



Overview of the **SiStER** code

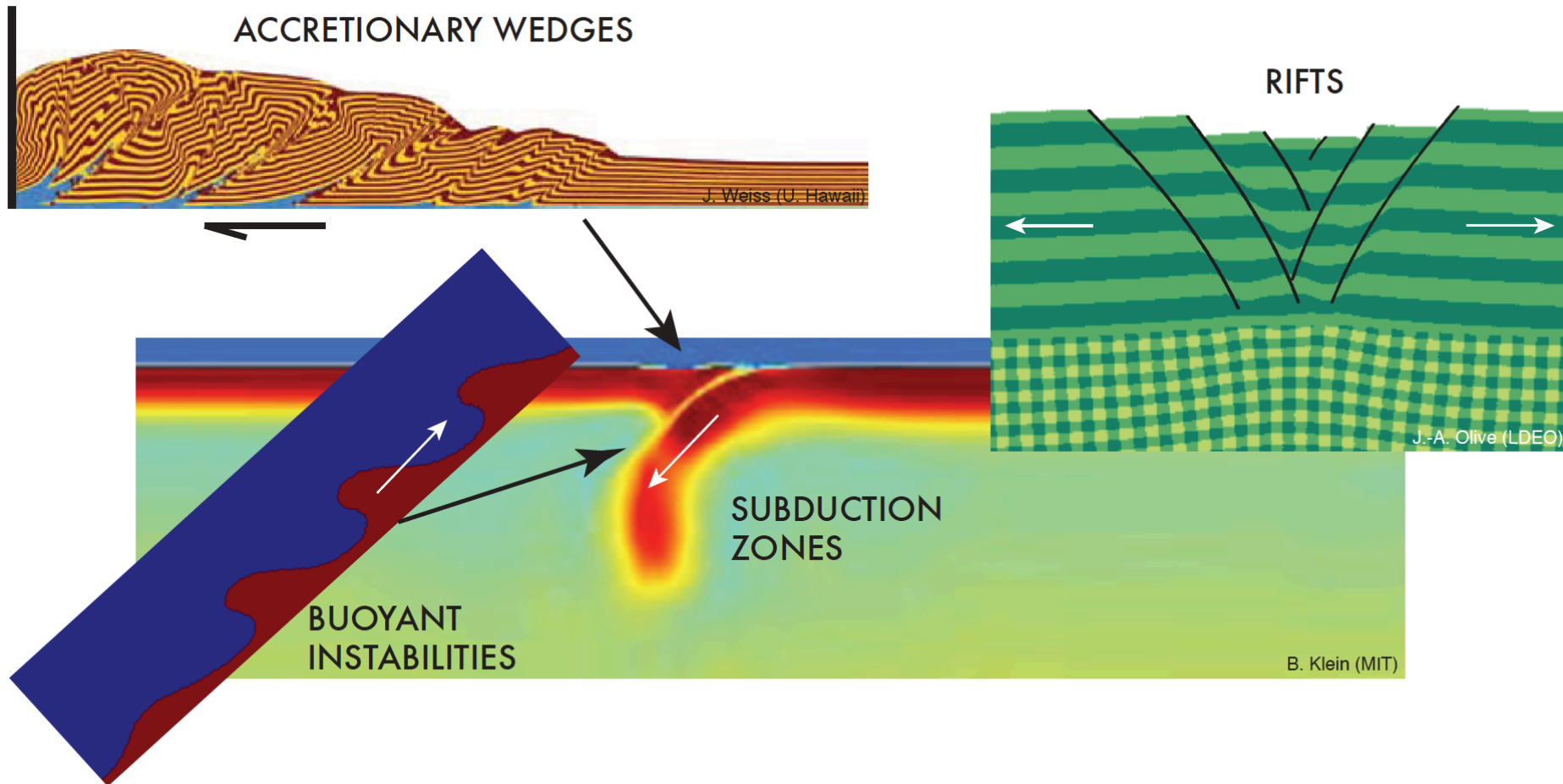
Simple Stokes solver with Exotic Rheologies

Jean-Arthur Olive (LDEO / Columbia University)

In collaboration with E. Mittelstaedt (U. Idaho), B.Z. Klein (MIT),
S. Howell (JPL), M.D. Behn (WHOI), and G. Ito (U. Hawaii)

Objectives & Challenges

Modeling lithosphere and mantle deformation with continuum mechanics:
Stokes flow with large strains, strain localization, non-linear rheologies, sharp contrasts in material properties, complex BCs.



Governing equations

- Conservation of mass and momentum

$$\partial_k v_k = 0$$

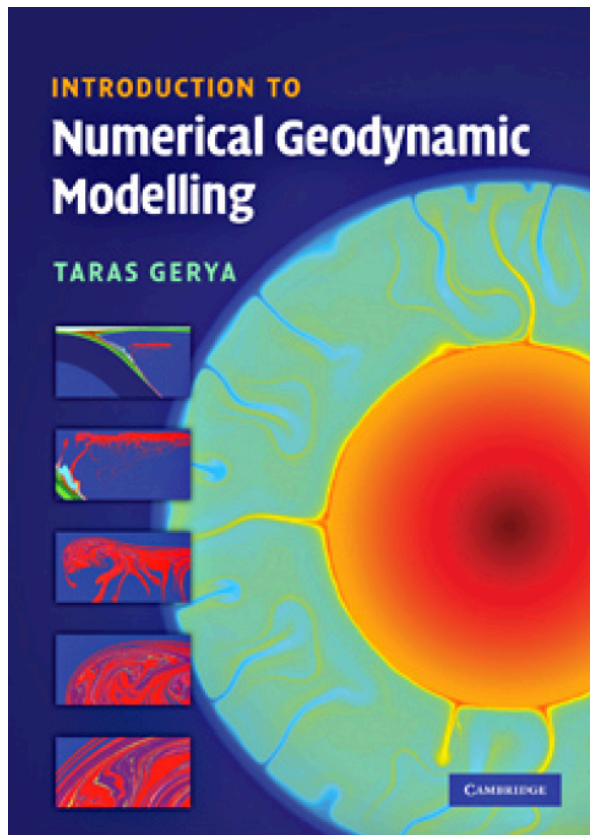
$$\partial_j \sigma'_{ij} - \partial_i P + \rho g = 0$$

- Non-linear viscosity law + elasticity + plasticity
- Conservation of energy

$$\frac{\partial T}{\partial t} + v \nabla T = \nabla \cdot (\kappa \nabla T) + H$$

Numerical methodology / Citations

- Finite Difference / Particle-in-Cell method.
Implementation largely based on *Gerya* [2010].



- Methodology partly described
in *Olive et al.* [2016, GJI]

Geophysical Journal International

Geophys. J. Int. (2016) **205**, 728–743
Advance Access publication 2016 January 27
GJI Geodynamics and tectonics

doi: 10.1093/gji/ggw044

The role of elasticity in simulating long-term tectonic extension

Jean-Arthur Olive,^{1,*} Mark D. Behn,² Eric Mittelstaedt,³ Garrett Ito⁴
and Benjamin Z. Klein⁵

Finite-difference discretization

Steady-state Stokes flow on Eulerian grid:

Conservative FD scheme
on a staggered grid:

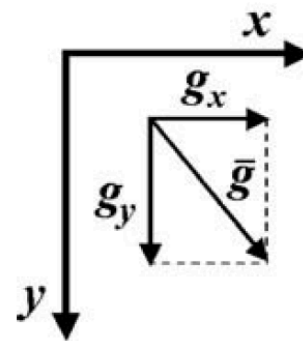
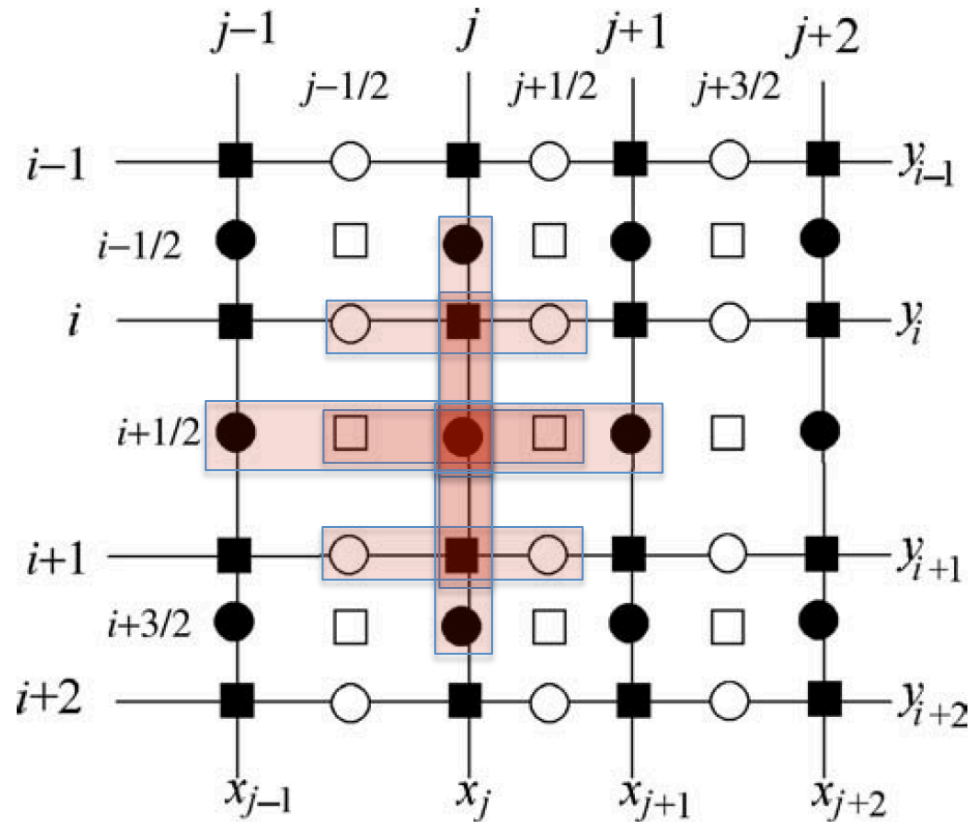
$$\nabla \cdot (\eta (\nabla \mathbf{v} + \nabla \mathbf{v}^T)) - \nabla P + \rho \mathbf{g} = \mathbf{RHS}$$

$$\nabla \cdot \mathbf{v} = 0$$

Leads to linear(ized) system:

$$\mathbf{JX} = \begin{pmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ P \end{pmatrix} = \begin{pmatrix} \mathbf{rhs} \\ 0 \end{pmatrix}$$

Solved with MATLAB's
“backslash” solver.



● v_x, q_y

○ v_y, q_x

■ $\dot{\epsilon}_{xy}, \sigma_{xy}, \eta_s, \mu_s, T, \rho, k, C_P, \alpha, H, \omega$

□ $P, \dot{\epsilon}_{xx}, \dot{\epsilon}_{yy}, \sigma'_{xx}, \sigma'_{yy}, \eta_n, \mu_n, \frac{D \ln \rho}{Dt}$

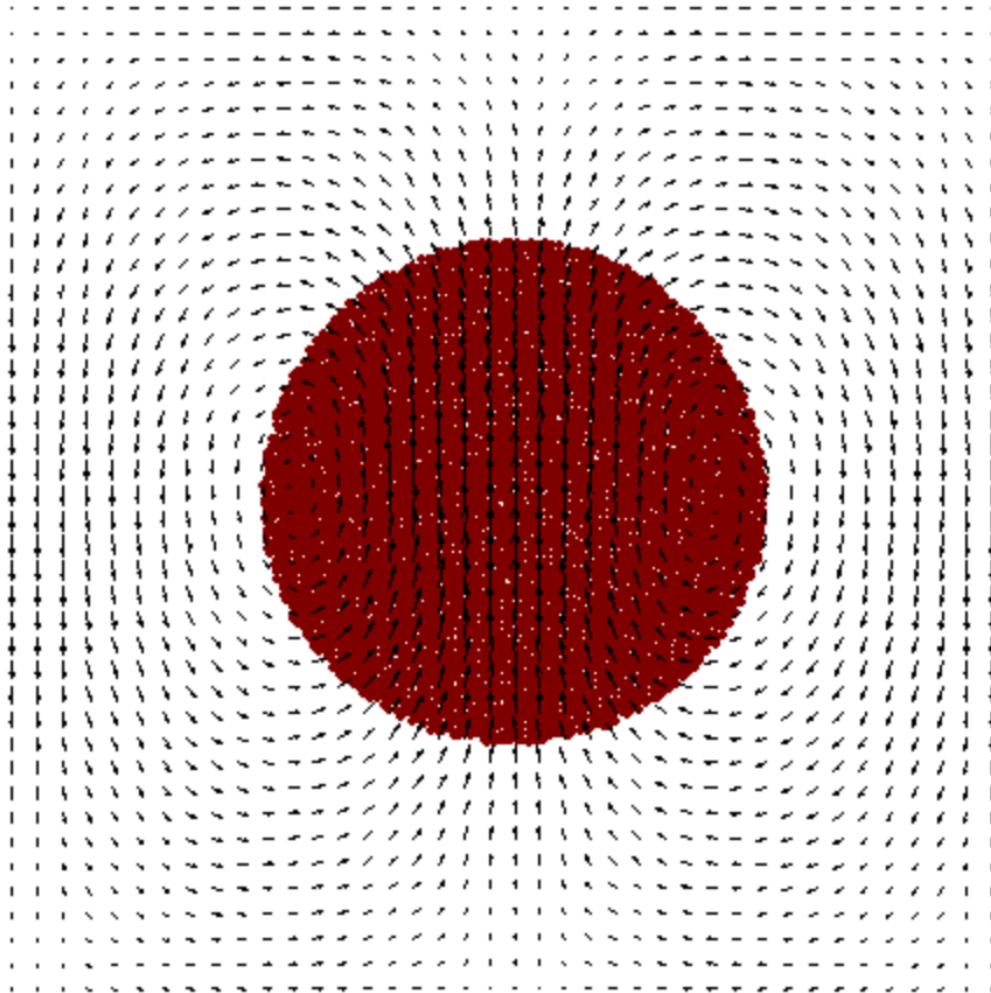
Gerya [2010]

Shear nodes

Normal nodes

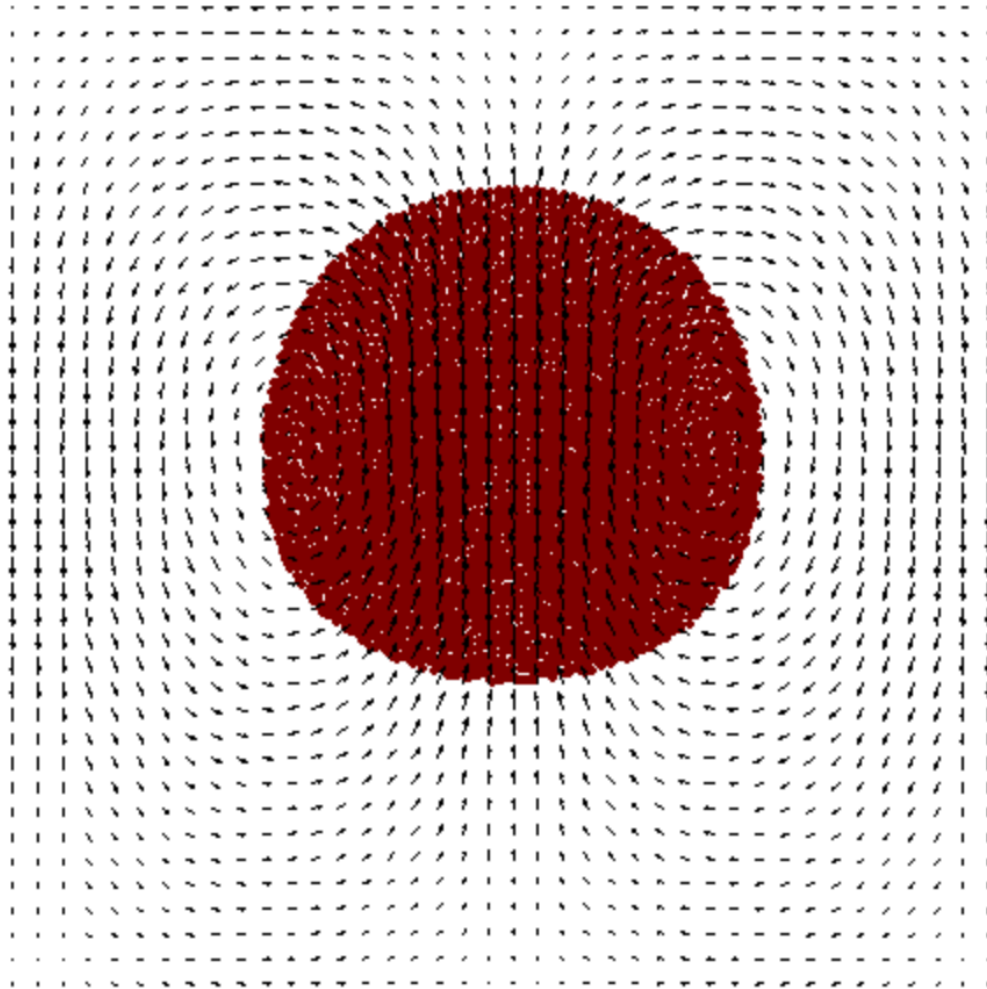
Particles handle material advection

- low density sphere in dense, low viscosity fluid:

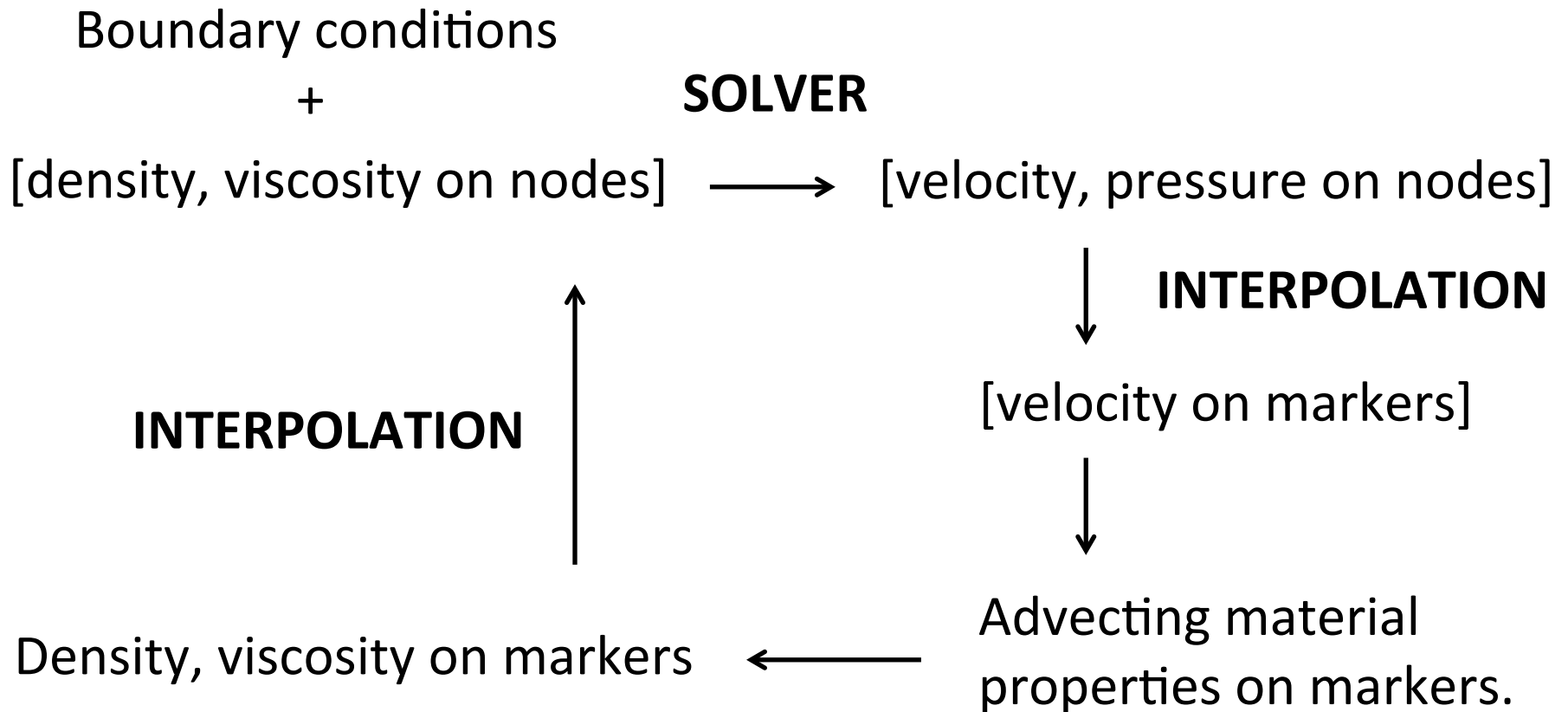


Particles handle material advection

- low density sphere in dense, low viscosity fluid:



Basic code structure



Handling non-linear creep laws

The formulation of viscosity strictly pertaining to non-linear creep laws writes:

$$\eta = \left(\frac{1}{\eta_{DIFF}} + \frac{1}{\eta_{DISC}} + \text{other terms} \right)^{-1}$$

This distinguishes the contributions from diffusion and dislocation creep.

Each creep law can be parameterized as: $\eta = A^{-\frac{1}{n}} \dot{\epsilon}_II^{\frac{1-n}{n}} e^{\frac{E}{nRT}}$

Picard iterations to handle non-linearity:

- 1/ Solve the flow equations using the latest viscosity map
- 2/ Calculate strain rate from latest solution
- 3/ Re-evaluate viscosity from the new strain rate map

Repeat until flow solution is accurate within specified tolerance.

Implementation of visco-elasticity

Maxwell **visco-elastic** rheology:

$$\dot{\epsilon}_{ij} = \underbrace{\frac{1}{2G} \frac{D\sigma_{ij}}{Dt}}_{\text{ELASTIC}} + \underbrace{\frac{1}{2\eta} \sigma_{ij}}_{\text{VISCOUS}}$$



FD approximation

$$\sigma_{ij}^{(t)} = 2Z\eta\dot{\epsilon}_{ij} + (1-Z)\sigma_{ij}^{(t-\Delta t)}$$

$$\text{with } Z = \frac{G\Delta t}{\eta + G\Delta t}$$

Implementation in a viscous solver:
[Moresi et al., 2003]

$$\nabla \cdot (2Z\eta\dot{\epsilon}_{ij}) - \nabla p + \rho g = -\nabla \cdot ((1-Z)\sigma_{ij}^{(t-\Delta t)})$$

$$\tau_{\text{Max}} = \frac{\eta}{G}$$

Maxwell time scale:

For **VISCOUS BEHAVIOR**
set very high G so that:
 $G\Delta t \gg \eta$, i.e., **$\Delta t \gg \tau_{\text{Max}}$**
(and $Z \approx 1$)

For **ELASTIC BEHAVIOR**
impose high viscosity.

Implementation of plasticity

Plastic (localizing) rheology

Mohr-Coulomb plasticity is implemented by weakening viscosity to cap σ'_{II} at σ_{yield} :

$$\eta = \left(\frac{1}{\eta_{REF}} + \frac{1}{\eta_{PLAS}} \right)^{-1} \quad \text{with} \quad \eta_{PLAS} = \frac{\sigma_{yield}}{2\dot{\epsilon}_{II}}$$

Yield stress prescribed through cohesion and friction

$$\sigma_{yield} = \sin\Phi \, \mathbf{P} + C \cos\Phi$$

Strain localization is promoted by **dropping cohesion** linearly with accumulated plastic strain ϵ_p . Full weakening occurs when fault offset reaches $h_C \approx 3 \Delta x \epsilon_{P\,CRIT}$ *Lavier et al. [2000]*

Accumulated plastic strain **heals** on a prescribed time scale

Solving visco-elasto-plastic flows

SiStER_flow_solve.m

Solution vector S contains (v_x, v_y, P) on nodes. Solution is inaccurate: has residual > 0 .

while residual $>$ tolerance

- Assemble the **nodal viscosity array** from current maps of material type, temperature, pressure, cohesion, friction, shear modulus...
Viscosity = harmonic average of ductile creep viscosities and plastic viscosity
Use **ductile creep laws** (SiStER_get_ductile_rheology-) formulated in terms of **strain rate** if elasticity is OFF, **deviatoric stresses** if elasticity is ON.
If elasticity is ON, a bisection algorithm helps evaluating stresses for use in the creep law (SiStER_viscosity_stress_bisection).
Plastic viscosity = yield stress / inelastic strain rate.
- Assemble additional correction terms relevant for elasto-plasticity.
(SiStER_VEP_rheology)
- Assemble **finite difference matrix** L from viscosity and density arrays + boundary conditions. Assemble **right-hand side vector** R from stresses. Matrix equation $L \cdot S = R$ represents discretized Stokes equation within Picard approximation. (SiStER_assemble_L_R)
- **Direct solve** for solution update:
 $S = S - L \setminus \text{residual}$
 $\text{residual} = L \cdot S - R$
- Calculate **strain rate** from newest solution.

Successive estimates of strain rate / stress help refine the estimate of viscosity and reach a consistent solution.

Detailed code structure (SiStER_MAIN.m)

Read input parameters from • SiStER_Input_File

• Initialization step (SiStER_Initialize). Set up grid, initial marker positions and material types

for t = 1:Nt

• Construct nodal maps of density, built-up elastic stresses, friction, cohesion, and shear modulus (interpolated from markers) for the next flow solve. (SiStER_material_props_on_nodes)

• Iterate to reach a **steady-state flow solution** (\mathbf{v}_x , \mathbf{v}_y , \mathbf{P}) consistent with current boundary conditions, viscosity, density, and built-up elastic stresses. (SiStER_flow_solve)

• Add an **increment of elastic stress** corresponding to the latest flow solution to all markers (SiStER_update_marker_stresses).

• If plasticity is ON, add an **increment of plastic strain** to markers wherever plastic yielding occurs. Allow an increment of strain **healing** everywhere. (SiStER_update_ep)

• **Output** variables of interest

• Set **time step** Δt such that markers don't move by more than a fraction of the grid size at a time.

• If elasticity is ON, **rotate deviatoric stresses** (on markers) in current flow field over Δt .

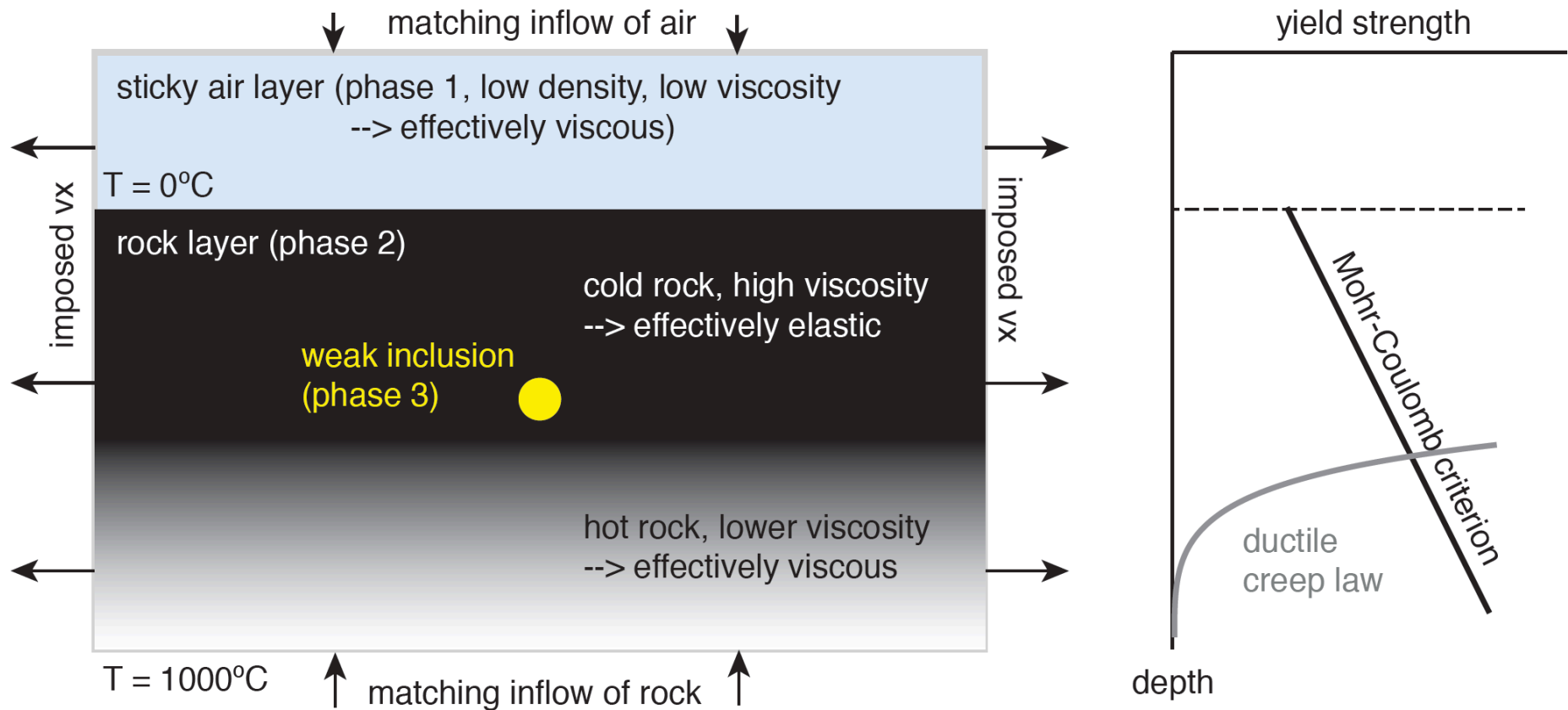
• Solve the non-advective part of the **heat equation**. (SiStER_thermal_update)

• **Advect** markers in latest flow field, over Δt . **Eliminate** markers which have left the model domain. **Seed** new markers in cell quadrants where marker density has fallen below a specified threshold. (SiStER_move_remove_and_reseed_markers)

• Advect **topographic marker chain** in latest flow field over Δt . Apply **surface processes** if specified. (SiStER_update_topography_markers)

Example case: rifting example

Run **SiStER_MAIN**, then enter **SiStER_Input_File_continental_rift**



Input file structure

Runs for 100 iterations, will output a file every 10 time steps

```
3
4  % DURATION OF SIMULATION AND FREQUENCY OF OUTPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 -  Nt=100; % max number of time iterations
6 -  dt_out=10; % output files every "dt_out" iterations
7
8  Top-left corner of the domain at (0,0), x>0 to the right, y>0 down
9  % DOMAIN SIZE AND GRIDDING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 -  xsize=90e3;
11 -  ysize=30e3;
12  % gridding- from 0 to GRID.x(1), grid size is GRID.dx(1)
13  % from GRID.x(1) to GRID.x(2), grid size is GRID.dx(1) etc...
14  % same for y
15 -  GRID.dx(1)=2000;
16 -  GRID.x(1)=30e3;
17 -  GRID.dx(2)=400;
18 -  GRID.x(2)=60e3;
19 -  GRID.dx(3)=2000;
20 -  GRID.dy(1)=2000;
21 -  GRID.y(1)=9e3;
22 -  GRID.dy(2)=400;
23 -  GRID.y(2)=22e3;
24 -  GRID.dy(3)=2000;
25
```

Box size 90 km × 30 km All units SI

Horizontal grid size = 2km between 0 and 30 km, 400m between 30 and 60 km, 2 km between 60 and 90 km.

Vertical grid size = 2km between 0 and 9 km, 400m between 9 and 22 km, 2 km between 22 and 30 km.

Input file structure

Seeding markers to advect material properties:

```
26  
27 % LAGRANGIAN MARKERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
28 - Mquad=8; % number of markers in the smallest quadrant  
29 - Mquad_crit=4; % minimum number of markers allowed in smallest quadrant (for reseeding)  
30
```

The initial marker density is defined such that a quarter of the smallest cell (defined above) has 8 markers, i.e., the smallest cell contains 32 markers. Bigger cells will contain more markers such that the density of markers (number of markers per m^2) is constant.

As markers move around, gaps may form in the marker distribution. New markers will be added to fill in the gaps in cells that have quadrants with less than 4 markers.

Input file structure

Setting up the initial geometry of the phases (material types)

Here there are 3 phases in total: a sticky air layer, a rock layer and a weak rock inclusion

```
30
31 % GEOMETRY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33 - Nphase=3; % number of phases
34
35 % phase 1
36 - GEOM(1).type=1; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
37 - GEOM(1).top=0;
38 - GEOM(1).bot=10e3;
39
40 % phase 2
41 - GEOM(2).type=1; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
42 - GEOM(2).top=10e3;
43 - GEOM(2).bot=30e3;
44
45 % phase 3
46 - GEOM(3).type=2; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
47 - GEOM(3).x0=xsize/2;
48 - GEOM(3).y0=20e3;
49 - GEOM(3).rad=1e3;
50
```

First phase (air) is a horizontal layer (type 1) spanning the whole width of the box, between $y=0$ (top) and $y=10$ km (bottom).

Input file structure

```
30
31 % GEOMETRY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33 Nphase=3; % number of phases
34
35 % phase 1
36 GEOM(1).type=1; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
37 GEOM(1).top=0;
38 GEOM(1).bot=10e3;
39
40 % phase 2
41 GEOM(2).type=1; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
42 GEOM(2).top=10e3;
43 GEOM(2).bot=30e3;
44
45 % phase 3
46 GEOM(3).type=2; % 1 = layer (then specify top and bot) or 2 = circle % 1 = layer (then specify top and bot) or 2 = circle
47 GEOM(3).x0=xsize/2;
48 GEOM(3).y0=20e3;
49 GEOM(3).rad=1e3;
50
```

Second phase (rock) is a horizontal layer (type 1) spanning the whole width of the box, between $y=10$ (top) and $y=30$ km (bottom).

Third phase (weak rock) is a circle (type 2) of radius 1 km centered at (45 km, 20 km).

Phase definitions overprint each other in the order they are defined- make sure they cover the entire domain!

Input file structure

Material properties must be specified for each phase defined above, in the following format (shown here for phase 1)

```
51
52 % MATERIAL PROPERTIES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53
54 % creep laws of the form: pre^(-1/n)*epsII^((1-n)/n)*exp(E/(nRT))
55 % harmonically averaging diffusion creep, dislocation creep
56 % (and plastic creep to simulate brittle failure)
57
58 % phase 1
59 - MAT(1).phase=1;
60 % density parameters
61 - MAT(1).rho0=0.01;
62 - MAT(1).alpha=0;
63 % thermal parameters
64 - MAT(1).k=3;
65 - MAT(1).cp=1000;
66 % elasticity
67 - MAT(1).G=1e18;
68 % diffusion creep parameters
69 - MAT(1).pre_diff=.5/1e18;
70 - MAT(1).Ediff=0;
71 - MAT(1).ndiff=1;
72 % dislocation creep parameters
73 - MAT(1).pre_disc=.5/1e18;
74 - MAT(1).Edisc=0;
75 - MAT(1).ndisc=1;
76 % plasticity
77 - MAT(1).mu=0.6;
78 - MAT(1).mumin=0.3;
79 - MAT(1).Cmax=40e6;
80 - MAT(1).Cmin=0.01e6;
81 - MAT(1).ecrit=0.1;
```

Phase 1 will be indexed in marker array im with value 1

Reference density and thermal expansion, see
SiStER_get_density function

Parameters for heat diffusion: conductivity & cp, see
SiStER_get_thermal_properties

Parameters for diffusion creep law

Parameters for dislocation creep law

See **SiStER_get_ductile_rheology...** functions

Parameters for plasticity (faulting),
see **SiStER_get_friction / cohesion** functions

Input file structure

Material properties must be specified for each phase defined above, in the following format (shown here for phase 1)

```
51
52 % MATERIAL PROPERTIES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53
54 % creep laws of the form: pre^(-1/n)*epsII^((1-n)/n)*exp(E/(nRT))
55 % harmonically averaging diffusion creep, dislocation creep
56 % (and plastic creep to simulate brittle failure)
57
58 % phase 1
59 - MAT(1).phase=1;
60 % density parameters
61 - MAT(1).rho0=0.01;
62 - MAT(1).alpha=0;
63 % thermal parameters
64 - MAT(1).k=3;
65 - MAT(1).cp=1000;
66 % elasticity
67 - MAT(1).G=1e18;
68 % diffusion creep parameters
69 - MAT(1).pre_diff=.5/1e18;
70 - MAT(1).Ediff=0;
71 - MAT(1).ndiff=1;
72 % dislocation creep parameters
73 - MAT(1).pre_disc=.5/1e18;
74 - MAT(1).Edisc=0;
75 - MAT(1).ndisc=1;
76 % plasticity
77 - MAT(1).mu=0.6;
78 - MAT(1).mumin=0.3;
79 - MAT(1).Cmax=40e6;
80 - MAT(1).Cmin=0.01e6;
81 - MAT(1).ecrit=0.1;
```

NOTES ON PHASE 1 (sticky air)

Note the low density
+ no thermal dependence of density

Unrealistically high shear modulus ensures
an effectively viscous behavior

Input file structure

Material properties must be specified for each phase defined above, in the following format (shown here for phase 1)

```
51
52 % MATERIAL PROPERTIES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53
54 % creep laws of the form: pre^(-1/n)*epsII^((1-n)/n)*exp(E/(nRT))
55 % harmonically averaging diffusion creep, dislocation creep
56 % (and plastic creep to simulate brittle failure)
57
58 % phase 1
59 - MAT(1).phase=1;
60 % density parameters
61 - MAT(1).rho0=0.01;
62 - MAT(1).alpha=0;
63 % thermal parameters
64 - MAT(1).k=3;
65 - MAT(1).cp=1000;
66 % elasticity
67 - MAT(1).G=1e18;
68 % diffusion creep parameters
69 - MAT(1).pre_diff=.5/1e18; ←
70 - MAT(1).Ediff=0;
71 - MAT(1).ndiff=1;
72 % dislocation creep parameters
73 - MAT(1).pre_disc=.5/1e18; ←
74 - MAT(1).Edisc=0;
75 - MAT(1).ndisc=1;
76 % plasticity
77 - MAT(1).mu=0.6;
78 - MAT(1).mumin=0.3;
79 - MAT(1).Cmax=40e6;
80 - MAT(1).Cmin=0.01e6;
81 - MAT(1).ecrit=0.1;
```

NOTES ON PHASE 1 (sticky air)

To enforce a constant Newtonian viscosity η_0 that does not depend on temperature, set $n = 1$, $E=0$, and prefactor = $0.5/\eta_0$ in both dislocation and diffusion creep laws

Input file structure

Material properties must be specified for each phase defined above, in the following format (shown here for phase 2)

```
84 % phase 2
85 - MAT(2).phase=2;
86 % density parameters
87 - MAT(2).rho0=2700;
88 - MAT(2).alpha=0;
89 % thermal parameters
90 - MAT(2).k=3;
91 - MAT(2).cp=1000;
92 % elasticity
93 - MAT(2).G=30e9;
94 % diffusion creep parameters
95 - MAT(2).pre_diff=.5/1e40;
96 - MAT(2).Ediff=0;
97 - MAT(2).ndiff=1;
98 % dislocation creep parameters
99 - MAT(2).pre_disc=1.0e-3;
100 - MAT(2).Edisc=167000*2.25;
101 - MAT(2).ndisc=2;
102 % plasticity
103 - MAT(2).mu=0.6;
104 - MAT(2).mumin=0.3;
105 - MAT(2).Cmax=40e6;
106 - MAT(2).Cmin=0.01e6;
107 - MAT(2).ecrit=0.1;
108
```

NOTES ON PHASE 2 (rock)

Very low prefactor in diffusion creep
to select only dislocation creep

Non-Newtonian creep law
with T-dependence

Plasticity parameters follow the formalism
of *Lavier et al.* [2000, JGR]: cohesion and friction
weaken from Cmax to Cmin and mu to mumin
when plastic strain reaches ecrit

Input file structure

Material properties must be specified for each phase defined above, in the following format (shown here for phase 3)

```
109
110 % phase 3
111 - MAT(3).phase=3;
112 % density parameters
113 - MAT(3).rho0=2700;
114 - MAT(3).alpha=0;
115 % thermal parameters
116 - MAT(3).k=5;
117 - MAT(3).cp=1000;
118 % elasticity
119 - MAT(3).G=30e9;
120 % diffusion creep parameters
121 - MAT(3).pre_diff=.5/1e40;
122 - MAT(3).Ediff=0;
123 - MAT(3).ndiff=1;
124 % dislocation creep parameters
125 - MAT(3).pre_disc=1.0e-3;
126 - MAT(3).Edisc=167000*2.25;
127 - MAT(3).ndisc=2;
128 % plasticity
129 - MAT(3).mu=0.6;
130 - MAT(3).mumin=0.3;
131 - MAT(3).Cmax=0.01e6;
132 - MAT(3).Cmin=0.01e6;
133 - MAT(3).ecrit=0.1;
```

NOTES ON PHASE 3 (weak rock)

Identical to phase 2,
but with very low initial cohesion
to promote rapid yielding of this weak seed

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNElast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNPlas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

Simulation incorporate elasticity and plasticity (brittle strain localization)

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNlast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNPlas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

Accumulated plastic strain (in weak yielded zones, i.e., faults) heals exponentially on a time scale specified here, in seconds [Lavier *et al.*, 2000, JGR]. This way, old, abandoned faults where no strain localization is sustained can disappear.

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNElast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNPlas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

Gravity in m/s^2

(both x and y components, i.e., cases with sloping gravity are possible)

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNlast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNplas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

The time step (used for marker advection) is set such that markers cannot move by more than $\text{PARAMS.fracCFL} \times$ the smallest grid size, at every time iteration (= CFL condition for advection).

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNlast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNPlas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

Hard limits on the viscosity

Here it cannot drop below 1e18 Pa.s, or exceed 1e25 Pa.s.

Input file structure

More parameters...

```
136 % ADDITIONAL PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 - PARAMS.YNlast=1; % elasticity on (1) or off (0)
138 - PARAMS.YNplas=1; % plasticity on (1) or off (0)
139 - PARAMS.tau_heal=1e12; % healing time for plasticity (s)
140 - PARAMS.gx=0; % gravity along x
141 - PARAMS.gy=9.8; % gravity along y
142 - PARAMS.fracCFL=0.5; % distance by which a marker is allowed to move over a time step
143 - PARAMS.R=8.314; % gas constant
144 - PARAMS.etamax=1e25; % maximum viscosity
145 - PARAMS.etamin=1e18; % minimum viscosity
146 - PARAMS.Tsolve=1; % yes (1) or no (0) solve for temperature
```

If set to 1, the temperature field will evolve by advection (in the solid flow field) and diffusion.

If set to 0, it will only be advected- the heat equation will not be solved.

Input file structure

Thermal parameters

```
147 % initial temperature profile, polynomial with depth
148 %  $T = a_0 + a_1 y + a_2 y^2 + a_3 y^3 + \text{amp} \cdot \sin(2 \cdot \pi \cdot X / \text{lam})$ 
149 % (make sure it matches the BCs)
150 - PARAMS.a0=0;
151 - PARAMS.a1=0;
152 - PARAMS.a2=0;
153 - PARAMS.a3=1000/(30e3)^3;
154 - PARAMS.amp=0; % amplitude of sinusoidal perturbation
155 - PARAMS.lam=1; % wavelength of sinusoidal perturbation
156 - PARAMS.ynTreset=1; % if ==1, reset  $T=T_0$  where  $\text{im}=1$  (sticky layer)
157 - PARAMS.T0=0;
158 % reference values for the constant diffusivity thermal solver
159 % ( $\text{kappa} = \text{kref} / (\text{rhoref} \cdot \text{cpref})$ )
160 - PARAMS.rhoref=MAT(2).rho0;
161 - PARAMS.kref=3;
162 - PARAMS.cpref=1000;
```

The initial temperature structure follows this functional form:

$$T = a_0 + a_1 y + a_2 y^2 + a_3 y^3 + \text{amp} \cdot \sin(2 \cdot \pi \cdot x / \text{lam})$$

See **SiStER_Initialize.m** – Make sure it is not incompatible with the imposed boundary conditions! And do not forget that $y=0$ is the top of the domain, not of the rock layer (it includes the sticky layer)

Input file structure

Thermal parameters

```
147 % initial temperature profile, polynomial with depth
148 %  $T = a_0 + a_1 y + a_2 y^2 + a_3 y^3 + \text{amp} \sin(2\pi X / \text{lam})$ 
149 % (make sure it matches the BCs)
150 - PARAMS.a0=0;
151 - PARAMS.a1=0;
152 - PARAMS.a2=0;
153 - PARAMS.a3=1000/(30e3)^3;
154 - PARAMS.amp=0; % amplitude of sinusoidal perturbation
155 - PARAMS.lam=1; % wavelength of sinusoidal perturbation
156 - PARAMS.ynTreset=1; % if ==1, reset  $T=T_0$  where  $\text{im}=1$  (sticky layer)
157 - PARAMS.T0=0;
158 % reference values for the constant diffusivity thermal solver
159 % ( $\text{kappa} = \text{kref} / (\text{rhoref} * \text{cpref})$ )
160 - PARAMS.rhoref=MAT(2).rho0;
161 - PARAMS.kref=3;
162 - PARAMS.cpref=1000;
```

With `PARAMS.ynTreset = 1`, the temperature field will be automatically set to `PARAMS.T0` throughout the sticky layer. This is a way to enforce T_0 at the air-rock interface.

Input file structure

Thermal parameters

```
147 % initial temperature profile, polynomial with depth
148 %  $T = a_0 + a_1*y + a_2*y^2 + a_3*y^3 + \text{amp}*\sin(2*\pi*X/\text{lam})$ 
149 % (make sure it matches the BCs)
150 - PARAMS.a0=0;
151 - PARAMS.a1=0;
152 - PARAMS.a2=0;
153 - PARAMS.a3=1000/(30e3)^3;
154 - PARAMS.amp=0; % amplitude of sinusoidal perturbation
155 - PARAMS.lam=1; % wavelength of sinusoidal perturbation
156 - PARAMS.ynTreset=1; % if ==1, reset  $T=T_0$  where  $\text{im}=1$  (sticky layer)
157 - PARAMS.T0=0;
158 % reference values for the constant diffusivity thermal solver
159 % ( $\text{kappa} = \text{kref} / (\text{rhoref}*\text{cpref})$ )
160 - PARAMS.rhoref=MAT(2).rho0;
161 - PARAMS.kref=3;
162 - PARAMS.cpref=1000;
```

If MAT.cp and MAT.k are not specified in the material properties above, the code uses these reference values to calculate a reference diffusivity ($k/(\rho*cp)$) constant over the entire domain.

Input file structure

Surface processes

```
164 % TOPOGRAPHY EVOLUTION (interface between rock and sticky air/water layer)
165 - PARAMS.Ntopo_markers=1000; % number of markers in marker chain tracking topography
166 - PARAMS.YNSurfaceProcesses=1; % surface processes (diffusion of topography) on or off
167 - PARAMS.topo_kappa=1e-8; % diffusivity of topography (m^2/s)
168
```

A chain of Ntopo_markers closely spaced markers is initially seeded between phase 1 and phase 2, to keep track of topography.

If YNSurfaceProcesses is set to 0, this marker chain passively tracks the sticky layer – rock interface. If set to 1, it can be diffused with a diffusivity topo_kappa to simulate simple erosion / deposition acting over the deforming landscape.

Input file structure

Picard iterations parameters

```
170 % Solver iterations
171 - PARAMS.Npicard_min=10; % minimum number of Picard iterations per time step
172 - PARAMS.Npicard_max=100; % maximum number of Picard iterations per time step
173 - PARAMS.conv_crit_ResL2=1e-9;
174 - PARAMS.pitswitch=0; % number of Picard iterations at which the solver switches to quasi-Newton
```

This run will always use at least 10 Picard iterations, but will stop after 100, regardless of the convergence criterion.

Input file structure

Picard iterations parameters

```
170 % Solver iterations
171 - PARAMS.Npicard_min=10; % minimum number of Picard iterations per time step
172 - PARAMS.Npicard_max=100; % maximum number of Picard iterations per time step
173 - PARAMS.conv_crit_ResL2=1e-9;
174 - PARAMS.pitswitch=0; % number of Picard iterations at which the solver switches to quasi-Newton
```

Convergence is considered achieved when the normalized residual of the non-linear system falls below $1e-9$, see `SiStER_flow_solve.m`

Input file structure

Picard iterations parameters

```
170 % Solver iterations
171 - PARAMS.Npicard_min=10; % minimum number of Picard iterations per time step
172 - PARAMS.Npicard_max=100; % maximum number of Picard iterations per time step
173 - PARAMS.conv_crit_ResL2=1e-9;
174 - PARAMS.pitswitch=0; % number of Picard iterations at which the solver switches to quasi-Newton
```

Eventually this feature will be used to switch to other solve strategies.
For now it should be left =0

Input file structure

Boundary conditions (pressure)

```
176  
177 % BOUNDARY CONDITIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
178  
179 % pressure  
180 - PARAMS.p0cell=0; % pressure in the top-left corner of the domain (anchor point)
```

Pressure is set to 0 in the top-left corner of the domain.
This is done to anchor the solution to a specific pressure
and stabilize the inversion.

Input file structure

```
183 % flow      Boundary conditions (flow)
184
185 % boundary conditions
186 % entries in BC correspond to
187 % 1/ rollers? (1=yes, 0=no)
188 % 2/ type of velocity normal to boundary (0=constant)
189 % 3/ value of normal velocity
190
191 - BC.top=[1 0 1.0563e-10];
192 - BC.bot=[1 0 -1.0563e-10];
193 - BC.left=[1 0 -3.1688e-10];
194 - BC.right=[1 0 3.1688e-10];
195 - PARAMS.BalanceStickyLayer=1; % if set to 1, the code will reset the inflow
196 % / outflow BCs to balance the inflow / outflow of sticky layer material,
197 % and rock separately, based on the position of the sticky layer / air
198 % interface
```

BCs are specified in this BC structure for the top, bottom, left and right edges of the domain. The first entry is 1 for free slip ($d v_{\text{tangential}} / d \text{normal} = 0$), 0 for no-slip ($v_{\text{tangential}} = 0$)

Input file structure

```
183 % flow      Boundary conditions (flow)
184
185 % boundary conditions
186 % entries in BC correspond to
187 % 1/ rollers? (1=yes, 0=no)
188 % 2/ type of velocity normal to boundary (0=constant)
189 % 3/ value of normal velocity
190
191 - BC.top=[1 0 1.0563e-10];
192 - BC.bot=[1 0 -1.0563e-10];
193 - BC.left=[1 0 -3.1688e-10];
194 - BC.right=[1 0 3.1688e-10];
195 - PARAMS.BalanceStickyLayer=1; % if set to 1, the code will reset the inflow
196 % / outflow BCs to balance the inflow / outflow of sticky layer material,
197 % and rock separately, based on the position of the sticky layer / air
198 % interface
```

The second entry specifies the type of normal velocity BC – it can only be 0 in the current version, which enforces a constant normal velocity (m/s), specified in the third entry (careful with sign conventions, and make sure inflow matched outflow!)

Input file structure

```
183 % flow      Boundary conditions (flow)
184
185 % boundary conditions
186 % entries in BC correspond to
187 % 1/ rollers? (1=yes, 0=no)
188 % 2/ type of velocity normal to boundary (0=constant)
189 % 3/ value of normal velocity
190
191 - BC.top=[1 0 1.0563e-10];
192 - BC.bot=[1 0 -1.0563e-10];
193 - BC.left=[1 0 -3.1688e-10];
194 - BC.right=[1 0 3.1688e-10];
195 - PARAMS.BalanceStickyLayer=1; % if set to 1, the code will reset the inflow
196 % / outflow BCs to balance the inflow / outflow of sticky layer material,
197 % and rock separately, based on the position of the sticky layer / air
198 % interface
```

In this example, a velocity of 1 cm/yr is applied towards the right on the right edge, and – 1 cm/yr on the left edge.

The top and bottom velocities are set to balance this outflow.

Input file structure

```
183 % flow      Boundary conditions (flow)
184
185 % boundary conditions
186 % entries in BC correspond to
187 % 1/ rollers? (1=yes, 0=no)
188 % 2/ type of velocity normal to boundary (0=constant)
189 % 3/ value of normal velocity
190
191 - BC.top=[1 0 1.0563e-10];
192 - BC.bot=[1 0 -1.0563e-10];
193 - BC.left=[1 0 -3.1688e-10];
194 - BC.right=[1 0 3.1688e-10];
195 - PARAMS.BalanceStickyLayer=1; % if set to 1, the code will reset the inflow
196 % / outflow BCs to balance the inflow / outflow of sticky layer material,
197 % and rock separately, based on the position of the sticky layer / air
198 % interface
```

Setting this parameter to 1 will balance the inflow of sticky air and rock separately, based on the position of the air-rock interface at any given time (See **SiStER_flow_solve**)

Input file structure

Boundary conditions (temperature)

```
201      % thermal
202
203      % entries in BCtherm correspond to
204      % 1/ type? (1=Dirichlet, 2=Neumann)
205      % 2/ value
206 -    BCtherm.top=[1 0];
207 -    BCtherm.bot=[1 1000];
208 -    BCtherm.left=[2 0];
209 -    BCtherm.right=[2 0];
```

Those will be used if PARAMS.Tsolve is set to 1.

If the first entry is 1 (Dirichlet), the temperature is set equal to the second entry. If the first entry is 2 (Neumann), the gradient of temperature is set equal to the second entry.

This example has $T = 0$ on the top, $T=1000^{\circ}\text{C}$ on the bottom, and no heat flux through the sides.

Visualizing the output

Output files contain (among other things):

time (in seconds)

Nodal arrays:

X, Y (coordinates of shear nodes)

vx, vy (velocities) and p (pressure)

Marker arrays:

xm, ym (marker coordinates)

im (phase) etam (viscosity) rhom (density) Tm (temperature)

ep (accumulated plastic strain) sxxm, sxym (deviatoric stresses)

epsllm (strain rate)

Topography:

topo_x, topo_y (coordinates of a dense chain of markers tracking the sticky layer – rock interface)

More variables can be output- see l. 58 of **SiStER_MAIN**

Visualizing the output

Useful matlab script to plot large marker arrays = **fastscatter**

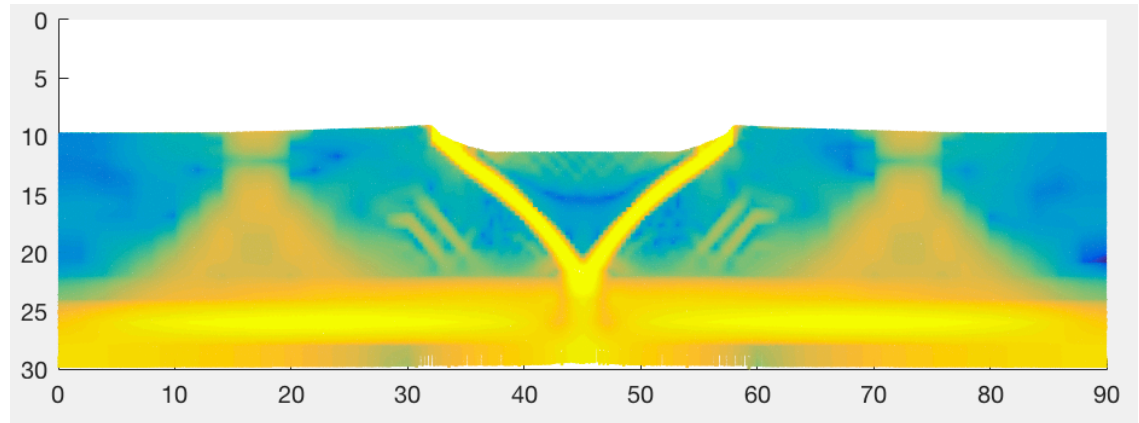
Can be downloaded from A. Grinsted on MATLAB File Exchange

<https://www.mathworks.com/matlabcentral/fileexchange/47205-fastscatter-m>

Visualizing the output

To visualize marker arrays (here the strain rate at iteration 50, excluding the sticky air layer):

```
>> fastscatter(xm(im>1)/1e3,ym(im>1) /1e3,log10(epsIIm(im>1)),'markersize',2);  
>> axis ij  
>> axis equal  
>> grid off  
>> colorbar  
>> caxis([-18 -13])
```



And the topography:

```
>> plot(topo_x,topo_y, 'k')  
>> set(gca,'YDir','reverse')
```

