 9

2.1.1 Cmake 9

2.1.2 Compilers 9

2.1.3 MPI and XML libraries 9

2.1.4 Other libraries 10

2.1.5 Optional libraries 10

2.2 Extra prerequisites for implicit time integration 10

2.2.1 PETSc Installation 10

2.2.2 Hypre Installation 10

3 **Configuring and Building Vaango** 13

3.1 Basic instructions 13

3.1.1 The Vaango repository 13

3.1.2 Out-of-source optimized build 13

3.1.3 Out-of-source debug build 13

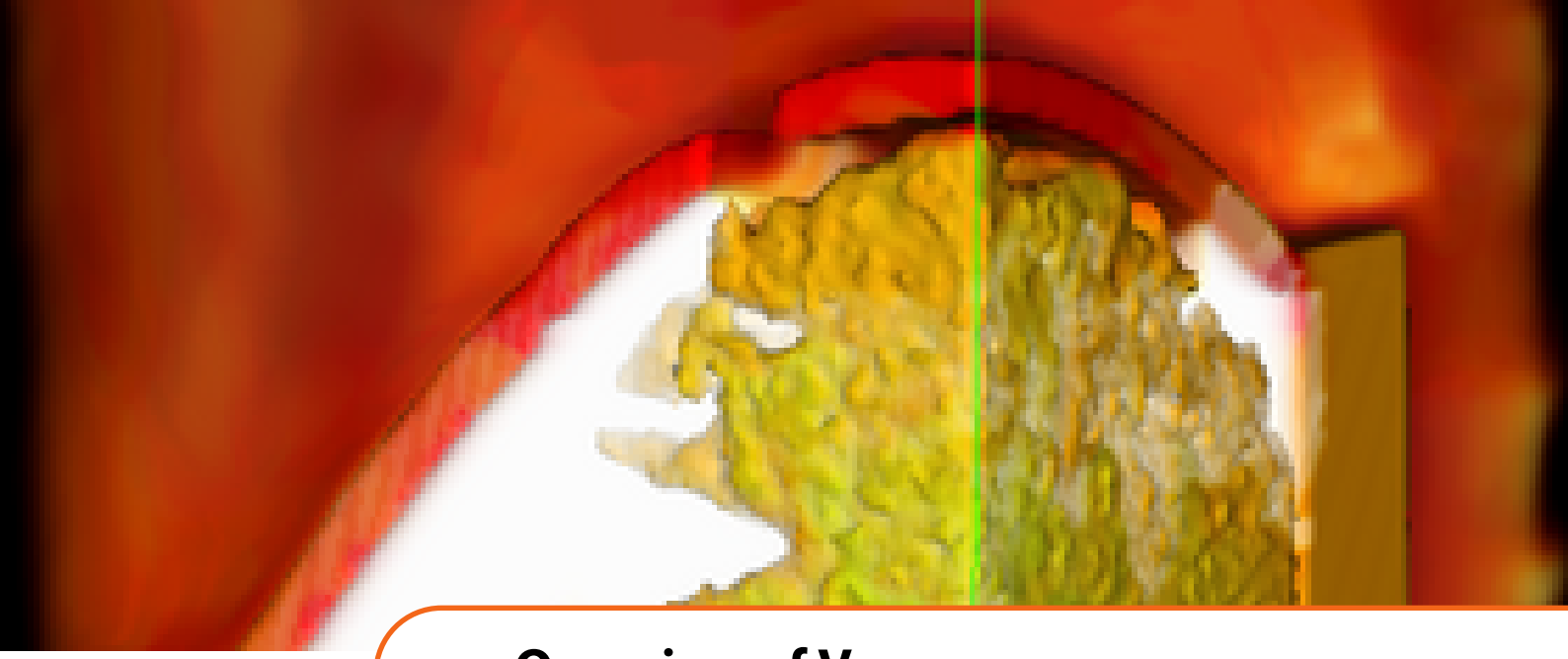
3.1.4 Unit tests 13

3.1.5 Clang compiler: 14

3.1.6 Visit build: 14

3.2 Compiling the code: 14

4 Installing VisIt 15
4.1 Directory structure 15
4.2 Using the VisIt install script 15
4.3 Using the VisIt build script 15
4.3.1 Remote visualization 16
4.3.2 Host Profile 16



1 — Overview of Vaango

The Vaango library is a fork of Uintah (http://uintah.utah.edu) intended for solid mechanics and coupled solid-fluid simulations.

The main components of Vaango are a material point method (MPM) code, a compressible fluid dynamics code (ICE), a coupled MPM-ICE code, and an experimental Peridynamics code. Efforts are underway to incorporate a Discrete Element Method (DEM) code and a Smoothed Particle Hydrodynamics (SPH) code into the framework.

The library framework in Uintah was designed for solving PDEs on structured AMR grids on large parallel supercomputers. In Vaango , we mainly use the grid for searches and scratch-pad calculations.

VisIt (https://wci.llnl.gov/simulation/computer-codes/visit/) is used to visualize data generated from Vaango .

1.1 Obtaining the Code

Vaango 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 Vaango code can be found in the ParSim/Vaango/src directory. Manuals for Vaango are in ParSim/Vaango/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 and json 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 Vaango, 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 Vaango, we strongly suggest that you fork the code from GitHub and submit your changes via pull requests.

1.3 Installation Overview

VAANGO can be installed individually and in conjunction with the visualization tool, VISIT . It is recommended for first time users to install VAANGO first and then install VISIT second after you have a working build of VAANGO .

> VISIT recognizes files in the UINTAH format automatically. VAANGO uses an older version of the UINTAH data file format that may not be recognized by the latest version of VISIT . We have found that VISIT 10.2 works with the VAANGO code.

For older versions of VISIT , the cmake command used to build VAANGO may have to be modified to include the location of the VISIT installation directory.

1.3.1 Explicit time integration only

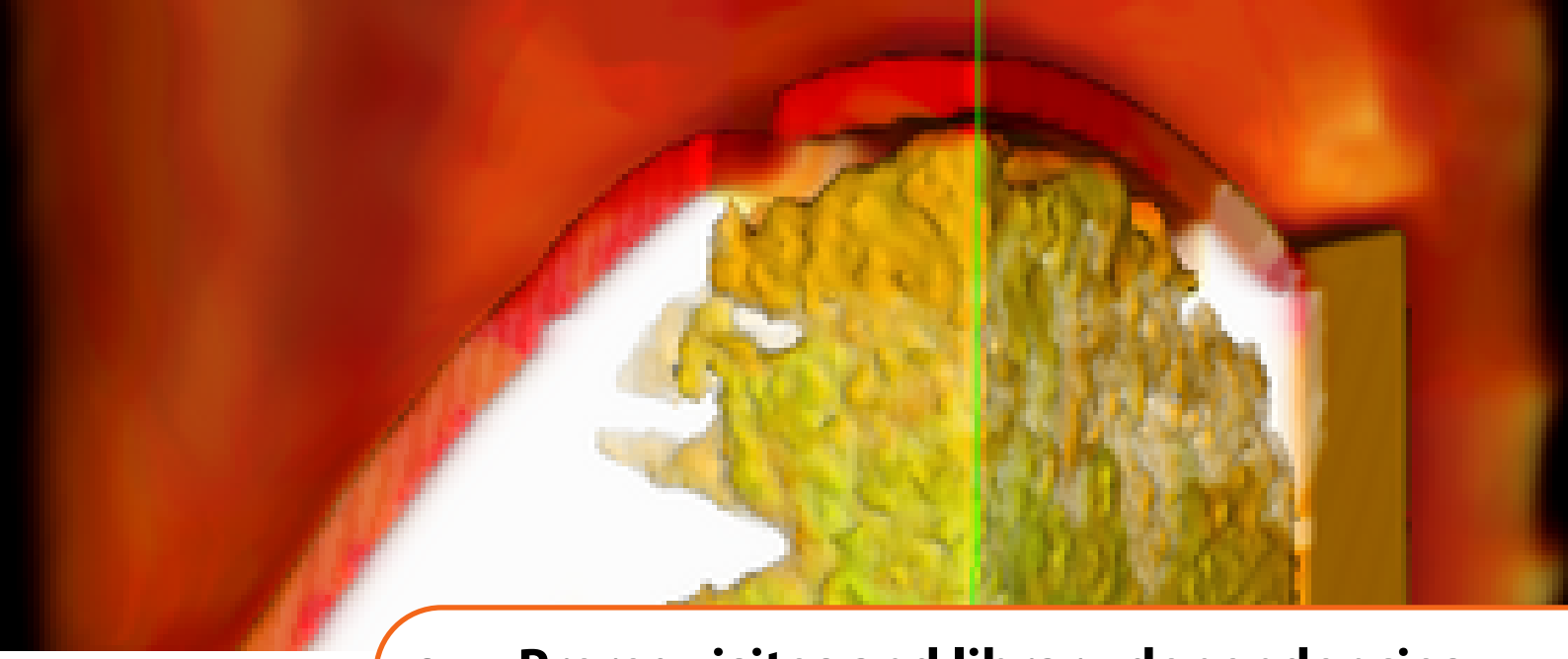
Most of the material models in VAANGO support only explicit time integration. You you want to use only these models, the installation process is relatively straightforward:

1. Install basic dependencies: MPI, etc..
2. Configure and compile VAANGO
3. Install visualization tools

1.3.2 Implicit and explicit time integration

A few VAANGO models need implicit time integration. You you need to use these models, the installation procedure follows the basic outline:

1. Install basic dependencies: MPI, Blas, Lapack, Hypre, PETSc, etc.
2. Configure and compile Vaango
3. Install visualization tools (optional if installing on a supercomputer or first time users) and reconfigure Vaango.



2 — Prerequisites and library dependencies

VAANGO 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 Prerequsites for explicit time integration only

2.1.1 Cmake

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

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

2.1.2 Compilers

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

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

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

- and a C Compiler such as gcc :

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

- You also need to have gfortran :

```
sudo apt-get install gfortran
```

- You may need Boost for parts of the code (and also some of the unit tests) to compile.

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

> We are trying to remove Boost dependencies from the code. Try to build the code without Boost , and only install the library if the build fails.

2.1.3 MPI and XML libraries

- Install the OpenMPI libraries:

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

- and libxml2 :

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

2.1.4 Other libraries

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

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

- The Peridynamics code uses parts of 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
```

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

2.1.5 Optional libraries

- In order for related code like MPM3D_xx etc. to work, 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
```

> This dependency is not needed for most installations.

- In some older versions of VAANGO , libpcl is used read point data file. In order to get PCL(The Point Cloud Library) you need to run these three commands:

```
sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
sudo apt-get update
sudo apt-get install libpcl-all
```

> This dependency is not needed for most installations.

2.2 Extra prerequisites for implicit time integration

Required libraries:

- blas
- lapack
- Hypre v2.8.0b (https://computation.llnl.gov/casc/hypre/software.html)
- PETSc v3.3 (http://www.mcs.anl.gov/petsc/petsc-as/download/index.html)

Note, we suggest using Hypre v2.8.0b and PETSc v3.3-p3.

2.2.1 PETSc Installation

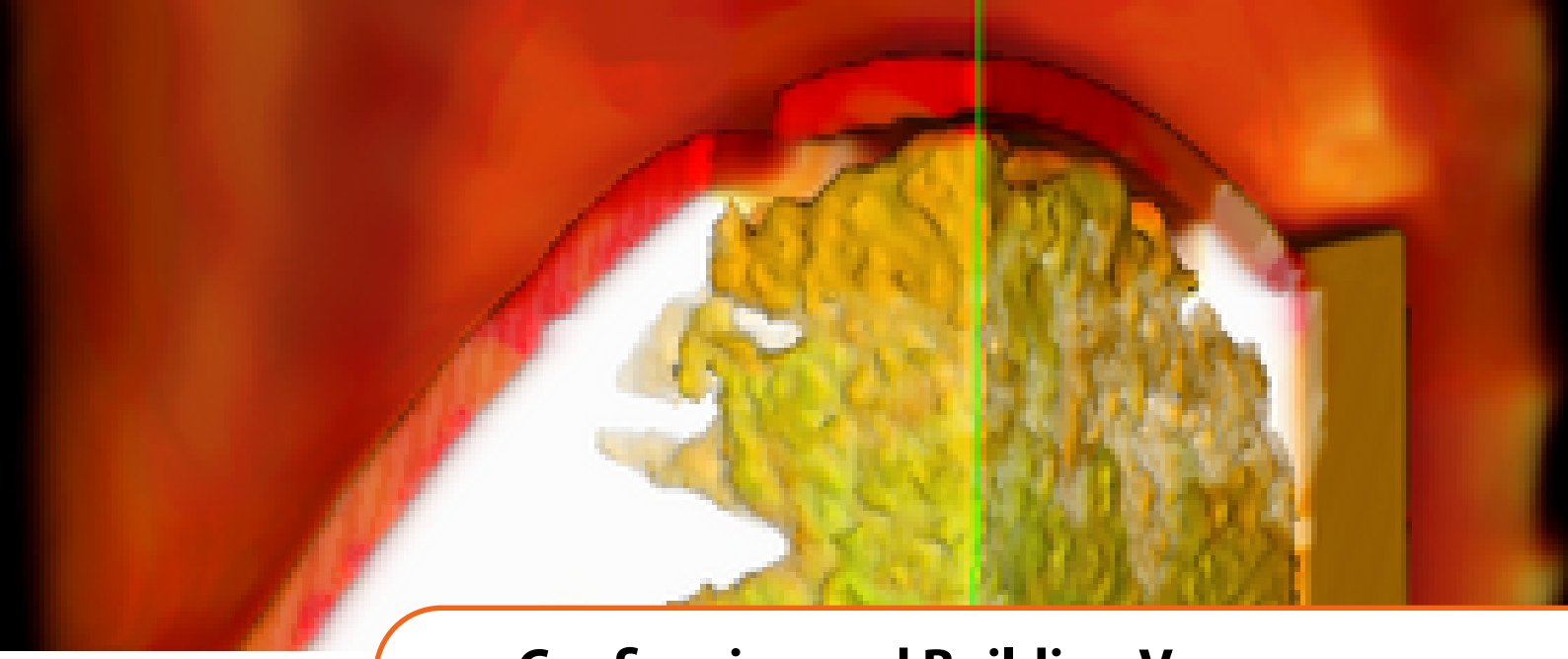
PETSc can be installed by executing the following simplified instructions. Please refer to the PETSc website for comprehensive installation instructions if you encounter any difficulties.

Download petsc-v3.3-p3 from http://www.mcs.anl.gov/petsc/petsc-as/download/index.html

2.2.2 Hypre Installation

Hypre can be installed by executing the following simplified instructions, however, if you encounter problems please refer to the hypre website for troubleshooting.

Download hypre-2.8.0b from https://computation.llnl.gov/casc/hypre/software.html



3 — Configuring and Building Vaango

3.1 Basic instructions

3.1.1 The Vaango repository

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

- Go to the Vaango directory:

```
cd ParSim/Vaango
```

- 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 Vaango :

```
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 Vaango :

```
mkdir dbg
cd dbg
```

- To create the makefiles for the debug build, use

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

3.1.4 Unit tests

If you want the units tests to be compiled, use the alternative command

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

3.1.5 Clang compiler:

If you want to used the clang compiler instead of gcc :

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

3.1.6 Visit build:

Older versions of Visit required the following extra step if you want to make sure that Visit is able to read Uintah output format files (also called UDA files) you will need to use

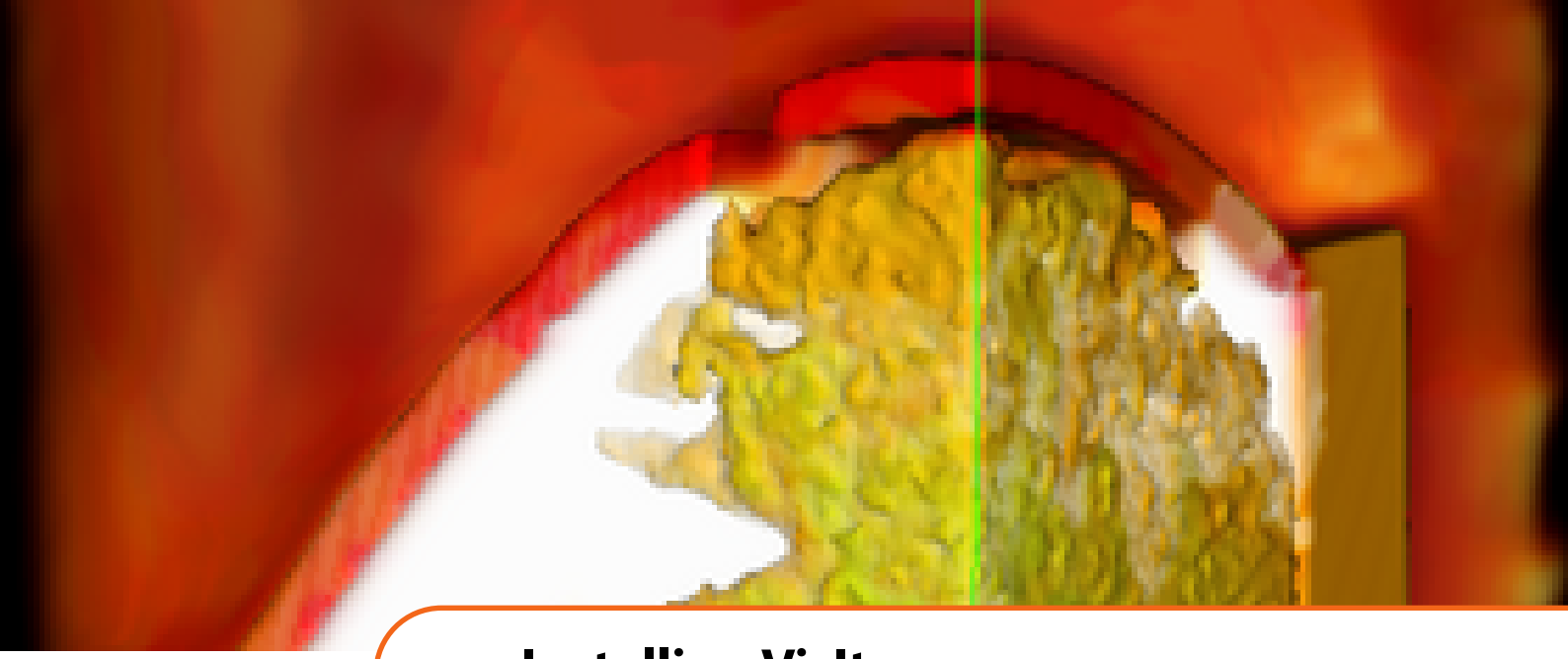
```
cmake ../src -DVISIT_DIR=/path/to/Visit
```

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 Vaango 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 Vaango data is supported by the Uintah 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. Vaango writes its output in the Uintah file format. If VisIt does not recognize the Uintah 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 Uintah):

   ```
   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 the the VisIt thirdparty directory:

   ```
   cd /mypath/VisIt/thirdparty
   ```

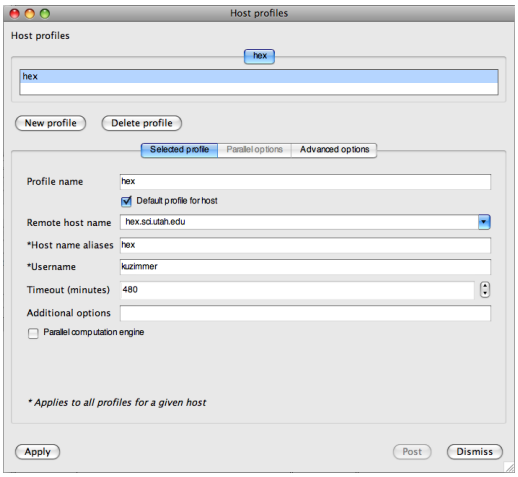


Figure 4.1: Setting up Host Profile

5. Run build_visit :

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

> 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

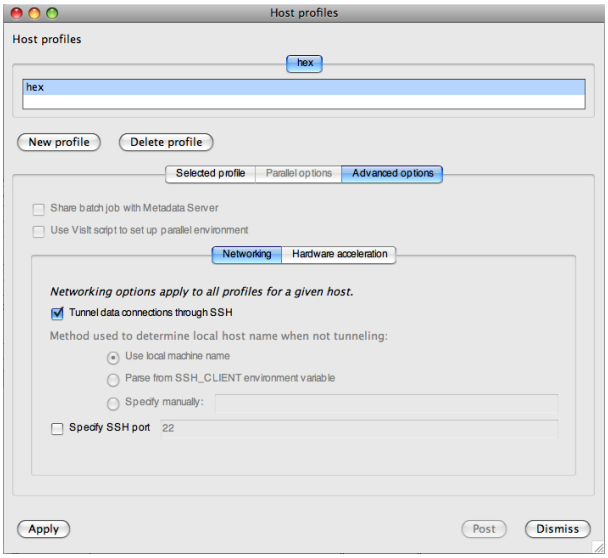
In order to read data on a remote machine, you will need to have a version of VisIt and the Uintah data format reader plugin installed on that machine. You will also need to make sure that the VisIt executable is in your path. You may need to edit your .<shell>rc file so that the path to VisIt is included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

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

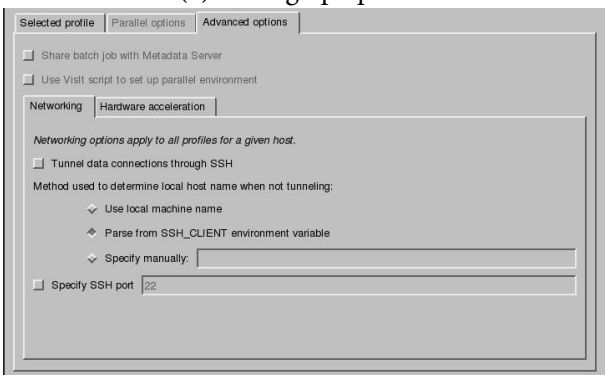
4.3.2 Host Profile

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, select the Advanced options tab, then Networking and check the Tunnel data connections through SSH option. This is illustrated in figure 4.2(a). Click on Apply and then do a Save Settings in the Options menu.



(a) Setting up options



(b) Setting up advanced options

Figure 4.2: Visit remote visualization options

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 doesn't works, try the option as shown in figure 4.2(b).

Now you should be able 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 UDA and load the data.