The Easy way to Computational Geodynamics

Gabriele Morra

Department of Physics and School of Geosciences, University of Louisiana at Lafayette, United States

Companions in the new world of Big Data:

David A. Yuen
Department of Earth Sciences, University of Minnesota
School of Env. Studies, China Univ. of Geosciences, Wuhan

Sang-Mook Lee
School of Earth Sciences, Seoul National University, S Korea

Collaborators and Students at UL at Lafayette

Dr Raphael Gottardi
Prasanna Gunawardana (Syracuse Univ.)
Brian Fischer
Daniel Conlin (Now at Exxon Mobil)
Kyle Spezia (Now at Hulliburton)

Numerical Modeling and Big Data

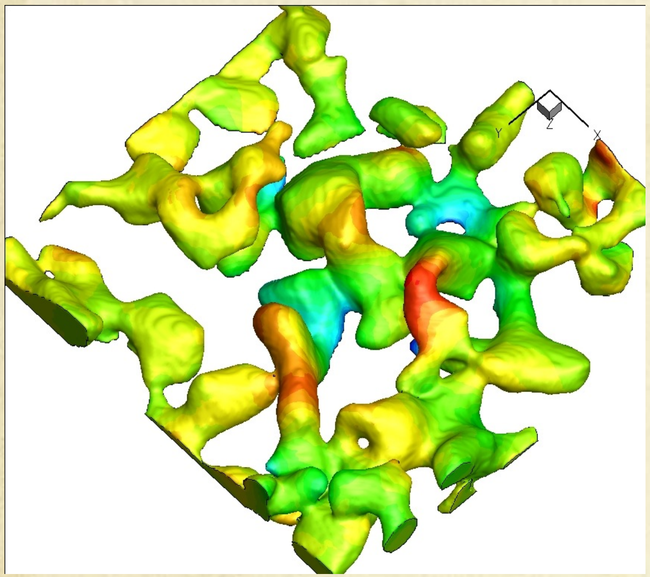
- Numerical Modellers model the physics with forward simulations. Key issues:
  - Model sizes increase. 3D models can output billions of data points per timesteps. Timesteps can be thousands.
  - Performance. <u>The computing challenge</u>. Scalability is the key.
  - Parallel programming is important. Solving PDE's requires fastest bandwidth available.
  - Pedagogy. How and when should students and professional learn numerical modeling?

Numerical Modeling and Big Data

- Numerical Modellers simulate the physics with forward simulations. Present solutions:
  - Model size. Selecting and compressing the output. Dimension reduction. I will present one example. Future machine learning. Can we use entropy to measure how much to record?
  - Performance. Hierarchical methods, like multigrid and multipole. I will show one example. New algorithms also welcome. Big problem.
  - Parallel programming. MPI is the main tool. GPUs and MICs also used. Heterogeneous parallel computing is the future.
  - Pedagogy. Python and its libraries: Numpy, Matplotlib, Cython, mpi4py, … **We don’t need Matlab anymore. Python is easier to learn and to use. New languages are emerging.**

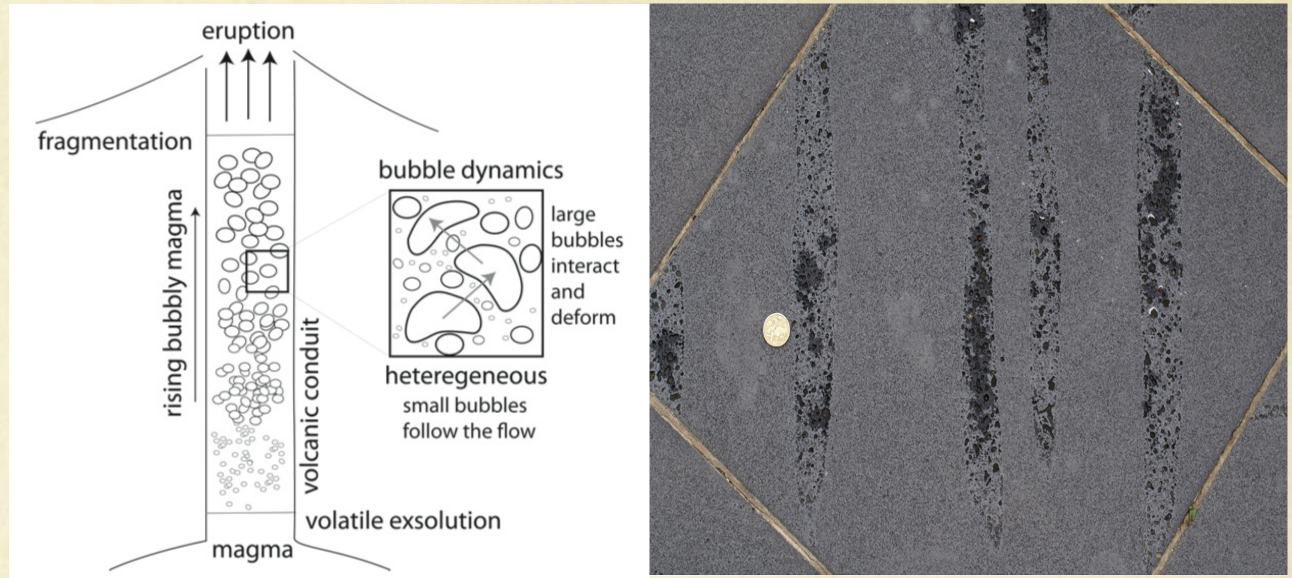
Geodynamic Motivation

Porous systems



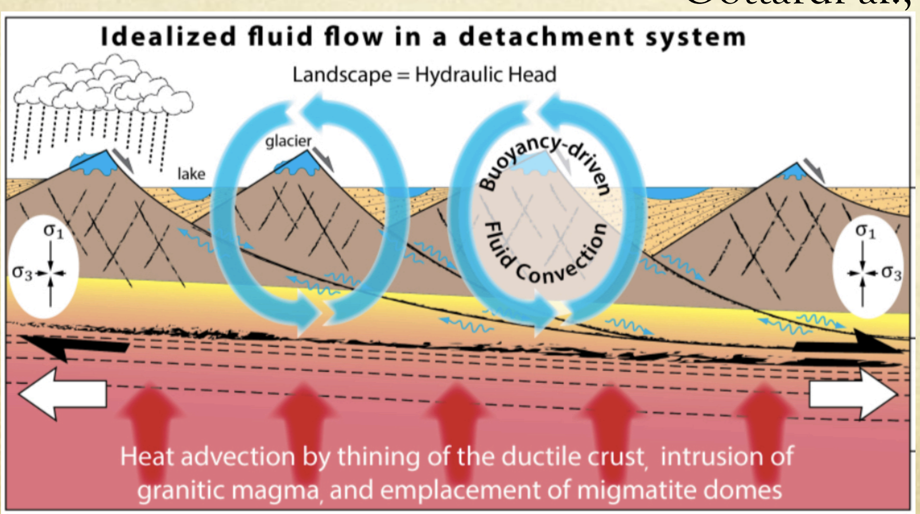
SNOW, data from the Snow Avalanche Institute, Davos

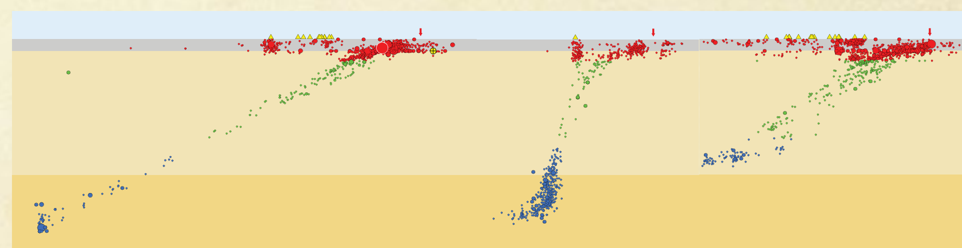
Two phase flow

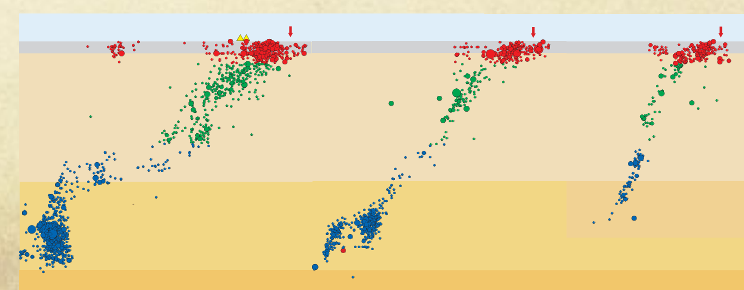


Morra et al., 2011

Gottardi al., 2011



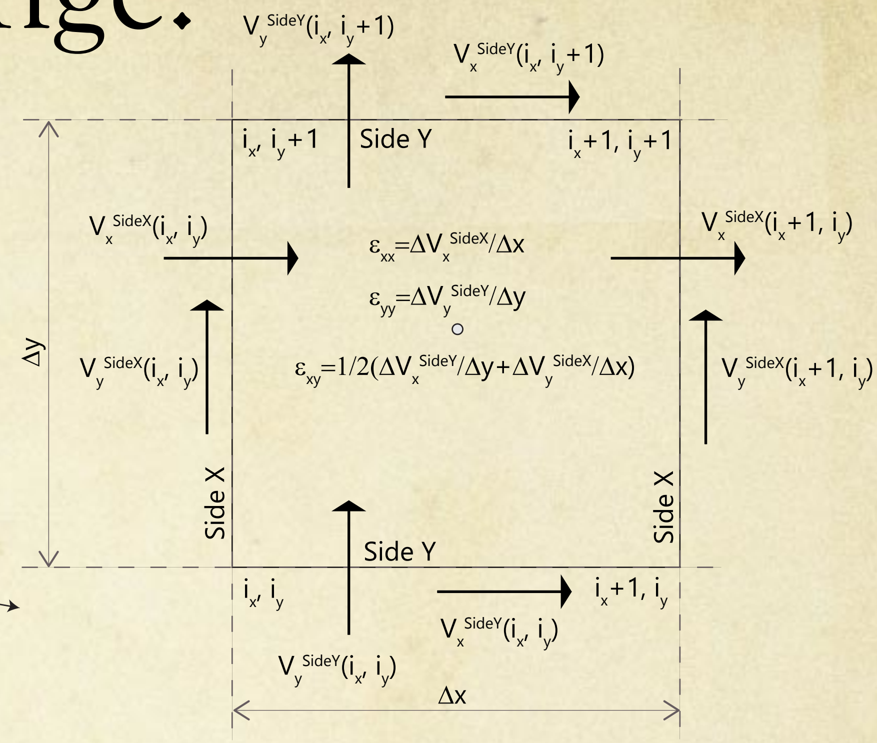




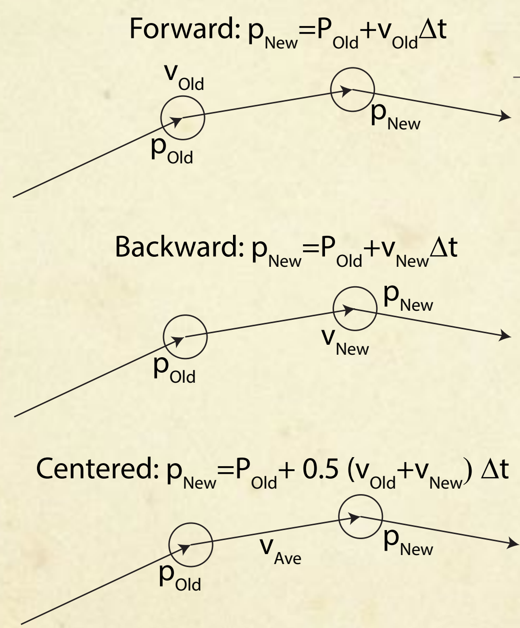
Deep seismicity

The computing challenge: particle in cell

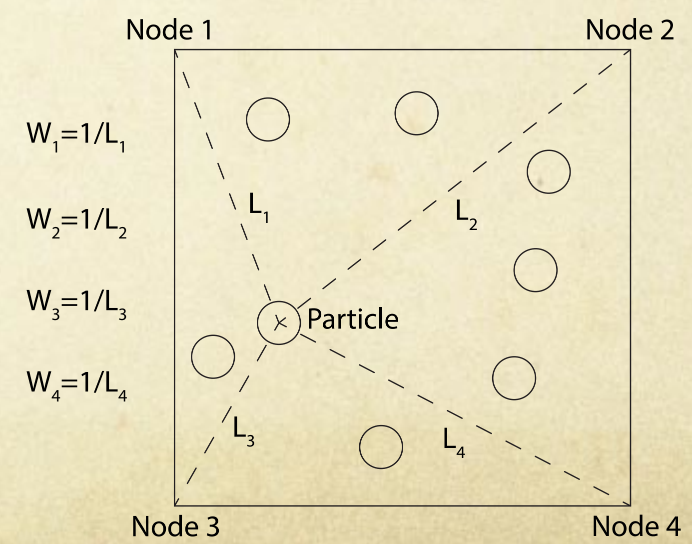
1. PDE's solution in a lattice



2. Particles advection



3. Projection of fields to and from the lattice



$F_{Particle} = (F_{N1} * W_1 + F_{N2} * W_2 + F_{N3} * W_3 + F_{N4} * W_4)/(W_1+W_2+W_3+W_4)$

The computing challenge: particle in cell

- Scalability

  1. Finite Differences and Finite Elements span maximum three orders of magnitude in space (10^9 cells = 1000^3).

  2. However particles allow increasing details. Particle advection can be immediately vectorized and do not weight on the overall computing time.

  3. Projection from and to lattice can be vectorized with a compact procedure.

- Implemented in Numerical Python. Easy to program.

The computing challenge: particle in cell

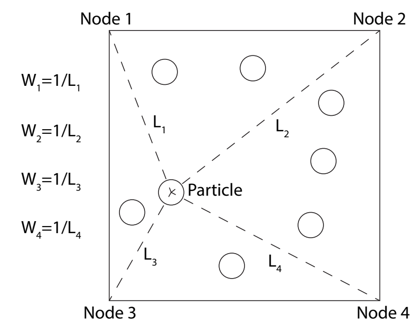
Implementation of the cell ←→ particle projections using NumPy.

```python
def projectLatticeToParticles(w1,w2,w3,w4,trIX,trIY,f,ft):
        ft[:]=(w1[:]*f[trIX[:],
                trIY[:]]+w2[:]*f[trIX[:]+1,
                trIY[:]]+w3[:]*f[trIX[:],
                trIY[:]+1]+w4[:]*f[trIX[:]+1,
                trIY[:]+1])/(w1[:]+w2[:]+w3[:]+w4[:])
        return (ft)
```

<u>One line.</u>
Vectorized.
Extremely fast.

```python
def projectParticlesToLattice(w1,w2,w3,w4,trIX,trIY,f,ft,tw):
        f[:,:]=0.0
        tw[:,:]=0.0
        f[trIX[:],trIY[:]]+=ft[:]*w1[:]
        f[trIX[:]+1,trIY[:]]+=ft[:]*w2[:]
        f[trIX[:],trIY[:]+1]+=ft[:]*w3[:]
        f[trIX[:]+1,trIY[:]+1]+=ft[:]*w4[:]
        tw[trIX[:],trIY[:]]+=w1[:]
        tw[trIX[:]+1,trIY[:]]+=w2[:]
        tw[trIX[:],trIY[:]+1]+=w3[:]
        tw[trIX[:]+1,trIY[:]+1]+=w4[:]
        f[:,:]/=tw[:,:]
        return (f)
```

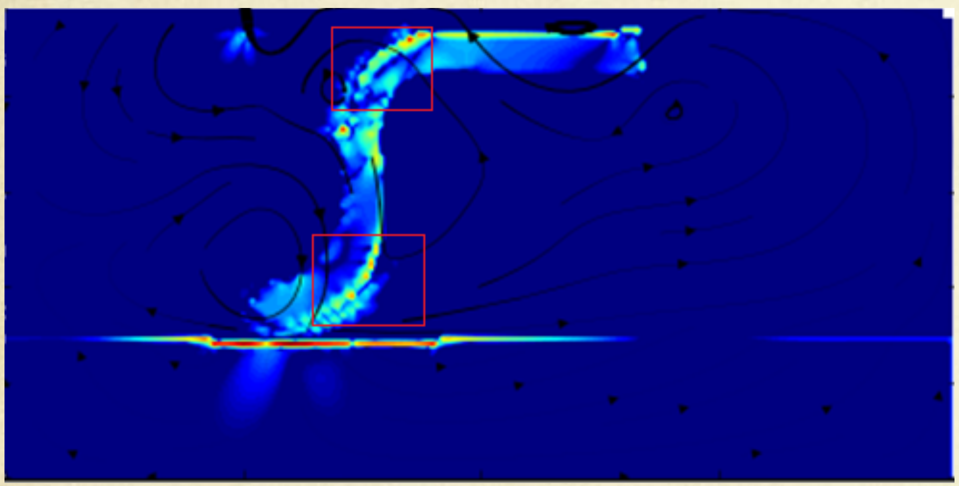
Compact.
Easy to understand and modify.
Minimum memory requirements



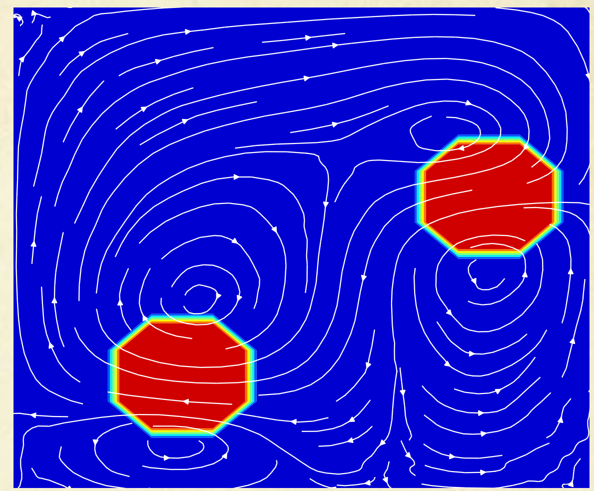
$$F_{Particle} = (F_{N1} * W_1 + F_{N2} * W_2 + F_{N3} * W_3 + F_{N4} * W_4)/(W_1+W_2+W_3+W_4)$$

The computing challenge: particle in cell

- Applications to Mantle flow (nonlinear Stokes), Porous media flow (Darcy equation) and suspension dynamics.



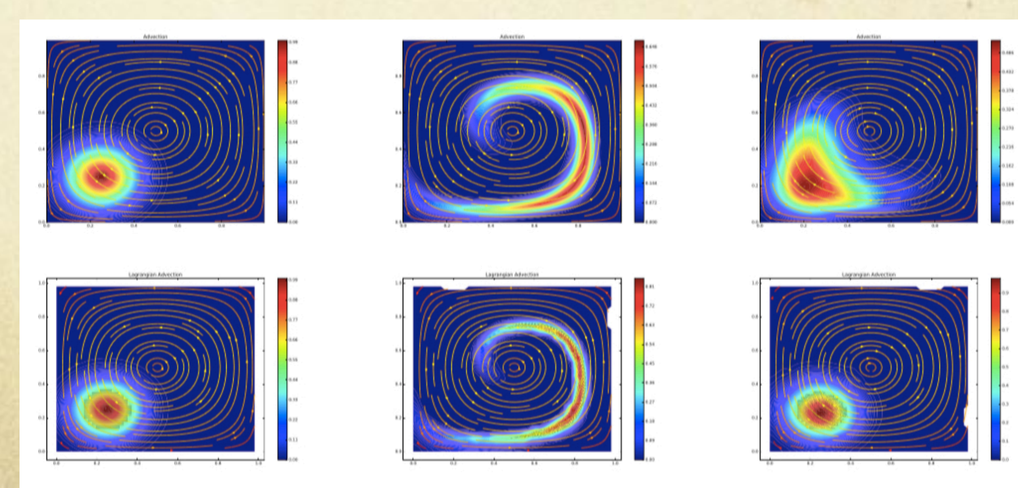
Gunawardana and Morra, submitted to Journal of Geodynamics



Suspended particles

G. Morra, from Springer Book, soon in press.

Vectorized Upwind scheme vs pure particles method



The computing challenge.

How is Numerical Python so fast?

1. Vectorization of most operations. NO LOOPS.

   In [27]: %timeit c=addArray(a,b) #*standard python*
   1 loops, best of 3: 639 ms per loop
   In [28]: %timeit c=a+b #*NumPy arrays broadcasting*
   100 loops, best of 3: 3.74 ms per loop

2. Cython (=C in Python) implementation of difficult routines.

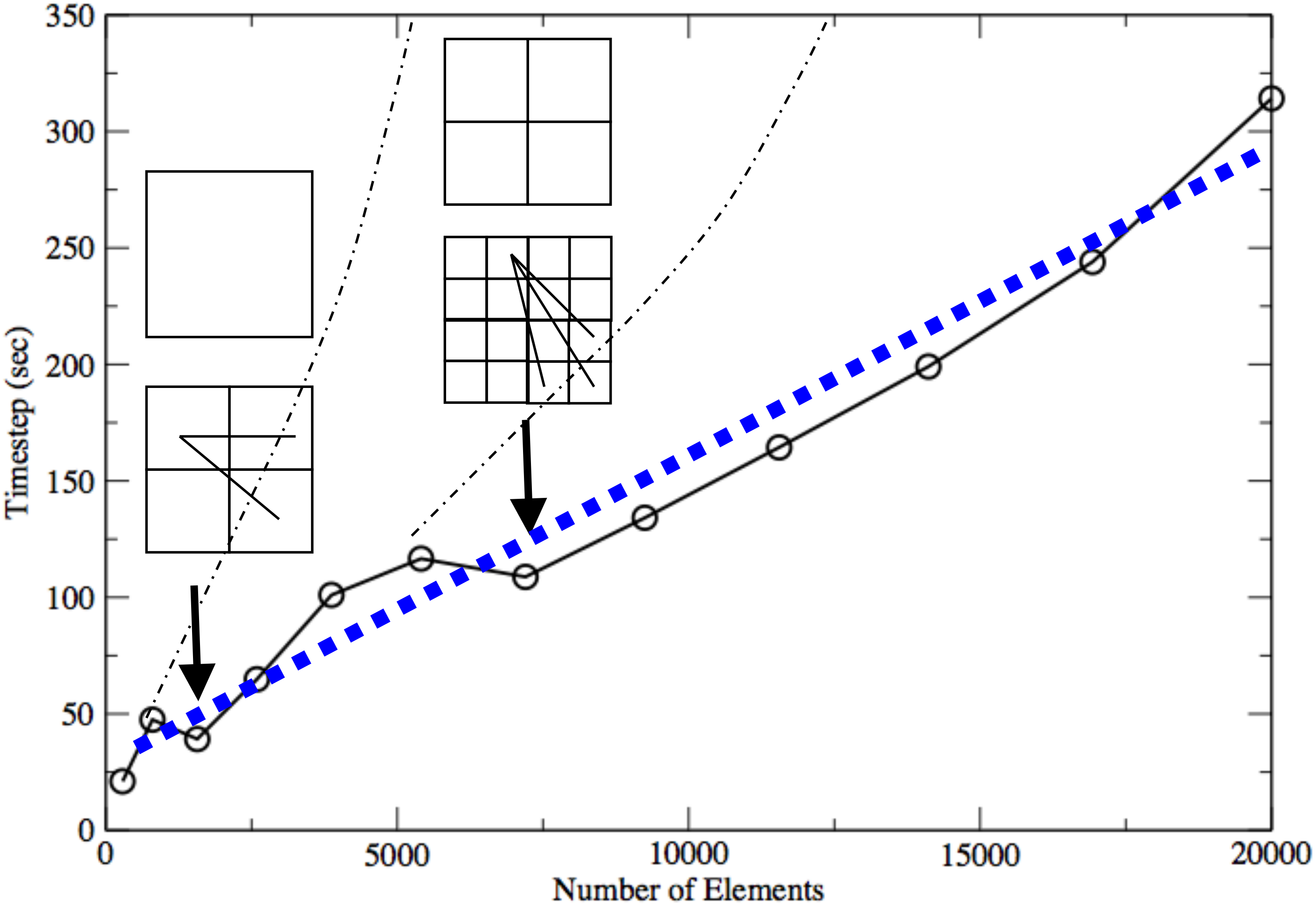
3. Lower understanding of the machine operations.

4. Many extension libraries (mpi4py, pyCuda, petsc4py).

Considerations on 3D modeling

N=L/RES	Number Of Elements	Solution Approach	Earth L= 10^4km RES:10km
Finite Element	Volume Cells $O(N^3)$	Sparse Matrix Multigrid Inversion time $O(N^3)$	N= 10^3 CPU time $O(10^9)$
Boundary Element	Surface Panels $O(N^2)$	Dense Matrix Inversion time $O(N^2 * N^2)$	N= 10^3 CPU time $O(10^{12})$

CPU time required for each timestep



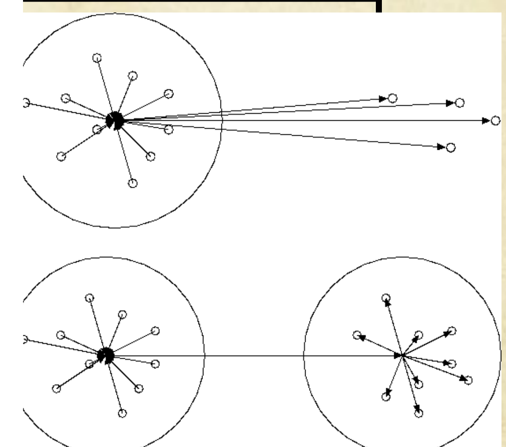
Furthermore it scales linearly on an MPI environment!

BEM

arth

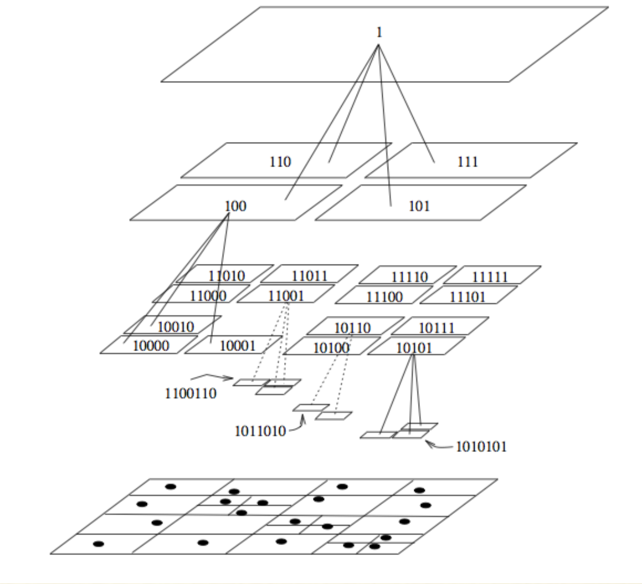
= 10^4km

ES:10km

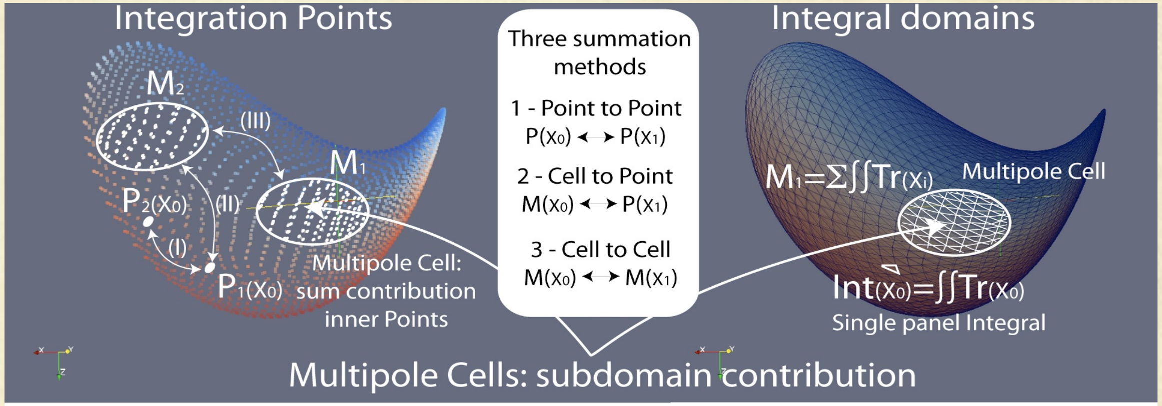


Boundary Element	Surface Panels $O(N^2)$	Multipole Inversion time $O(2N^2 \log N)$	N= 10^3 CPU time $O(10^7)$ p.s.

Immediate 3d modeling with NumPy



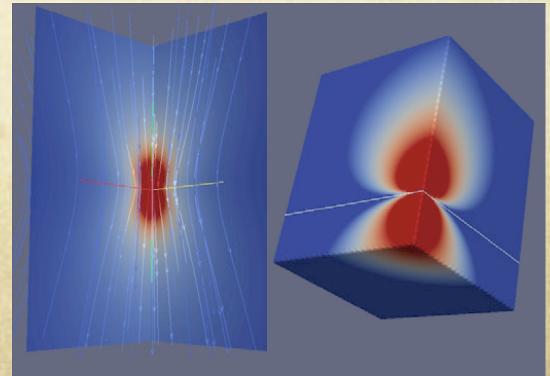
1. Tree representation



2. Fast Integration

3. Lagrangian motion

4. Cartesian representation



Immediate 3d modeling with NumPy

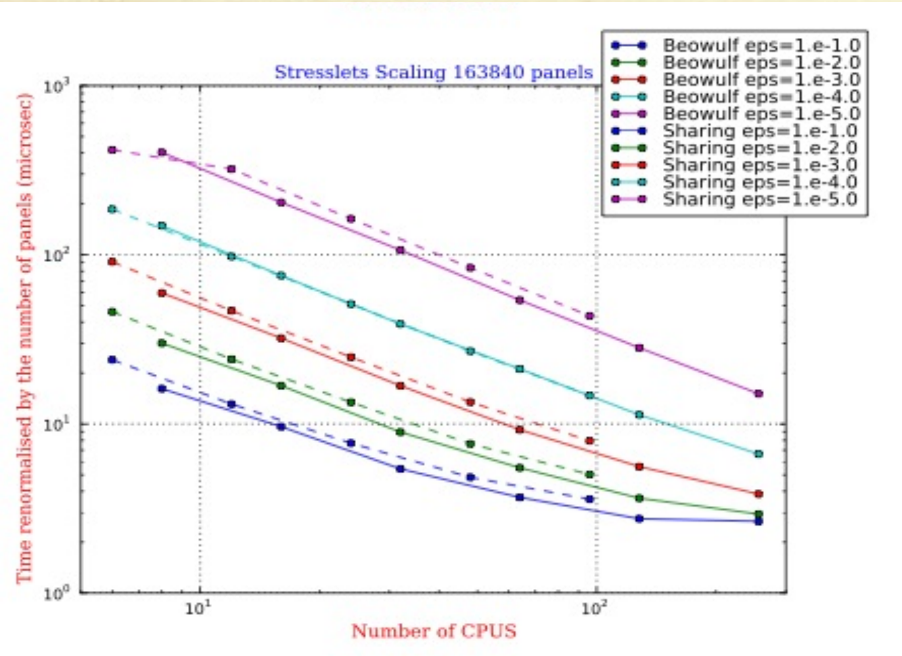
1. Tree representation:

    from scipy import spatial
    x, y, z = np.mgrid[0:5, 2:8, 3:7]
    tree = spatial.KDTree(zip(x.ravel(), y.ravel(), z.ravel()))

2. Many-body calculations enable N-logN scaling.

3. Fast Integrals with NumPy

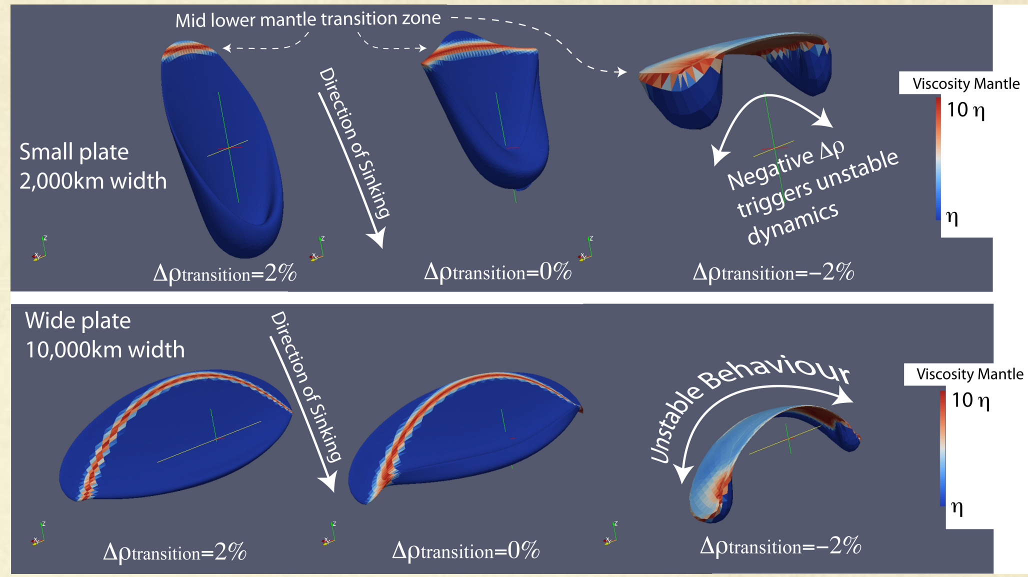
4. MPI Parallelization

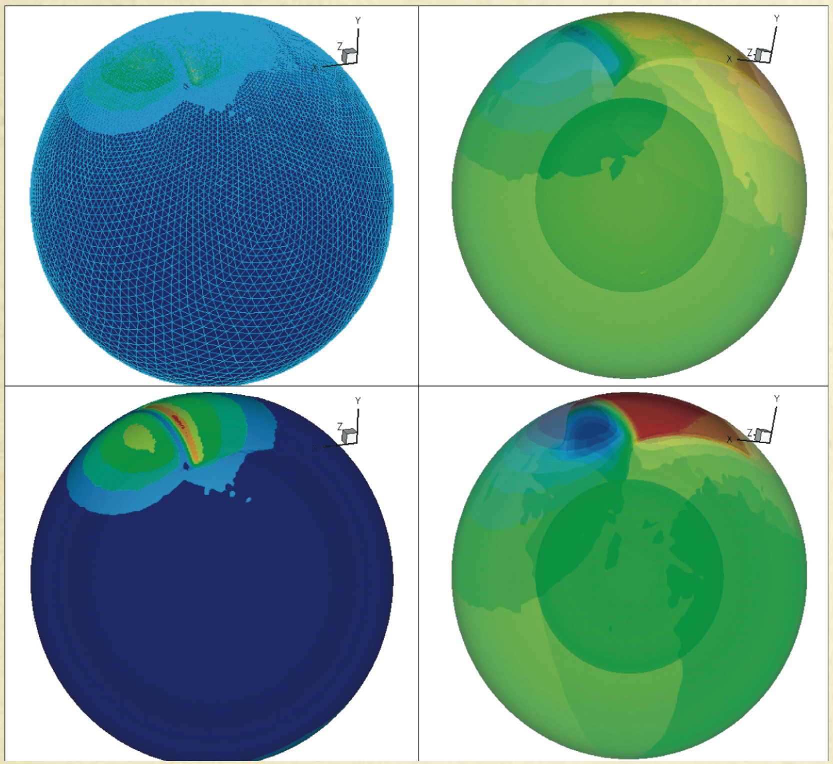



```python
def integralOverTriangle(collocationPosition, element,
    gaussPoints,gaussWeights,xiCoord,etaCoord,
    coords,nodesOnElement, alpha,beta,gamma):

# Integrates the Green's function over a non-singular triangle
# GE[0:3,0:3] are the integrated component
# gaussPoints is the order of triangle quadrature

   GE=np.zeros((3,3),float)

    (gaussPosition,
    gaussNormalVector,
    gaussSurfaceMetrics,
    gaussDerivatives)=interpolate(
       coords[node1],coords[node2],coords[node3],
       coords[node4],coords[node5],coords[node6],
       alpha,beta,gamma,
       gaussXi,gaussEta,
       1)

    d = gaussPosition-collocationPosition
    dd = np.outer(d,d)
    i = np.identity(3)
    r = np.linalg.norm(d)
    velocityGreenFunction = i / r + dd / r**3

    prefactor=0.5*gaussSurfaceMetrics*gaussWeights[gaussPoint]
    GE += prefactor*velocityGreenFunction

    return GE
```

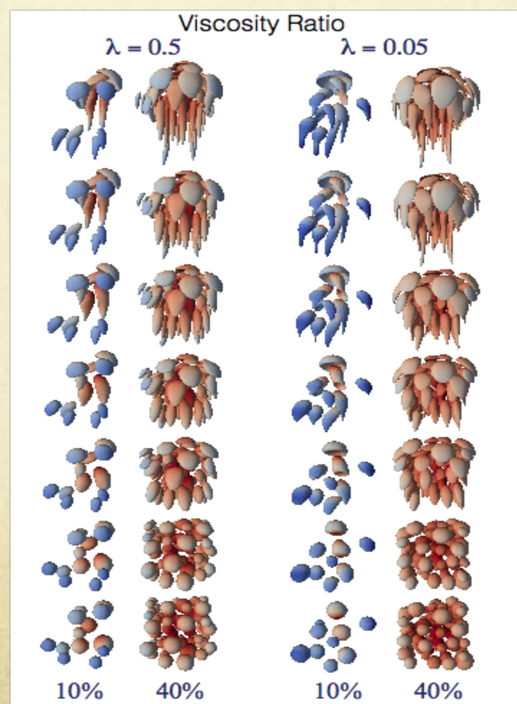
Applications in global geodynamics and multiphase flow

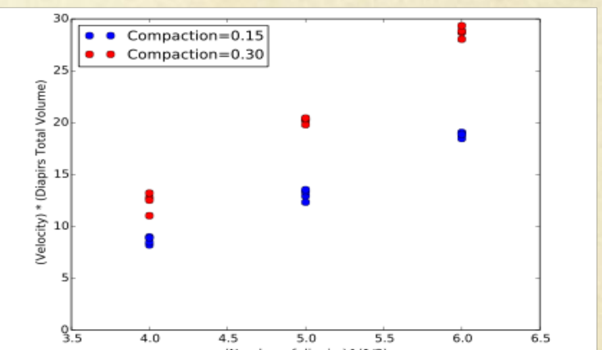
Morra et al., PEPI, 2010



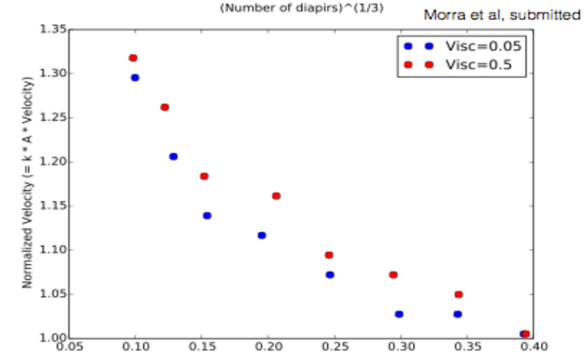


Morra et al., 2012





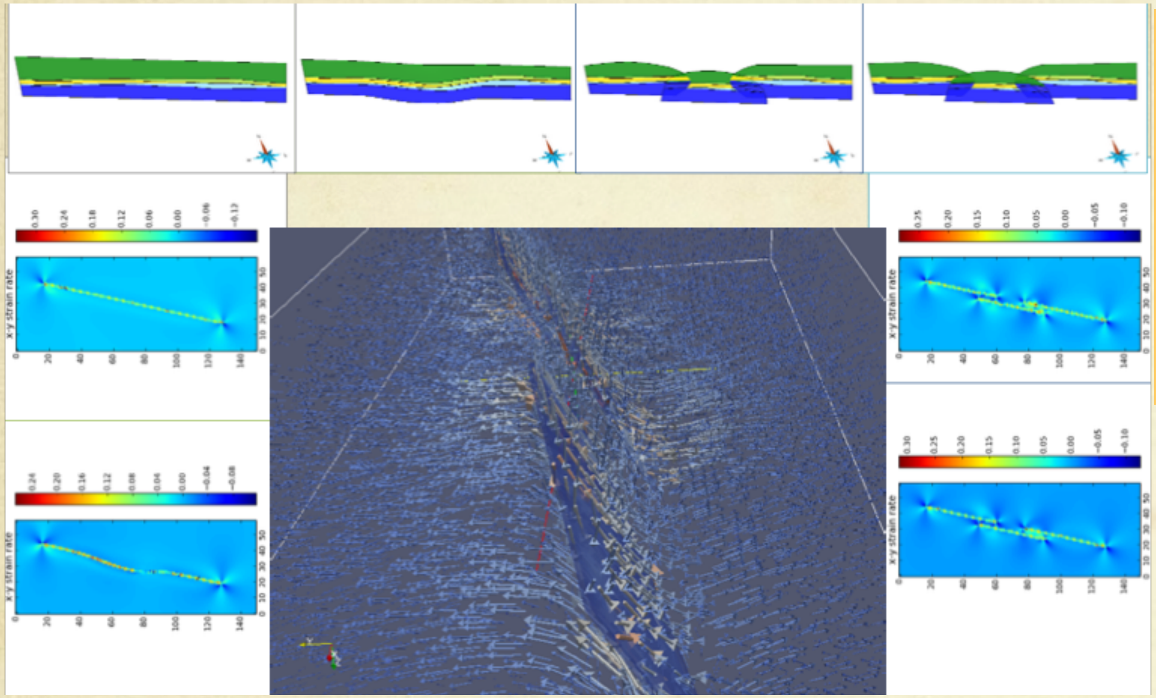
Morra et al, submitted

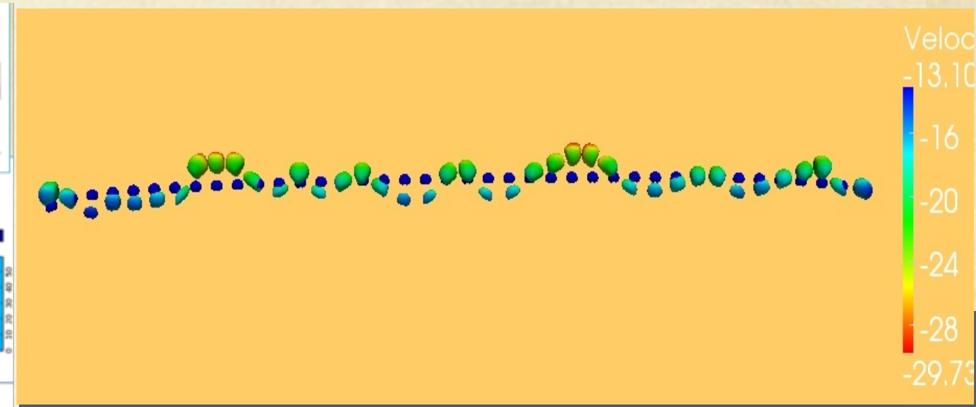


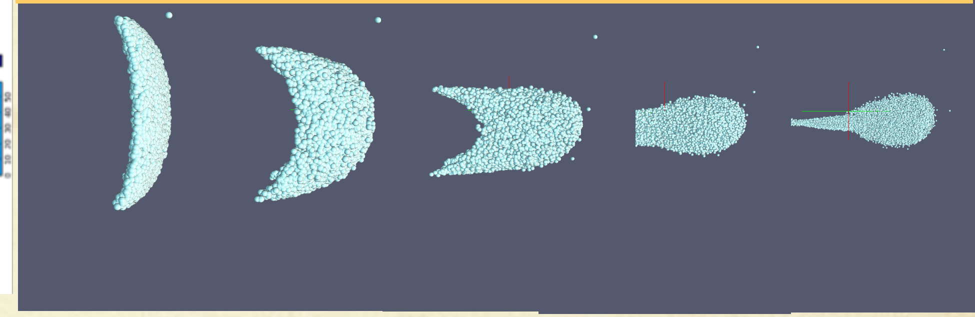
Morra et al., 2015

Fast computing allows large scale models

Crustal Dynamics







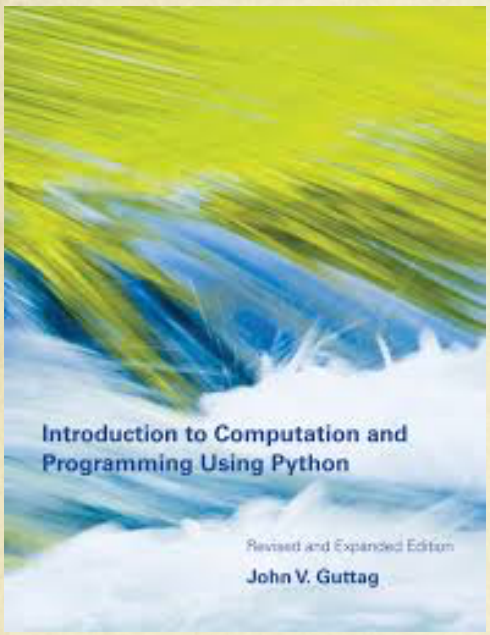
Every bubble made by 1000 triangles!

Homogeneous Long wave Instability

Morra et al., 2015

Learning Fast Computing

A new generation of programming languages is emerging and replacing C, C++ and Fortran. Python is the most used, but other options have emerged emerge such as Julia and Ruby, or Java based Scala and Hadoop. Presently Python is the easiest, most compact and powerful and is replacing the glorious but not free and not open Matlab.



Some universities offer a mandatory "Introduction to Computer Science and Programming" at the beginning of every science program. Future geoscientists will use computing for every task. The earlier they familiarize with how computers "think", the sharper they will use their computing tools.

Many tools for machine learning are now mainly interfaced with Python.
For example TensorFlow, from Google, that is open source and free to use, and allows organizing visual data/model output.

TensorFlow

Conclusions and Perspectives

- Students and professionals have now more accessible tools to learn programming, which are simple and accessible new languages.

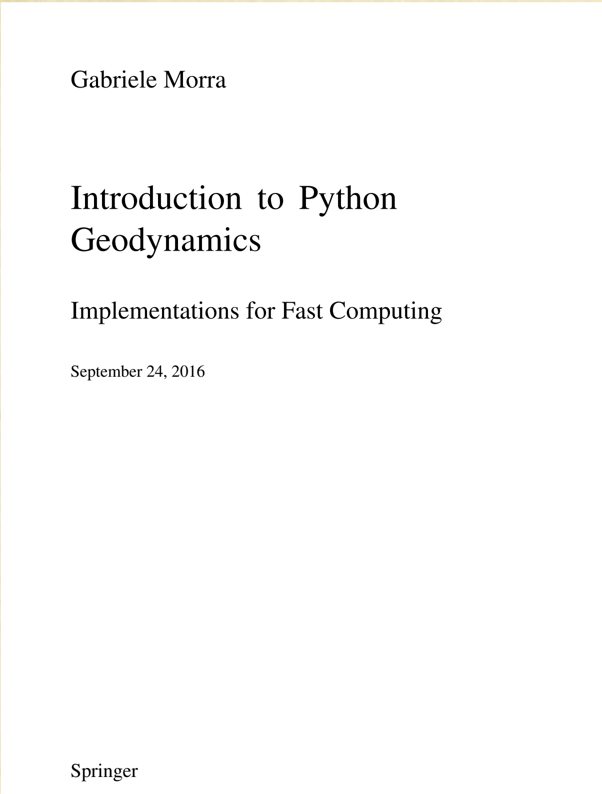
- Also hybrid approaches such as PARTICLES IN CELL and FAST MULTIPOLE -- BOUNDARY ELEMENTS can be implemented without great overhead because can be completely vectorized.

- For example by using Python geodynamics codes, in 2D as well as in 3D, are compact, run fast, are parallelized in a straightforward way.

- Most open projects are now interfaced, and sometimes directly developed, in Python and similar.

- To use Numerical Python and associated libraries is presently the EASY WAY TO learn COMPUTATIONAL GEODYNAMICS.

New initiatives


Gabriele Morra

Introduction to Python Geodynamics

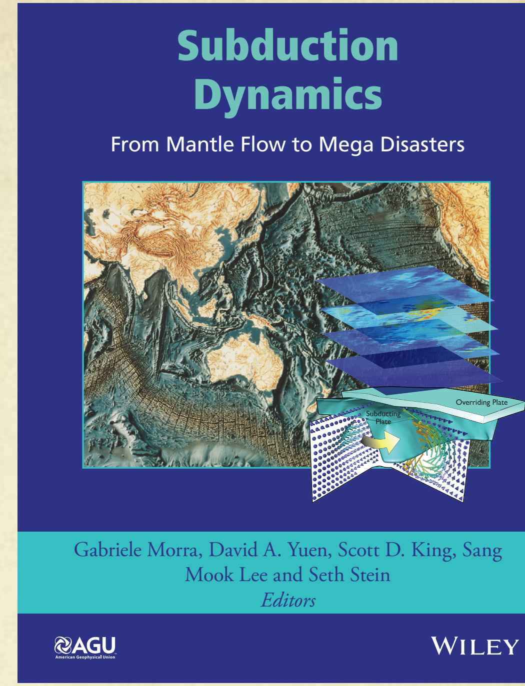
Implementations for Fast Computing

September 24, 2016

Springer

An introductory level book on Geodynamics with Python, specifically for undergraduate students. Lecture Notes in Earth Sciences Springer Verlag.

Big Data Training and International Conference on Haikou, Hainan Island, South China Sea. January 4 to 11, 2017
http://mcdata-consult.com



Special Volume with numerical techniques on geodynamics.