Thoughts about time stepping in real-world multiphysics PDE solvers

Wolfgang Bangerth

Colorado State University

Joint work with Juliane Dannberg, Menno Fraters, Rene Gassmoeller, Anne Glerum, Timo Heister, Bob Myhill, John Naliboff, and many many other contributors

With funding by:

National Science Foundation
WHERE DISCOVERIES BEGIN

CIG COMPUTATIONAL INFRASTRUCTURE for GEODYNAMICS

Theory v Practice

Wolfgang Bangerth

Colorado State University

Joint work with Juliane Dannberg, Menno Fraters, Rene Gassmoeller,
Anne Glerum, Timo Heister, Bob Myhill, John Naliboff,
and many many other contributors

With funding by:

National Science Foundation
WHERE DISCOVERIES BEGIN

COMPUTATIONAL INFRASTRUCTURE for GEODYNAMICS

Take-away message (charitable version):

The world is messy:

- **Models are messy**
- **Development histories are messy**

Take-away message (not-so-charitable version):

The developers of this code are amateurs:

- **We were (and are) not experts in time stepping**
- **We failed to anticipate future directions when coming up with the original design**

In particular: There was no “holistic design” at the beginning.

Take-away message (realistic version):

Real-world simulators often do not fit into typical categories.

Practicalities of code development limit both original choice of and later conversion to different methods.

Overview:

- The model, and how we solve it in the Advanced Solver for Planetary Evolution, Convection and Tectonics (ASPECT)

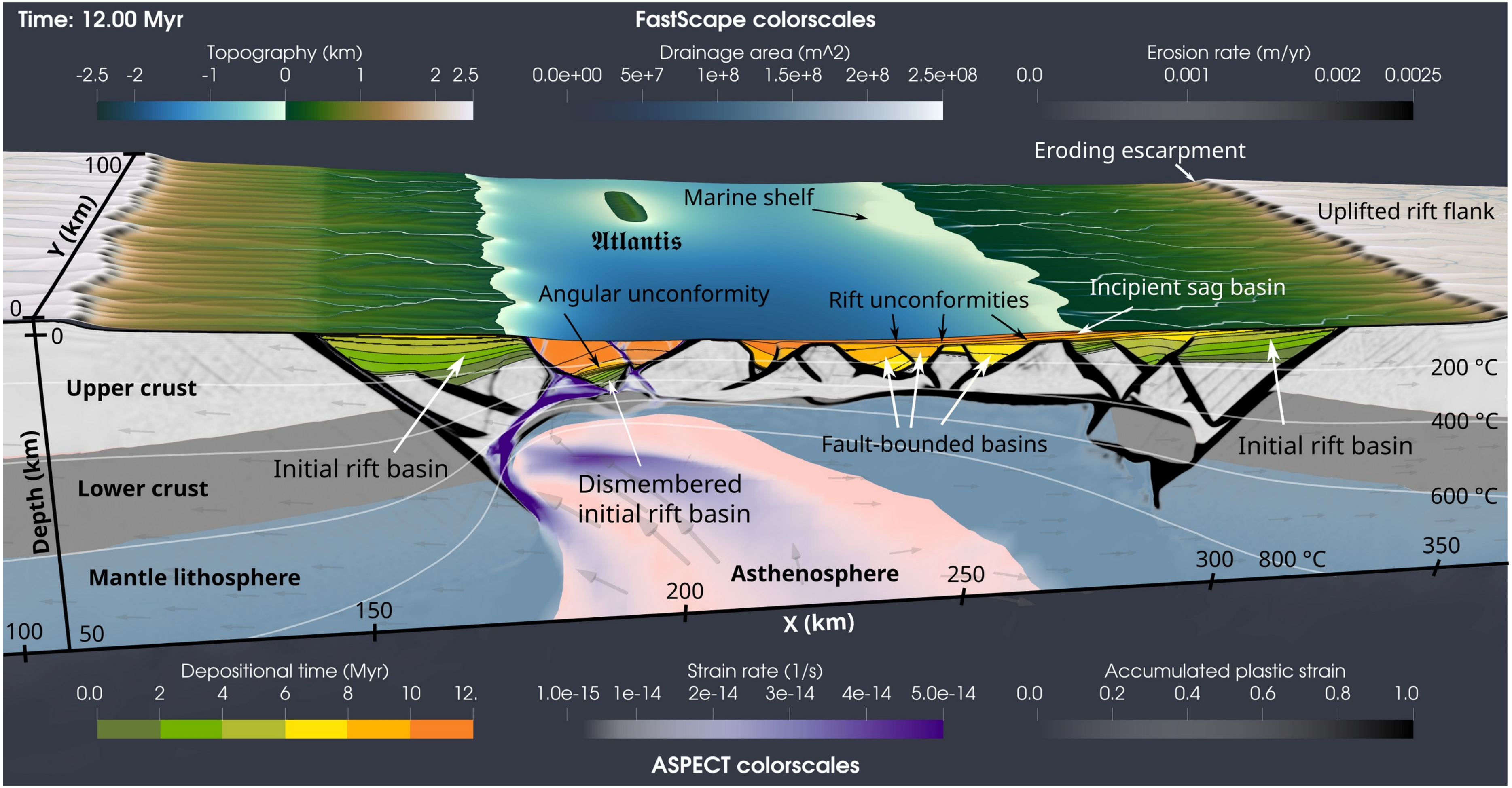
- What I would love to do

- Why I can’t do it

- What *does* work today

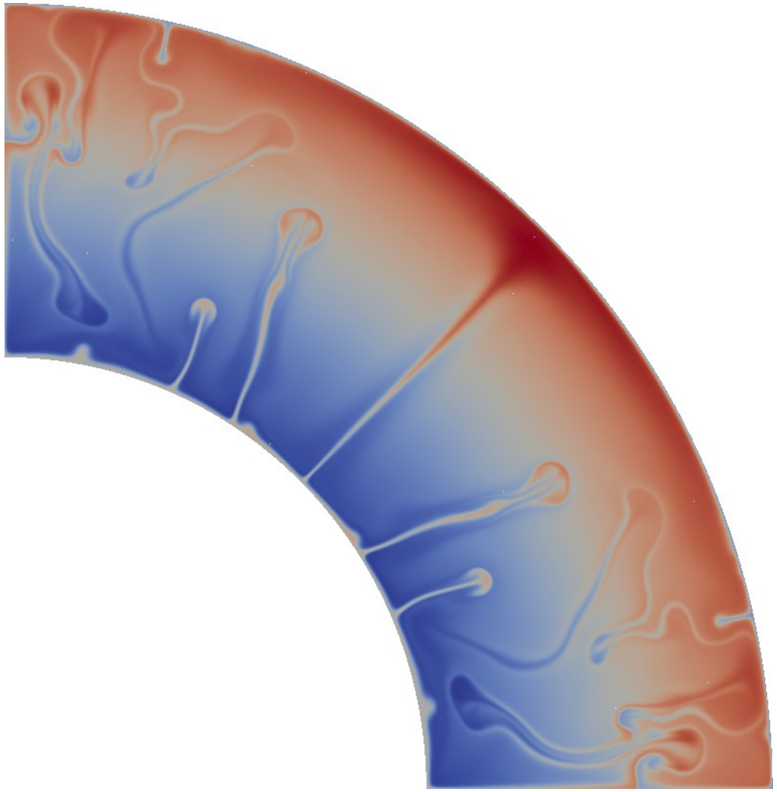
The model and how we solve it

The what and the why



Component 1:

“Classical” mantle convection



The model

Motion in the deep Earth is driven by density differences due temperature differences ("convection").

On long time scales, rocks behave like viscous Stokes flow:

$$-\nabla \cdot [2\eta \epsilon(u)] + \nabla p = g \rho(T)$$

$$\nabla \cdot u = 0$$

$$\frac{\partial T}{\partial t} + u \cdot \nabla T - \kappa \Delta T = \gamma + \alpha \left(\frac{\partial p}{\partial t} + u \cdot \nabla p \right) + \eta (\nabla u)^2$$

Stokes + advection-diffusion: not dissimilar from a typical "model problem". (But note: It's a DAE.)

Challenges: Problem size

For (global) convection in the earth mantle:

Depth:	2890 km
Volume:	~10^{12} km^3
Resolution required:	<10 km
Uniform mesh:	~10^9 cells
Using Taylor-Hood (Q_2/Q_1) elements:	33B unknowns
At 100k-1M DoFs/processor:	30k-300k processors!

Better: Can reduce these numbers 10x to 100x through adaptive mesh refinement.

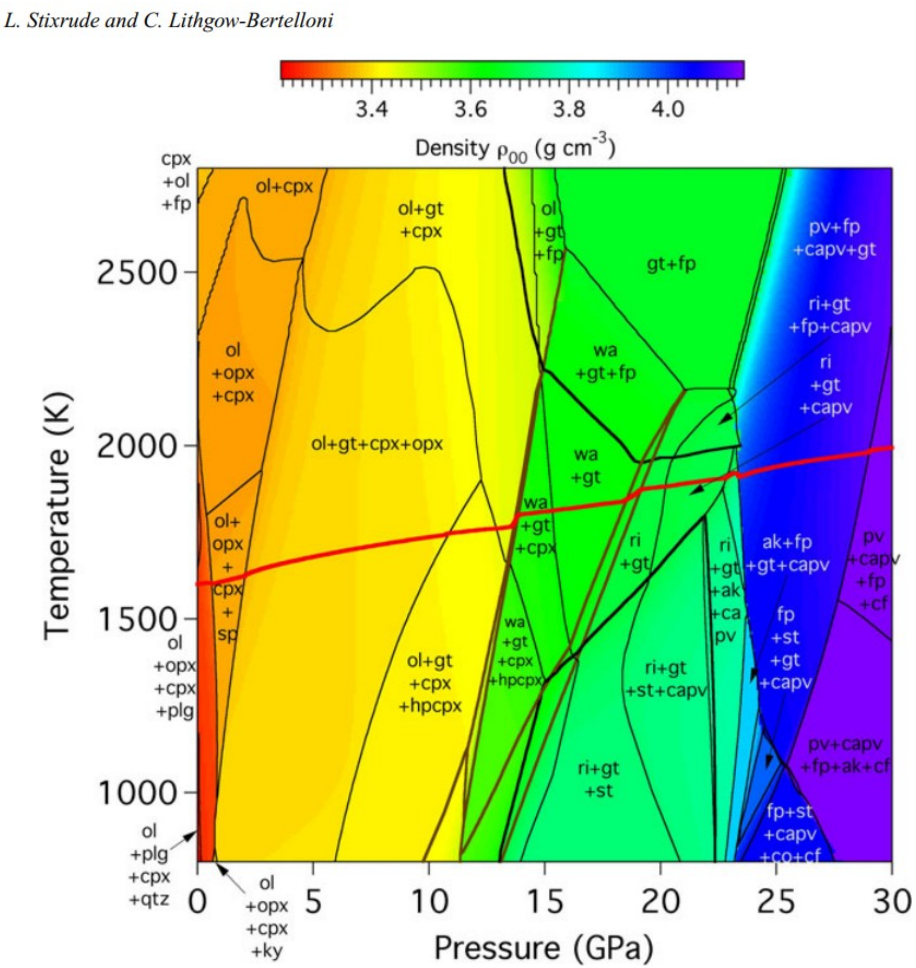
Challenges: Model complexity

However, in reality:

- All coefficients depend nonlinearly on
  - pressure
  - temperature
  - strain rate
  - chemical composition
- Dependency is not continuous

Moreover:

- Viscosity varies by at least 10^6
- Material is compressible
- Geometry depends on solution



Solutions

Among the mathematical techniques we use are:

- *“Higher” order time stepping schemes*
- Higher order finite elements
- Fully adaptive, dynamically changing 3d meshes
- Iterate out the nonlinearity via fixed-point and Newton methods
- Silvester/Wathen-style block preconditioners with F-GMRES
- Algebraic or geometric multigrid for the elliptic part
- Parallelization using MPI, threads, and tasks

To make the code usable by the community:

- Use object-oriented programming, build on external tools
- Make it modular, separate concerns
- Extensive documentation
- Extensive and frequent testing

Time discretization

Recall the model:

$$-\nabla\cdot[2\eta\epsilon(u)]+\nabla p=g\rho(T)$$

$$\nabla\cdot u=0$$

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

This is a nonlinear Differential-Algebraic Equation (DAE).

In practice:

- Solve Stokes with extrapolated *T* for *u*,*p*
- Solve advection/diffusion with *u*,*p* for temperature *T*

Time discretization

Overall algorithm:

While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step via CFL condition
- Solve for temperature field
- Advance time

Time discretization for the temperature

Time equation:

$$\frac{\partial T}{\partial t} + u \cdot \nabla T - \kappa \Delta T = \gamma + \alpha \left(\frac{\partial p}{\partial t} + u \cdot \nabla p \right) + \eta (\nabla u)^2$$

Considerations:

- Adaptive mesh refinement → no high-order multistep methods
- Variable time step size → no high-order multistep methods
- Segregated solver → velocity not available at intermediate times → what to do about RK methods?
- Spatial error dominant (?) → high order not necessary (?)

Our choice: BDF2

Time discretization for the temperature

BDF2 applied to

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

results in

$$\alpha_n T^n+u^n\cdot\nabla T^n-\kappa\Delta T^n=F\left(u^{n-1},T^{n-1},u^{n-2},T^{n-2}\right)$$

Considerations:

- We need an efficient linear solver for the discretized system
- The matrix is non-symmetric
- Treat advection as explicit instead:

$$\alpha_n T^n-\kappa\Delta T^n=-u^n\cdot\nabla T^*+F\left(u^{n-1},T^{n-1},u^{n-2},T^{n-2}\right)$$

Time discretization for the temperature

Semi-implicit BDF2:

$$\alpha_n T^n - \kappa \Delta T^n = -u^n \cdot \nabla T^* + F\left(u^{n-1}, T^{n-1}, u^{n-2}, T^{n-2}\right)$$

Consequences:

- The matrix is now symmetric
- Efficient linear solvers are easy to construct
- **But:** We now have a time step restriction

$$k_n \leq C_{\mathrm{BDF2}} \min_{K \in T} \frac{h_K}{\|u\|_{L_\infty(K)}}$$

Time discretization for the temperature

CFL condition – the struggle is real:

$$k_n \leq C_{\mathrm{BDF2}} \min_{K \in T} \frac{h_K}{\|u\|_{L_\infty(K)}}$$

Questions:

- What is C_{BDF2}?
- What is h_K on unstructured 3d meshes with curved edges?
- How does all of this relate to the eigenvalues of the matrix?

After much experimentation:

- Choose h_K as the diameter of K
- Choose C_{BDF2}=0.085 → quite small actually

Time discretization for the temperature

After much agony, change of mind – go back to fully implicit:

$$\alpha_n T^n + u^n \cdot \nabla T^n - \kappa \Delta T^n = F\left(u^{n-1}, T^{n-1}, u^{n-2}, T^{n-2}\right)$$

Considerations:

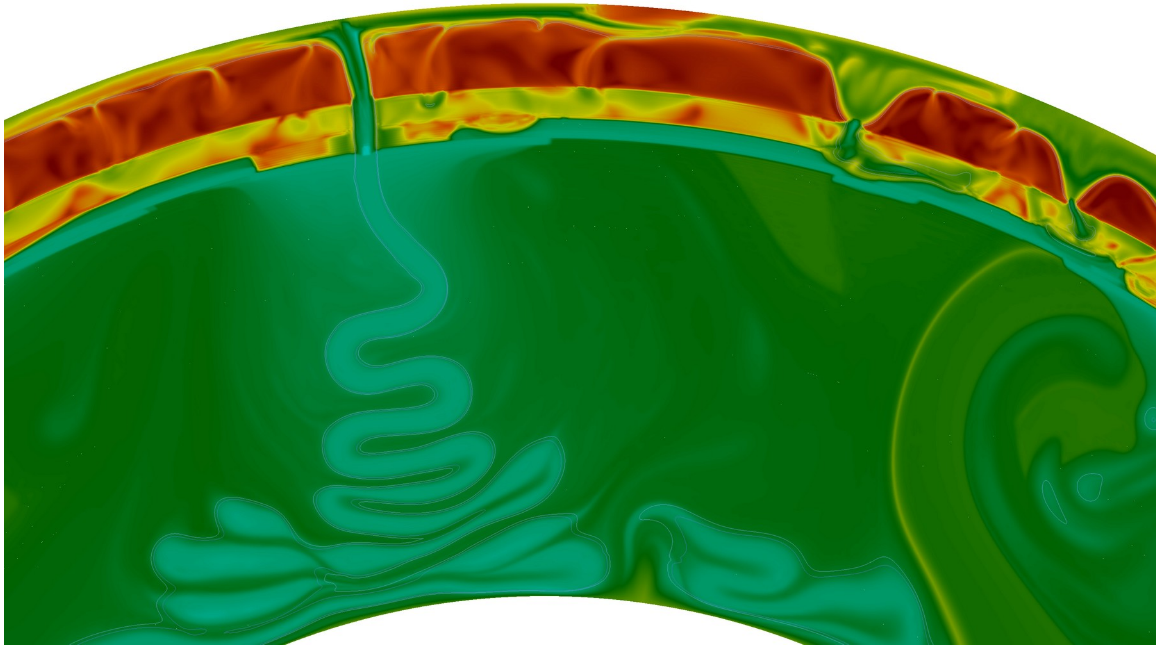
- Need to work harder to solve linear system
- But no longer time-step restricted; choose

$$k_n \;=\; 1 \cdot \min_{K \in T} \frac{h_K}{\|u\|_{L_\infty(K)}}$$

Because the Stokes solve is so expensive, the larger time step easily balances the more expensive temperature solve.

Component 2:

Compositional fields



Compositional fields

Juliane Dannberg (2012):

We also want to track chemical compositions:

$$-\nabla\cdot[2\eta\epsilon(u)]+\nabla p=g\rho(T)$$

$$\nabla\cdot u=0$$

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

$$\frac{\partial c_1}{\partial t}+u\cdot\nabla c_1=q_1(u,p,T,\vec{c})$$

$$\ldots$$

$$\frac{\partial c_N}{\partial t}+u\cdot\nabla c_N=q_N(u,p,T,\vec{c})$$

Chemical compositions

Considerations:

- Originally meant to track *compositions* → zero right hand sides
- Then *mineral compositions* → chemical reactions
- Then also other quantities (melting, accumulated strains, level sets, …) → many many such fields

- Would like to solve in a coupled fashion, but too memory expensive
- Solving advection equations suddenly becomes expensive

- Solve in segregated fashion, treat rhs explicitly

Time discretization

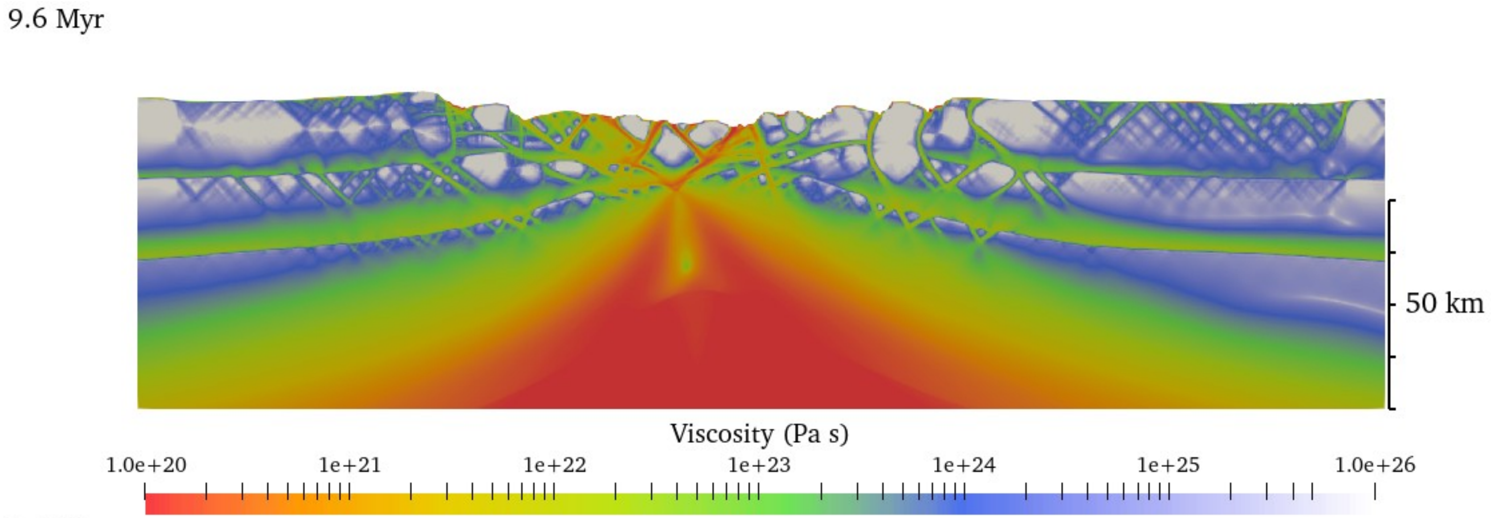
Overall algorithm:

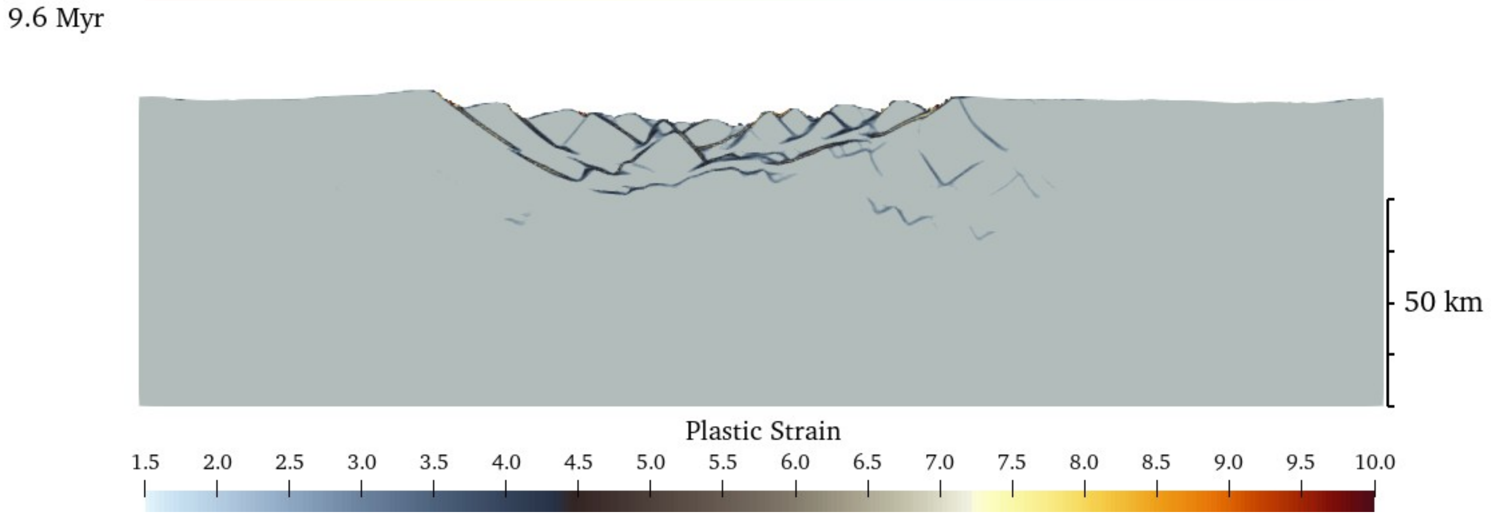
While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1 with implicit BDF2, explicit rhs
- …
- Solve for compositional field *N* with implicit BDF2, explicit rhs
- Advance time

Component 3:

Stiff source terms





Stiff source terms

John Naliboff, Juliane Dannberg, et al. (2017):

“compositional field” is elastic stress, which decays quickly in time:

$$-\nabla\cdot[2\eta\epsilon(u)]+\nabla p=g\rho(T)$$

$$\nabla\cdot u=0$$

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

$$\frac{\partial c_1}{\partial t}+u\cdot\nabla c_1=\boxed{q_1(u,p,T,\vec{c})}$$

$$\ldots$$

$$\frac{\partial c_N}{\partial t}+u\cdot\nabla c_N=\boxed{q_N(u,p,T,\vec{c})}$$

Stiff source terms

Considerations:

- Elastic stress relaxes on a time scale faster than the flow
  $\rightarrow$ "multirate" system
- Impossible to resolve this time scale in a coupled scheme
- Treat rhs via operator splitting:
  – currently uses Lie splitting
  – originally used pointwise ODE with fixed micro timestep
  – now uses SUNDIALS ARKODE

Time discretization

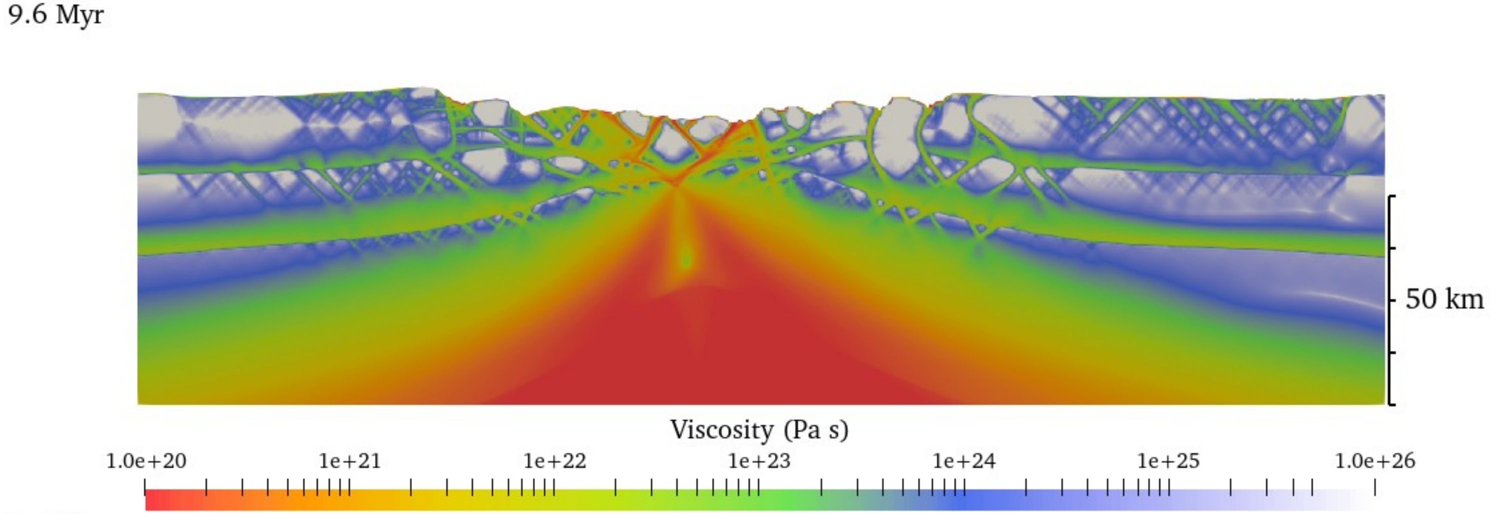
Overall algorithm:

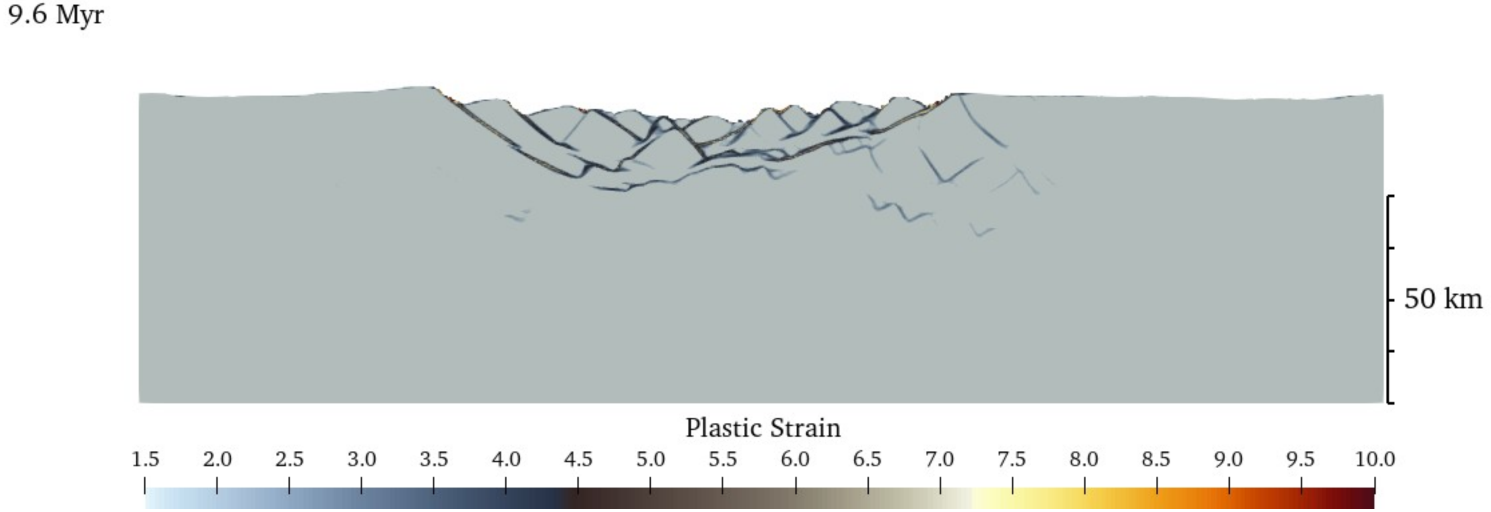
While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1, implicit BDF2, operator splitting
- …
- Solve for compositional field 1, implicit BDF2, operator splitting
- Advance time

Component 4:

Free surfaces





Free surfaces

Ian Rose and Timo Heister (2014):

- We also want to deform the surface of the domain
- Evaluate residual stresses at the surface, move nodes at boundary and in domain
- Requires one Laplace solve

Anne Glerum (2020):

- Diffuse the surface to mimic erosion

Free surfaces

Equations now:

$$-\nabla\cdot[2\eta\epsilon(u)]+\nabla p=g\rho(T)$$

$$\nabla\cdot u=0$$

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

$$\frac{\partial c_1}{\partial t}+u\cdot\nabla c_1=q_1(u,p,T,\vec{c})$$

$$\dots$$

$$\frac{\partial c_N}{\partial t}+u\cdot\nabla c_N=q_N(u,p,T,\vec{c})$$

$$\boxed{\frac{\partial h}{\partial t}-A\Delta h=r(u,p)}$$

Time discretization

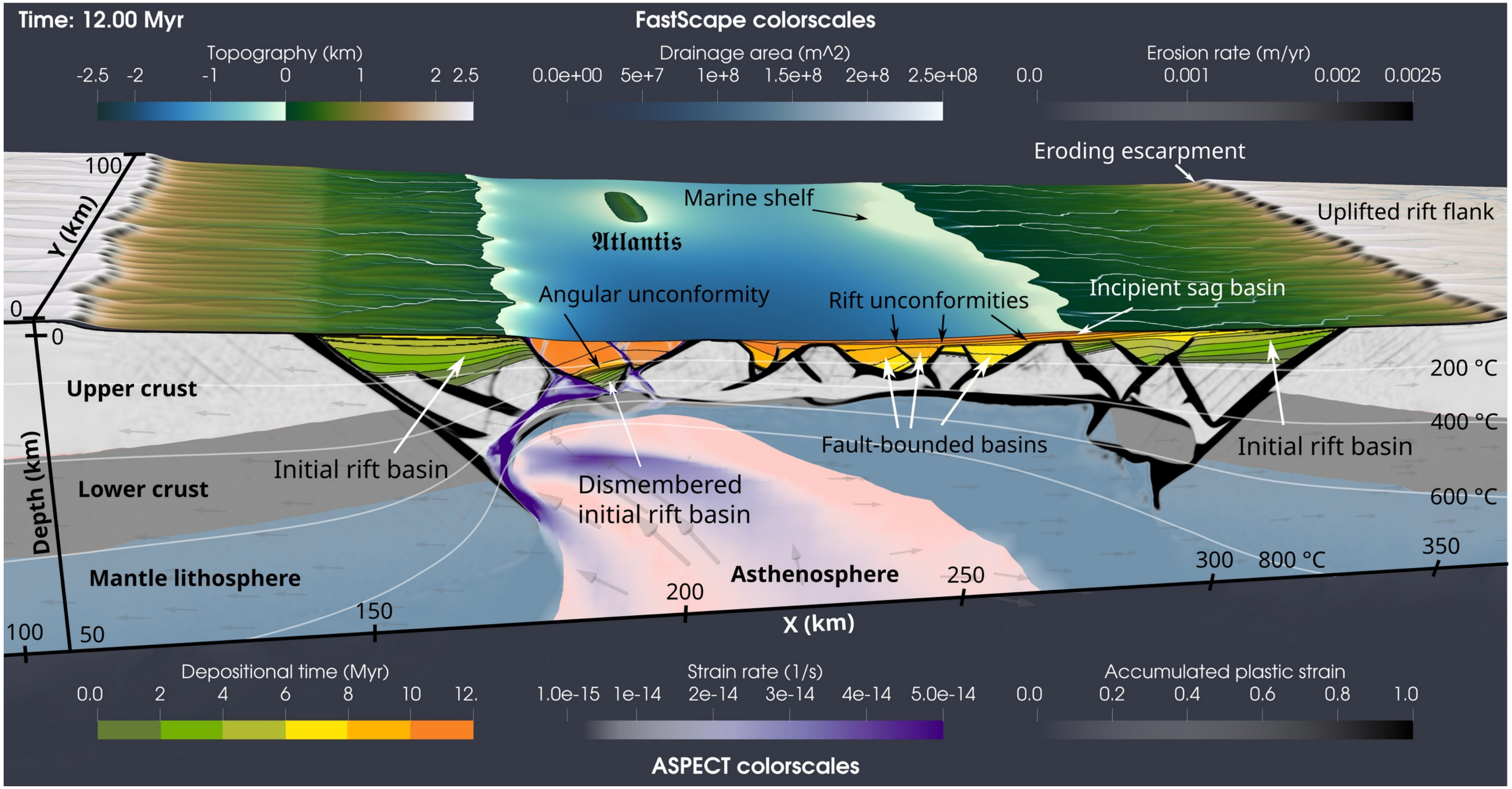
Overall algorithm:

While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1, implicit BDF2, operator splitting
- …
- Solve for compositional field 1, implicit BDF2, operator splitting
- Solve for surface deformation
- Advance time

Component 5:

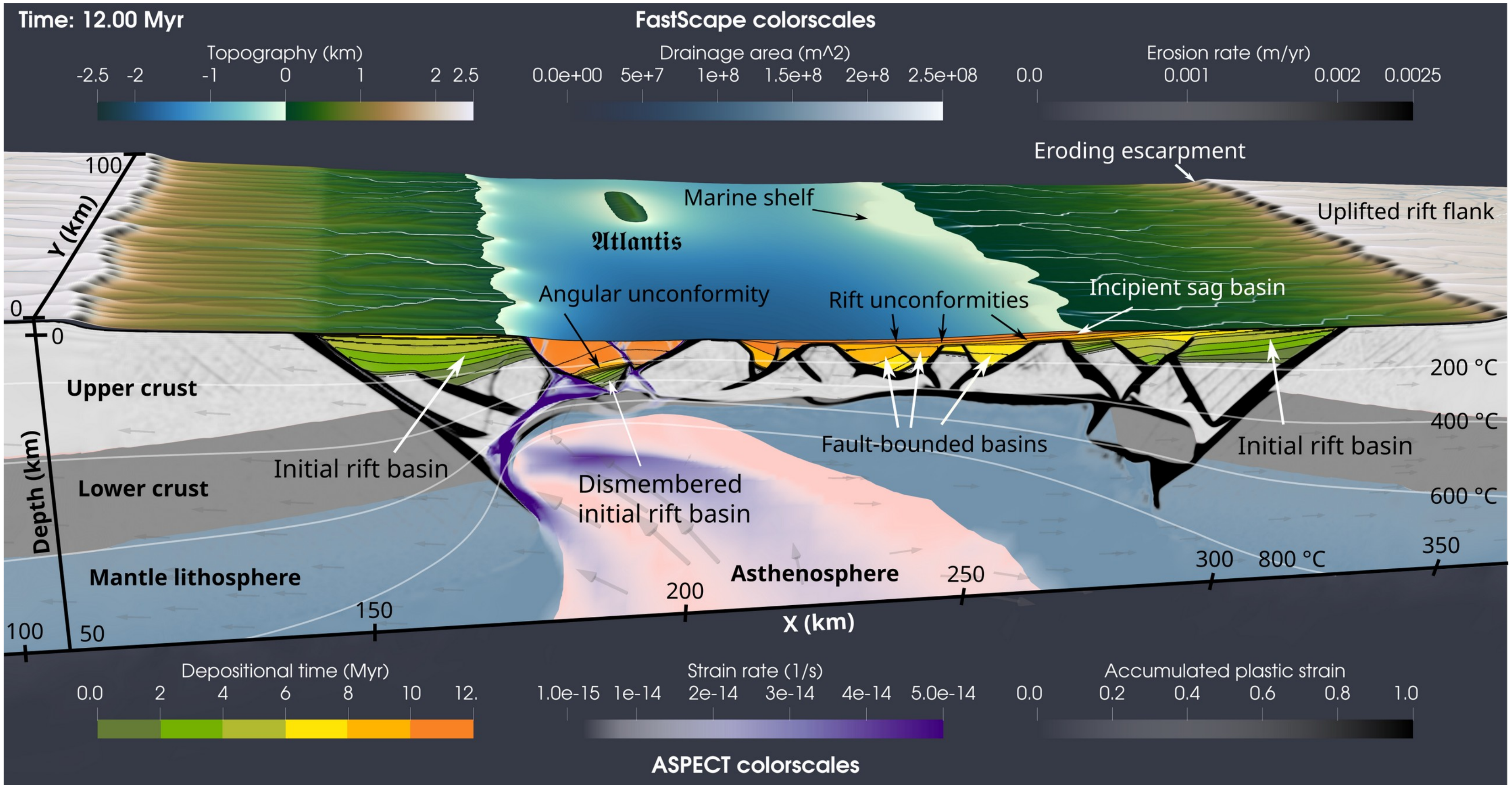
Surface evolution



Realistic surfaces

Derek Neuharth et al. (2021):

"Real" surface models are too complicated to re-implement in ASPECT. Couple with an external code: FastScape or others



Time discretization

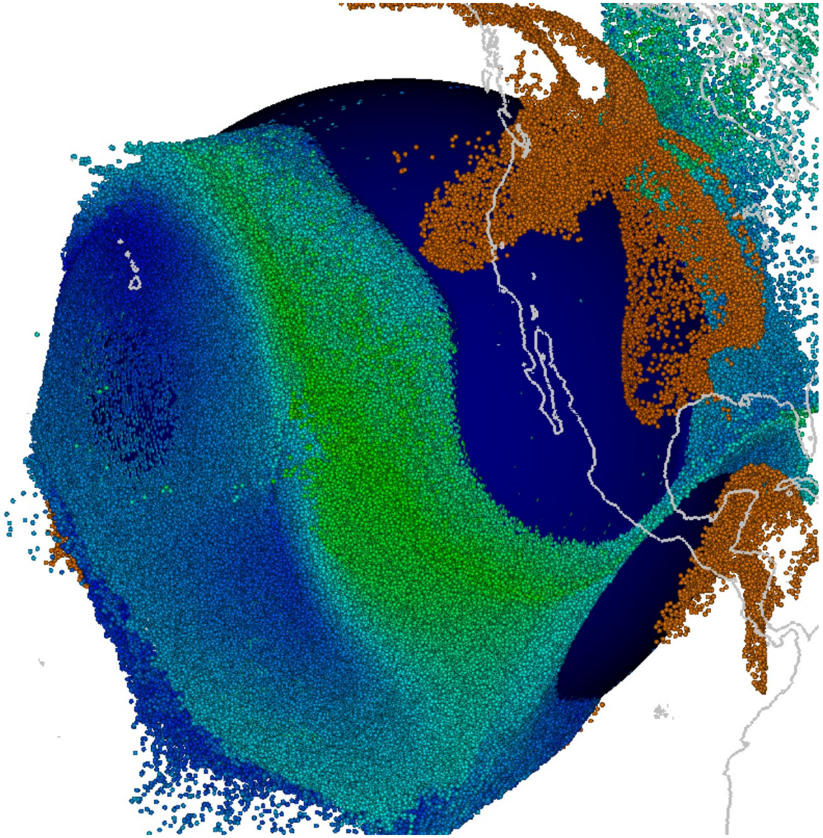
Overall algorithm:

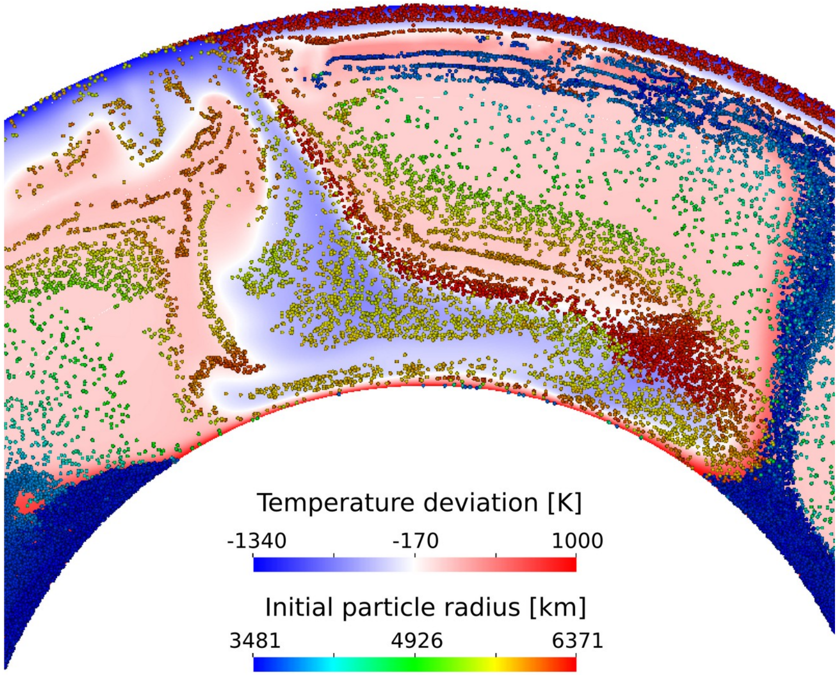
While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1, implicit BDF2, operator splitting
- …
- Solve for compositional field 1, implicit BDF2, operator splitting
- Solve for surface deformation, coupled with external tool
- Advance time

Component 6:

Particles

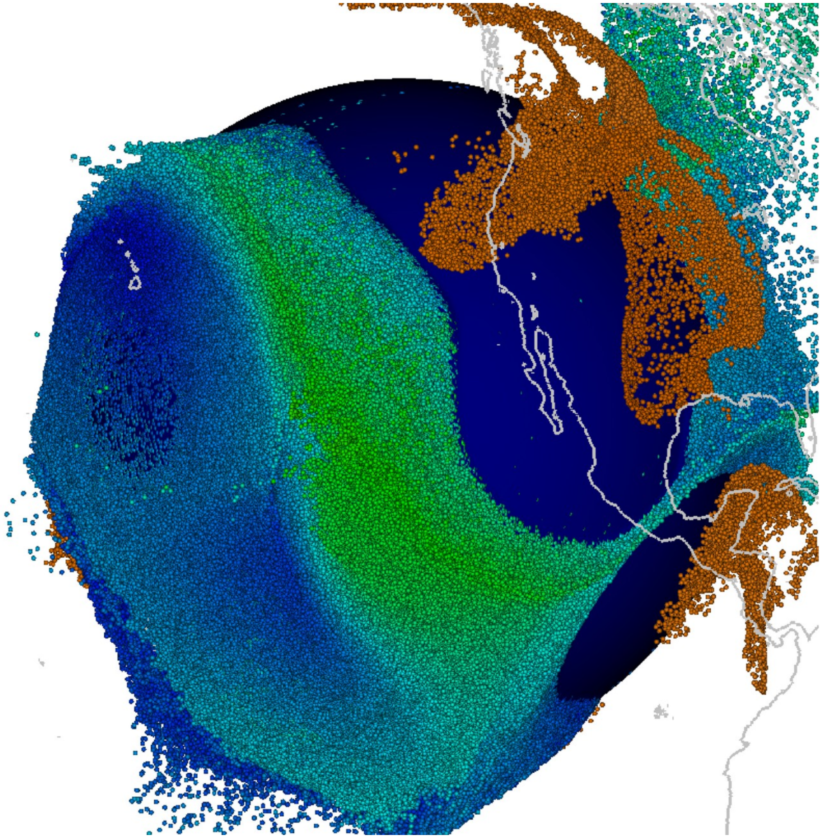


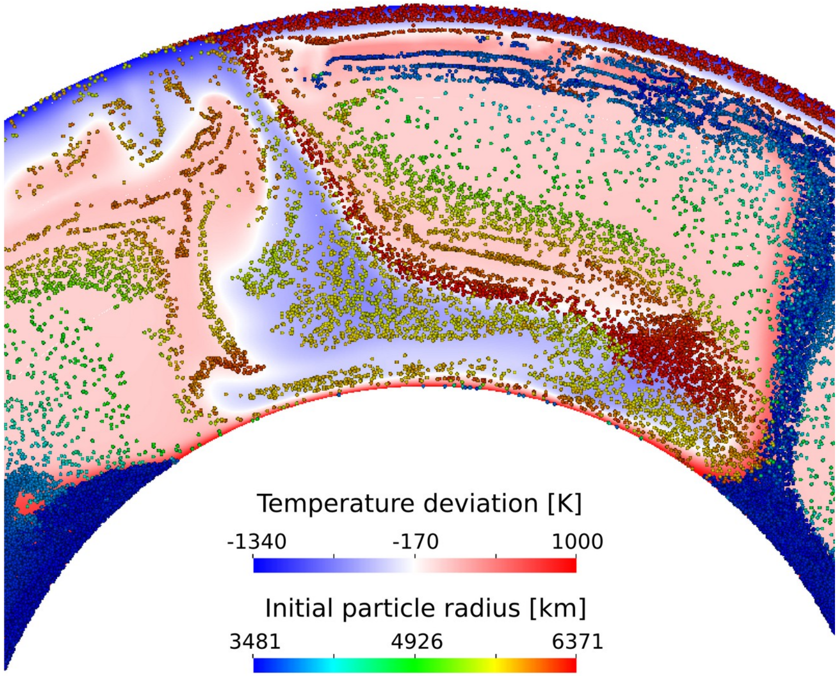


Particles

Rene Gassmoeller (~2013):

Fields are expensive. We want to track particles (and properties) as they move along with the flow.





Particles

Equations now:

$$-\nabla\cdot[2\eta\epsilon(u)]+\nabla p=g\rho(T)$$

$$\nabla\cdot u=0$$

$$\frac{\partial T}{\partial t}+u\cdot\nabla T-\kappa\Delta T=\gamma+\alpha\left(\frac{\partial p}{\partial t}+u\cdot\nabla p\right)+\eta(\nabla u)^2$$

$$\frac{\partial c_1}{\partial t}+u\cdot\nabla c_1=q_1(u,p,T,\vec{c})$$

$$\dots$$

$$\frac{\partial c_N}{\partial t}+u\cdot\nabla c_N=q_N(u,p,T,\vec{c})$$

$$\frac{\partial h}{\partial t}-A\Delta h=r(u,p)$$

$$\boxed{\frac{dx_i(t)}{dt}=u(x_i(t)) \qquad \frac{dp_{i,j}(t)}{dt}=s(u,p,T,\vec{c},\vec{p}_i)}$$

Particles

Considerations:

- Velocity affects particle locations
- Sometimes particle properties affect flow equations

- Computationally quite different from field-based methods
- Efficiency requires CFL<=1 → Particles transported at most one cell per time step
- Evaluation of rhs is *very* expensive

- Do one explicit Euler/RK2/RK4 step per (macro) step

Time discretization

Overall algorithm:

While ($t<t_{end}$):

- Solve Stokes equation
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1, implicit BDF2, operator splitting
- …
- Solve for compositional field 1, implicit BDF2, operator splitting
- Solve for surface deformation, coupled with external tool
- Advance particle positions and properties
- Advance time

Time discretization

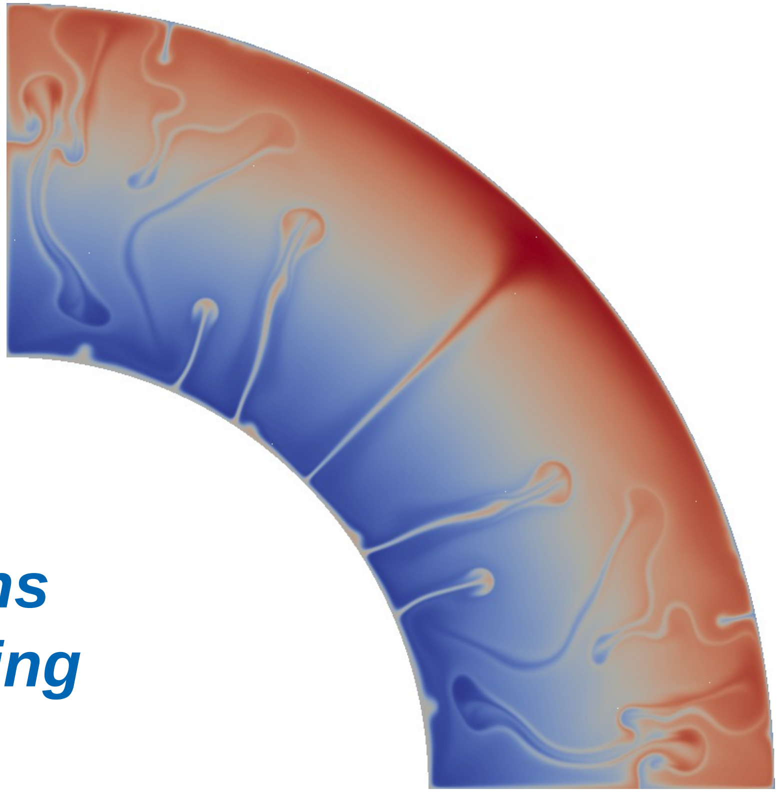
Overall algorithm:

While ($t<t_{end}$):

- Solve Stokes equation **60%**
- From velocity, compute time step
- Solve for temperature field with implicit BDF2
- Solve for compositional field 1, implicit BDF2, operator splitting
- … **15%**
- Solve for compositional field 1, implicit BDF2, operator splitting
- Solve for surface deformation **10%**
- Advance particle positions and properties **15%**
- Advance time

Conclusions of this part

ASPECT has turned out to be a very good tool to do interesting science!



- We can produce lots of colorful pictures → something must be right!

- ***But it is not an easy set of equations***
- ***We use a first-order operator splitting***
- ***How could we do this better?***

What I would love to do

My wishlist

Right now:

- First order operator splitting: Solve one physics after the other

What I wished:

- The system is a DAE. Solve it as such.
- Or at least use a (second-order) Strang splitting.

- At least do a study that assesses how much of an error we introduce – perhaps time discretization is not a major source of error?

My wishlist

Right now:

- Temperature + compositional fields used fixed-order BDF2
- BDF-2 (like all multistep methods) requires awkward start-up procedure

What I wished:

- Build on a library that abstracts the details (hard-coded coefficients, special-casing first time step)
- Adjusts method order automatically
- Allows playing with different methods

My wishlist

Right now:

- Particle integrator can use forward Euler ... RK4
- Time step is fixed fraction of global time step

What I wished:

- Build on a library that abstracts the details (hard-coded coefficients, special-casing first time step)
- Is able to adjust method order, time step length

- Should probably use SUNDIALS' ARKODE

My wishlist

Right now:

- Global time step is based on CFL condition, not error
- We have methods of iterating out the nonlinearity in each time step, but their use is not automatic

What I wished:

- Default to nonlinear iteration
- Control all this with an error criterion
- Let some library handle the details, rather than do it ourselves

Why I can’t do what I’d love to do

Obstacles

Human issues:

Complexity:

- Core of ASPECT has ~40,000 lines of code
- Core has contributions from ~25 people over 15 years

Modularity is an important consideration!

Obstacles

Historical issues:

- Core of ASPECT was written in ~2010
- SUNDIALS first released early 2000s, but not widely known in the PDE community
- PETSc TS paper is from ~2018

Building on others’ code is important – put your ideas into widely used packages!

Obstacles

Technical factors:

- Solving *coupled* system requires *much* more peak memory
- Coupling different codes in one time stepping scheme is *hard*!
- Discretized PDEs are *not* just large systems of ODEs:
  - Boundary conditions
  - Constraints (e.g., hanging nodes)
  - Meshes change from time step to time step
- High order operator splitting/multirate integrators not widely available (until recently?)

Again, put new ideas into widely used packages that know how to deal with these practicalities!

Obstacles

Needs assessment:

- It isn't actually clear that more accurate time stepping methods are necessary
- The only clear need is for larger time steps
  (Caveat: Accuracy, particles requires CFL ~ 1)

Things that do work already today

What does work

Packaging works:

- SUNDIALS, PETSc TS are good solutions
- We use the SUNDIALS KINSOL and PETSc SNES as nonlinear solvers elsewhere
- ASPECT uses SUNDIALS' ARKODE for stiff compositional right hand sides

What does work

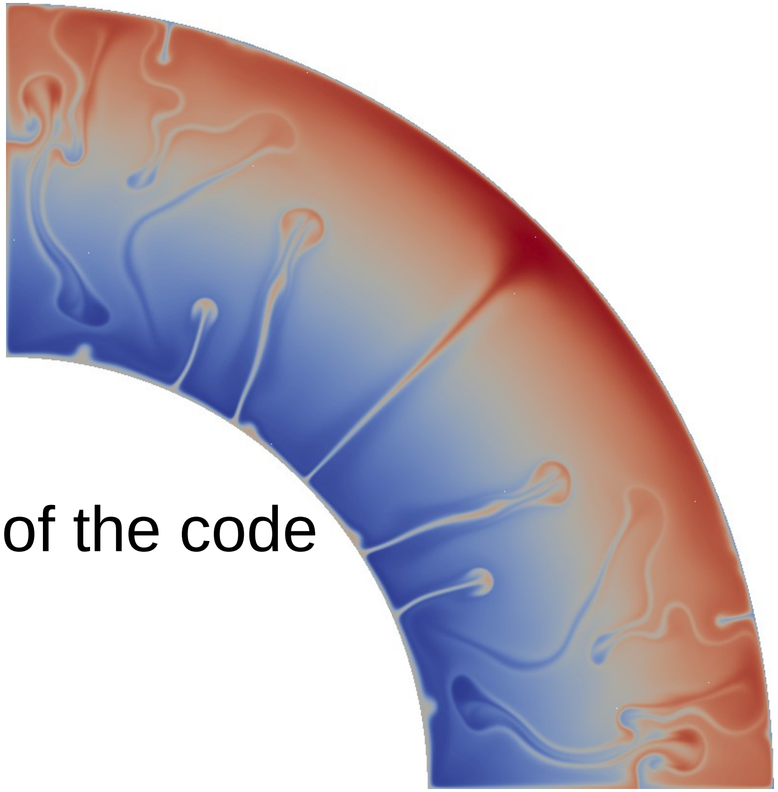
PDEs vs ODEs:

Packaged solvers (e.g. PETSc TS) understand that discretized PDEs are not just large ODE systems:

- Know how to deal with constrained DoFs
- Know how to deal with changing meshes
- Are used in large-scale multiphysics codes

Summary

Dealing with real world problems is hard:



- Problems are coupled, aren't always easy to characterize
- Software is written by humans:
  - Who are not experts in everything
  - With limited knowledge of the rest of the code
  - Over a long time
- The *need* for higher order methods is not always clear
- Packaging makes everyone's life easier!

Conclusions

Aspect – the Advanced Solver for Planetary Evolution, Convection, and Tectonics:

http://aspect.geodynamics.org/

References:

M. Kronbichler, T. Heister, W. Bangerth:
High accuracy mantle convection simulation through modern numerical methods.
Geophysics Journal International, 2012.

T. Heister, J. Dannberg, R. Gassmoeller, W. Bangerth:
High accuracy mantle convection simulation through modern numerical methods. II: Realistic models and problems
Geophysics Journal International, 2017.