
INSTALLATION GUIDE

CalculiX Solver ccx 2.9 on Mac OS X El Capitan

¹B. Graf

Abstract. CalculiX comprises the solver ccx (CalculiX CrunchiX) and the graphical pre- and postprocessor cgx (CalculiX GraphiX). Except of the html-help files and the installation path the installation of cgx and ccx is independent from each other. Accordingly, separate step-by-step installation guides for ccx and cgx on Mac OS X are available (available guides for Mac OS X Mavericks, Yosemite and El Capitan listed under the “Release Notes” of the present guide). In the present guide a basic installation of ccx from source and Mac OS X El Capitan is considered, whereas the aforementioned basic installation is defined as follows: ccx 2.9 for multithreading (MT), external solvers SPOOLES (serial and MT) and ARPACK (serial). The optional installation of the solvers POSIX and TAUCH, of the SGI Library as well as of a 8 byte version of ccx for large CFD problems, therefore, is not subject of the present basic installation. Besides the basic installation of ccx also the installation of required Mac-specific software is considered (Xcode, c- and Fortran compilers). Furthermore, tests of the functionality of the installation are provided.

Acknowledgement. Special thanks to G. Dhondt (developer of ccx, MTU Aero Engines Munich, Germany) for his support. to William L. Polhemus, Jr. P.E. (Houston, Texas) for testing the installation guide for Mac OS X Yosemite and removing typos and to Enzo Maier (Edmonton, Canada) for testing the installation guide for OS X El Capitan.

¹Prepared by B. Graf, based on the general installation instructions for Unix/Linux by G. Dhondt (developer of ccx, MTU Aeoro Engines, Munich, Germany).

Dr. B. Graf

Consulting Engineer

Zurich

graf_bernhard@bluewin.ch

RELEASE NOTES

¹ Installation Guides	
Revision	Title Changes and Remarks
Draft, June 18, 2014	Installation Guide - ccx on Mac OS X - Installation of ccx 2.7 on Mac OS X 10.9.3 Mavericks.
February 10, 2015	INSTALLATION GUIDE – Calculix Solver ccx 2.8 on Mac OS X Mavericks - Installation of ccx 2.8 on Mac OS X 10.9.5 Mavericks. - GCC 4.9 (gcc) used for compilation of ccx c-routines instead of previously used GCC 5.1 - Editorial changes.
May 30, 2015	INSTALLATION GUIDE – Calculix Solver ccx 2.8 on Mac OS X Mavericks & Yosemite - Documentation of the installation of ccx 2.8 on Mac OS X 10.10.3 Yosemite added to previous revision from February 2015. - Tab. 1 with list of tested program versions added. - Last line on page 21: <code>CC=/usr/local/bin/g++</code> changed to: <code>CC=/usr/local/bin/gcc</code> because compilation with g++ fails. - Typos removed and editorial changes.
November 30, 2015	INSTALLATION GUIDE – Calculix Solver ccx 2.9 on Mac OS X El Capitan - Previous revision from May 30, 2015 modified for the installation of ccx 2.9 on Mac OS X 10.11.1 El Capitan. - Typos removed and editorial changes.

¹Last revision available on Calculix site, previous revisions available on request from author.

CONTENTS

	RELEASE NOTES	i
1.	INTRODUCTION.....	1
1.1	General.....	1
1.2	Basic Installation of ccx and Optional Additional Software	2
2.	SOFTWARE REQUIREMENTS.....	3
2.1	Table of Tested Program Versions.....	3
2.2	Mac OS X Specific	3
2.2.1	OS X El Capitan – Terminal App	3
2.2.2	Xcode, CLTs (Command-Line Tools) and Modified GCC (GNU Compiler Collection).....	3
2.2.3	Unmodified GCC (GNU Compiler Collection) for OS X.....	4
2.2.4	Fortran Compiler gfortran for OS X.....	4
2.3	ccx (CalculiX ChrunchiX).....	5
2.4	External Solvers SPOOLES and ARPACK.....	5
3.	PRE-ARRANGEMENTS FOR INSTALLATION.....	6
3.1	Compilers – modified GCC, unmodified GCC and gfortran	6
3.2	Terminal, Terminal Commands, HOME, LOGNAME, Login and Permissions	6
3.3	Editors, .bash_profile, Editing of Hidden File .bash_profile and Usage of .bash_profile to Set Paths and Export Paths	7

CalculiX • ccx Installation Guide Mac OS X •

4.	INSTALLATION path and installation OF CCX FILES.....	10
4.1	General.....	10
4.2	Installation Path “calculix_inst_path“ and File System	10
4.3	Install ccx Files	11
5.	INSTALL MAC-SPECIFIC SOFTWARE	13
5.1	Xcode and Associated Modified GCC.....	13
5.2	Unmodified GCC.....	13
5.3	Fortran Compiler gfortran.....	14
6.	INSTALL EXTERNAL SOLVERS.....	16
6.1	General.....	16
6.2	SPOOLES	16
6.3	ARPACK.....	18
7.	INSTALL AND LAUNCH CCX, TEST INSTALLATION, GETTING STARTED AND TEST EXAMPLES	21
7.1	Install ccx.....	21
7.2	Launch ccx and Test Installation.....	23
7.3	Getting Started With ccx.....	23
7.4	Test Examples.....	23
8.	TESTS - TEST EXAMPLES, NUMERICAL ACCURACY AND MULTI- THREADING.....	24
8.1	Test Examples and Numerical Accuracy	24
8.2	Invoke and Test Multi-Threading (MT) for Spooles	25
9.	HINTS AND MAC-SPECIFIC PROBLEMS	29
9.1	Hints.....	29
9.2	Mac-Specific Problems.....	29
	REFERENCES.....	30 - 30

1. INTRODUCTION

1.1 General

CalculiX comprises the solver `ccx` (CalculiX CrunchiX) and the graphical pre- and post-processor `cgx` (CalculiX GraphiX). Except of the `html-help` files and the installation path the installation of `ccx` is independent from the installation of `cgx`. Accordingly, the sequence of the installation of `ccx` and `cgx` may be chosen freely.

The present guide covers the installation of `ccx` on Mac OS X El Capitan (cf. [4] for the installation of `cgx`). Available installation guides for previous releases of OS X and `ccx` are listed under the “Release Notes” on page i.

For a brief introduction to the installation of `ccx` on Unix and Linux systems, respectively, please read the instructions by G. Dhondt [1], Section A. Based on the aforementioned reference [1] in the present document a guide for the step-by-step installation of `ccx` on Mac OS X El Capitan is provided. Installation from source is considered. Furthermore, the installation of `ccx` requires the pre-installation of Mac-specific software like Xcode and compilers GCC (GNU compiler collection) and `gfortran`.

The present guide covers the installation of Mac-specific software as well as the so called basic installation of `ccx` which according to Section 1.2 is defined as follows: installation of `ccx` and optional external solvers SPOOLES and ARPACK. Furthermore, in Section 1.2 also the installation of ²additional (in addition to the basic installation) optional software is briefly addressed. The installation of the aforementioned additional software, therefore, is not subject of the present installation instructions.

Section 2 contains an overview of the required software. In particular, a list of the tested program versions is given in Section 2.1 (cf. also Tab. 1). The subsequent Section 3 contains a summary of pre-arrangements and tools, which are required for the installation of `ccx`, followed by the installation instructions in Section 4 to 7. The installation should follow the sequence as given by Sections 4 to 7. First of all, in Section 4 the installation path and the file system are explained as well as the installation of the source code and `html-help` files of `ccx`. In Section 5 Mac-specific software is installed (Xcode, compilers GCC and `gfortran`). The external solvers SPOOLES and ARPACK are installed in the subsequent Section 6. Finally, the installation of `ccx` is completed with the build of `ccx` in Section 7. Furthermore, Section 7 covers the following subjects: launch of `ccx` and test of installation, how to get started with `ccx` and test examples. In Section 8 the functionality of `ccx` is tested, in particular the multi-threading capability of `ccx` as well as the functionality of the external solvers SPOOLES and ARPACK. The final Section 9 is provided for the future documentation of hints and known Mac-specific problems. The references are provided on page 30 and the release notes on page i.

²Installation of the additional external solvers POSIX and TAUCH, of the SGI Library as well as of a 8 byte version of `ccx` for large CFD problems.

1.2 Basic Installation of ccx and Optional Additional Software

- *Basic installation of ccx.* The so-called basic installation of ccx is subject of the present instructions and comprises the following installations: ccx 2.9 for multi-threading (MT), external solvers SPOOLES (serial and MT version) and ARPACK (serial).

Note. As further outlined in Section 5 below, the installation of Arpack in addition requires the installation of BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage).

- *Optional additional software.* In references [1], [2] the installation as well as the capabilities of the following additional software are explained: external solvers POSIX and TAUCH, SGI Library, 8 byte version of ccx for large CFD problems. The installation of the aforementioned additional software is not subject of the present installation guide.

2. SOFTWARE REQUIREMENTS

2.1 Table of Tested Program Versions

The installation of ccx 2.9 for OS X EL Capitan has been tested for the software versions from Tab. 1 below. Tab. 1 also contains cross-references to program details and related installation instructions. Links for the download of programs are given separately below in context with the installation instructions.

As long as not otherwise stated, the installation may work for program versions which differ from the tested ones from Tab. 1. Furthermore, available installation guides for previous versions of ccx and OS X are listed under the “Release Notes” on page i.

2.2 Mac OS X Specific

2.2.1 OS X El Capitan – Terminal App

The installation of ccx has been tested for the OS X-version of El Capitan from Tab. 1. Among others, the Unix-like Darwin OS and the Terminal App are inherent parts of Mac OS X.

The aforementioned Terminal App – also denoted as “Terminal” or “Terminal Window” - provides a command-line interface to OS X (cf. Section 3.2 below for details). As will be shown below, Unix commands and commands for software installation will be entered via Terminal. Furthermore, ccx will be launched via Terminal.

2.2.2 Xcode, CLTs (Command-Line Tools) and Modified GCC (GNU Compiler Collection)

Xcode is a broad suite for software development on Mac OS X and is provided by Apple. However, because Xcode is not an inherent part of OS X it needs to be installed in Section 5.1. The Xcode-version which has been tested together with El Capitan is listed in Tab. 1. Furthermore, only some of the capabilities of Xcode are required for the present installation as further outlined below.

Among others, the required modified GCC (GNU compiler collection modified by Apple) is an inherent part of Xcode. Furthermore, the required CLTs (command-line tools) are provided by Xcode; e.g. the command: “make” which will be applied below for the installation of software and the command: “gcc” to invoke the modified GCC. The aforementioned commands are entered via Terminal (cf. Section 2.2.1 above for “Terminal” and also Section 3.2 below).

Software	Tested Program Versions	References to Sections for Program Details and Installation	
		Details	Installation
Mac-Specific	OS X 10.11.1 El Capitan	2.2.1	-
	Xcode 7.1.1	2.2.2	2.2.2
	GCC 7.0.0 (modified GCC)	2.2.2	2.2.2
	GCC 4.9 (unmodified GCC)	2.2.3	5.2
	gfortran 5.2	2.2.4	5.3
CalculiX	ccx 2.9	2.3	4.3 (install files) 7.1 (install ccx)
External Solvers	SPOOLES 2.2	2.4	6.2
	ARPACK 2.1	2.4	6.3

Table 1 List of tested program versions. Details and installation of programs considered in referenced sections (cf. last two columns of table).

Except of the c-routines of ccx, the aforementioned modified GCC will be applied for the compilation of c-routines. For the compilation of the c-routines of ccx, therefore, the unmodified GCC from Section 2.2.3 is required (cf. also Section 3.1 for details).

The aforementioned CLTs and the modified GCC are inherent parts of Xcode and, therefore, are installed together with Xcode in Section 5.1 below (cf. Tab. 1 for the tested program versions).

Notes. Xcode is a broad suite for software development on OS X. For the installation of ccx, however, only the aforementioned CLTs of Xcode are required (installation of Xcode requires 5 GB, the installation of the CLTs only ca. 300 MB). Among others, the CLTs include the aforementioned modified GCC.

Alternatively to Xcode, only the CLTs may be installed. The separate installation of the CLTs, however, is not subject of the present installation instructions. Link for download of the CLTs:

<http://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/>

2.2.3 Unmodified GCC (GNU Compiler Collection) for OS X

The unmodified GCC is required for the compilation of the c-routines of ccx and is not identical with the modified GCC from Section 2.2.2 above (cf. also explanation of Xcode in Section 2.2.2 above and Section 3.1 below for details). The unmodified GCC-version from Tab. 1 will be installed in Section 5.2 below.

2.2.4 Fortran Compiler gfortran for OS X

The Fortran compiler: “gfortran” is required for the compilation of the Fortran-routines of ccx and ARPACK. For the present installation gfortran is considered as the default Fortran compiler (cf. also Section 3.1 for details). The installation of gfortran is subject of Section 5.3 below.

2.3 ccx (CalculiX CrunchiX)

The installation instructions are provided in Section 4 and Section 7 below. First of all, the source code for the ccx version from Tab. 1 is installed in Section 4. Furthermore, in Section 4 also the installation path and the file system of ccx is explained. The installation will be completed in Section 7 with the build of the binary of ccx.

2.4 External Solvers SPOOLES and ARPACK

- *SPOOLES*. SPOOLES (SParse Object Oriented Linear Equations Solver) is an established linear equation solver and among others provides the following capabilities: (a) single-threading, respectively serial processing, (b) MT (multi-threading), (c) MP (multi-processing for several separate CPU's). For the present basic installation of ccx (cf. Section 1.2) the aforementioned capabilities (a) and (b) for single- and multi-threading are considered. Single- and multi-threading may be invoked with the help of user defined environment variables (cf. Section 8.2 for examples). SPOOLES will be installed for the aforementioned capabilities in Section 6.2 below.
- *ARPACK*. ARPACK (ARnoldi PACKAGE) is a collection of Fortran routines for the solution of large-scale eigenvalue problems (for details consider the link in footnote 3 below). For the considered basic installation of ccx (cf. Section 1.2) a serial version of ARPACK will be installed. The dependencies of ARPACK on BLAS and LAPACK will be further discussed in context with the installation of ARPACK in Section 6.3 below.

³Link for details of ARPACK: <http://www.caam.rice.edu/software/ARPACK/>

3. PRE-ARRANGEMENTS FOR INSTALLATION

3.1 Compilers – modified GCC, unmodified GCC and gfortran

- *Modified GCC.* The modified GCC (GNU compiler collection) is installed together with Xcode and is the default compiler for the compilation of c-routines (cf. Section 5.1 for the installation of Xcode and details concerning the version and installation path of the modified GCC). The modified GCC includes associated versions of the individual compilers gcc (GNU c-compiler), g++ and clang. Except of the c-routines of ccx, the modified GCC will be applied for the compilation of c-routines.
- *Unmodified GCC for compilation of c-routines of ccx.* The unmodified GCC will be installed in Section 5.2 below and includes associated versions of the individual compilers: gcc (GNU c-compiler), g++ and clang (cf. Section 5.2 for details concerning the version and installation path of the unmodified GCC). The unmodified gcc will be applied in Section 7.1 for the compilation of the c-routines of ccx.
- *gfortran for compilation of Fortran-routines.* The Fortran compiler: “gfortran“ will be installed in Section 5.3 and is the default compiler for the compilation of the Fortran-routines of ccx and ARPACK (cf. Section 5.3 for details concerning the version and installation path of gfortran).

3.2 Terminal, Terminal Commands, HOME, LOGNAME, Login and Permissions

General. The following notes are provided for Mac users who usually don’t work with Unix. Also search the web for more instructions concerning the below discussed subjects.

Terminal. First of all, as further explained below, “Terminal“, respectively the Terminal App is a pre-installed application under Mac OS X (cf. also Section 2.2.1 above). The notation “Terminal“ or “Terminal window“ here is also used for the bash (bourne-again shell) Unix-shell which is provided by the Terminal App. Terminal provides the command-line interface to OS X, respectively the interface for the input of Unix-commands which are required for the installation of ccx. Furthermore, ccx may be launched from a Terminal window (cf. Section 7.2 for details).

To open the Terminal application, double click the Terminal-App in your “/Applications/Utilities“ folder. The Terminal icon then will appear in your Dock and a Terminal window will pop up on your screen. Use “Help“ in the menu bar of “Terminal“ to gain further instructions (how to open additional Terminal-windows, how to open further tabs in some window etc.) or visit e.g. the following site for additional instructions:
http://www.maclife.com/article/feature/25_terminal_tips_every_mac_user_should_know.

Terminal Commands. Commands, respectively Terminal commands, which are highlighted in the following with red letters, are entered after the \$-prompt in the Terminal window (cf. examples below). For execution of commands use the return key (also denoted as enter key).

HOME, LOGNAME. After a new Terminal window was opened, the shell wakes up in your home directory, denoted as HOME (shortcut ~; use keystrokes: alt + N for tilde ~). Use command `env` (for environment):

```
$ env <return>
```

to display the path to your HOME (usually: HOME=~=/Users/LOGNAME). The `env`-command also displays your LOGNAME as well as the path: PWD to your current (private) working directory. The path to your current working directory, respectively current folder also may be displayed with command: `$ pwd <return>` (`pwd` for “print working directory”).

Login. If you are not already logged in with your LOGNAME, use command: `$ login <return>` in Terminal. Afterwards, you will be asked for your LOGNAME and password to login.

Permissions. For several installation steps you need expanded access rights to the file system. To gain these rights, here the `sudo`-command will be used; syntax (examples for commands are given below):

```
$ sudo command <return>.
```

Afterwards, the OS will ask you first for your password before the command is executed. If required, in the present guide the `sudo`-command is set for installation commands. However, if you run into any permission-problems, first try to use `sudo` as specified above. If `sudo` doesn't work, you must login as root.

3.3 Editors, .bash_profile, Editing of Hidden File .bash_profile and Usage of .bash_profile to Set Paths and Export Paths

General. The following notes are provided for Mac users who usually don't work with Unix. Also search the web for more instructions concerning the below discussed subjects.

Editors. The Finder App, respectively the “Finder“ is a convenient tool for file handling on your Mac. However, many of the files on your Mac are hidden in ⁴Finder windows, among others the file: “.bash_profile” which is required later for the ccx installation. In the following it will be explained how hidden files may be created, opened and edited. Furthermore, the path setting in file: “.bash_profile” is briefly explained.

File .bash-profile. The file: “.bash-profile” is located in your HOME and amongst other settings contains path settings, respectively is used to ⁵export paths. The dot in the first place of the file name marks file: “.bash_profile“ as a hidden file which, therefore, is not

⁴Note. Commands for displaying hidden files in Finder are available, but for several reasons will be not used here.

⁵Note. If you open a Terminal window, the OS “exports“ the path settings found in .bash_profile and then uses these paths to search for executables. Furthermore, amongst others, the path: /usr/bin is a default path which the OS uses to search for executables.

displayed in the Finder window. A simple example for a `.bash_profile` with path settings is given below.

Editing of hidden files, respectively of file: “.bash_profile”. The editing will be demonstrated for setting, respectively exporting paths with `.bash_profile`. First, execute the following **commands** in Terminal (don't type in **# comments**; shortcut `~` for your HOME is used; keystrokes for typing tilde `~`: `⌘ + n`):

```
$ cd ~/ <return> # Change directory (cd) to your home directory ~ (Home),
                  # where your .bash_profile is located.

$ ls -l <return> # Long (option -l) list (command ls) displays visible files in ~.
                  # Note that file .bash_profile is not displayed. Command ls -l,
                  # therefore, displays all visible files.

$ ls -la <return> # Long (option “l”) list (ls) for all (option “a“ for all visible and
                  # invisible) files in ~. Now, .bash_profile is contained in the list
                  # of files. Command ls -la, therefore, displays all visible and
                  # invisible files.
```

If `.bash_profile` is not listed after the execution of the last command: `ls -la`, then create it now with the following commands:

```
$ cd ~/ <return> # Change directory (cd) to your home directory HOME.

$ touch .bash_profile <return> # Create .bash_profile.

$ ls -la <return> # Verify, if .bash_profile has been created.
```

Next, open `.bash_profile` with editor TextEdit by using the commands:

```
$ cd ~/ <return> # Set home directory as your working directory.

$ open -e .bash_profile <return> # Opens .bash_profile in TextEdit.
```

Set paths in file: “.bash_profile” and export paths. Now, the `.bash_profile` is ready for editing, respectively to enter a new path-setting in `.bash_profile`. Next an example is given, how paths may be set in `.bash_profile`. For path setting you may add the following lines in your `.bash_profile`:

```
#!/bin/bash

export PATH=$PATH
export PATH=$PATH:/Users/LOGNAME/NETGEN_INSTALL/bin
```

In the above lines, “`export`” is a Unix command, `PATH` an environment variable, `$PATH` the value of that variable, colon “`:`” is used as separator between paths. The line: “`export PATH=$PATH`” updates the `PATH` variable and the last line sets the added path: `/Users/LOGNAME/NETGEN_INSTALL/bin` to some executable which is contained in

CalculiX • ccx Installation Guide Mac OS X •

directory: “bin“. Add as any many paths as you need. Finally, save `.bash_profile` and quit TextEdit. The required path settings for the considered software are specified in the installation instructions below.

To activate the changes, respectively to export the new paths in `.bash_profile` use commands:

```
$ cd ~/ <return> # Set home directory where .bash_profile is located.
```

```
$ source .bash_profile <return>
```

in current Terminal window. Alternatively, you may open a new Terminal window, because as soon as a new Terminal window is opened the OS will automatically execute the commands in `.bash_profile`, i.e. export path settings etc.

Control the path settings with the following command:

```
$ echo $PATH <return>.
```

Notes. The OS also automatically searches the default paths: `/usr/bin` as well as `/usr/local/bin` for executables. Instead of TextEdit, also one of the command line editors may be used (vim-editor etc.).

4. INSTALLATION PATH AND INSTALLATION OF CCX FILES

4.1 General

In the present section the ccx source code and html-help files will be downloaded and installed. The installation path may be chosen according to Section 4.2. Furthermore, the file system for the installation of cgx and ccx as well of the external solvers is explained in Section 4.2.

Note. After the installation of ccx files continue installation in the sequence of Section 5 to Section 7.

4.2 Installation Path “calculix_inst_path“ and File System

Installation Path

Both ccx and cgx as well as the external solvers are installed for the same installation path. First of all, choose option A or B for the installation ⁶path: “calculix_inst_path“:

- **Option A** (default installation path):

```
“calculix_inst_path” =/usr/local
```

- **Option B** (example for user defined path):

```
“calculix_inst_path”=~/Applications=/Users/LOGNAME/Applications
```

Option A is in compliance with the Unix file system hierarchy standard and also is the default path for the installation of ccx. For option B the path may be chosen freely. For option B an example is given above, for which ccx will be installed under directory “Applications“ under your home directory “~” (cf. Section 3.2 for the home directory).

Because in ccx the paths are already set relative to directory “src” of ccx (cf. file system in Fig. 1), neither for option A nor for option B paths must be set for the installation of ccx (cf. Section 7.1 for details).

Prior to the installation the directories of the path: “calculix_inst_path” must exist. Create them now, if necessary. Use the Finder to create directories or the sequence of Unix **commands**: \$ **cd** “path_to_Directory” <return> (change to directory: “path_to_Directory” under which a subdirectory has to be created), then type: \$ **mkdir** “dir_name” <return> to create the subdirectory “dir_name“. Furthermore, use: \$ **ls -l** <return> to list the content of some directory, use **sudo** before commands (e.g. **sudo mkdir**) if the permission for the making of directories is denied.

⁶Note. The notation “calculix_inst_path“ here is used for documentation reasons only and, therefore, is not an environment variable for the installation of ccx. Otherwise stated, replace “calculix_inst_path“ in commands with full path specification according to options A, B.

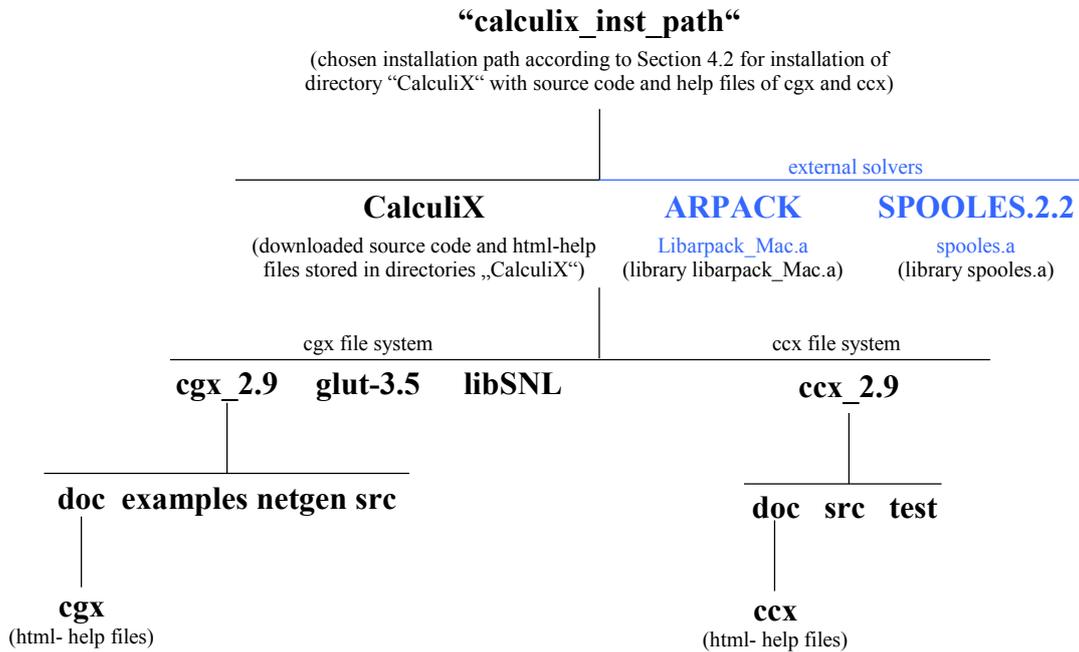


Figure 1 File system after installation of source code and html-help files of ccx and cgx as well as of external solvers SPOOLES and ARPACK. Directories printed in bold.

File System

After the installation of the ccx source code and the external solvers the file system for the present ccx installation looks as depicted in Fig. 1. Calculix (ccx and cgx) and the external solvers, therefore, must be installed for the installation path “calculix_inst_path” from Section 4.2. The ccx source code is contained in directory “src“, the html-help files in directory “ccx” and the test examples in directory “test”. Furthermore, the libraries of the external solvers are build in directories “SPOOLES.2.2” and “ARPACK” as depicted in Fig. 1.

4.3 Install ccx Files

Steps a, b for installation of files for the ccx-version from Tab. 1:

(a) Download the following packages from Calculix site <http://www.dhondt.de/>:

- “the source code“ ccx_2.9.src.tar.bz2,
- “the documentation html“ ccx_2.9.htm.tar.bz2,
- “test examples“ ccx_2.9.test.tar.bz2.

The files will be downloaded on your desktop folder.

The links for the download of the Installation Guide [1] and User’s Guide [2] are provided together with the list of references on page 30.

- (b) Next, install downloaded packages. First drag packages: “ccx_2.9.src.tar.bz2“, “ccx_2.9.htm.tar.bz2“ and “ccx_2.9.test.tar.bz2“ from Desktop to your installation directory: “calculix_inst_path” from Section 4.2 (cf. also the file system in Fig. 1). Then use Terminal **commands** (without **comments**) to extract the files from the packages one by ⁷one as follows:

```
$ cd "calculix_inst_path" <return> # Set directory for installation of ccx.

$ bunzip2 ccx_2.9.src.tar.bz2 <return> # Unzip package with source code.

$ tar -xvf ccx_2.9.src.tar <return> # Installs directory "src" with source code. Path:
# "calculix_install_path"/CalculiX/ccx_2.9/src
# (cf. file system in Fig. 1).

$ bunzip2 ccx_2.9.test.tar.bz2 <return> # Unzip package with test examples.

$ tar -xvf ccx_2.9.test.tar <return> # Installs directory "test" with examples. Path:
# "calculix_install_path"/CalculiX/ccx_2.9/test
# (cf. file system in Fig. 1).

$ bunzip2 ccx_2.9.htm.tar.bz2 <return> # Unzip package with html-help files.

$ tar -xvf ccx_2.9.htm.tar <return> # Installs directory "ccx" with html-help files. Path:
# Path:"calculix_install_path"/CalculiX/ccx_2.9/doc/ccx
# (cf. file system in Fig. 1).
```

The directories: “src“ with the source code, “doc” with the html-documentation and “test“ with the test examples now are installed under the path (cf. also Fig. 1 for the file system): “calculix_install_path”/CalculiX/ccx_2.9.

⁷Note. Wildcards don’t work for tar-command.

5. INSTALL MAC-SPECIFIC SOFTWARE

5.1 Xcode and Associated Modified GCC

For the installation instructions the tested program versions from Tab. 1 are considered (cf. also Section 2.1 for details).

Xcode may be downloaded for free from “Categories“ in the App Store. Then, follow installer instructions. The Xcode App will be installed in your “Applications“ folder.

As already mentioned before in Section 2.2.2, the modified GCC comes together with Xcode and is installed under the default path: `/usr/bin`. Furthermore, the modified GCC includes the individual compilers: `gcc` (GNU c-compiler), `g++`, `clang`. The compiler version depends on the Xcode-version and may differ from the versions given for the modified GCC in Tab. 1. The modified GCC will be applied according to Section 3.1.

The installation of the modified GCC now will be tested exemplarily for GCC 7.0.0 and `g++`. If the installation was successful, after the execution of the Terminal **command**: `$ which g++` <return> the path to the `g++` compiler is displayed as follows:

```
/usr/bin/g++
```

Because the afore-displayed path: `/usr/bin` is a default search path for executables, the modified GCC is the default c-compiler on your Mac.

After the execution of Terminal **command**: `$ g++ -v` <return> (option `-v` for version) the screen output for the version of `g++` will be displayed as follows:

```
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/
include/c++/4.2.1
Apple LLVM version 7.0.0 (clang-700.1.76)
Target: x86_64-apple-darwin15.0.0
Thread model: posix
```

As already explained in Section 3.1 above, in addition the unmodified GCC is required for the compilation of `ccx` and, therefore, will be installed next.

5.2 Unmodified GCC

For the installation instructions the tested program version from Tab. 1 is considered (cf. also Section 2.1 for details). Furthermore, for the installation point 2 of the following instructions is considered:

<https://wiki.helsinki.fi/display/HUGG/Installing+the+GNU+compilers+on+Mac+OS+X>

Steps 1 – 4 for the installation of GCC 4.9 from Tab. 1:

- (1) Download pre-build GCC 4.9, respectively gunzipped file `gcc-4.9-bin.tar.gz` for Mavericks, Yosemite & El Capitan from site:

http://sourceforge.net/projects/hpc/files/hpc/gcc/gcc-4.9-bin.tar.gz/download?use_mirror=cznic&download=

- (2) Open Terminal window and then go to download directory (usually directory "Downloads" or directory "Desktop") of gcc-4.9-bin.tar.gz. If for example gcc was downloaded on your Desktop use terminal **command**:

```
$ cd ~/Desktop <return>.
```

- (3) If gcc-4.9-bin.tar.gz was not automatically unzipped, then unzip file with Terminal **command**:

```
$ gunzip gcc-4.9-bin.tar.gz <return>
```

- (4) For installation of GCC 4.9 use terminal command (without **comment**):

```
$ sudo tar xvf gcc-4.9-bin.tar -C / <return> # Installs GCC 4.9 under default directory:
# /usr/local/bin.
```

After step 4, among others the compilers gcc and g++ are installed in: /usr/local/bin.

The test of the installation will be explained exemplarily for gcc. If the installation was successful the Terminal **command**: `$ which /usr/local/bin/gcc <return>` will display the path to the unmodified GNU gcc compiler as follows (screen output):

```
/usr/local/bin/gcc
```

Furthermore, after execution of Terminal **command**: `$ /usr/local/bin/gcc -v <return>` (option -v for version) the screen output for the version of g++ should look as follows:

```
Using built-in specs.
COLLECT_GCC=/usr/local/bin/gcc
COLLECT_LTO_WRAPPER=/usr/local/libexec/gcc/x86_64-apple-darwin14.0.0/4.9.2/lto-wrapper
Target: x86_64-apple-darwin14.0.0
Configured with: ../gcc-4.9-20141029/configure --enable-languages=c++,fortran
Thread model: posix
gcc version 4.9.2 20141029 (prerelease) (GCC)
```

As already explained in Section 3.1 above, the unmodified GCC 4.9 will be applied only for the compilation of the c-routines of ccx in Section 7.1 below. The path for gcc: /usr/local/bin/gcc has to be set manually in the Makefile of ccx (cf. Section 7.1 for details).

5.3 Fortran Compiler gfortran

The tested Fortran compiler gfortran 5.2 from Tab. 1 will be considered for the installation. First, download the binary of gfortran 5.2 for OS X El Capitan from site: <http://hpc.sourceforge.net/>.

The zipped tar-file gfortran-5.2-bin.tar.gz will be downloaded on your Desktop. Next, install compiler with the following Terminal **commands** (without **comments**):

```
$ cd ~/Desktop <return> # Change to directory Desktop.
```

```
$ gunzip gfortran-5.2-bin.tar.gz <return> # Unzip gfortran-5.2-bin.tar.gz to
# gfortran-5.2-bin.tar.
```

```
$ sudo tar -xvf gfortran-5.2-bin.tar -C / <return> # Installs gfortran for default path:
# /usr/local/bin. Binary “gfortran”,
# therefore, located under directory:
# /usr/local/bin/gfortran.
```

After the execution of the last command the binary: “gfortran” will be installed under the default path: /usr/local/bin/gfortran. No additional path settings are required.

Next, test the installation of binary “gfortran” and the installed version. Change to some working directory and use Terminal **commands** (without **comments**):

```
$ which gfortran <return> # Displays path to binary “gfortran”.
```

```
$ gfortran -v <return> # Display version (option “-v”) of gfortran.
```

The installation of gfortran was successful if the path is displayed as follows:

```
/usr/local/bin/gfortran
```

and the version of gfortran as follows:

```
Using built-in specs.
COLLECT_GCC=gfortran
COLLECT_LTO_WRAPPER=/usr/local/gfortran/libexec/gcc/x86_64-apple-darwin14/5.2.0/lto-wrapper
Target: x86_64-apple-darwin14
Configured with: ../gcc-5.2.0/configure --prefix=/usr/local/gfortran --with-gmp=/Users/fx/devel/gcc/deps-
static/x86_64 --enable-languages=c,c++,fortran,objc,obj-c++ --build=x86_64-apple-darwin14
Thread model: posix
gcc version 5.2.0 (GCC)
```

The Fortran compiler gfortran 5.2 will be applied according to Section 3.1.

6. INSTALL EXTERNAL SOLVERS

6.1 General

According to the definition of the basic installation of ccx from Section 1.2, the external solvers SPOOLES (SParse Object Oriented Linear Equations Solver) and ARPACK (AR-noldi PACKAGE) will be installed (cf. also Section 2.4 for details).

For the sake of simplicity, in the text of the subsequent installation instructions the tested program versions from Tab. 1 are considered (cf. also Section 2.1 for details).

6.2 SPOOLES

- *General.* The tested linear equations solver SPOOLES 2.2 from Tab. 1 will be considered for the subsequent installation instructions (cf. Section 2.4 for solver details).

SPOOLES will be installed for the chosen installation path from Section 4.2. The required library: “spooles.a” with the single- and multi-threading (MT) routines of SPOOLES then will be build in directory: “spooles.2.2“ (cf. Fig. 1 for the file system): “calculix_inst_path”/spooles.2.2/spooles.a.

The present installation instruction is based on the instruction: “Install” which may be downloaded from the SPOOLES-site: <http://netlib.sandia.gov/linalg/spooles/spooles.2.2.html>

- *Installation.* Steps a to f:

- (a) Download SPOOLES package “spooles.2.2.tgz” from site:

<http://netlib.sandia.gov/linalg/spooles/spooles.2.2.html>

The package: “spools.2.2.tgz” will be downloaded on your Desktop.

- (b) Install source files. Double click icon: “spooles.2.2.tgz” to unzip package, then drag unzipped directory: “spooles.2.2“ from Desktop to the installation directory according to your installation path: “calculix_inst_path”. The file system for the installation of SPOOLES now should look as depicted in Fig. 1.
- (c) Modify files Lock.h, drawTree.c, Make.inc:

- Backup files prior to modifications.

- File: “Lock.h” – set thread type to POSIX. Location of file: “calculix_inst_path”/spooles.2.2/Lock/Lock.h. Open file with an editor of your choice and then check, if after line (without quotes): “#define TT_POSIX 2” the line: “#define THREAD_TYPE TT_POSIX” is already set for the required thread type POSIX. If not, insert line (without quotes): “#define THREAD_TYPE TT_POSIX” after existing line “#define TT_POSIX 2“. Save file and quit editor.

CalculiX • ccx Installation Guide Mac OS X •

- File “makeGlobalLib” – replace file name: “drawtree.c” in considered file with “draw.c” (cf. [1]). Location of file: “calculix_inst_path”/spooles.2.2/Tree/src/makeGlobalLib. Open file “makeGlobalLib” with an editor of your choice. Then find “drawtree.c” and replace it with “draw.c” (without quotes). Save file.

Note. File “drawTree.c” does not exist under path: “calculix_inst_path”/spooles.2.2/Tree/src.

- File “Make.inc” – modify code. Location of file: “calculix_inst_path”/spooles.2.2/Make.inc. Open file with an editor of your choice, then **activate green lines** (remove “#” at first place of line) and **deactivate red lines** (add “#” at first place of line) as depicted below:

```
# place your favorite compiler here
#
# for solaris
#
CC = gcc
# CC = /usr/lang-4.0/bin/cc
#
# for sgi
#
# CC = cc
#
# for hp
#
# CC = /opt/mpi/bin/mpicc
#
#-----
#
# set the compiler flags
#
# OPTLEVEL =
# OPTLEVEL = -g -v
# OPTLEVEL = -O
# OPTLEVEL = -xO5 -v
OPTLEVEL = -O3
# OPTLEVEL = -O4
# CFLAGS = -Wall -g
# CFLAGS = -Wall -pg
# CFLAGS = $(OPTLEVEL) -D_POSIX_C_SOURCE=199506L
# CFLAGS = $(OPTLEVEL)
CFLAGS = -Wall $(OPTLEVEL)
#
#-----
```

Save file and quit editor.

- (d) Build serial library spools.a. Use Terminal **commands** (without **comments**):

```
$ cd “calculix_inst_path”/spooles.2.2 <return> # Set directory.

$ make lib <return> # Create serial library spooles.a. Location:
# “calculix_inst_path”/spooles.2.2/spooles.a.
```

Note. The last command generates several warnings which may be ignored.

- (e) Build multi-threading (MT) library and merge library in spools.a. Use Terminal **commands** (without **comments**):

```
$ cd calculix_inst_path/spooles.2.2/MT/src <return> # Set directory.
# Consider that the next command "makeLib" is case sensitive:
$ make makeLib <return> # Creates library and merges MT library in library
# spools.a from previous step (d).
```

Note. The last command automatically removes all object files. Warnings may be ignored.

- (f) Repeat installation of libraries. If you want to repeat the installation for some reason, first make the installation clean with Terminal **commands** (without **comments**):

```
$ cd "calculix_inst_path"/spooles.2.2 <return> # Set directory.
$ make clean <return> # Removes object files and file "library.a"
```

- *Test installation.* The installation of SPOOLES will be tested in Section 8.2 below.

6.3 ARPACK

- *General.* The tested solver ARPACK 2.1 from Tab. 1 will be considered for the subsequent installation instructions (cf. Section 2.4 for solver details).

For the considered basic installation of ccx (cf. Section 1.2) a serial version of ARPACK will be installed. Furthermore, ARPACK will be installed for the chosen installation path from Section 4.2. The required library Libarpack_Mac.a then will be build in directory "ARPACK" (cf. file system in Fig. 1): "calculix_inst_path"/ARPACK/libarpack_Mac.a.

According to Section 1.2 above, libraries for BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage) are required for the installation of ARPACK. Basically, the following two options a, b exist to provide the aforementioned libraries: (a) compile ARPACK with the routines for BLAS and LAPACK which come with the ARPACK-package, (b) link ARPACK to libraries for BLAS and LAPACK. For the reasons outlined in the note below, option (a) has been chosen, i.e. build of a self-contained library for ARPACK, which includes the required routines for BLAS and LAPACK.

The present installation instruction of ARPACK is based on the "README" which comes with the ARPACK-package.

Note. As already outlined above, alternatively to option (a), ARPACK may be installed according to option (b) with links to libraries of BLAS and LAPACK. Options for links: (1) link against libraries which come with OS X, (2) Install source for BLAS and LAPACK separately, build libraries, then link against these libraries, (3) build libraries with the routines of BLAS and LAPACK which come

with ARPACK, then link against these libraries. The aforementioned option (1) was tested only with links to the libraries: /usr/lib/ libf77lapack.dylib, /usr/lib/libblas.dylib which come with OS X. Benchmark tests revealed that for option (1) the errors of calculated eigenvalues are larger if compared to the results which are derived with the aforementioned self-contained libraries according to option (a). The latter indicates that the calculated eigenvalues may be sensitive with respect to the versions of the libraries of BLAS and LAPACK. The aforementioned sensitivity is also in accordance with the following note in the “README“ which is provided with the ARPACK-package: “*** NOTE *** Unless the LAPACK library on your system is version 2.0, we strongly recommend that you install the LAPACK routines provided with ARPACK.“ Accordingly, for the present installation, the LAPACK and BLAS routines have been used which come with the ARPACK-package. Otherwise stated, calculated eigenvalues should be carefully tested against benchmark calculations if ARPACK is installed with links to BLAS and LAPACK according to options (1) to (2) above. Exception: link according to option (3) above.

- *Installation.* Steps a to f:

- (a) Download GNU-zipped tar-files: “arpack96.tar.gz“ and “patch.tar.gz“ from site:

<http://www.caam.rice.edu/software/ARPACK/>

The tar-files will be downloaded on your Desktop.

- (b) Install source files. Drag tar-files: “arpack96.tar.gz“ and “patch.tar.gz“ from Desktop to the installation directory according to the chosen installation path from Section 4.2: “calculix_inst_path“. The tar-files, therefore, now should be located on the same directory level as directory “CalculiX“ from Fig. 1.

Next, install files with Terminal **commands** (without **comments**):

```
$ cd “calculix_inst_path” <return> # Set working directory.
```

```
$ tar -xzf arpack96.tar.gz <return> # Creates directory “ARPACK“ and installs
# source files under directory “ARPACK”.
```

```
$ tar -xzf patch.tar.gz <return> # Installs patch of source code under
# directory “ARPACK”.
```

The source files of BLAS and LAPACK are installed in directories “BLAS” and “LAPACK” (cf. “calculix_inst_path“/ARPACK/BLAS and “calculix_inst_path“/ARPACK/LAPACK for the aforementioned source files).

- (c) Modify files second.f, ARmake.inc:

- Backup files prior to modifications.

- File “second.f” – modify code. Location: “calculix_inst_path“/ARPACK/UTIL/second.f. Open file “second.f” with an editor of your choice and

replace the following two existing lines:

```
*   REAL          ETIME
    EXTERNAL      ETIME
```

with the following two lines:

```
* Modified June, 2014 for gfortran 4.9.0
  external real etime
```

Save file and quit editor.

- File “ARmake.inc” – modify code. Location of file: “calculix_inst_path”/ARPACK/ARmake.inc. Open file with an editor of your choice. Then replace **red code** to the left of arrows with **green code** to the right of arrows and in addition replace: “*calculix_install_path*” in the first line below with the full path to the chosen installation directory (cf. Section 4.2 for the installation path):

```
home = $(HOME)/ARPACK → home = “calculix_inst_path”/ARPACK
PLAT = SUN4 → PLAT = MAC
LAPACKLIB = → # LAPACKLIB =
BLASLIB = → # BLASLIB =
ALIBS = $(ARPACKLIB) $(LAPACKLIB) $(BLASLIB) → ALIBS = $(ARPACKLIB)
FC = f77 → FC = gfortran
FFLAGS = -O -cg89 → FFLAGS = -c -O -v### -dumpmachine
MAKE = /bin/make → MAKE = make
SHELL = /bin/sh → SHELL = /bin/bash
```

Save file and quit editor.

- (d) Build serial library libarpack_MAC.a. Use Terminal **commands** (without **comments**):

```
$ cd “calculix_inst_path”/arpack <return> # Set directory.
$ make lib <return> # Create serial library libarpack_MAC.a. Location:
# “calculix_inst_path”/ARPACK/ libarpack_MAC.a.
```

Note. The last command generates several warnings, which may be ignored.

- (e) Remove object files. Use Terminal **commands** (without **comments**):

```
$ cd calculix_inst_path/arpack <return> # Set working directory.
$ make clean <return> # Removes object files.
```

- (f) Repeat installation of library. If you want to repeat the installation for some reason, first remove object files according to previous step (e) above.

- *Test installation.* The installation of ARPACK will be tested in Section 8.1 below.

7. INSTALL AND LAUNCH CCX, TEST INSTALLATION, GETTING STARTED AND TEST EXAMPLES

7.1 Install ccx

The source files of ccx as well as the libraries of the external solvers already have been installed in Section 4 and Section 6 for the installation path:

“calculix_install_path” (cf. Section 4.2 for details).

In the present section the basic installation of ccx according to Section 1.2 will be completed; i.e. the installation of ccx for multi-threading (MT) together with the additional external solvers SPOOLES (single- and multi-threading) and ARPACK (single threading). The installation first requires some minor modifications of files and finally to build and install ccx. The location of the libraries of external solver is already set relative to the build-directory: “src” of ccx with path (cf. Fig. 1): “calculix_install_path”/CalculiX/ccx_2.9/src. Furthermore, the binary of ccx will be copied to the default search path: /User/local/bin for executables. Accordingly, no explicit path setting is required for the installation of ccx.

Installation steps (1) to (4):

- (1) *Modify file: “calculix.h”*. Location: “calculix_install_path”/calculix/ccx_2.9/src/calculix.h. Open file with an editor of your choice. Then find line (located close to top of file):

```
#define Linux 1
```

and insert the following two lines before existing line “#define Linux 1”:

```
/* mac include pthread_t.h */
```

```
#include <pthread.h>
```

Save file and quit editor.

- (2) *Modify file: “Makefile”*. Location: “calculix_install_path”/calculix/ccx_2.9/src/Makefile. Open file with an editor of your choice. Then replace the following **existing red code** to the left of arrows with **new green code** to the right of arrows:

```
CFLAGS = -Wall -O3 -I ../../SPOOLES.2.2 -DARCH="Linux" -DSPOOLES -DARPACK -  
DMATRIXSTORAGE
```

→

```
CFLAGS = -Wall -O3 -pthread -I ../../SPOOLES.2.2 -DARCH="Linux" -DSPOOLES -DARPACK -  
DMATRIXSTORAGE -DUSE_MT
```

```
CC=cc → CC=/usr/local/bin/gcc
```

```
LIBS = \
    $(DIR)/spooles.a \
    ../../../../ARPACK/libarpack_INTEL.a \
    -lpthread -lm -lc
```

→

```
LIBS = \
    $(DIR)/spooles.a \
    ../../../../ARPACK/libarpack_MAC.a
```

Save file and quit editor.

- (3) *Build and install ccx*. Use below specified Terminal **commands** (without **comments**) and replace “calculix_install_path” in commands with full path specification according to Section 4.2:

```
$ cd “calculix_install_path”/calculix/ccx_2.9/src <return> # Change to directory with
# source code and makefile.
```

```
$ rm *.o <return> # Remove object files. Use command always before
# re-building ccx.
```

```
$ make <return> # Build binary: “ccx_2.9” of ccx. Some warnings will appear
# which are not relevant for the functionality of ccx.
```

Command “**make**“, therefore, builds the binary “ccx_2.9” of ccx in directory “src” (cf. also the file system in Fig. 1): “calculix_install_path”/calculix/ccx_2.9/src/ccx_2.9.

Before re-building ccx, remove object files with **command**: `$ rm *.o<return>` as already specified above.

- (4) *Install binary of ccx in: /usr/local/bin*. First, check if: /usr/local/bin already exists with Terminal **command**: `$ cd /usr/local/bin <return>`. If directory “bin” already exists, then continue with: “Further steps” below. Otherwise, i.e. if the screen message: “No such file or directory” appears, then create: /usr/local/bin now with Terminal **command**: `$ sudo mkdir /usr/local/bin <return>`.

Further steps:

```
$ cd “calculix_install_path”/calculix/ccx_2.9/src <return> # Change to directory src
# which contains binary of
# ccx.
```

```
$ sudo cp ccx_2.9 /usr/local/bin <return> # Copy binary: “ccx_2.9” in /usr/local/bin.
```

The copy: “ccx_2.9” of the binary of ccx in /usr/local/bin/ccx_2.9 enables to launch ccx from working directories. You also might change the protection with **command**: `$ sudo chmod ao+rx /usr/local/bin/ccx_2.9 <return>`.

Alternatively, instead of copying the binary of ccx in: /usr/local/bin as explained above, you may set the path: “calculix_install_path”/calculix/ccx_2.9/src to the binary of “ccx“ in your .bash_profile (cf. Section 3.3 for instructions and examples).

7.2 Launch ccx and Test Installation

For the launch and test of the installation of ccx the example: “beamp.inp” will be considered. The input file “beamp.inp” is provided in directory “test”: “calculix_install_path”/calculix/ccx_2.9/test/beamp.inp.

Use Terminal **commands** without **comments**:

```
$ cd calculix_inst_path/calculix/ccx_2.9/test <return> # Set working directory: “test”.
```

```
$ ccx_2.9 beamp <return> # Run ccx with input file “beamp.inp“.
```

The installation basically works, if after the last command the message: “Job finished“ is displayed on your screen and if in directory: “test” the files: “beam.dat” (contains user-directed output) and “beam.frd” (contains plot data for cgx) have been created (cf. also the ccx User’s Guide [2] for further explanation of the file types: .inp, .dat and .frd). The tests are continued in Section 8 below.

See the ccx manual [2] and also the subsequent Section 7.3 below how to get started with ccx.

7.3 Getting Started With ccx

Consider the html-help files of cxx which are provided by cgx (cf. cgx installation guide [4] for details) and references:

- User’s guide [2].
- Tutorial [3]: “Getting Started With CalculiX“.

7.4 Test Examples

Test examples are provided in the ccx manual [2]. Furthermore, the functionality of ccx will be tested for several test examples in Section 8 below.

8. TESTS - TEST EXAMPLES, NUMERICAL ACCURACY AND MULTI-THREADING

8.1 Test Examples and Numerical Accuracy

The test examples in the test-directory: “`calculix_install_path`”/calculix/ccx_2.9/test will be considered (cf. also [2], Section 10 for details). For the aforementioned examples the following files are provided in the test-directory (cf. also the ccx manual [2] for the file extensions: `.inp`, `.dat`, `.frd`):

<filename>.inp: input files for ccx.

<filename>.dat.ref: files with reference data for user-directed output.

<filename>.frd.ref: files with reference data for ccx-directed plot data.

If you run one of the test examples, ccx provides the output files: <filename>.dat and <filename>.frd, which may be compared with the aforementioned reference output files.

In the test-directory also the script: “compare” is provided which automatically runs all examples and also automatically compares the computational results with the aforementioned reference data. Run script: “compare” with Terminal **commands** (without **comments**):

```
$ cd “calculix_inst_path”/calculix/ccx_2.9/test # Set working directory test.
```

```
$ ./compare # Run script “compare”.
```

During runtime of “compare” screen output is provided with a brief list of executed examples and general error messages. If for the aforementioned comparison errors exceed pre-specified values, then a message is written to file (cf. example below): “error.<random number>” in the test-directory; random number automatically generated. Accordingly, file: “error.<random number>” is empty, if all errors are within the pre-specified ranges. Otherwise, follow the instructions given in references [1], [2].

For the present installation the execution of “compare” finished with the following three lines of screen output:

```
check the existence of file error.22856
if this file does not exist, the present results
agree with the reference results
```

and in file: “error.22856” the following errors are reported:

```
deviation in file beamptied6.dat
line: 14 reference value: 2.339439e+06 value: 2.086567e+06
absolute error: 2.528720e+05
largest value within same block: 3.164432e+06
relative error w.r.t. largest value within same block: 7.991071 %
```

beamread.dat and beamread.dat.ref do not have the same size !!!!!!!!!!!

deviation in file cubef2f1.frd

line: 7502 reference value: 3.947900e+01 value: 3.981660e+01

absolute error: 3.376000e-01

largest value within same block: 3.360410e+02

relative error w.r.t. largest value within same block: 0.100464 %

deviation in file cubef2f2.frd

line: 7468 reference value: -1.882140e+01 value: -3.039420e+06

absolute error: 3.039401e+06

largest value within same block: 3.374260e+02

relative error w.r.t. largest value within same block: 900760.812326 %

deviation in file segment.frd

line: 182049 reference value: 2.110000e+04 value: -2.029550e+04

absolute error: 4.139550e+04

largest value within same block: 1.488060e+05

relative error w.r.t. largest value within same block: 27.818435 %

deviation in file simplebeampipe3.dat

line: 50 reference value: -8.419138e+03 value: -1.496024e+04

absolute error: 6.541102e+03

largest value within same block: 8.748470e+04

relative error w.r.t. largest value within same block: 7.476853 %

For 6 of the 329 provided test examples, therefore, the errors exceed the pre-specified values. The reported errors are currently subject of further investigations.

8.2 Invoke and Test Multi-Threading (MT) for Spoolers

- *General.* According to Section 2 in the ccx manual [2] several MT-capabilities are available for ccx and external solvers.

According to Section 2 in [2] MT may be invoked with the help of user defined ENVs (environment variables), whereas prior to the execution of ccx one or more ENVs may be set with Terminal **command**: `$ export ENV=value <return>`.

General procedure to invoke MT according to the following steps 1 - 3 (cf. also Section 2 in [2] for details):

- (1) If required, set number of available logical cores for MT. The notation “logical cores“, respectively “logical CPUs“ is explained in subsection: “*Physical Cores and Logical Cores*” below. Use: `NUMBER_OF_CPUS=ncpu` to set number ncpu of logical CPUs which are available for MT.

For the present Mac-installation ccx detects the number: ncpu of logical cores automatically. Accordingly, the ENV: “NUMBER_OF_CPUS” is not required for the present Mac-version of ccx.

- (2) Set optional global switch for MT. The ENV: `OMP_NUM_THREADS=nthreads` invokes all MT-capabilities which are available for ccx and the considered external solvers (cf. Section 2 in [2] for details), whereas the number: `nthreads` sets the number of threads for MT. Additional ENVs may be necessary for fine-tuning of specific MT-capabilities, for example if user subroutines for materials are used (cf. Section 2 in [2] for details).
- (3) Set optional switches for specific MT-capability. For example the ENV: `CCX_NPROC_EQUATION_SOLVEROMP_NUM_THREADS=nthreads` invokes MT for solvers only, whereas the number: `nthreads` sets the number of threads for MT. Additional ENVs for specific MT-capabilities are available in [2], Section 2.

The further explanations are restricted on the ENVs introduced above, which may be set as follows to invoke MT for Spooles:

- `NUMBER_OF_CPUS`,
- `OMP_NUM_THREADS`,
- `CCX_NPROC_EQUATION_SOLVEROMP_NUM_THREADS`.

Furthermore, according to steps 1 – 3 above for the present Mac-version of ccx basically the ENV: `CCX_NPROC_EQUATION_SOLVEROMP_NUM_THREADS` is sufficient to invoke MT for solvers, respectively for Spooles.

- *Physical Cores and Logical Cores.* The CPU on your Mac (for OS X) has a certain number of physical cores, respectively physical CPUs. The number of available logical cores (addressable from OS and, therefore, relevant for parallel-processing), however, in general is different from the number of physical cores if hyper-threading is supported. Use Terminal **commands** (without **comments**) to display the number of available physical and logical cores:

```
$ sysctl hw.physicalcpu <return> # Display number of physical cores.
```

```
$ sysctl hw.ncpu <return> # Display number of logical cores.
```

For the further explanation of MT the number: `ncpu` of available logical cores is relevant.

- *Default Threading-Mode.* If none of the below specified environment variables is set, single-threading for Spooles (and ccx) is invoked.
- *MT for Spooles.* For MT (parallel-processing) the max number: `nthreads` of threads for MT may be set as explained below. The upper bound for `nthreads` is determined by the number: `ncpu` of available logical cores: $0 < nthreads \leq ncpu$; display `ncpu` with previously explained Terminal **command**: `$ sysctl hw.ncpu <return>`.

As already outlined in subsection “General“ above, MT may be invoked by setting one or more of the following environment variables ENVs:

- *NUMBER_OF_CPUS=ncpu*. With the ENV: *NUMBER_OF_CPUS* the number: *ncpu* of available logical CPUs may be set manually. For the considered installation of *ccx* on Mac OS X, however, the ENV: *NUMBER_OF_CPUS* is not required because *ccx* determines the number of available logical CPUs automatically (cf. also Section 2 in [2]).
- *OMP_NUM_THREADS=nthreads*. The ENV: *OMP_NUM_THREADS* invokes MT for both Spooles and *ccx*. The number: *nthreads* for the max number of threads may be set. MT for *ccx* may be controlled with further environment variables as explained in Section 2 in [2].
- *CCX_NPROC_EQUATION_SOLVER=nthreads*. The ENV: *CCX_NPROC_EQUATION_SOLVER* invokes MT for Spooles only. The number: *nthreads* for max number of threads may be set.
- *OMP_NUM_THREADS* and *CCX_NPROC_EQUATION_SOLVER*. If both of the latter ENVs are set, ENV: *CCX_NPROC_EQUATION_SOLVER* is relevant for MT. Accordingly, MT will be invoked for Spooles only as already explained above.

If in the above statements the number: *nthreads* is set larger than the number: *ncpu* of available logical CPUs, then *ccx* automatically resets *nthreads* to *ncpu*.

MT for Spooles only, therefore, may be invoked by setting the environment variable: *CCX_NPROC_EQUATION_SOLVER=nthreads* and MT for both Spooles and *ccx* by setting the variable: *OMP_NUM_THREADS=nthreads* (cf. also example below).

The ENVs may be set with the following Terminal **command**:

```
$ export <name environment variable>=nthreads <return>
```

with:

```
<name environment variable>= OMP_NUM_THREADS,  
<name environment variable>= CCX_NPROC_EQUATION_SOLVER,  
nthreads: max number of threads for MT.
```

Spooles provides output for the actually used number of threads (which may differ from *nthreads*) in file: “spooles.out” (cf. example below). For single-threading the file “spools.out” is empty.

- *Test of MT for ccx and Spooles*. MT for *ccx* and Spooles will be tested for the test example: “rotor” from the test-directory of *ccx*, a CPU with *ncpu* = 4 logical CPUs, *OMP_NUM_THREADS*=10 and, therefore, for *nthread* = 10 > *ncpu*=4. As already explained above, the ENV: *OMP_NUM_THREADS* invokes MT for both Spooles and *ccx* (to invoke MT for Spooles only use ENV: *CCX_NPROC_EQUATION_SOLVER*).

Run test with Terminal **commands** (without **comments**):

```
$ cd calculix_inst_path/claculix/ccx_2.9/test <return> # Set working directory test.
```

```
$ export OMP_NUM_THREADS=10 <return> # Set environment variable.
```

```
$ ccx_2.9 rotor <return> # Run example "rotor".
```

Extract of screen output:

```
Using up to 4 cpu(s) for spooles.
```

```
Using up to 4 cpu(s) for the stress calculation.
```

According to the screen output above up to 4 CPUs, respectively threads are used for MT of Spooles and ccx. The input: `nthread=10`, therefore, has been reset to `nthread=4=ncpu` and, therefore, (correct) to the number `ncpu=4` of available logical CPUs. Output for the number of threads, which Spooles actually has used for MT is provided in file: "spools.out". For the present example file: "spools.out" contains the output: "Using 4 threads". Spooles, therefore, actually has used 4 threads. The number of threads given in file: "spooles.out" may be less than `nthreads` (according to [1], actually used number of threads set by optimization routines which are implemented in Spooles). Furthermore, the above screen output reveals that ccx used up to 4 threads (CPUs) for the stress calculation. As already mentioned above, MT of ccx may be controlled with further environment variables as explained in Section 2 in [2].

If: `CCX_NPROC_EQUATION_SOLVER=10` is used for the aforementioned test example, then MT is invoked for Spooles only. For ccx, then single-threading is invoked (Terminal output now: "Using up to 1 cpu(s) for the stress calculation"), whereas MT for Spooles is the same as explained above for the usage of ENV: `OMP_NUM_THREADS=10`.

Tests with various other settings of the above specified environment variables also revealed that MT works.

9. HINTS AND MAC-SPECIFIC PROBLEMS

9.1 Hints

Section provided for later use.

9.2 Mac-Specific Problems

Section provided for later use.

REFERENCES

- [1] README.INSTALL. G. Dhondt, February 3, 2015. Short overview of installation of ccx 2.9 for Unix/Linux. Download: http://www.dhondt.de/ccx_2.9.README.INSTALL.
- [2] CalculiX CrunchiX USERS'S MANUAL version 2.9. G. Dhondt, November 19, 2015. Download of PDF: http://www.dhondt.de/ccx_2.9.pdf.
- [3] Getting Started With CalculiX. Tutorial. J. Baylor. Download of PDF: <http://www.bconverged.com/content/calculix/doc/GettingStarted.pdf>
- [4] Installation Guide – CalculiX ccx 2.9 Graphical Pre- and Postprocessor on Mac OS X El Capitan. B. Graf, November 30, 2015.