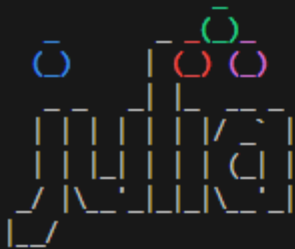


LaMEM short course

17-21 02 2025 Heidelberg
Nicolas Riel- nriel@uni-mainz.de

How to create a LaMEM model?

```
seph42@DESKTOP-2V82075:~/LaMEM_course/01_falling_block_isoviscous$ julia
```



Documentation: <https://docs.julialang.org>

Type "?" for help, "]"? for Pkg help.

Version 1.10.0 (2023-12-25)

Official <https://julialang.org/> release

```
julia> using LaMEM
```

```
julia> Model()
```

LaMEM Model setup

```
|
|-- Scaling           : GeoParams.Units.GeoUnits{GeoParams.Units.GEO}
|-- Grid              : nel=(16, 16, 16); x∈(-10.0, 10.0), y∈(-5.0, 5.0), z∈(-10.0, 0.0)
|-- Time              : nstep_max=50; nstep_out=1; time_end=1.0; dt=0.05
|-- Boundary conditions : noslip=[0, 0, 0, 0, 0, 0]
|-- Solution parameters : eta_min=1.0e18; eta_max=1.0e25; eta_ref=1.0e20; act_temp_diff=0
|-- Solver options      : direct solver; superlu_dist; penalty term=10000.0
|-- Model setup options : Type=files;
|-- Output options      : filename=output; pvd=1; avd=0; surf=0
|-- Materials          : 0 phases;
```

```
julia> []
```

e.g. BoundaryConditions

Access information by typing:

```
julia> BoundaryConditions()
LaMEM Boundary conditions :
  noslip           = [0, 0, 0, 0, 0, 0]
  open_top_bound   = 0
  temp_top         = 0.0
  temp_bot         = 1300.0
  exx_num_periods  = 3
  exx_time_delims  = [0.1, 5.0]
  exx_strain_rates = [1.0e-15, 2.0e-15, 1.0e-15]
  eyy_num_periods  = 2
  eyy_time_delims  = [1.0]
  eyy_strain_rates = [1.0e-15, 2.0e-15]
  exy_num_periods  = 2
  exy_time_delims  = [1.0]
  exy_strain_rates = [1.0e-15, 2.0e-15]
  exz_num_periods  = 2
  exz_time_delims  = [1.0]
  exz_strain_rates = [1.0e-15, 2.0e-15]
  eyz_num_periods  = 2
  eyz_time_delims  = [1.0]
  eyz_strain_rates = [1.0e-15, 2.0e-15]
  bg_ref_point     = [0.0, 0.0, 0.0]
  VelocityBoxes    = VelocityBox[]
```

There is a bunch of available parameters, but what do they mean?

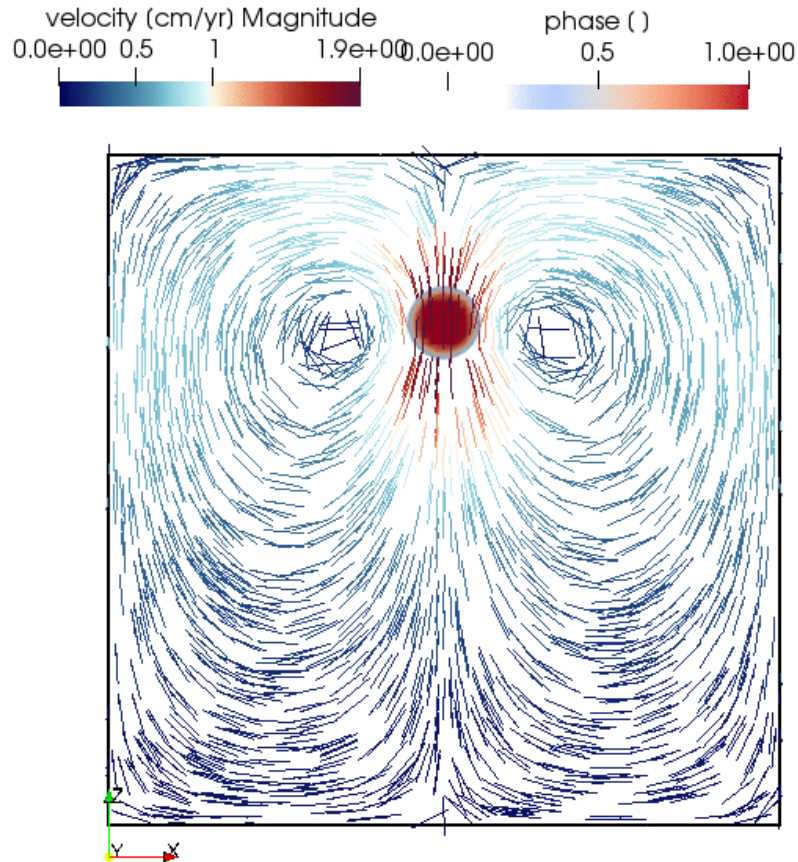
e.g. BoundaryConditions

```
help>> BoundaryConditions  
search: BoundaryConditions
```

Structure that contains the LaMEM boundary conditions information.

- `noslip::Vector{Int64}`: No-slip boundary flag mask (left right front back bottom top)
- `open_top_bound::Int64`: Stress-free (free surface/ininitely fast erosion) top boundary flag
- `temp_top::Float64`: Constant temperature on the top boundary
- `temp_bot::Float64`: Constant temperature on the bottom boundary
- `exx_num_periods::Int64`: number intervals of constant background strain rate (x-axis)
- `exx_time_delims::Vector{Float64}`: time delimiters (one less than number of intervals, not required for one interval)
- `exx_strain_rates::Vector{Float64}`: strain rates for each interval
- `eyy_num_periods::Int64`: `eyynumperiods`
- `eyy_time_delims::Vector{Float64}`: `eyytimedelims`
- `eyy_strain_rates::Vector{Float64}`: `eyystrainrates`
- `exy_num_periods::Int64`: `exynumperiods`
- `exy_time_delims::Vector{Float64}`: `exytimedelims`
- `exy_strain_rates::Vector{Float64}`: `exystrainrates`
- `exz_num_periods::Int64`: `exznumperiods`
- `exz_time_delims::Vector{Float64}`: `exztimedelims`
- `exz_strain_rates::Vector{Float64}`: `exzstrainrates`
- `eyz_num_periods::Int64`: `eyznumperiods`
- `eyz_time_delims::Vector{Float64}`: `eyztimedelims`
- `eyz_strain_rates::Vector{Float64}`: `eyzstrainrates`
- `bg_ref_point::Vector{Float64}`: background strain rate reference point (fixed)
- `VelocityBoxes::Vector{VelocityBox}`: List of added velocity boxes

Falling sphere: introduction



Directory: 01_falling_block_iso_viscous

- 01_falling_block_iso_viscous.jl
- 01_falling_block_iso_viscous.gif
- 01_falling_block_iso_viscous.png
- 01_falling_block_viscous_PVstate.pvsm

Model setup: overview

01_falling_block_iso_viscous.jl

PART1: package and output directory

Bunch of comments useful to keep track of what does the script

List the packages used for the simulation

Output_directory
(relative to REPL position)

```
1 #####
2 # 01_falling_block_iso_viscous NR.02-24
3 # simple setup to simulate a dense viscous sphere falling down a less
4 #
5 #
6 # if you have blue wavy underline below the model definition lines add
7 #####
8
9
10 # load needed packages, GeophysicalModelGenerator is used to create sh
11 using LaMEM, GeophysicalModelGenerator, GMT, Plots
12
13
14 # directory you want your simulation's output to be saved in
15 out_dir = "output"
16
```

Model setup: overview

PART2: model setup

Structure holding model infos

Model dimensionalization

Model size and resolution

Thermal and mechanical
Boundary conditions

Time and timestep definition

Other model parameters

Outputted variables

Solver options

```
18 model = Model(  
19     # Scaling parameters, this ensure non-dimensionalisation in LaM  
20     Scaling(GEO_units( temperature = 1000,  
21                     stress      = 1e9Pa,  
22                     length     = 1km,  
23                     viscosity   = 1e20Pa*s) ),  
24     # This is where you setup the size of your model (as km as set  
25     Grid( x = [-50.0,50.0],  
26           y = [-1.0,1.0],  
27           z = [-100.0,0.0],  
28           nel = (96,1,96) ),  
29     # sets the conditions at the walls of the modelled domain  
30     BoundaryConditions( temp_bot = 20.0,  
31                       temp_top = 20.0,  
32                       open_top_bound = 0,  
33                       noslip = [0, 0, 0, 0, 0, 0]),  
34     # set timestepping parameters  
35     Time( time_end = 10.0,  
36           dt = 0.01,  
37           dt_min = 0.000001,  
38           dt_max = 0.1,  
39           nstep_max = 80,  
40           nstep_out = 1 ),  
41     # set solution parameters  
42     SolutionParams( eta_min = 1e19,  
43                   eta_ref = 1e20,  
44                   eta_max = 1e22),  
45     # what will be saved in the output of the simulation  
46     Output( out_density = 1,  
47            out_melt_fraction = 1,  
48            out_j2_strain_rate = 1,  
49            out_temperature = 1,  
50            out_surf_velocity = 1,  
51            out_dir = out_dir ),  
52     # here we define the options for the solver, it is advised to  
53     Solver( SolverType = "direct",  
54            DirectSolver = "mumps" )  
55 )
```

Model setup: overview

PART3: materials properties (shapes and rheologies)

Set background phase

Add ellipsoid shape

Creates mantle phase

Creates eclogite phase

Add phases to the model

```
66 ##### define phases (different materials) of the model #####
67
68 model.Grid.Phases                                     .= 0;           # here we first define the bac
69
70 add_ellipsoid!(model, cen                             = (0,0,-25),      # defines centre of the spher
71                               axes                     = (5,5,5),        # define size of the sphere
72                               StrikeAngle              = 0,
73                               DipAngle                 = 0,
74                               phase                    = ConstantPhase(1), # we attribute phase id = 1, n
75                               T                        = ConstantTemp(20));
76
77
78 ##### define material properties of the phases #####
79
80 mantle = Phase(                                         Name       = "Mantle",      # let's call phase 0 mantle
81                               ID                       = 0,              # not that ID here points to p
82                               rho                      = 3300,          # set mantle density
83                               eta                      = 1e20,          # set mantle viscosity
84                               G                       = 5e10 );          # set elastic modulii
85
86 eclogite = Phase(                                       Name       = "eclogite",
87                               ID                       = 1,
88                               rho                      = 4000,
89                               eta                      = 1e24,
90                               G                       = 5e10 );
91
92 add_phase!( model, mantle, eclogite )                  # this adds the phases
93
94 plot_cross_section(model, y=0, field=:phase)
95 savefig("01b_falling_block_isoviscous_with_tracers.png")
96 ##### perform simulation #####
97
98 run_lamem(model, 2)
```


Model setup: overview

PART3: materials properties (shapes and rheologies)

Set background phase

Add ellipsoid shape

Creates mantle phase

Creates eclogite phase

Add phases to the model

Saves a cross-section
through your setup

Perform simulation
using 2 cores

```
66 #===== define phases (different materials) of the model =====#
67
68 model.Grid.Phases                                     .= 0;           # here we first define the bac
69
70 add_ellipsoid!(model, cen                             = (0,0,-25),      # defines centre of the spher
71                               axes                     = (5,5,5),        # define size of the sphere
72                               StrikeAngle              = 0,
73                               DipAngle                 = 0,
74                               phase                    = ConstantPhase(1),  # we attribute phase id = 1, n
75                               T                        = ConstantTemp(20));
76
77
78 #===== define material properties of the phases =====#
79
80 mantle = Phase( Name      = "Mantle",           # let's call phase 0 mantle
81                 ID        = 0,                  # not that ID here points to p
82                 rho        = 3300,              # set mantle density
83                 eta         = 1e20,             # set mantle viscosity
84                 G           = 5e10 );            # set elastic moduli
85
86 eclogite = Phase( Name      = "eclogite",
87                  ID        = 1,
88                  rho        = 4000,
89                  eta         = 1e24,
90                  G           = 5e10 );
91
92 add_phase!( model, mantle, eclogite )           # this adds the phases
93
94 plot_cross_section(model, y=0, field=:phase)
95 savefig("01b_falling_block_isoviscous_with_tracers.png")
96 #===== perform simulation =====#
97
98 run_lamem(model, 2)
```

Simulation

- Execute the script “01_falling_block_iso_viscous.jl”

option1

Copy and paste
In the julia REPL

```
seph42@DESKTOP-2V82075:~/LaMEM_course/01_falling_block_iso_viscous$ julia

Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.
Version 1.10.0 (2023-12-25)
Official https://julialang.org/ release

julia> # load needed packages, GeophysicalModelGenerator is used to create shapes
model before running the simulation
using LaMEM, GeophysicalModelGenerator, GMT, Plots
```

option2

Use include()

```
seph42@DESKTOP-2V82075:~/LaMEM_course/01_falling_block_iso_viscous$ julia
^[[A

Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.
Version 1.10.0 (2023-12-25)
Official https://julialang.org/ release

julia> include("01_falling_block_iso_viscous.jl")
Loading GMT routines within GMG
WARNING: using GMT.meshgrid in module GeophysicalModelGenerator conflicts with an existing identifier.
Adding Plots.jl plotting extensions for LaMEM
```

option3

Execute the script
In the terminal

```
seph42@DESKTOP-2V82075:~/LaMEM_course/01_falling_block_iso_viscous$ julia 01_falling_block_iso_viscous.jl
Loading GMT routines within GMG
WARNING: using GMT.meshgrid in module GeophysicalModelGenerator conflicts with an existing identifier.
Adding Plots.jl plotting extensions for LaMEM
```

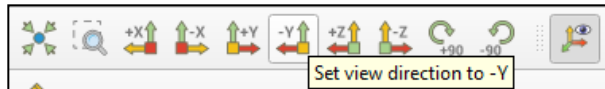
Visualization

- Use Paraview to visualize the results of the simulation and get a similar result:

Tips:

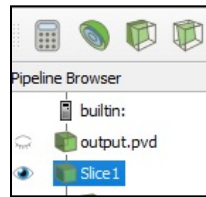
1

Set view direction to -Y



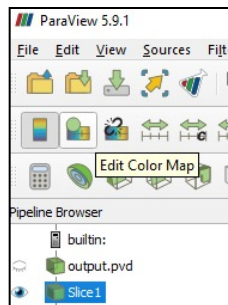
2

Slice
"output.pvd"
Y normal



3

Display
velocity field
on the slice
and change
colormap

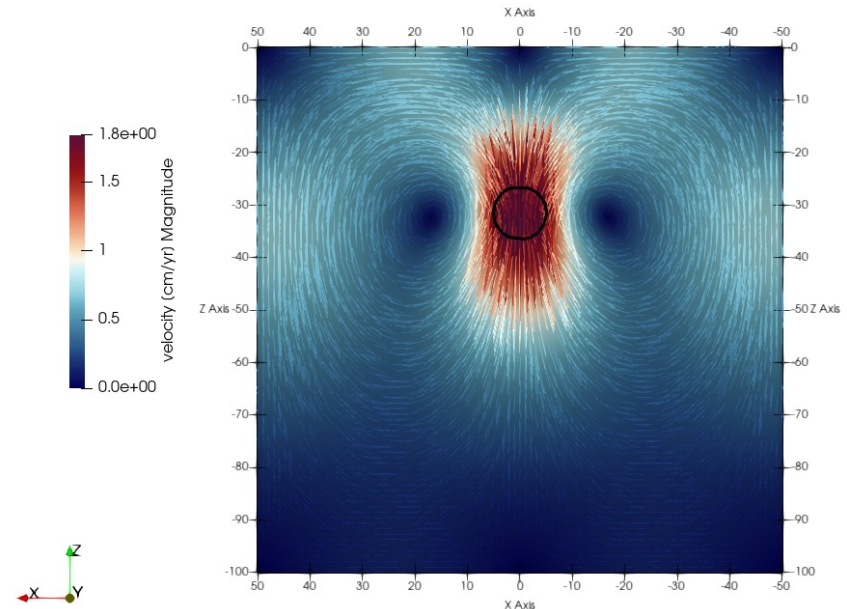
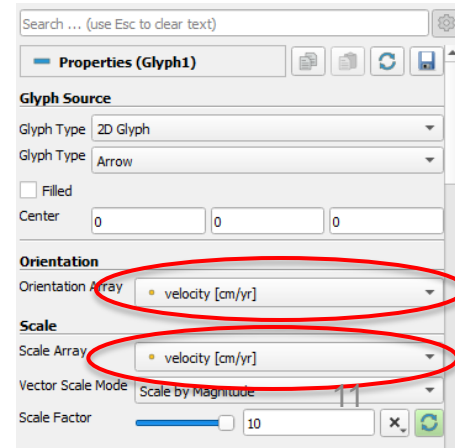


4

Contour phase
(select slice)

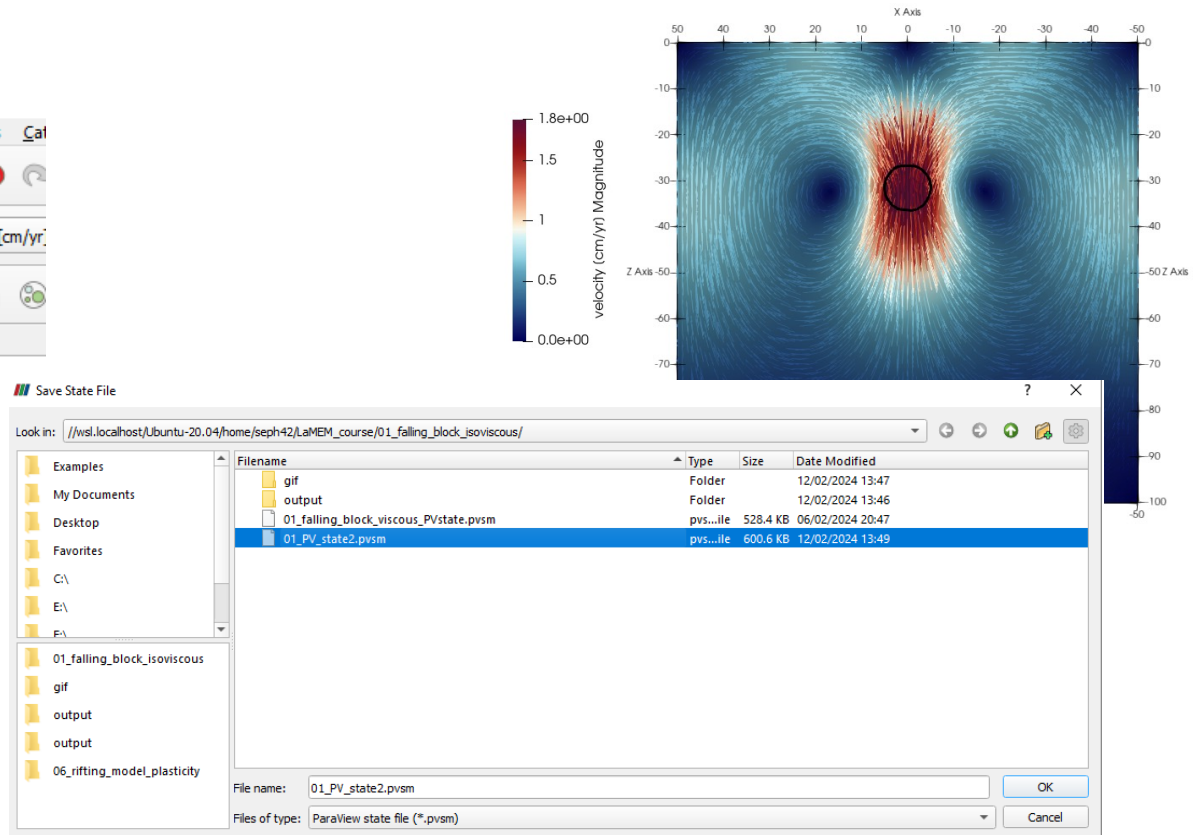
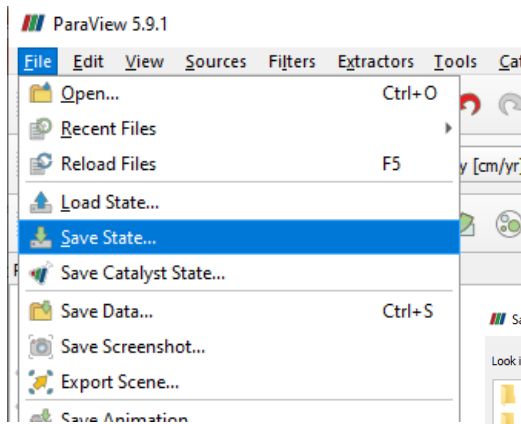
5

Add glyph
use velocity



Visualization - save Paraview state

- Saving “Paraview state” allow to re-use it for other (similar) simulation

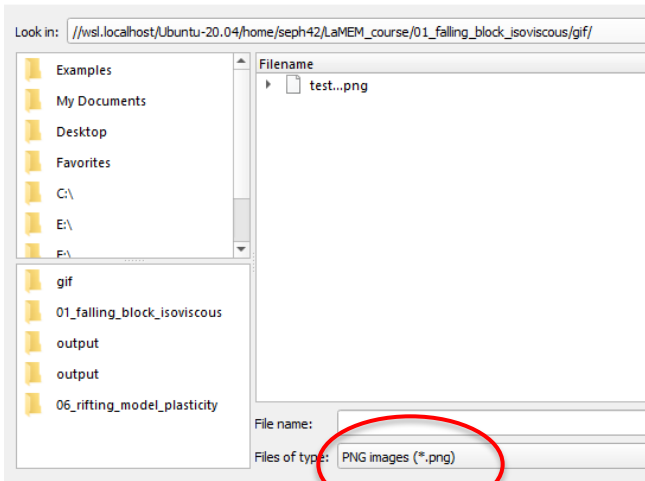


- This is important as you don't want to redo the visualization process from scratch all the time!

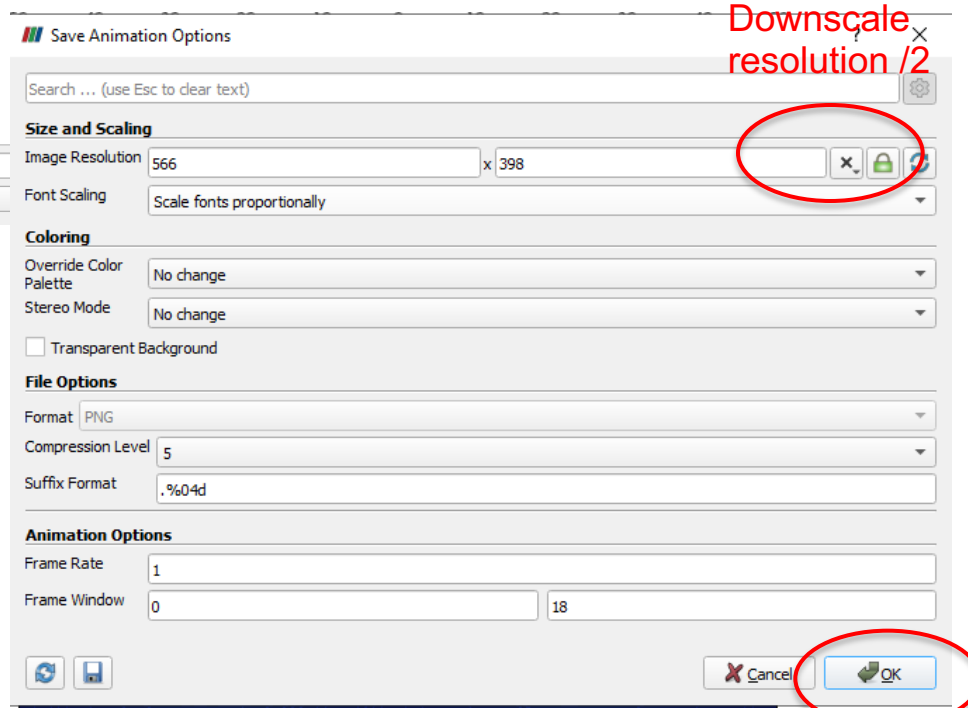
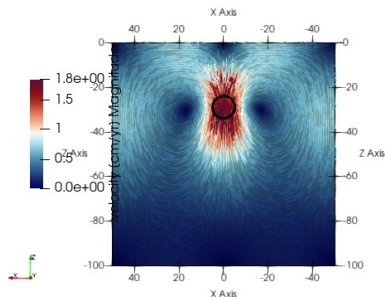
Visualization

- If you are happy with how it looks like, save an animation
- First create a directory next to “output” called “gif”

Save Animation



Choose png



Create a gif

- Install imagemagick (open a terminal in VS-code)

```
sudo apt-get update  
sudo apt-get install imagemagick
```

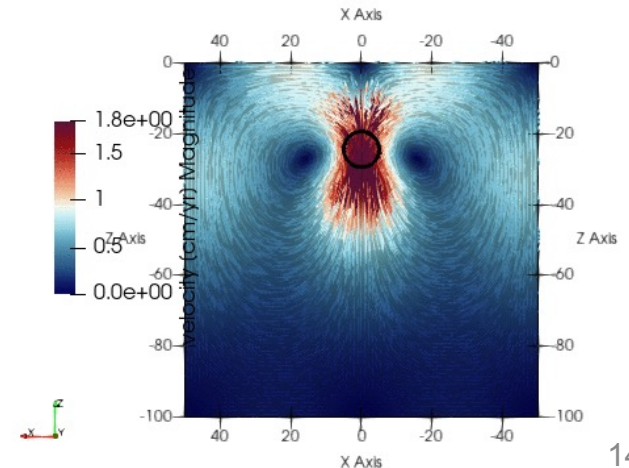


```
brew install imagemagick
```



- Go in gif directory (in the terminal using VS-code)

```
convert -delay 2 -loop 0 *.png ../01_falling_block_isoviscous.gif
```



Getting used to shapes and phases

- Copy and paste the falling sphere setup to a new directory
- Change the shape of the falling sphere to falling square:
 - Instead of using `add_ellipsoid!()`, use `add_box!()`. Think about using **Julia> ?add_box!** To get help!

```
add_box!(model; xlim=(minX, maxX),
               ylim=(minY, maxY),
               zlim=(minZ, maxZ),

               Origin          = nothing,
               StrikeAngle     = 0,
               DipAngle        = 0,
               phase           = ConstantPhase(1),
               T               = ConstantTemp(20) )
```

- Perform the simulation!
- Test out several options:
 - Change size (xlim and zlim) and DipAngle
 - Add a new phase (2) sphere or box, partially overlapping phase 1 (eclogite). Don't forget to add it to the `add_phase!()`
 - Change the lower half of the box into a lower density material (with respect to the upper half)