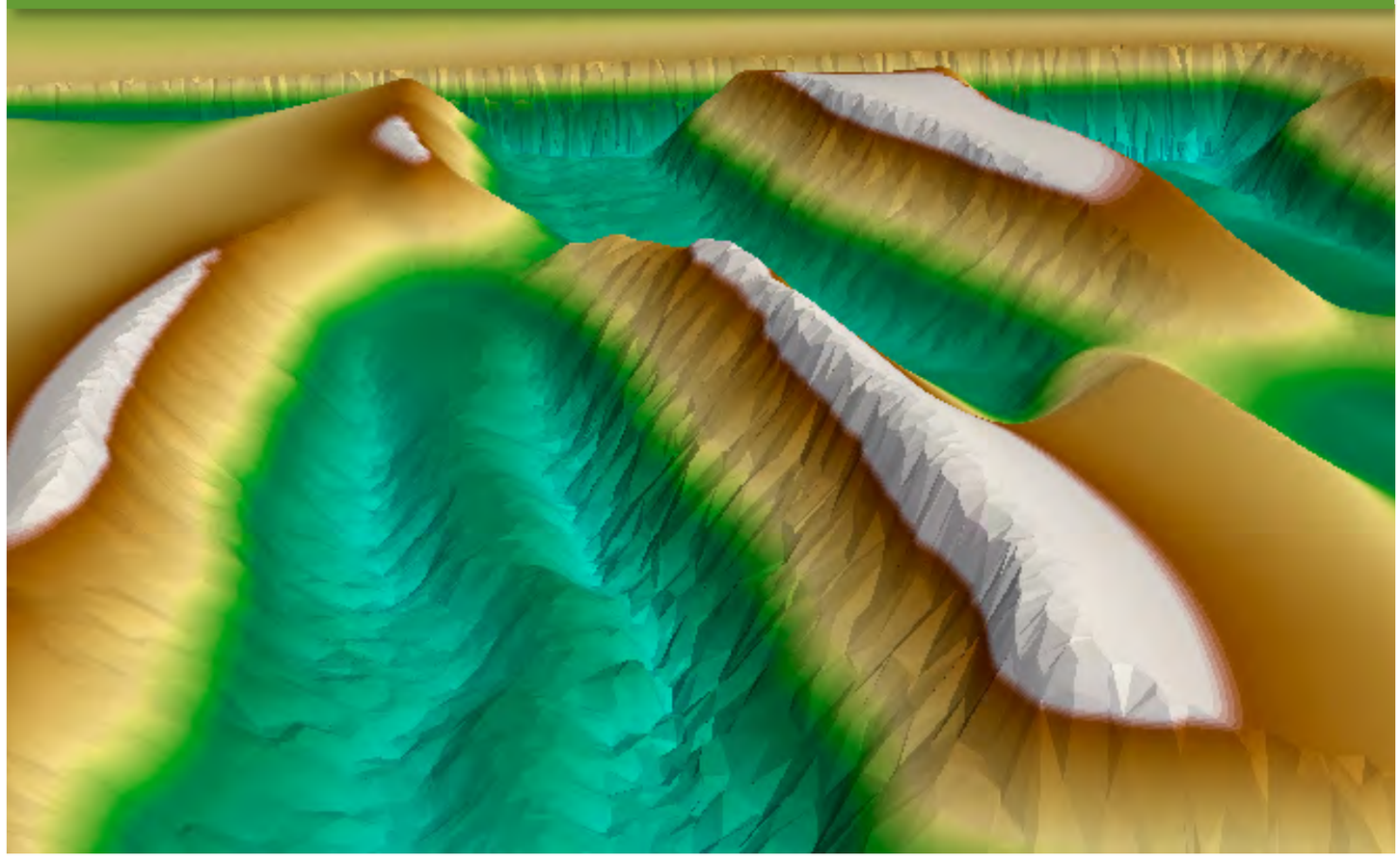
Computational Tectonics with *Underworld*



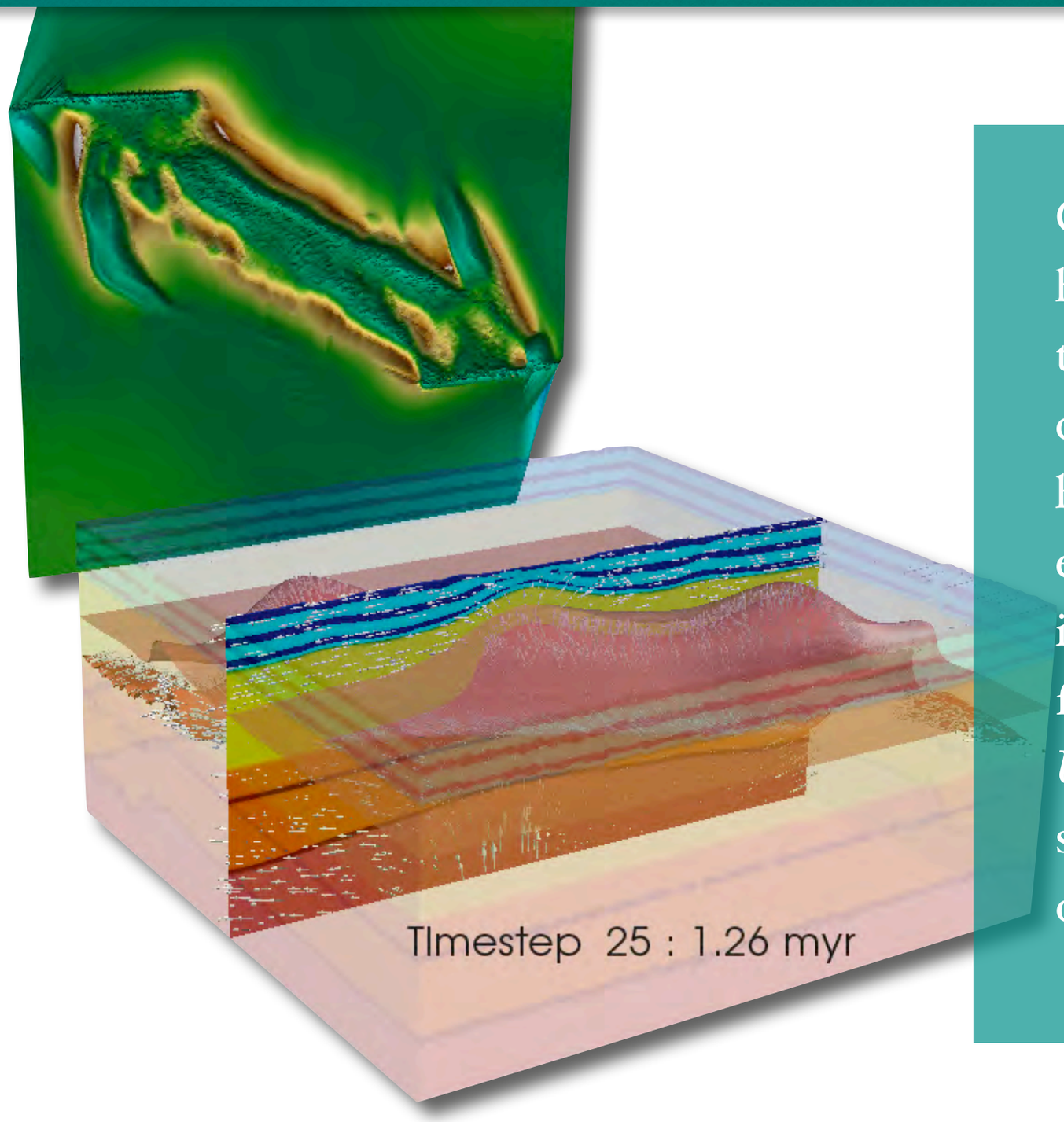
Patrice F. Rey

Basin Genesis Hub

Earthbyte Research Group

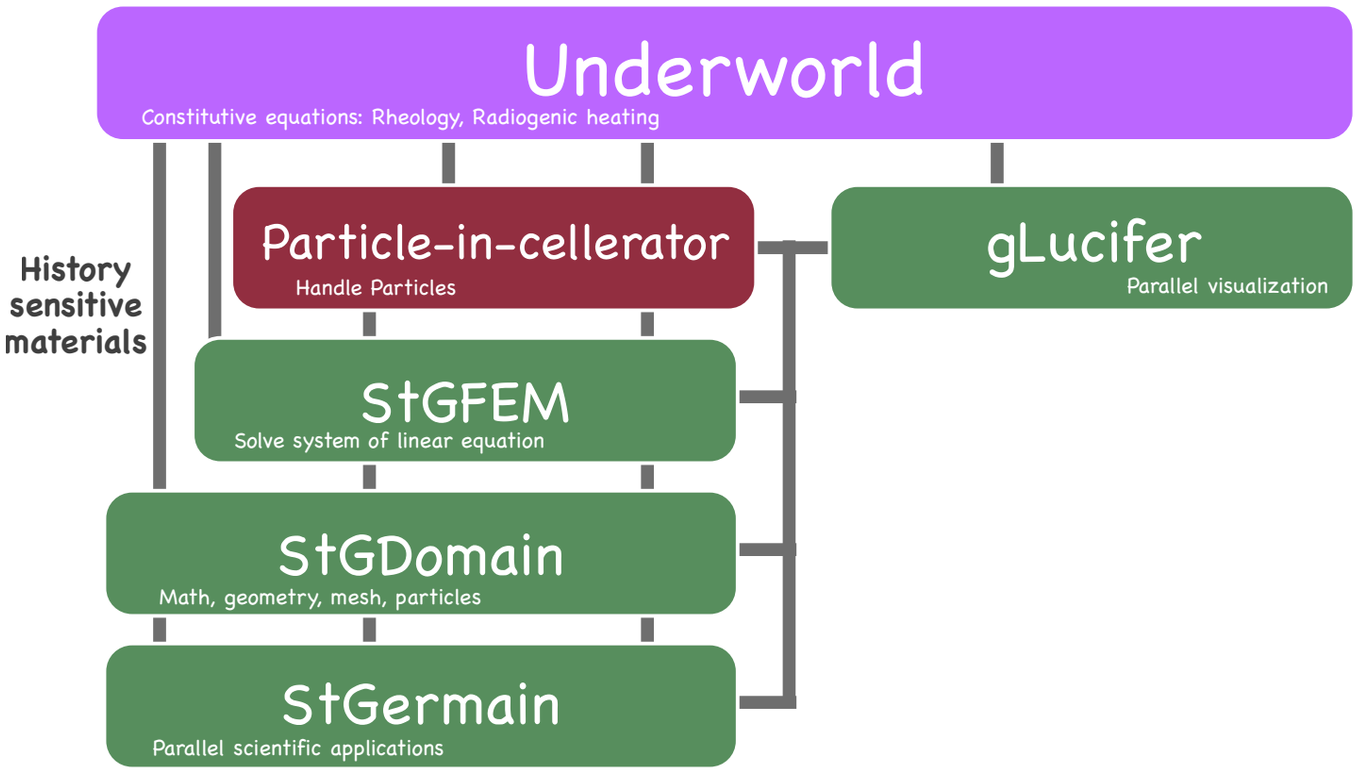
Chapter 1

Underworld



Growing access to high-performance computers, and their relentless increase in power are changing the way we do Science. We have now the numerical capability to explore complex non-linear processes in a well constrained physical framework. To tectonicists, *Underworld* is a physics laboratory; a space where hypothesis testing can be done through experiments.

What is it?



Contributors to Underworld (alphabetic order): Cecile Duboz, Rebecca Farrington, Justin Freeman, Julian Giordani, Luke Hodkinson, Megan Hough, Kathleen Humble, Owen Kaluza, Walter Landry, David Lee, Vincent Lemiale, Alan Lo, John Mansour, Wendy Mason, Belinda May, David May, Catherine Meriaux, Luke Mondy, Louis Moresi, Steve Quenette, Jerico Revote, Wendy Sharples, John Spencer, Jay Staord, Dave Stegman, Patrick Sunter, Robert Turnbull, Mirko Velic, Sergio Zlotnik.

Underworld **is a modular, hierarchical, computational framework, designed to perform parallel 2D-3D time-dependent coupled thermal and mechanical tectonic and geodynamic experiments. It uses an expandable range of rheologies covering elastic, plastic and viscous behaviors. It accommodates radiogenic (i.e. internal) heating, partial melting - and other metamorphic reactions - with feedback on rheologies, densities and temperature.**

Underworld **development is lead by Prof. L. Moresi at the University of Melbourne.** ***Underworld*** **started in 2005 with funding from the Victorian Partnership of Advanced Computing, and carried on with support from NCRIS/Auscope program.** ***Underworld*** **development is now supported by ARC-ITRH Basin Genesis Hub.**

Underworld key reference papers...

ELSEVIER

Physics of the Earth and Planetary Interiors 163 (2007) 69–82

PHYSICS OF THE EARTH AND PLANETARY INTERIORS
www.elsevier.com/locate/pepi

Computational approaches to studying non-linear dynamics of the crust and mantle

L. Moresi [a,*], S. Quenette [b], V. Lemiale [a], C. Mériaux [a], B. Appelbe [b], H.-B. Mühlhaus [c]

[a] *School of Mathematical Sciences, Monash University, Melbourne, Victoria 3800, Australia*
[b] *Computational Software Development, VPAC, Melbourne, Victoria 3053, Australia*
[c] *Earth Systems Science Computational Centre (ESSCC), The University of Queensland, St. Lucia, Queensland 4072, Australia*

ACADEMIC PRESS

Journal of Computational Physics 184 (2003) 476–497

JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A Lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials

L. Moresi *, F. Dufour, H.-B. Mühlhaus

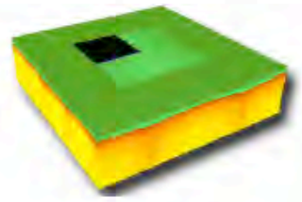
CSIRO Exploration and Mining, Australia Resources Research Centre, P.O. Box 1130, Bentley, Western Australia, Australia

Underworld

a long term geodynamics simulation platform

Underworld User Manual

for Version 1.5.x



Created by

VPAC

MONASH University Science

for

AuScope simulation analysis modelling

Editor:
Kathleen HUMBLE

March 3, 2011

Section 2

What does it do?

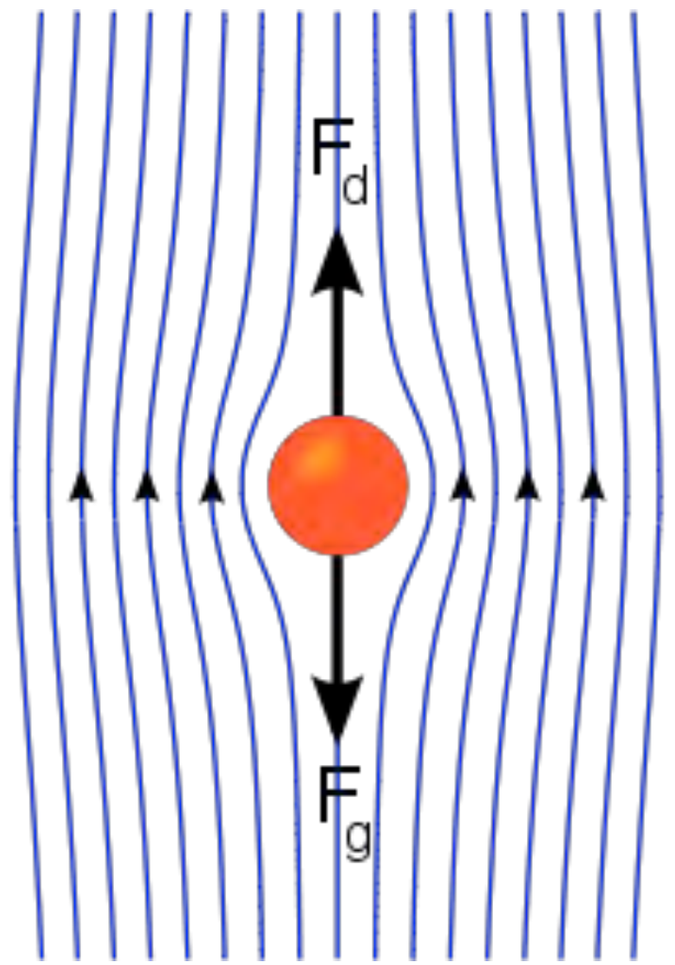
Underworld solves the equations governing Stokes flow (after George Gabriel Stokes 1819-1903) a class of flow characterized by a small Reynolds number (after Osborne Reynolds, 1842-1912) a dimensionless number which expresses the ratio between inertial forces to viscous forces. Stokes flow are characterized by smooth laminar fluid motion, in contrast to turbulent and chaotic flow occurring at high Reynolds number.

Inertia is the resistance to a change of motion it is expressed by Newton's second law of motion F=m.a. Earth geodynamics (mantle flow processes), and tectonics (plate tectonic processes) involve very small acceleration, hence very small inertia. In contrast, viscous forces are the resistive forces to flow, i.e. the viscosity of rocks. The viscosity of rocks is very large (>1e18 Pa.s).

Let's consider a tectonic plate 4000 km x 2500 km x 150 km accelerating from 0 to 5 $cm.yr^{-1}$ in 100,000 years. The density of the plate is 3000 $kg.m^{-3}$, and the viscosity of the asthenosphere is 1e20 Pa.s. What is the Reynolds number of the mantle flow induced by this acceleration, assuming that the plate motion causes shearing distributed over 10 km of asthenosphere?

nb: shear stress = ~ strain rate x viscosity

Example of smooth Stoke flow



Underworld solves the Stokes equation on a cartesian 2D or 3D grid assuming the flowing material is incompressible (which imposes a zero divergence condition on the velocity field, and therefore the mass continuity equation: $u_{i,i} = 0$ which ensures conservation of mass). The Stokes' equation in a tensorial form is:

$$\sigma_{ij} = \tau_{ij} - p\delta_{ij} = f_i$$

This equation expresses the conservation of momentum. It states that the total stress tensor σ_{ij} (the sum of the deviatoric stress tensor τ_{ij} and the pressure tensor $p\delta_{ij}$) balances the buoyancy force f_i due to density variations (also known as the gravitational body force) .

The constitutive equation expresses the relationship between the deviatoric stress tensor, the viscosity tensor (or constitutive tensor) and the strain rate tensor:

$$\tau_{ij} = 2\eta_{ijkl} \,.\, \dot{\epsilon}_{kl}$$

Hence, the resulting Stokes' equation:

$$2\,\eta_{ijkl} \,.\, \dot{\epsilon}_{kl} - p\delta_{ij} = f_i$$

or in a vectorized form the momentum equation is:

$$\eta\,\nabla^2 u - \nabla p = \rho g$$

In which $\nabla^2 u$ and ∇p are the velocity and pressure gradient respectively, η is the viscosity, ρg the driving force and $\eta\,\nabla^2 u$ is the stress gradient. The left term is the divergence of stress, the right term is the body force.

Underworld solves the Stokes' equation and retrieves pressure and velocity at the nodes of a 2D or 3D mesh. Material properties including density, viscosity, strain-rate, etc are updated and stored into particles which are advected through the grid.

Keeping track of the temperature is critical as it impacts on both densities and viscosities. Partial melting is a metamorphic reaction of particular importance because it consumes vast amount of latent heat, and because event a small fraction of melt can drastically reduces the viscosity. The following third equation, describes the rate of change of temperature as a function of heat diffusion (κ), radiogenic heat production (H), heat advection (u_z), and partial melting (*DF*: variation of melt fraction):

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T + \frac{\mathrm{H}}{C_p} - \left(u_z \bullet \nabla \right) T - T \frac{\Delta S}{C_p} \frac{DF}{Dt}$$

The equation above ensures conservation of energy and allow the coupling of densities and viscosities to temperature:

$$\rho\left(T\right) = g\rho_0\left(1 - \alpha(\,\Delta T\,) - \beta F\right)$$

$$\eta\left(T\right) = \frac{1}{2} A^{-1/n} \cdot Exp\left(\frac{E}{n \cdot R \cdot T}\right) \cdot \dot{\varepsilon}^{(1-n)/n}$$

Where A is the pre-stress factor, n is the stress exponent, E is the activation energy, and $\dot{\epsilon}$ is the strain rate.

With this in mind the momentum equation becomes:

$$\eta(T)\,\nabla^2 u - \nabla p = g\rho(T)$$

Plastic deformation is introduced by considering that above a yield stress (τ) plastic materials become viscous. The yield stress is:

$$\tau = \left(C_0 + \mu_{eff} \cdot \left(\rho g z\right)\right) \cdot f(\epsilon)$$

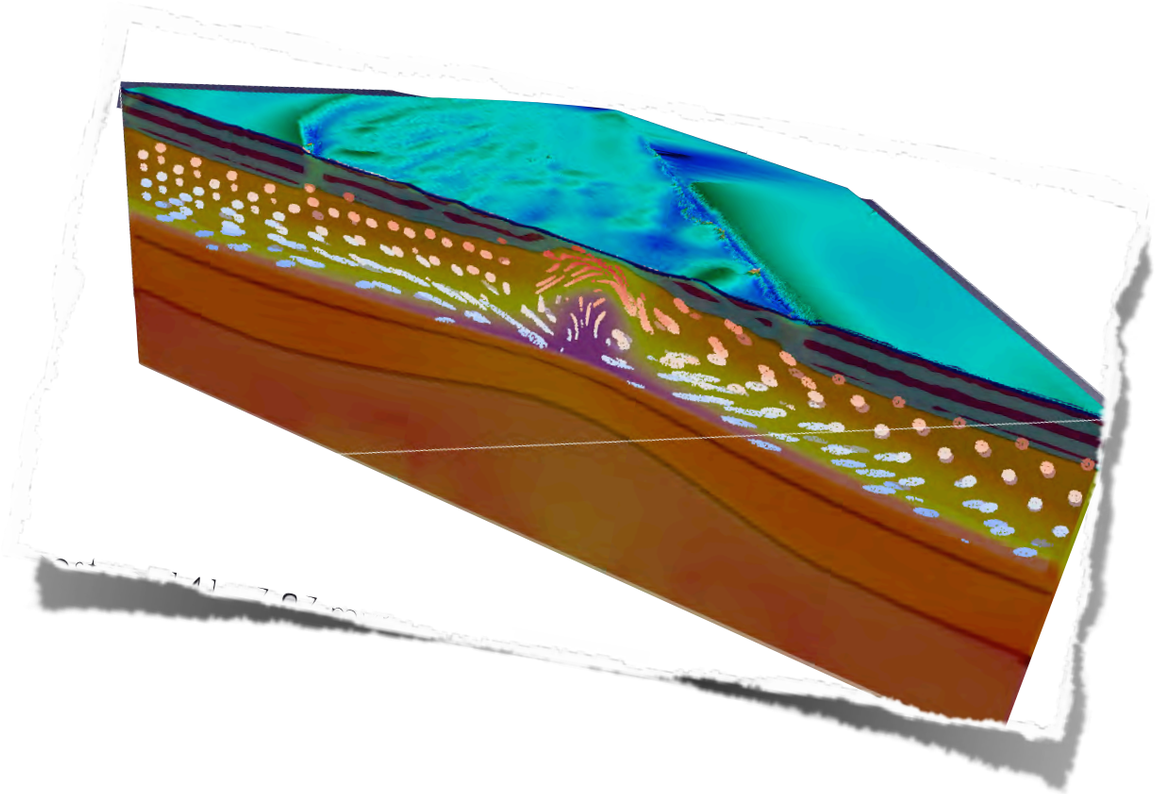
Where $f(\epsilon)$ is strain weakening function (fault zones become weaker as strain accumulate), C_o is the cohesion, μ_{eff} is the effective coefficient of friction (i.e. it takes into account the effect of pore pressure), $\rho g z$ is the confining pressure.

The post-yielding viscosity:

$$\eta_{yield} = \frac{\tau}{2 \cdot \left(\frac{1}{2}\sum_{ij} \dot{\epsilon}_{ij}\dot{\epsilon}_{ij}\right)^{1/2}}$$

Where the denominator is twice the second invariant of the strain tensor (also called the effective strain rate).

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right]$$



TENSOR- Tensors describe linear relationships between scalars, vectors and other tensors. The dimensionality of tensor, its rank, informs about the nature of the tensor, e.g.:
a tensor of rank 0 is a scalar;
a tensor of rank 1 is a vector (1- dimensional array);
a tensor of rank 2 matrix (2-dimensional array) ...
Tensors provide a concise way for formulating relationship between quantities, the drawback is they can be hard to read.

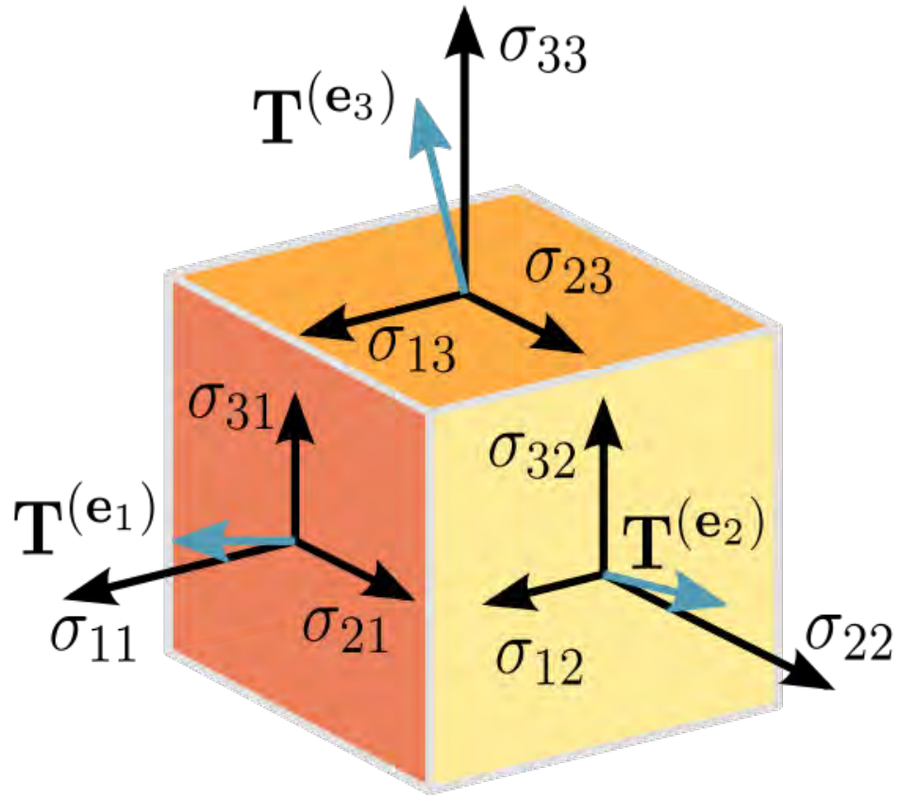
Stress tensor

The stress tensor is a tensor or second order which collects into columns the coordinates of 3 traction vectors (T_{ei}) acting on 3 perpendicular faces of a cube. The value of these coordinates depends on the choice of the coordinate system, however, these coordinates fully describe the state of stress on the central point in the cube. By convention the component σ_{ij} refers to coordinates along axis i of the traction acting on face j. The number of indices relates to the order of the tensor (*order*, *degree*, *rank* are synonymous).

Einstein's summation convention implies summation when an index variable appears twice in a single term:

$$c_i x_i \iff \sum_{i=1}^{3} c_i x_i \iff c_1 x_1 + c_2 x_2 + c_3 x_3$$

$$\text{hence: } \tau_{ij} = 2\eta_{ijkl} \cdot \dot{\epsilon_{kl}} = \sum_{k=1}^{3} \sum_{l=1}^{3} 2\eta_{ijkl} \cdot \dot{\epsilon_{kl}}$$



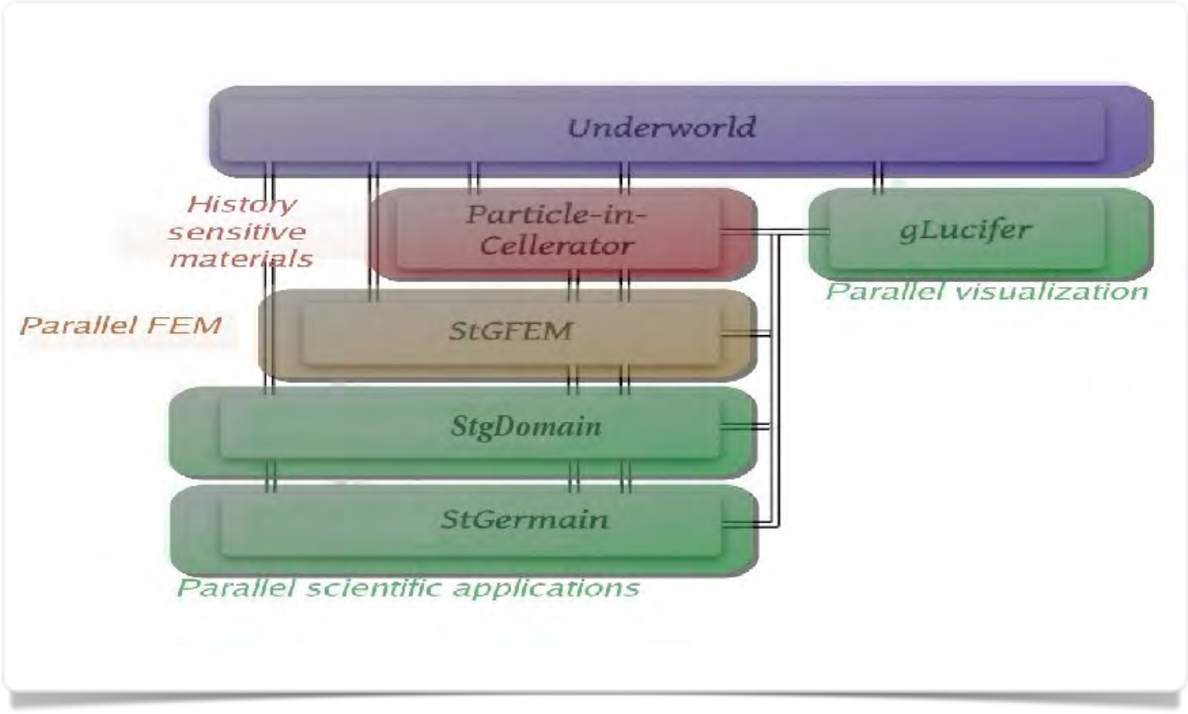
$$\sigma = \left[\mathbf{T}^{(\mathbf{e}_1)}\mathbf{T}^{(\mathbf{e}_2)}\mathbf{T}^{(\mathbf{e}_3)}\right] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

Section 3

How it does it?

Underworld is one component into a modular numerical framework made of a stack of specialized components (numerical libraries) working together to solve efficiently Stokes' equations in the context of a tectonics and/geodynamics model. This framework also provides gLucifer a visualization platform.

- *Underworld* - Geophysical formulations, e.g. rheologies.
- *gLucifer* - Visualisation package.
- *PICellerator* - Particle in Cell methods (Lagrangian / Eulerian framework).
- *StgFEM* - Finite Element Method (FEM) framework (Eulerian). Includes input files and plugins stored in StgFEM_Components or StgFEM_plugins directory.
- *StgDomain* - Basic mathematical formulations, model geometry, mesh, shape and basic swarm framework.
- *StGermain* - Core parallel programming architecture, memory management and Object Oriented frameworks.



Many of these components are built themselves upon other libraries (PETsc, OpenMPI or MPIC, ...) which explains that the deployment of the *Underworld* framework can be challenging.

Users access *Underworld* through "input files" the purpose of which is to describe the architecture of the model, the physical properties of materials that compose it, the time-dependent boundary conditions under which the model exists, and the optional and user specified passive markers necessary to capture the thermal and mechanical evolution of the model.

Underworld input files are xml files (xml: extensible markup language). xml allows the description, storage and transport of data in a simple text format with a trees structures (elements, sub-elements, sub-sub-elements ...).

xml have a hierarchical structure allowed to stitch several xml files together, using the component <include>:
<include> mycrustRheology.xml </include>

Tags (<tag></tag>) in xml are not pre-defined like in HTML. xml allows for customizable case sensitive tags with quoted attributes: <density=”2700”>

Because tags are defined by users, xml is self descriptive. and more information can be added using <!-- blabla -->:
<--! this is a comment -->

Example1:
Here is a shape:

```
<struct name="my_diapir">
<param name="Type">Sphere</param>
<param name="CentreX" units=”km” > 0.5 </param>
<param name="CentreY" units=”km” > 0.5 </param>
<param name="CentreZ" units=”km” > minZ </param>
<param name="radius" units=”km” > 0.1 </param>
</struct>
```

Here is another shape involving the Intersection (!) function:

```
<struct name="my_diapir_host">
<param name="Type">Intersection</param>
<list name="shapes">
<param> myrectangle </param>
<param> ! my_diapir </param>
</list>
</struct>
```

Here we define a simple viscous rheology:
Example2:

```
<struct name="my_diapirViscosity">
<param name="Type">FrankKamenetskii</param>
<param name="TemperatureField">TemperatureField</param>
<param name="eta0" units="Pa*s">1.0e26</param>
<param name="theta">13.815510558</param>
</struct>
```

Here we define a viscosity limiter function:

```
<struct name="viscosityLimiter">
<param name="Type">ViscosityLimiter</param>
<param name="maxViscosity" units="Pa">1e23</param>
<param name="minViscosity" units="Pa">1e18</param>
</struct>
```

Here we define a plastic rheology:
Example 3

```
<struct name="myVonMises">
<param name="Type">VonMises</param>
<param name="StrainRateField">StrainRateField</param>
<param name="cohesion">50.0</param>
<param name="StrainWeakening">strainWeakening</param>
</struct>
```

Here we define a weakening function:
Example 4

```
<struct name="strainWeakening">
<param name="Type">StrainWeakening</param>
<param name="TimeIntegrator">timeIntegrator_aux</param>
<param name="MaterialPointsSwarm">materialSwarm</param>
<param name="softeningStrain">0.1</param>
<param name="initialDamageFraction">0.0</param>
<param name="initialDamageWavenumber">0.5</param>
<param name="initialDamageFactor">0.5</param>
<param name="healingRate">0.0</param>
</struct>
```

We put everything together adding a few bits and pieces to define the thermal and mechanical properties of materialOne

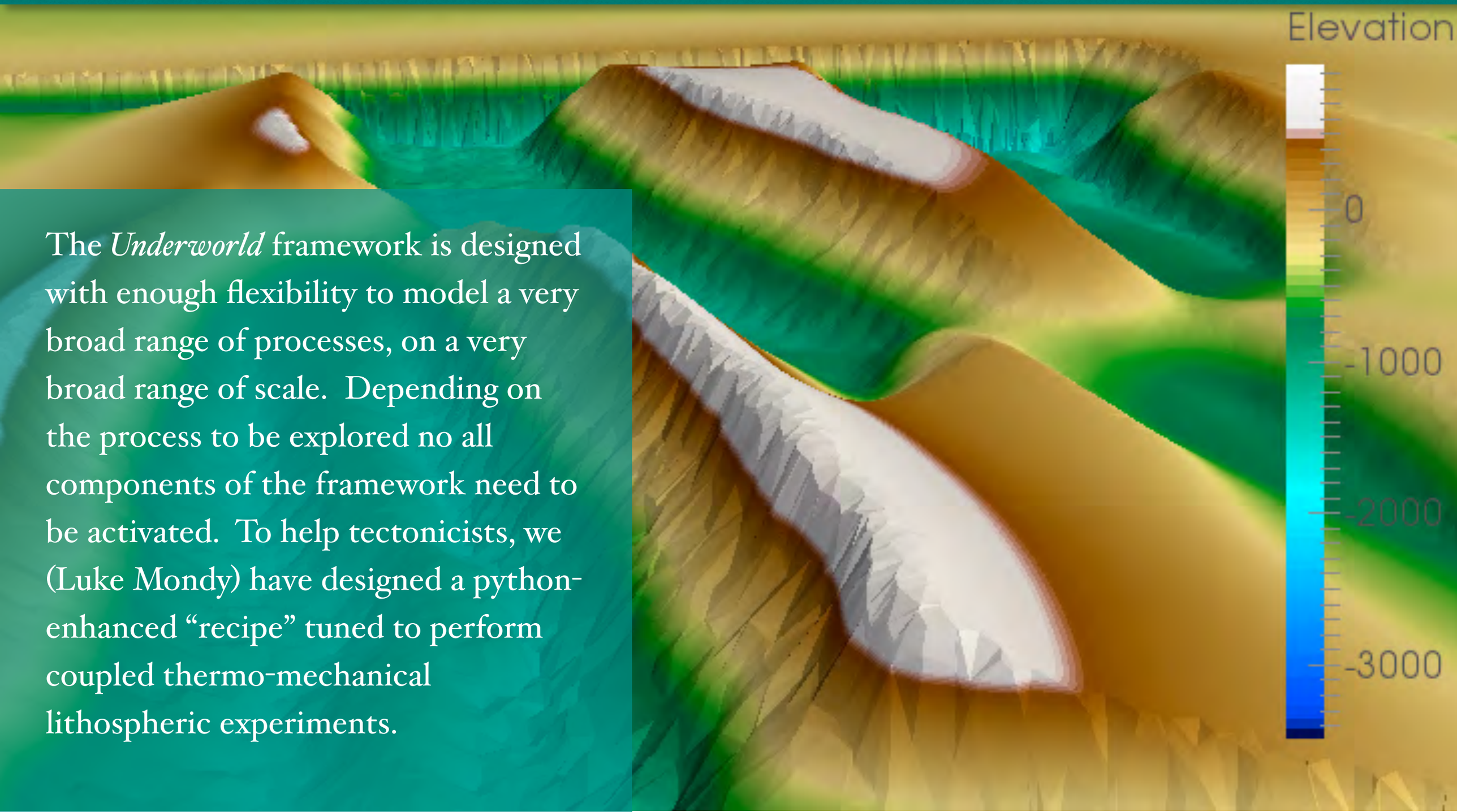
```
<struct name="materialOne">
        <param name="Shape"> my_diapir_host </param>
        <param name="density" units=”kg.m-3”>2800</param>
        <param name="alpha" units=”K-1”>3e-5</param>
        <param name="Diffusivity">1e-6</param>
        <list name="Rheologies">
                <param> myVonMises </param>
                <param> strainWeakening </param>
                <param> my_diapirViscosity </param>
                <param>viscosityLimiter</param>
        </list>
        <list name="heatingElements">
                <param name="Q" units=”W.kg-1”>1.8e-12</param>
                <param name="Cp" units=”J.kg-1.K-1”>1000</param>
        </list>
</struct>
```

A list of shape functions and rheology functions can be found in the *StgDomain* and *Underworld* component codex menu:
http://underworldproject.org/codex-v1.6.0/

Chapter 2

Lithospheric Model Recipe

Luke Mondy Recipe

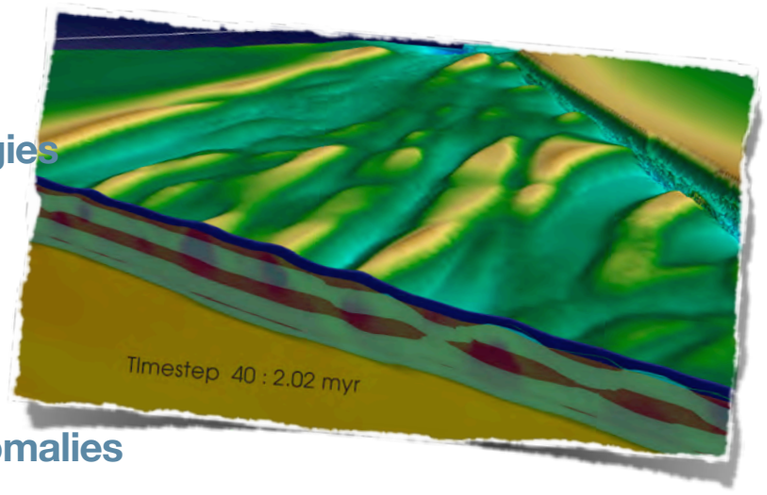


The *Underworld* framework is designed with enough flexibility to model a very broad range of processes, on a very broad range of scale. Depending on the process to be explored no all components of the framework need to be activated. To help tectonicists, we (Luke Mondy) have designed a python-enhanced "recipe" tuned to perform coupled thermo-mechanical lithospheric experiments.

Introduction

The Lithospheric Recipe Model (*LMR*) consists in a stack on xml files documenting a specific aspects of a model, providing access to optional routines (erosion, partial melting, ...), fine tuning the solver etc. This modular input file makes the process of designing a model easier. In addition, python script (lmrRunModel.py) is used to pass the input file into *Underworld* framework for processing.

lmrMain	**Stitch together the various following components**
lmrInitials	**Initial temperature field**
lmrMaterials	**Initial geometry**
lmrRheologyLibrary	**Rheological parameters for a range of rheologies**
lmrThermalEquilibration	**Thermal equilibration routine**
lmrOtherProcesses	**Erosion, sedimentation**
lmrThermalBoundaries	**Thermal boundaries, & temperature anomalies**
lmrVelocityBoundaries	**Velocity boundary conditions**
lmrPassiveTracers	**Passive tracers, PTt recorders**
lmrStart	**Resolution, temperature equilibration, model duration, checkpoint**
lmrNumerics	**Numerical, thermal and outputs parameters, melt**
lmrSolvers	**Solver**
lmrRunModel.py	**To run your model**



Model Architecture and Thermo-Mechanical Properties

ImrMaterials stores all the parameters controlling the geological architecture composing a model. It includes all the cartesian coordinates of the various rock layers, and the parameters describing the physical properties for each rock types involved in the model.

The architecture describe by the ImrMaterials on the right include a number of passive “markers” in the form of material layers (e.g. uppercrustMarkers, mantleMarkers). This is not the smarter way to introduce passive markers. We advice to use ImrPassiveTracers which 1/ are not involves involves in the calculations, and 2/ are leaner files because involving of less tracers than the normal particle swarm.

Examine the *ImrMaterials* on the right and draw an accurate 3D block diagram of the model it represents.

ImrMaterials

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

    <!-- === Domain size ========================================== -->

    <!-- Axis 0: x -->
    <param name="minX" units="km">  0 </param>
    <param name="maxX" units="km"> 256 </param>

    <!-- Axis 1: y -->
    <param name="minY" units="km"> -118 </param>
    <param name="maxY" units="km"> 10 </param>

    <!-- Axis 2: z -->
    <param name="minZ" units="km"> 0 </param>
    <param name="maxZ" units="km"> 256 </param>

    <!-- ========================================================== -->

    <!-- === User-defined Variables for ease of use ============ -->
    <!--     These variables make easier to quickly change layers thickness. -->

    <param name="topAir" units="km">10</param>
    <param name="botAir" units="km">0</param>

    <param name="topUpperCrust" units="km">0</param>
    <param name="botUpperCrust" units="km">-16</param>

    <param name="topLowerCrust" units="km">-16</param>
```

ImrRheologyLibrary- To keep *ImrMaterials* concise, densities, thermal properties and rheological properties of various types of rocks are stored into *ImrRheologyLibrary*. Users can pick which rheology to use for the various rocks involved in their models, or make their own; and control density and thermal properties (including partial melting).

Using some of the thermal and mechanical properties listed in the *ImrMaterials* file on the right, assemble and plot (use python, R, MatLab ...) the temperature profile, the density profile and the rheological profile of your favorite model of continental lithosphere.

ImrRheologyLibrary

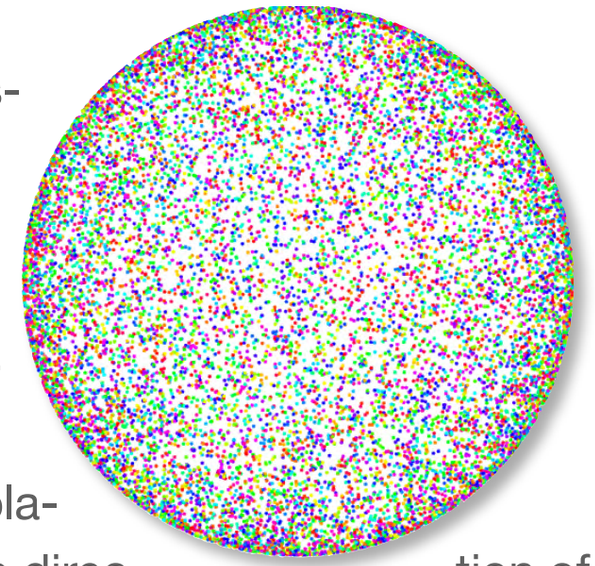
```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

<struct name="components" mergeType="merge">

        <!-- ================================================================== -->
        <!-- ========================= Rheology library ======================= -->
        <!-- ================================================================== -->

        <!--
                The rheology library is a collection of pre-defined rheological parameters, including: viscous and brittle flow laws; densities; thermal properties (radiogenic heating, diffusivity, etc.).
                The best way to use this document is to:
                  1) Browse and pick the rheological parameters you want to use;
                  2) copy the name of the property;
                  3) open your ImrMaterials.xml file, locate the appropriate material (e.g. "lower_crust");
                  4) replace or create the name of the rheological property in the material definition.
                Looking through the default ImrMaterials.xml should make this obvious.

                The structure of the library is as such:
                    - Densities,
                    - Thermal properties,
                    - Partial melt functions,
                    - Flow laws
                        - Viscous laws:
                            - Isoviscous
                            - Temperature dependent
                            - Strain-rate/Temperature dependent
                            - Partial melt viscosity modifiers
                        - Brittle laws
                            - Drucker-Prager
                            - Stress Limiters

                The document is also grouped into Earth-like layers: upper crust, lower crust, mantle, etc.
```

ImrPassiveTracers stores the information necessary to distribute passive tracers inside a model, in a structured or random manner. The purpose of these passive tracers is to record time-dependent physical properties attached to a small volume of rock (for instance to construct PTtime path), record finite strain (tracers can be assembled into a bunch of spheres, or into horizontal or vertical planes or layers), or keep track of density interfaces in a way which doesn't require saving the entire particle swarm at each time step (a very space and time consuming exercise).

Given the initial position of these tracers *Underworld* generates a user_passiveSwarm, and advects it passively with the rest of the model. The user can save at each time step a collection of time-dependent properties experienced by each individual tracers.

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

   <struct name="components" mergeType="merge">

      <!-- ========================================================================== -->
      <!-- ======================== Passive Tracer Distribution ======================= -->
      <!-- ========================================================================== -->


     <!-- ============================== Passive Horizons ============================ -->
     <struct name="surface_marker_PTL">
        <param name="Type"> PlaneParticleLayout </param>
        <param name="totalInitialParticles">262144</param>
        <param name="planeAxis">y</param>
        <param name="planeCoord" units="km">0</param>
     </struct>

     <struct name="BDT_marker_PTL">
        <param name="Type"> PlaneParticleLayout </param>
        <param name="totalInitialParticles">100000</param>
        <param name="planeAxis">y</param>
        <param name="planeCoord" units="km">-16</param>
     </struct>
```

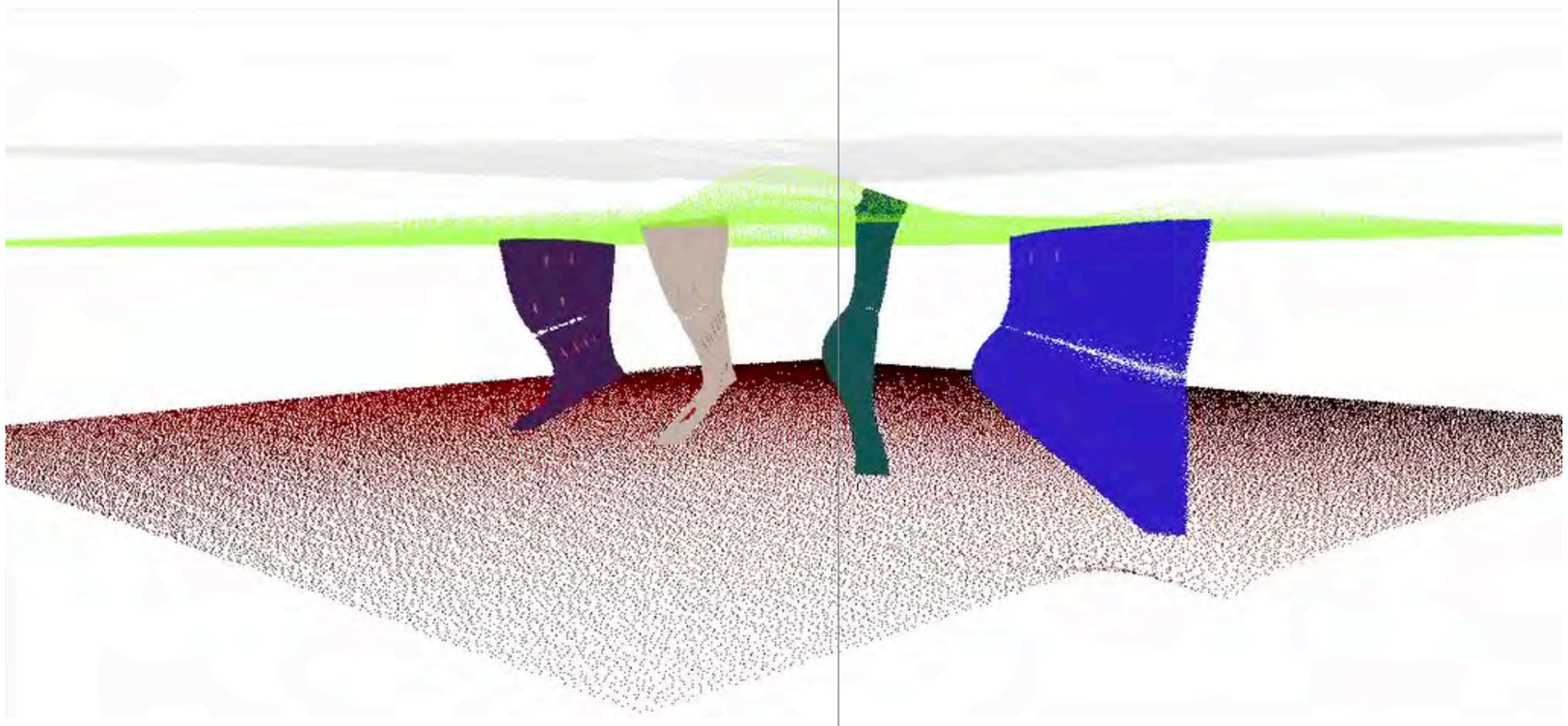
For capturing the evolution of regional strain, we have the option to embed in the model an array of empty spheres the surface of which is covered by a uniform distribution of tracers. As the model deforms, these spheres stretch and flatten recording the strain history. When deformation is homogeneous the spheres progressively transform into ellipsoids, from which homogeneous finite strain can analyzed. The flattening of the ellipsoids (i.e. the XY plane of the strain finite) represents the the foliation planar fabric, the direction of maximum elongation (i.e. the X axis of the finite strain ellipsoid) represents the stretching lineation within the foliation plane. Anarray of spheres with a good density can capture time-dependent strain field in a manner which is easily readable by structural geologists.



```
library(rgl)
##########################################
# Uniform distribution of nb_points in a square
##########################################
nb_points = 200
x1 = runif(nb_points)
x2 = runif(nb_points)
# the option pch = '.' change the symbol for the graph into dot.
# cex = 2 doubles the size of the dots
plot(x1,x2, col = 'blue', pch = '.', cex = 2)

##########################################
# Uniform distribution on a sphere
##########################################
# This function transform the spherical coordinates into cartesian coordinates
```

To produce this array of spheres we have written a *Python* and a *R* scripts. Both output HDF5 files that can be passed into *Underworld* via the *ImrPassiveTracer* file, via the *FileParticleLayout* Type.

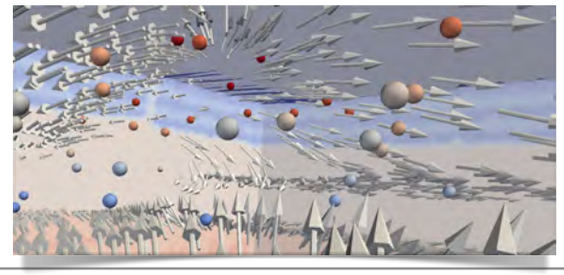
Movie 2.1 Underworld experiment of crustal extension



Four vertical walls filled with passive tracers are dragged into a dome structure. Grey plane shows the surface topography; bright green is the brittle-ductile interface; purple plane is the Moho.

Section 3

Internal and Boundary Conditions



A lithosphere model is not complete without i/ an initial temperature field which give the temperature at the nodes of the computational grid; ii/ the temperature boundary conditions imposed on its edges; iii/ the kinetic (i.e. velocity) or dynamic (i.e. stress) imposed on its edges and/or inside the model; and iv/ the processes governing the evolution of the topography at its the surface (i.e. erosion and sedimentation).

Initial Temperature Field

The initial temperature field (i.e. temperature field at time = t_0) is designed by running a static experiment (experiment whose edges are maintained fixed, and whose inside is at rest). This experiment can be performed on your desktop or laptop (no need for HPC) as only the heat equation is involved (i.e. Stokes' equation are not solved). The initial geotherm is defined in *ImrThermalEquilibration* which also optimizes the numerics to speed up the compute time. *ImrThermalEquilibration* uses as an input an exponential function that approximates the steady state geotherm. The model is then run under static conditions to let the model thermally equilibrate.

ImrThermalEquilibration

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

   <!-- ================== Initial Geotherm Definition ================== -->
   <struct name="components" mergeType="merge">
      <struct name="tempProfile">
         <param name="Type">Ppc_Thermal_Profile</param>
         <param name="StartCoord" units="km">   0.0</param>
         <param name="EndCoord"   units="km"> -118.0</param>
         <param name="MinTemp" units="K"> 293.15 </param>
         <param name="MaxTemp" units="K"> 1573.15 </param>
         <param name="LinearCoefficient">0.00465139</param>
         <param name="ExponentialCoefficient1">774.73 </param>
         <param name="ExponentialCoefficient2">1.23996e-5</param>
         <param name="axis"> @VerticalAxis </param>
      </struct>
   </struct>

   <!-- The above function defines a geotherm via:
         T = min_T + LinearCoefficient * (start_coord - depth) +
               ExponentialCoefficient1 * (1 - exp(-ExponentialCoefficient2 * (start_coord - depth)))
```

To run a static experiment and create the initial temperature field:

1/ In lmrMain:

```
# Comment out the following time consuming routine:
<!-- <include>lmrPassiveTracers.xml</include>
```

2/ In lmrStart:

```
<run_thermal_equilibration_phase> true </run_thermal_equilibration_phase>
<Underworld_binary> /your_path_to_your_Underworld_executable/Underworld </Underworld_binary>
<CPUs> 6 </CPUs> # The choice of number of CPU
<supercomputer_mpi_format> false </supercomputer_mpi_format>  # true if you are using an HPC
```

3/ Then in a terminal window: `python ./lmrRunModel.py`
This will create a folder called: “initial-condition_4x48x4_laterally_homog” containing the initial temperature at each node of a grid.

The initial temperature field is then passed onto the model by prescribing in *lmrInitial* the path to the relevant temperature and grid files:

4/ in lmrInitial:

```
<param name="FeVariableHDF5Filename"> /path_to/initial-condition_4x48x4_laterally_homog/TemperatureField.07884.h5 </param>
<param name="MeshHDF5Filename"> /path_to/initial-condition_4x48x4_laterally_homog/Mesh.LinearMesh.00000.h5 </param>
```

5/ in lmrStart turn off the thermal equilibration routine, and on the HPC routine:

```
<run_thermal_equilibration_phase> false </run_thermal_equilibration_phase>
<Underworld_binary> /path_to_your_Underworld_executable/Underworld </Underworld_binary>
<supercomputer_mpi_format> true </supercomputer_mpi_format>
```

6/ in lmrMain revert to:

```
 <include> lmrPassiveTracers.xml </include>
```

Temperature boundary conditions

The surface of the lithosphere is maintained at a constant temperature usually around 20ºC. Its base is either maintained at a constant temperature usually around 1330ºC - in which case the temperature gradient at the base of the lithospheric mantle is allowed to change -, OR a constant heat flow which imposes the temperature gradient - in which case the temperature at the base of the lithosphere is allowed to change. The heat flow entering the lithosphere is usually in the range of 10 to 20 mW/m^2 (Jaupart and Mareschal, Treatise on Geophysics, 2007).

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

   <struct name="temperatureBCs">
      <param name="type"> CompositeVC </param>
      <list name="vcList">
         <struct>
            <param name="type"> WallVC </param>
            <param name="wall"> top </param>
            <list name="variables">
               <struct>
                  <param name="name"> temperature </param>
                  <param name="type"> double </param>
                  <param name="value" units="K"> 293.15 </param>
               </struct>
            </list>
         </struct>

         <struct>
```

Velocity boundary conditions

Kinematic boundary conditions are applied along the vertical walls of the model to drive extensional, contractional and transcurrent tectonics. These conditions are implemented via *ImrVelocityBoundaries*. This xml file controls also the isostasy at the base of the model.

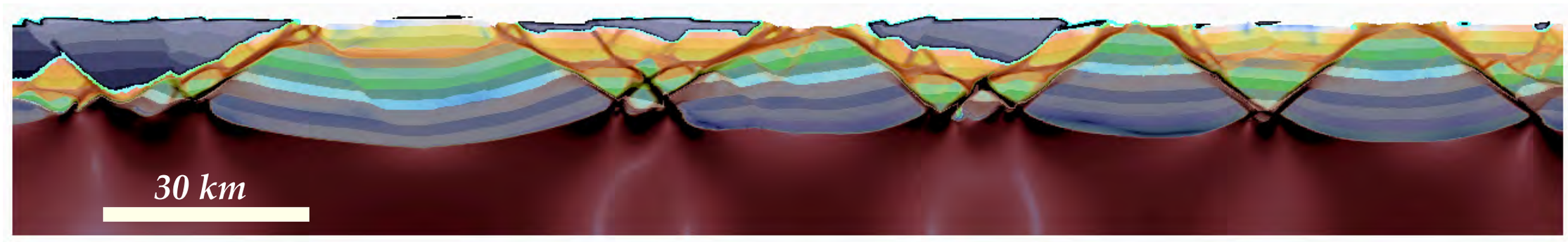
Underworld can either deform the model with the computational grid, or through it in which case the computational grid remains fixed that material flow out of it.

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

   <struct name="velocityBCs">
      <param name="type"> CompositeVC </param>
      <list name="vcList">
         <struct>
            <param name="type"> WallVC </param>
            <param name="wall"> left </param>
            <list name="variables">
               <struct>
                  <param name="name"> vx </param>
                  <param name="type"> func </param>
                  <param name="value"> left-wall-VC </param>
                  <!--param name="type"> double </param>
                  <param name="value" units="cm/yr"> -1.0 </param-->
               </struct>
```

Surface processes

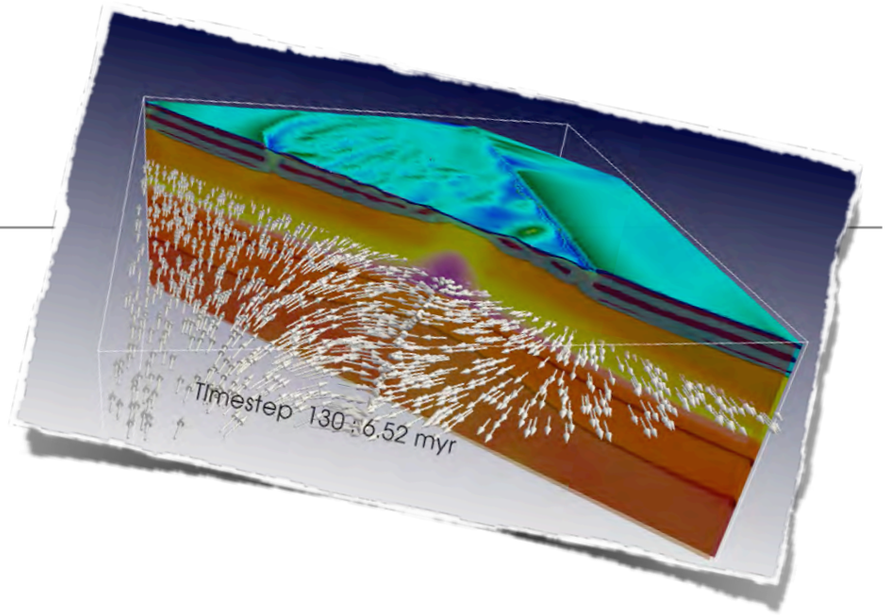
The surface of the lithosphere is exposed to weathering, erosion and sediment deposition. In *Underworld,* users can define an elevation above which materials disappear (e.i. transformed into air material) through erosion, and a base level below which sediments are deposited (air is transformed into sediments). Base level, and erosional level are implemented in the *ImrOtherProcesses* file. A bit crude, better than no surface processes, but not even near what *Badlands* can do.

```
<?xml version="1.0" encoding="UTF-8"?>
<StGermainData xmlns="http://www.vpac.org/StGermain/XML_IO_Handler/Jun2003">

   <struct name="components" mergeType="merge">
      <struct name="default_ppcManager" mergeType="merge">
         <list name="Properties" mergeType="merge">
            <!-- nb: If your extra property has a dependence on something Underworld
                     calculates (e.g., pressure, strain-rate), and you have non-linear
                     rheologies (e.g., you use yielding, or strain-dependent viscosities),
                     then your property must have StoreNLOnParticle = True.
                    To see more information, see ImrNumerics.xml -->
            <struct>
               <param name="PropertyName"> SurfaceProcesses </param>
               <param name="PassiveProperty"> True </param>
               <param name="StorePreviousValue"> True </param>
               <param name="StoreOnParticles"> True </param>
               <param name="StoreOnMesh"> True </param>
            </struct>
         </list>
      </struct>

      <!-- ========================= Surface Processes ======================== -->
      <struct name="erosion_height">
         <param name="Type"> Bas_Constant </param>
```



Section 4

Numerics and Solvers



Time step and solver

The default solver in *Underworld* uses the largest thermal diffusivity involved in the model and the size of the grid cells to calculate a time step which insures a stable and robust calculation. This time step is:

$$dt = C_f \cdot \frac{dx^2}{\kappa_{max}}$$

where C_f is the courant factor. This definition of the time step is dependent on dx^2, which means that time steps becomes unreasonably small for models run at high-resolution and/or for models or small dimensions (~a few 10's of km). This formulation also explains why air diffusivity is a problem in our experiments. The nominal diffusivity of air ~20 times larger than that of rocks. Therefore, in experiments involving a layer of air, the maximum diffusivity is that of the air, the temperature evolution of which is of no concern to us. Yet, the presence of air reduces for very little benefit by one order of magnitude. Hence, it may be a good idea to use a rock-like thermal diffusivity for the air.

When the dimension of the model and/or its resolution become an issue, the user can call upon the semi-lagrangian solver which doesn't depend on the size of the cells. The semi-lagrangian solver uses an interpolation method that is stable and accurate, and requires less amount of computational time (Kim et al., Eurographics, 2008, 27-2).

Fluid simulation at 200x150x100 grid resolution (Kim et al., 2008)

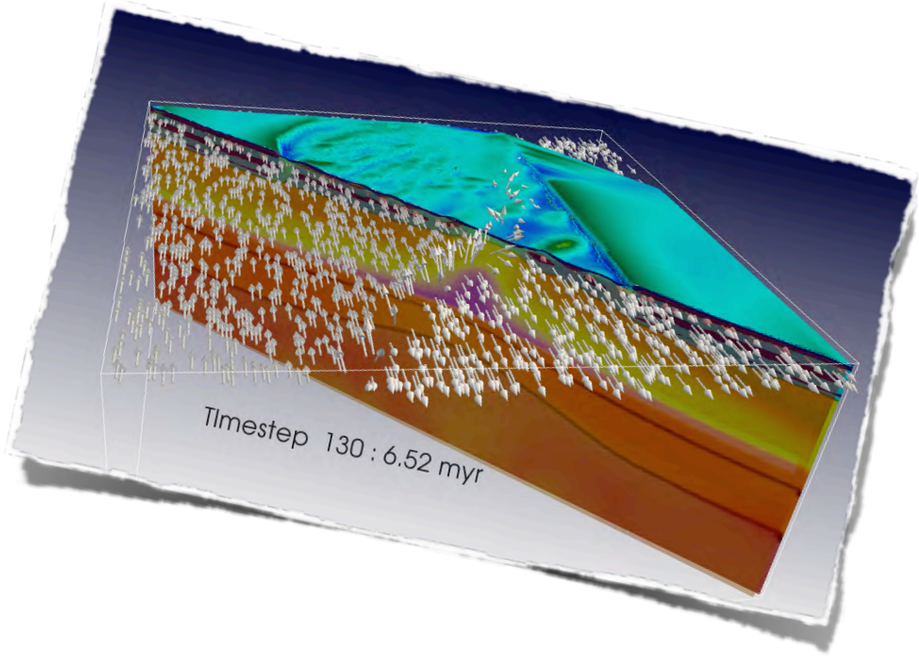
Switching to from the default to the semi-lagrangian solver:

In lmrMain.xlm replace

<include>StgFEM/TempAdvDiff.xml</include> by
<include>Underworld/BaseApps/SemiLagrangianADESolverHijack.xml</include>

In lmrNumerics.xml: Replace the “thermalEqn” structure by:

```
    <struct name="thermalEqn">
        <param name="Type">VectorAssemblyTerm_NA__F</param>
        <param name="ForceVector">fVector</param>
        <param name="functionLabel">sourceTerms_thermalEqn</param>
        <param name="Swarm">picIntegrationPoints</param>
        <param name="Manager">default_ppcManager</param>
    </struct>
<struct name="laplacianMatPpc" mergeType="replace">
        <param name="Type">Ppc_Operation</param>
        <param name="Operation"> * </param>
        <list name="Properties">
            <param> 0.5 </param>
            <param> DiffusivityProperty </param>
            <param> dtPpc </param>
        </list>
</struct>
<struct name="laplacianVecCoeffPpc" mergeType="replace">
        <param name="Type">Ppc_Operation</param>
        <param name="Operation"> * </param>
        <list name="Properties">
            <param> -0.5 </param>
            <param> DiffusivityProperty </param>
            <param> dtPpc </param>
        </list>
</struct>
<struct name="laplacianMatrixTerm" mergeType="merge">
```



```
        <param name="Swarm" mergeType="replace">picIntegrationPoints</param>
    </struct>
    <struct name="massMatrixTerm" mergeType="merge">
        <param name="Swarm" mergeType="replace">picIntegrationPoints</param>
    </struct>
    <struct name="laplacianVectorTerm" mergeType="merge">
        <param name="Swarm" mergeType="replace">picIntegrationPoints</param>
    </struct>
    <struct name="massVectorTerm" mergeType="merge">
        <param name="Swarm" mergeType="replace">picIntegrationPoints</param>
    </struct>
            <!-- ====== End of Thermal Energy ======================== -->
```

and in lmrThermalBoundaries.xml replace

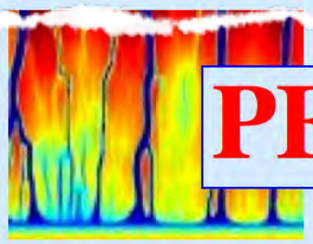
```
<param name="ForceVector"> residual </param>   by
<param name="ForceVector"> fVector </param>
```

Chapter 3

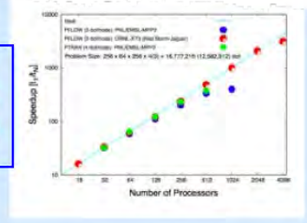
Installating *Underworld* & *LMR*

Underworld

a long term geodynamics simulation platform



PETSc



Portable, Extensible Toolkit for Scientific Computation

MPICH2 is a high-performance and widely portable implementation of the Message Passing Interface (MPI) standard (both MPI-1 and MPI-2).

The HDF Group

Installing *Homebrew* package manager ...

Installating *Underworld* framework on your local computer or a remote HPC framework is made easier by using a package installation manager, such as *MacPort*, *Fink* or *Homebrew* whose job it is to facilitate the deployment of libraries and plugins. We will use *Homebrew* as our favorite package manager.

nb: To get on with the job on deploying *Underworld*, one has to have basic knowledge of Unix and the use of Terminal windows. Installing *Underworld* requires no more than a dozen Unix commands.

Homebrew **is a package manager that simplifies the installation of *libraries* (also called *packages*, or *formula* in the *Homebrew* lingo) on your computer. These libraries are optimised to perform specialised tasks.**

To install *Homebrew*:
On Mac http://brew.sh/
On Linux use: http://brew.sh/linuxbrew/
... and follow the instructions.

On Mac, you need to load the Xcode suite available at: https://developer.apple.com/xcode/downloads/

Once *Homebrew* is installed, one can easily install packages by running the following command: *brew install package_name*

For a list of all Homebrew packages: https://github.com/Homebrew/homebrew/tree/master/Library/Formula

Installing *Underworld* dependencies

Dependencies refers to the set of libraries necessary to the normal operation of a package. *Underworld* depends on a range of dependencies including mpich2, petsc3.1, and hdf5. The order of installation is important.

1/ MPICH2

MPICH is a high-performance portable implementation of the *Message Passing Interface* (MPI) standard, a key ingredient to parallel applications managing the exchange of information across cores and shared memory. http://www.mpich.org/

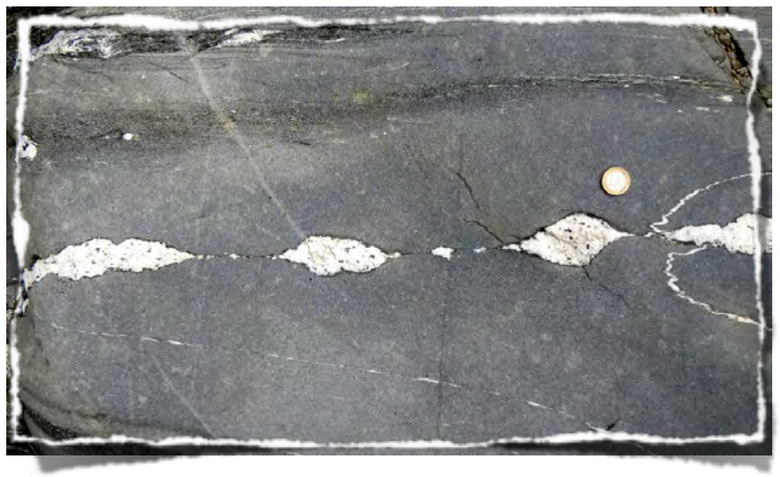
Open a terminal window and run the following commands:
Download the last version of mpich
cd mpich3.1.4
./configure --enable-shared --enable-sharedlibs=osx-gcc --enable-fast=all --with-mpe --prefix=/usr/local/mpich3 --with-pm=hydra CC=gcc CXX=g++ FC=gfortran F77=gfortran
make
make install

export MPI_DIR=/usr/local/mpich3
export PATH=$MPI_DIR/bin:$PATH
export LD_LIBRARY_PATH=$MPI_DIR/lib:$LD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=$MPI_DIR/lib:$LD_LIBRARY_PATH

2/ PETSC3.1

PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for parallel solution of scientific applications solving partial differential equations.

http://www.mcs.anl.gov/petsc/index.html



This is a comment.
brew install petsc-3.1-p8
export PETSC_DIR=`pwd`
uname -a # This return the operating system name and version, here darwin 10.8.0
export PETSC_ARCH=darwin10.8.0-c-opt
./config/configure.py --with-debugging=0 --prefix=/usr/local/Cellar/petsc3.1-p8 --with-shared=1 --with-mpi-dir=/usr/local-/Cellar/mpich2 --with-hypre=1 --download-hypre=ifneeded --with-ml=1 --download-ml=ifneeded --with-superlu-dist=1 --with-mumps=1 --download-mumps=ifneeded --with-parmetis=1 --download-parmetis=ifneeded --with-scalapack=1 --download-scalapack=ifneeded --with-blacs --download-blacs=ifneeded

make PETSC_DIR=/Users/Patrice/Documents/petsc-3.1-p8 PETSC_ARCH=darwin10.8.0-c-opt all # put your own path
make PETSC_DIR=/Users/Patrice/Documents/petsc-3.1-p8 PETSC_ARCH=darwin10.8.0-c-opt install # put your own path
make PETSC_DIR=/usr/local/Cellar/petsc3.1-p8 test
PETSC_DIR=/usr/local/Cellar/petsc3.1-p8; export PETSC_DIR
unset PETSC_ARCH

3/ HDF5

HDF5 is a data model and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data.

http://www.hdfgroup.org/HDF5/

brew install homebrew/science/hdf5

4/ Other useful libraries

brew install jpeg
brew install libtiff
brew install libpng
brew install ffmepg # Turn images into a customized movie.

5/ Mercurial

Mercurial is a free, source control management tool. It allows the copy of sources and synchronization to sinks. We will use Mercurial to maintain our *Underworld* source up-to-date.

http://mercurial.selenic.com/

brew install mercurial

Installing *Underworld*

Now that the dependencies are installed ...

Create a folder named bleedingEdge and clone into it the *Underworld* source from underworldproject website:
hg clone https:/www.underworldproject.org/hg/stgUnderworld bleedingEdge
cd bleedingEdge
hg up default
./obtainRepositories.py

To keep your *Underworld* source up-to-date:
cd bleedingEdge
./pull+update-all.sh
To compile your *Underworld* source:
cd bleedingEdge
./configure.py --with-debugging=0 --petsc-dir=/usr/local/Cellar/petsc3.1-p8 petsc_arch=darwin10.8.0-c-opt
--mpi-dir=/usr/local/Cellar/mpich2/3.0.4
./scons.py -j 8

cd earthbyte_additions/
UW_DIR=/path_to-your/bleedingEdge/build/ ./configure.py --with-debugging=0 --petsc-dir=/usr/local/Cellar/petsc3.1-p8
petsc_arch=darwin10.8.0-c-opt --mpi-dir=/usr/local/Cellar/mpich2/3.0.4
UW_DIR=/path_to-your/bleedingEdge/build/ ./scons.py -j 8

Installing *LMR*

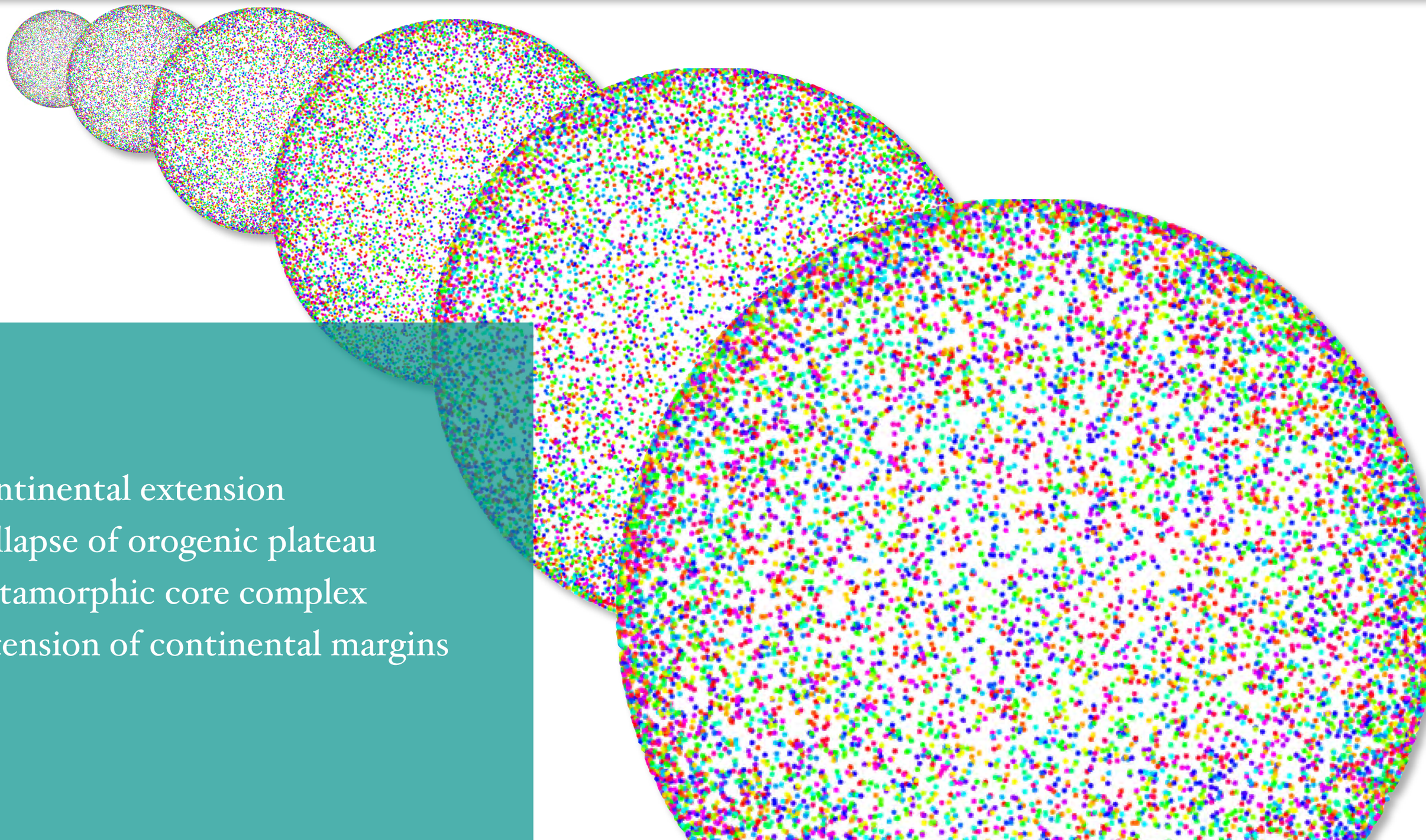
Clone Luke's *Lithospheric Model Recipe* from *Bitbucket* into your *BleedingEdge* directory:
hg clone https://patrice_rey@bitbucket.org/lmondy/lithosphericmodellingrecipe
change of *lithosphericmodellingrecipe* into something shorter like *LMR*
In the folder *LMR* open the file *lmrStart*
At the bottom of the file, replace */home/luke/Programs/uw-be/build/bin/Underworld* by *your own path to Underworld* and save.
Test your installation by executing: python ./lmrRunModel.py

Luke's useful python scripts

- **swarm_deleter.py**: cleans up materialSwarm data based on time. Eg:
python swarm_deleter.py test_data --keep_interval 3.15569e13
Would keep a checkpoint about every 1 Myr. All others get deleted. It runs in test mode by default. To actually delete stuff, run:
python swarm_deleter.py test_data --keep_interval 3.15569e13 --for_real
- **swarm_splitter.py**: splits the passive tracer swarms so Paraview can read them. Eg:
python swarm_splitter.py test_data
Would create some new XDMF.temporalSwarm... files, one for each swarm.
- **xdmf_generator.py**: cleans up the XDMF.temporalFields.xmf (sometimes needed if the model has been restarted a lot). Eg:
python xdmf_generator.py test_data
Creates a new XDMF.temporalFields.xmf file, where each timestep is only listed once, and in order. This can be an issue if the model has re-started many times.
- **xml_diff.py**: Tells the difference between two flattened UW input files. Eg:
python xml_diff.py test_flattened_xmls/input1.xml test_flattened_xmls/input2.xml
Prints out a rough guide to where the two files differ. Only works on the flattened output (usually in results/input.xml).

Chapter 4

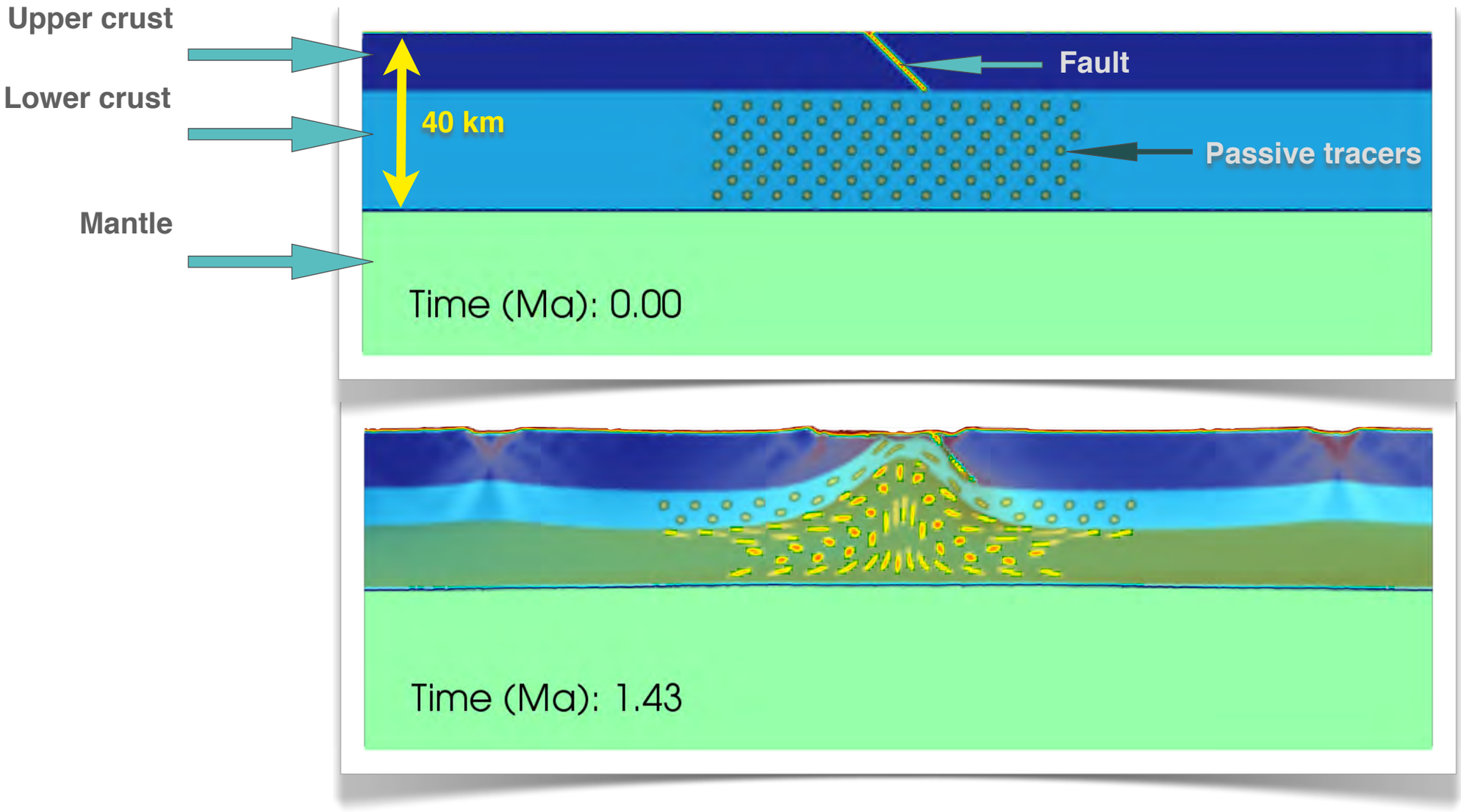
2D Underworld Examples



Continental extension
Collapse of orogenic plateau
Metamorphic core complex
Extension of continental margins

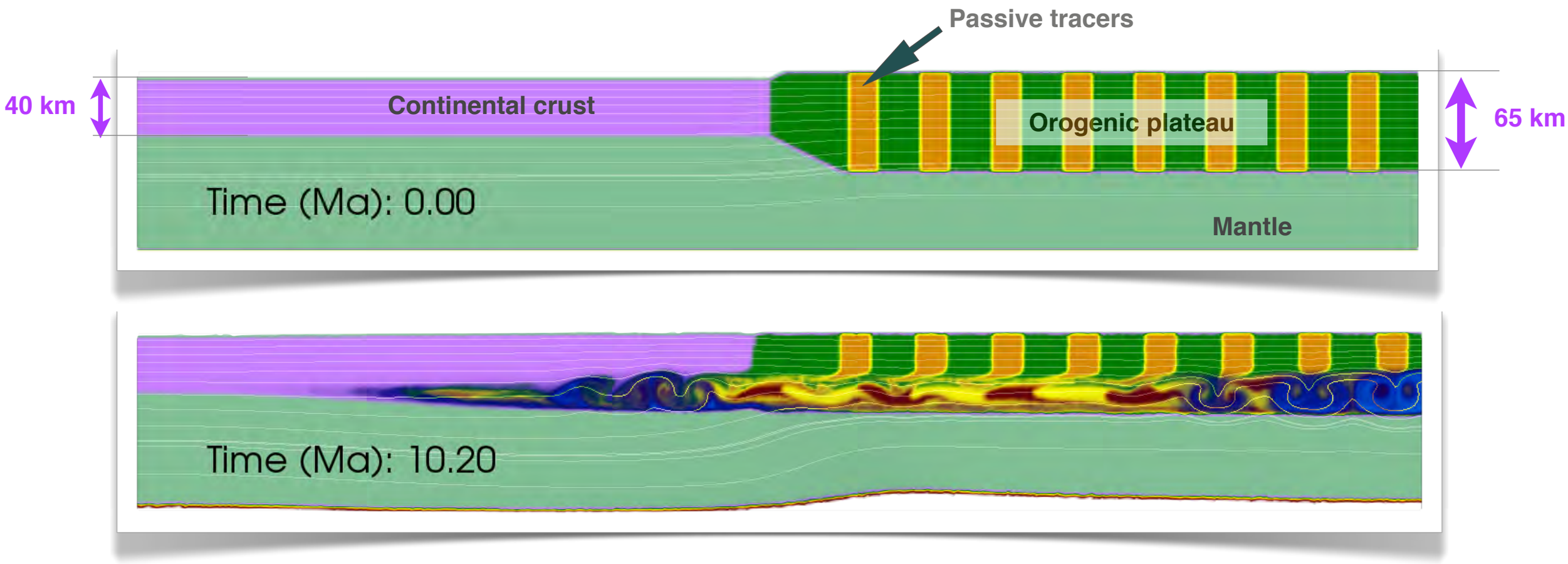
Continental extension

Extension of a “normal” continental lithosphere leads to the development of continental margins. Using appropriate thermo-mechanical properties (radiogenic heat production, diffusivity, rheology, density etc), and boundary conditions, design on a 120 km x 360 km domain an *Underworld* model (resolution 2 km) to explore the formation of continental margins.



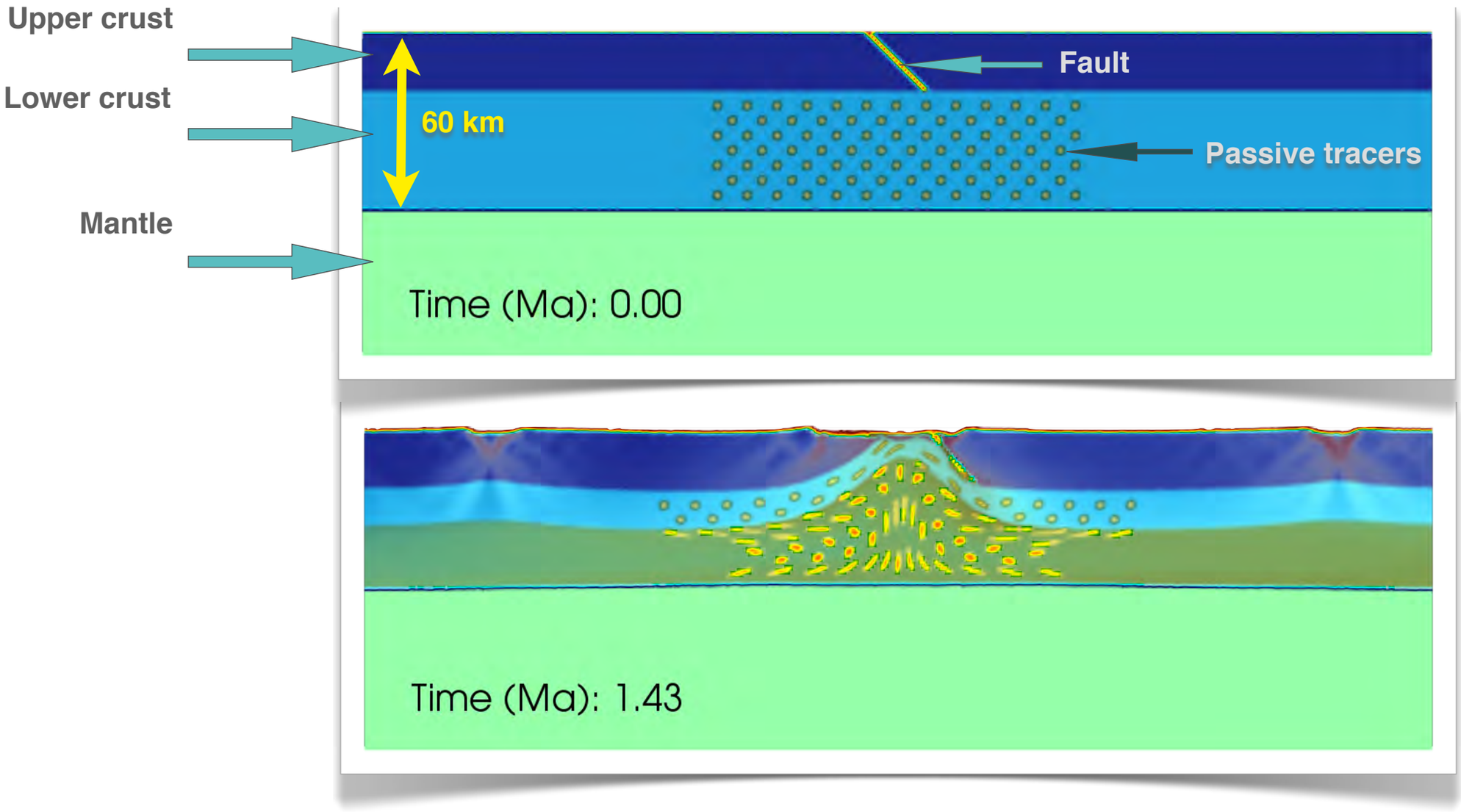
Collapse of an orogenic plateau

During their formation, orogenic plateaux store significant gravitational potential energy. This excess of energy drives their gravitational collapse, and - in the case of fixed boundary collapse - contractional shortening of foreland regions. Using appropriate thermo-mechanical properties (radiogenic heat production, diffusivity, rheology, density etc), and boundary conditions, design on a 150 km x 900 km domain an *Underworld* model (resolution 2 km) to explore the process of plateau collapse.

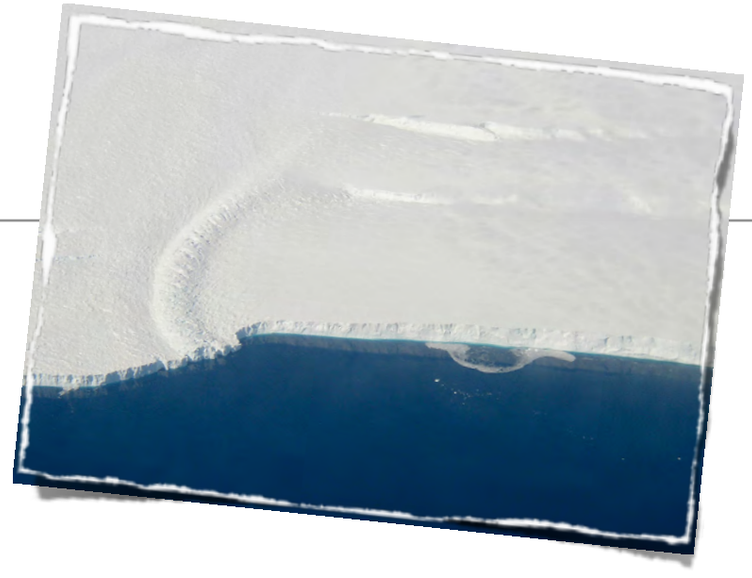


Metamorphic core complex

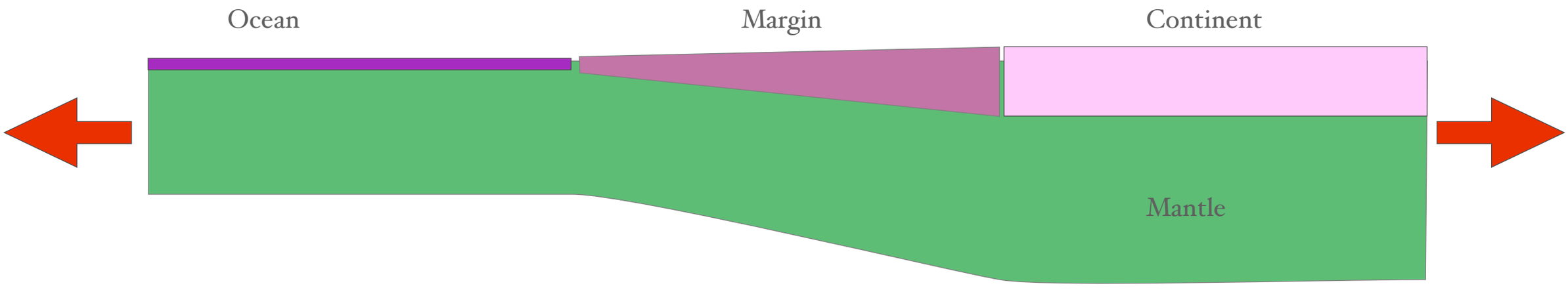
Extension of thick and hot crust leads to the development of metamorphic core complex (MCC). Using appropriate thermo-mechanical properties (radiogenic heat production, diffusivity, rheology, density etc), and boundary conditions, design on a 120 km x 360 km domain an *Underworld* model (resolution 2 km) to explore the formation of MCC.



Extension of a continental margin



Continental extension occurs is a limited number of modes. Supercontinents break-apart into a few continents via extension focussing on pre-existing zones of weakest (often ancient suture zones). Because continents have insulating properties, the temperature anomaly that buildup in the mantle underneath continents is an active driver to continental breakup. Fragmentation of continents through the detachment of blocks at continental margins, is another mode of continental extension. It is not unlike the shedding of iceberg from the margin of polar ice caps. In this exercise we design a model of continental margin submitted to extension. The model dimension is 768 x 192 km. The margin itself is 256 km long, it joints a segment of oceanic lithosphere to a segment of continental lithosphere both segments are 256 km long each.



Chapter 5

Relax, Pylith

As demonstrated by aftershocks, earthquakes modify the local stress field on a scale proportional to the earthquake magnitude. Using a simple elastic halfspace with uniform isotropic elastic properties, it is possible to calculate stress changes following an earthquake, and predicts which faults have evolved toward their point of rupture. This information can be turned into probabilistic maps showing potential for future earthquakes.

Pylith and Relax are two open source, multi-platform codes able to evaluate Coulomb stress changes and displacement associated to earthquakes, interseismic deformation, volcanoes inflation, dike emplacement etc.

Section 1

Stress

In tectonics and structural geology, as well as mechanical engineering, the concept of *stress* is often substituted to the concept of *force*. This is because a force (i.e what changes the state of rest of an object, or its motion along a straight line, or its constant velocity) can lead to a different mechanical outcome when applied to different objects, even objects made of the same material.

The concept of stress is therefore intimately associated to the object on which the force is applied, whereas the definition of the applied force does not depend on the object on which it is applied (i.e. a traction a cable exerts on an object). This idea is illustrated on the sketch on the right.

Importantly, the concept of stress applies to “static” or “quasi-static” objects (objects that at equilibrium: which do not deform or accelerate or rotate significantly). In geology, most deformation occurs at a very slow rate (nanometer per second, or cm per year), and can therefore be treated as quasi-static deformation. Only during earthquakes (seismic slip) does deformation occurs at a speed in the range of meter per second.

Key papers

Toda, S., R. S. Stein, K. Richards-Dinger and S. Bozkurt, 2005, Forecasting the evolution of seismicity in southern California: Animations built on earthquake stress transfer, Journal of Geophysical Research, v. 110, B05S16, doi:10.1029/2004JB003415.

Lin, J. and R.S. Stein, 2004, Stress triggering in thrust and subduction earthquakes, and stress interaction between the southern San Andreas and nearby thrust and strike-slip faults, Journal of Geophysical Research,, v. 109, B02303, doi:10.1029/2003JB002607.

Fred F. Pollitz (USGS), Ross S. Stein (USGS), Volkan Sevilgen (USGS contractor), Roland Bürgmann (UC Berkeley), The 11 April 2012 M=8.6 East Indian Ocean earthquake triggered large aftershocks worldwide, Nature, doi: 10.1038/nature11504

Key videos

This video explains the basics of earthquake mechanics

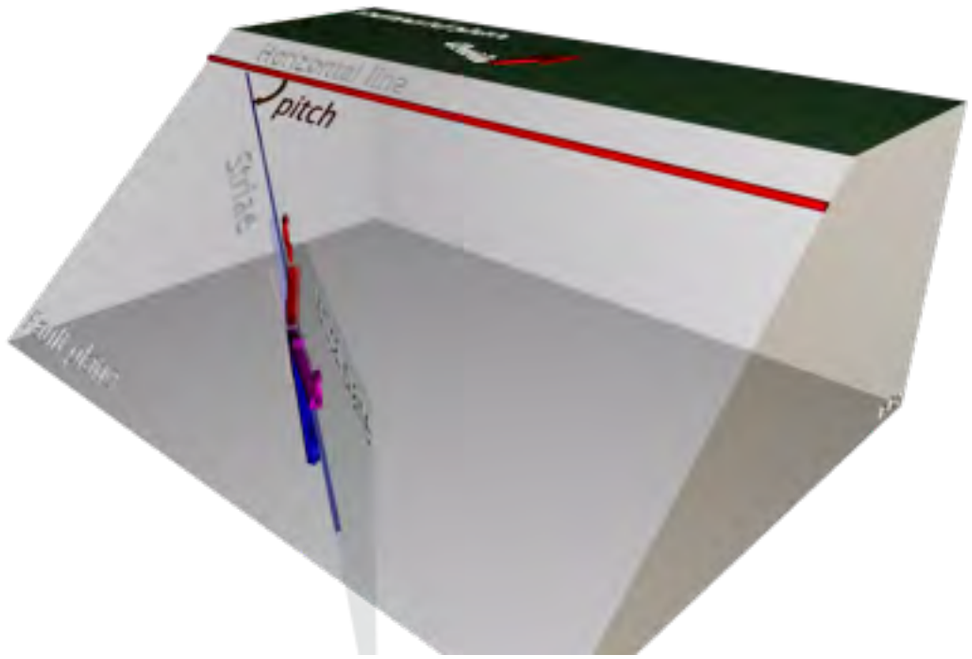
http://www.youtube.com/watch?v=eFrFdiWf8CA

MathLM reference: http://zvon.org/comp/r/ref-MathML_2.html

Coulomb stress

The Coulomb stress (σ) acting on a fault plane is represented by a couple of co-linear vectors (purple arrows below) pointing toward the fault. This stress can be decomposed into a normal stress (σ_n , *in red*) and a shear stress component (τ, in blue). The effective coefficient of friction (μ_{eff} , varies between 0 and 1) describes the fault's frictional resistance to slip. This parameter reduces the normal stress into an effective normal stress ($\sigma_n{}' = \mu_{eff} \cdot \sigma_n$). In contrast to the shear stress, the effective normal stress has a stabilizing effect since it clamps the fault by pressing the blocks on both side or the fault towards each others. The relationship between the Coulomb stress, the effective normal stress and the shear stress is:

$$\sigma = \tau + \mu_{eff} \cdot \sigma_n$$

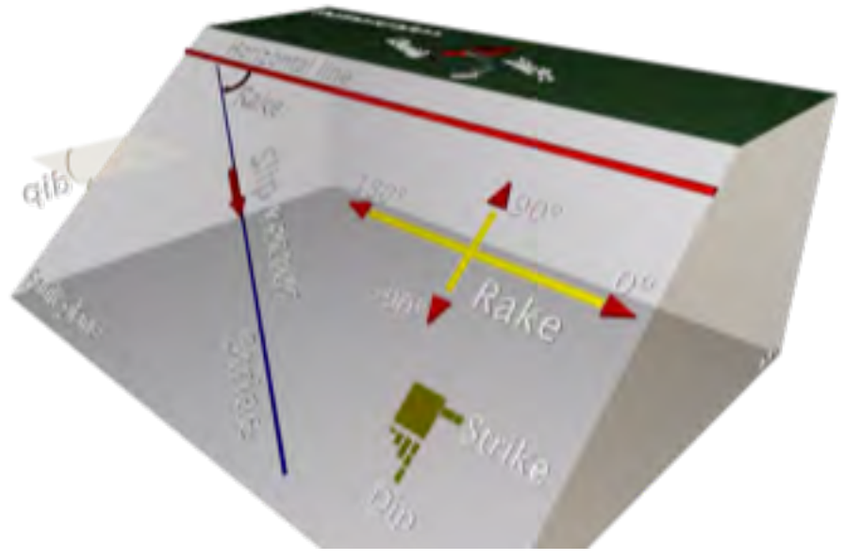


Earthquake occurs when the destabilizing effect of shear stress overcomes the stabilizing effective normal stress, at which point the Coulomb stress reaches the yield stress (σ_r). This is described by :

$$\sigma_r = \tau_r + \mu_{eff} \cdot \sigma_{nr}$$

in which τ_r and σ_{nr} are the the shear stress and normal stress at the time of rupture.

Earthquakes relax stress on ruptured faults but increase or decrease the Coulomb stress on adjacent faults. Further fault failure is promoted when the Coulomb stress acting on adjacent faults approaches the yield stress. This can be achieved by increasing the shear stress, decreasing the normal stress, decreasing the effective coefficient of friction or any combination of the above.



Following an earthquake, the Coulomb Failure Stress Change (CFSC) is described by:

$$\Delta\sigma = \Delta\tau + \mu_{eff} \cdot \Delta\sigma_n$$

By convention, when positive Coulomb failure stress change brings a fault toward its yield stress. This requires that $\Delta\tau$ is positive in the direction of fault slip, and that $\Delta\sigma_n$ is positive when the fault is unclamped. Therefore one needs to specify faults data (e.i.: strike, dip, rake) using the right-hand rule convention which imposes that the right-hand thumb points parallel to the strike direction, and the right-hand index points parallel to the dip of the fault plane. The rake is negative (0 to -180°) for faults with a normal component, and positive (0 to +180°) for faults with a reverse component. A left-lateral fault has a rake of 0°.

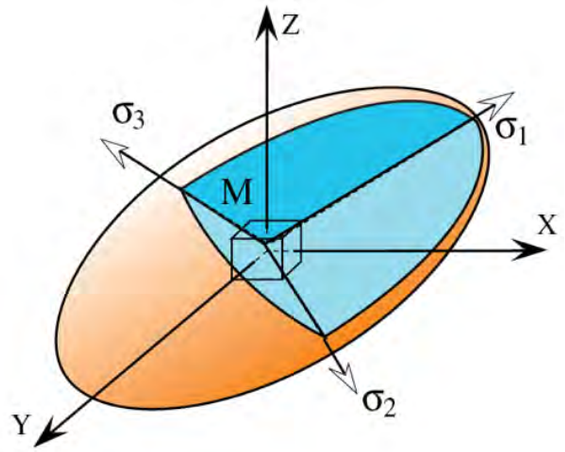
Coulomb stress and the general stress ellipsoid

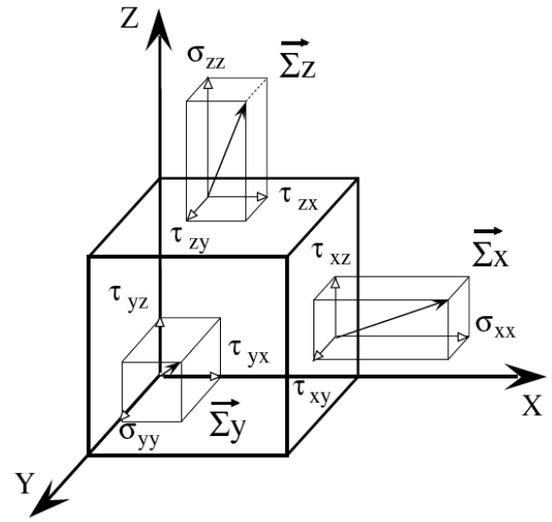
Coulomb stress refers to the stress acting on planar surface such as a fault. In contrast, the stress acting on a volume is described by a stress ellipsoid made of three perpendicular principal stress axes ($\sigma_1 \geq \sigma_2 \geq \sigma_3$). The sketch on the right considers a small cube of rock acted upon by a local stress field characterized by a stress ellipsoid. The Coulomb stress (Σ_x, Σ_y, Σ_z) acting three orthogonal cube faces can be written as a function of the principal stress axes of the local stress ellipsoid (σ_1 , σ_2 , σ_3), and the orientation of cube face (e.i., its normal in an orthogonal reference frame). The Coulomb stresses can be written as functions for their respective normal (σ_{ii}) and shear (τ_{ij}) stress components expressed in the *x, y, z* orthogonal reference framework attached to the cube:

$$\vec{\Sigma}x = \sigma xx\,\vec{i} + \tau xy\,\vec{j} + \tau xz\,\vec{k}$$
$$\vec{\Sigma}y = \tau yx\,\vec{i} + \sigma yy\,\vec{j} + \tau yz\,\vec{k}$$
$$\vec{\Sigma}z = \tau zx\,\vec{i} + \tau zy\,\vec{j} + \sigma zz\,\vec{k}$$

Note that the Coulomb stresses (Σ_x, Σ_y, Σ_z) acting on the cube faces vary in orientation and magnitude with the orientation of the cube.

Normal and shear stresses define the components of the stress tensor, which fully describes the local state of stress Although the components of this tensor vary in magnitude as a function of the orientation of the cube, the sum of the normal stresses ($\sigma_{xx} + \sigma_{yy} + \sigma_{zz}$) remains constant, and shear stresses component are symmetrically distributed about the normal stresses diagonal ($\tau_{ij} = \tau_{ji}$, i.e: the moments acting on each face cancel out, preventing the rotation of the cube).





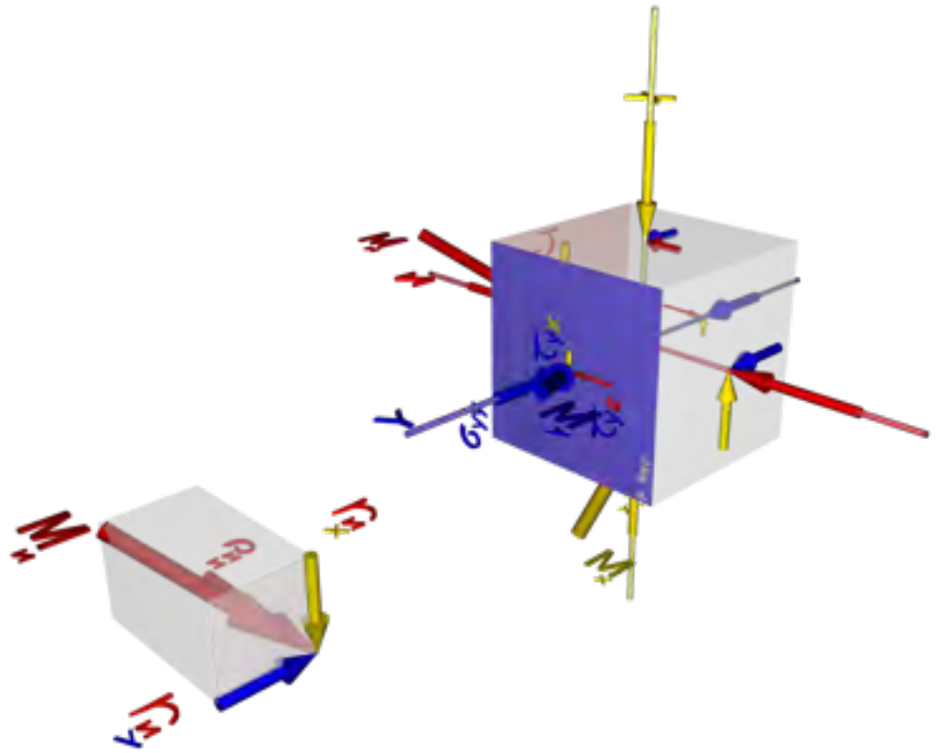
Stress tensor

$$\begin{bmatrix} \sigma xx & \tau xy & \tau xz \\ \tau yx & \sigma yy & \tau yz \\ \tau zx & \tau zy & \sigma zz \end{bmatrix}$$

$$\tau ij = \tau ji$$

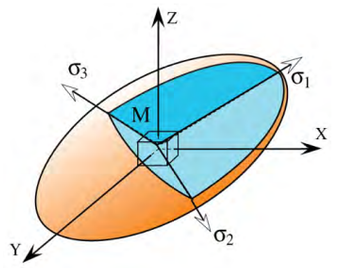
$\sigma xx + \sigma yy + \sigma zz$: Invariant

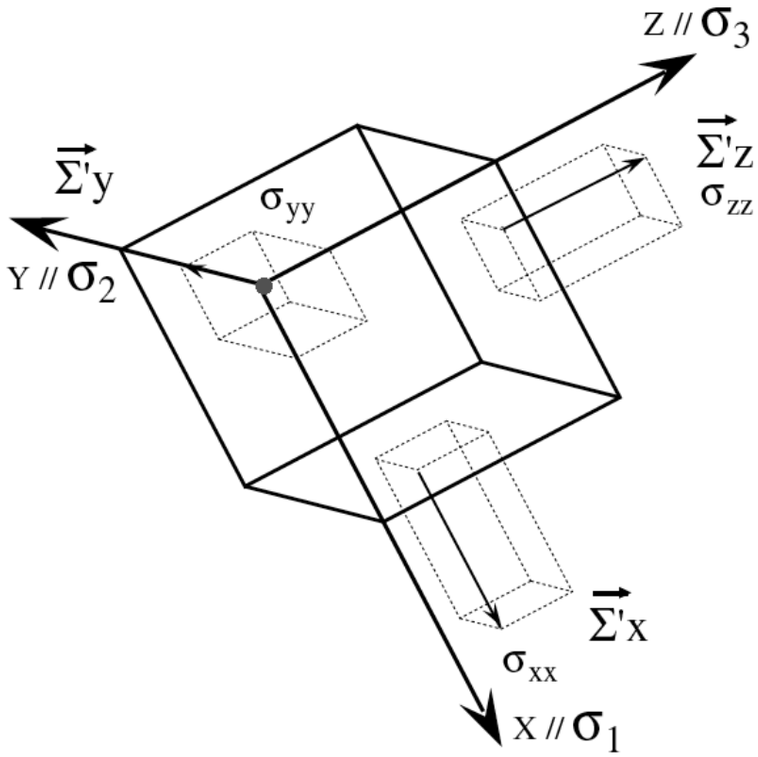
Interactive 5.1 General stress



NB: τ_{ij}, i refers to the face on which the stress is applied, j refers to the direction of the stress on this face.

There is one, and only one, orientation of the cube for which the Coulomb stresses act strictly perpendicularly to the cube faces (i.e., $\tau_{ij} = 0$). For this orientation, the principal stress axes of the stress ellipsoid (σ_1 , σ_2 , σ_3) make up the diagonal components (also called "*Eigen values*" of the stress tensor) of the stress tensor. This orientation is such that the *x, y, z* orthogonal reference framework attached to the cube is parallel to σ_1 , σ_2 , σ_3 principal stress axes. The angles between the axes of the old reference framework (six-component tensor) and that of the new reference framework (3-component tensor) are called the "*Eigen-vectors*".





Let's try to summarize the important points:

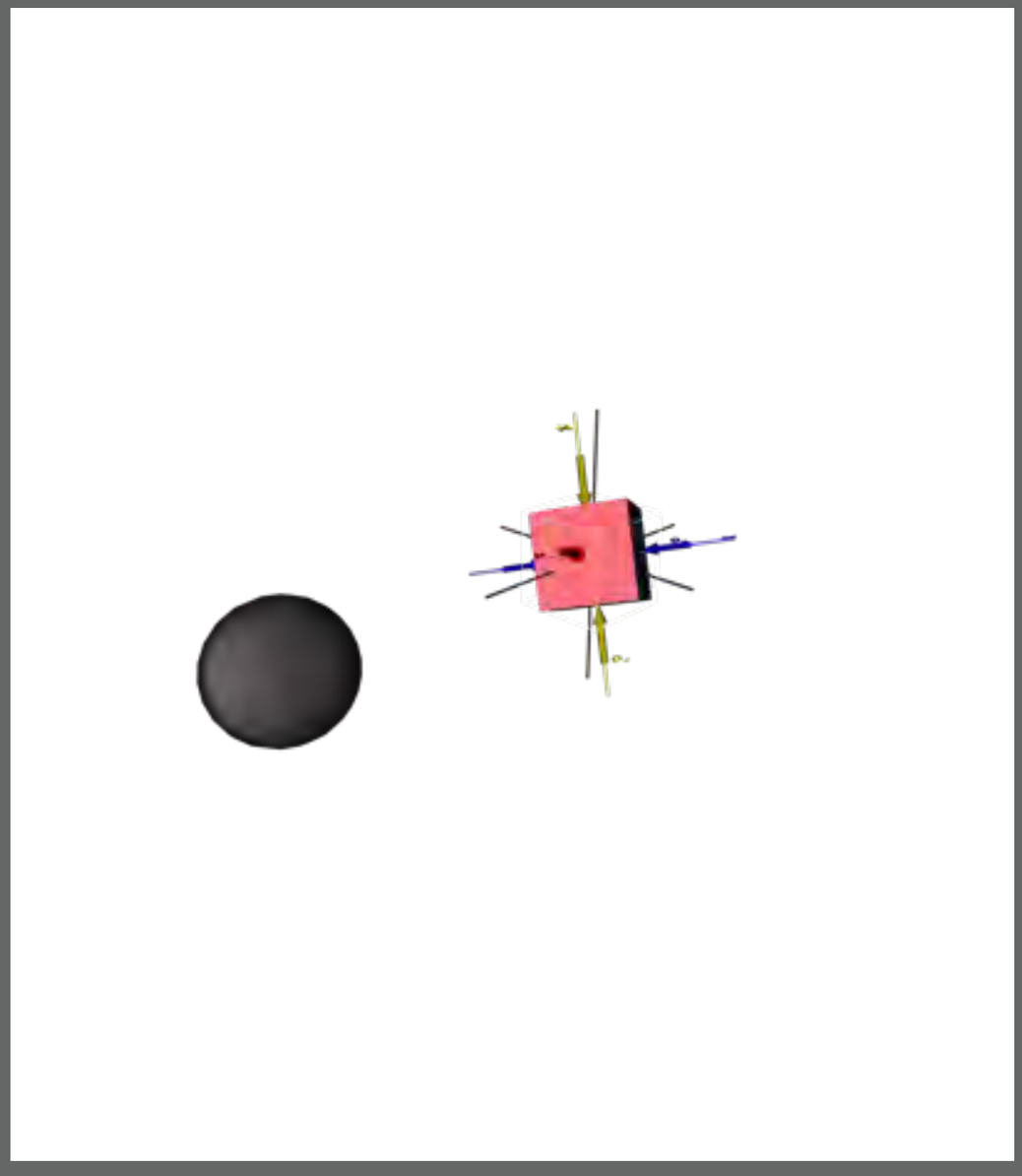
1/ The Coulomb stress acts on planar surfaces (real like fault plane, or virtual like the faces of an imaginary cube).

2/ The normal stress (σ_{nr}) and the shear stress (τ_r) are the components of the Coulomb stress in a framework attached to the planar surface.

3/ The stress ellipsoid (geometry) or stress tensor (mathematics) describes the stress acting on a volume (including an infinitely small volume). The stress ellipsoid is fully defined by the magnitude and orientation of the orthogonal principal stress axes ($\sigma_1 \geq \sigma_2 \geq \sigma_3$).

$$\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

Interactive 5.2 Stress tensor

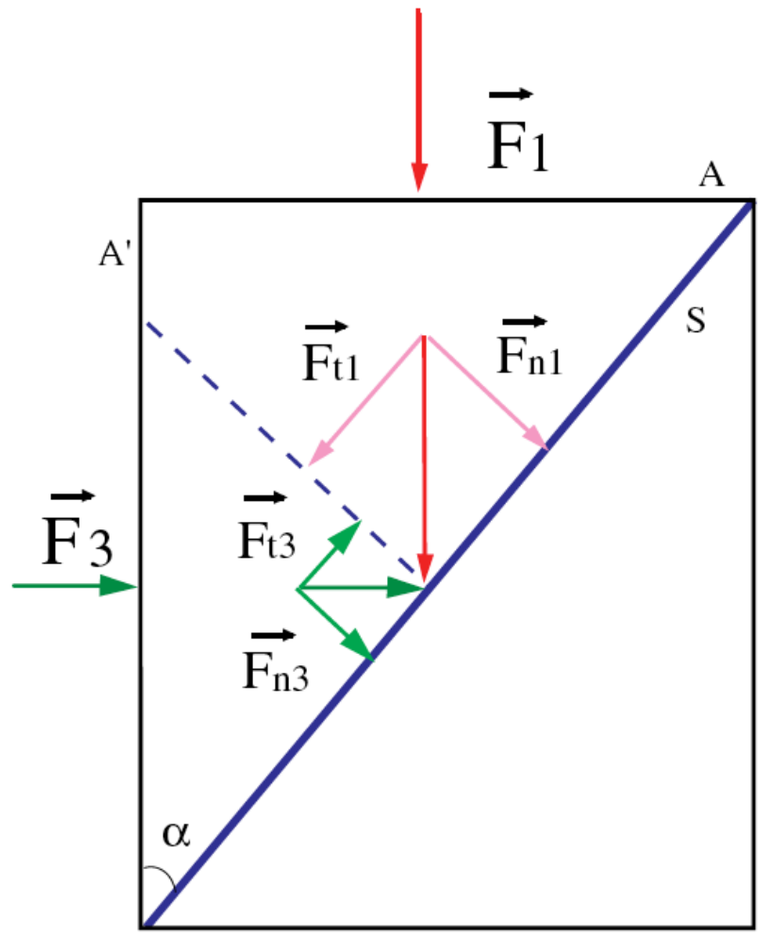


It exists one orientation of the cube for which the principal axes of the stress ellipsoid are perpendicular to the face of the cube

Coulomb stress from stress tensor and fault orientation: 2D problem

Here we derive the Coulomb stress acting on a planar surface *S* acted upon by a general stress ellipsoid.

We consider a simple cubic volume of rock. The block's faces are acted upon by forces F_1, F_2 and F_3 oriented perpendicularly to the block's faces. Inside the block, we consider an imaginary planar surface *S* oriented by an angle α with respect to F_1. We simplify the problem to a 2D problem by considering a planar surface parallel to F_2. Therefore F_2 does not contribute to normal stress or shear stress acting the surface *S*. This simplification is relevant to newly formed fault optimally oriented to maximize the Coulomb stress.



F1 and F3 can be decomposed into their normal and shear component:

$$F_{n1} = F_1 . \mathrm{Sin}(\alpha) \qquad F_{n3} = F_3 . \mathrm{Cos}(\alpha)$$
$$F_{t1} = F_1 . \mathrm{Cos}(\alpha) \qquad F_{t3} = -F_3 . \mathrm{Sin}(\alpha)$$

We switch from forces to stresses (stress = force/S) by nothing that $S = A/\mathrm{Sin}(\alpha)$

$$\sigma_{n1} = \sigma_1 . \mathrm{Sin}^2(\alpha) \Rightarrow \sigma_{n1} = \sigma_1 . (1-\mathrm{Cos}(2\alpha))/2 \qquad \sigma_{n3} = \sigma_3 . \mathrm{Cos}^2(\alpha) \Rightarrow \sigma_{n3} = \sigma_3 . (1+\mathrm{Cos}(2\alpha))/2$$
$$\tau_1 = \sigma_1 . \mathrm{Sin}(\alpha) . \mathrm{Cos}(\alpha) \Rightarrow \tau_1 = \sigma_1 . \mathrm{Sin}(2\alpha)/2 \qquad \tau_3 = -\sigma_3 . \mathrm{Cos}(\alpha) . \mathrm{Sin}(\alpha) \Rightarrow \tau_3 = -\sigma_3 . \mathrm{Sin}(2\alpha)/2$$

The resulting Coulomb normal stress and shear stress are the sum of the contributions from F1 and F3:

$$\sigma_n = \frac{\sigma_1 + \sigma_3}{2} - \frac{\sigma_1 - \sigma_3}{2} . \mathrm{Cos}(2\alpha) \qquad \tau = \frac{\sigma_1 - \sigma_3}{2} . \mathrm{Sin}(2\alpha)$$

These expressions give the Coulomb normal stress and the Coulomb shear stress acting on a surface containing σ_2 and making an angle α with σ_1.

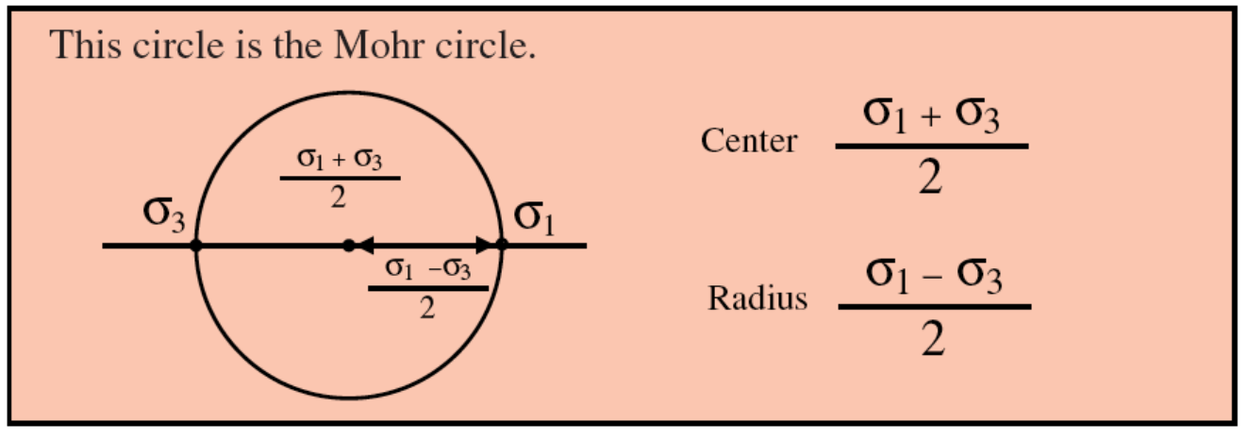
$$\sigma_n = \frac{\sigma_1 + \sigma_3}{2} - \frac{\sigma_1 - \sigma_3}{2}.\mathrm{Cos}(2\alpha), \quad \tau = \frac{\sigma_1 - \sigma_3}{2}.\mathrm{Sin}(2\alpha)$$

The shear stress varies between 0 (when σ_1 is perpendicular to the surface), to half the difference between σ_1 and σ_3 (when the surface is at α = 45° to σ_1).

The normal stress varies between σ_1 (when σ_1 is perpendicular to the surface) and σ_3 (when σ_3 is perpendicular to the surface).

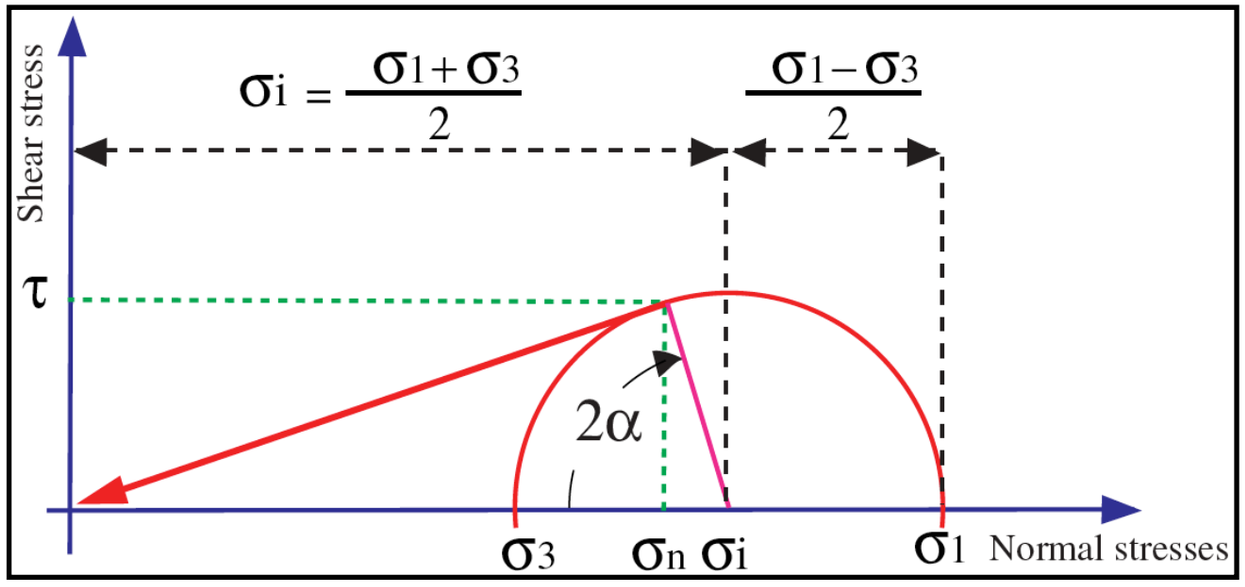
The Mohr circle (1)

If one plots σ_1 , σ_2 and σ_3 along an axis, one can draw a circle called the Mohr circle the center and radius of which are $\frac{\sigma_1 + \sigma_3}{2}$ and $\frac{\sigma_1 - \sigma_3}{2}$ respectively.



Along this axis one can also plot the Coulomb normal stress σ_n. However, since the Coulomb shear stress is perpendicular the Coulomb normal stress, the shear stress is better plotted on an axis perpendicular to the axis carrying the normal stresses.

The resulting Coulomb stress can be derived from both the normal and shear stress components (e.i. the red vector on the sketch on the right). This construction reveals also the value of the angle α between the surface *S* and σ_1.



The Mohr circle (2)

The geometric construction on the right gives the Coulomb stress acting on a surface *S* acted upon by a state of stress: σ_1 , σ_2 , σ_3.

Givens: σ_1 , σ_2 , σ_3 and the angle α between *S* and σ_1. *S* contains σ_2.

Find: σ, σ_n and τ.

Solution:

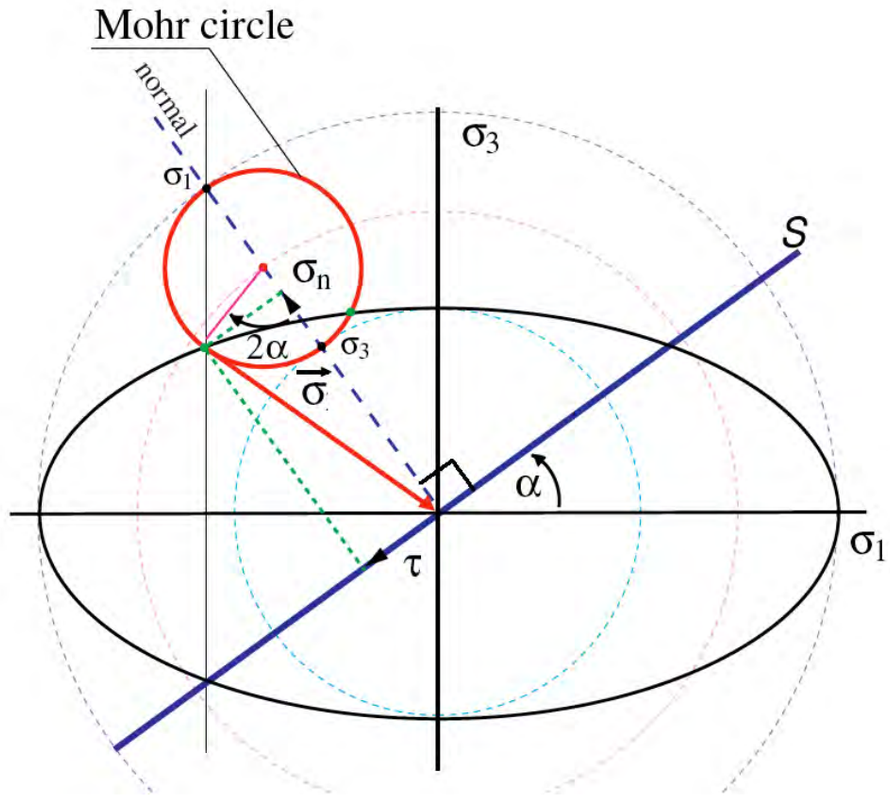
1/ Draw two orthogonal axes representing the orientation of σ_1 and σ_3.

2/ Draw the ellipse defined by σ_1 and σ_3 (e.i, $\frac{x^2}{\sigma^2{}_1} + \frac{y^2}{\sigma^2{}_3} = 1$).

3/ Draw the surface S at an angle α to σ_1, and its normal.

4/ On the normal to *S* draw the Mohr circle (center: $\frac{\sigma_1 + \sigma_3}{2}$, radius: $\frac{\sigma_1 - \sigma_3}{2}$).

5/ Locate the Mohr circle radius with an angle 2α with the normal to *S*.



6/ The Coulomb stress σ is the vector defined by the center of the ellipse and the point common to the stress ellipse and the Mohr radius at an angle 2α from the normal to *S*. The Coulomb stress σ can then be decomposed into its normal and shear components.

What have we learnt?

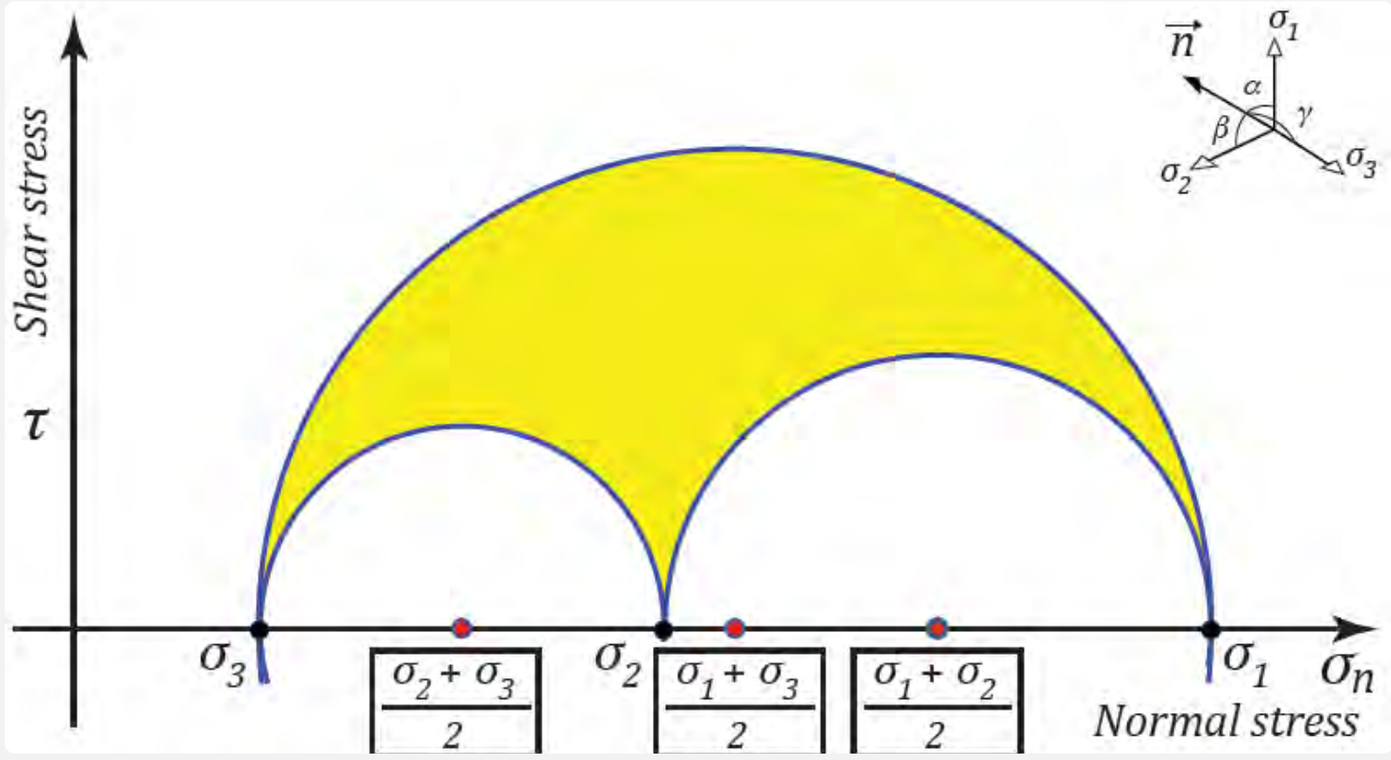
Given the magnitude and orientation of the principal stress axes (σ_1 and σ_3), and given a surface containing σ_2 and oriented an angle (α) with respect to σ_1, it is possible - using the construction of a Mohr diagram - to determine the orientation and magnitude of the Coulomb stress and that of its normal and shear components.

Coulomb stress from stress tensor and fault orientation: 3D problem

What if the consider planar surface does not contain a principal stress axis? How the normal and shear stress components of the Coulomb stress would depend on the principal stress axes?

When the surface under consideration contains one principal stress axis (σ_1 or σ_2 or σ_3.), then the Mohr circle as well as the normal and shear stress are defined by the two other principal stress axes. In each case the Coulomb stress joins the origin of the stress reference frame (σ_n, τ) to a point on the Mohr circle determined by the Mohr radius at a 2α angle from the normal stress axis at the center of the Mohr circle.

Gallery 5.1 3D Mohr circle



Given the three principal stress (σ_1, σ_2 and σ_3) construct the three Mohr circles as shown.

When the surface does not contain any of the principal stress axis, then the Coulomb stress vector ends up on the surface defined by the 3 Mohr circles (e.i. the surface colored in yellow on the sketch above). The tip on the Coulomb stress vector is at the intersection between three circles (dashed lines) centered at: $\frac{\sigma_1+\sigma_2}{2}$, $\frac{\sigma_1+\sigma_3}{2}$, and $\frac{\sigma_2+\sigma_3}{2}$. The radiuses of each Mohr circle are defined by the intersections P_i between each Mohr circle and the green line at an angle α from σ_1 , β from σ_2 , and γ from σ_3. α, β and γ are the angles between the normal *n* to the surface *S* and σ_1, σ_2 and σ_3 respectively. Once the Coulomb stress vector is constructed its component can be determined.

$$\eta = \frac{1}{2A^{1/n}} Exp(\frac{Q}{nRT}) \dot{\epsilon}^{\frac{1-n}{n}}$$