/*————————————————*/

MDOODZ6.0

Documentation

MDOODZ Developper Team

May 26, 2019

Contents

Installation

0.1 Prequisites

So far MDOODZ has been successfully built on LINUX/UNIX and MAC OS systems. The code can be built with GCC compiler from GNU (http://gcc.gnu.org) or with ICC compiler from Intel MKL library (https://software.intel.com/en-us/intel-mkl).

The code relies on two libraries:

1. SuiteSparse provides efficient linear algebra and matrix manipulation routines. It is available at: http://www.suitesparse.com

2. HDF5 is the main format for output files and is readable into MATLAB. It is available at: http://www.hdfgroup.org

0.2 Installation on Mac OS using MacPorts

All components that are required to build the code can be installed via the MacPorts platform on Mac OS by the following steps:

1. Download the MacPorts platform following the instructions on the webpage: https://www.macports.org
2. Install the required components by opening a terminal window and type
   a) `sudo port install git`
   b) `sudo port install gccX` (f.e., gcc7 for version 7)
   c) `sudo port install suitesparse`
   d) `sudo port install hdf5`
3. Make sure the installed gcc compiler is active
   a) `sudo port select --list gcc` (list of all installed versions of gcc)
   b) `sudo port select --set gcc mp-gccX` (activate gcc compiler version X)

Alternatively, one can use Homebrew instead of macports. In this case use the command `brew install` instead of `sudo`

For a successful build the user has to link libraries and set certain environmental variables in the bash script named: ".profile" (usually located at /Users/YOUR_USERNAME/)

If this file doesn't exist, one has to create it. The first line of .profile has to be:

```
\#!/usr/bin/env bash
```

Further, one has to set:

```
export HDF5\_USE\_FILE\_LOCKING='FALSE'
```

For the HDF5 libraries, to include the C path via:

```
export C\_INCLUDE\_PATH=/opt/local/include
```

And finally link the libraries:

```
export LIBRARY\_PATH=/opt/local/lib
```

Note that .profile is executed once each time a new terminal session is started, thus it is very useful to set frequently used commands, global variables etc. in this file to avoid finger pain.
If using Homebrew replace `/opt/local/` by `/usr/local/` in the above.
The user has installed all required components and linked all libraries correctly now. Prior to compilation, one needs to figure out which makefile to use (or to design your own makefile). Examples of makefiles valid for different systems are available in the Makefiles folder.
The user needs to copy the appropriate makefile into the source folder (../) and to rename it "makefile_XXXX" into simply "makefile". One has to be careful and set the compiler accordingly in the makefile like:

```
# Compiler
ifeq ($(GNU),yes)
         CC=/opt/local/bin/gcc-mp-7
#endif
```

To avoid changing the compiler in the makefile one can create a link between the command "gcc" and the desired version. For example, the makefile would contain the line

```
CC=gcc
```

Then in a terminal:

```
cd /opt/local/bin/
ln -s  gcc-mp-7  gcc
```

Then, when the makefile calls `gcc` it will be bumped back to `gcc-mp-7`. Note that `/opt/local/bin/` should be `/usr/local/bin/` when using homebrew. To verify that the links work type:

```
gcc --version
```

which should return something like

```
gcc (Homebrew GCC 9.1.0) 9.1.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

In this particular case the link goes to gcc 9.1.0 installed with homebrew.

0.3 Source files

The source consists of one ".h" file and a several ".c" files. The file "head.h"' contains the definition of data structures uses in the codes as well as function headers. The ".c" files contains all the functions of the codes. Key files are for example "Main_MDOODZ_GREAT.c" that contains the main function, "flow_laws.c" that contains the flow law database, and the file "set_XXXX.c" in which user-defined setups are constructed (XXXX is a user defined name). For compiling the source a "makefile" is provided. In order to run a simulation, an additional "XXXX.txt" that contains runtime parameters (solver options, material properties, key parameters) is also required.

0.4 Compilation

Compilation flags:

- MODEL_PATH corresponds to the path towards the directory containing the "set_XXXX.c".
- MODEL corresponds to the model name, as for "set_XXXX.c" and the "XXXX.txt" files.
- OPT: optimized built should be set to yes or no.
- OMP: OpenMP Parallelization should be set to yes or no.

Non-optimized build:

```
make clean all MODEL_PATH=BENCHMARKS MODEL=SubBenchCase1 OPT=no OMP=no
```

Optimized build:

```
make clean all MODEL_PATH=BENCHMARKS MODEL=SubBenchCase1 OPT=yes OMP=yes
```

Benchmarks

0.1 Subduction benchmark case 1, Schmeling et al. [2008]

Results after 50 time steps

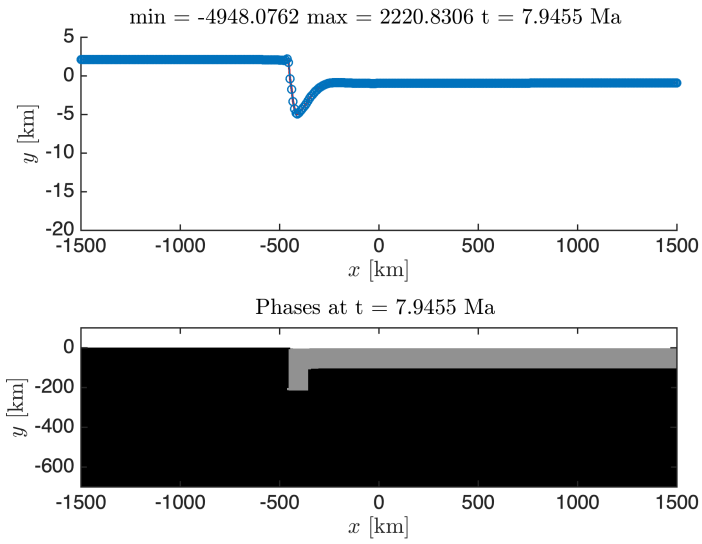

Figure 0.1: Results after 50 time steps.

0.2 Subduction benchmark case 3, Schmeling et al. [2008]

0.3 Compressible layer test, Moulas et al. [2018]

Need to use the following options:

```
compressible = 1
lsolver      = -1
```

The results of the simulation can be visualised with the file:

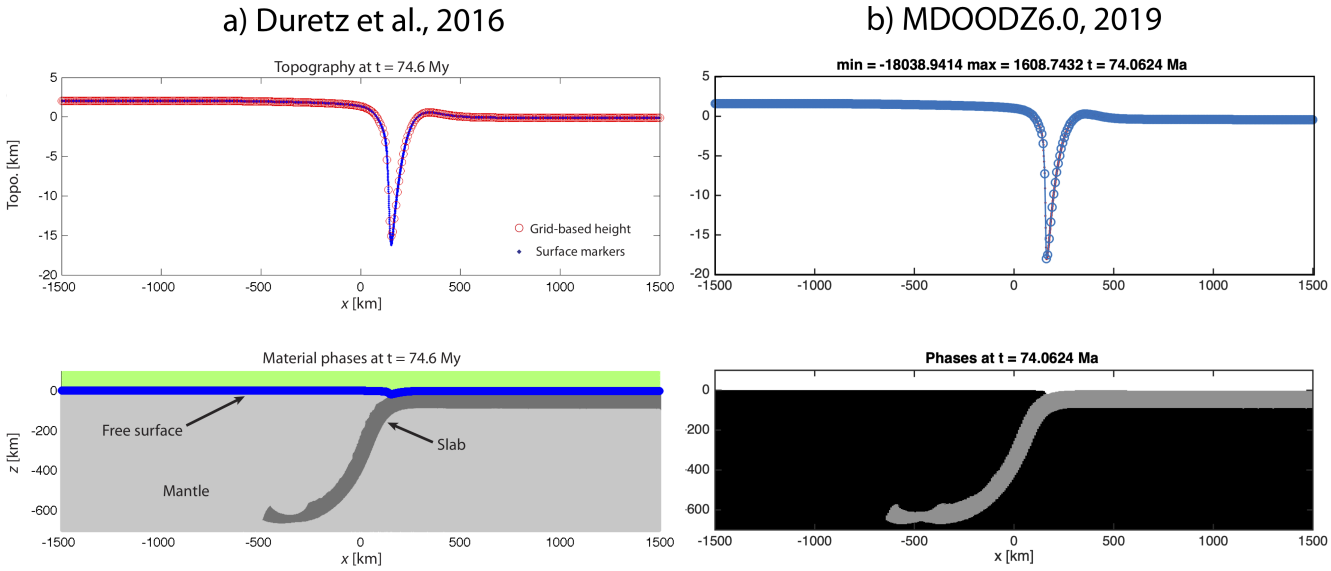

Figure 0.2: Comparison between results of Duretz et al. [2016] and current version of MDOODZ. After ~ 76 My, The minimum bathymetry is ~ −16 km and trench position is ~ 150 km.

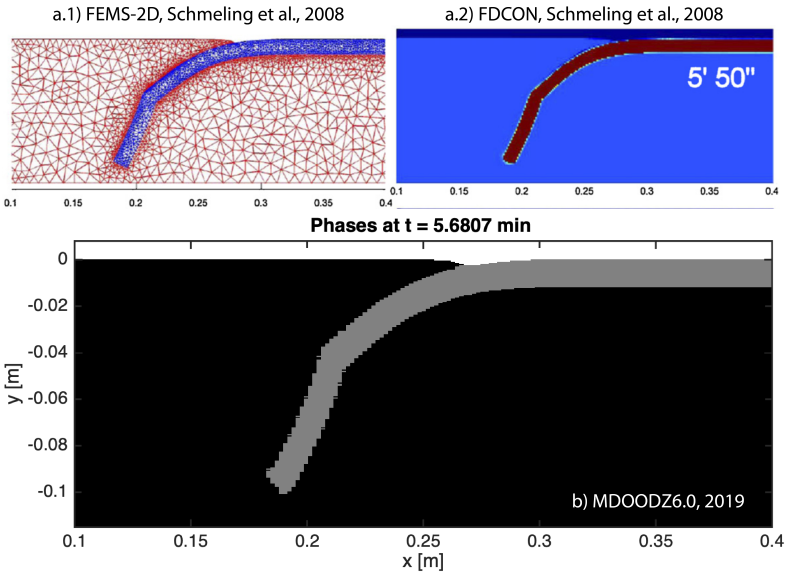

Figure 0.3: Comparison between results of Schmeling et al. [2008] and current version of MDOODZ. After ~ 5.7 min.

`ReadHDF5_VangelisTest.m`

The results can also be reproduced using the M2Di example:

`M2Di_CompressibleViscoElasticVangelis2_Beta.m`

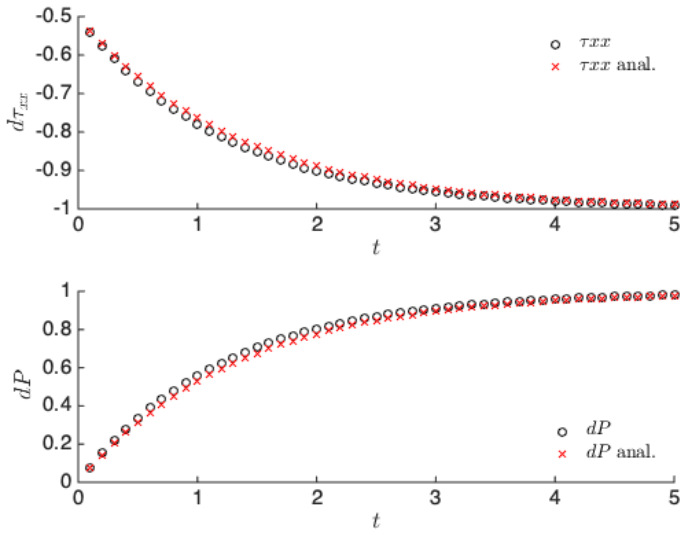

Figure 0.4: Top panel: Layer/matrix τ_{xx} difference. Bottom panel: Layer/matrix P difference.

0.4 Locally compressible models

Need to use the following options:

```
compressible = 1
lsolver      = -1
```

The results can also be reproduced using the M2Di example:

`M2Di_LocallyCompressibleViscoElastic_InclusionBeta.m`

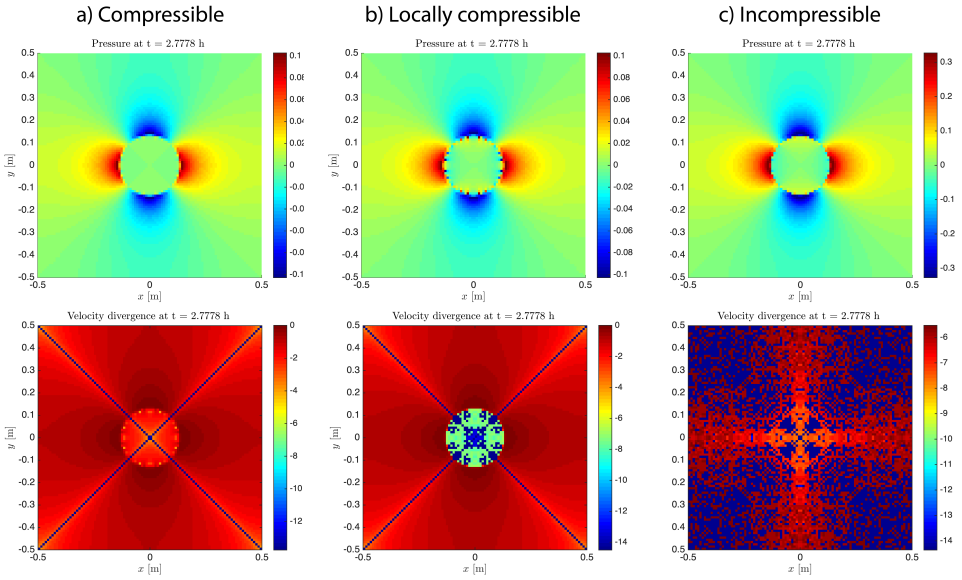

Figure 0.5: a) $\beta_{\text{matrix}} = \beta_{\text{inclusion}} = 10^5$. b) $\beta_{\text{matrix}} = 10^5$, $\beta_{\text{inclusion}} = 0.0$. c) $\beta_{\text{matrix}} = \beta_{\text{inclusion}} = 0.0$.

Matlab Examples

0.1 Finite Strain ellipsoid and tensor and vector rotation

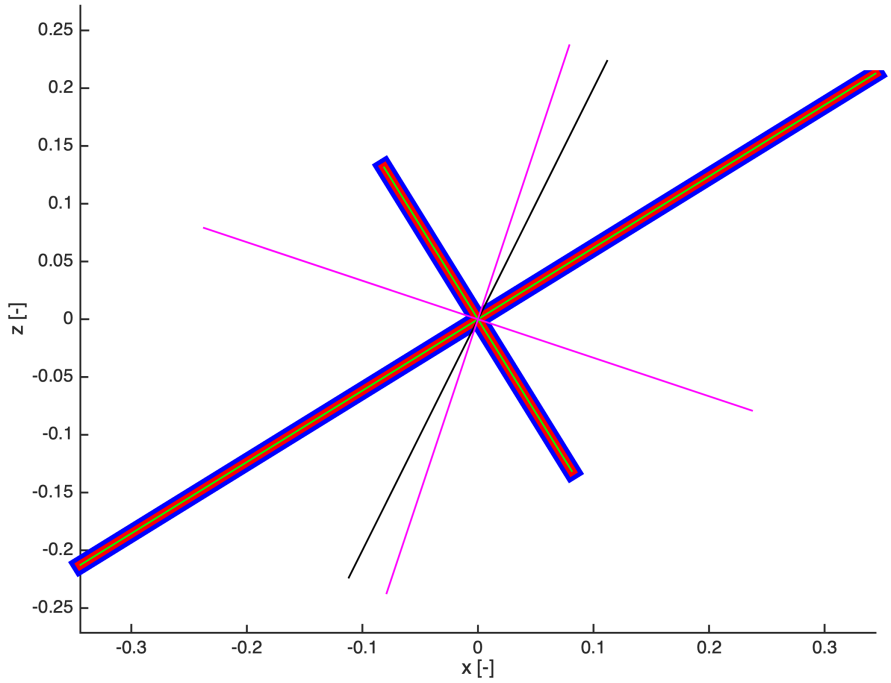

Figure 0.1: Simple shear deformation with γ = 1.0. Finite strain principal axis (thick lines). Rotated tensor principal axis (pink lines) originally oriented at 45 degrees. Rotated vector (black line) originally vertical.

Python Code Generation

Bibliography

T. Duretz, D.A. May, and P. Yamato. A free surface capturing discretization for the staggered grid finite difference scheme. *Geophysical Journal International*, 204(3): 1518–1530, 2016. doi: 10.1093/gji/ggv526.

Evangelos Moulas, Stefan M. Schmalholz, Yury Podladchikov, Lucie Tajčmanová, Dimitrios Kostopoulos, and Lukas Baumgartner. Relation between mean stress, thermodynamic and lithostatic pressure. *Journal of Metamorphic Geology*, 0(ja), 2018. doi: 10.1111/jmg.12446.

H. Schmeling, A. Y. Babeyko, A. Enns, C. Faccenna, F. Funiciello, T. V. Gerya, G. J. Golabek, S. Grigull, B .J. P. Kaus, G. Morra, S. M. Schmalholz, and J. van Hunen. A benchmark comparison of spontaneous subduction models - Towards a free surface. *Physics of the Earth and Planetary Interiors*, 171(1-4):198–223, 2008.