Escript: Open Source Environment For Solving Large-Scale Geophysical Joint Inversion Problems in Python

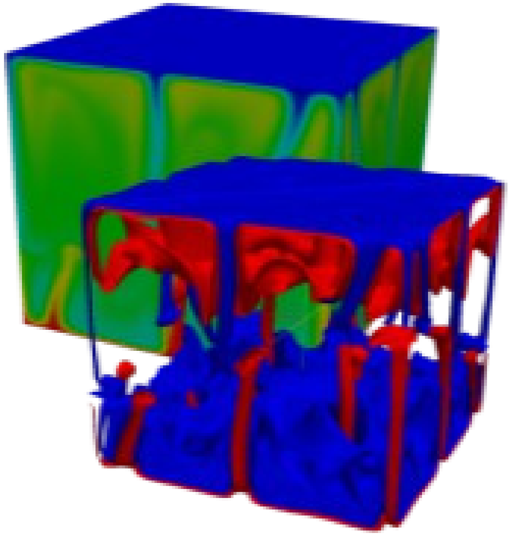
Lutz Gross, Cihan Altinay, Joel Fenwick, Troy Smith
School of Earth Sciences
The University of Queensland
St Lucia, Australia

With contributions from
C. Kemp, Geoscience Australia
Simon Shaw UQ

May 2014

escript

- Environment for mathematical modelling with partial differential equations (PDEs)
  - Multi-physics modelling
  - python interface
  - uses PDE terminology
  - 2D+3D FEM
  - MPI+OpenMP



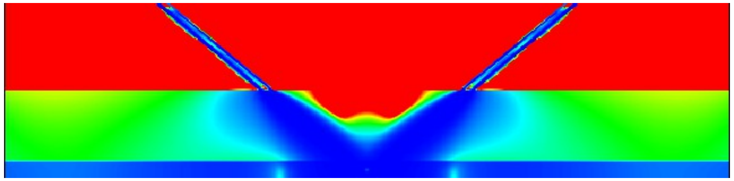
see https://launchpad.net/escript-finley

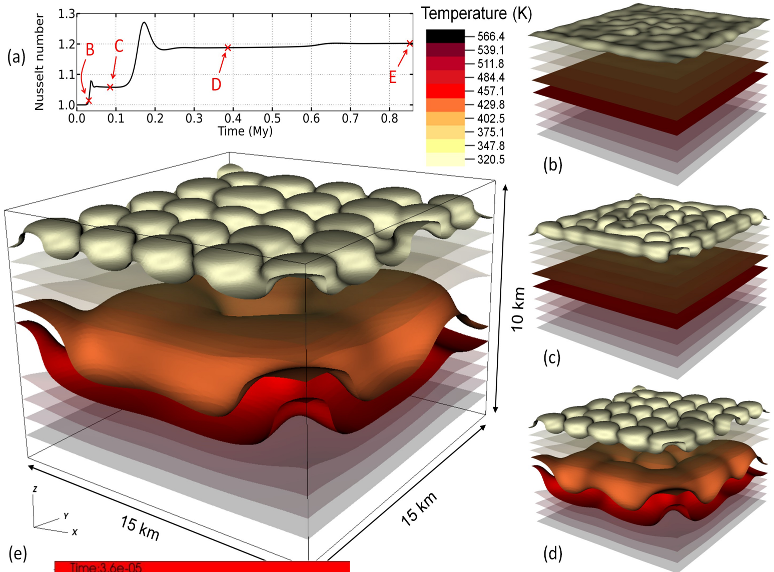
Escript status

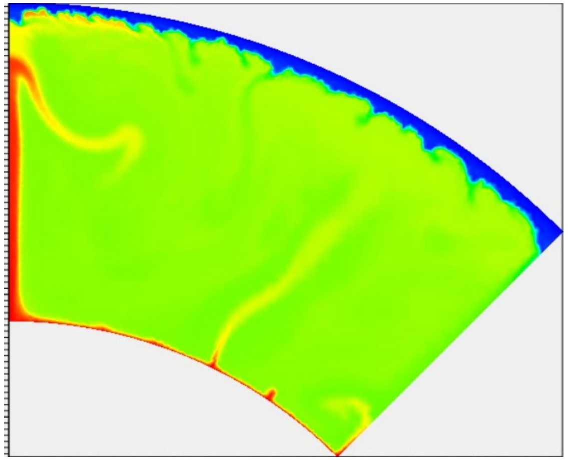
- Since 2003 with about two software engineers at any time
  - AuScope NCRIS, CRIS, NCRIS2
  - AGOS EIF
  - Total Funding $A ~ 2.6 Million (until July'15)
- Open Source under OSF 3.0
- Using best best practice
  - SVN
  - Nightly testing
    - >60000 functional tests
    - On ~10 platforms
  - Source and binary distribution via https://launchpad.net/escript-finley
  - Bug tracking

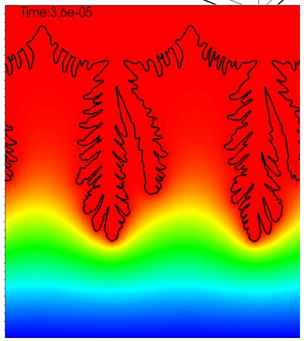
escript applications

- Geophysical inversion
- Mantel Convection
- Pores Media Flow
- Earthquakes
- Reactive Transport
- Volcanoes
- Tsunamis









escript.LinearPDE

Interface to a general linear PDE in variational form:

$$\text{find } u_k : \Omega \rightarrow \mathbb{R},\ u_k = r_k \text{ where } q_k > 0$$

$$\text{for all } v_i : \Omega \rightarrow \mathbb{R},\ v_i = 0 \text{ where } q_i > 0$$

$$\int_\Omega A_{ijkl} v_{i,j} u_{k,l} + B_{ijk} v_{i,j} u_k + C_{ikl} v_i u_{k,l} + D_{ik} v_i u_k \, dx$$

$$= \int_\Omega X_{ij} v_{i,j} + Y_i v_i \, dx$$

with PDE coefficients A_{ijkl}, B_{ijk}, C_{ikl}, D_{ik}, X_{ij}, Y_i, q_k, r_k

Example

Gravitational field anomaly g_i due to density variation ρ

$$-u_{,ii}=4\pi\cdot\rho \qquad -\Delta u=4\pi\cdot\rho$$

$$g_i=-u_{,i} \qquad \boldsymbol{g}=-\nabla u$$

Example (cont.)

Variational Formulation of the PDE:

$$\text{for all } v:\Omega \rightarrow \mathbb{R}$$
$$\int_{\Omega} \underbrace{\delta_{jl}}_{A_{jl}} v_{,j} u_{,l} \, dx = \int_{\Omega} \underbrace{4\pi \cdot \rho}_{Y} \cdot v \, dx$$

$$\text{for all } v:\Omega \rightarrow \mathbb{R}$$
$$\int_{\Omega} \nabla^t v \nabla u \, dx = \int_{\Omega} 4\pi \cdot \rho \cdot v \, dx$$

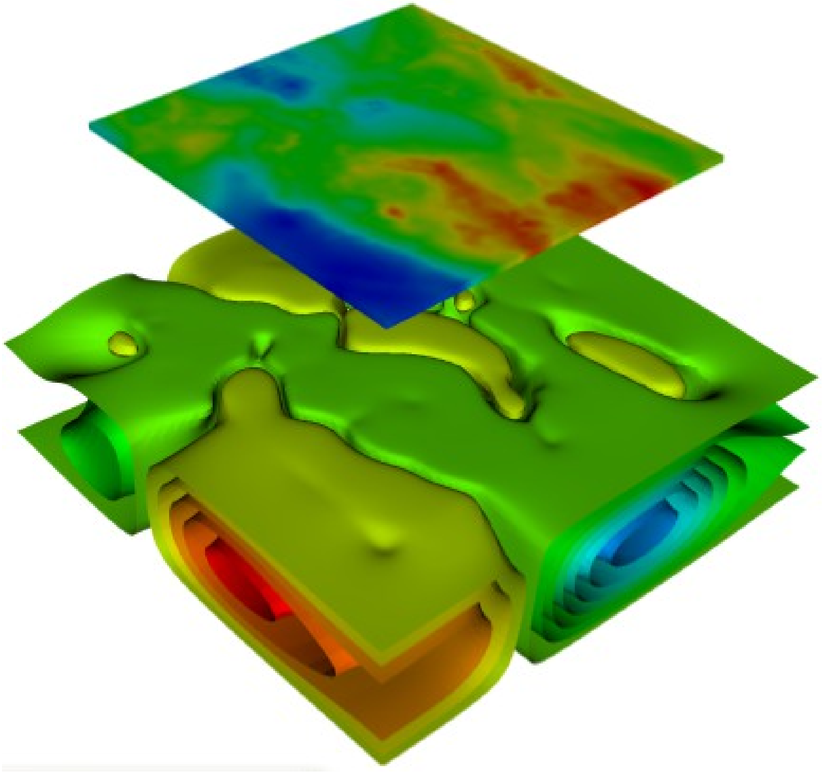
Implementation in python

```
from esys.escript import *
import esys.escript.unitSI as U
rho=200*U.kg/U.m**3
# 100x100x10km with 200m resolution
domain=Brick(500,500,50,100*U.km,100*U.km,10*U.
# solve the PDE:
myPDE=LinearPDE(domain)
z=domain.getX()[2] # get z coordinates
mypde.setValue(A=kronecker(3), Y=4*pi*rho, \
               q=whereZero(z-sup(z)) # fix potential @ top
u=myPDE.getSolution()
g=-grad(u)
# calculate misfit to vertical gravity data
d=getMyGravityData(...)
D=0.5*integrate((g[2]-d)**2)
```

- constant
- tagging
- PDE solution

Inversion

Recover physical properties in the subsurface from measurements near/above surface.



PDE constraint Optimisation

Find $\underset{\rho}{\arg\min}\, J(\rho) := D(u) + \mu \cdot R(\rho)$

data misfit : $D(u) = \frac{1}{2} \int_{\Omega} (u_{,2} - \hat{g})^2 \, dx$

PDE constraint : $u = Q[\rho]$ defined by
for all $v : \int_{\Omega} v_{,l} u_{,l} \, dx = \int_{\Omega} 4\pi \cdot \rho \cdot v \, dx$

regularization: $R(\rho) = \frac{1}{2} \int_{\Omega} \rho_{,l} \rho_{,l} dx$

escript.downunder

find property functions : $\boldsymbol{m} = (\text{density}, \text{susceptibility}, \ldots)$

cost function $J(\boldsymbol{m}) := R(\boldsymbol{m}) + \sum_f D^f(u^f)$

data misfit for forward model f : $D^f(u^f) = \frac{1}{2} \int_\Omega \left(w_i^f \cdot u_{,i}^f - \hat{g}^f \right)^2$

PDE constraints : $u^f = Q^f[\boldsymbol{m}]$

Regularization Term

$$R(\boldsymbol{m}) = \sum_{k} \int_{\Omega} \mu_k S(m_k) + \sum_{i<k} \int_{\Omega} \mu_{ik} C(m_k, m_i)$$

$$\text{smoothing}: S(m) = \frac{1}{2}\left(w_1 \cdot m_{,l} m_{,l} + w_0 \cdot m^2\right)$$

$$\text{cross-gradient}: C(m,n) = \frac{1}{2}\left(\left(m_{,l} m_{,l}\right) \cdot \left(n_{,l} n_{,l}\right) - \left(m_{,l} \cdot n_{,l}\right)^2\right)$$

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

May 2014

Gradient

find $\boldsymbol{m} : \langle \nabla J(\boldsymbol{m}), \boldsymbol{p} \rangle = 0$ for all increments $\boldsymbol{p}$ to $\boldsymbol{m}$

$$\langle \nabla J(\boldsymbol{m}), \boldsymbol{p} \rangle = \int_{\Omega} X_{ij}\, p_{i,j} + Y_i\, p_i \; dx$$

- with suitable functions X_{ij} and Y_i of $\boldsymbol{m}$
- calculate via superposition
- extra adjoint PDE per forward problem

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

May 2014

Quasi-Newton Method

iteration count $\nu=0$; initial guess $\boldsymbol{m}^{(\mathbf{0})}$

$$\boldsymbol{g}^{(\nu)}=\nabla J\left(\boldsymbol{m}^{(\nu)}\right)$$

find $\boldsymbol{p}^{(\nu)}$ with: for all $\boldsymbol{q}:\left\langle \mathrm{H}^{(\nu)}\boldsymbol{p}^{(\nu)},\boldsymbol{q}\right\rangle=\left\langle \boldsymbol{g}^{(\nu)},\boldsymbol{q}\right\rangle$

$\mathrm{H}^{(\nu)}\approx$ Hessian Operator at $\boldsymbol{m}^{(\nu)}$

$m^{(\nu+1)}=m^{(\nu)}+\alpha\cdot p^{(\nu)}$ with

$\min_{\alpha} J\left(m^{(\nu)}+\alpha\cdot p^{(\nu)}\right)$ via line search

$$\nu\leftarrow\nu+1$$

Hessian Operator

- Ignore forward models
- If no cross gradient term

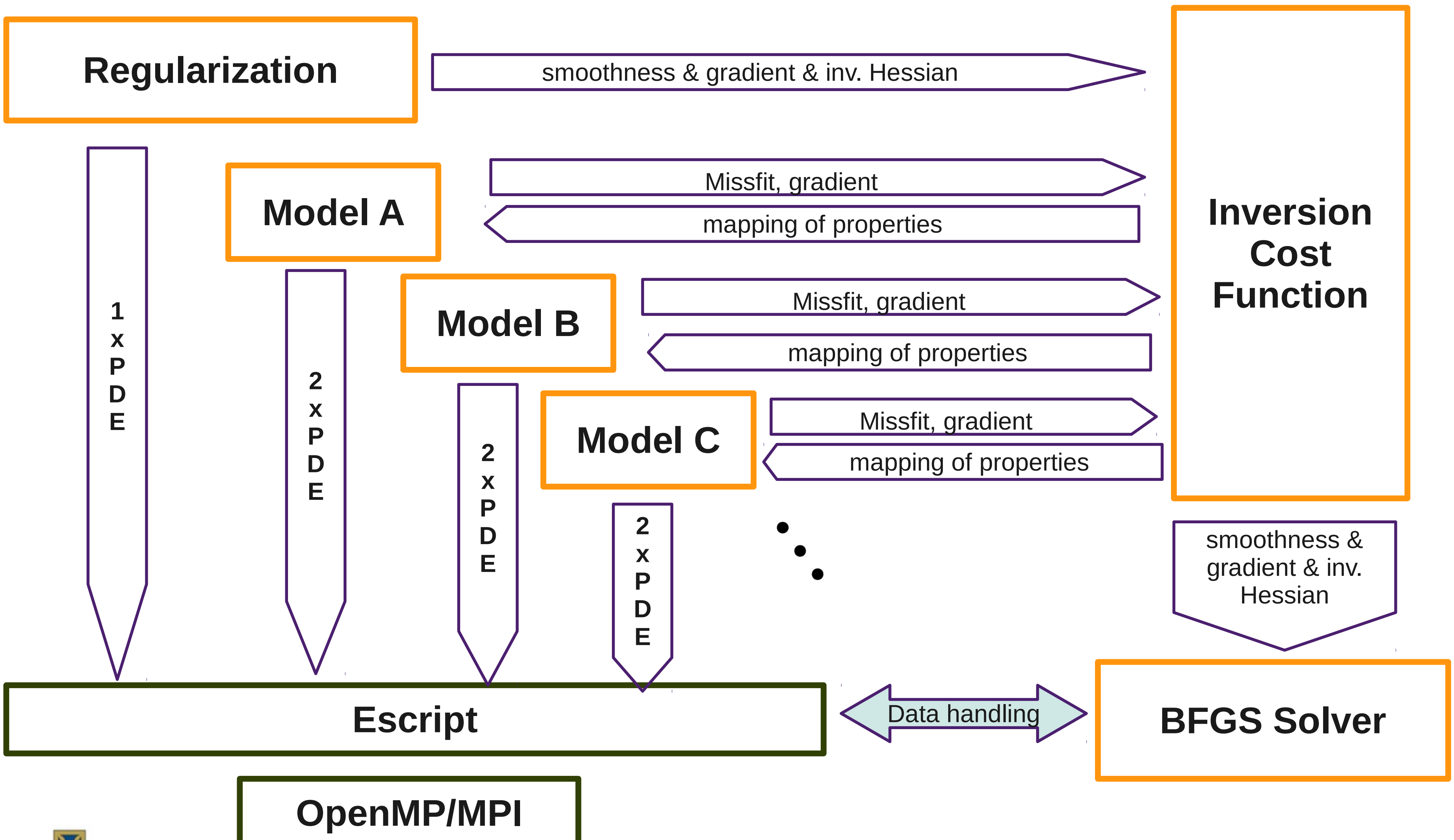
$$\left\langle \mathrm{H}^{(\nu)} \boldsymbol{p}^{(\nu)}, \boldsymbol{q} \right\rangle = \left\langle \boldsymbol{g}^{(\nu)}, \boldsymbol{q} \right\rangle$$

$$\downarrow$$

$$\int_{\Omega} w_{1\mathrm{k}}\, p_{k,l}\, q_{k,l} + w_{0\mathrm{k}} \cdot p_k q_k \; dx = \int_{\Omega} X_{ij}\, q_{i,j} + Y_i q_i \; dx$$

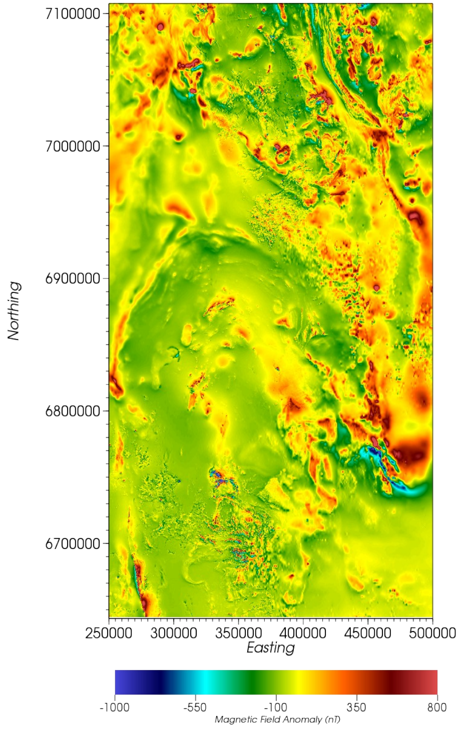
→ PDE solved by escript

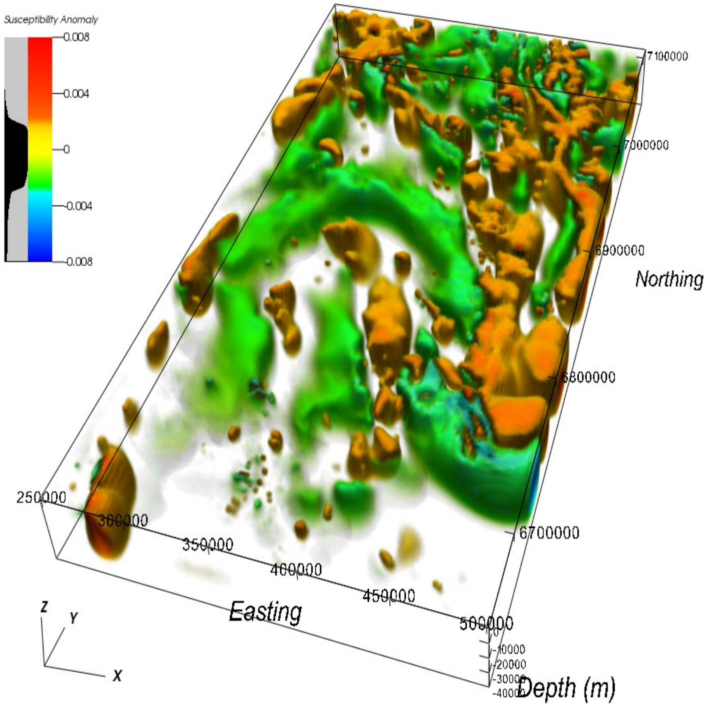
Inversion Framework in Python



Example: Clarence-Moreton Basin

- Magnetic data from vgl.auscope.org.au, resolution <100m
- 500Million cells on 8000 cores in < 1.5 hours





May 2014

Summary

- Inversion as PDE constraint optimization
  - "First optimize then discretize"
  - Adaptive meshing
  - Computational scalability
- Escript based inversion framework
- Visit https://launchpad.net/escript-finley