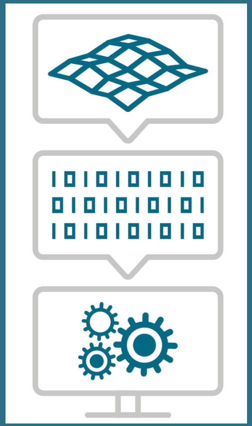
Australian Government
Geoscience Australia

Exploring for the Future



Julia based geophysical optimization and Bayesian inference

Anandaroop Ray

With grateful thanks to

Ross C. Brodie, Richard Taylor, Yusen Ley-Cooper, Neil Symington, Andrew McPherson, Karol Czarnota, Kerry Key, Thomas Bodin, Jan Dettmer, Steve Constable, Catherine Constable, Brent Wheelock, Sam Kaplan, John Washbourne, Uwe Albertin, Daniel Blatter, Negin Moghaddam, Malcolm Sambridge, …

High Quality Geophysical Analysis: HiQGA.jl

- Can do a variety of modeling, inversion and inference:
- AEM, SMRI, MT, CSEM, image regression, and Gauss-Newton/Occam inversion
- Can also do generic joint inversion, e.g., MT and AEM
- Open source, very flexible MIT license

https://github.com/GeoscienceAustralia/HiQGA.jl

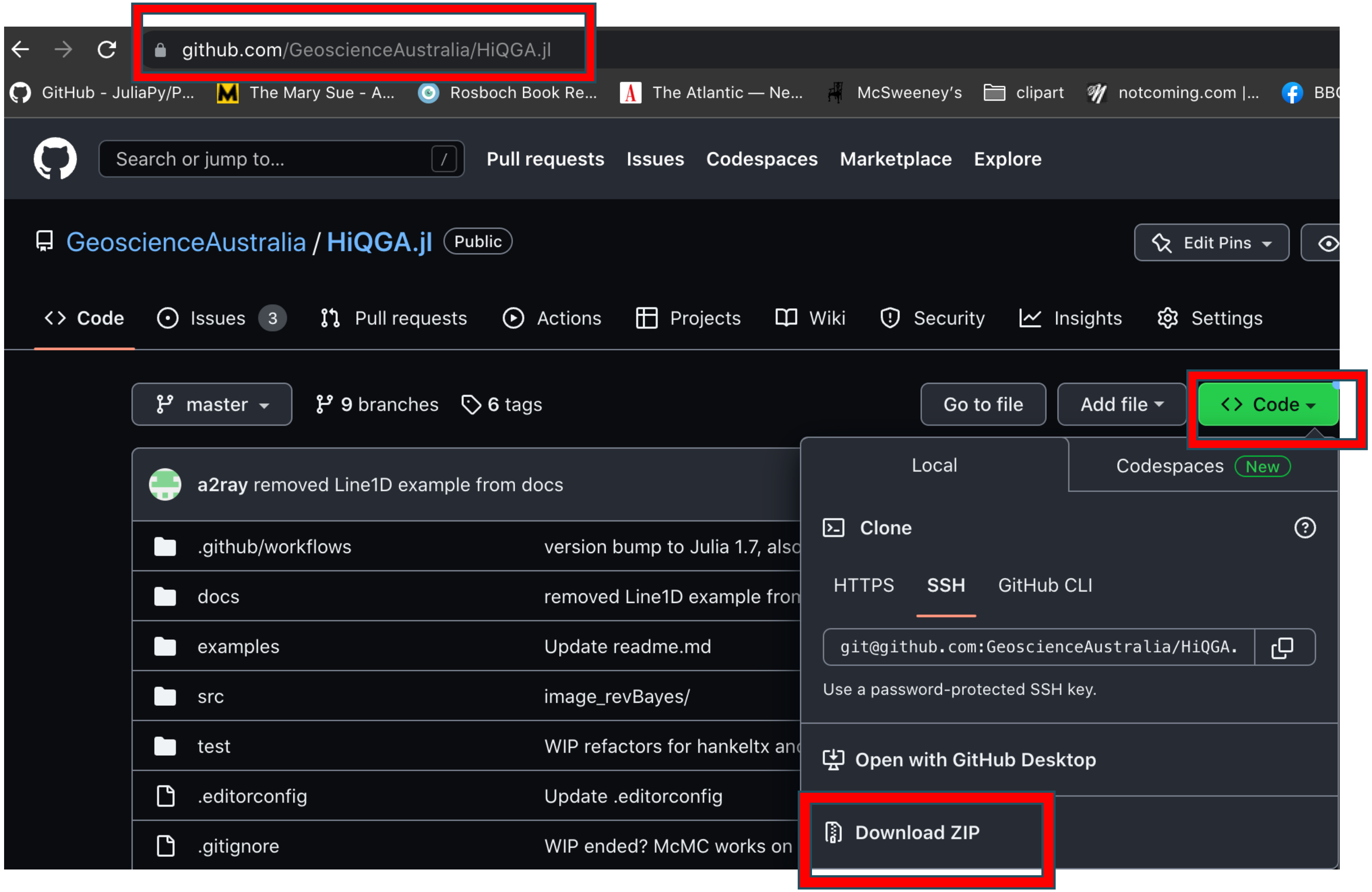
Installation

To install the latest stable release, in a perfect world we'd use Julia's `Pkg` REPL by hitting `]` to enter `pkg>` mode. Then enter the following, at the `pkg>` prompt:

```
pkg> add HiQGA
```

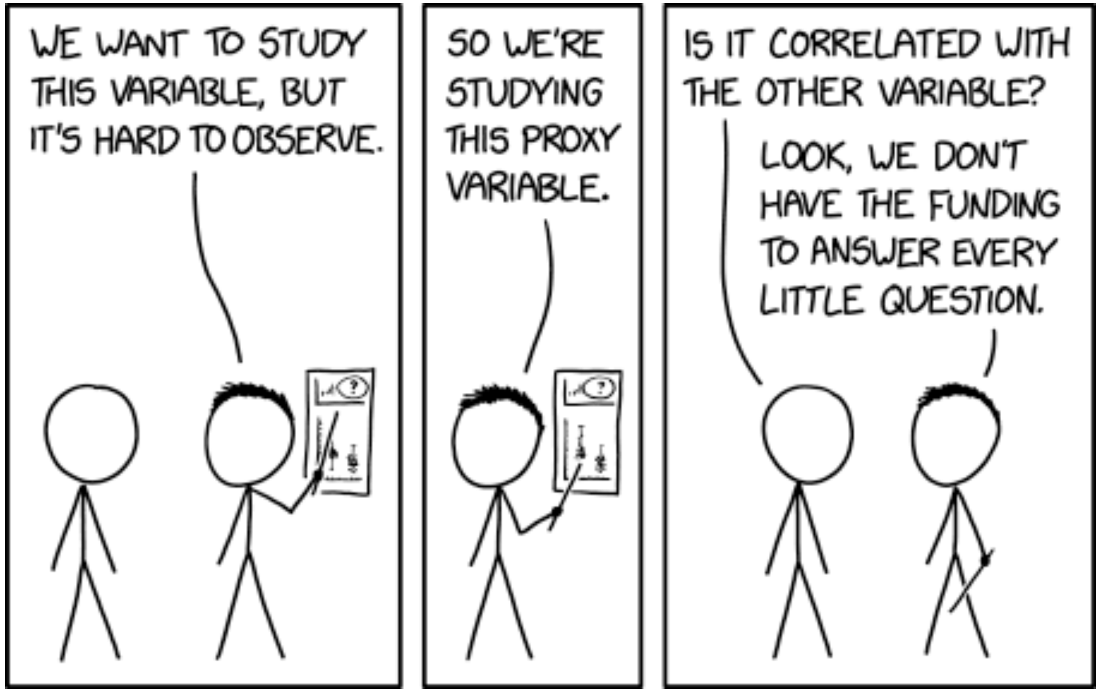
https://julialang.org/downloads

Download the entire HiQGA package



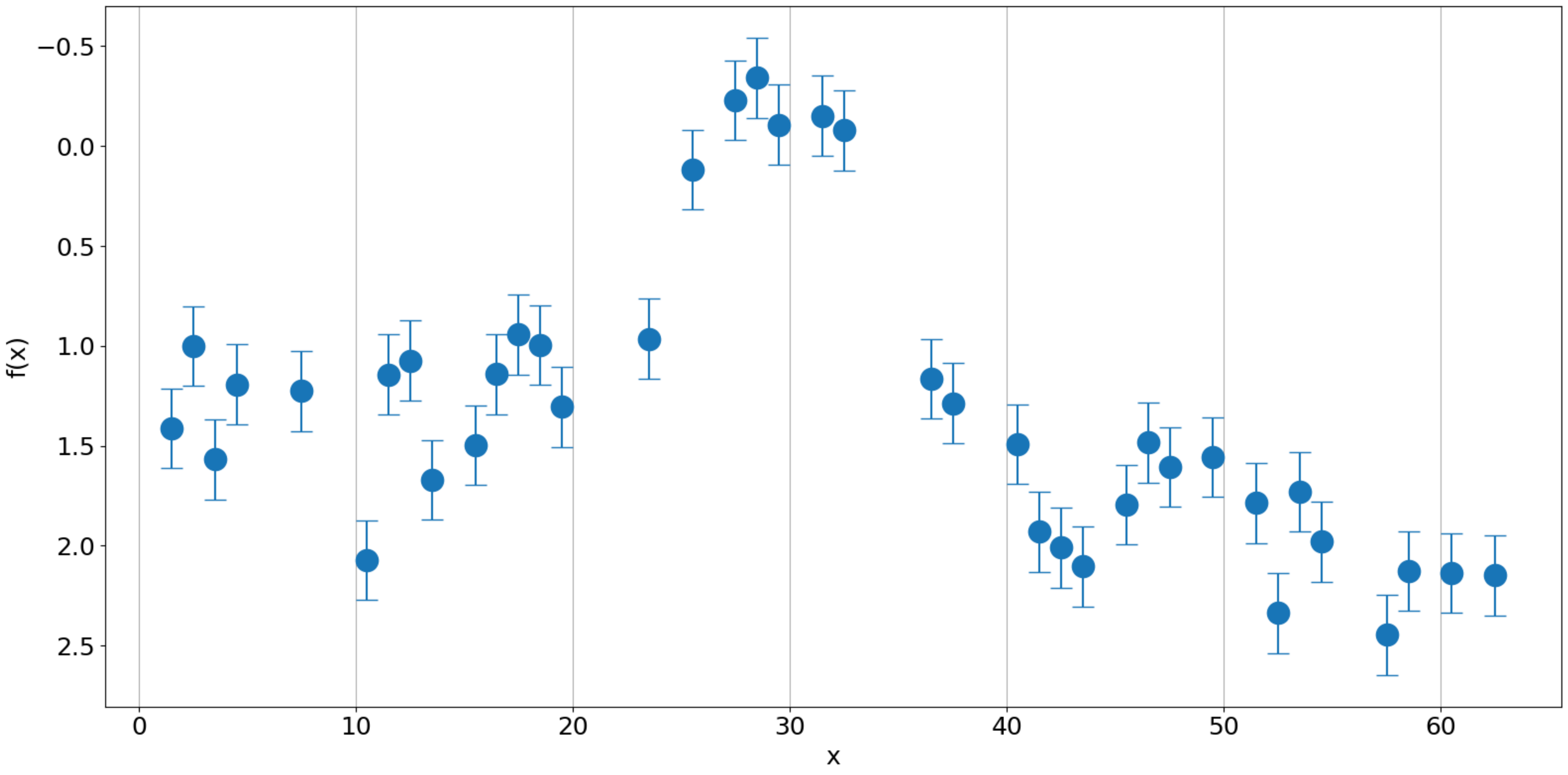
- Go to the highlighted URL
- Hit the Code box
- Download the entire package as a zip file
- Or you can clone it with git

Inference in a nutshell



https://xkcd.com/2652/

How would you fit this?



m = 39 data points

One way to represent this

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$n = 65$

Representing one observation

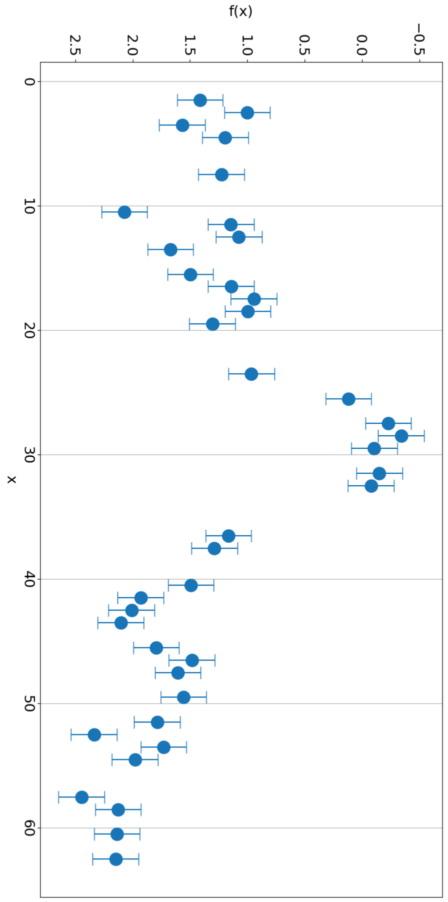
$$
\begin{bmatrix} y_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \mathbf{e}_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
$$

But we don’t have all observations

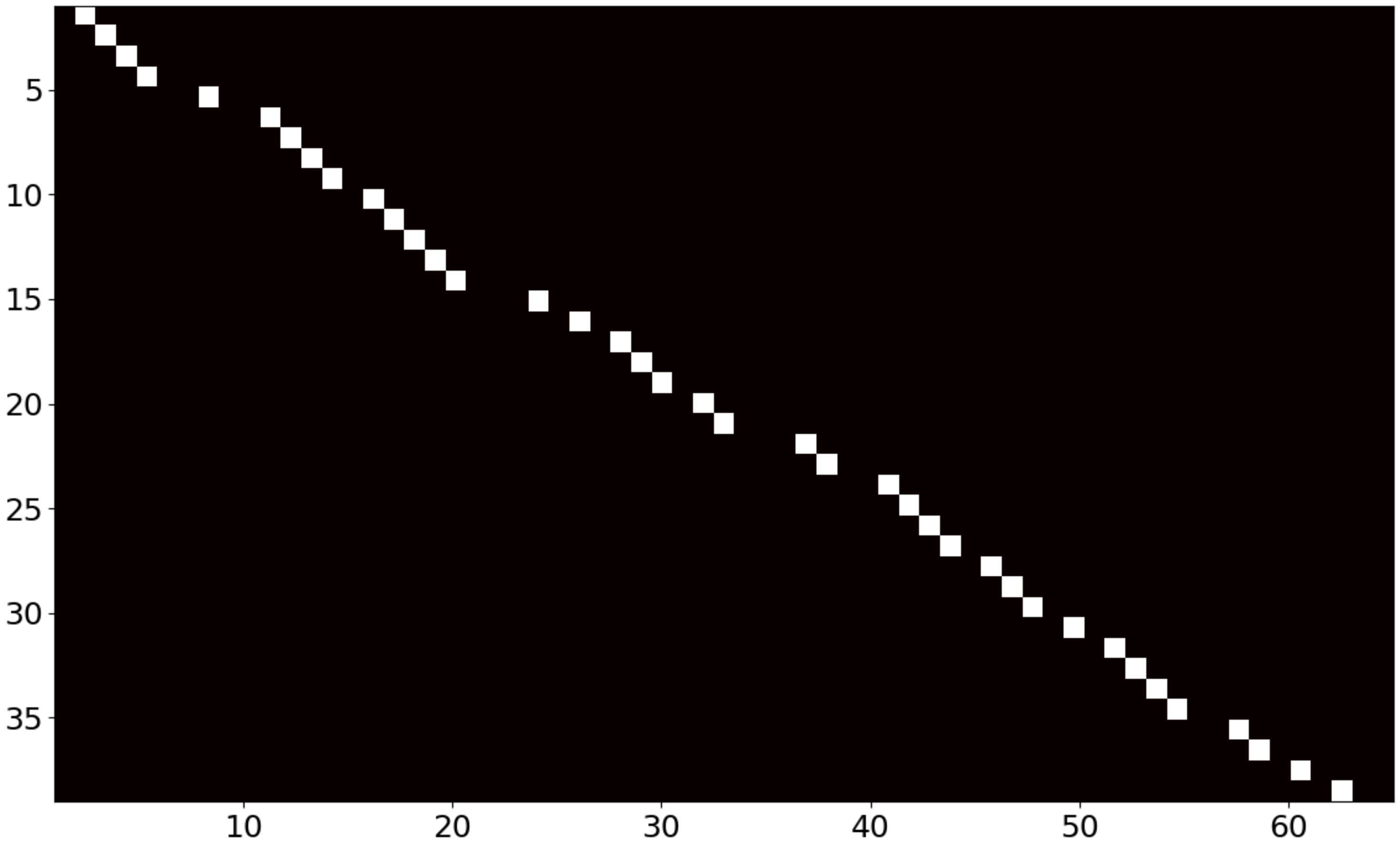
$$\begin{bmatrix} y_2 \\ y_4 \\ y_9 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_4 \\ \mathbf{e}_9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

m = 3 data points for example

A system of equations



=



$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

y A x

Least squares … and we're done!

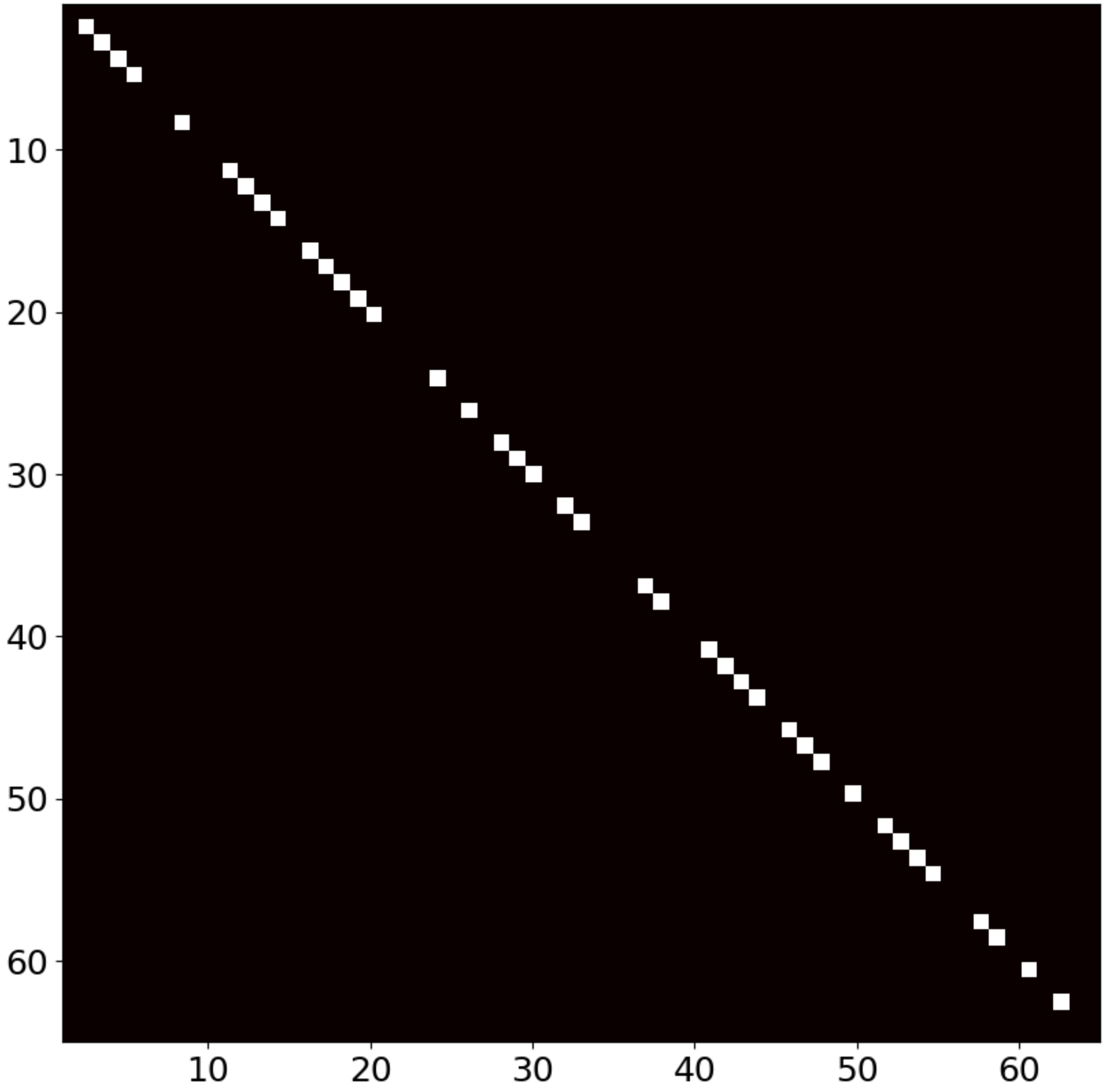
$$\phi = ||\mathbf{y} - \mathbf{A}\mathbf{x}||^2,$$

$$\text{set } \nabla_x \phi = 0,$$

$$\hat{\mathbf{x}} = (\mathbf{A}^\mathbf{t}\mathbf{A})^{-1}\mathbf{A}^\mathbf{t}\mathbf{y}.$$

Or are we?

$$A^t A$$

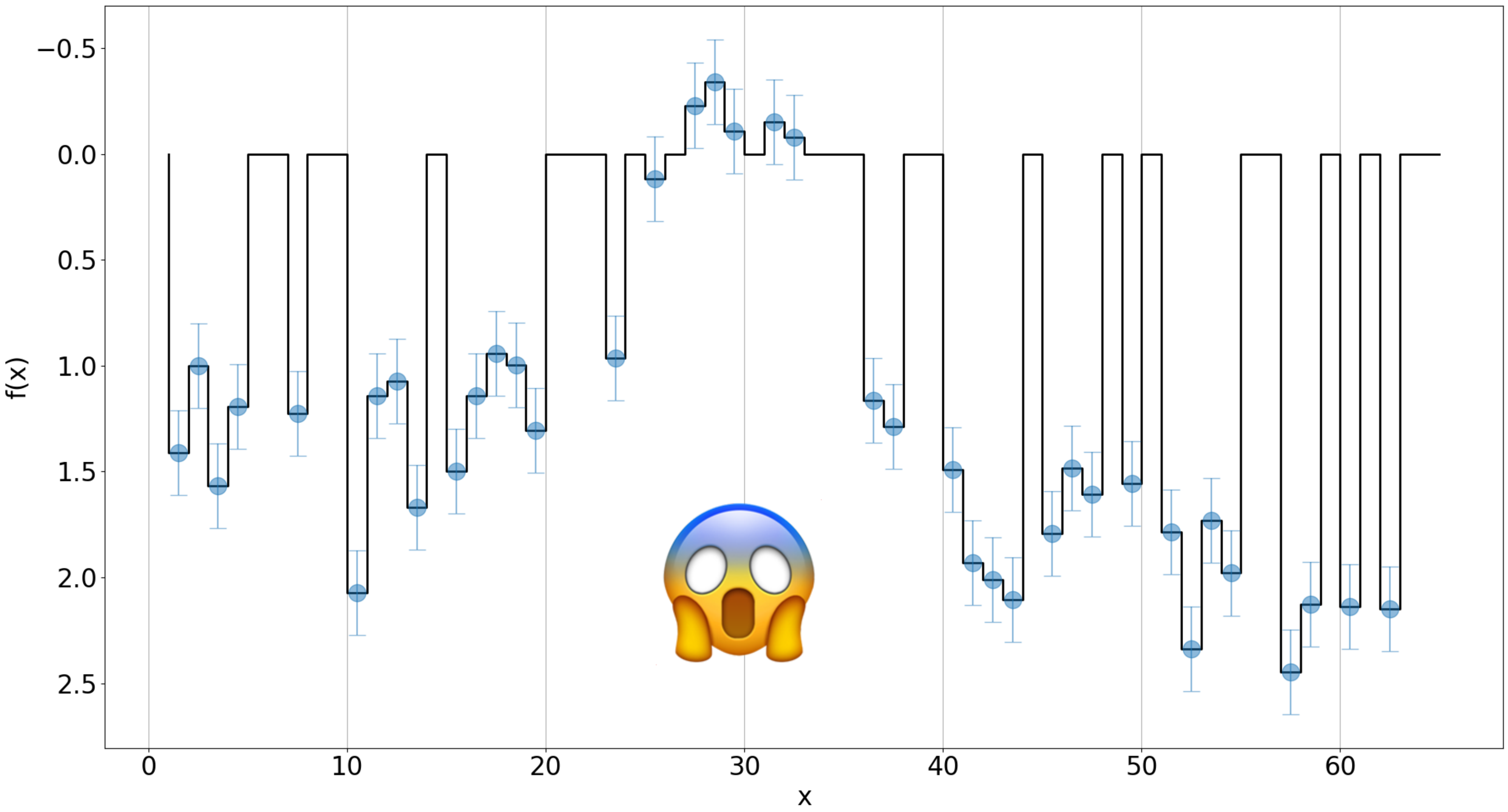


Add to the diagonal of A^tA

$$\hat{\mathbf{x}}_{\mathrm{ridge}} = (\mathbf{A}^{\mathbf{t}}\mathbf{A} + \delta^2 \mathbf{I})^{-1}\mathbf{A}^{\mathbf{t}}\mathbf{y}.$$

10^{-16}

Ridge solution

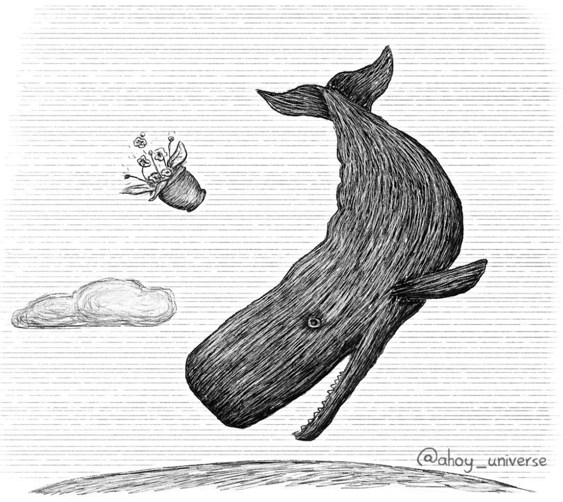


Enforce smoothness instead

$$\begin{bmatrix} & 0 & & & \\ -1 & 1 & & 0 & \\ & -1 & 1 & & \\ & & & \cdots & \\ & 0 & & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ -x_1 + x_2 \\ -x_2 + x_3 \\ \vdots \\ -x_{n-1} x_n \end{bmatrix}$$

R x = diff(x)

Oh no, not again …



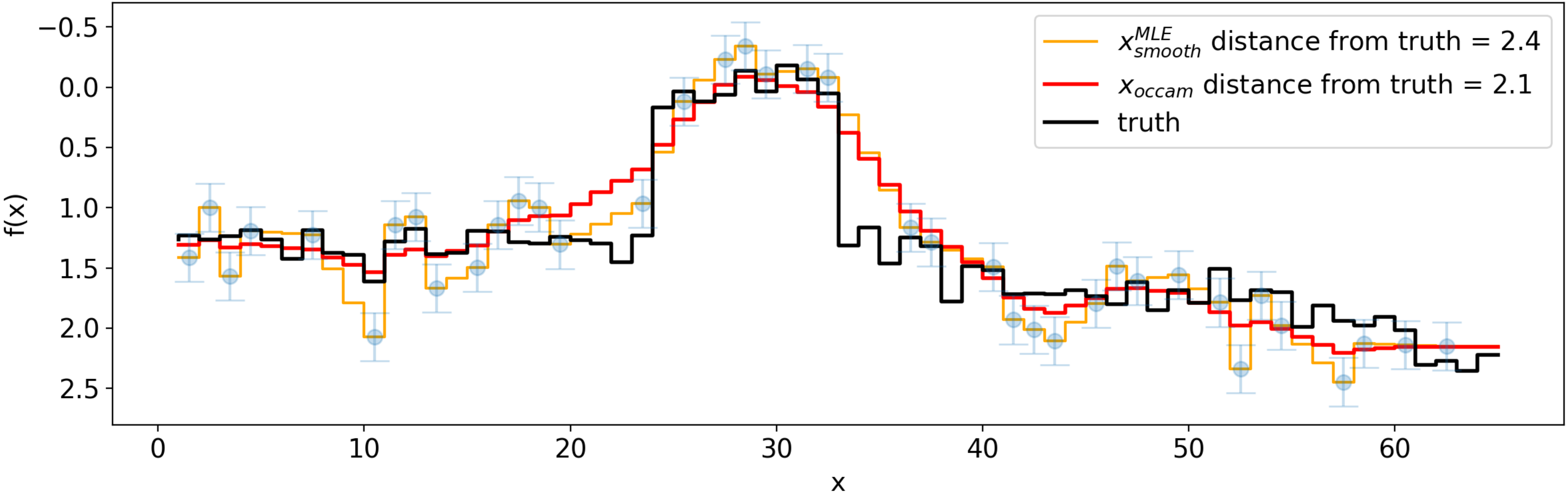
$$\phi = ||\mathbf{y} - \mathbf{Ax}||^2 + \lambda^2||\mathbf{Rx}||^2,$$

$$\text{set } \nabla_x \phi = 0,$$

$$\hat{\mathbf{x}}^{\text{MLE}}_{\text{smooth}} = (\mathbf{A^t A} + \lambda^{\mathbf{2}}\mathbf{R^t R})^{-1}\mathbf{A^t y}$$

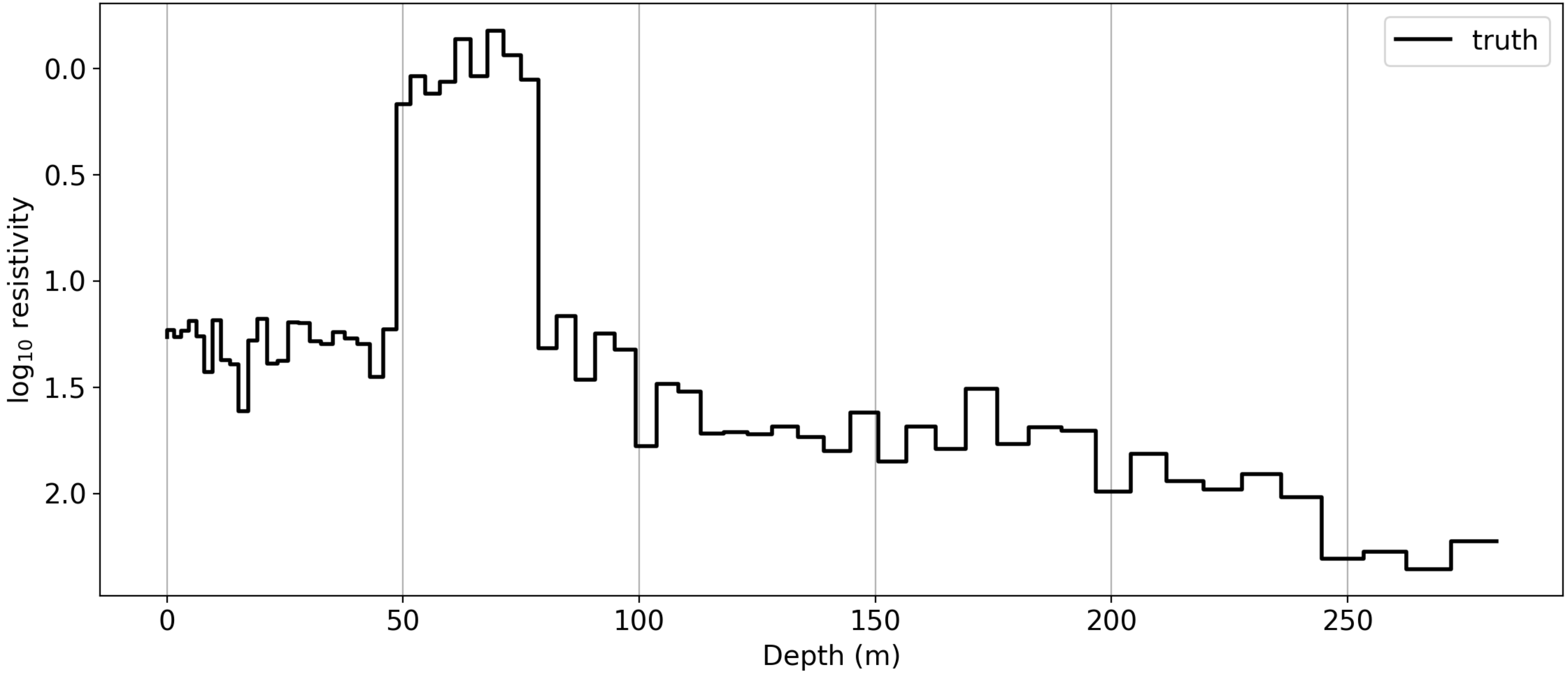
10^{-16}

Occam vs MLE and "distance from the truth"



Occam = ***?Guaranteed¿ smoothest model*** within data noise!

But the truth this is from a well log!



The general, non-linear case

$$\phi(\mathbf{m}) = \frac{1}{2}\left(||\mathbf{W}(\mathbf{d} - \mathbf{f}(\mathbf{m}))||^2 + \lambda^2||\mathbf{Rm}||_p^p\right),$$

but now set $p = 2$,

$$\phi(\mathbf{m}) = \frac{1}{2}\left(||\mathbf{W}(\mathbf{d} - \mathbf{f}(\mathbf{m}))||^2 + \lambda^2||\mathbf{Rm}||^2\right),$$

but how to set $\nabla_m \phi = 0$?

linearize $\phi(\mathbf{m})$ to $\phi(\mathbf{m} + \mathbf{\Delta m})$ i.e.,

$\mathbf{f}(\mathbf{m}) \rightarrow \mathbf{f}(\mathbf{m} + \mathbf{\Delta m}), \mathbf{Rm} \rightarrow \mathbf{R}(\mathbf{m} + \mathbf{\Delta m})$ first.

$$\boxed{\mathbf{f}(\mathbf{m} + \mathbf{\Delta m}) \approx \mathbf{f}(\mathbf{m}) + \mathbf{J\Delta m}.}$$

Radiohead said it … creep

first write residual $\mathbf{r} \approx \mathbf{f}(\mathbf{m}) - \mathbf{d}$

derive with respect to $\mathbf{\Delta m}$,

set $\dfrac{\partial \phi}{\partial \mathbf{\Delta m}} = 0,$ giving,

$$\mathbf{\Delta m} = -\left(\mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{J} + \lambda^2\mathbf{R}^t\mathbf{R}\right)^{-1}\left(\mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{r} + \lambda^2\mathbf{R}^t\mathbf{R}\mathbf{m}\right)$$

note also, that $\nabla_m \phi = \mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{r} + \lambda^2\mathbf{R}^t\mathbf{R}\mathbf{m}.$ Gradient

note finally, that $\dfrac{\partial(\nabla_m \phi)}{\partial \mathbf{m}} = \mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{J} + \lambda^2\mathbf{R}^t\mathbf{R}.$ Approximate Hessian

Gradient descent!

$$\boxed{\mathbf{m}_{\text{new}} = \mathbf{m} + \boldsymbol{\Delta}\mathbf{m}}$$

$$\text{writing } \nabla_m \phi = \mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{r} + \lambda^2\mathbf{R}^t\mathbf{R}\mathbf{m},$$

$$\text{and } \eta = \left(\mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{J} + \lambda^2\mathbf{R}^t\mathbf{R}\right)^{-1} \text{ we now say,}$$

$$\boxed{\mathbf{m}_{\text{new}} = \mathbf{m} - \eta\nabla_m \phi}$$

Successive linearization:
Replace **m** with $\mathbf{m}_{\text{new}}$
Continue until residual is within noise
Find smoothest model within data error, as usual.

Gradient descent → Bayes theorem

rewriting $\phi(\mathbf{m}) = \frac{1}{2}\left(||\mathbf{W}(\mathbf{d} - \mathbf{f}(\mathbf{m}))||^2 + \lambda^2||\mathbf{R}\mathbf{m}||^2\right)$ as,

$$\phi(\mathbf{m}) = \frac{1}{2}\left([\mathbf{d} - \mathbf{f}(\mathbf{m})]^t\mathbf{W}^t\mathbf{W}[\mathbf{d} - \mathbf{f}(\mathbf{m})] + \lambda^2\mathbf{m}^t\mathbf{R}^t\mathbf{R}\mathbf{m}\right),$$

identifying $\lambda^2\mathbf{R}^t\mathbf{R} = \mathbf{C}_m^{-1}$,

and $\mathbf{W}^t\mathbf{W} = \mathbf{C}_d^{-1}$,

$$\phi(\mathbf{m}) = \frac{1}{2}\left([\mathbf{d} - \mathbf{f}(\mathbf{m})]^t\mathbf{C}_d^{-1}[\mathbf{d} - \mathbf{f}(\mathbf{m})] + \mathbf{m}^t\mathbf{C}_m^{-1}\mathbf{m}\right),$$

further identifying $\log p(\mathbf{m}|\mathbf{d}) = -\phi(\mathbf{m}) + \text{const},$

and $\log p(\mathbf{d}|\mathbf{m}) = -\frac{1}{2}\left([\mathbf{d} - \mathbf{f}(\mathbf{m})]^t\mathbf{C}_d^{-1}[\mathbf{d} - \mathbf{f}(\mathbf{m})]\right),$

and $\log p(\mathbf{m}) = -\frac{1}{2}\mathbf{m}^t\mathbf{C}_m^{-1}\mathbf{m},$

we can write for the non-linear yet Gaussian case,

posterior $p(\mathbf{m}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{m}) \cdot p(\mathbf{m}).$

A traditional Bayesian view

$$\text{updated belief} \propto \text{likelihood of belief} \cdot \text{prior belief}$$

$$p(\mathbf{m}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{m}) \cdot p(\mathbf{m})$$

Given the observed geophysical data **d,** new belief in model **m**

Given the model **m,** accuracy of geophysical prediction

m is a model obtained from prior notions, e.g., well data, geology, etc.

Equivalence of Bayes' theorem with optimization

$$\text{updated belief} \propto \text{likelihood of belief} \cdot \text{prior belief}$$

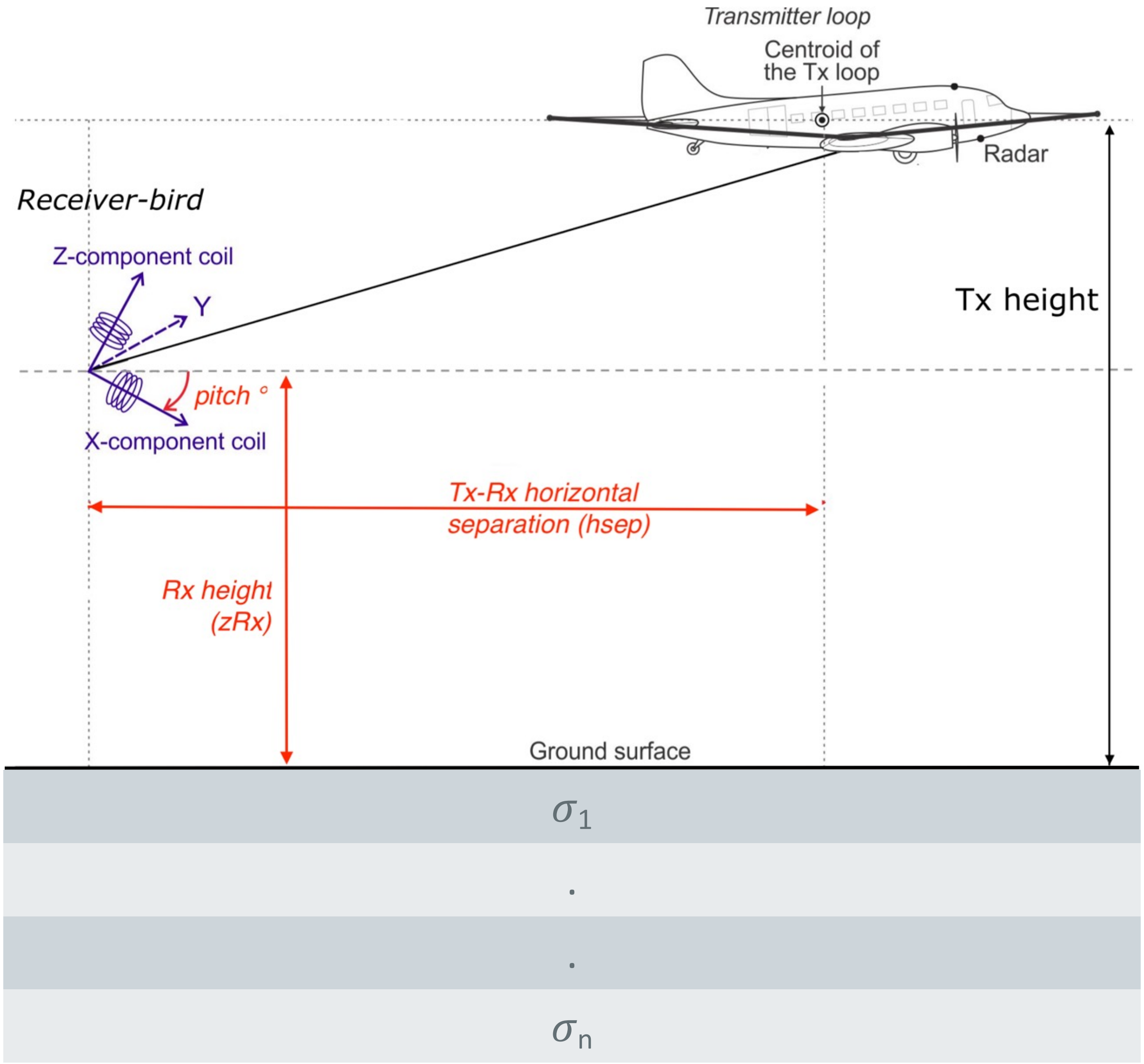
$$p(\mathbf{m}|\mathbf{d}) \propto p(\mathbf{d}|\mathbf{m}) \cdot p(\mathbf{m})$$

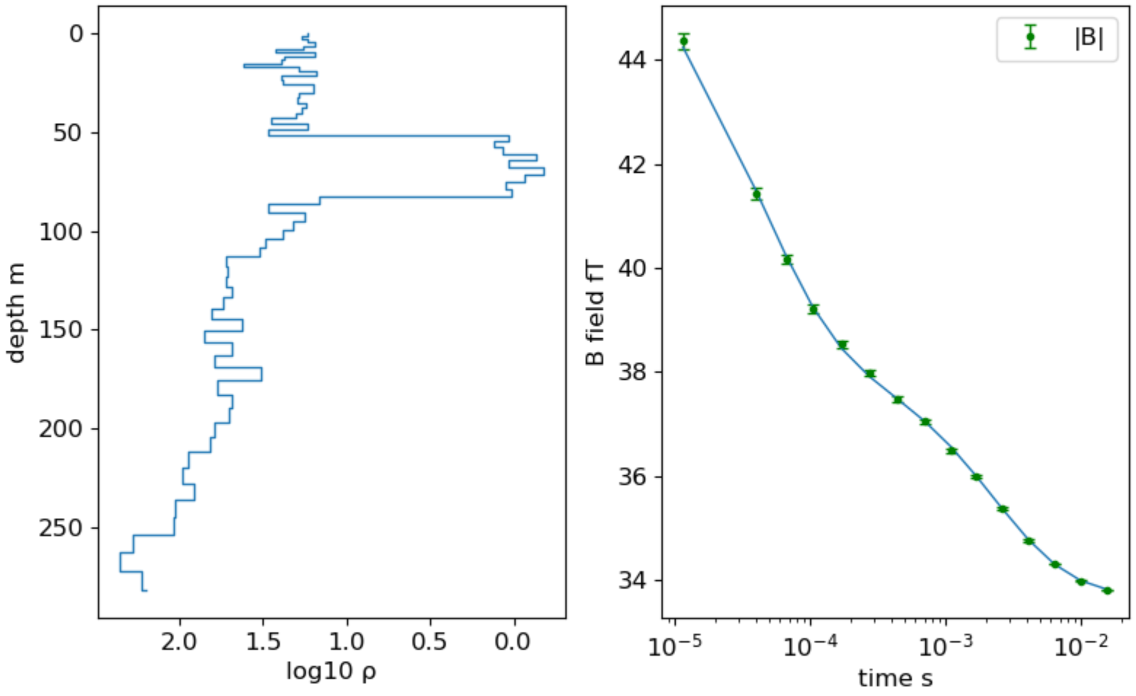
$$\arg\min \phi(\mathbf{m}) = ||\mathbf{W}(\mathbf{d} - \mathbf{f}(\mathbf{m}))||_2^2 + \lambda^2 ||\mathbf{R}\mathbf{m}||_p^p$$

There is NO objective, unbiased inversion.

- Choices need to be made!
- Occam is **one** good choice

An AEM inverse problem





Occam and posterior solutions

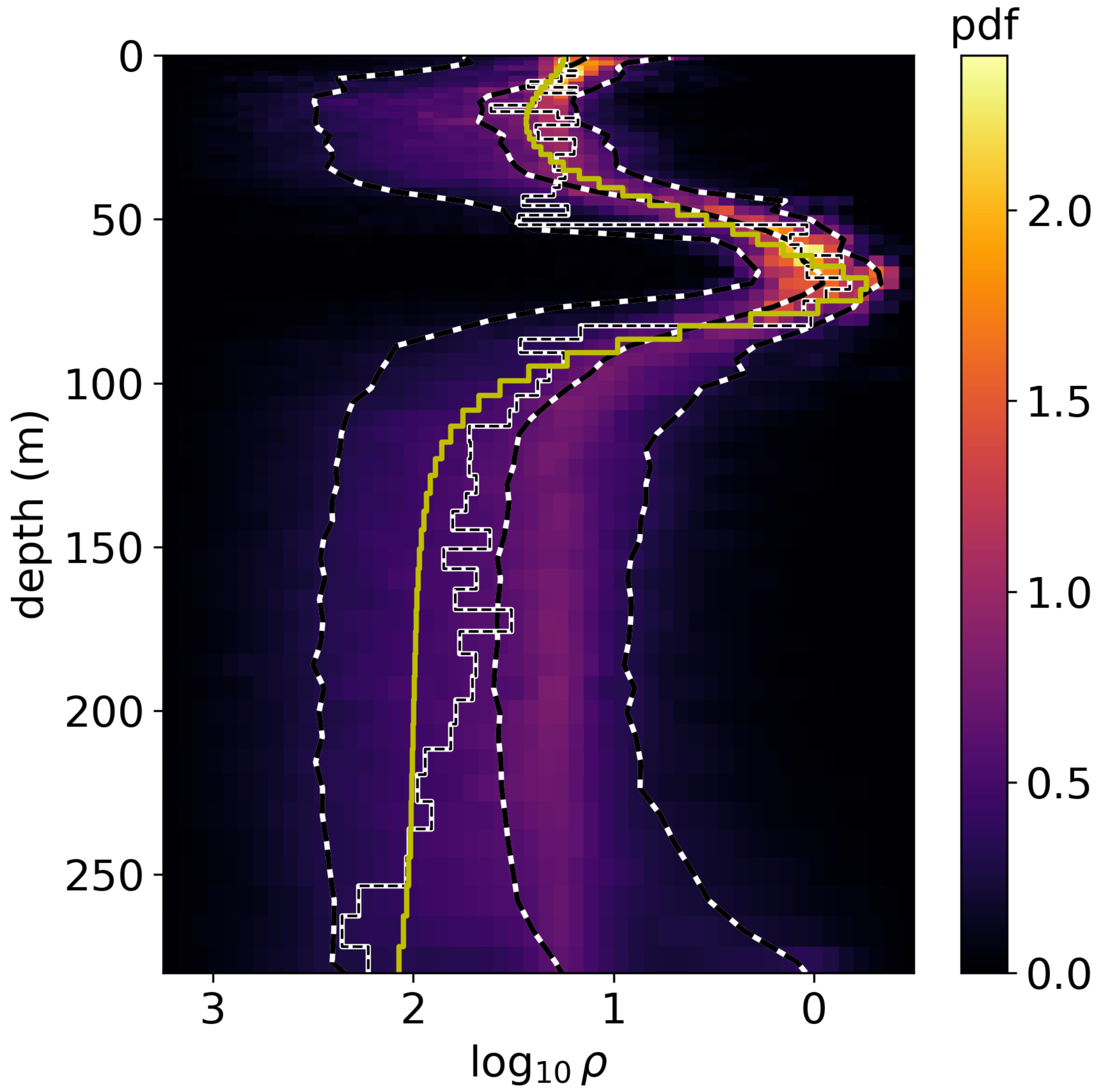
HiQGA.jl / examples / tempest / synth / gradientbased /

- 01_make_model.jl
- 02_set_options.jl
- 03_run_inversion.jl
- 03_run_inversion_nuisance.jl
- 04_plot_results.jl

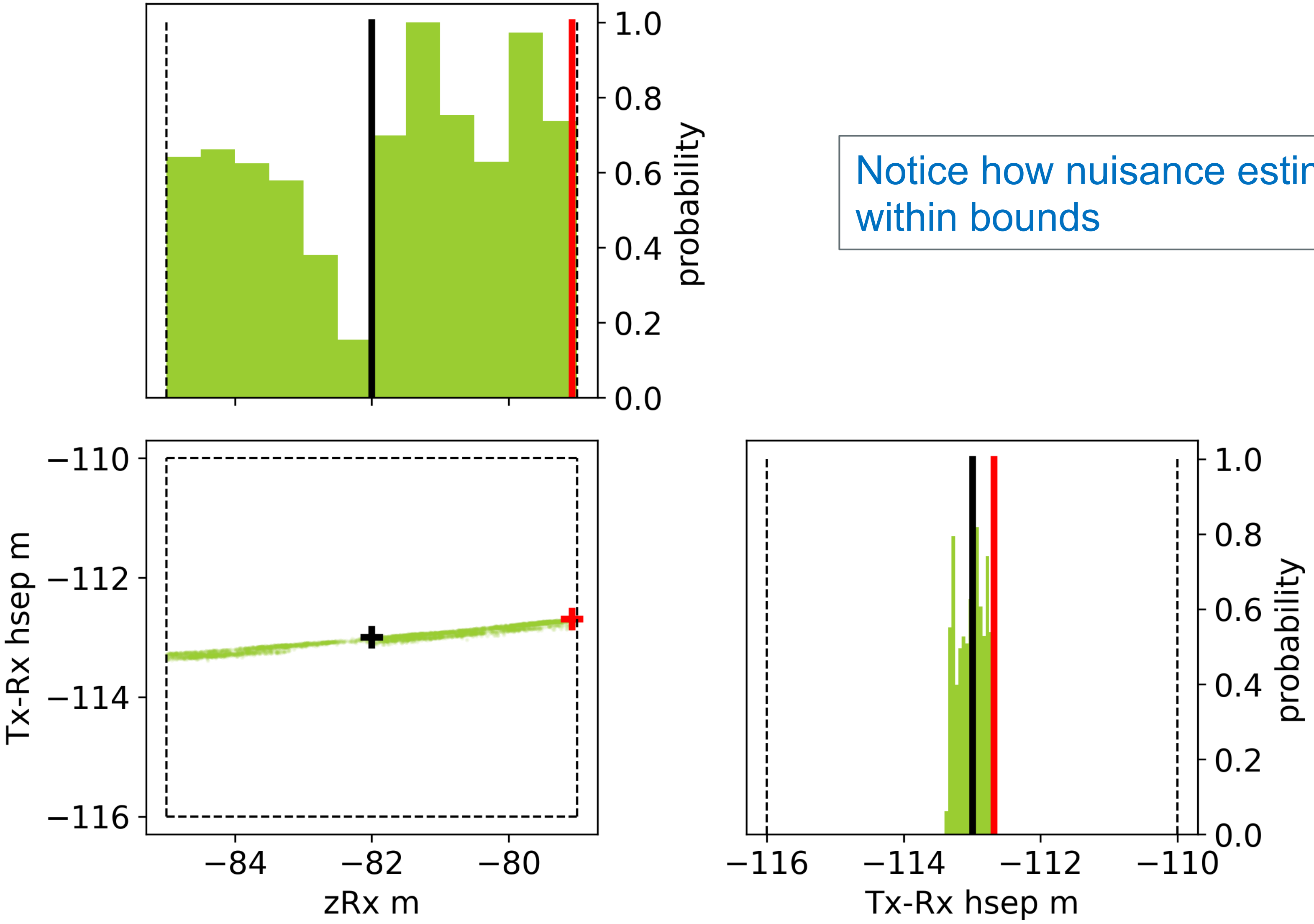
HiQGA.jl / examples / tempest / synth / McMC /

- 01_make_model.jl
- 02_make_aem_inversion_opts_nuisance.jl
- 03_run_aem_inversion_s.jl
- 04_plot_results.jl
- electronics_halt.jl

Notebook style code execution



Hierarchical Bayesian nuisance crossplots



Notice how nuisance estimates are within bounds

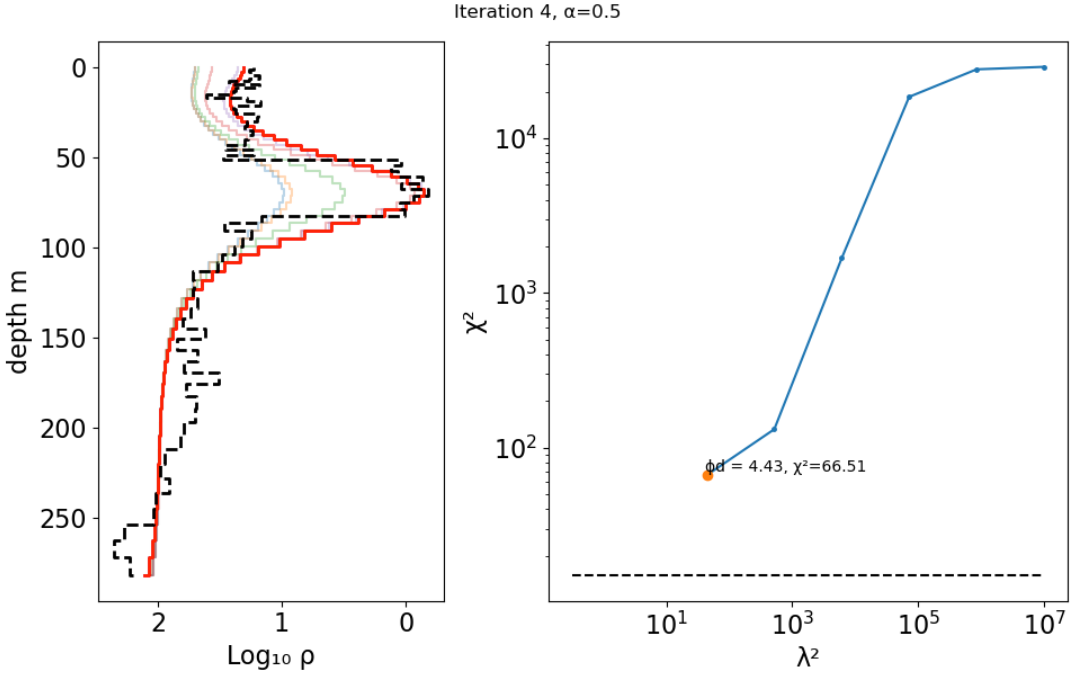
Julia code

```
m, nu, χ², χ²nu, λ², idx = transD_GP.gradientinv(σstart, σ0, nustart, tempest;
                                  nstepsmax=30,
                                  # Occam stuff
                                  λ²min, λ²max, β², ntries,
                                  lo, hi, regtype,
                                  # optim stuff
                                  nubounds,
                                  ntriesnu = 5,
                                  boxiters = 2,
                                  usebox = true,
                                  reducenuto = 0.2,
                                  debuglevel = 2,
                                  breaknuonknown = false);
```

Same physics operator, different Julia methods

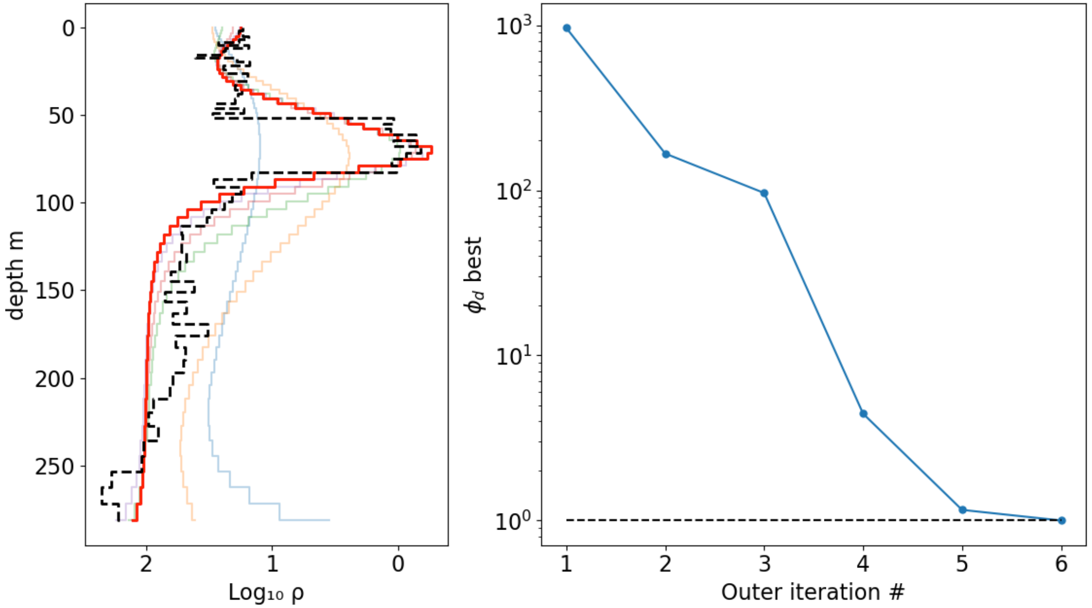
```
## run McMC
@time transD_GP.main(opt, optn, tempest, Tmax=Tmax, nsamples=nsamples, nchains=nchains, nchainsatone=nchainsatone)
```

Get insight into the inversion, within Julia



Line searches and step sizes are a nasty bag of tricks!

2 stage alternating inversion
- Conductivities (within bounds Occam)
- Tx-Rx Geometry (Barrier BFGS)



Write code as the math is derived

```julia
function makeregR1(F::Operator1D)
    n = length(F.ρ) - F.nfixed
    LinearMap(R1Dop, Rt1Dop, n)
end

function R1Dop(x::Vector)
    vcat(0, diff(x))
end

function Rt1Dop(y::Vector)
    x = vcat(-diff(y),y[end])
    x[1] = -y[2]
    x
end
```

```julia
JtW, Wr = F.J'*F.W, F.W*F.res
H = (JtW*(JtW)' + λ²*R'R + λ²*β²*I)
U = cholesky(Positive, H, Val{false}).U
```

Matrix free regularization operator construction!

```
julia> R = transD_GP.makeregR1(tempest)
65×65 LinearMaps.FunctionMap{Float64}(R1Dop, Rt1Dop; ismutating=false, issymmetric=false, ishermitian=false, is
posdef=false)
```

Inspect it explicitly

```
julia> Matrix(R)
65×65 Matrix{Float64}:
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
 -1.0   1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
  0.0  -1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
  ⋮                        ⋮                        ⋱  ⋮                        ⋮
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0  -1.0   1.0   0.0  0.0
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0  -1.0   1.0  0.0
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0  -1.0  1.0
```

Cascade operators and inspect!

```
julia> Matrix(R*R)
65×65 Matrix{Float64}:
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
 -1.0   1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
  1.0  -2.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0  0.0
  ⋮                        ⋮                        ⋱  ⋮                        ⋮
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  1.0  -2.0   1.0   0.0  0.0
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   1.0  -2.0   1.0  0.0
  0.0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0     0.0  0.0  0.0  0.0  0.0  0.0   0.0   1.0  -2.0  1.0
```

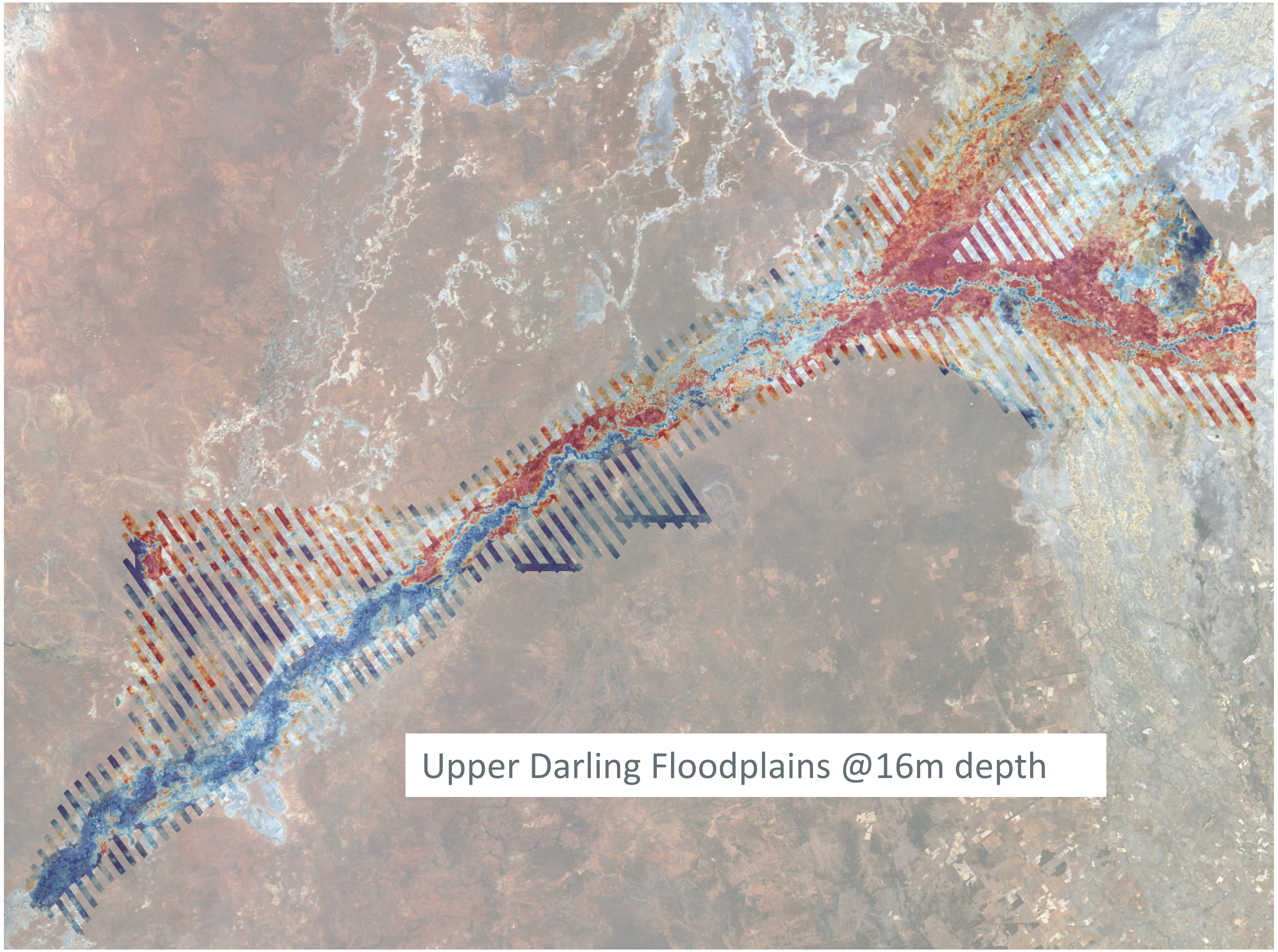
note finally, that $\dfrac{\partial(\nabla_m \phi)}{\partial \mathbf{m}} = \mathbf{J}^t\mathbf{W}^t\mathbf{W}\mathbf{J} + \lambda^2\mathbf{R}^t\mathbf{R}.$ Approximate Hessian

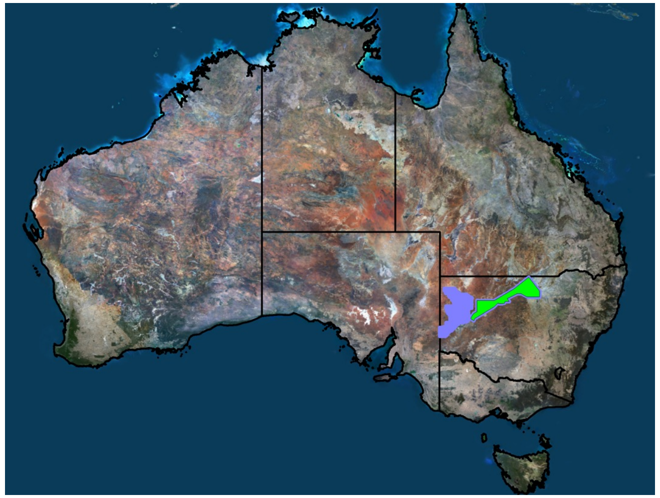
Natively parallel programming paradigm

```
function domcmciters(iterlast, nsamples, chains, m::DArray{ModelStat},
            opt::DArray{OptionsStat}, stat,
            current_misfit, F, wp, nominaltime)
    # purely stationary GP moves

    t, tlong = map(x->time(), 1:2)
    for isample = iterlast+1:iterlast+nsamples
        swap_temps(chains)
        @sync for (chain_idx, chain) in enumerate(chains)
            # purely stationary GP moves
            @async chain.misfit = remotecall_fetch(do_mcmc_step, chain.pid,
                                        m, opt, stat,
                                        current_misfit, F,
                                        chain.T, isample, wp, chain_idx, chain.master_pid)
        end
        t, tlong, doquit = disptime(isample, t, tlong, iterlast, nsamples, nominaltime)
        doquit && break
    end
end
```

One-sided parallelism – no need to do something different depending on MPI rank

Scale up your prototype, *within* Julia



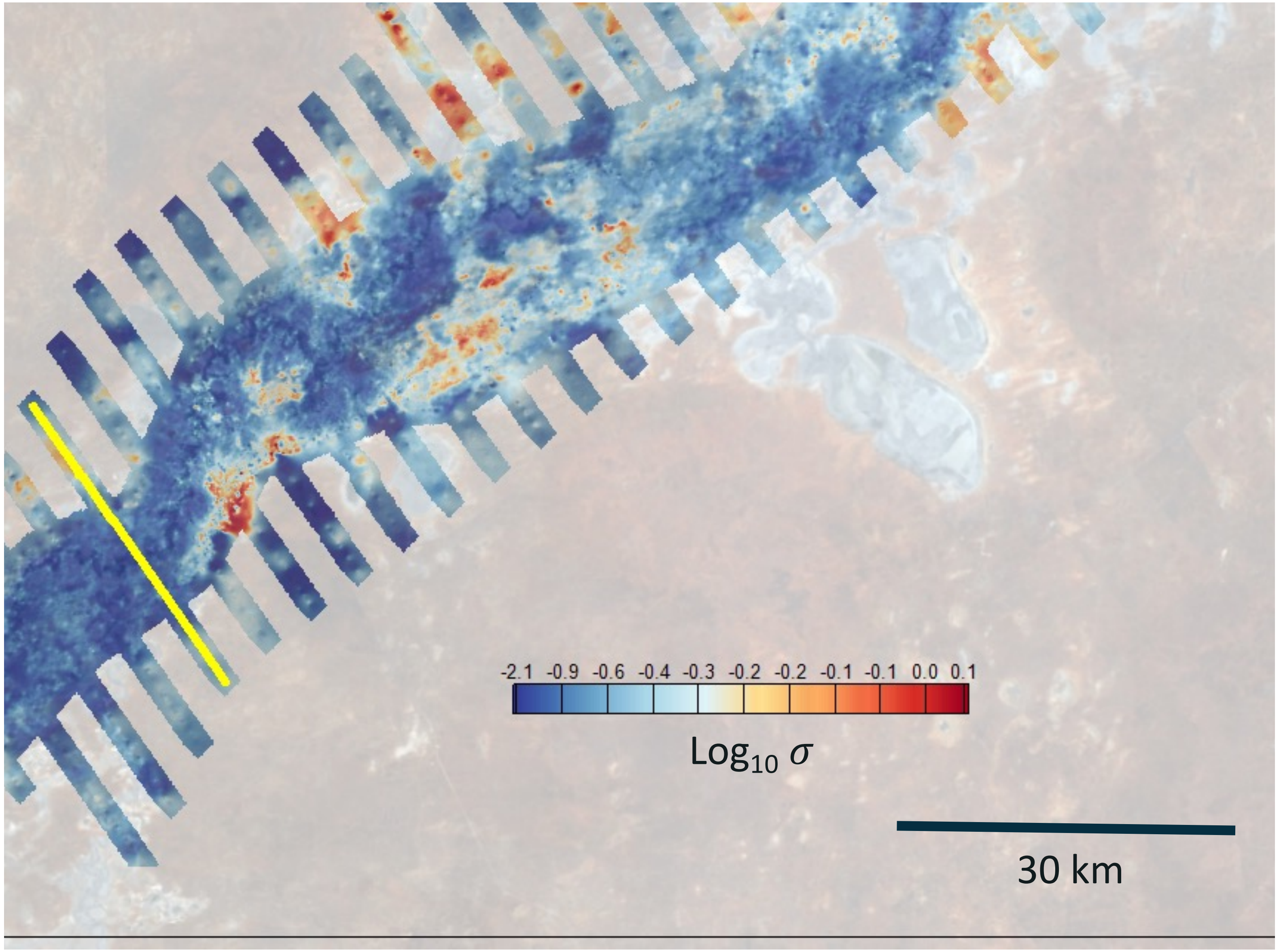


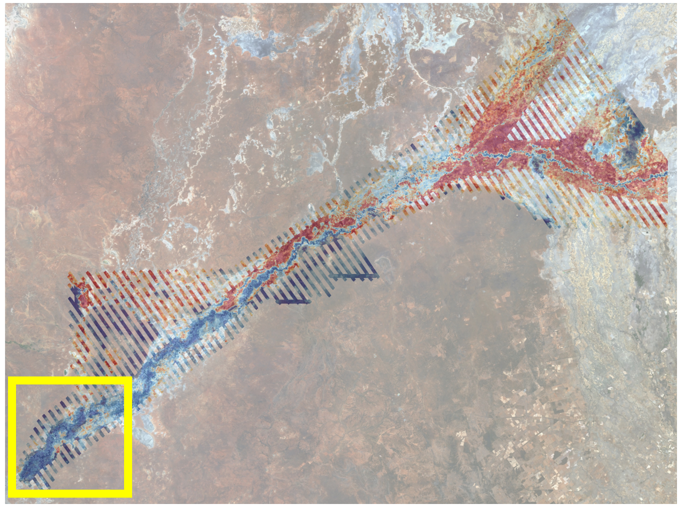
25,000 line-km of AEM data, inverted using HiQGA.jl

Australian Government
Geoscience Australia

NSW GOVERNMENT
Planning and Environment

Zooming in



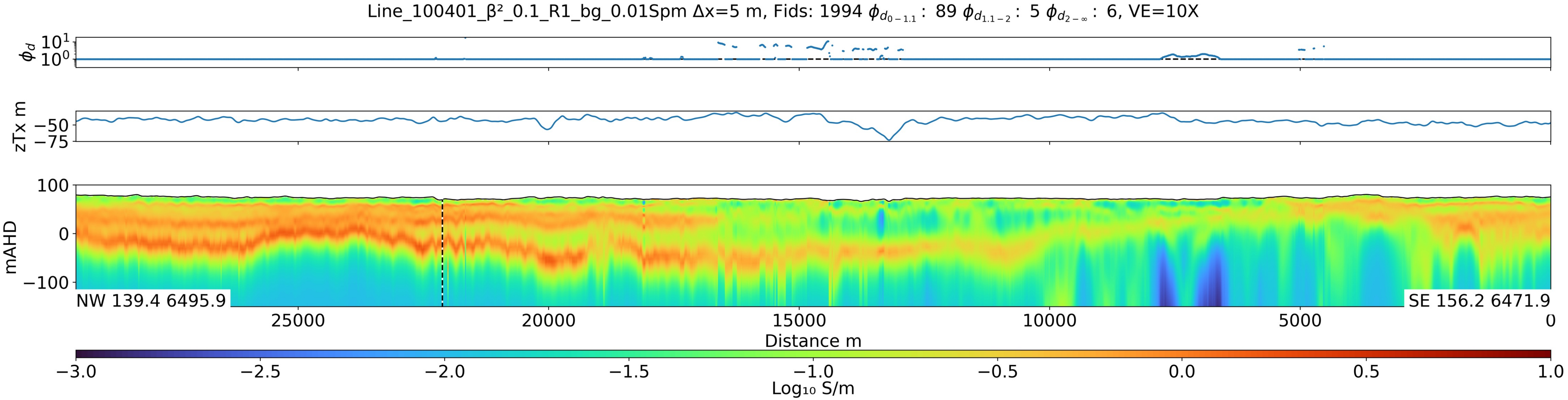


25,000 line-km of AEM data, inverted using HiQGA.jl

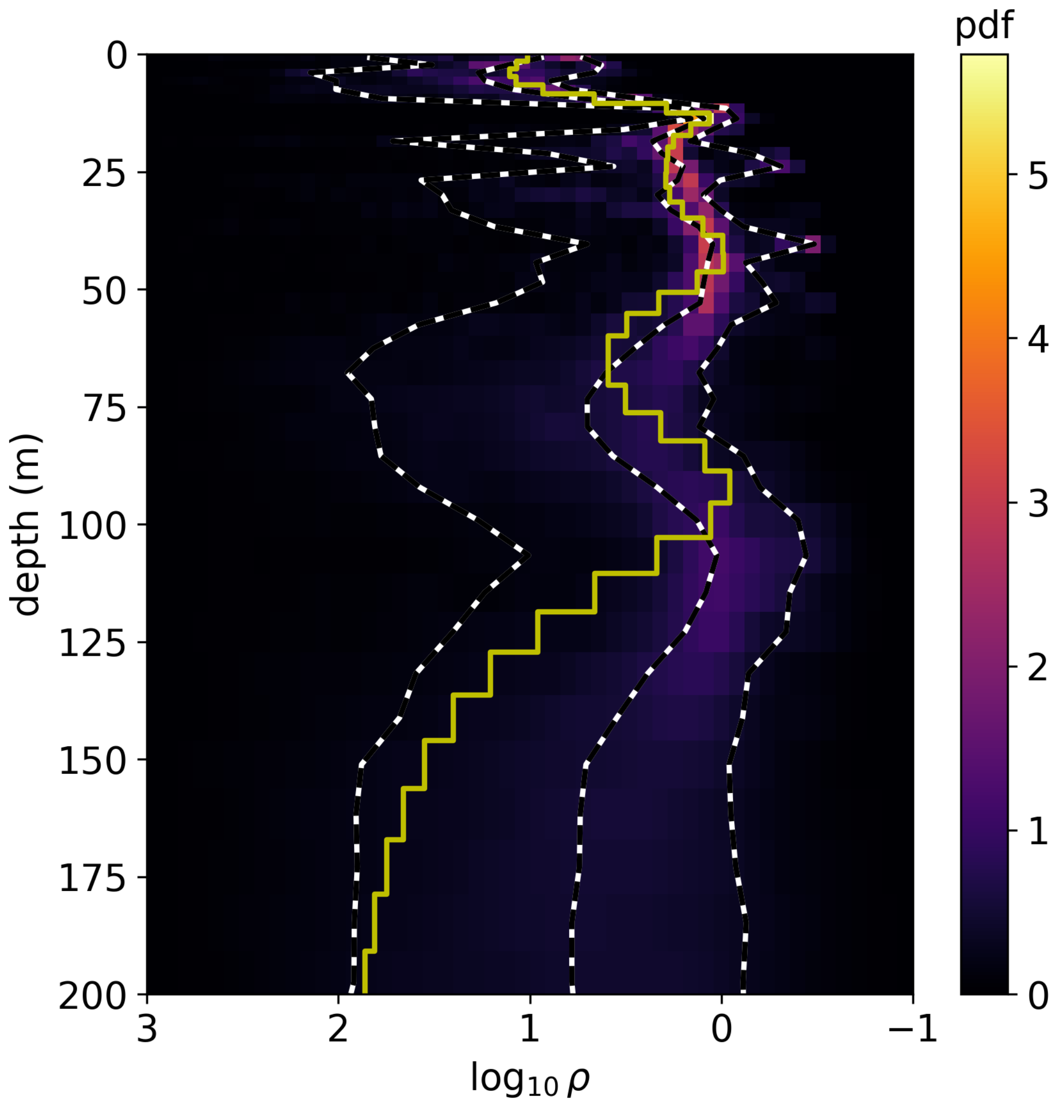
Australian Government
Geoscience Australia

NSW GOVERNMENT
Planning and Environment

Beautiful, layered earth

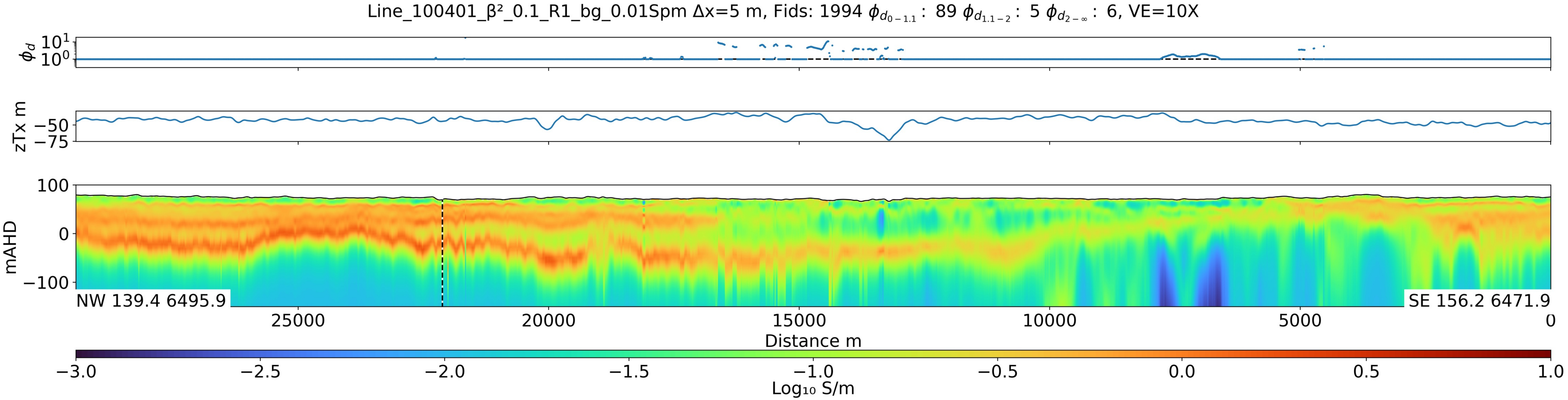


Inspect probabilities around one sounding

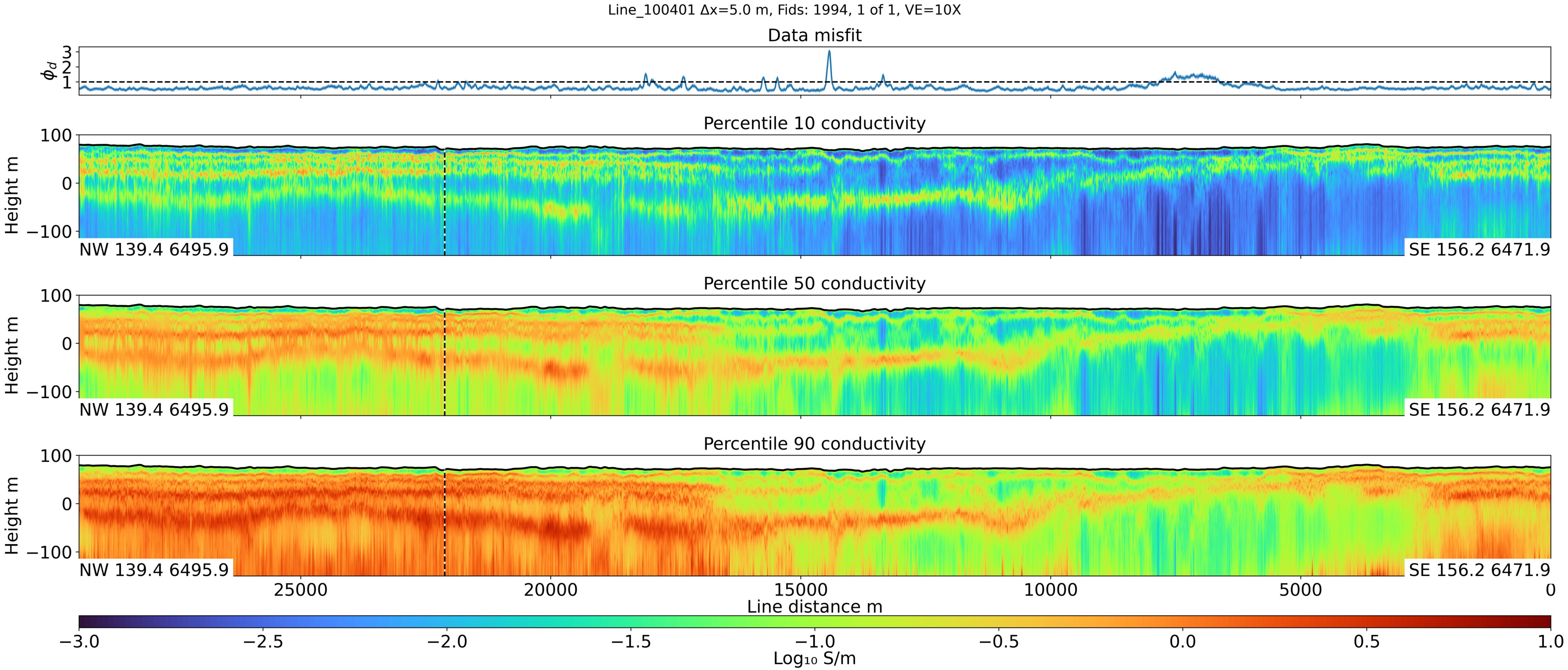


Remember, the Occam model is an *extremal* model! There are other models in high probability regions (in chicken neck CIs)

Back to the deterministic section

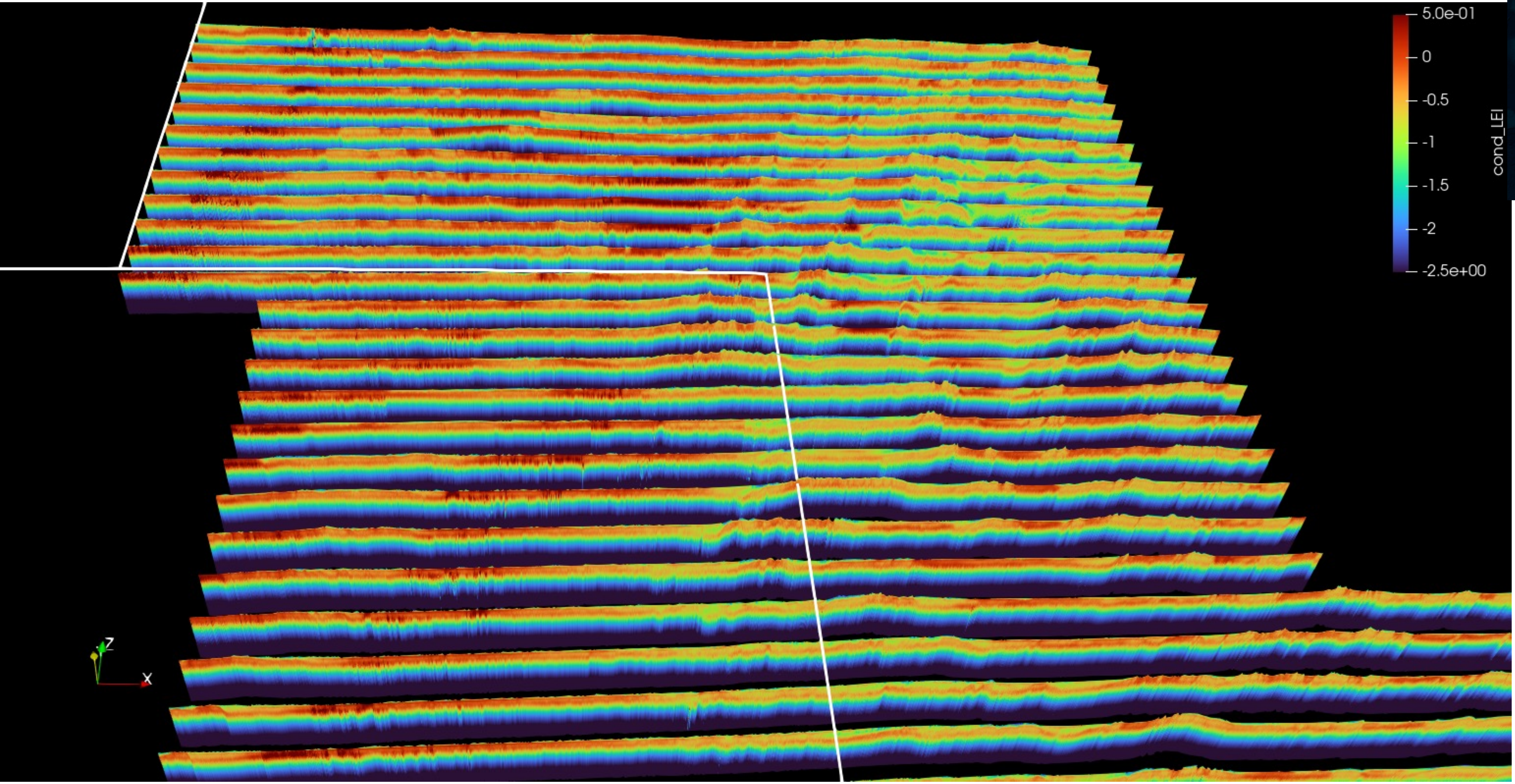


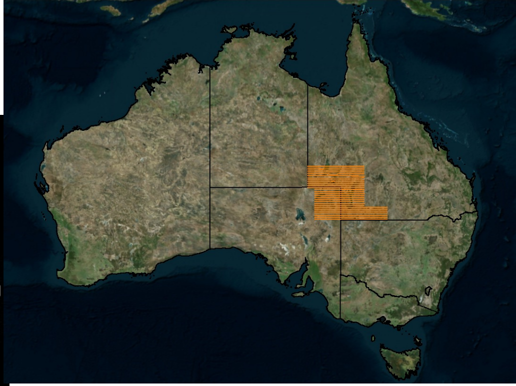
Compare: P10, median and P90 section displays



Far more interpretable information in percentiles

Go big: GA-LEI Occam inversions for AusAEM



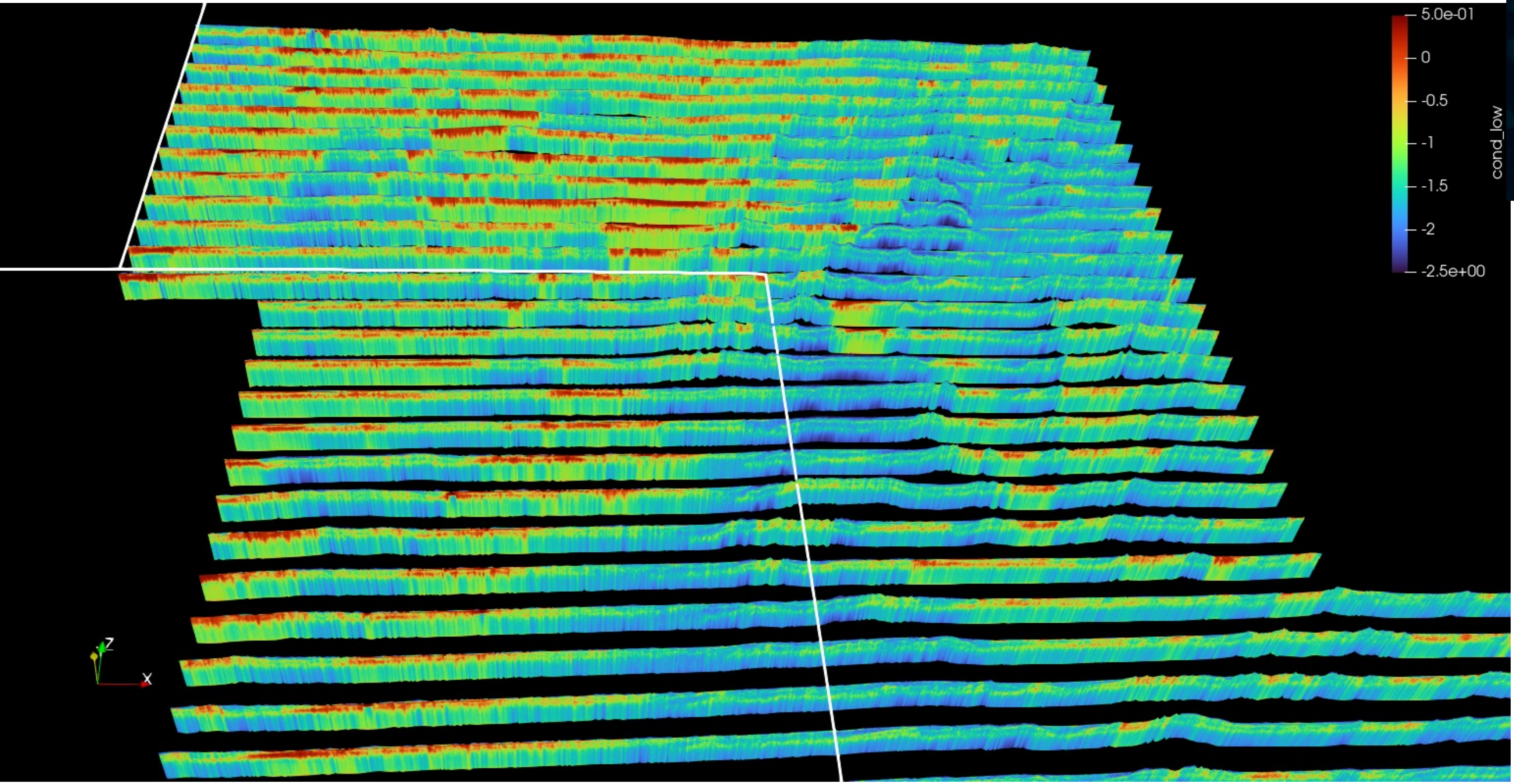


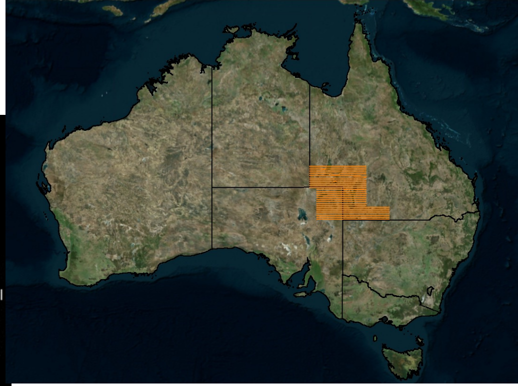
Deterministic inversion goes back to reference model at depth

40X Vert. Exagg.
Max depth ~350 m

20 km spaced continental scale AEM data https://dx.doi.org/10.26186/145744
Lines are approx. **500 km** long

Go big: Compare P10 section

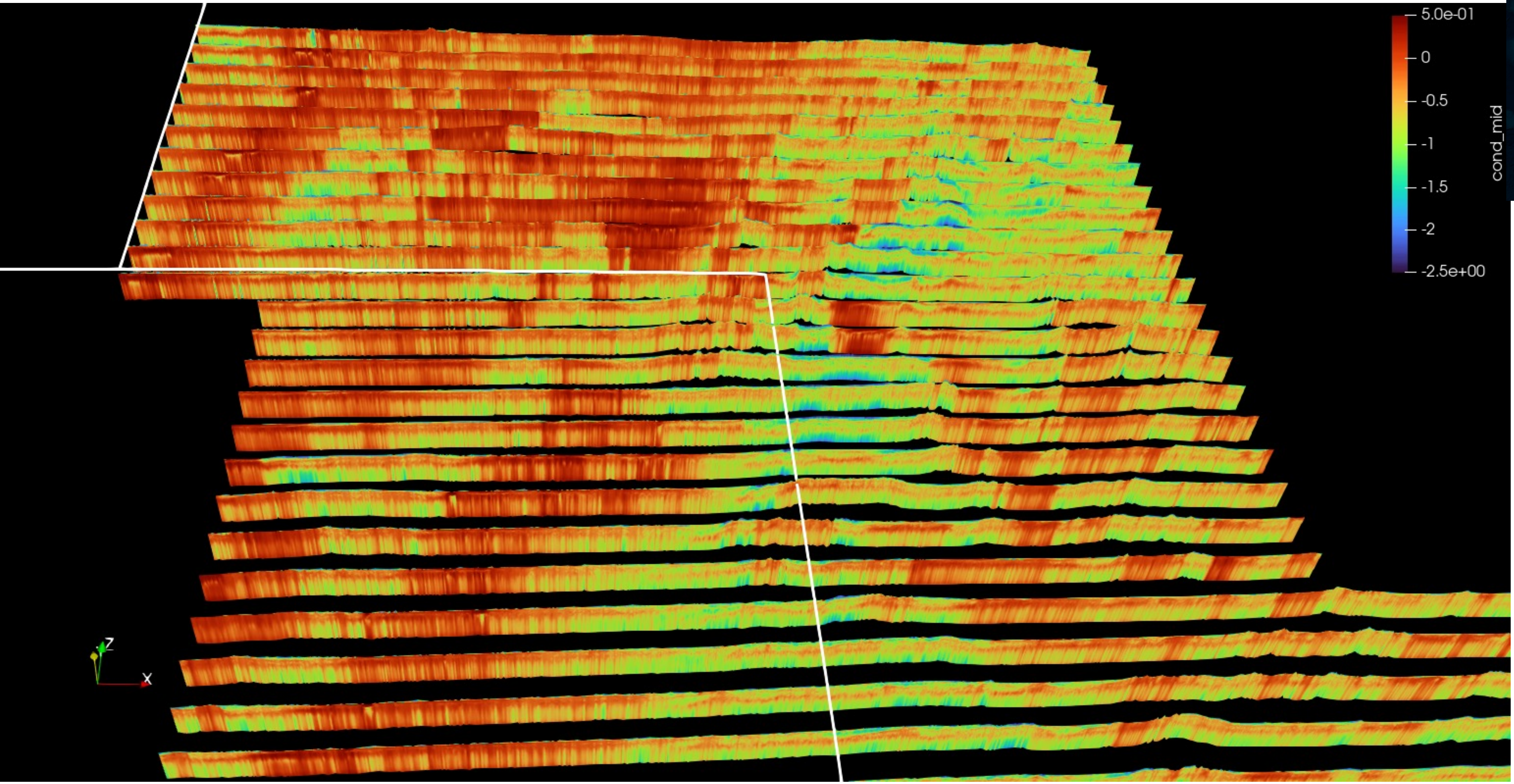


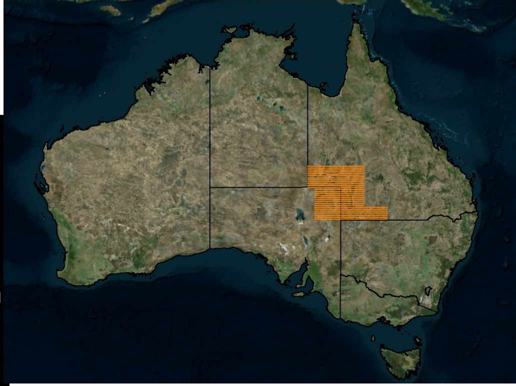


40X Vert. Exagg.
Max depth ~350 m

High probability ***conductors*** stand out

Go big: Compare median section

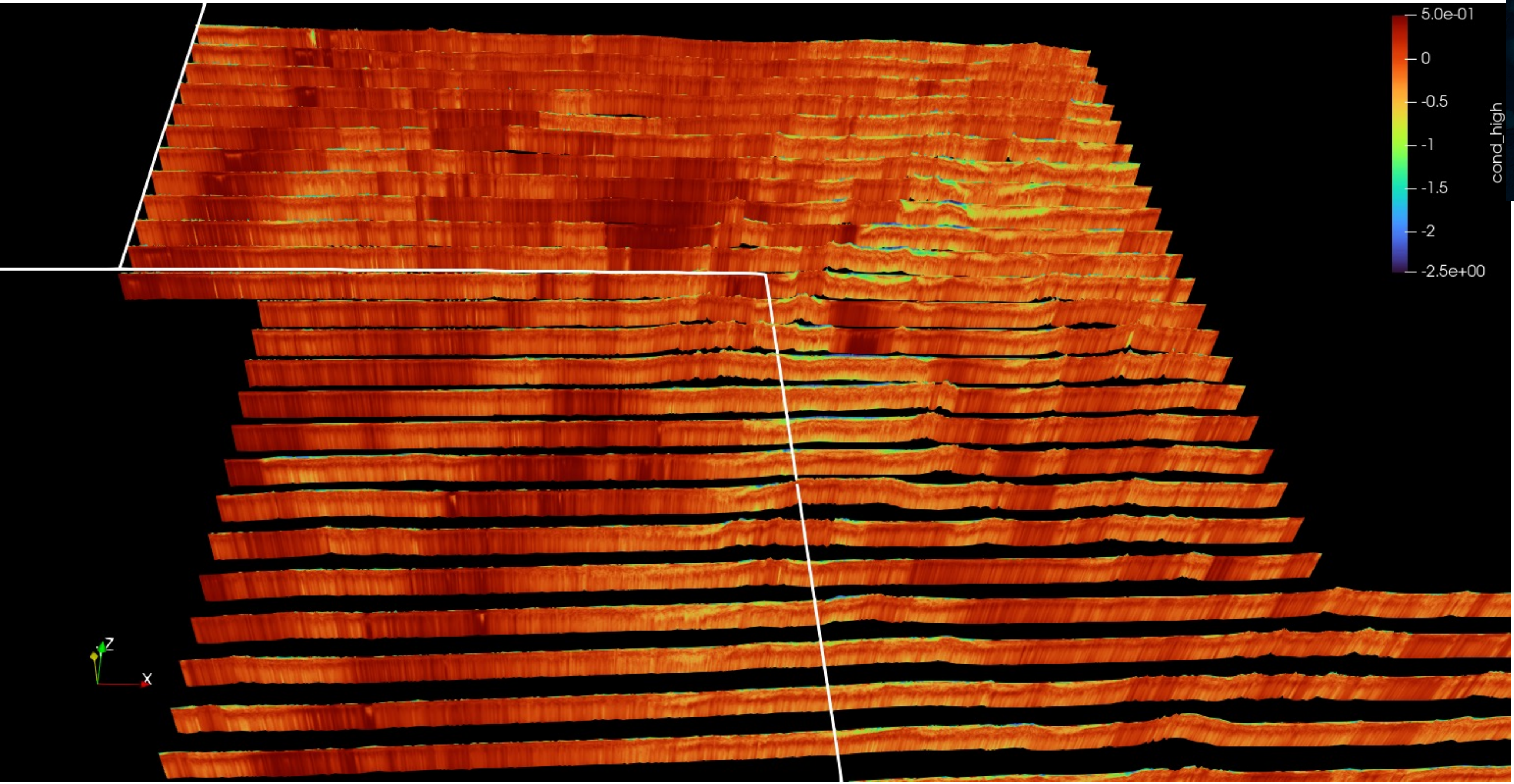


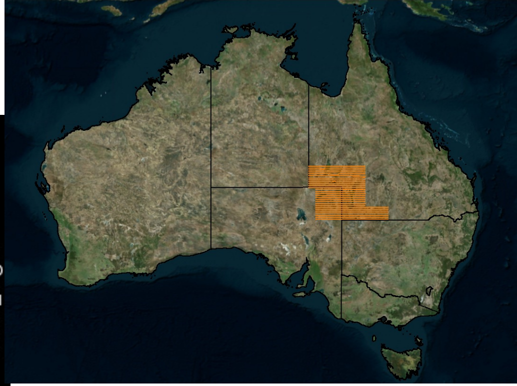


40X Vert. Exagg.
Max depth ~350 m

Median features

Go big: Compare P90 section

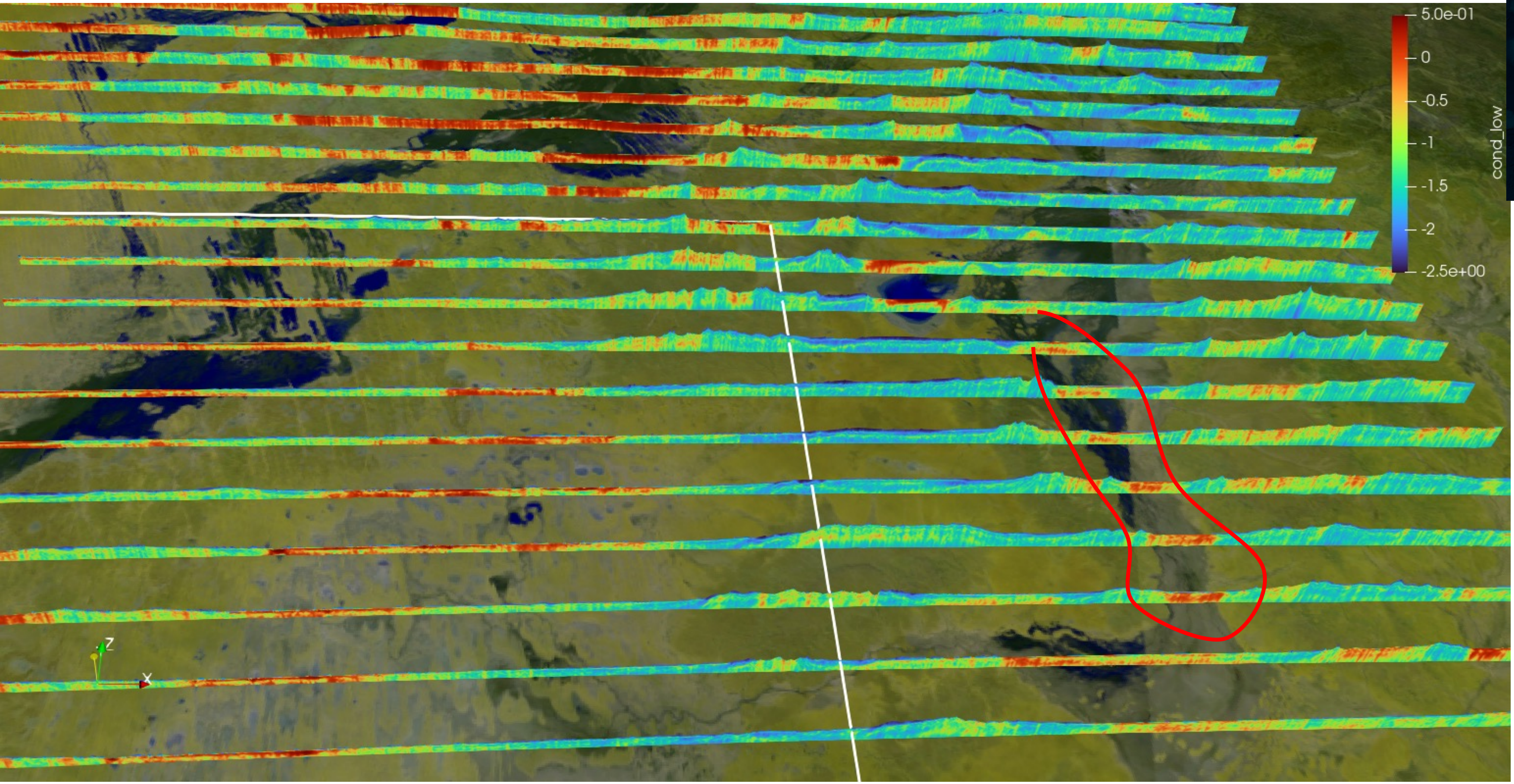


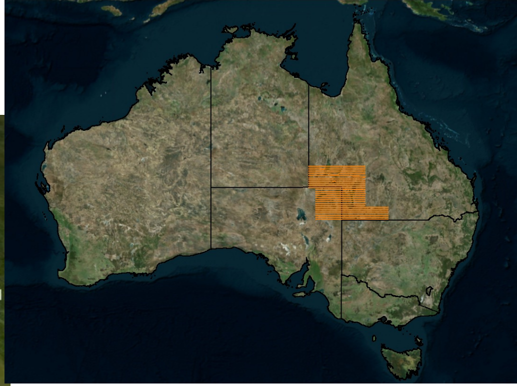


40X Vert. Exagg.
Max depth ~350 m

High probability ***resistors*** stand out

P10 section: Conductors line up to show saline groundwater

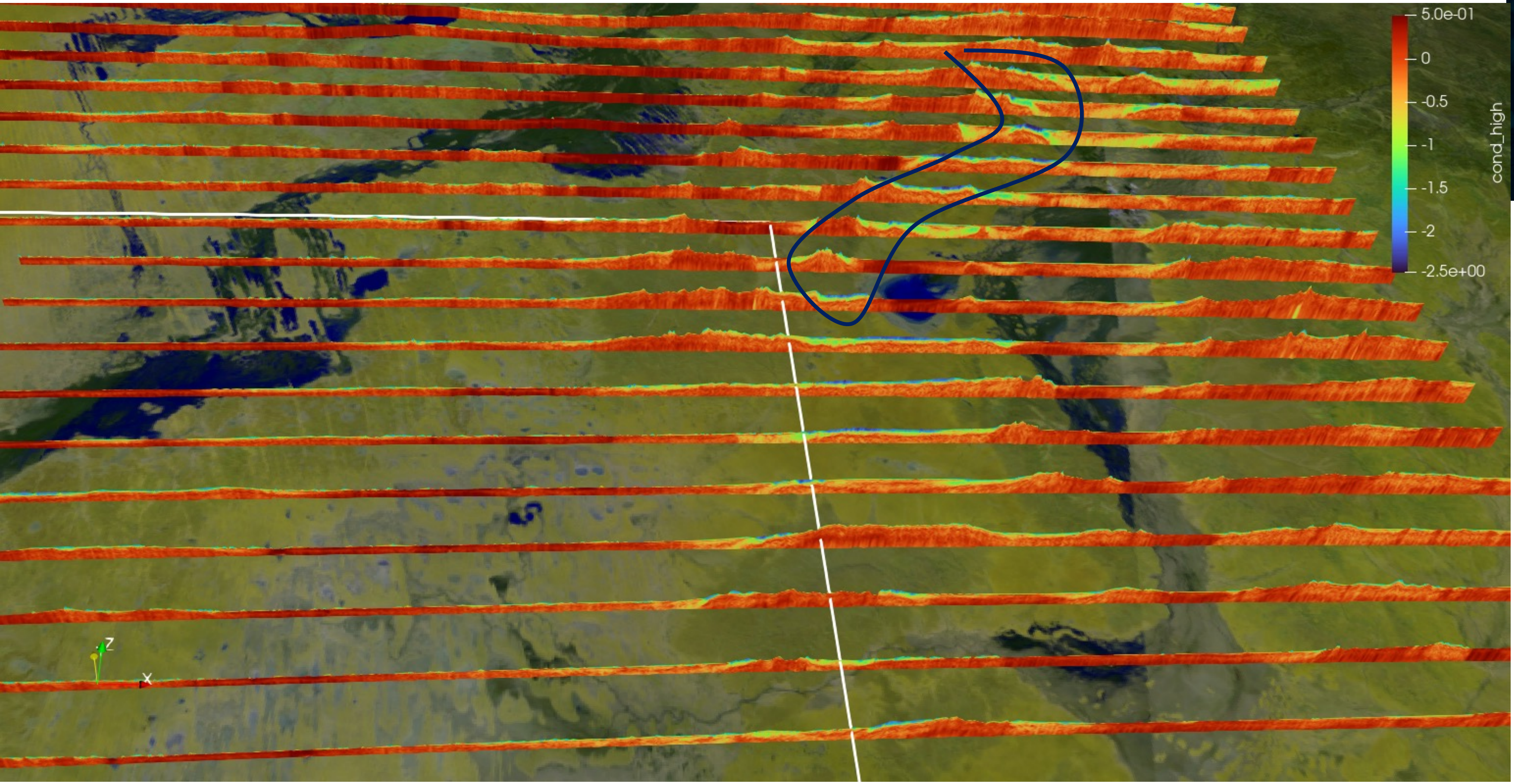


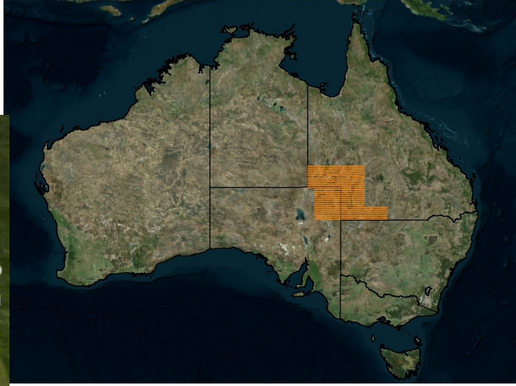


Landsat imagery in the background

Reds: 90% probability mass at ***conducting*** end

P90 section: Resistors line up near freshwater zones

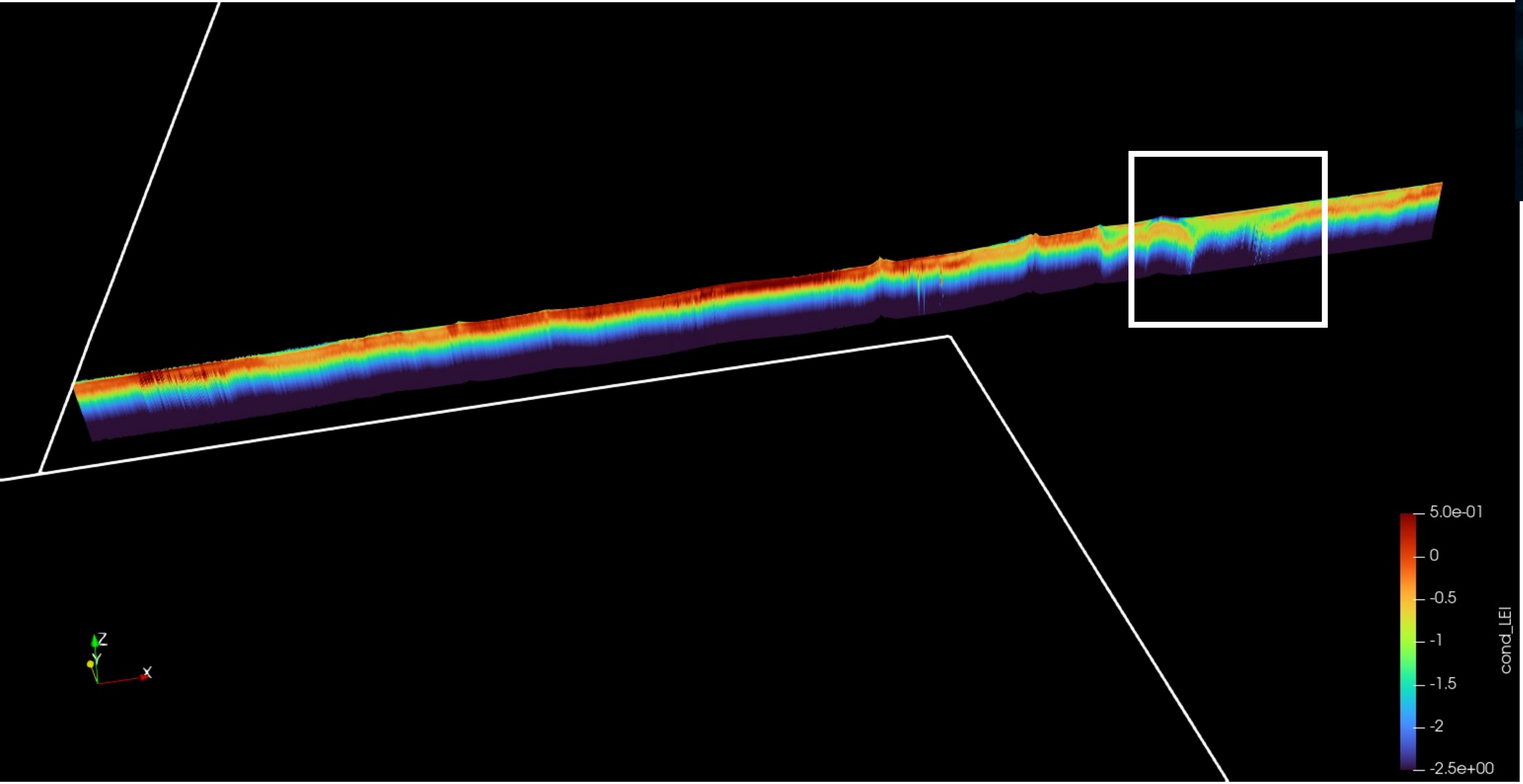


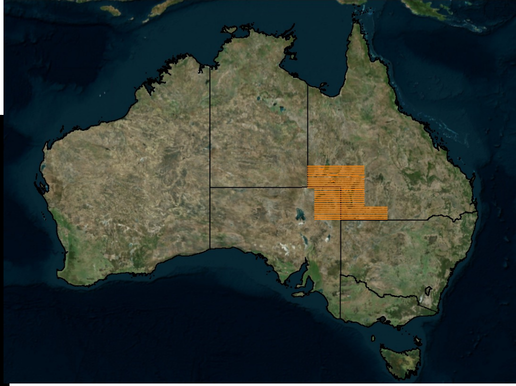


Landsat imagery in the background

Blues: 90% probability mass at ***resistive*** end

Identify ambiguous structure: GA-LEI (Occam)

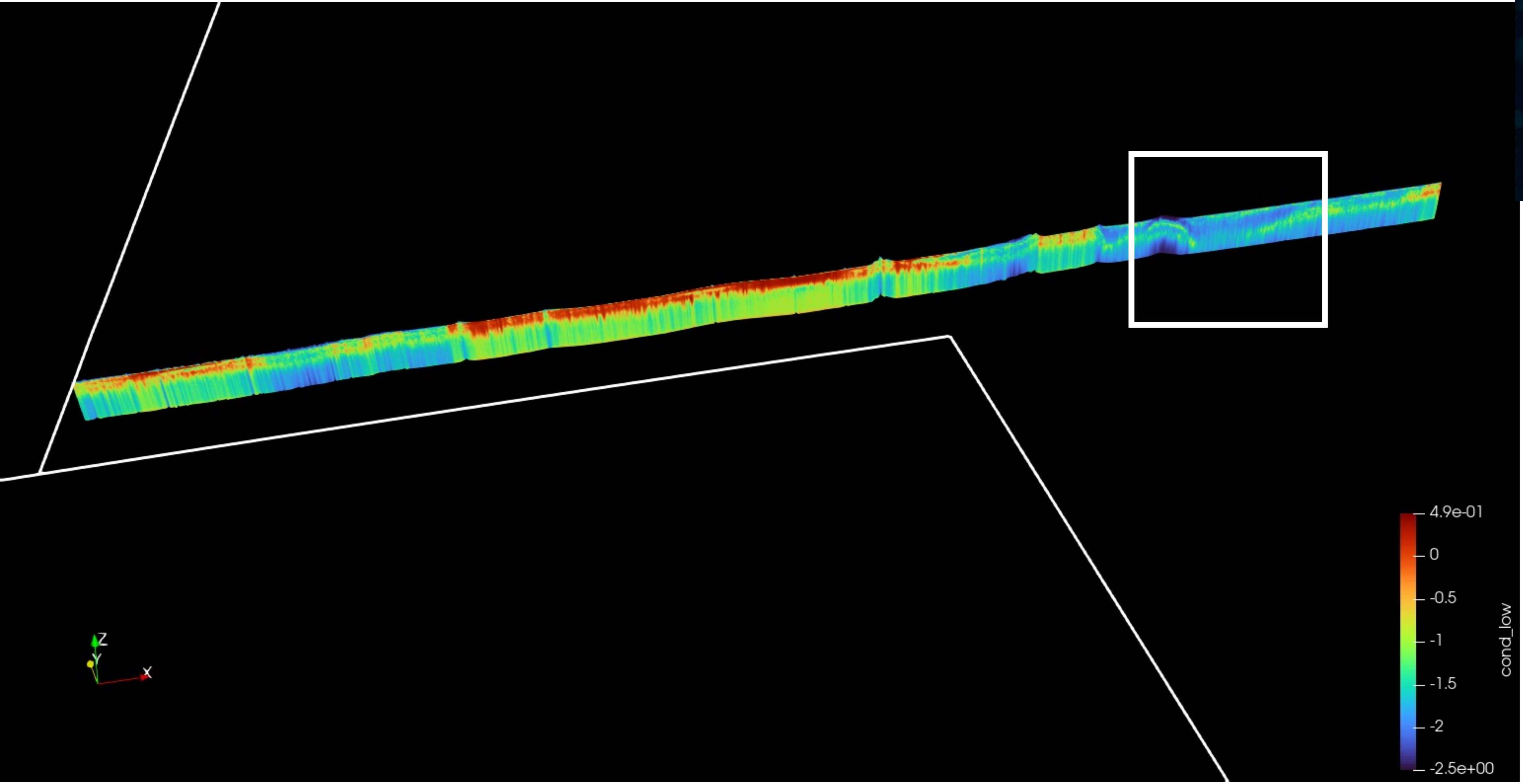


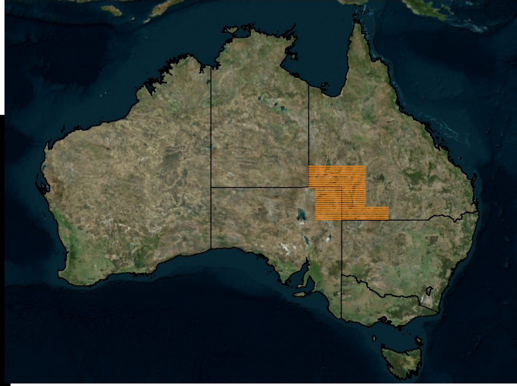


Deterministic inversion goes back to reference model at depth

2 incised palaeovalleys?

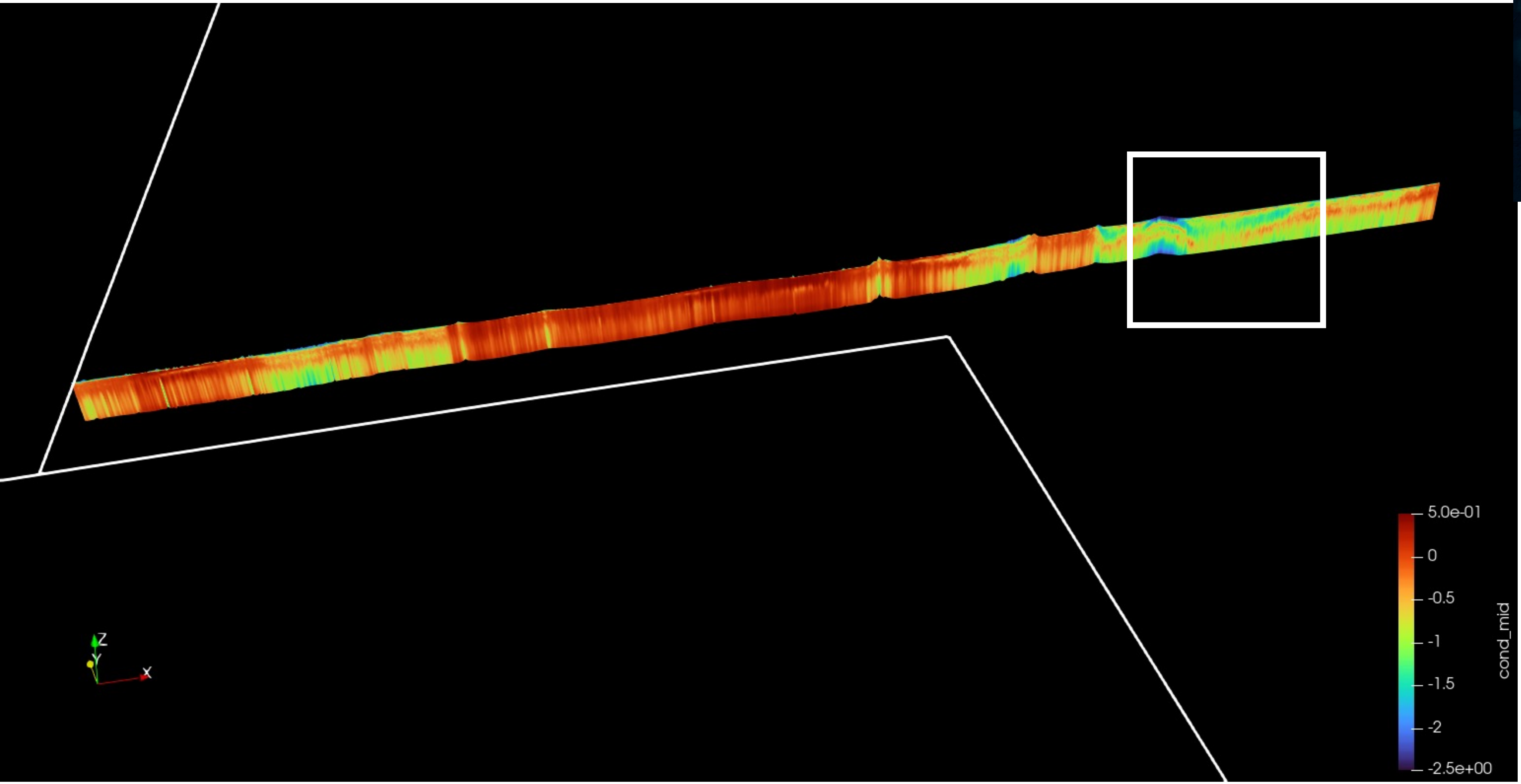
Identify ambiguous structure: P10 section

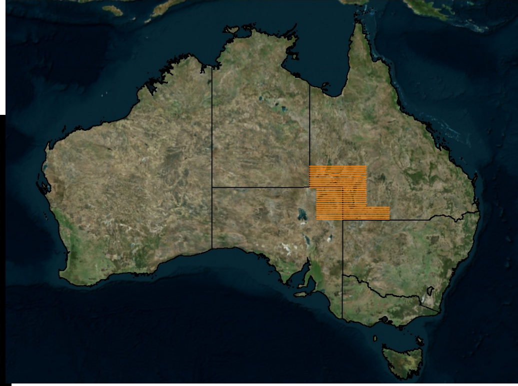




Maybe not

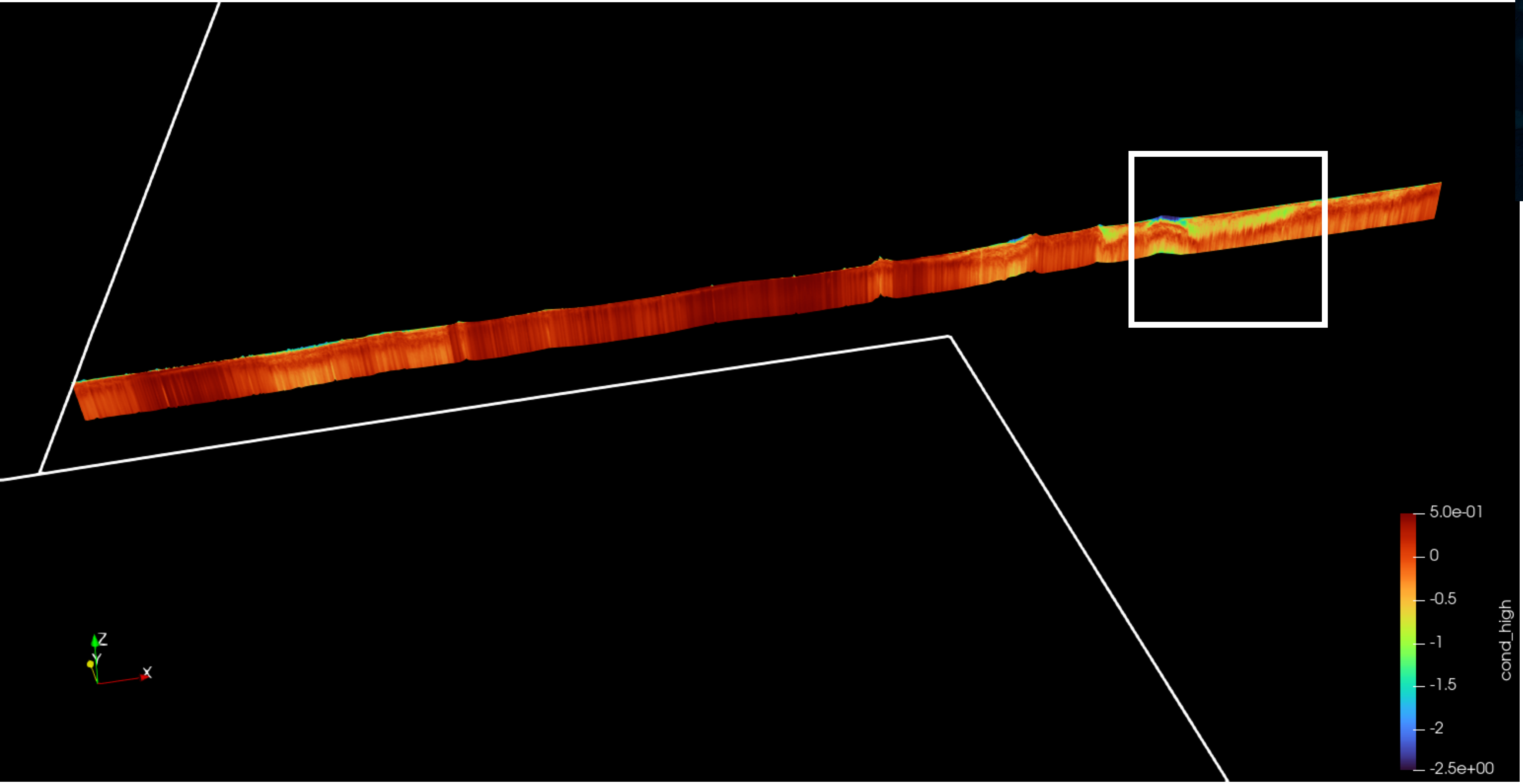
Identify ambiguous structure: median section

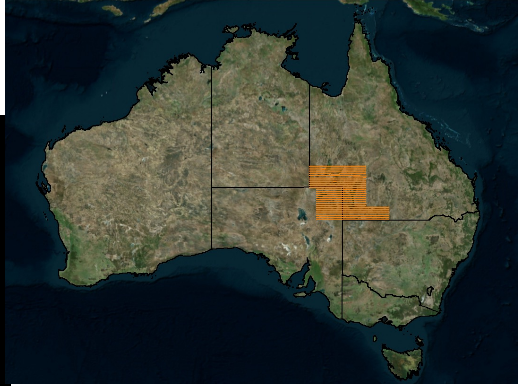




Looking more like not

Identify ambiguous structure: P90 section





HPC requirements for one line (500 km) with 1039 soundings

- 8320 cpus
- 04:30:00 hours

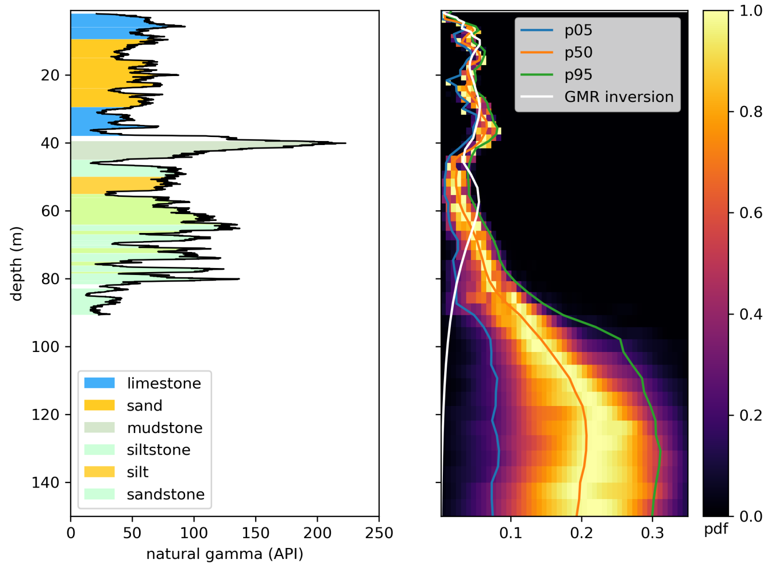
5 lines inverted simultaneously with McMC: 41,600 cpus

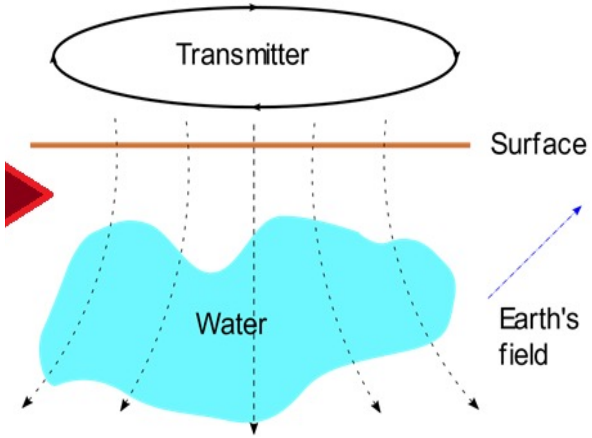
One synclinal palaeovalley!

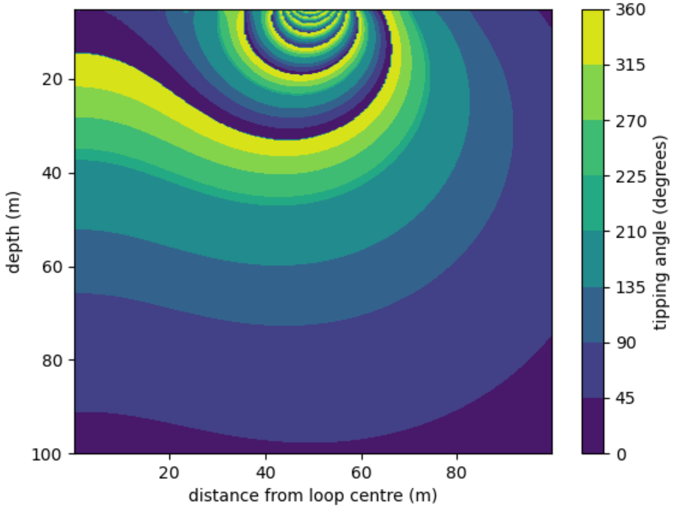
Different geophysics problems, same Julia interface

Surface Magnetic Resonance (SMR) imaging

```
pkg> add SMRPInversion
```

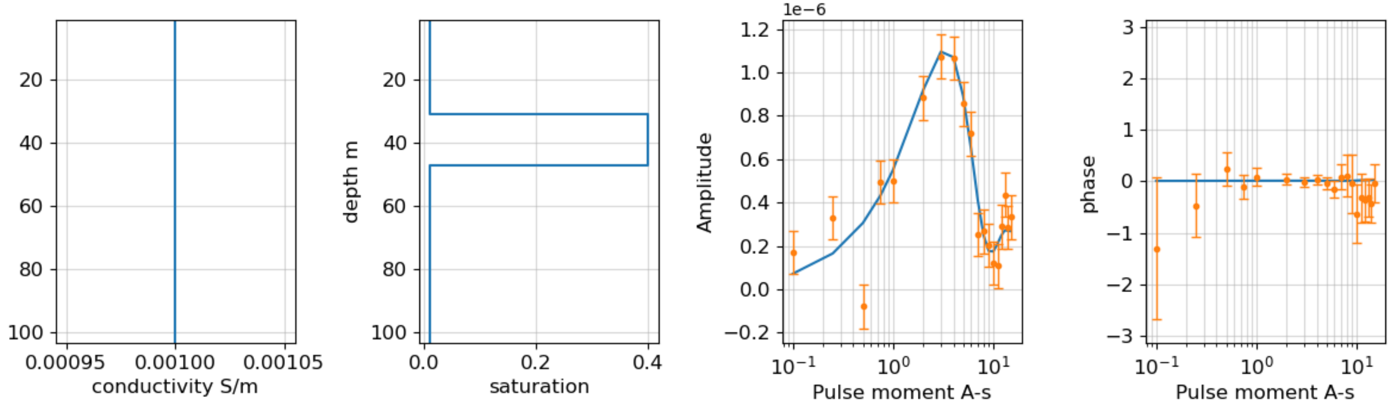






Julia Subtypes <: Different forward problems, same McMC interface

```
julia> typeof(sounding)
SMRPInversion.SMRSoundingKnown

julia> typeof(sounding)<:transD_GP.Operator
true
```

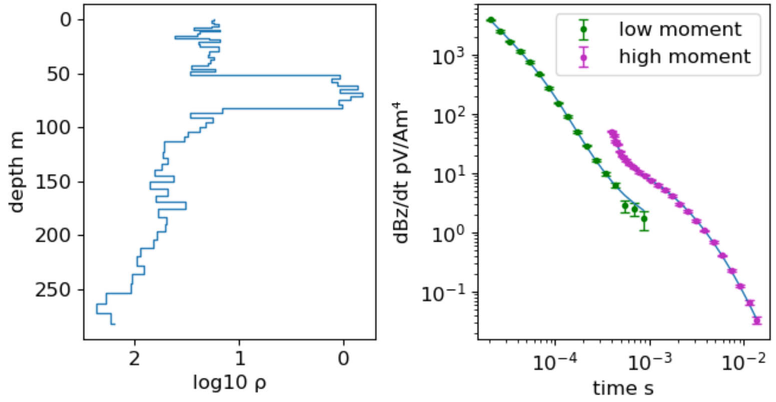



```
transD_GP.main(opt, sounding, Tmax=Tmax, nsamples=nsamples, nchains=nchains, nchainsatone=nchainsatone)
```

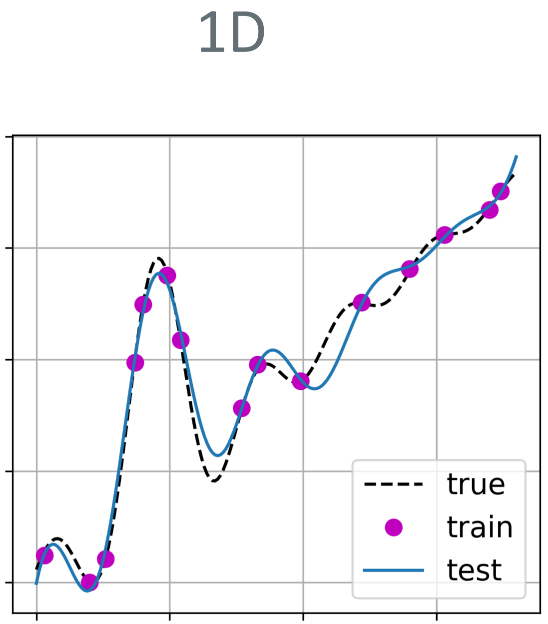
Different physics operator, same McMC method

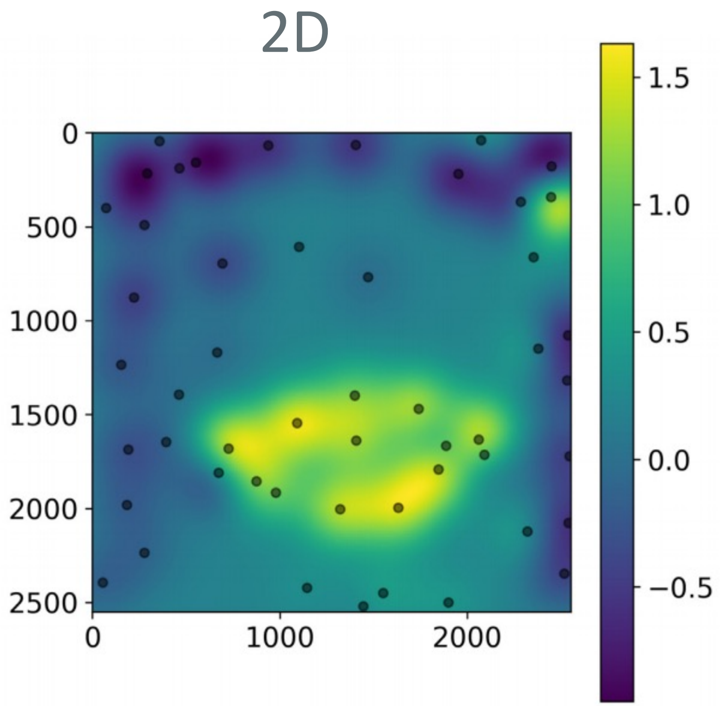
```
transD_GP.main(opt, aem, Tmax=Tmax, nsamples=nsamples, nchains=nchains, nchainsatone=nchainsatone)
```

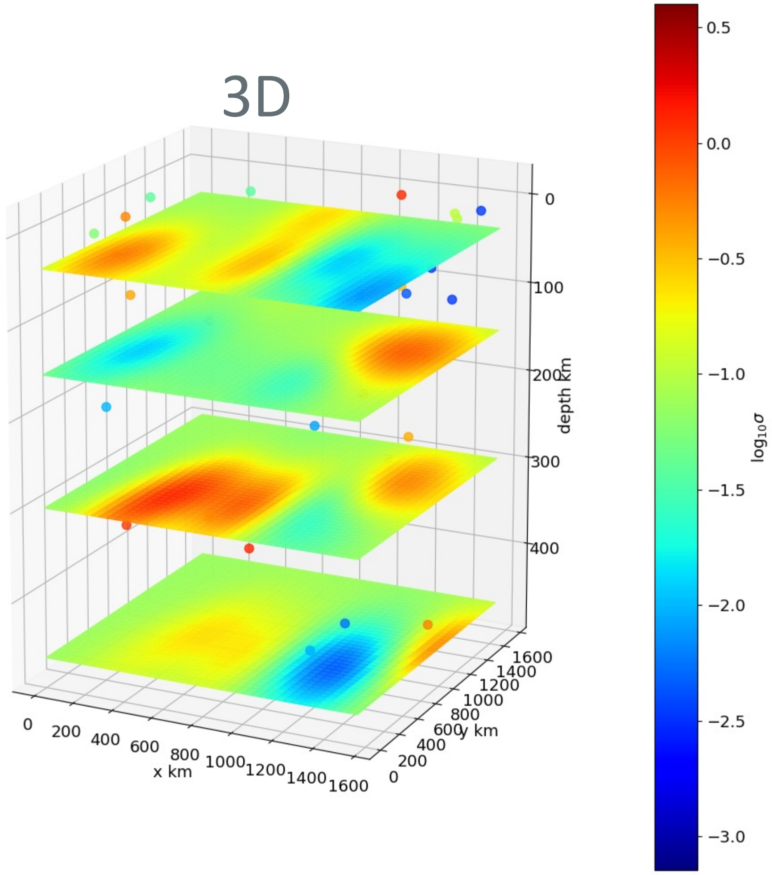
```
julia> typeof(aem)
HiQGA.transD_GP.SkyTEM1DInversion.dBzdt

julia> typeof(aem)<:transD_GP.Operator
true
```



Under the McMC hood: Represent n_d earth properties with *same* equation







$$\mu_* = \mathbf{K}_*\mathbf{K}_m^{-1}\mathbf{m}$$

Full model vector
$n \times 1$ **(LARGE)**

"Cross" kernel matrix
$n \times r$

"Self" kernel matrix
$r \times r$

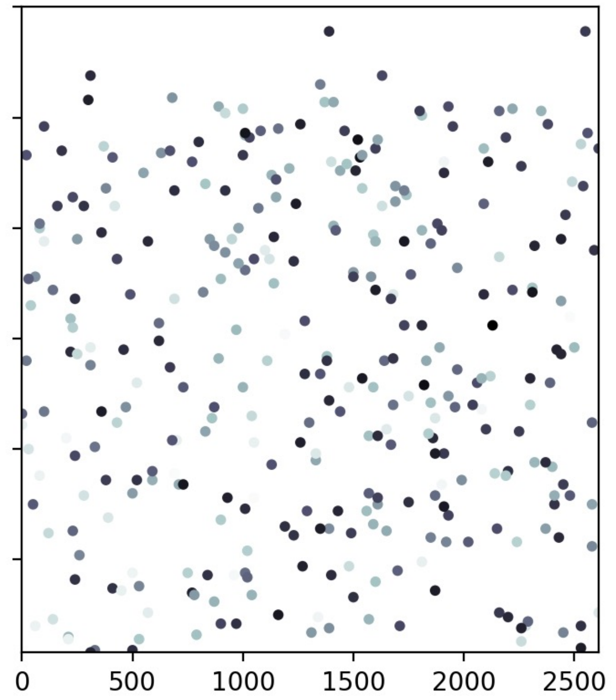
GP nuclei model vector
$r \times 1$ **(small)**

$$K(\mathbf{y}, \mathbf{y}') = \exp\left(-\frac{1}{2}[\mathbf{y}-\mathbf{y}']^t \mathbf{C}_\lambda{}^{-1}[\mathbf{y}-\mathbf{y}']\right), \text{ where } \mathbf{y} \in \mathbb{R}^{n_d}$$

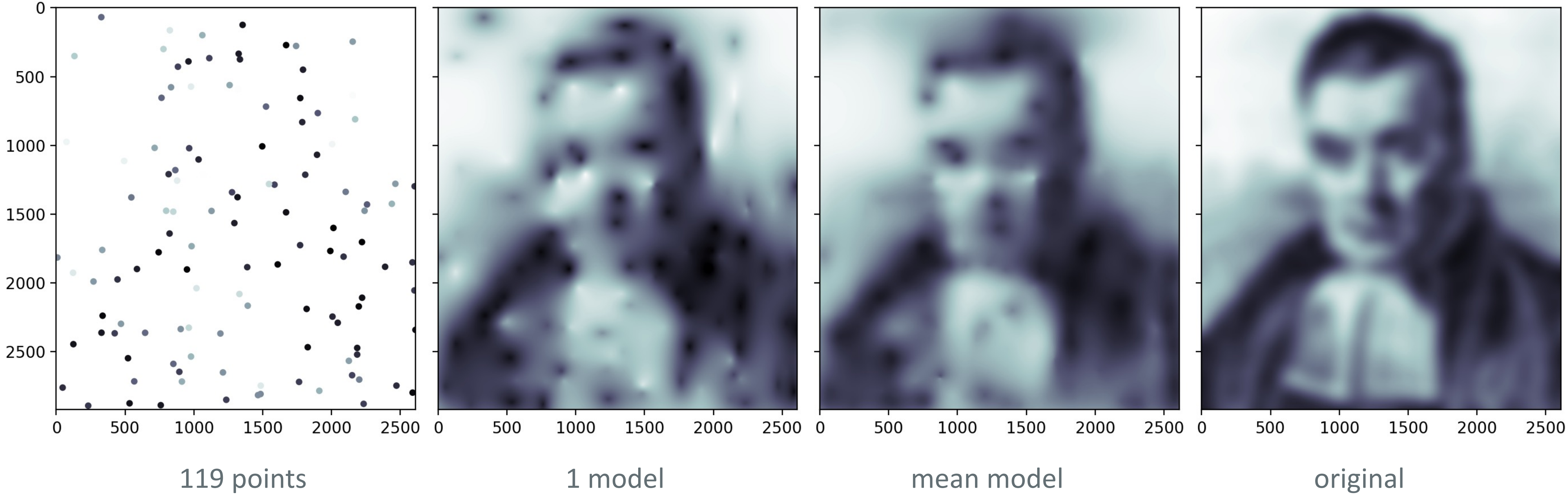
Rasmussen & Williams (2006)
Now we can use these *r* points to do Bayesian Trans-D McMC

Regress images with McMC

I’m sure you recognize what you’re looking at – these are 332 points sampled from a 293x262 image

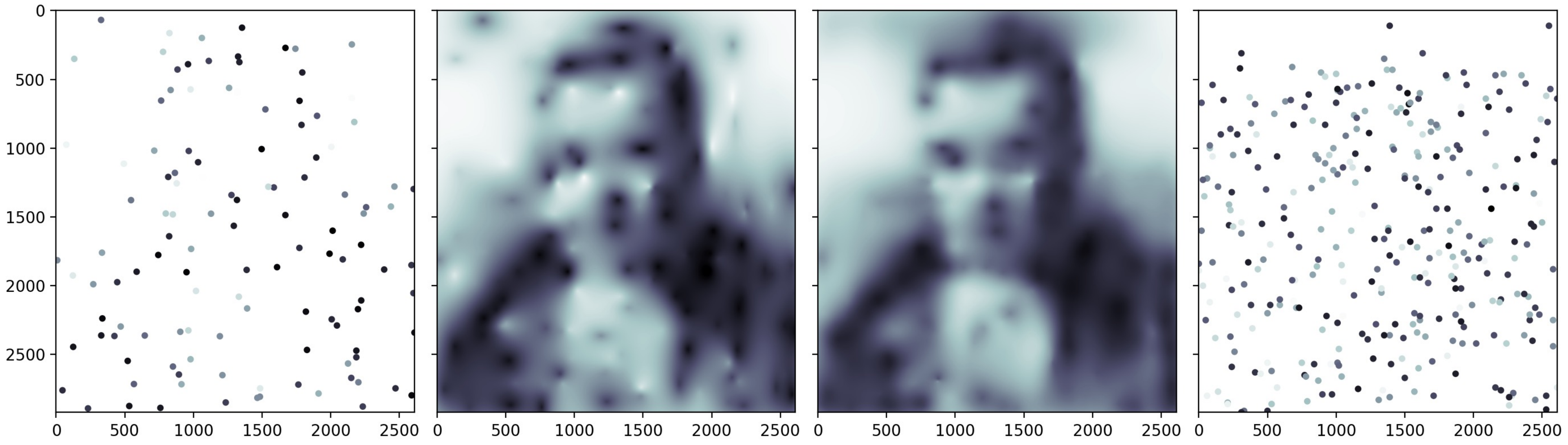


Reconstructing an image using deep GP parameterisations (2-layer)



HiQGA.jl / examples / 2D / image_revBayes /

But we had started here …



119 points

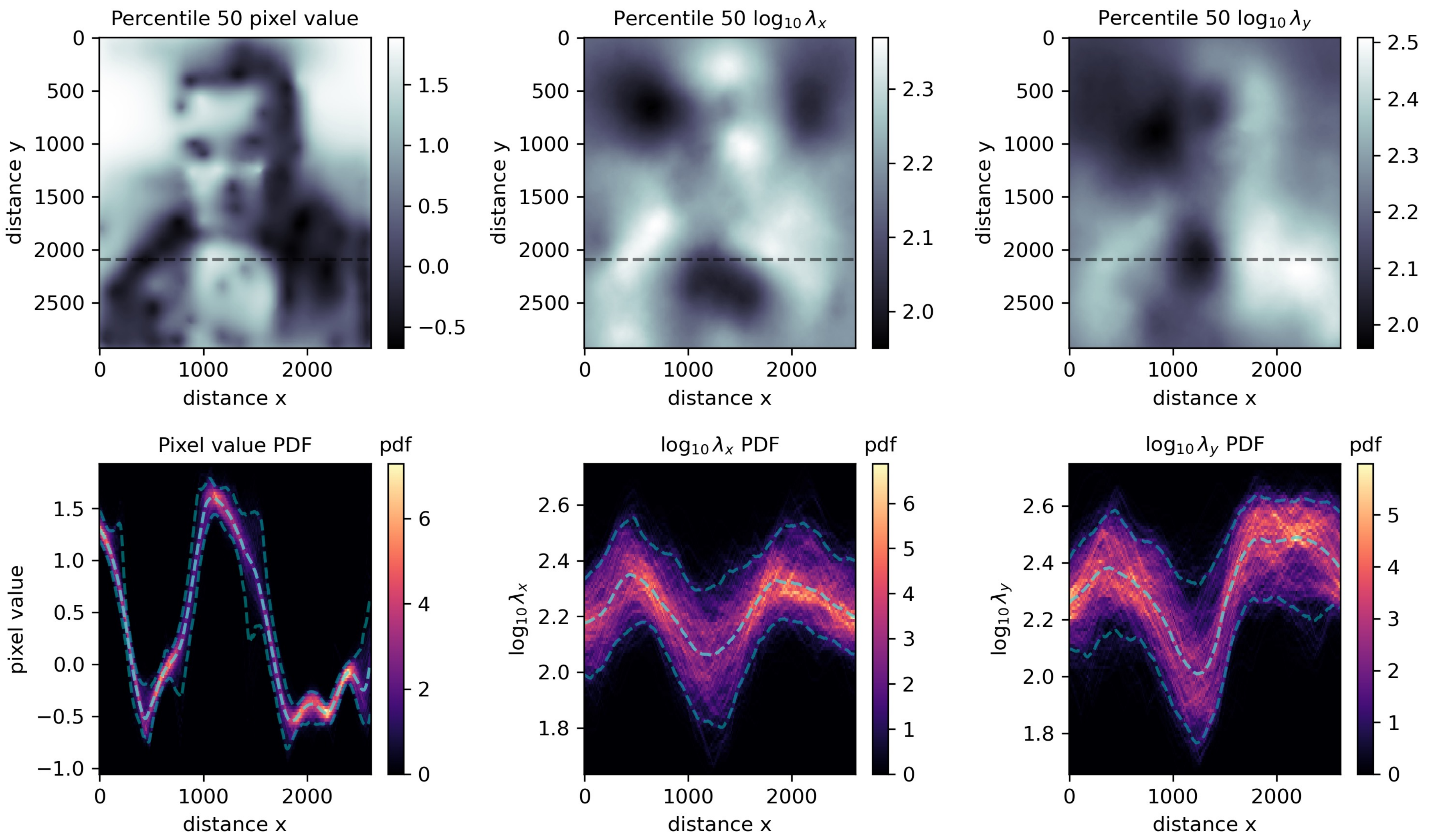
1 model

mean model

332 points only

HiQGA.jl / examples / 2D / image_revBayes /

Using a 2-layer GP



With 200-300 GP nuclei, we can represent a 293x262 image = 76,766 pixels – a compression of ~300X

Geophysical Journal International, 2019, 2021

Bayesian inversion using nested trans-dimensional Gaussian processes

Anandaroop Ray

https://academic.oup.com/gji/article/226/1/302/6189704

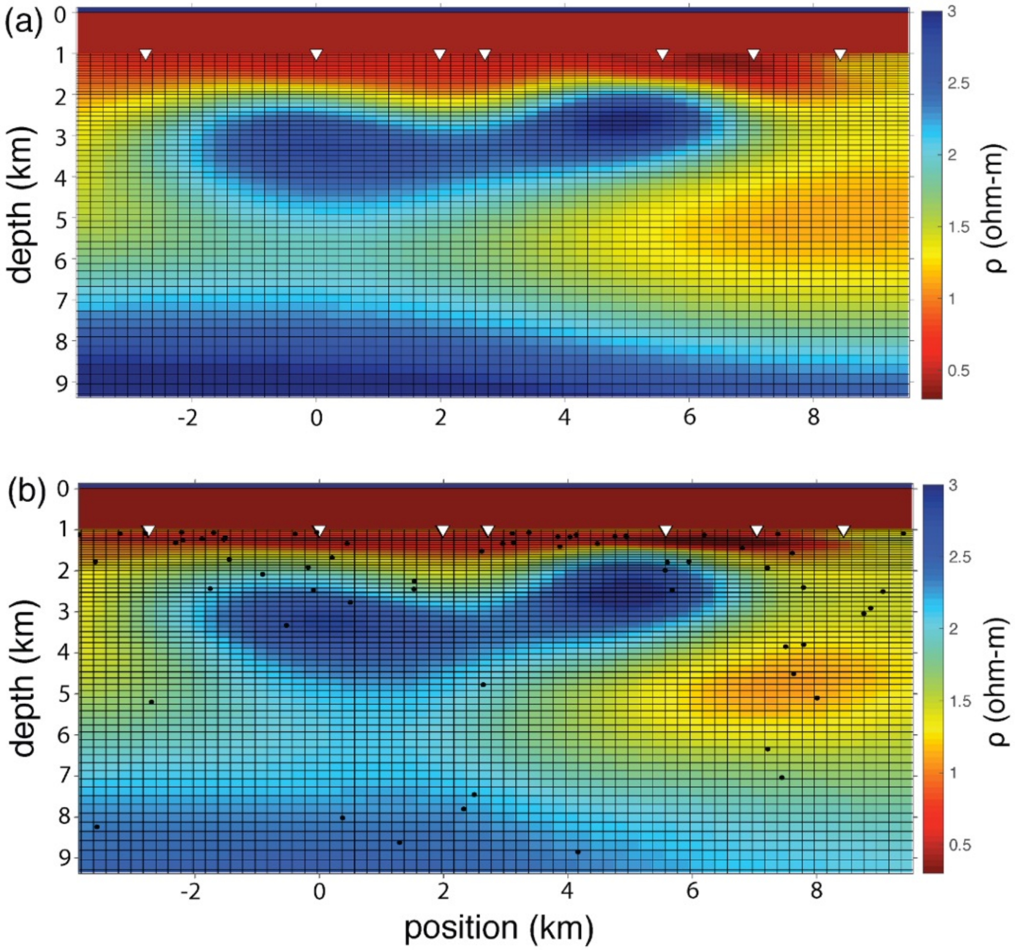
Bayesian geophysical inversion with trans-dimensional Gaussian process machine learning

Anandaroop Ray[1] and David Myer[2]

https://academic.oup.com/gji/article/217/3/1706/5366736

2D marine MT with a 1-layer GP parameterization, 2-layer would be better!

MARE/Occam

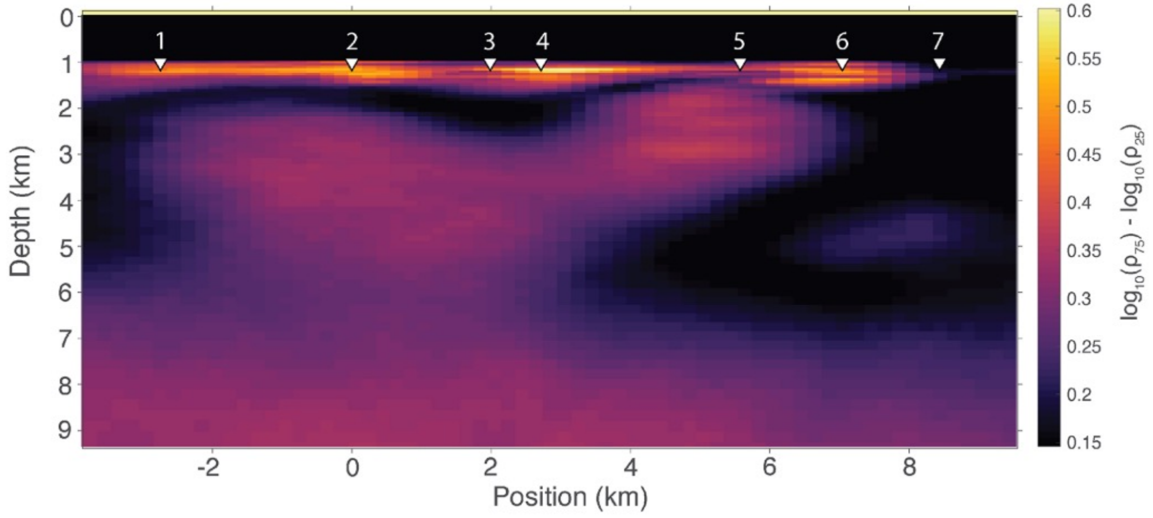


TransD-GP

https://academic.oup.com/gji/article-abstract/226/1/548/6188387

- 168 processors, 10 days, 1_000_000 samples
- 0.85 s per forward
- 10 frequencies, 7 sites
- 8424 cell inversion model
- Native *Julia* on Columbia University's Habanero cluster

TransD-GP uncertainty

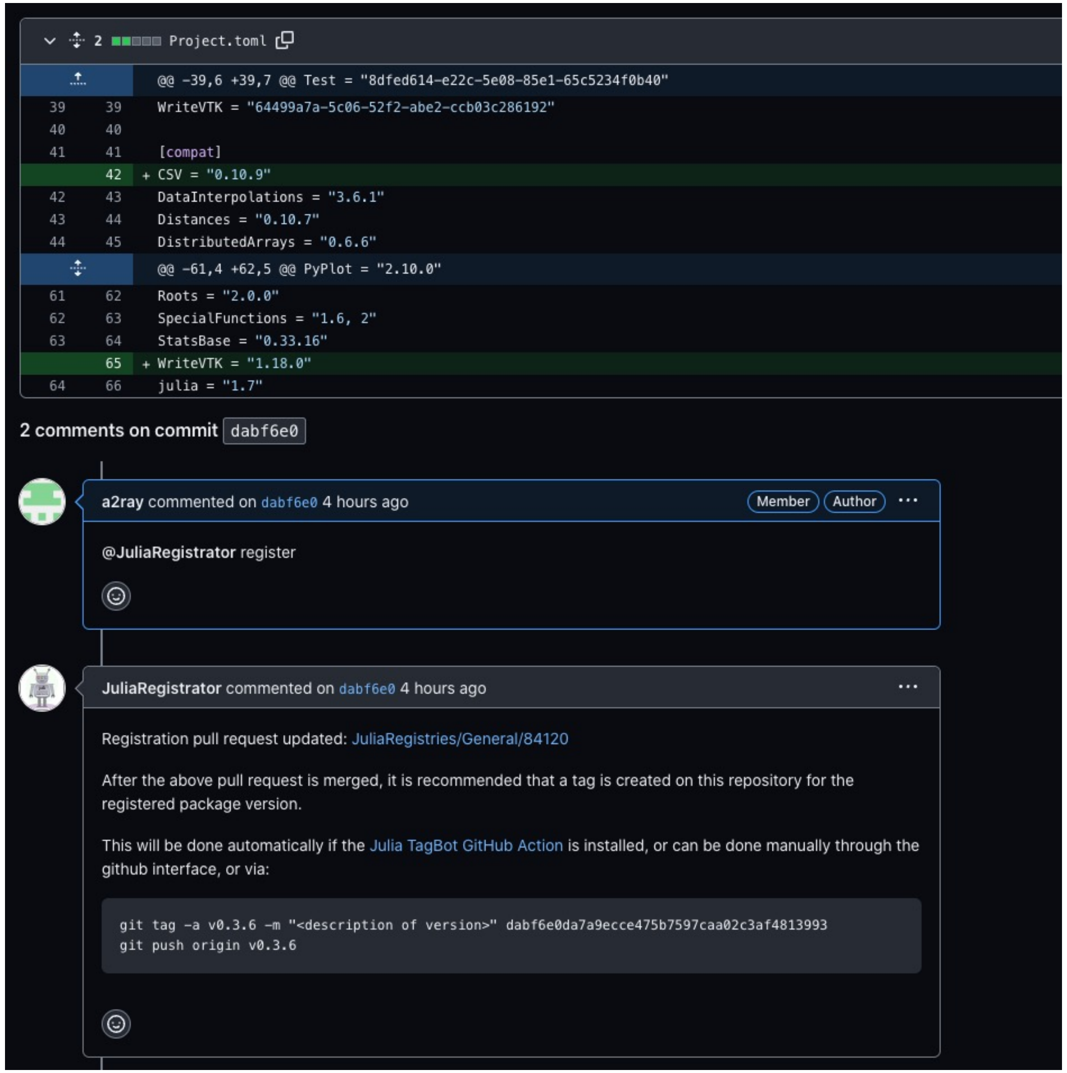


Geophysical Journal International, 2021

Two-dimensional Bayesian inversion of magnetotelluric data using trans-dimensional Gaussian processes

Daniel Blatter[1], Anandaroop Ray[2] and Kerry Key[1]

Last and often ignored: Distribute your code

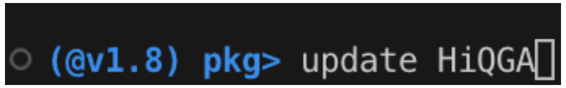


Package management in Julia

Make your changes

Invoke the Julia **Registrator** bot on GitHub

Wait for the pull request to complete
Users can then do:



To conclude

- Occam inversion models have **low entropy**
- Many geophysics priors should generally **encourage low entropy**
- Bayesian posteriors encourage **rapid**, **probabilistic interpretation** of geology
- A general Julia **inversion framework** with these ideas are at:
  - https://github.com/GeoscienceAustralia/HiQGA.jl
- Julia's type hierarchy makes it easy to ***dispatch*** generic optimizers or samplers to the right physics type
- Julia is just in time **compiled** and ***fast …*** see https://julialang.org/benchmarks/
- Excellent numerical package **libraries** are available (FFTW, interpolations, Bessel etc.)
- Code **reads like math** and is easy to follow
- **Do it all in Julia**, no more Python prototyping → C++/Fortran/MPI production → Python visualization
- Avoid dealing with Makefiles et al. – **incremental recompilation** massively boosts productivity
- Julia is excellent for prototyping to production and **package distribution**

Get involved!

https://github.com/GeoscienceAustralia/HiQGA.jl

We welcome your contributions

Anandaroop.Ray@ga.gov.au

Backup slides

Uncorrelated posterior realizations are unsatisfactory

10 x 10 cell model

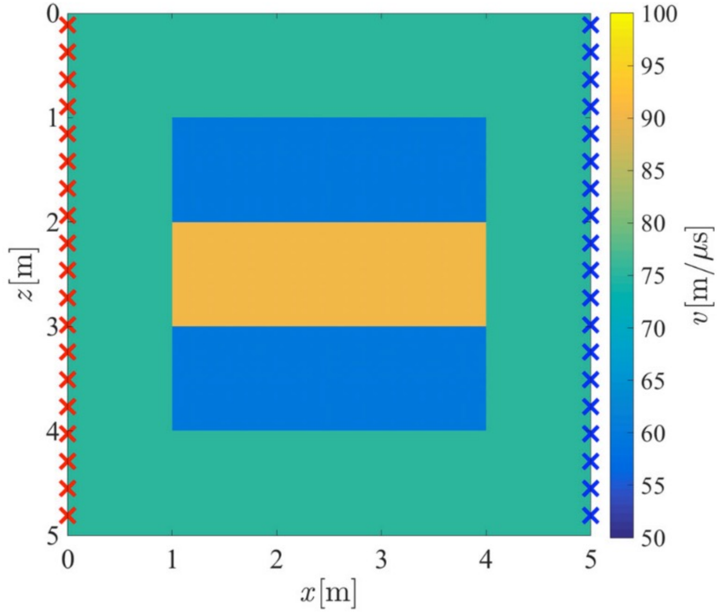
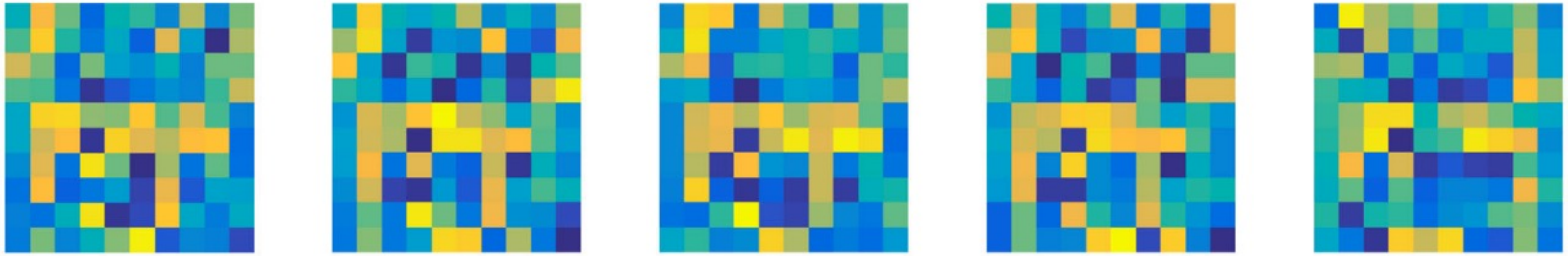
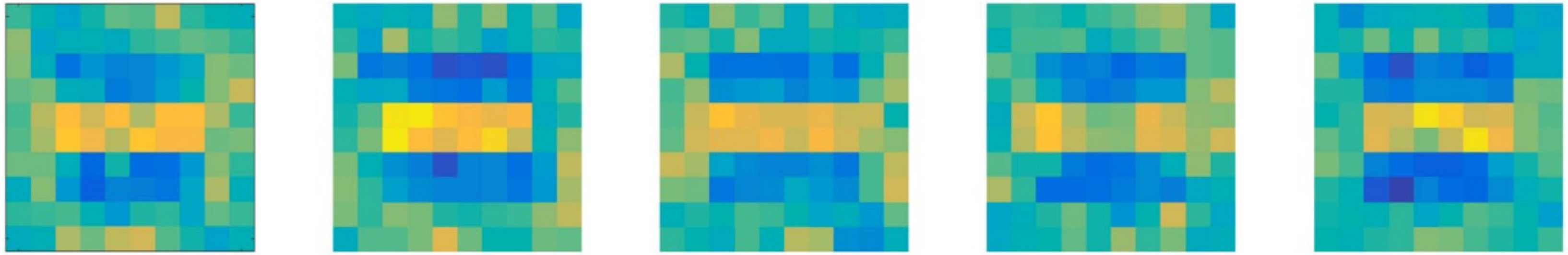


Figure 4. A simple synthetic test model of radar wave speed *v*. Red crosses represents the 19 equally spaced sources while the blue crosses represents the 19 equally spaced receivers for the crosshole GPR experiment.



Uncorrelated realizations



Structure constrained realizations

Geophysical Journal International, 2017

On structure-based priors in Bayesian geophysical inversion

G. de Pasquale and N. Linde

Why are effective parameterizations necessary?

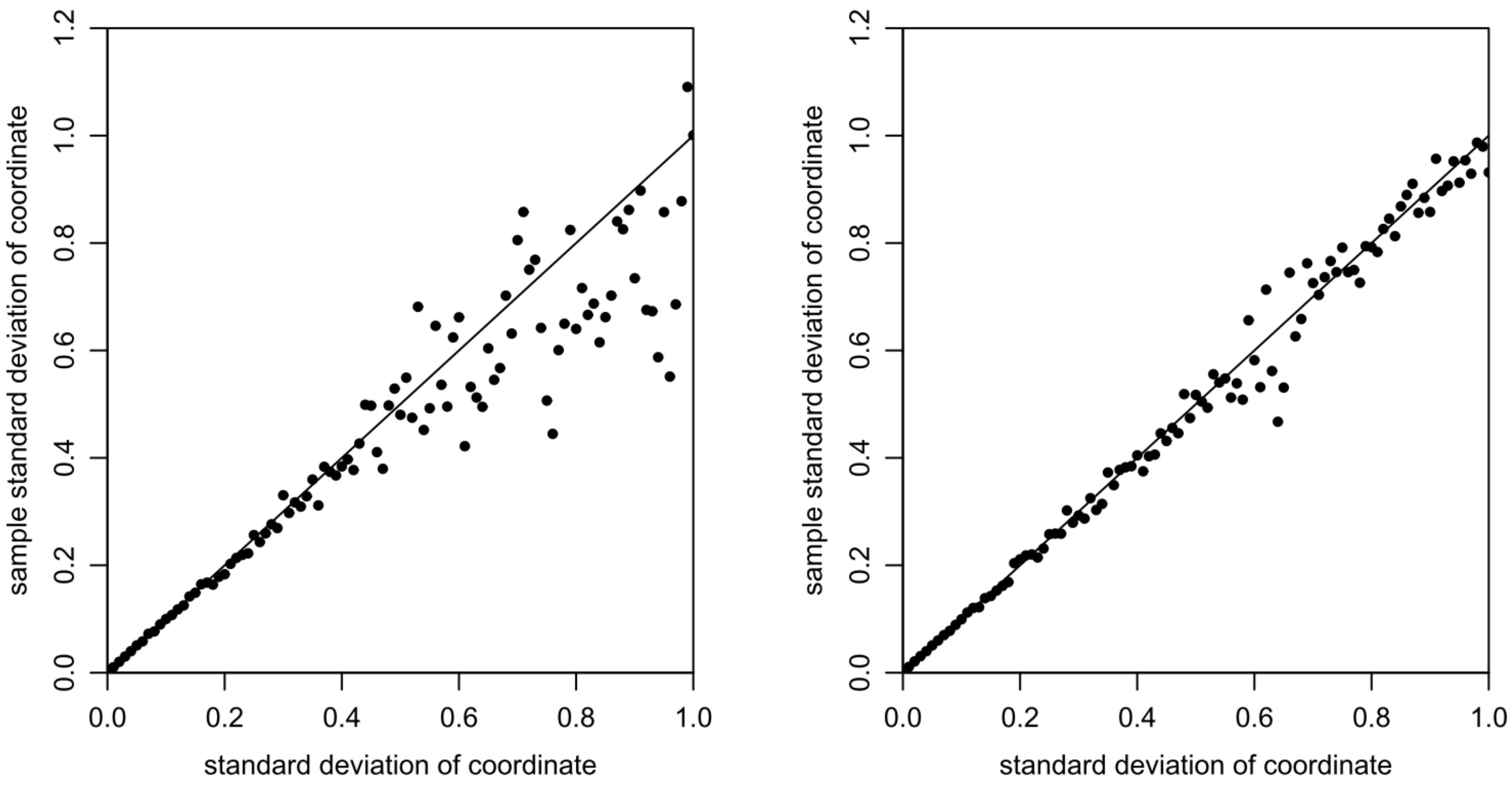


Figure 7: Estimates of means (top) and standard deviations (bottom) for the 100-dimensional example, using random-walk Metropolis (left) and HMC (right). The 100 variables are labelled on the horizontal axes by the true standard deviaton of that variable. Estimates are on the vertical axes.

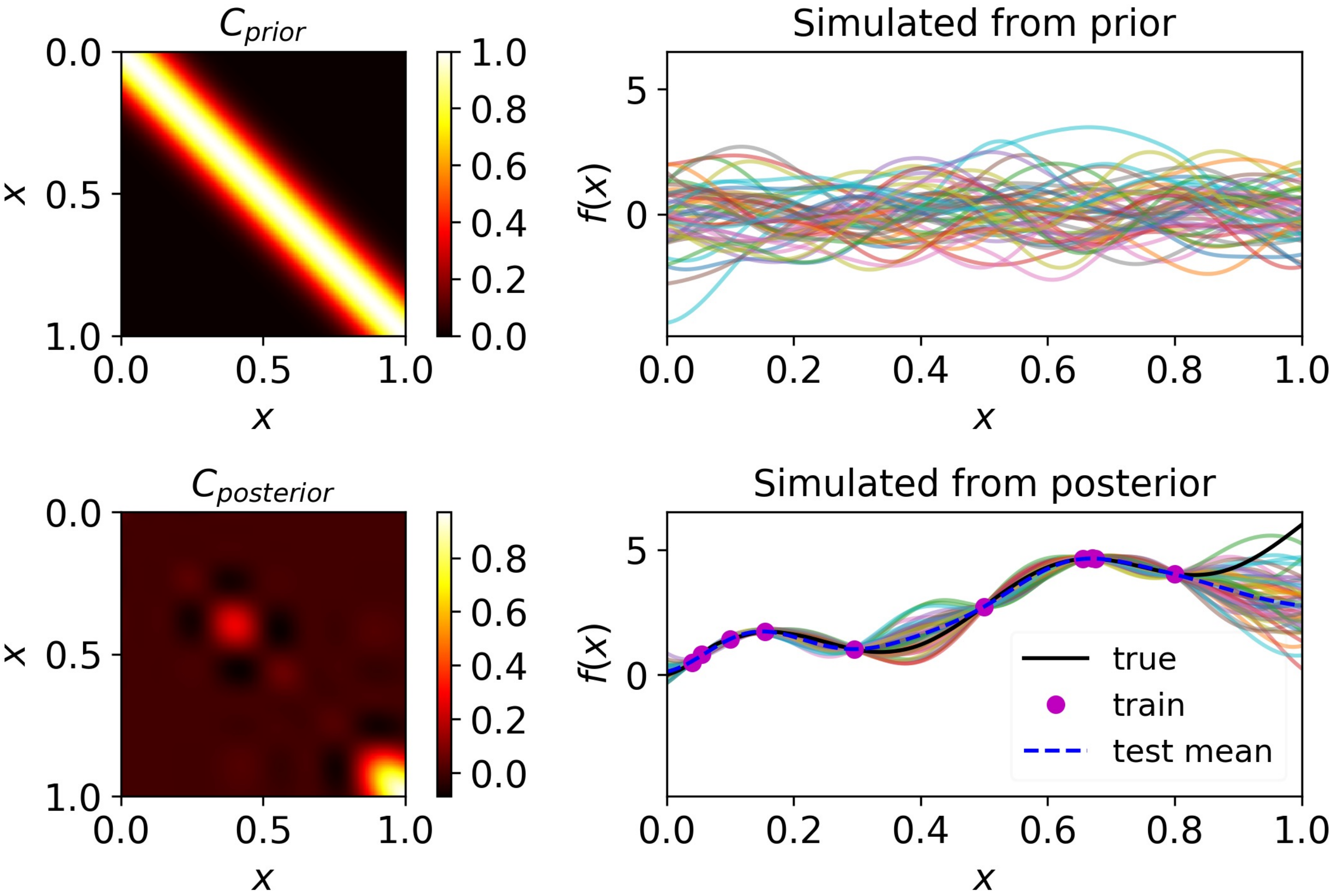
10^2 pixels is hard.
What about 10^5 ?

Handbook of McMC, 2011

MCMC using Hamiltonian dynamics

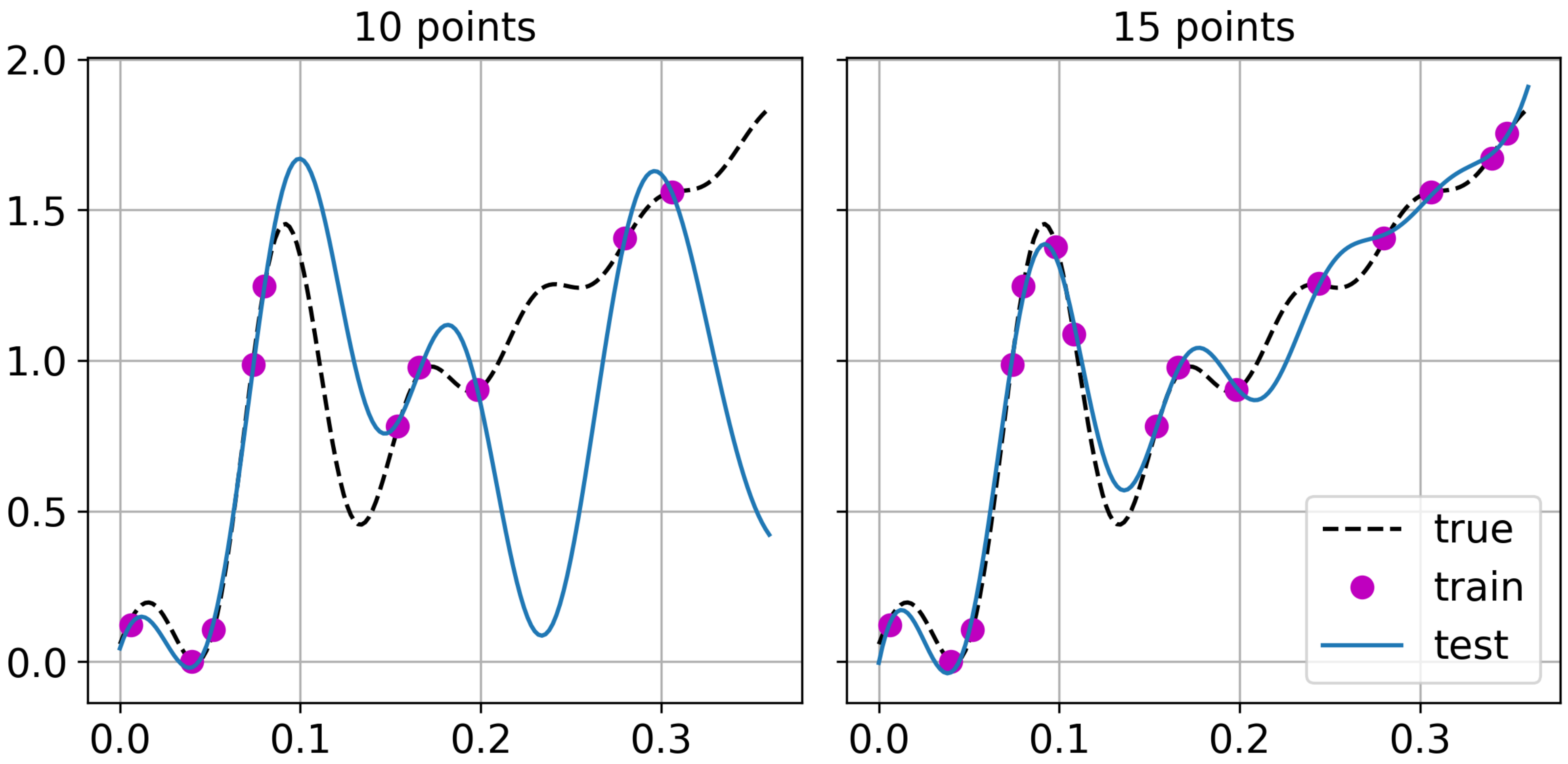
Radford M. Neal, University of Toronto

Gaussian Processes – naturally Bayesian



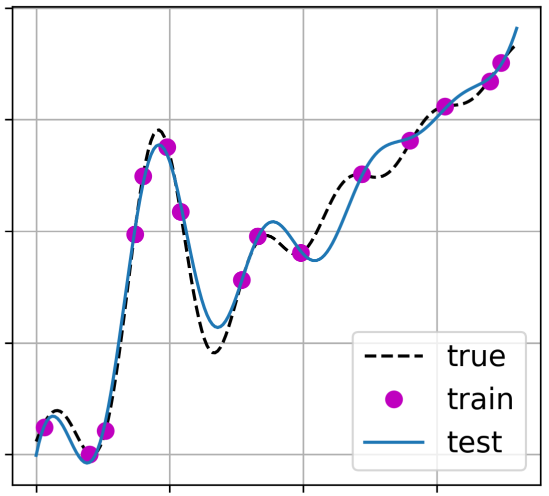
Rasmussen & Williams (2006)
Ray & Myer 2019

Fitting a function using a Gaussian process mean

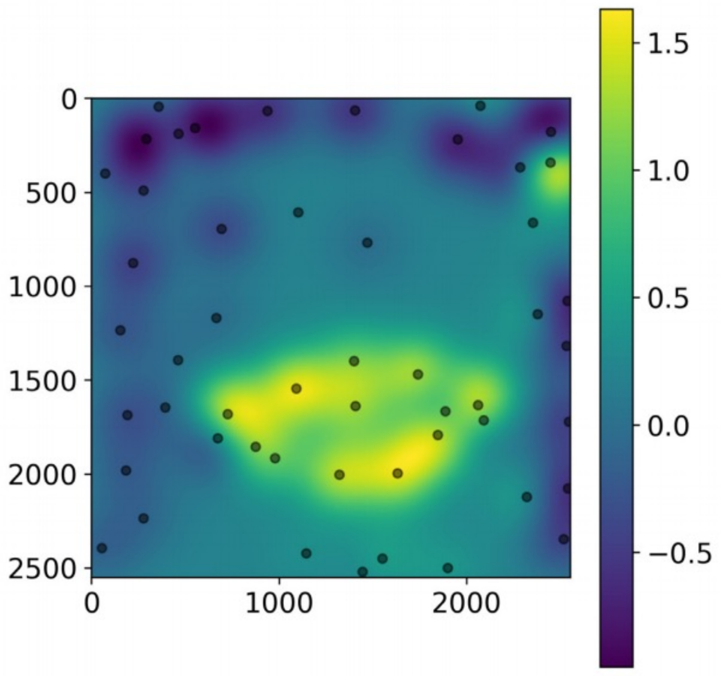


Ray & Myer 2019

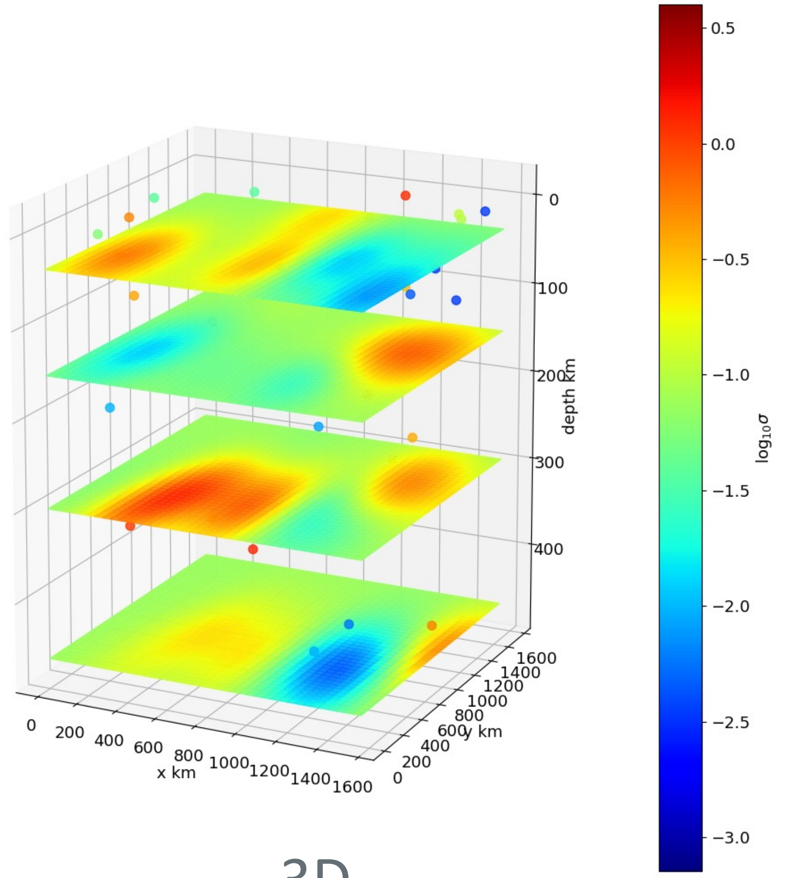
Represent N_d functions with *same* equation



1D



2D

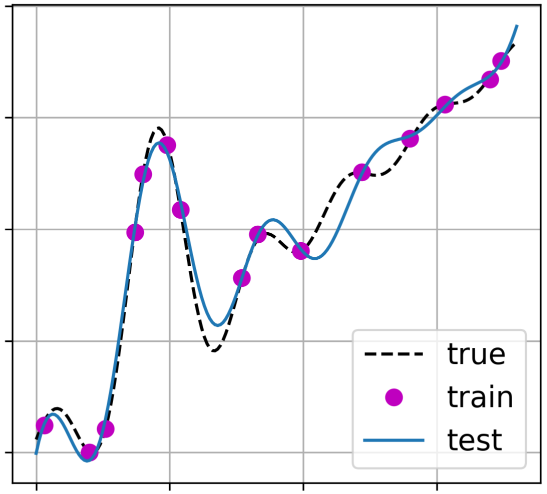


3D

Gaussian process mean

$$K(\mathbf{y}, \mathbf{y}') = \exp\left(-\frac{1}{2}[\mathbf{y} - \mathbf{y}']^t \mathbf{C}_\lambda{}^{-1}[\mathbf{y} - \mathbf{y}']\right), \text{ where } \mathbf{y} \in \mathbb{R}^{nd}$$

Rasmussen & Williams (2006)



1D

$$\boldsymbol{\mu}_* = \mathbf{K}_* \mathbf{K}_m^{-1} \mathbf{m}$$

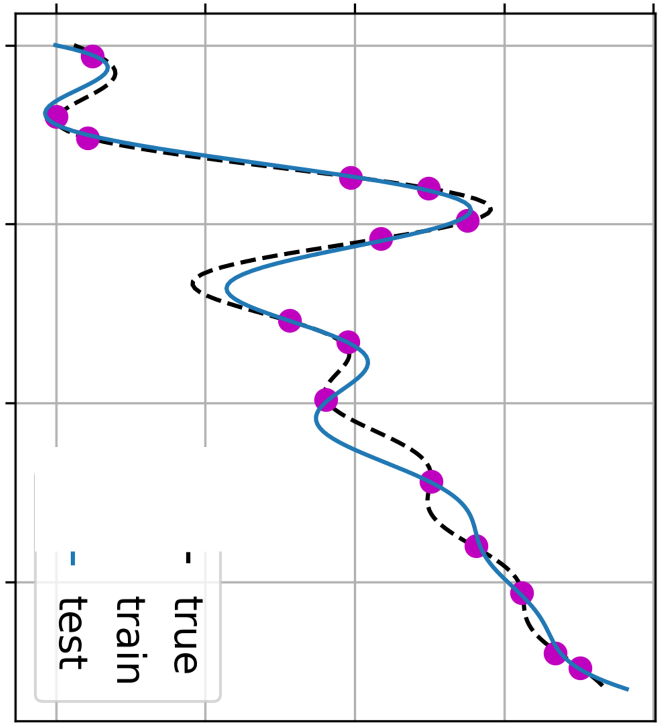
Full model vector
n × 1 **(LARGE)**

"Cross" kernel matrix
n × r

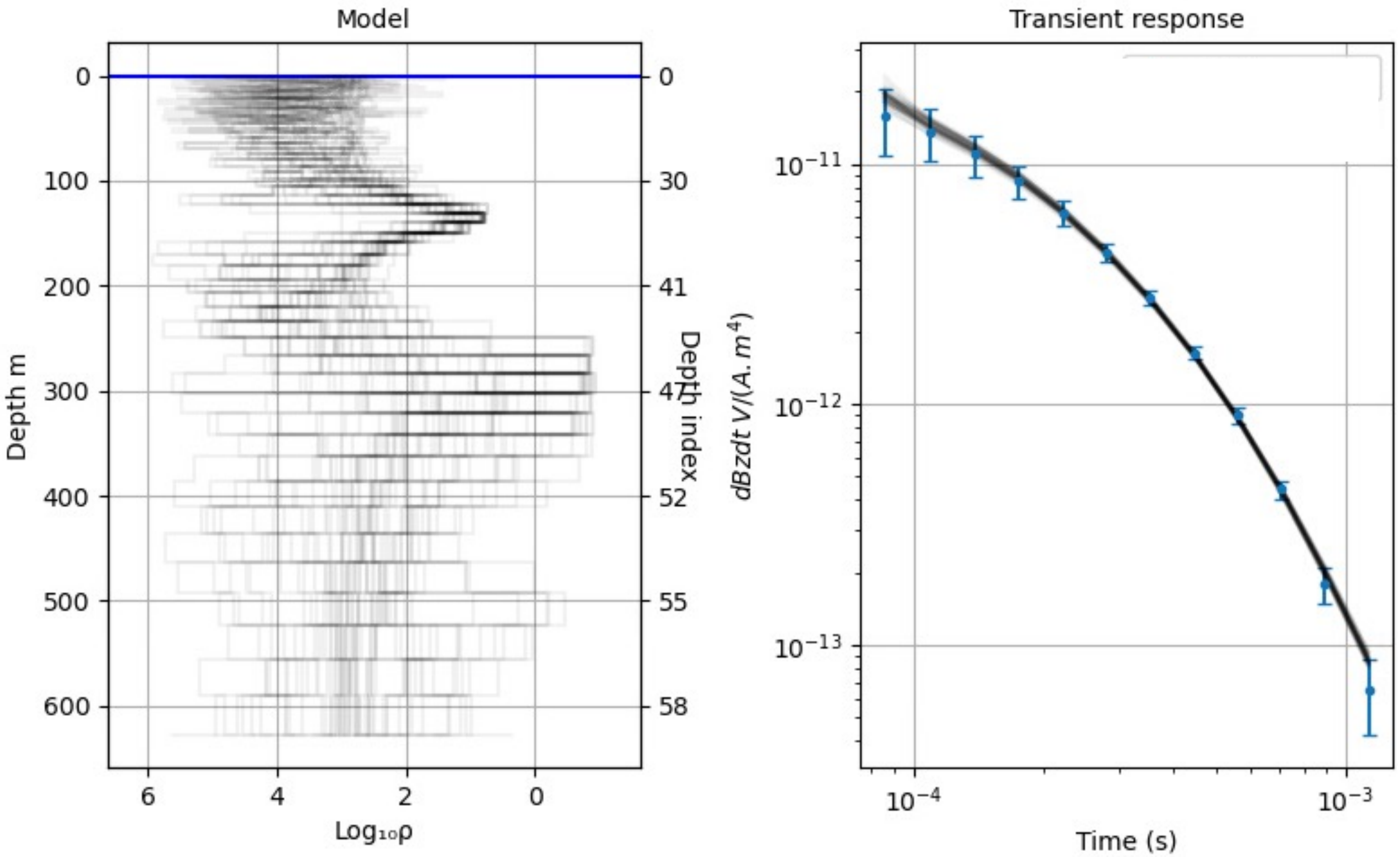
"Self" kernel matrix
r × r

GP nuclei model vector
r × 1 **(small)**

Now does this look like an earth property?



Transdimensional Gaussian processes (**TDGP**)
Ray & Myer 2019



What we'll do different now: self parameterisation

Ordinary McMC	Change model parameters while sampling
trans-D McMC (and TDGP)	***Add/delete*** parameters while sampling
Nested TDGP	Construct above parameters using ***another*** trans-D Gaussian process

$$\mathbf{C}_{\text{avg}} = \frac{\mathbf{C}_i + \mathbf{C}_j}{2}.$$

$$k(\mathbf{y}_i, \mathbf{y}_j) = |\mathbf{C}_i|^{\frac{1}{4}} |\mathbf{C}_j|^{\frac{1}{4}} |\mathbf{C}_{\text{avg}}|^{-\frac{1}{2}} R(\sqrt{Q_{ij}}),$$

Ray 2021
Following Paciorek & Schervish 2003

2-layer Gaussian process

contains length scale values and their locations

$$\boxed{\boldsymbol{\theta}_{\mathrm{s}}}, \boldsymbol{\lambda}_{\mathrm{s}}, \sigma_{\mathrm{s}} \xrightarrow[\text{with Equation (4)}]{\text{use Equation (1)}} \boldsymbol{\mu}_{*\mathrm{s}}$$

contains property values and their locations

$$\boxed{\boldsymbol{\theta}_{\mathrm{ns}}}, \boldsymbol{\mu}_{*\mathrm{s}}, \sigma_{\mathrm{ns}} \xrightarrow[\text{with Equation (7)}]{\text{use Equation (1)}} \boldsymbol{\mu}_{*\mathrm{ns}}$$

To compute the misfit, we need $\boldsymbol{\mu}_{*ns}$

To compute $\boldsymbol{\mu}_{*ns}$ we need $\boldsymbol{\mu}_{*s}$

Ray 2021

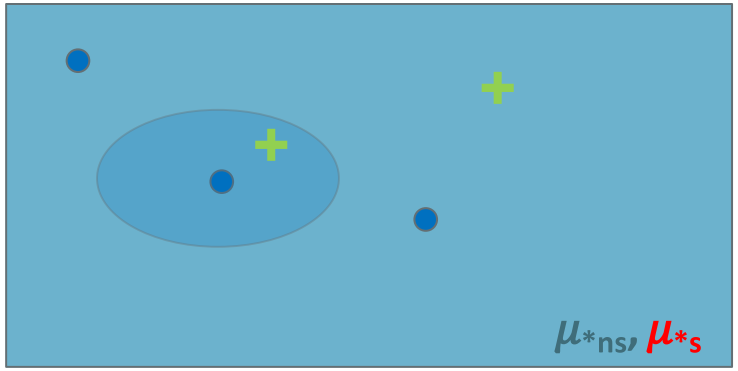
Structure of changes in an update #2

contains length scale values and their locations

$$\boxed{\boldsymbol{\theta}_{\mathrm{s}}}, \boldsymbol{\lambda}_{\mathrm{s}}, \sigma_{\mathrm{s}} \xrightarrow[\text{with Equation (4)}]{\text{use Equation (1)}} \boldsymbol{\mu}_{*\mathrm{s}}$$

contains property values and their locations

$$\boxed{\boldsymbol{\theta}_{\mathrm{ns}}}, \boldsymbol{\mu}_{*\mathrm{s}}, \sigma_{\mathrm{ns}} \xrightarrow[\text{with Equation (7)}]{\text{use Equation (1)}} \boldsymbol{\mu}_{*\mathrm{ns}}$$



- Only propagate significant changes in $\boldsymbol{\mu}_{*s}$ to $\boldsymbol{\mu}_{*ns}$
- KDTree searches for elements of $\boldsymbol{\theta}_{ns}$ (+) “close” to $\boldsymbol{\theta}_{s}$ (•)

Ray 2021