/*————————————————*/

MDOODZ6.0

Documentation

MDOODZ Developper Team

April 26, 2021

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, ?

Results after 50 time steps

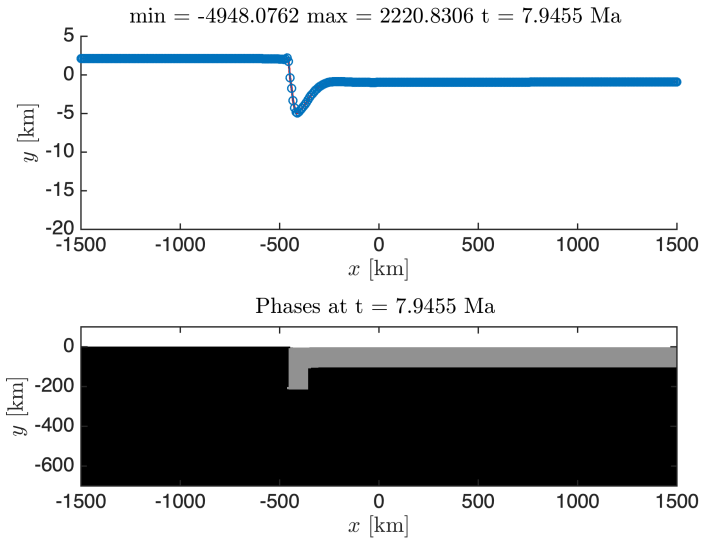

Figure 0.1: Results after 50 time steps.

0.2 Subduction benchmark case 3, ?

0.3 Compressible layer test, ?

Need to use the following options:

```
compressible = 1
```

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

`ReadHDF5_VangelisTest.m`

The results can also be reproduced using the M2Di example:

`M2Di_CompressibleViscoElasticVangelis2_Beta.m`

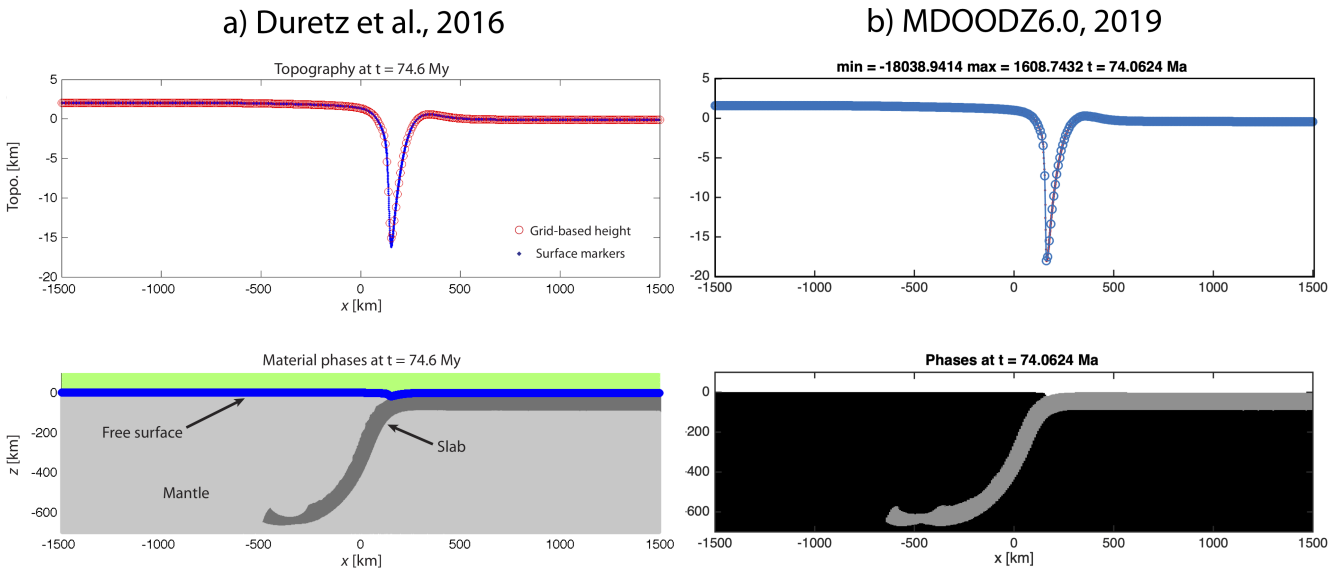

Figure 0.2: Comparison between results of ? and current version of MDOODZ. After ~ 76 My, The minimum bathymetry is ~ −16 km and trench position is ~ 150 km.

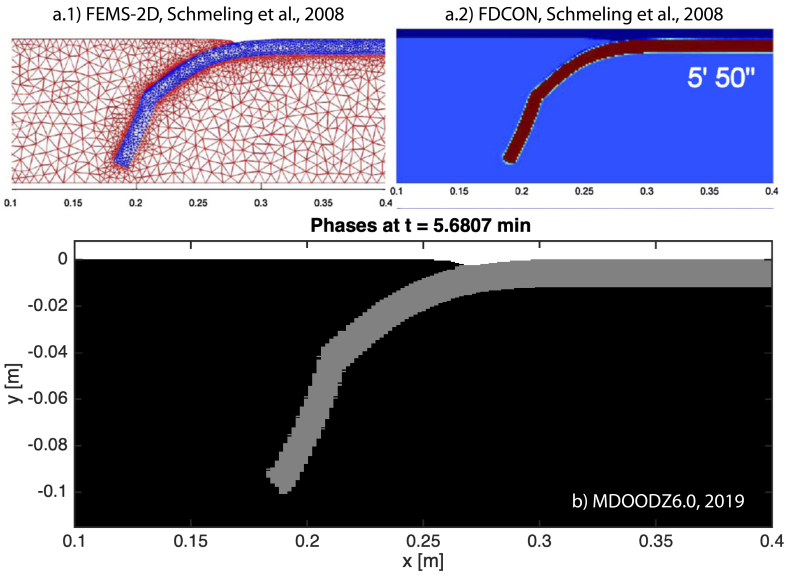

Figure 0.3: Comparison between results of ? and current version of MDOODZ. After ~ 5.7 min.

0.4 Locally compressible models

Need to use the following options:

```
compressible = 1
```

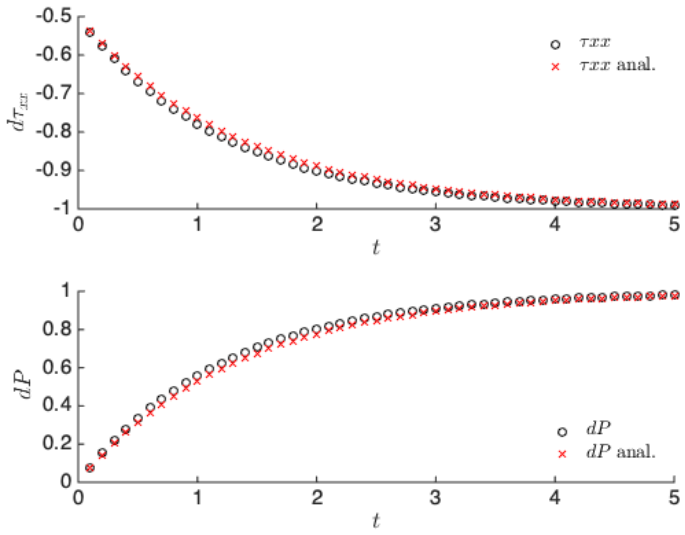

Figure 0.4: Top panel: Layer/matrix τ_{xx} difference. Bottom panel: Layer/matrix P difference.

The results can also be reproduced using the M2Di example:

`M2Di_LocallyCompressibleViscoElastic_InclusionBeta.m`

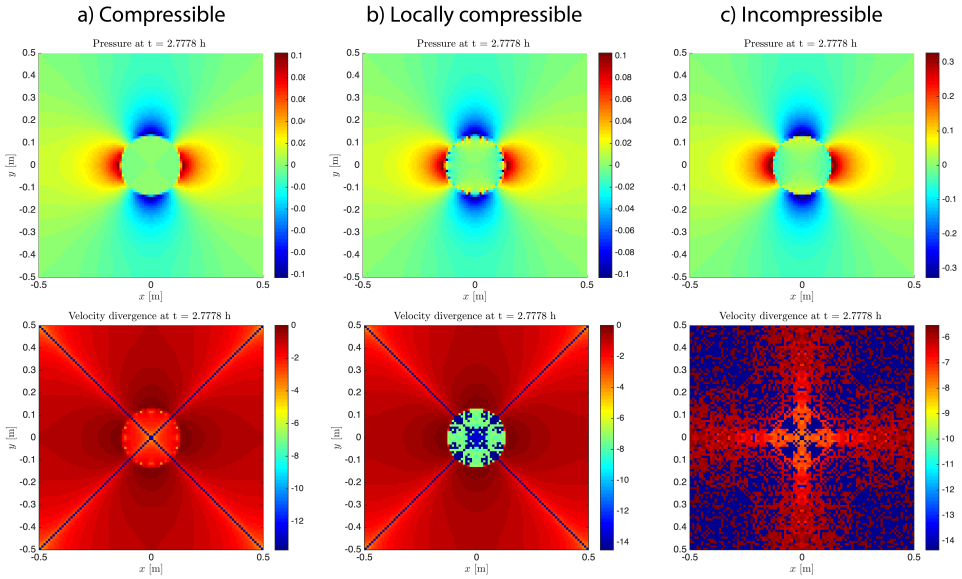

Figure 0.5: a) $\beta_{\text{matrix}} = \beta_{\text{inclusion}} = 10^5$. b) $\beta_{\text{matrix}} = 10^5$, $\beta_{\text{inclusion}} = 0.0$. c) $\beta_{\text{matrix}} = \beta_{\text{inclusion}} = 0.0$.

0.5 Periodic models with linear rheology

setup file:

`Inc2Boules.txt`

Need to use the following options:

```
isperiodic_x = 1
```

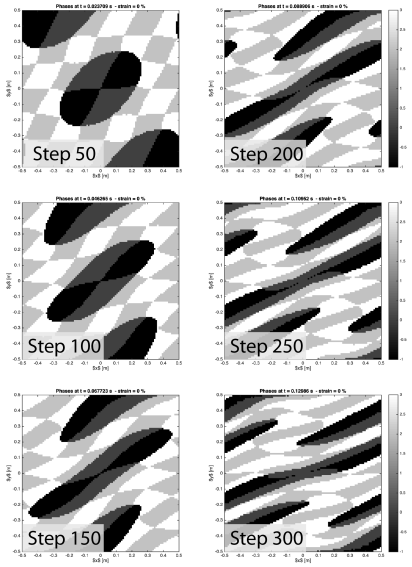

Figure 0.6: Evolution of the composition field in time.

0.6 Visco-elasticity with linear rheology: "Viagra test"

setup file:

`VE_rebound.txt`

```
StressRotation = 0 (No stress rotation) \\
StressRotation = 1 (Analytical stress rotation) \\
StressRotation = 2 (Stress rotation and finite strain: Upper convected rate)
```

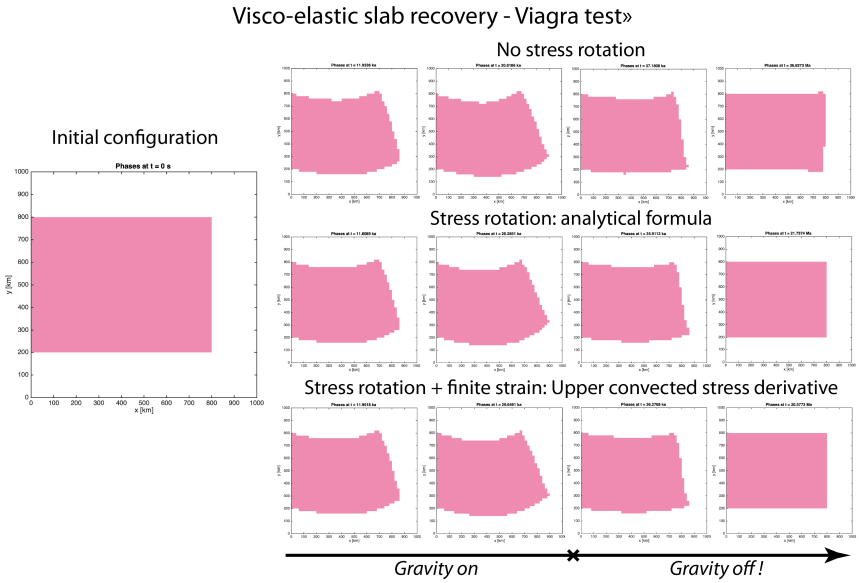

Figure 0.7: Recovery of an elastic slab

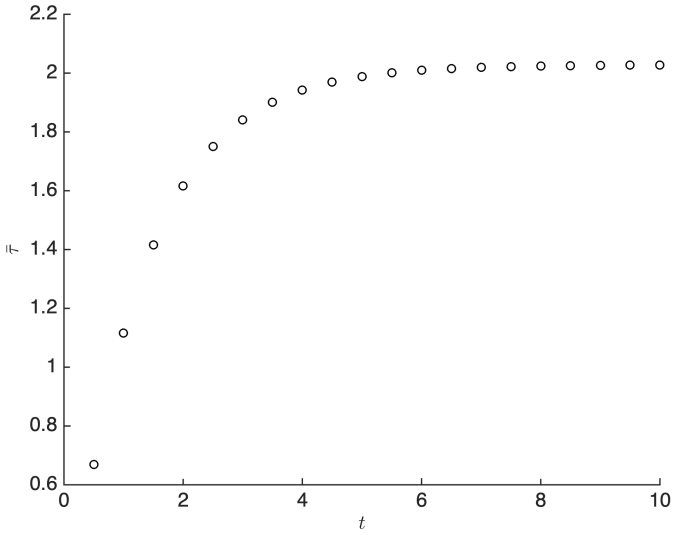

Figure 0.8: Evolution of effective stress.

0.7 Visco-elasticity with linear rheology: Pure shear

setup file:

`PureShear_VE.txt`

Linear rheology with viscosity contrast of 10 (weak inclusion). Same shear modulus in both the matrix and the inclusion.
The results can also be reproduced using the M2Di example:

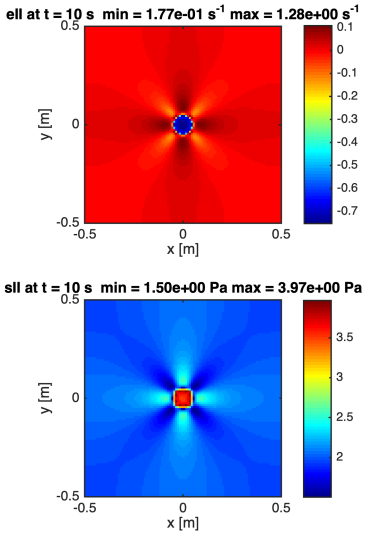

Figure 0.9: Color maps of strain rate ($\log_{10}$) and stress.

`M2Di2_LinearVE_Newton_PureShear_MDOODZ.m`

0.8 Power-law rheology: Pure shear

setup file:

`PureShear_pwl.txt`

Power-law exponent of matrix is 30, inclusion is 3. Newton solver converges in 7 nonlinear iterations. No time dependence, only one time step yields.
The results can also be reproduced using the M2Di example:

`M2Di2_PowerLaw_Newton_PureShear_MDOODZ.m`

0.9 Visco-elasticity with power-law rheology: Pure and simple shear

setup file:

`Shear_VEVP.txt`

Time dependence, therefore stress builds up to eventually yield a comparable result than in the viscous end-member (section above). The model can be run in either pure shear if

`shear_style = 0`

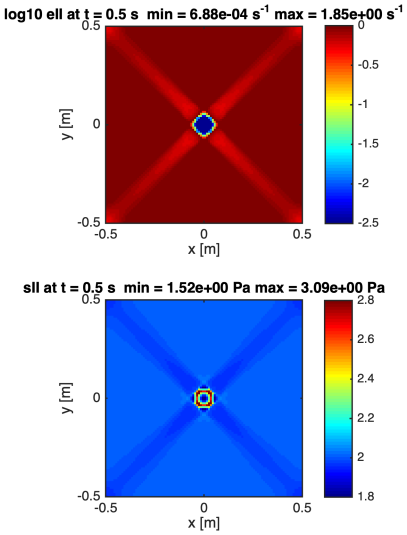

Figure 0.10: Color maps of strain rate ($\log_{10}$) and stress.

or periodic simple shear if

```
shear_style = 1
```

Turn off plasticity by setting, for each phase, a very large cohesion

```
C = 2.35e10
```

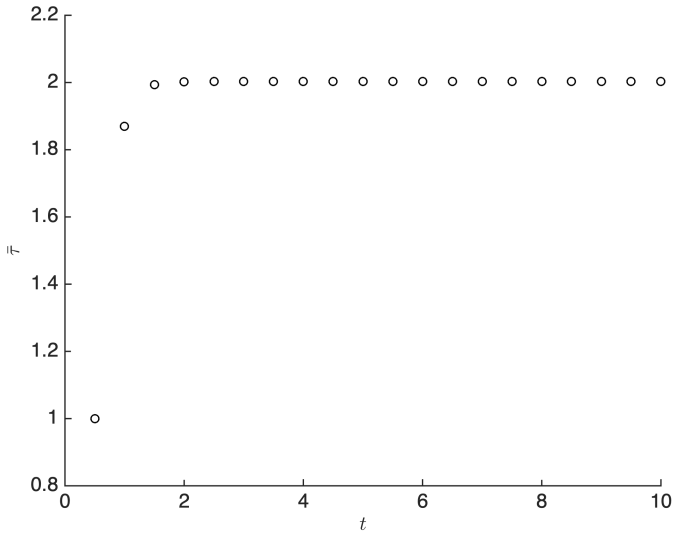

Figure 0.11: Evolution of effective stress.

Pure shear model results can also be reproduced using the M2Di example:

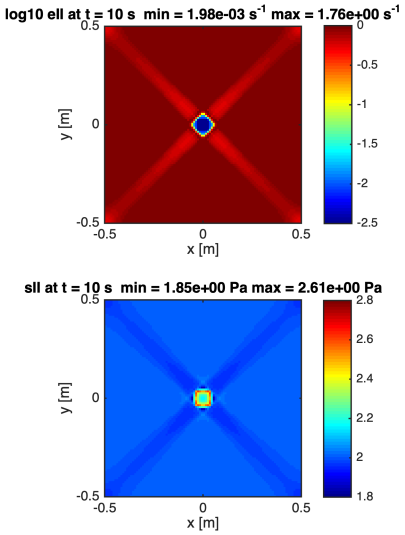

Figure 0.12: Color maps of strain rate ($\log_{10}$) and stress for pure shear.

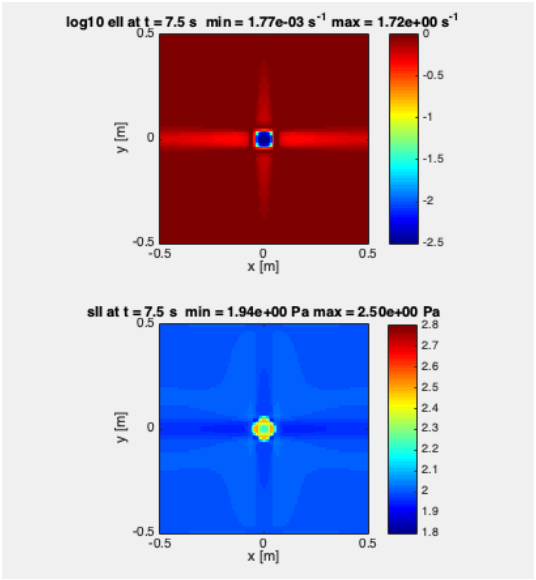

Figure 0.13: Color maps of strain rate ($\log_{10}$) and stress for periodic simple shear.

`M2Di2_PowerLawVE_Newton_PureShear_LocIt_MDOODZ.m`

0.10 Visco-elasticity-viscoplasticity with power-law rheology: Pure and simple shear

setup file:

`Shear_VEVP.txt`

Time dependence, therefore stress builds up to eventually yield a comparable result than in the viscous end-member (section above). The model can be run in either pure shear if

```
shear_style = 0
```

or periodic simple shear if

```
shear_style = 1
```

Turn on plasticity by setting, for each phase,

```
C = 2.35
```

Adjust regularisation viscosity of each phase with

```
eta_vp = 1e-1
```

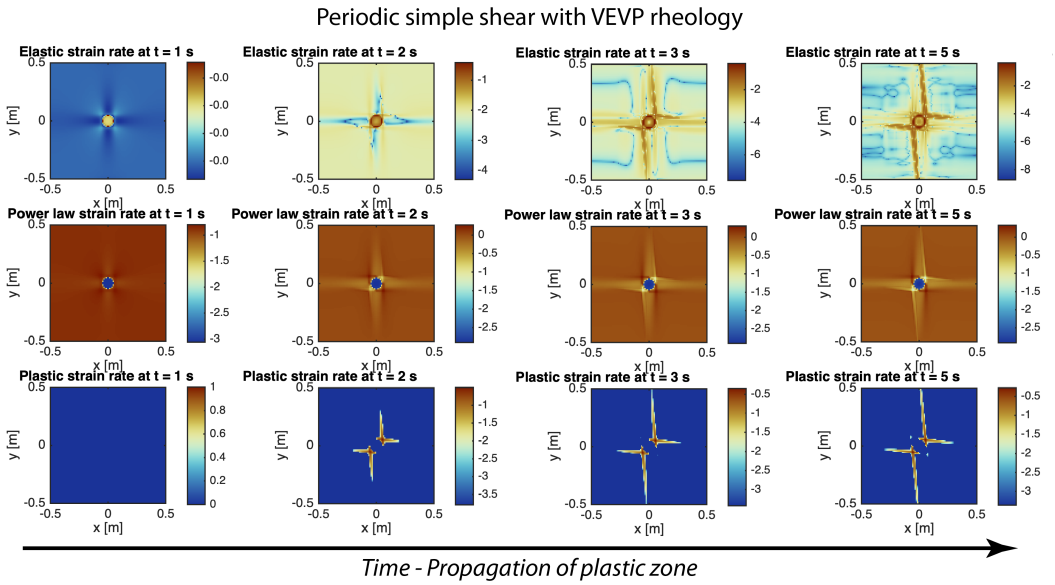

Figure 0.14: Evolution of strain rate components under simple shear.

0.11 Ideal elasticity-plasticity: Duretz et al., (2018) benchmark test 1

Setup file:

`VEP_Duretz18.txt`

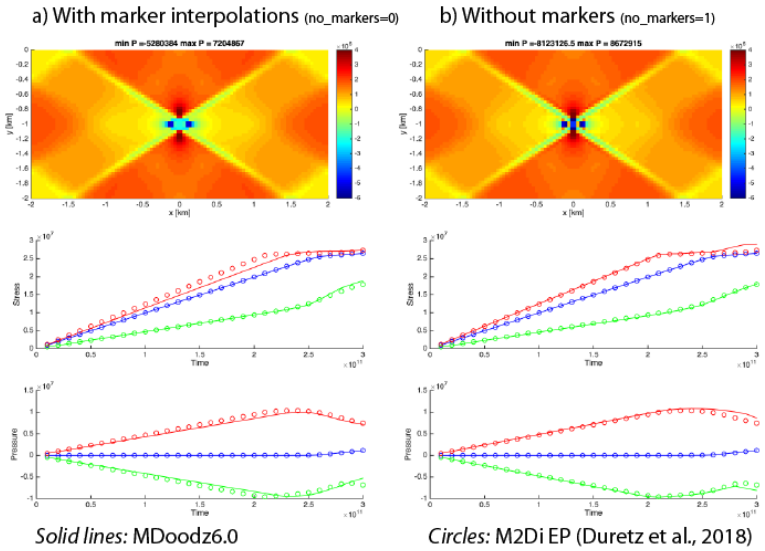

Figure 0.15: Colourmap: Pressure field after 30 time steps. Models were run with and without marker interpolations.

0.12 Visco-elasticity-viscoplasticity with power-law rheology: Crustal-scale model

Setup file:

`CrustalShearBanding_GRL.txt`

Adjust regularisation viscosity of each phase with

```
eta_vp = 1e21
```

Compression:

```
EpsBG<0
```

and extension

```
EpsBG>0
```

0.13 Visco-elasticity-viscoplasticity with power-law rheology: Lithosphere-scale model

Setup file:

`LithoScale.txt`

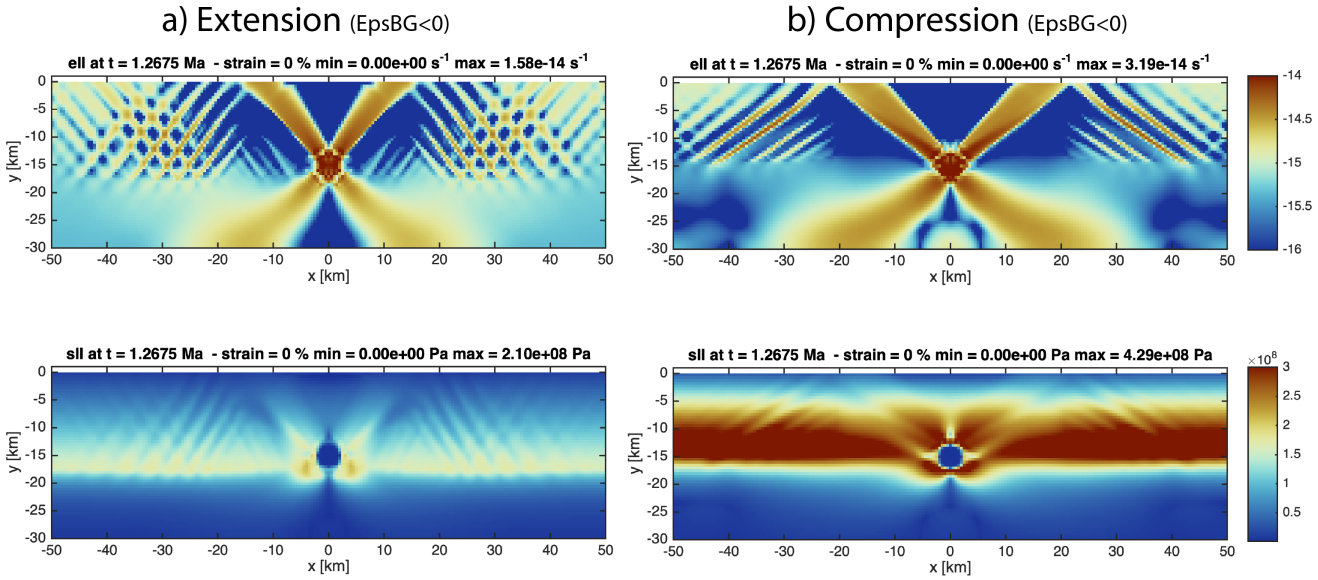

Figure 0.16: Strain rate and stress after 100 time steps in extension and compression.

Adjust regularisation viscosity of each phase with

```
eta_vp = 2e20
```

From now on conservative velocity interpolation scheme can be activated.

```
ConservInterp = 1
```

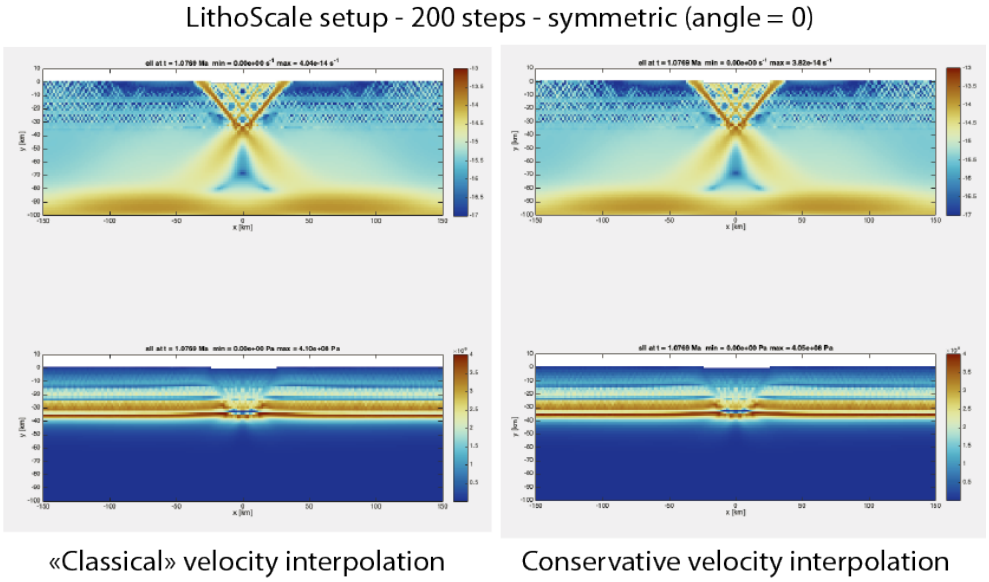

Figure 0.17: Strain rate and stress after 200 time steps with classical and conservative velocity interpolation schemes.

0.14 The rifting model of Chenin et al. (2020)

No viscoplasticity activated - therefore no global convergence to be expected.
Setup file:

`RiftingPaulineMD6.txt`

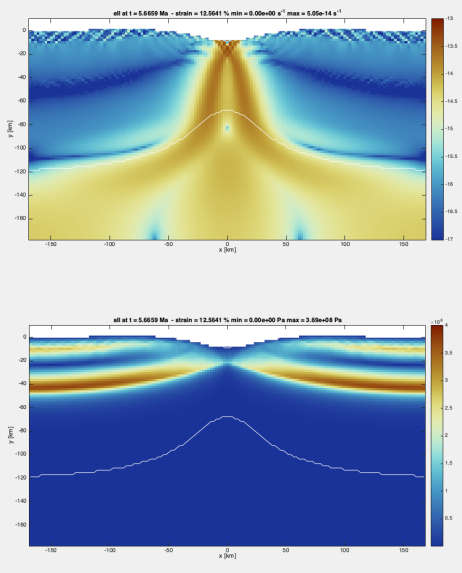

Figure 0.18: Strain rate and stress after 100 time steps.

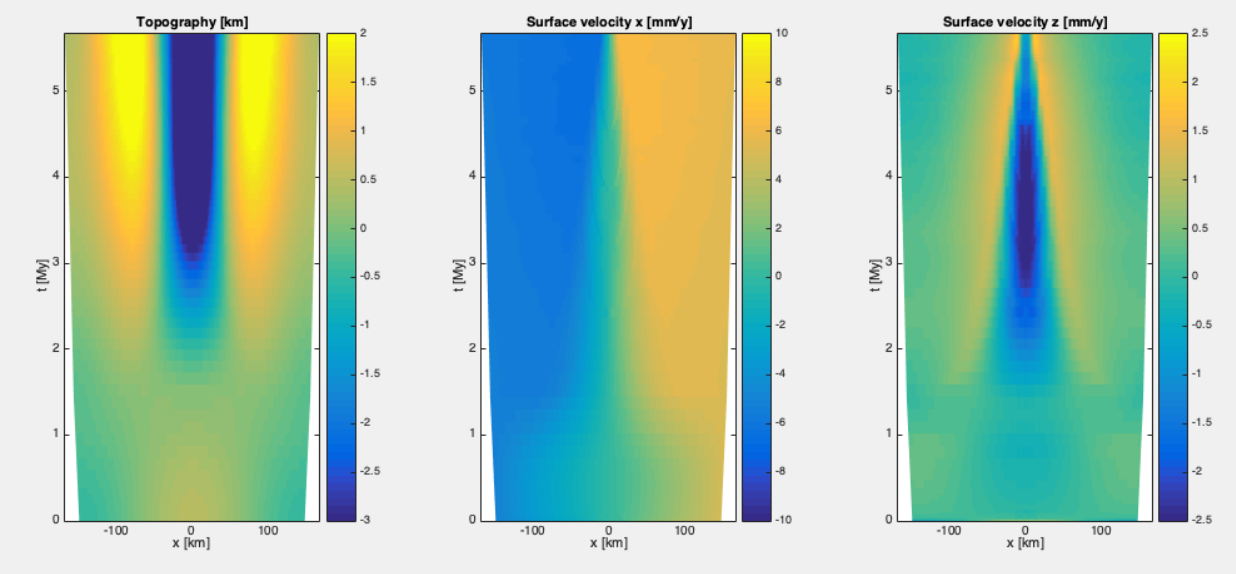

Figure 0.19: Topography and surface velocity after 100 time steps.

Matlab Examples

0.1 Finite Strain ellipsoid and tensor and vector rotation

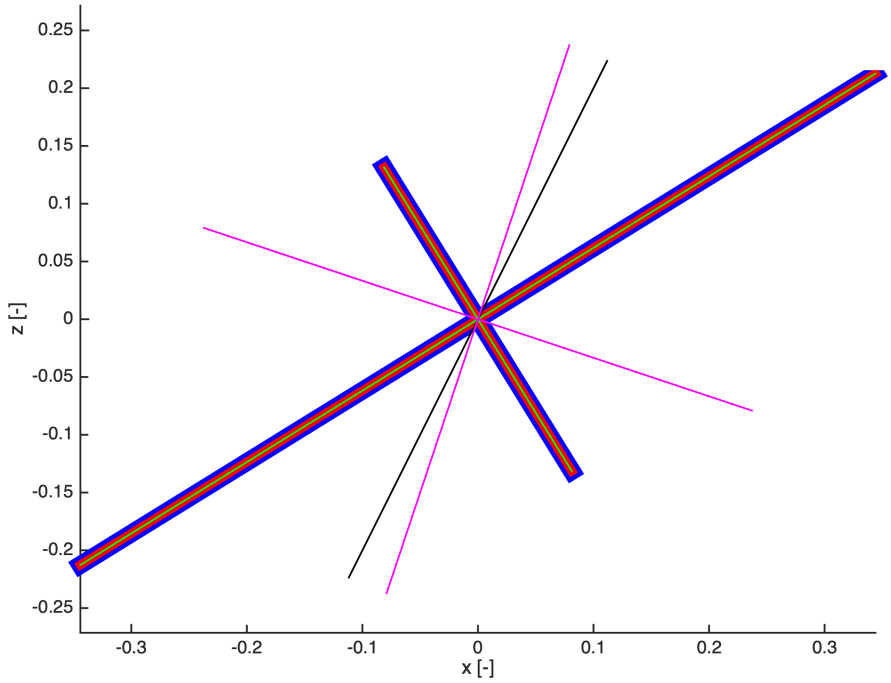

Figure 0.1: Simple shear deformation with γ = 1.0. Finite strain principal axis (thick lines). Rotated tensor principal axis (pink lines) originally oriented at 45 degrees. Rotated vector (black line) originally vertical.

Python Code Generation