LaMEM short course

17-21 02 2025 Heidelberg

Nicolas Riel- nriel@uni-mainz.de

Passive tracers

- Points passively transported with velocity field
- Track phase ID, position and Pressure/Temperature conditions in time

Adding passive tracers

- Copy and paste 01_falling_block_iso_viscous.jl into a new folder
- In the model definition of the script “model = Model(…)” add the following function:

```
PassiveTracers(     Passive_Tracer      = 1,
                    PassiveTracer_Box   = [-2.5,2.5,-1,1,-27.5,-22.5]),
```

```
# set solution parameters
SolutionParams(     eta_min         = 1e19,
                    eta_ref         = 1e20,
                    eta_max         = 1e22),

PassiveTracers(     Passive_Tracer      = 1,
                    PassiveTracer_Box   = [-2.5,2.5,-1,1,-27.5,-22.5]),

# what will be saved in the output of the simulation
Output(             out_density         = 1,
                    out_melt_fraction   = 1,
                    out_j2_strain_rate  = 1,
                    out_temperature     = 1,
                    out_surf_velocity   = 1,
                    out_dir             = out_dir ),
```

Don’t forget to use the help!

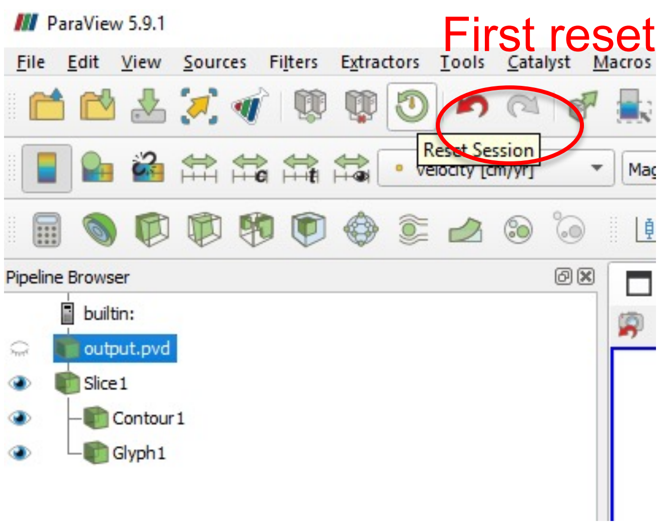
julia>?PassiveTracers
julia> PassiveTracers()

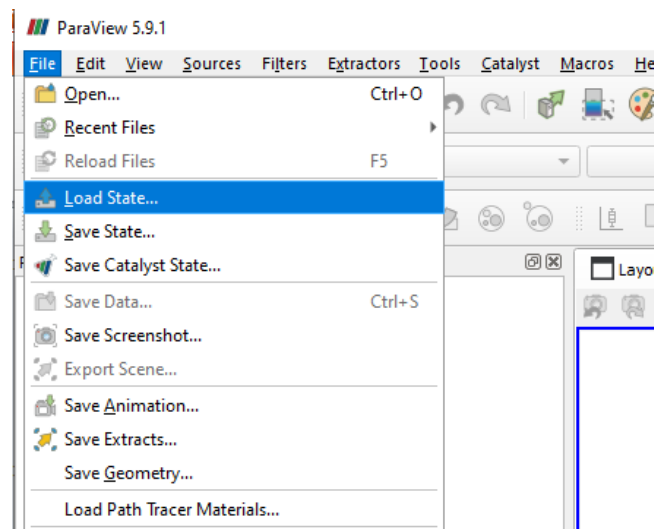
```
help?> PassiveTracers
search: PassiveTracers PassiveTracer_Time

  Structure that contains the LaMEM passive tracers parameters.

    • Passive_Tracer::Int64: activate passive tracers?"

    • PassiveTracer_Box::Union{Nothing, Vector{Float64}}: Dimensions of box in which we distribute passive tracers [Left, Right, Front, Back, Bottom, Top]

    • PassiveTracer_Resolution::Vector{Int64}: The number of passive tracers in every direction

    • PassiveTracer_ActiveType::Union{Nothing, String}: Under which condition are they activated? ["Always"], "Melt_Fraction", "Temperature", "Pressure", "Time"

    • PassiveTracer_ActiveValue::Union{Nothing, Float64}: The value to activate them
```

Adding passive tracers

- Perform the simulation!
- Use the previously made Paraview state to open the new output with passive tracers

First reset session

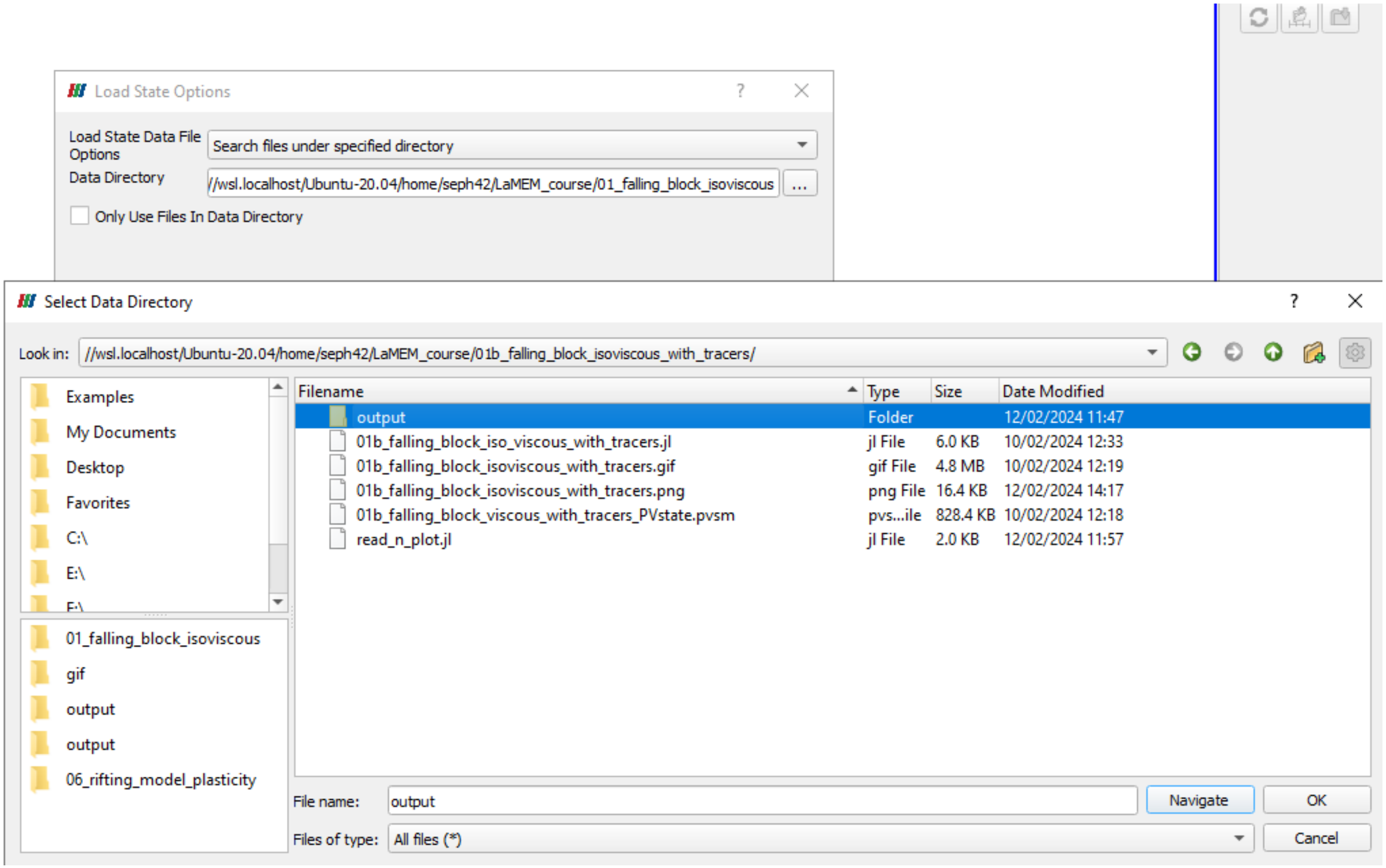




- Then load Previously saved paraview state

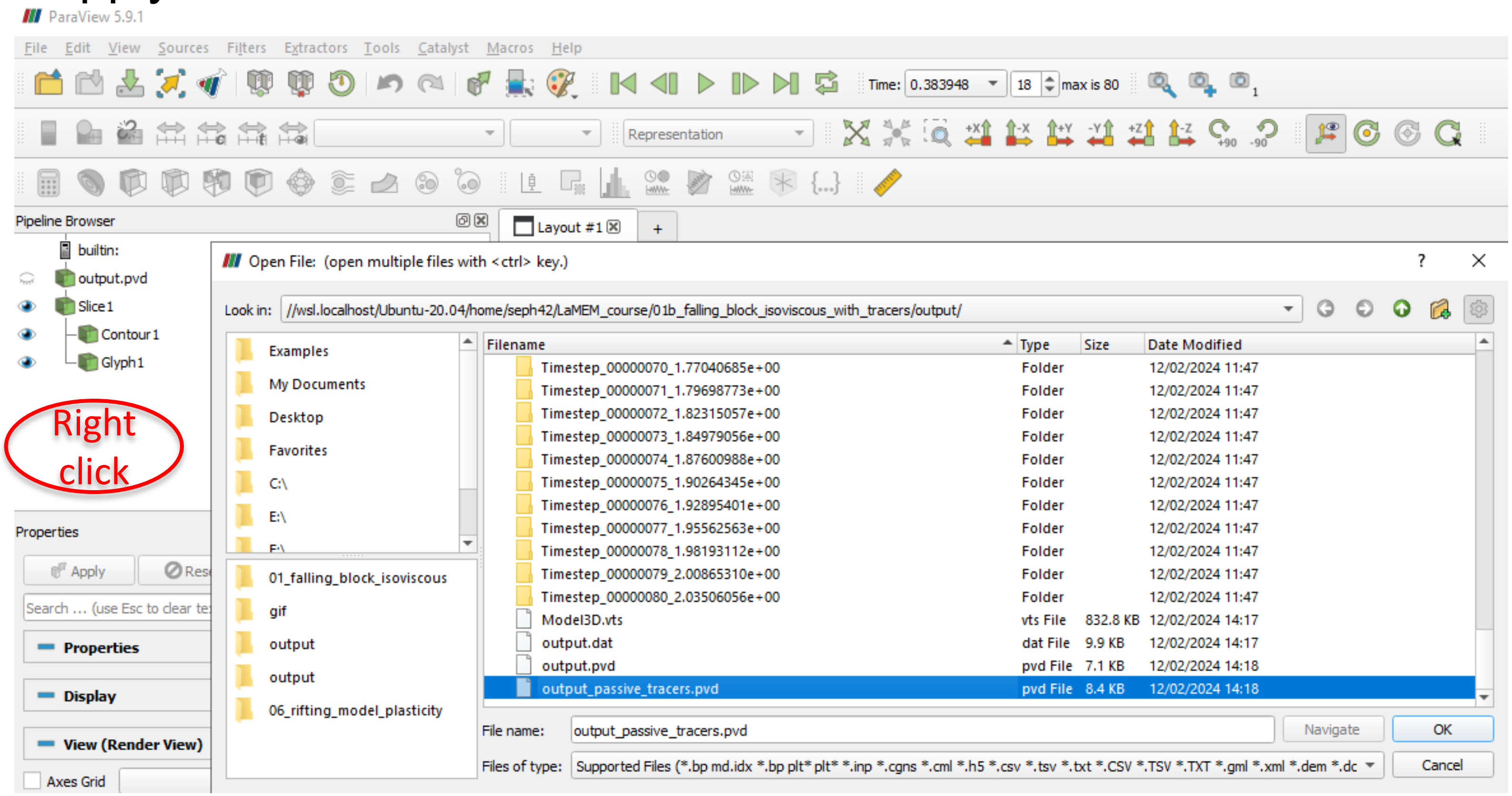
Adding passive tracers

- Select the right directory



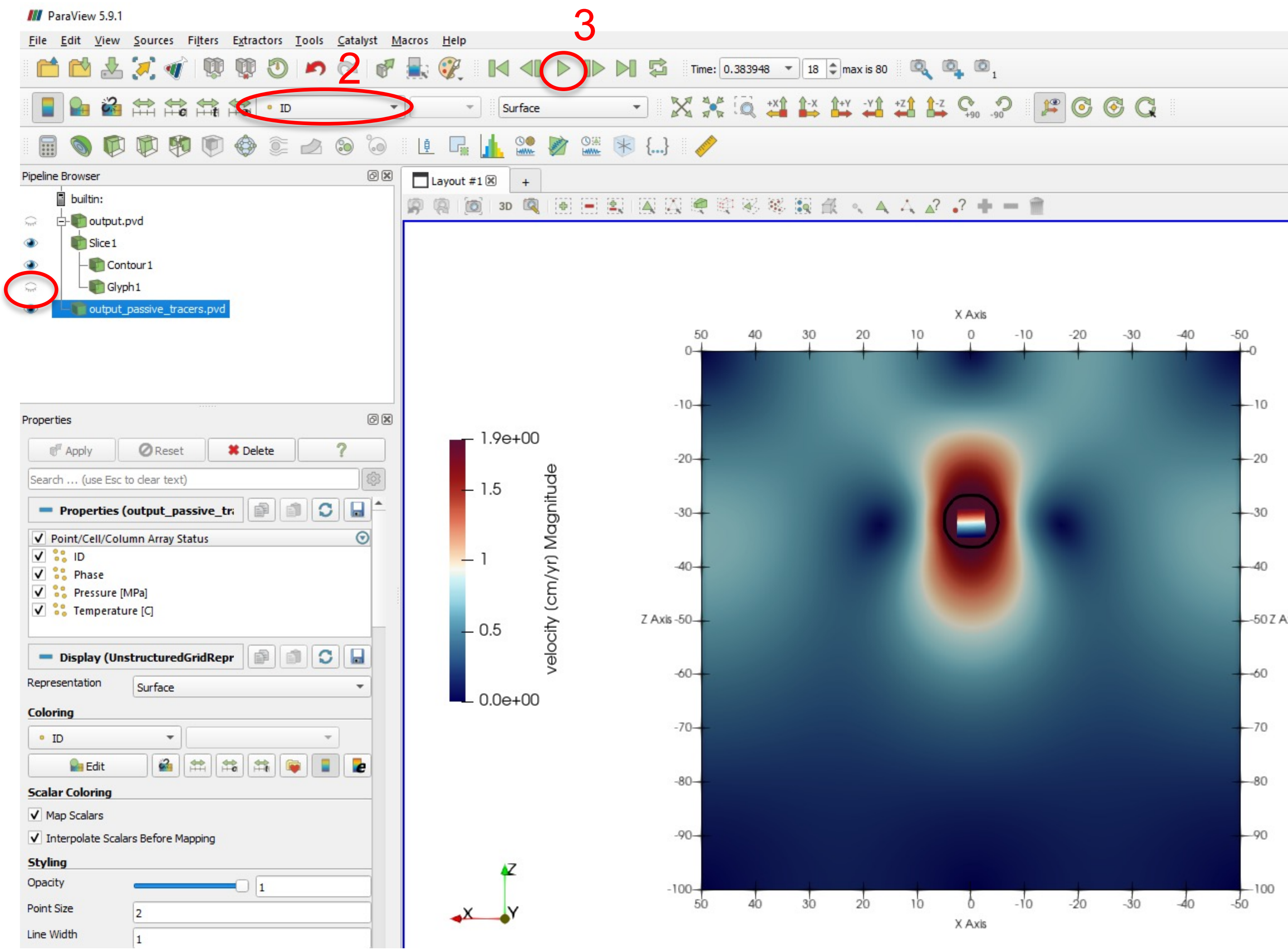
Adding passive tracers

- Then in right click in the “pipeline browser”
  → open
  → Select “output_passive_tracers.pvd”
  → apply



Visualize passive tracers

- Hide glyph
- Change display to ID
- Click play button



- Save state!

Extract passive tracer information

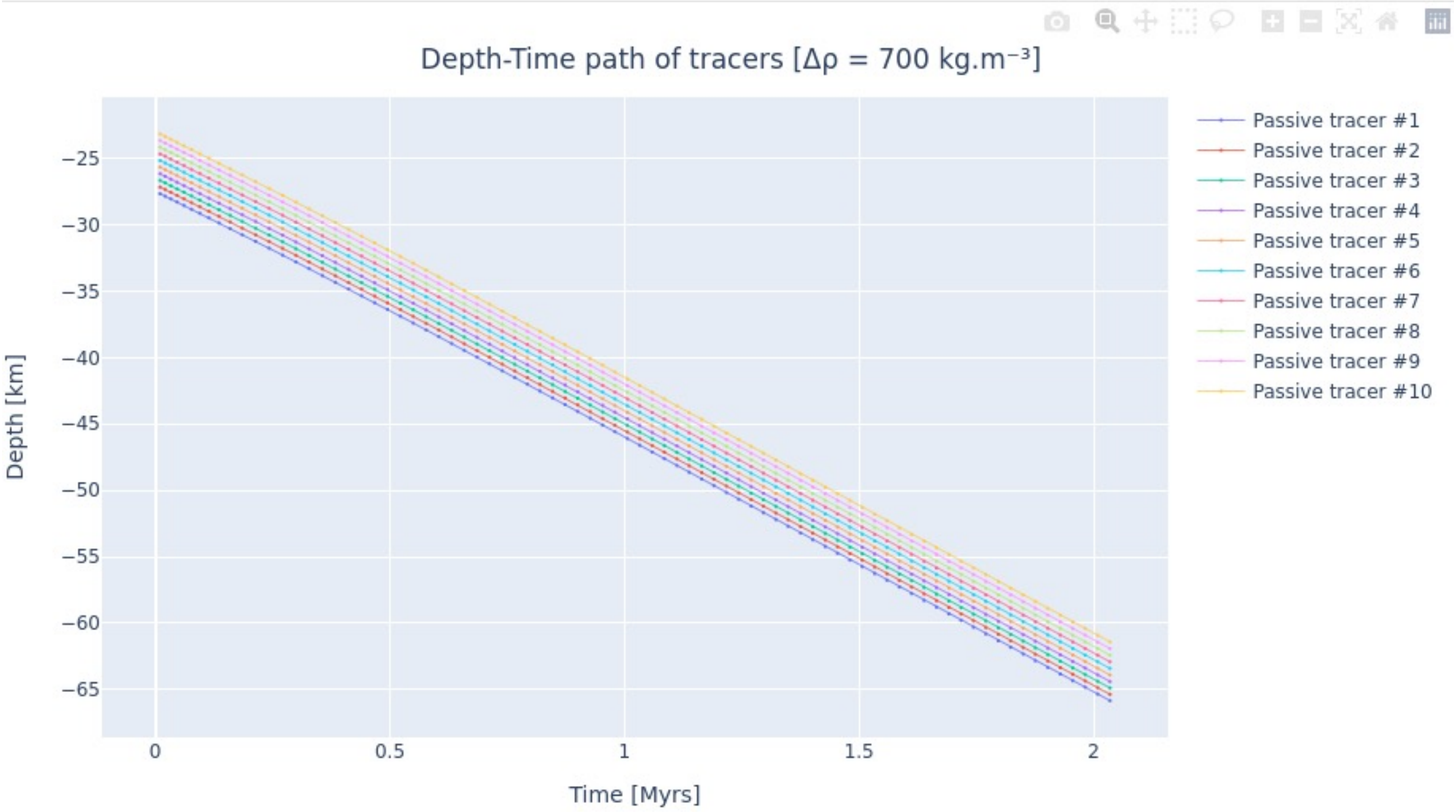
- Create a new Julia script where the output folder is located
- Copy and paste the following lines of code

```
using LaMEM
using PlotlyJS              # here we use PlotlyJS as it gives a bit more flexibility than the
default Plots package

dir         = "output"
input       = "output"

data_pt, time_pt = read_LaMEM_timestep(input, 1, dir, passive_tracers=true)

# display fields:
keys(data_pt.fields)
# (:Phase, :Temperature, :Pressure, :ID)
# the following gets the ID of the passive tracers of the timestep 1
# data_pt.fields[:ID]

nt          = length(data_pt.fields[:ID])
step        = 1000
in2         = collect(1:step:nt)
n_traces    = length(in2)
f_pt        = passivetracer_time( in2, input, dir)
```

Plot depth evolution

```
# the following uses PlotlyJS to create an array of traces
data_plot   = Vector{GenericTrace{Dict{Symbol, Any}}}(undef, n_traces);

# fill the tracers using the Pressure-Time paths of the selected passive tracers
for i=1:n_traces

    data_plot[i] = PlotlyJS.scatter(    x               = f_pt.Time_Myrs[2:end],
                                        y               = f_pt.z[i,2:end],
                                        name            = "Passive tracer #"*"$i",
                                        hoverinfo       = "skip",
                                        mode            = "markers+lines",
                                        marker          =  attr(size    = 2.0,),
                                        line            =  attr(width   = 0.75) )
end

layout  = Layout(

    title= attr(   text    = "Depth-Time path of tracers [Δρ = 700 kg.m⁻³]",
                   x       = 0.5,
                   xanchor = "center",
                   yanchor = "top"
    ),

    yaxis_title = "Depth [km]",
    xaxis_title = "Time [Myrs]",
)

fig = PlotlyJS.plot(data_plot,layout)
```

Plot depth evolution



Stoke's law

- Velocity of a falling sphere in a viscous medium

Density contrast

Gravity acceleration

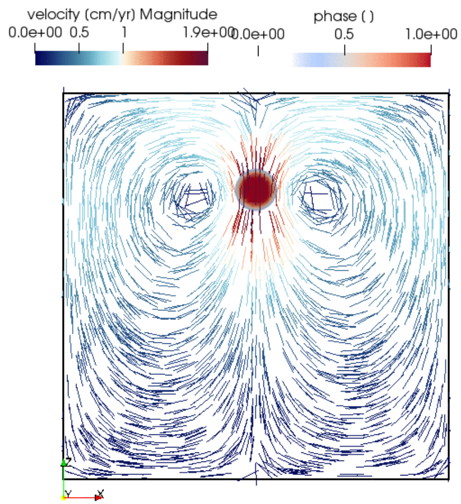
Sphere radius

Velocity

$$v = \frac{\Delta\rho\, g R^2}{3\eta}$$

Viscosity

Stoke’s law



$$v = \frac{\Delta\rho\, g R^2}{3\eta}$$

<u>Exercise 1</u>

- Using falling block examples 01 and 01b
  - Change default sphere radius and host rock viscosity
  - Compute numerical vs analytical falling block velocity for several Δρ
  - Plot the results

    → How does that compare?
  - Change boundary conditions to no slip on all walls

    → How does the relationship change?

<u>HELP:</u>

just compute the velocity of one tracer per simulation

Stoke’s law

<u>Exercise 1</u>

- Using falling block examples 01 and 01b
  - Change default sphere radius and host rock viscosity
  - Compute numerical vs analytical falling block velocity for several Δρ
  - Plot the results
    → How does that compare?
  - Change boundary conditions to no slip on all walls
    → How does the relationship change?

$$v = \frac{\Delta\rho\, g R^2}{3\eta}$$

<u>HELP:</u>

- Only compute the velocity of one tracer per simulation (you don’t need more)
- Use previous “trace” plotting routine to compute the velocity:

```
v_km_per_myr    = (f_pt.z[1,end] - f_pt.z[1,2])/(f_pt.Time_Myrs[end] - f_pt.Time_Myrs[2])
```

- Convert to m/s!
- Add the explored Δρ and computed velocities to array e.g., [v1,v2,v3, …,vn]
- You can create 2 traces and merge them during plotting as

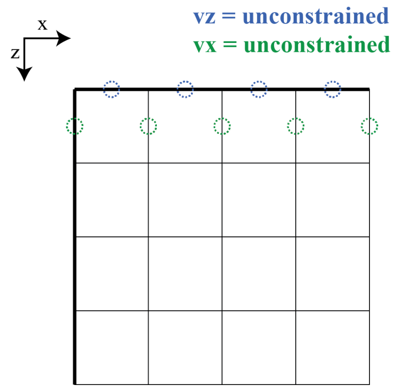
```
fig = PlotlyJS.plot([trace1,trace2],layout)
```

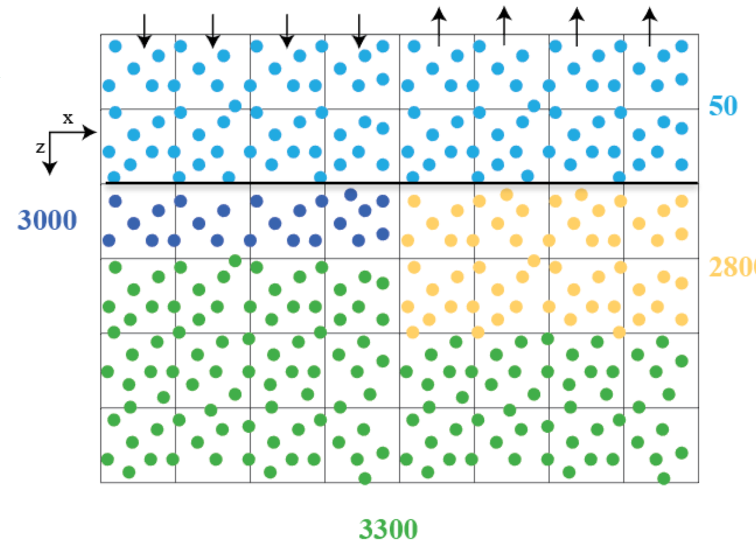
Free surface

- Allow to model topograpy
- Several timesteps are needed to reach isostatic equilibrium!

Adding free surface

- **Free surface: “sticky air”**





- Copy and paste “01b_falling_block_iso_viscous_with_tracers.jl” and rename it as “02_falling_block_iso_viscous_free_surface.jl“

- Change Grid(z = [-100.0,0.0]) to Grid(z = [-100.0, **10.0**])

```
Grid(               x               = [-50.0,50.0],
                    y               = [-1.0,1.0],
                    z               = [-100.0,10.0],
                    nel             = (96,1,96) ),
```

- Change open top to 1

```
# sets the conditions at the walls of the modelled domain
BoundaryConditions( temp_bot        = 20.0,
                    temp_top        = 20.0,
                    open_top_bound  = 1,
                    noslip          = [0, 0, 0, 0, 0, 0]),
```

Adding free surface

- Modify SolutionsParams(…) by adding FSSA = 1.0 entry

```
# set solution parameters
SolutionParams(     eta_min         = 1e19,
                    eta_ref         = 1e20,
                    eta_max         = 1e22,
                    FSSA            = 1.0 ), # Free surface stabilization algo
```

- Add a “FreeSurface” section

```
FreeSurface(        surf_use        = 1,  # use free surface
                    surf_level      = 0.0, # initial surf depth
                    surf_air_phase  = 0,   # phase ID
                    surf_max_angle  = 40.0
                ),
```

- Add surface output to the “output” section of the Model():

```
out_surf            = 1,                    #
out_surf_velocity   = 1,                    #
out_surf_pvd        = 1,                    #
out_surf_topography = 1,                    #
```

Adding free surface

- Add the air phase, first we retrieve the Z array then set the value above surface to phase id 0

```
Z                                       = model.Grid.Grid.Z;
model.Grid.Phases[Z.>0.0]              .= 0;                        # if Z > 0 then we attribute the air phase value 0
```

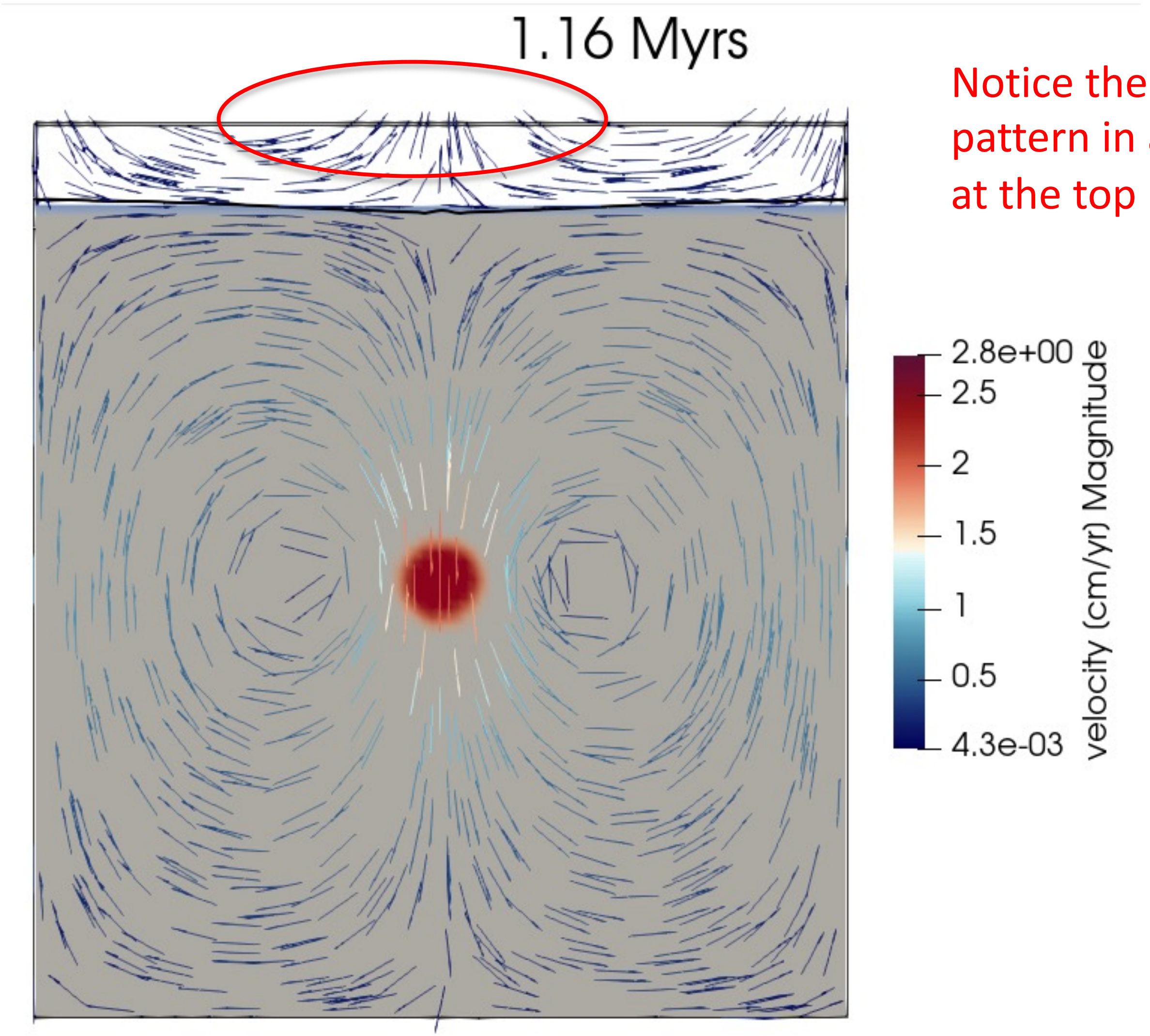
- Create the air phase rheology:

```
air = Phase(            Name            = "Air",
                        ID              = 0,
                        rho             = 50,                       # prescribe a relatively low density for the air. Mind that realistic density for the air may lead to numerical instability
                        eta             = 1e20,                     # here we set the viscosity on the air as the minimum viscosity in our simulation, so the one of the mantle
                        G               = 5e10 );
```

- Perform the simulation!
- Display the results in Paraview using the paraview state you generated for the example without free surface

Adding free surface

- It should look like this

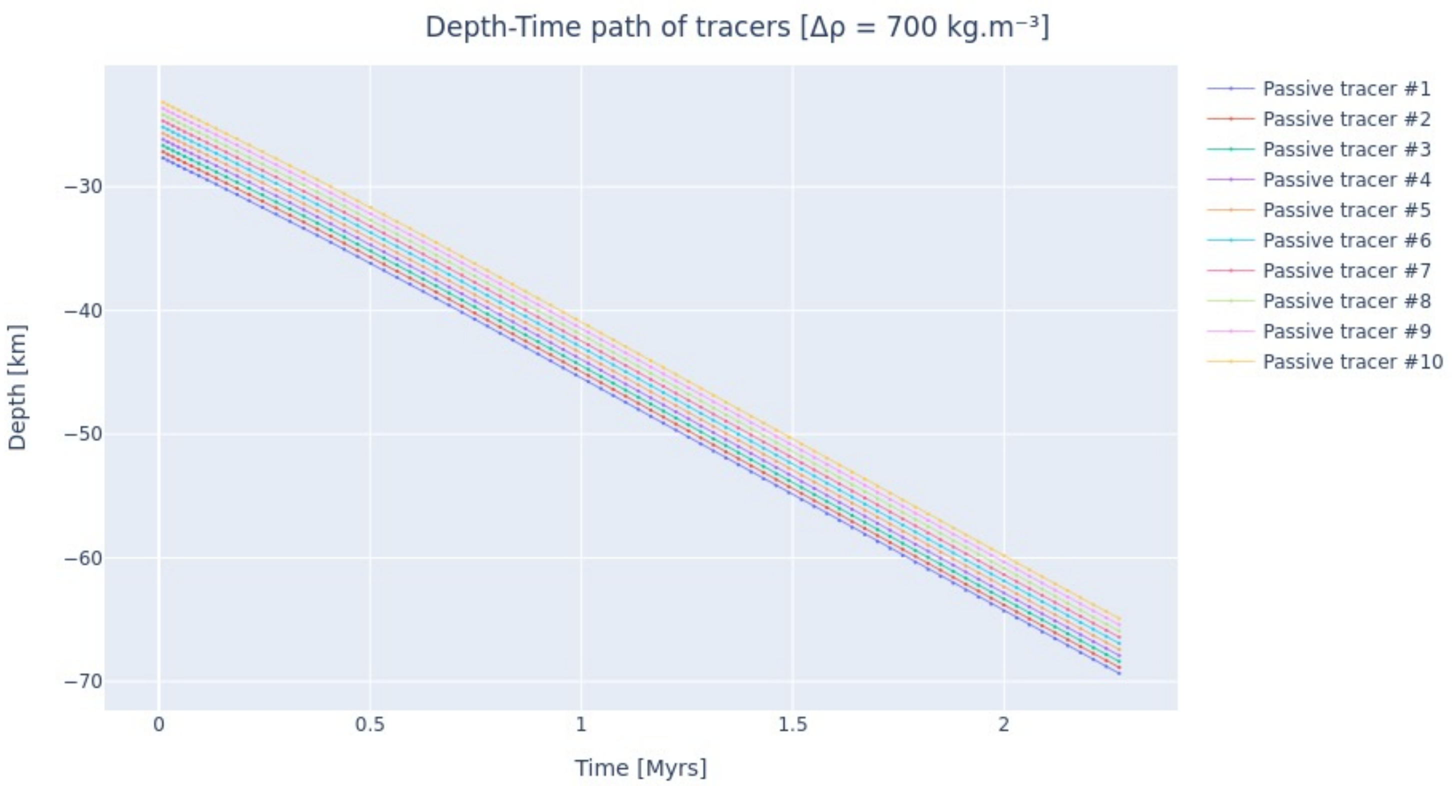


Notice the flow pattern in and out at the top boundary

Adding free surface

- Plot the Depth-Time tracer path using the previous read_n_plot.jl script

- Is there a difference between free slip and free surface for this setup?



Heat transport

- Accounts for temperature evolution of the system i.e. heat diffusion and advection
- Allow feedback between temperature and mechanical deformation e.g., thermal expansivity
- Allow to use (more realistic) temperature-dependent rheology e.g. diffusion and dislocation creep

Adding temperature

- Up to now, temperature was defined as a constant and not linked to phases
- Copy previous script (falling sphere with surface) to a new directory and change its name.
- In the Model(…) definition change top and bottom boundary temperature

```
# sets the conditions at the walls of the modelled domain
BoundaryConditions( temp_top        = 20.0,
                    temp_bot        = 2020,
                    open_top_bound  = 0,
                    noslip          = [0, 0, 0, 0, 0, 0]),
```

- In the SolutionsParams() we need to add one line to activate thermal diffusion:

```
# set solution parameters
SolutionParams(     eta_min         = 1e19,
                    eta_ref         = 1e20,
                    eta_max         = 1e24,
                    act_temp_diff   = 1,
                    FSSA            = 1.0 ),
```

Adding temperature

- We now need to change the starting temperature field. Right at the beginning of “phases definition section” of the input script:

```
#================== define phases (different materials) of the model ========================#
```

- Add the following lines:

```
Z                                        = model.Grid.Grid.Z;

geotherm                                 = 20.0        # K/km
model.Grid.Temp                          = -Z.*geotherm .+ model.BoundaryConditions.temp_top;
model.Grid.Temp[Z.>0.0]                 .= 20.0;
```

- We access the temperature of the particles using “.” such as for the definition of the phases “model.Grid.Phase”

- Because Z is negative below the surface, temperature is defined as –Z .* geotherm

- Don’t forget the point-wise operator “.” before the “*” and “+” operators

Adding temperature

- We also want to change the temperature of the falling sphere.

- Because we want temperature to be driving mechanism, we set it to a low value (with respect to the surrounding mantle)

```
# Note that the shape is defined after the linear temperature gradient is applied to the whole
domain
add_ellipsoid!(model,    cen               = (0,0,-25),                # defines centre of the sphere
                        axes              = (8,8,8),                  # define size of the sphere
                        StrikeAngle       = 0,
                        DipAngle          = 0,
                        phase             = ConstantPhase(2),
                        T                 = ConstantTemp(200));
```

Adding temperature

- We now need to change the starting temperature field. Right at the beginning of “phases definition section” of the input script:

```
#================== define phases (different materials) of the model =========================#
```

- Add the following lines:

```
Z                                         = model.Grid.Grid.Z;

geotherm                                  = 20.0        # K/km
model.Grid.Temp                           = -Z.*geotherm .+ model.BoundaryConditions.temp_top;
model.Grid.Temp[Z.>0.0]                  .= 20.0;
```

- We access the initial temperature of the particles using “.” such as for the definition of the phases “model.Grid.Phase”

- Because Z is negative below the surface, temperature is defined as –Z .* geotherm

- Don’t forget the point-wise operator “.” before the “*” and “+” operators

Adding temperature

- Material properties also need to be upgraded to account for heat transfer

```
#===================== define material properties of the phases ============================#
air = Phase(            Name            = "Air",
                        ID              = 0,
                        k               = 100,
                        Cp              = 1e6,                   # heat capacity
                        rho             = 50,
                        eta             = 1e20,
                        G               = 5e10 );

mantle = Phase(         Name            = "Mantle",
                        ID              = 1,
                        alpha           = 3e-5,
                        k               = 3,                     # conductivity
                        Cp              = 1000,                  # heat capacity
                        rho             = 3300,
                        eta             = 1e20,
                        G               = 5e10 );

cold_mantle = Phase(    Name            = "cold_mantle",
                        ID              = 2,
                        alpha           = 3e-5,
                        k               = 3,                     # conductivity
                        Cp              = 1000,                  # heat capacity
                        rho             = 3300,
                        eta             = 1e24,
                        G               = 5e10 );
```

- alpha = thermal expansivity, k = conductivity, Cp = heat capacity
- Note that for the “Air” the values are artificially high to keep temperature constant!

Adding temperature

- In previous setup (without temperature), sinking was triggered by prescribed density contrast between “eclogite” and “mantle” (4000 vs 3300)

- In this setup we want that temperature controls density contrast. Therefore, we change the name of the eclogite phase to cold_mantle and set the reference density as 3300 (like for the “mantle” phase)

Setup without temperature

```
eclogite = Phase(        Name                = "eclogite",
                         ID                  = 2,
                         rho                 = 4000,
                         eta                 = 1e24,
                         G                   = 5e10 );
```

Setup with temperature

```
cold_mantle = Phase(     Name                = "cold_mantle",
                         ID                  = 2,
                         alpha               = 3e-5,
                         k                   = 3,
                         Cp                  = 1000,
                         rho                 = 3300,
                         eta                 = 1e24,
                         G                   = 5e10 );
```

- Don’t forget to update the name here too:

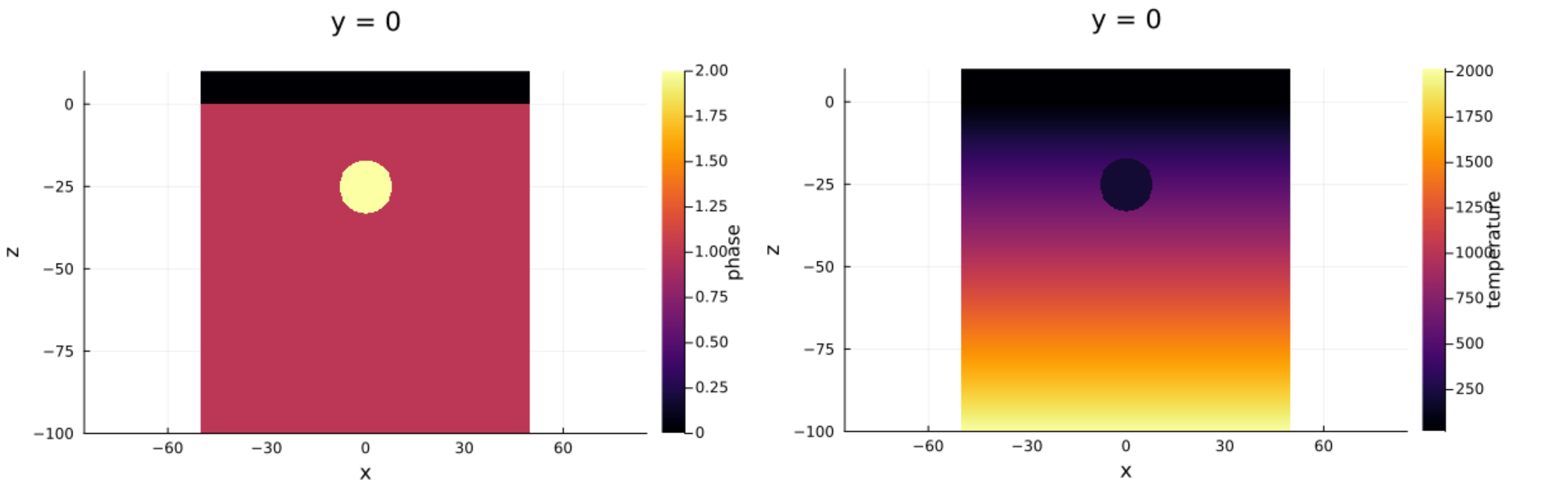
```
add_phase!( model, air, mantle, cold_mantle)
```

Adding temperature

- Generate the cross-sections

```
plot_cross_section(model, y=0, field=:phase)
savefig("03_T_falling_block_iso_viscous_free_surface_phase.png")
plot_cross_section(model, y=0, field=:temperature)
savefig("03_T_falling_block_iso_viscous_free_surface_temp.png")
```

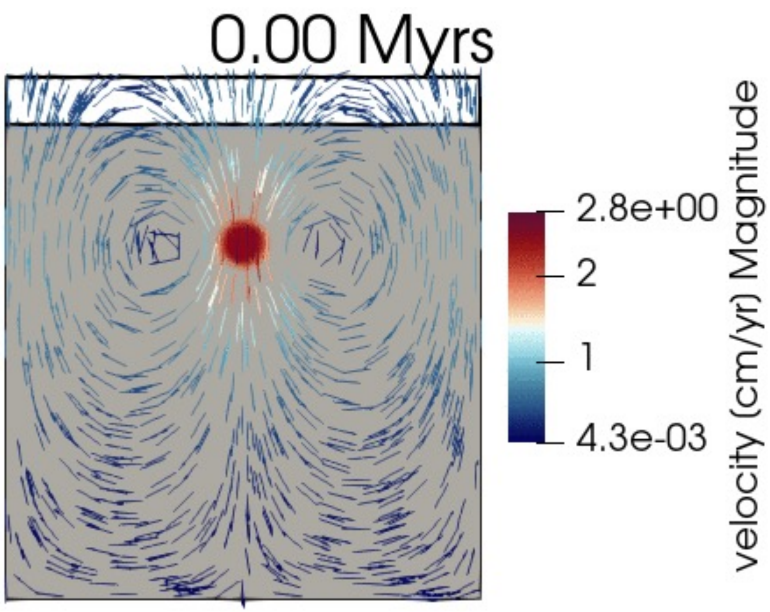
It should look like this!

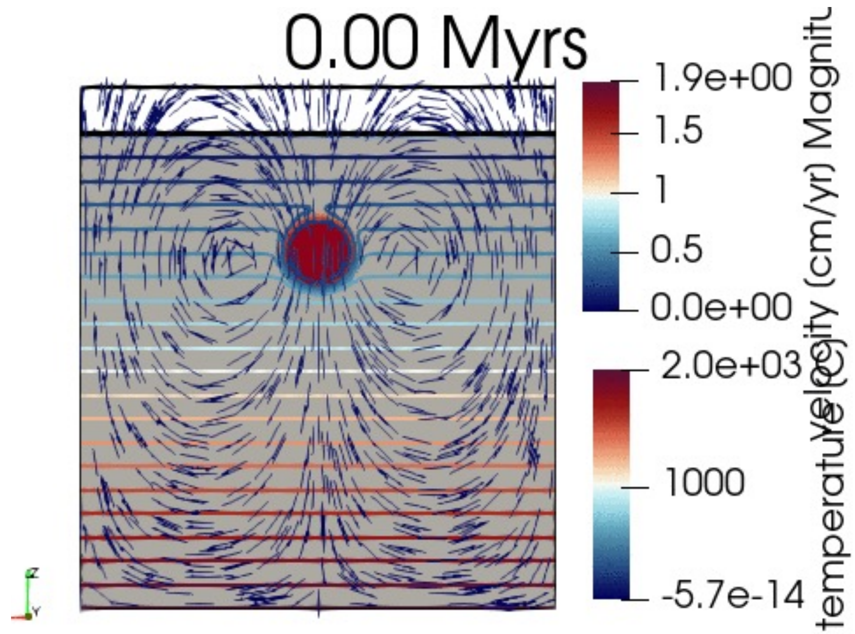


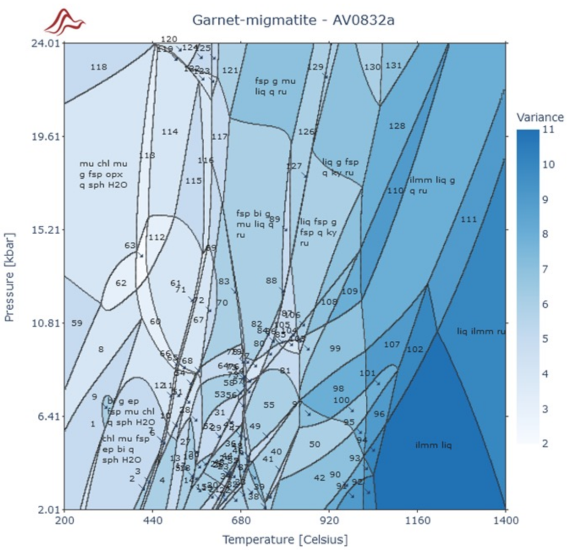
- Perform the simulation!

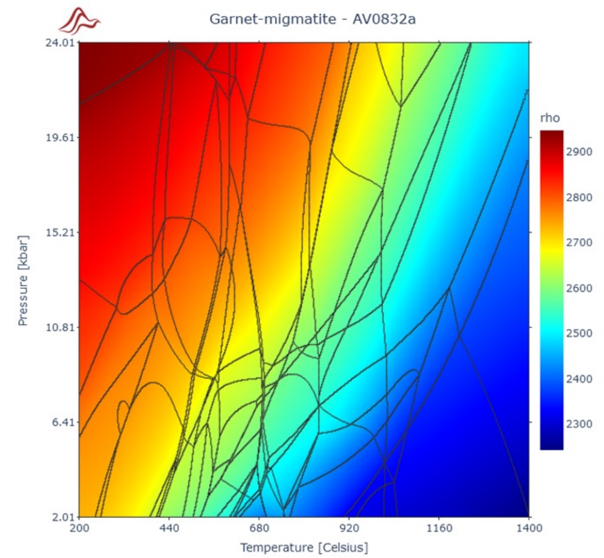
Adding temperature

- What differ from the simulation without heat transport?









Density diagrams

- Allow to use P-T density from thermodynamic equilibrium calculation
- You can use MAGEMinApp to compute and export density diagrams for LaMEM

Adding density diagrams

- In previous setups density was given as a reference value and made temperature dependent when heat transport was activated

- What if, instead, we want density to be computed by thermodynamic calculation and used in LaMEM?

- In the phase_diagrams_4_LaMEM is provided density diagrams for common rock-types. They can be directly used in your simulations.

  - CFB.in
  - ContinentalCrust.in
  - Greywacke.in
  - LCC_H2O-saturated.in
  - LCC_H2O-undersaturated.in
  - NMORB.in
  - Peridotite.in
  - Peridotite_depleted_10.in
  - Serpentinite.in
  - UCC.in

- Note that these diagrams do not account for melt.

Adding density diagrams

- Copy previous setup (falling sphere with temperature)

- Then from the directory “phase_diagrams_4_LaMEM” copy the “Peridotite.in” in your new setup.

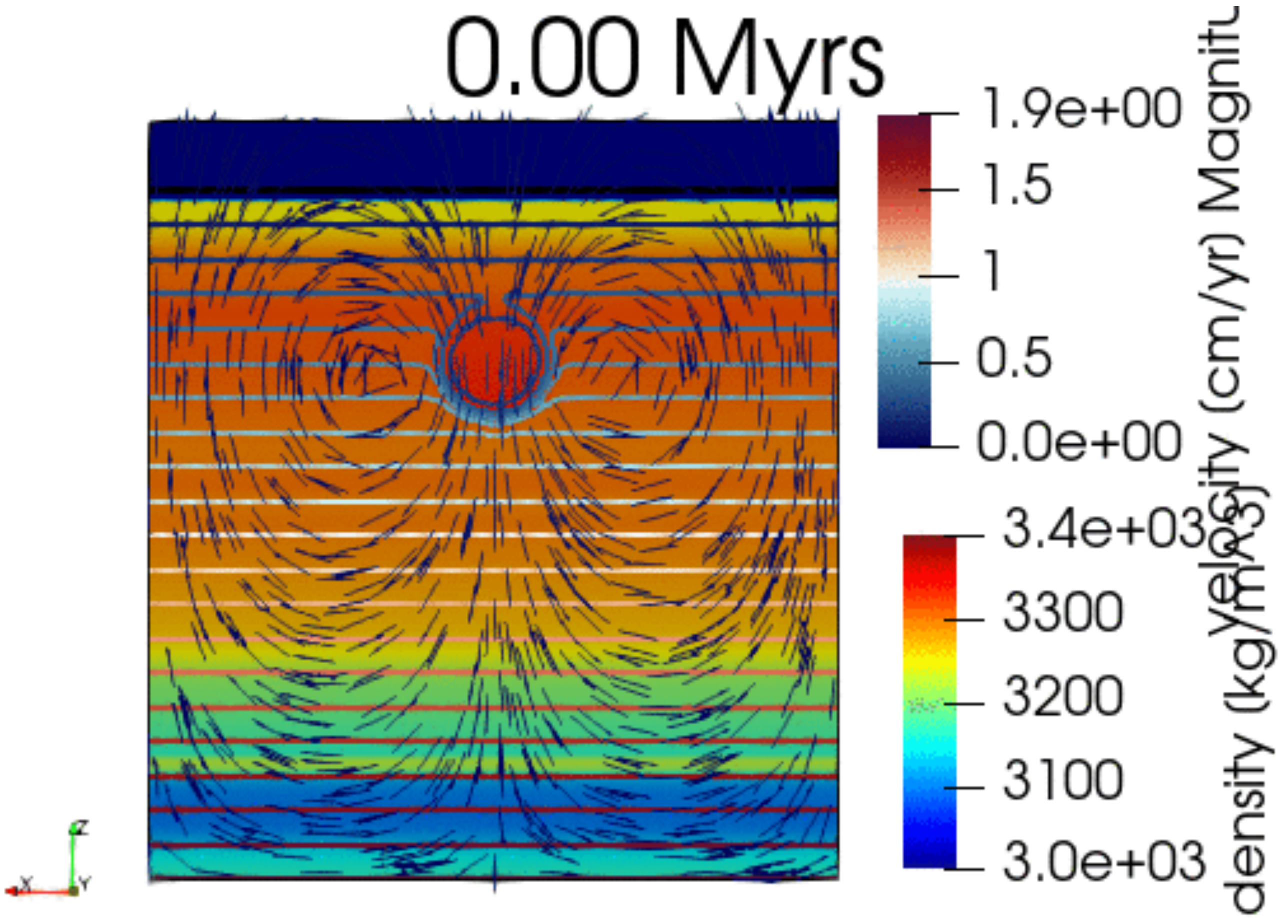
- Then simply add rho_ph = “peridotite” to your mantle phase definitions:

```
mantle = Phase(          Name           = "Mantle",                # let's call phase 0 mantle
                         ID             = 1,
                         alpha          = 3e-5,
                         k              = 3,                       # conductivity
                         Cp             = 1000,                    # heat capacity
                         rho            = 3300,                    # set mantle density
                         eta            = 1e20,                    # set mantle viscosity
                         G              = 5e10,
                         rho_ph         = “../Peridotite" );

cold_mantle = Phase(     Name           = "cold_mantle",
                         ID             = 2,
                         alpha          = 3e-5,
                         k              = 3,                       # conductivity
                         Cp             = 1000,                    # heat capacity
                         rho            = 3300,
                         eta            = 1e24,
                         G              = 5e10,
                         rho_ph         = “../Peridotite" );
```

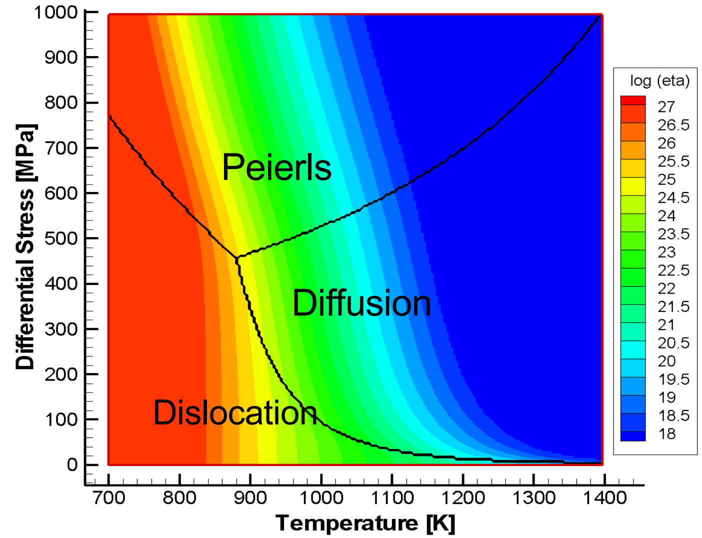
- Note that “rho = 3300” is still defined as we need it for the initial guess

Adding density diagrams

- Perform the simulation!



- How different is the density field compared to previous simulation?



Viscous Creep laws

- Strain-rate, temperature (pressure) dependent

$$
\begin{aligned}
\eta_l &= \tfrac{1}{2}\,(A_l)^{-1} \\
\eta_n &= \tfrac{1}{2}\,(A_n)^{-\frac{1}{n}}\,(\dot{\varepsilon}^*_{II})^{\frac{1}{n}-1} \\
\eta_p &= \tfrac{1}{2}\,(A_p)^{-\frac{1}{s}}\,(\dot{\varepsilon}^*_{II})^{\frac{1}{s}-1}
\end{aligned}
$$

$$
\begin{aligned}
A_l &= B_l\ \exp\left[-\tfrac{E_l+pV_l}{RT}\right], \\
A_n &= B_n\ \exp\left[-\tfrac{E_n+pV_n}{RT}\right] \\
A_p &= \tfrac{B_p}{(\gamma\tau_p)^s}\ \exp\left[-\tfrac{E_p+pV_p}{RT}\,(1-\gamma)^q\right]
\end{aligned}
$$

Adding Viscous creep laws

- So far, we only used iso-viscous rheology. Let’s add more realistic, strain-rate, temperature and pressure dependent rheologies.

- In the github repo: https://github.com/NicolasRiel/LaMEM_course, you access at the end of the paper to all natively available creep-laws

LaMEM creep laws

Diffusion creep:

[Hirth, G. & Kohlstedt (2003), D. Rheology of the upper mantle and the mantle wedge: A view from the experimentalists]

- "Dry_Olivine_diff_creep-Hirth_Kohlstedt_2003"
- "Wet_Olivine_diff_creep-Hirth_Kohlstedt_2003_constant_C_OH"
- "Wet_Olivine_diff_creep-Hirth_Kohlstedt_2003"

[Rybacki and Dresen, 2000, JGR]

- "Dry_Plagioclase_RybackiDresen_2000"
- "Wet_Plagioclase_RybackiDresen_2000"

Dislocation creep:

[Ranalli 1995]

- "Dry_Olivine-Ranalli_1995"

(...)

Adding Viscous creep laws

- For this example, we will be using one diffusion and one dislocation creep law, namely:

  "Dry_Olivine_diff_creep-Hirth_Kohlstedt_2003"
  and
  "Dry_Olivine_disl_creep-Hirth_Kohlstedt_2003"

- For details about these laws and how they are calibrated refer to the publication.

- We will discuss in a later stage how to choose which law to use!

- As usual copy the previous example (falling sphere with temperature and density diagram) to a new folder

Adding Viscous creep laws

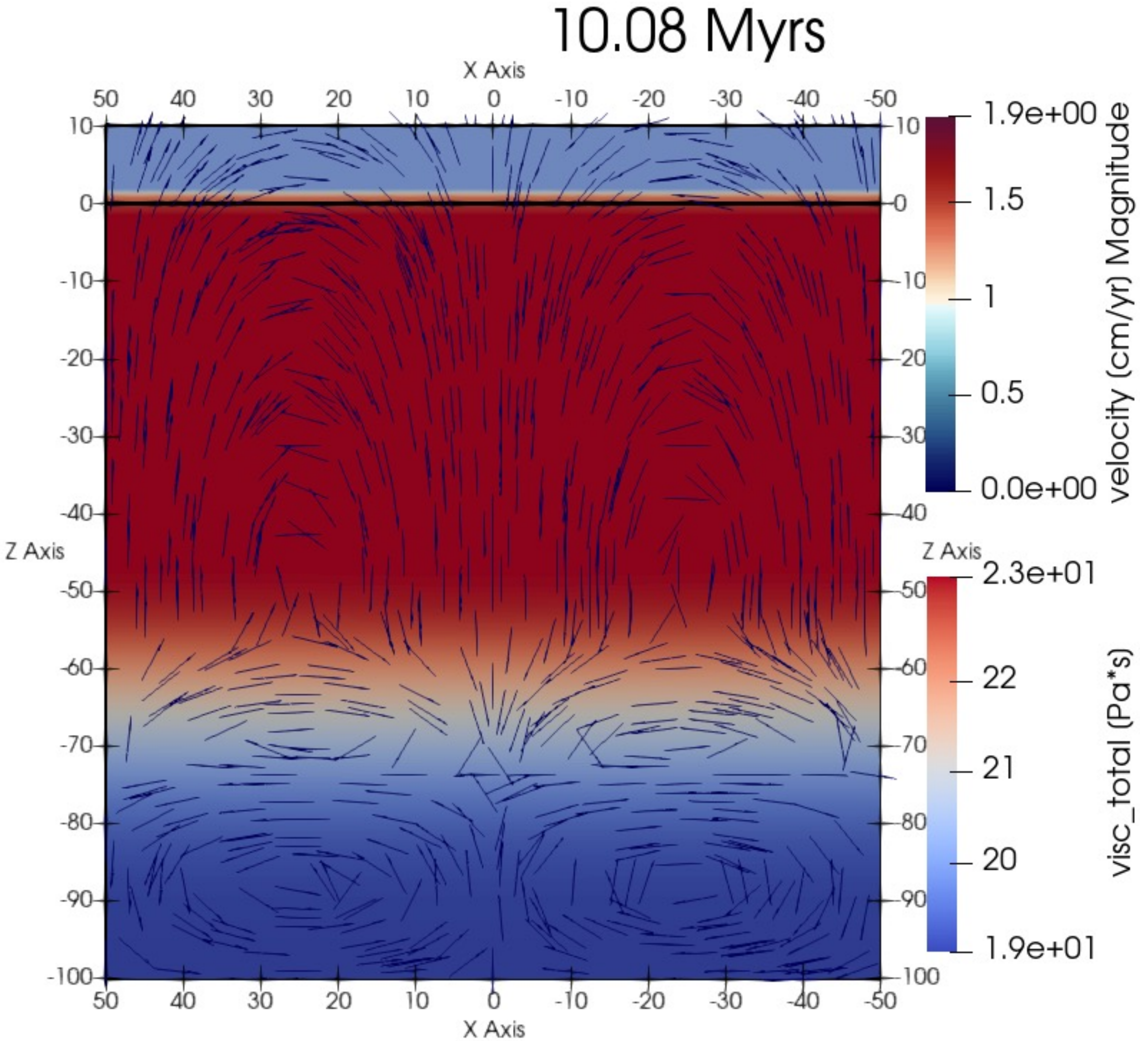
- Then change eta = (…) to:

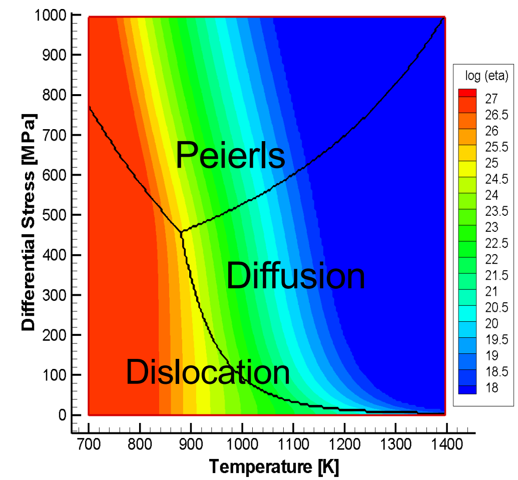
```
mantle = Phase(          Name            = "Mantle",                         # let's call phase 0 mantle
                         ID              = 1,
                         alpha           = 3e-5,
                         k               = 3,                                # conductivity
                         Cp              = 1000,                             # heat capacity
                         rho             = 3300,                             # set mantle density
                         disl_prof       = "Dry_Olivine_disl_creep-Hirth_Kohlstedt_2003",
                         diff_prof       = "Dry_Olivine_diff_creep-Hirth_Kohlstedt_2003",
                         G               = 5e10,
                         rho_ph          = "../Peridotite" );                           # set elastic modulii

cold_mantle = Phase(     Name            = "cold_mantle",
                         ID              = 2,
                         alpha           = 3e-5,
                         k               = 3,                                # conductivity
                         Cp              = 1000,                             # heat capacity
                         rho             = 3300,
                         disl_prof       = "Dry_Olivine_disl_creep-Hirth_Kohlstedt_2003",
                         diff_prof       = "Dry_Olivine_diff_creep-Hirth_Kohlstedt_2003",
                         G               = 5e10,
                         rho_ph          = "../Peridotite" );
```

- That’s all! Run the simulation!

Adding Viscous creep laws

- What is happening there?





- Large viscosity in the lithosphere
- Max value is capped at 1e23 Pa s, why?

Adding Viscous creep laws

- Try with Wet olivine instead

```
mantle = Phase(          Name              = "Mantle",                    # let's call phase 0 mantle
                         ID                = 1,
                         alpha             = 3e-5,
                         k                 = 3,                           # conductivity
                         Cp                = 1000,                        # heat capacity
                         rho               = 3300,                        # set mantle density
                         disl_prof         = "Wet_Olivine_disl_creep-Hirth_Kohlstedt_2003",
                         diff_prof         = "Wet_Olivine_diff_creep-Hirth_Kohlstedt_2003",
                         G                 = 5e10,
                         rho_ph            = "../Peridotite" );                      # set elastic modulii

cold_mantle = Phase(     Name              = "cold_mantle",
                         ID                = 2,
                         alpha             = 3e-5,
                         k                 = 3,                           # conductivity
                         Cp                = 1000,                        # heat capacity
                         rho               = 3300,
                         disl_prof         = "Wet_Olivine_disl_creep-Hirth_Kohlstedt_2003",
                         diff_prof         = "Wet_Olivine_diff_creep-Hirth_Kohlstedt_2003",
                         G                 = 5e10,
                         rho_ph            = "../Peridotite" );
```

- Does that help?