FEAP - - A Finite Element Analysis Program

Version 8.6 Installation Manual

Robert L. Taylor & Sanjay Govindjee
Department of Civil and Environmental Engineering
University of California at Berkeley
Berkeley, California 94720-1710

June 2020

Contents

Chapter 1

Introduction

The source files for the *FEAP* system are delivered electronically from the web site. The program is furnished under license by the Department of Civil and Environmental Engineering at the University of California, Berkeley. It is for use by the licensee only and may not be redistributed to others in any form without prior authorization by the University of California, Berkeley.

It is recommended that a directory with the name 'feap', or similar be created and all information that is downloaded be copied into this directory. The source files will reside in a directory with the name 'ver86'. When the files are copied to the 'feap' directory the program structure will have the directory structure shown in Table 1.1.

Within the licensed unit it is permitted to make public versions of the following:

1. An executable version and/or an archive (library) file(s);
2. The files 'feap86.f' (main program); 'contact.f' (dummy file to eliminate contact module); 'pplotf.f' (dummy file to eliminate graphics).
3. include files; and
4. User files in the directory 'user' (e.g.,'elmt01.f', 'umacr1.f', 'umesh1.f', 'usetm1.f', etc.).

All other files are considered to be the property of the licensee and should not be made available to others.

The routine 'feap86.f' may be modified to set parameters as necessary. The current *FEAP* system uses dynamic memory allocation for all the main arrays during solution. As such the maximum size of problems that can be solved by the program is limited only by the available memory of the computer used.

```
------------------------------------------------------------
   feap
     ver86
       contact ------------+- main
       elements ---+       +- ntrnd
       main        |       +- nts2d
       maintain    |       +- nts3d
       plot        |       +- ptpnd
       program     |       +- tie2d
       unix        |       +- util
       user        +------------------- elements
       windows ----+- memory              +- acousticnd
       packages ---+- arpack -+- archive  +- frame
       unix        +- blas                +- material
       user        +- lapack              |    +- small
       window1     +- meshmod             |    +- finite
       window2                            +- shells
       include                            +- solid1d
                                          +- solid2d
       parfeap ---+- partition            +- solid3d
                  +- unix                 +- thermal
                  +- windows              +- torsion
                  +- packages -+          +- couple3d
                               +- arpack  +- robin
       fe2                                +- winkler

       iga --------------------+- contact
                               +- elements------+- cable
       fluids                  +- include       +- dummy
                               +- interpolation +- frame
       vem--+- vem_source      +- piegl         +- shell
            +- main            +- plot          +- shell_1d
                               +- program       +- slope
                               +- quadrature    +- solids
                               +- refinement
                               +- user
                               +- main
------------------------------------------------------------
```

Table 1.1: Directory structure for *FEAP* system

Please report any installation problems by e-mail to: *feap@berkeley.edu*.

1.1 Compilers

The code has been checked with Intel compilers (Linux and Windows) and works with version 14.0.3 or newer. If using the GNU Compiler Compilation (gcc and gfortran), one must use gfortran version 5 or newer. Note these are often not loaded by default and must be separately installed, for example on Ubuntu use:

```
add-apt-repository ppa:ubuntu-toolchain-r/test; apt update;
apt install gcc-X gfortran-X g++-X
```

On the `apt` line above, use the highest available version for `X > 5`.

Chapter 2

Installations

2.1 UNIX/Linux/Mac Computer Installations

The build in a UNIX/Linux/Mac environment is controlled by the data contained in files `makefile` and `makefile.in` located in the directory "ver86". Example makefiles configured for the use of gfortran and ifort are provided in the files `makefile.in_gfortran` and `makefile.in_intel`.

2.1.1 Editing files `makefile` and `makefile.in`

Use a text editor to make changes to the files `makefile` and `makefile.in` located in the directory `ver86`. The location of specific parts of the *FEAP* program is controlled by the parameter `$(FEAPHOME8_6)`. This parameter may be set using the system command:

```
setenv FEAPHOME8_6 /home/..../feap/ver86
```

or for *bash*

```
export="FEAPHOME8_6 /home/..../feap/ver86"
```

where the last parameter is the path name where the individual subdirectories in `ver86` are located. Alternatively, the command may be inserted within the `.cshrc`, `.tcshrc` or whatever user filename is located in the user root directory.

Editing `makefile.in`

It is necessary to edit the file `makefile.in` as indicated below. Note that comments in this file are set by placing the character `#` in the first column.

Edit file `makefile.in` as follows:

1. Select the appropriate include files to use for the `FINCLUDE` parameter. Some systems also require a path for the C-includes. The path is assigned to the `CINCLUDE` parameter.

2. For builds where all integer declarations are to be promoted to integer*8 using a compiler directive such as "-i8" set ipr = 1 in the feap86.f file. In addition replace the routines `cmemck.c` and `cmem.c` in the directory `unix` by those in the directory `unix/largemem`.

3. In section *Which compilers to use* set the name of your Fortran compiler after `FF =` (Different options are indicated with all but one commented with the `#` symbol). Also set the name of your C compiler after `CC =` (Again, different options are indicated).

4. Set optimization level to use. Currently this is set to `O2` and the flag for all warnings also is active.

5. In Section *Source Types*:

   (a) Generally no changes are needed for source types (it is blank, i.e., `FSOURCE =` and `CSOURCE =`).

6. In the section *Source Extender*:

   (a) Generally no changes are needed for the extender (i.e., they are just the Fortran `FEXT = f` and the C `CEXT =c`).

7. Generally, no options are needed for `FOPTIONS =` or `COPTIONS =`; however, if you experience difficulties some may need to be inserted.

8. In section *What options to be used by the loader* select the correct X-library (i.e., folder location). If a non-standard installation is made some changes may be required.

9. If the `jpeg` screen dump option is desired then add `-ljpeg` after `-lX11`. Also copy the file `jpgd.c` from the `unix/jpeg` directory into the `unix` directory.

10. In section *What archiving to use* standard options are given. Usually no change is necessary.

Editing `makefile`

Generally, this file does not need any modifications unless new options are to be added.

Ubuntu and Similar Linux Systems

On Ubuntu and other similar Linux systems there is one runtime and one installation issue that arise frequently.

1. The Helvetica fonts are not pre-installed; the error will manifest itself at run time when you try to plot something. You can install the needed fonts using `apt install xfonts-75dpi xfonts-100dpi`. You may need to log out and log back in (or even restart your Xserver) for the changes to take effect. You can check that you have the fonts loaded by running `xlsfonts | grep helvetica`. If all is well, you should see a reasonably long listing of helvetica fonts. If this does not resolve the issue you can optionally try loading additional font packages for your distribution.
2. The other issue that arises has to do with the default behavior of the GNU Compiler Compilation loader and how it treats unresolved symbols in static archives (files ending in `.a`). The makefiles in `$FEAPHOME8_6/main`, `$FEAPHOME8_6/igafeap/main`, and `$FEAPHOME8_6/parfeap` contain special instructions for resolving the loader issue. The basic change is that the variables holding the names of some of the static library files need to be replaced by syntax pointing to the directory where they are contained, along with syntax that forces a complete load of the particular static library. In particular one needs to replace:
   `$(ARPACKLIB) \` on the loader lines by:

   ```
   -L$(FEAPHOME8_6)/packages/arpack \
   -Wl,-whole-archive -larpack -Wl,-no-whole-archive \
   ```

   `$(ARCHIVELIB) \` on the loader lines by:

   ```
   -L$(FEAPHOME8_6)/packages/arpack/archive \
   -Wl,-whole-archive -larchive -Wl,-no-whole-archive \
   ```

   `$(PARPACKLIB) \` on the loader lines by

   ```
   -L$(FEAPHOME8_6)/parfeap/packages/arpack \
   -Wl,-whole-archive -lparpack -Wl,-no-whole-archive \
   ```

   and `$(ARIFEAP) \` on the loader lines by:

   ```
   -L$(FEAPHOME8_6)/igafeap \
   -Wl,-whole-archive -lifeap8_6 -Wl,-no-whole-archive \
   ```

2.1.2 Installing the program

If necessary, change directories until you are in `ver86`. The installation of the program is made using the command:

```
make install
```

Each subdirectory should be processed and the compiled object files placed in the archive named in the `makefile.in`. A successful compilation should deposit the executable (named `feap`) in the subdirectory `main`.

If errors occur it is necessary to correct them and then recompile the program using the command `make install`.

2.1.3 Running *FEAP*

After a successful installation step the *FEAP* program is ready for use. To permit running the program from any directory it is convenient to define a path to the location of the executable. This may be done by placing the one of the following lines in an appropriate file in the root directory;

```
alias feap '/fullpath/ver86/main/feap'
```

or

```
alias feap="/fullpath/ver86/main/feap"
```

where `fullpath` is the complete path to the `ver86` directory.

The program may now be executed from any directory by first preparing an input file (see the *User Manual* for preparing this file) and issuing the instruction

```
feap
```

from the command line in any window. If graphics is to be enabled it may be necessary to create an `X`-window using the command:

```
startx &
```

Full testing requires the preparation of an *input file* as described in the *FEAP User Manual*. Some input files for test problems are available from the location where the source program was downloaded.

2.2 Windows Installation: Intel Fortran with Visual Studio

An executable version of *FEAP*, including all graphics options, may be built using the Intel Fortran compiler with Visual Studio.

Generally, it is desirable to place all parts of the program except the main program (`feap86.f` in the `main` directory) into a single library and then finally build a main (executable) program. For example, a build with the library named *lib86* places all basic parts of the program together. A main program called *feap* may then be constructed which includes this library. However, alternate combinations separating the library into parts may be selected.

2.2.1 Build of Library

The following steps may be used to build the necessary library for the *FEAP* program:

1. Open the Developer Studio for a new project.
2. Under *File* select *New*. (N.B. Options to be selected are shown in italics).
   (a) Under *Projects* tab select *Fortran static library*. **Do not select a dynamic link library (DLL)**.
   (b) In *location* window set path to a location for build files. The path must exist, if not use standard Windows steps to create the folder before doing this step.
   (c) In *Project name* assign a library name (e.g., lib86). (N.B. Items to be selected and named by the user are indicated by underlines).
   (d) Press *OK* button to start (N.B. small upper window should now have the notation **Workspace program**).
3. Under *Build*:
   (a) Select *Set Active Configuration* and choose between *Release* and *Debug* (generally I use Release for most builds, however, if you use *Debug* it will be necessary to set the compile option to ignore array bounds).
4. Under *Project* select ~~*Project Settings*~~:
   (a) Choose *Fortran* tab and set Category window to *Preprocessor*.

(b) In *INCLUDE and USE paths* window insert the path to where the include files are located. (The path will generally be set when you install the program - e.g., `c:\feap\ver86\include` and `c:\feap\ver86\include\integer4` – or `integer8` for 64-bit machines). (N.B. Setting both the `include` and the `include\integer4` (or `include\integer8`) paths is essential to get any compile to work properly!)

(c) Press *OK* button to finish settings.

WARNING! STEPS 3 and 4 must be set in the sequence shown above. In particular if a change between *Release* and *Debug* is made it is necessary to set the INCLUDE paths again.

5. Under *Project* select *Add to Project* which causes a pop-up window to appear. Select *Files* which will pop-up another window called *Insert files into project*. Use the *Look in* window to select the folder where source programs are located and find the *feap* folder. The select *ver86* (double click on folder button will change path), followed by *contact* and then *main*. If *Files type* window is set to *Fortran files(*.for,*.f90,...)* all the files to be selected will appear in the large window. To select all files place mouse cursor over last file in folder and while holding the "Shift" key press the left mouse button. All files should now be highlighted. Press *OK* button to have highlighted files placed in project.

   N.B. Instead of using the *Look in* window to find directories, it is possible to use the *Up one level* button to traverse the folder structure to locate where source files are located.

6. Repeat step 5 for all the source folder names in *contact* (i.e., *ptpnd*, etc.). Repeat for all subdirectories in *element*. Finally, load the files in the *plot*, *program*, *user* and *windows* directories. Finally, include the files from either the directory *window1* or the directory *window2* – but not both.[1] After all files (except `feap86.f` are loaded into the library proceed to compile the program.

7. Under *Build* tab select *Build lib86.lib* (or name you selected for this project or *Rebuild all*.

   Compiler should process each file in the project and finish with a statement: "lib86.lib - 0 error(s), 0 warning(s)".

   If errors are present changes are necessary. First thing to ensure is that the path to the INCLUDE files is properly set (see step 4. above).

At this stage the library "lib86.lib" for the *FEAP* program has been built. It is now necessary to build the final executable program.

[1] The files from *window1* create a compined text/graphics window while those from *window2* create separate graphics and text windows.

2.2.2 Build of Executable

The following steps may be used to build an executable for the *FEAP* program:

1. Under *File* select *New*.
   (a) Under *Projects* tab select *Fortran Standard Graphics* or *QuickWin Application*.
   (b) In *location* window path to location for build files should still be set for the library build. This is ok, but can be changed if you wish (recommend no change for this). The path must exist, if not use standard Windows steps to create the folder before doing this step.
   (c) In *Project name* assign a program name (e.g., feap).
   (d) Press *OK* button to start (N.B. small upper window should now have the notation *Workspace 'feap'*).
   (e) New pop-up window gives choice between a QuickWin and a Standard Graphics mode. Select *QuickWin* and then press *Finish*.
2. Repeat steps 3 and 4 above which are now applicable to this project. (e.g., must set *Release* or *Debug* mode and path for INCLUDE files).
3. Under *Projects* tab select *Settings*, followed by the *Link* tab. In *Category* window select *Input*.
4. Use *Project* tab and select *Add to Project*. Then select *Files* and select the folder *Main* (see step 5 above). Add the main program file 'feap86.f' to the project. (If you do not want to include the graphics option also add the file 'contact.f').
5. Use *Project* tab and select *Add to Project*. Select *Files* tab and go to folder where library "lib86.lib" is located. This is the path you set in the first build followed by the name of the library (e.g., "lib86") and either *release* or *debug* depending on which you built. Nothing will appear in the main window until a selection is made in the *Files of type* window is set to: *Library Files (lib)*. It may be necessary to scroll to find this or just enter "l" in the window and scrolling will occur automatically.

   Add the lib86.lib to the project by placing the mouse over the name in the window and double clicking.
6. Under *Build* tab select *Build feap.exe* (or name you selected for this project or *Rebuild all*. Compiler should process each file in the project and finish with a statement: "feap.exe - 0 error(s), 0 warning(s)". If errors are present changes are necessary. First thing to ensure is that the path to the INCLUDE files is properly set (see Step 4. in the instructions for building the libraries).

Program is ready to use. The executable will be placed in the *release* or *debug* directory where the build of the executable was designated (see Step 3 and 4 in Section 2.2.2). It is usually desirable to place an executable icon on the 'Desktop'.

2.2.3 Alternate Windows graphics forms

As indicated in Section 2.2.1 alternate window structures may be built by using the source files in either the directory `window1` or those in `window2`. This permits a build in which there is a single *feap* window (`window1`) containings areas for both graphics and text. In addition the graphics may be positionied in three different areas. Very limited text is visible during execution.

The alternative build using the files from directory `window2` creates two separate windows: one containing text and the other graphics. Only one graphics area is available, however, there is more area for text outputs. This may be useful when developing new modules for the program, or in situations where text output is more important to the user than graphics.

2.2.4 Running *FEAP*

After a successful installation step the *FEAP* program is ready for use. The program may be run in two modes:

1. From a command line in a "Command prompt" window. In this case it is convenient to place a "bat" file (e.g., feap.bat) in a directory located on the system `PATH`. This file has the structure:

   ```
   c:\fullpath\feap\ver86\build\release\feap
   ```

   where it is assumed the executable resides in the directory `build` and is a release version. The program may now be executed by giving the command

   ```
   feap
   ```

   in any directory.

2. From an icon reached by traversing the directories to the location where the executable resides after the build. For convenience the icon may be placed on the 'Desktop' and executed there. A pop-up window will appear to locate the desired input file (see User Manual for preparing this file).