CB KIM Setup Documentation

Brian Muldoon [*]

October 9, 2020

Contents

[*]Mechanical Engineering Ph.D. Student at the University of California, Berkeley

1 CB-KIM Setup

1.1 SSH Key and Git Clone

The cb-kim package can be obtained from a GIT repo maintained at UMN. To properly clone the repo, it is necessary to set up a personal SSH key. To do this, see the following link to generate your own SSH key:

```
https://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key
```

After creating a unique ssh key, if one does not already exist for your computer, locate the file

```
~/.ssh/id_rsa.pub
```

and share it with Professor Ellad Tadmor at UMN. This will add your device with unique SSH key to the access list for distributions of CB-KIM from GitHub. After being placed on the CB-KIM access list, you can clone CB-KIM from Git Hub using the following command in a directory of your choice.

```
>> git clone git-tadmor@git.enet.umn.edu:cb-kim
```

Now, you should have a cb-kim folder saved to the location in which the git clone command was run.

1.2 KIM-API Download

Installation instructions for installing the KIM API package can be found at the following destination:

```
https://openkim.org/doc/usage/obtaining-models/
```

I am using a Mac, so the necessary installation command would be

```
>> brew install openkim-models
```

1.3 KIM Make

Next, we will make CB-KIM using the linux Makefiles commands. Navigate to the src directory of cb-kim in your terminal and then type 'make' as follows:

```
  >> cd cb-kim/src/

  >> make
```

This will build the open kim API for your device.

If you receive the error

```
"Package libkim-api was not found in the pkg-config search path.
Perhaps you should add the directory containing ‘libkim-api.pc’
to the PKG\_CONFIG\_PATH environment variable
No package ’libkim-api’ found
Makefile:3: *** pkg-config libkim-api does not work.
Something is wrong with your KIM API package setup.  Stop."
```

then you will need to do 2 things. The first thing we need to is set the PKG_CONFIG_PATH environment variable. This can be done within your .bashrc or .bash_profile file within your terminal home directory. In your terminal, take the following actions.

```
>> vi ~.bashrc
>> (within .bashrc file) export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

The second action that needs to be taken is to point to the proper location of the kim-api within your /usr/local/lib directory such that the limkim-api is found in the pkg-config search path. This can be performed in your terminal as follows.

```
>> cd /usr/local/lib
>> ln -s /usr/local/opt/kim-api/lib/kim-api/
```

This will set up a link in your local/lib folder pointing to the kim-api location. After completing these two actions try to make cb-kim again using,

```
>> cd cb-kim/src/

>> make
```

2 FEAP Build

Initially installing FEAP on Mac or Linux can be easily done by following the installation video created by Professor Sanjay Govindjee. The installation video can be found at the following link.

```
https://www.youtube.com/watch?v=_ohQ__rqq3Y
```

Additional FEAP resources for installation or example FEAP input files can be found at

```
http://projects.ce.berkeley.edu/feap/
```

When I was installing onto my personal Mac, I primarily referred to Professor Govindjee's video about installing FEAP from 2015. However, I encountered a few issues between initial installation and successfully running a sample input file because I had never run FEAP on Mac. Therefore, a few of the lessons learned along the way are the following:

1. Check the location of xQuartz contents in folder X11. Do this by checking for the X11 directory located at

   ```
   >> ls ~/opt/X11
   ```

   If no X11 folder exists, then you will need to download XQuartz for the FEAP GUI.

2. Within makefile.in, the FINCLUDE and CINCLUDE variables need to reflect the /opt/X11 location properly.

```
FINCLUDE = $(FEAPHOME8_6)/include -I$(FEAPHOME8_6)/modules -I/usr/local/include

CINCLUDE = /opt/X11/include
```

3. If you haven't installed gcc or gfortran before, use the following commands to obtain these packages.

```
>> brew install gcc

>> brew cask install gofortran
```

4. Change, the FF and CC variables within the makefile.in to reflect the name of your gfortran and gcc versions. Therefore,

```
FF = /usr/local/bin/gfortran-10

CC = /usr/local/bin/gcc-10
```

5. After completing all of the this, try the 'make feap' command again in your terminal from the 'Applications/FEAP/ver86' or where your FEAP makefile is located.

Finally, run a test input from the FEAP website premade input files to check that FEAP gives output and operates normally.

3 Testing CB-KIM

Testing CB-KIM functionality can be done by performing the following commands in the terminal.

```
>> cd /cb-kim/testsuite

>> ./runtest.sh
```

This will run a number of tests on my computer documenting which pass or fail. If a model claims to have failed, check the output files in the /cb-kim/testsuite/feap directory. Check the numbers for stress and elastic stiffness matrix between the expected output and true output to determine if the discrepancy in outputs in due to more than numerical precision of the computer.

4 Building FEAP with CB-KIM

Copy the following files from Professor Govindjee into FEAP/ver86/main.

```
umati9.f

umatl9.f

umesh9.f

Isi
```

Next, go to feap/main/makefile and make the adjustments as outlined below.

```
include $(FEAPHOME8_6)/makefile.in

CBKIMHOME = ($your cb-kim location$)/cb-kim

OBJECTS = feap86.o umati9.o umatl9.o umesh9.o

MOBJECTS = $(CBKIMHOME)/src/mod_feapkim.o

feap: $(MOBJECTS) $(OBJECTS) $(ARFEAP)

        ranlib $(ARFEAP)

        $(FF) -o feap $(MOBJECTS) $(OBJECTS) \

        $(ARFEAP) -L$(CBKIMHOME)/bin -lcbkim -lkim-api $(LDOPTIONS)

        @@echo "--> FEAP executable made <--"

       dsymutil feap
```

Additionally, we need to adjust the makefile.in such that the .o files are handled properly. Therefore, it is necessary to make the following changes.

```
\%.o: \%.F

          $(FF) -c $(FFOPTFLAG) -I$(FINCLUDE) $< -o $@

\%.o: \%.f

          $(FF) -c $(FFOPTFLAG) -I$(CBKIMHOME)/src -I$(FINCLUDE) $< -o $@

\%.o: \%.F90

          $(FF) -c $(FFOPTFLAG) -I$(CBKIMHOME)/src -I$(FINCLUDE) $< -o $@

\%.o: \%.f90

          $(FF) -c $(FFOPTFLAG) -I$(FINCLUDE) $< -o $@

\%.o: \%.c

          $(CC) -c $(CCOPTFLAG) -I$(CINCLUDE) $< -o $@
```

After completing the makefile-in edits, type "make" in the terminal withing the ver86/main directory to build FEAP with cb-kim.

4.1 Testing CB-KIM

Within FEAP/ver86/main run FEAP using the input file "Isi" after following the feap prompts in the terminal. FEAP should then begin running and the GUI shown below should actively display the simulation plots for each time step.

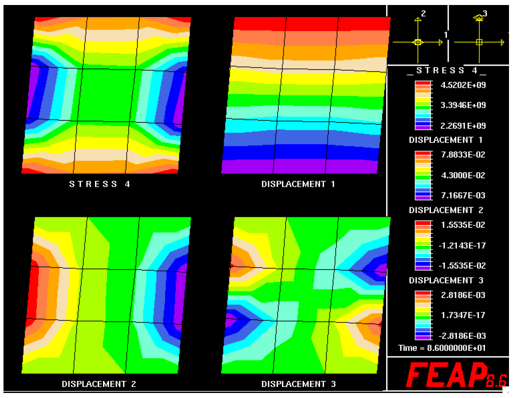

Figure 1: Sample FEAP output for Stillinger Weber Si sample shear simulation