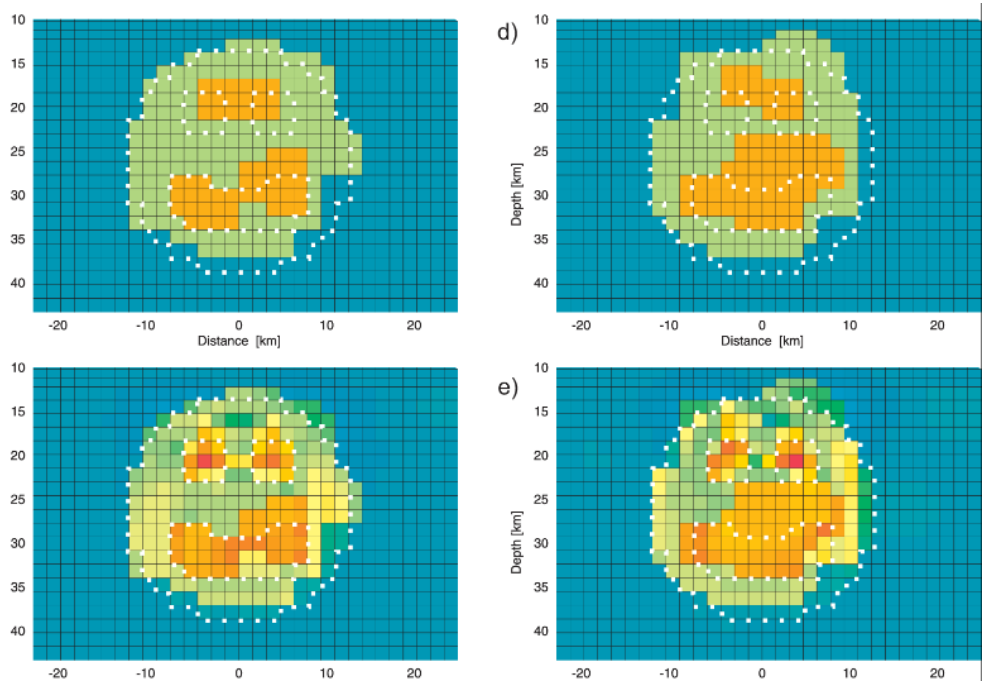
Remarks on Jacobians in large-scale inverse problems



Volker Rath

DIAS
Institiúid Ard-Léinn Bhaile Átha Cliath
Dublin Institute for Advanced Studies

Roadmap

- Motivation
- Jacobians from ModEM: JacoPyAn
- Computational issues
  - How to get them from Modem
  - Some properties of the Jacobian
    Sensitivity
- (Nullspace shuttle)

Bayesian MAP Estimation: P- and D-Space

$$\Theta_B = (\mathbf{d}-\mathbf{g}(\mathbf{p}))^T \mathbf{C}_\mathbf{d}^{-1}(\mathbf{d}-\mathbf{g}(\mathbf{p})) + (\mathbf{p}-\mathbf{p}_a)^T \mathbf{C}_\mathbf{p}^{-1}(\mathbf{p}-\mathbf{p}_a) = \min!$$

Differentiate with respect to parameters and apply, e.g., Gauss-Newton method gives iteration formulae (normal equation)

$$\mathbf{p}^{k+1} = \mathbf{p}^a + (\mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} + \mathbf{C}_p^{-1})^{-1} \cdot \mathbf{J}^T \mathbf{C}_d^{-1}[\mathbf{d}-\mathbf{g}(\mathbf{p}^k)]$$

$$\mathbf{p}^{k+1} = \mathbf{p}^a + \mathbf{C}_p \mathbf{J}^T (\mathbf{J}\mathbf{C}_p \mathbf{J}^T + \mathbf{C}_d)^{-1} \cdot [\mathbf{d}-\mathbf{g}(\mathbf{p}^k)]$$

$$J_{ij} = \frac{\partial g_i}{\partial p_j}, \quad \mathbf{J} \in \mathbb{R}^n \times \mathbb{R}^m$$

Jacobian

Note:

- Covariance matrices or their inverses for D- and P-space, respectively.
- matrices are $M \times M$ and $N \times N$, $M \gg N$

$$\mathbf{p}^{k+1} = \mathbf{p}^a + (\mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} + \mathbf{C}_p^{-1})^{-1} \cdot \mathbf{J}^T \mathbf{C}_d^{-1}[\mathbf{d}-\mathbf{g}(\mathbf{p}^k)] \qquad \text{(P-space)}$$

$$\Rightarrow (\mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} + \mathbf{C}_p^{-1}) \cdot \Delta\mathbf{p} = \mathbf{J}^T \mathbf{C}_d^{-1}[\mathbf{d}-\mathbf{g}(\mathbf{p}^k)]$$

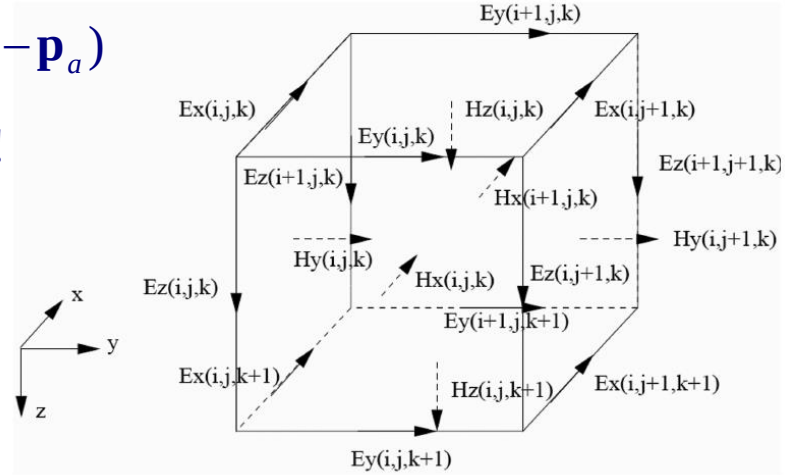
Bayesian MAP Estimation: ModEM

- recast as optimization: NLCG (and QN, LBFGS) methods only need gradient of objective function:

$$\Theta = (\mathbf{d}-\mathbf{g}(\mathbf{p}))^T \mathbf{C}_\mathbf{d}^{-1}(\mathbf{d}-\mathbf{g}(\mathbf{p})) + (\mathbf{p}-\mathbf{p}_a)^T \mathbf{C}_\mathbf{p}^{-1}(\mathbf{p}-\mathbf{p}_a)$$

$$= \left\| \mathbf{C}_\mathbf{d}^{-1/2}(\mathbf{d}-\mathbf{g}(\mathbf{p})) \right\|_2^2 + \left\| \mathbf{C}_\mathbf{p}^{-1/2}(\mathbf{p}-\mathbf{p}_a) \right\|_2^2 = \min!$$

$$\nabla\Theta = -0.5(\mathbf{J}^T \mathbf{C}_d^{-1}(\mathbf{d}-\mathbf{g}(\mathbf{p})) + \mathbf{C}_p^{-1}(\mathbf{p}-\mathbf{p}_a))$$



- transformed setup: forward problem calculates E-fields for 2-Polarisations – more transformations (linear interpolation), lots of geometrical (precomputable) matrices….

Bayesian MAP Estimation: ModEM

Transformed equations lead to more elegant (simpler) algorithm:

Note: geometrical transforms not touched.

$$\Theta = (\mathbf{g}(\mathbf{p}) - \mathbf{d})^T \mathbf{C}_\mathbf{d}^{-1} (\mathbf{g}(\mathbf{p}) - \mathbf{d}) + (\mathbf{p} - \mathbf{p}_a)^T \mathbf{C}_\mathbf{p}^{-1} (\mathbf{p} - \mathbf{p}_a)$$

$$= \left\| \mathbf{C}_\mathbf{d}^{-1/2} (\mathbf{g}(\mathbf{p}) - \mathbf{d}) \right\|_2^2 + \left\| \mathbf{C}_\mathbf{p}^{-1/2} (\mathbf{p} - \mathbf{p}_a) \right\|_2^2$$

$$\tilde{\mathbf{p}} = \mathbf{C}_\mathbf{m}^{-1/2} \left(\mathbf{p} - \mathbf{p}_a \right)$$

$$\tilde{\mathbf{d}} = \mathbf{C}_\mathbf{d}^{-1/2} \mathbf{d}$$

$$\tilde{\mathbf{g}}(\tilde{\mathbf{p}}) = \mathbf{C}_\mathbf{d}^{-1/2} \mathbf{g}\left(\mathbf{C}_\mathbf{m}^{1/2} \tilde{\mathbf{p}} \right)$$

$$\tilde{\mathbf{J}} = \mathbf{C}_\mathbf{d}^{-1/2} \mathbf{J} \mathbf{C}_\mathbf{m}^{1/2}$$

$$\tilde{\Theta}(\tilde{\mathbf{p}}, \tilde{\mathbf{d}}) = \left\| \tilde{\mathbf{d}} - \tilde{\mathbf{g}}(\tilde{\mathbf{p}}) \right\|^2 + \lambda \left\| \tilde{\mathbf{p}} \right\|^2$$

Transform back to actual model space:

$$\mathbf{p} = \mathbf{C}_\mathbf{m}^{1/2} \tilde{\mathbf{p}} + \mathbf{p}_a$$

$$\mathbf{J} = \mathbf{C}_\mathbf{d}^{1/2} \tilde{\mathbf{J}} \mathbf{C}_\mathbf{m}^{-1/2}$$

Bayesian MAP Estimation: ModEM

$$\nabla\Theta = -0.5\left(\mathbf{J}^T\mathbf{C}_d^{-1}(\mathbf{d}-\mathbf{g}(\mathbf{p}))+\mathbf{C}_p^{-1}(\mathbf{p}-\mathbf{p}_a)\right)$$

- $\mathbf{J}^T\mathbf{r}$ product can be calculated by just 1 additional forward solution per iteration, with data residuals as sources (reciprocity).
- Internally, Jacobian $\mathbf{J}$ is calculated by N repeated solution.
- $\mathbf{J}$ output is not scaled by (diagonal) inverse data covariance

Given a differentiable function $f(\mathbf{x})$, and an initial solution $\mathbf{x}^{(0)}$, let $\beta_0 = 0$, $\mathbf{p}^{(-1)} = \mathbf{0}$, and $k = 0$.

1. Let $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
2. If $k > 0$, let $\beta_k = \frac{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}{\mathbf{g}^{(k-1)T}\mathbf{g}^{(k-1)}}$.
3. Let $\mathbf{p}^{(k)} = -\mathbf{g}^{(k)} + \beta_k\mathbf{p}^{(k-1)}$.
4. Find α_k to minimize $f(\mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)})$.
5. Let $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}$.
6. Let $k = k + 1$.
7. Repeat the previous steps until convergence.

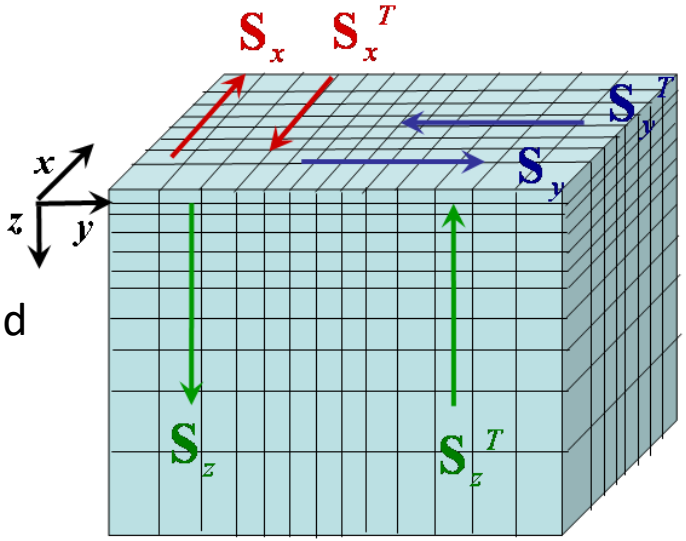
Bayesian MAP Estimation: ModEM

ModEM uses a simple covariance $\mathbf{C}_\mathbf{m}$, implemented as a series of 1D dimensional autoregressive smoothers $\hat{\varepsilon}_n = \alpha \hat{\varepsilon}_{n-1} + \varepsilon_n$

- 1D AR smoothers $\mathbf{S}$ applied successively in x, y, z

$$\mathbf{C}_\mathbf{m}^{1/2} = \mathbf{D}\mathbf{S}_x\mathbf{S}_y\mathbf{S}_z$$

- $\mathbf{D}$ is diagonal scaling operator, (a) for numerical stability; (b) spatial variations in *a priori* model uncertainty; (c) account for cell size variations in grid



- Convenient to make symmetric:

$$\mathbf{C}_\mathbf{m}^{1/2} = \mathbf{D}\mathbf{S}_x\mathbf{S}_y\mathbf{S}_z\mathbf{S}_z^T\mathbf{S}_y^T\mathbf{S}_x^T\mathbf{D}$$

→ Adjoint smoothers ($\mathbf{S}^T$): reverse direction of AR smoothers (and reverse order: z-y-x)

Bayesian MAP Estimation: ModEM

Smoothing step can be expressed as a matrix (here x direction):

$$\mathbf{S}_x = diag(\ S^x_{11}\quad S^x_{21}\quad \cdots\quad S^x_{N_yN_z}\ )$$

$$S^x_{ij} \equiv \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha & 1 & 0 & \cdots & 0 \\ \alpha^2 & \alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{N_x-1} & \alpha^{N_x-2} & \cdots & \cdots & 1 \end{bmatrix} \qquad \left[S^x_{ij}\right]^{-1} \equiv \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\alpha & 1 & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}$$

- Inverse of S_{ij} is sparse, so also $\mathbf{S}^{-1}$
- $\alpha < 1$. (default 0.2) $\Rightarrow$ $\mathbf{S}$ approximately sparse

Proof: Tarantola 1987, 2005; Rodgers 2000

Bayesian MAP Estimation: ModEM

Procedure can be shown to be equivalent to multiply with (kind of) exponential covariance

$$(1 - a^2)^{-1} \exp\left[\left|i - i'\right| \ln a\right]$$

Nominal length scale (one step): $L = -1 / \ln \alpha$

Important:

- *penalty defined on a uniform grid*
- *actual cell dimensions not used*
- *length scale not in physical units*
- *large cells not penalized!*

Bayesian MAP Estimation: ModEM

$$S_{ij}^{x} \equiv \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha & 1 & 0 & \cdots & 0 \\ \alpha^{2} & \alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{N_x-1} & \alpha^{N_x-2} & \cdots & \cdots & 1 \end{bmatrix}$$

$$\left[S_{ij}^{x}\right]^{-1} \equiv \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\alpha & 1 & 0 & \cdots & 0 \\ 0 & -\alpha & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}$$

How to get Jacobians from ModEM

During inversion, the Jacobian is never formed!
Calculation of the Jacobian is only implemented as a testing tool (special, rarely used option).

```
mpirun -np `cat $OAR_FILE_NODES|wc -l` --machinefile
$OAR_NODE_FILE -mca plm_rsh_agent "oarsh" \

 /home/sbyrd/bin/gMod3DMTS_MKL.x \

-J \
SABA8_P.rho SABA8_P.dat SABA8_P.jac SABA.fwd > SABA8_P.out
```

Dahu stuff

Executable with modifications

model file

data file

Jacobian (huge, should be on special storage!)

model ctrl

Ouput to screen

Options for Jacobian calculations

How to get Jacobians from ModEM

```
…
-J \
SABA8_P.rho SABA8_P.dat SABA8_P.jac SABA.fwd > SABA8_P.out
```

model file data file Jacobian model ctrl

Remarks:

1. Here the model is from a phase tensor inversion, but the data file can be whatever you want.
2. As the Jacobian is independent of the data, this feature can be used for survey planning or optimization using virtual sites.
3. Model-based Optimal Experimetal Design (OED) using the SVD

Computational Issues

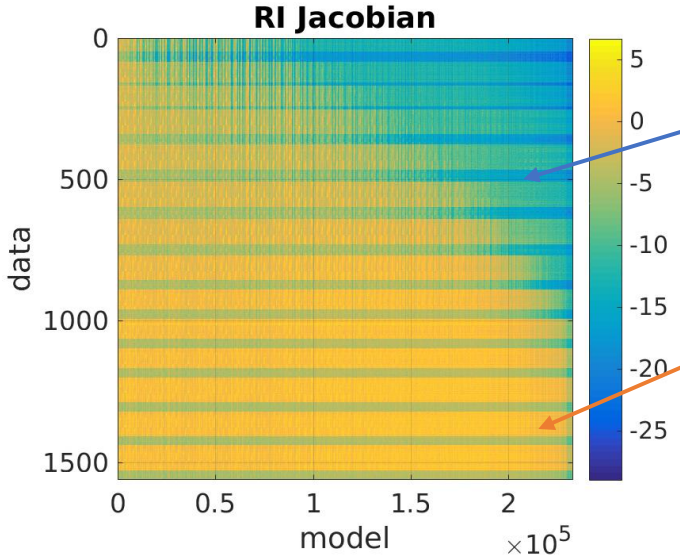
- Jacobians in this formulation are independent of data, but should be scaled with inverse errors
- Jacobians are large-to-huge, but have many “zeros”, and are “structured”
- Can be sparsified or compressed offline or “on the fly”, i.e., column-wise within ModEM
- Values for

Rathlin Basin (RB):
$68 \times 59 \times 82 = 328984$ Cells, 15196 data
$\approx 5 \cdot 10^9$ elements, 37 GB

Rathlin Island (RI):
$53 \times 61 \times 72 = 232776$ Cells, 1560 data
$\approx 4 \cdot 10^6$ elements, 2.6 GB

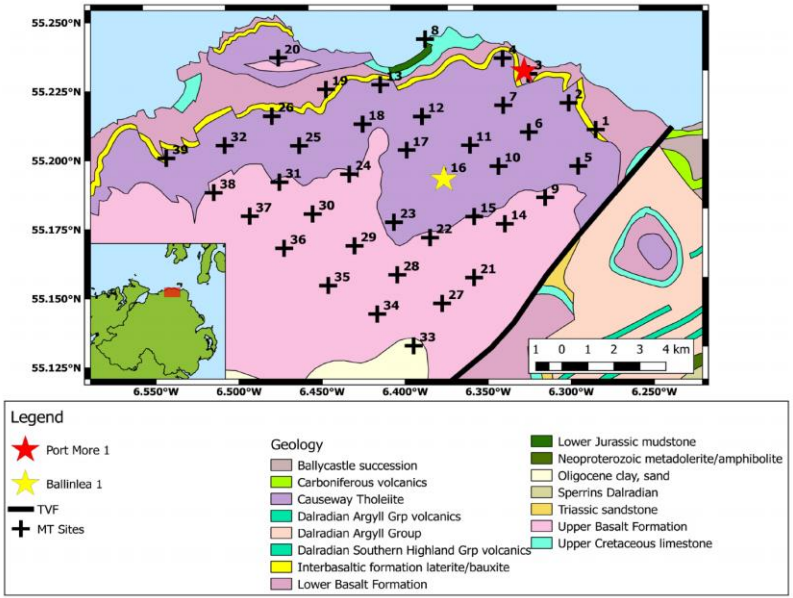
Thresh	nnz RB (RI)	Size RB (RI)
1e-4	<1% (2%)	170 MB (0.13 MB)
1e-6	14% (20%)	6.1 GB (12 MB)
1e-8	45% (38%)	19.4 GB (324 MB)
1e-10	58% (50%)	25.1 GB (1 GB)
1e-12	68% (67%)	29.2 GB (1.4 GB)
1e-14	81% (75%)	34.8 GB (1.8 GB)

How sparse can we make the Jacobian?



Tipper

Impedances



Rathlin Basin (RB):

$68 \times 59 \times 82 = 328984$ Cells, 15196 data

$\approx 5 \cdot 10^9$ elements, 37 GB

Rathlin Island (RI):

$53 \times 61 \times 72 = 232776$ Cells, 1560 data

$\approx 4 \cdot 10^6$ elements, 2.6 GB

JacoPyAn I

- Python (3.11) toolbox, with few packages required:
  (numpy, scipy, matplotlib …)
- Preprocessing (sparsify),
- manipulation (split/merge)
- sensitivity calculations
- Randomized SVD
- Nullspace shuttle for hypothesis testing and MC (linear domain)
- Experimental design
- Model space reduction

Installation:

1. Set up conda environment:
   `conda env create –n as-you-like python=3.11 numpy scipy …`
2. Download JacoPyAn:
   git clone https://github.com/volkerrath/JacoPyAn.git

JacoPyAn II

Installation:

1. Set up conda environment:
   ```
   conda env create -n as-you-like python=3.11 numpy
   scipy … (spyder, jupyterlab, …)
   ```

2. Download JacoPyAn:
   ```
   git clone https://github.com/volkerrath/JacoPyAn.git
   ```

3. Set environment variables (e.g. in `.bashrc`):
   ```
   export JACOPYAN_ROOT=where-you-have-installed-it
   export JACOPYAN_DATA=where-you-store-your-data
   ```

Sensitivity I

Error-normalized Jacobian $\tilde{\mathbf{J}} = \mathbf{C}_d^{-1/2}\mathbf{J}$

Definition of sensitivity is not unique, and various forms can be found in the literature:

1. *"Raw" sensitivities*, defined as

   No absolute values are involved, hence positive and negative elements occur.
   This does not conform to what we expect of sensitivity (positivity)

2. *"Euclidean" sensitivities* (most common) are is defined as:

   The square root of this sensitivity is often preferred.

3. *Coverage*. Here, the absolute values of the Jacobian are used:

For model blanking/shading, forms (2) and (3) can be used.

In some communities the Jacobian is called “sensitivity matrix”. Not here!

Sensitivity II

When moving from the error-normalised Jacobian to sensitivity, there are more choices for further normalisation, depending on the understanding and use of this parameter. If sensitivity is to be interpreted as an approximation to a continuous field over the volume of the model, it seems useful normalize by the cell volume. On the other hand, effect of the size is important when investigating the true role of this cell in the inversion. Finally, for comparing different data (sub)sets, it is convenient to do a final normalization by the maximum value in the model. All these options are implemented in the
toolbox.

Prior Covariances

$\mathbf{C}_d$

- Symmetric Positive Definite (SPD)
- Nearly always assumed diagonal, i.e. extremely sparse
- Often estimated from data

$$\mathbf{C}_d \equiv \mathrm{diag}\left(\begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_N^2 \end{bmatrix}\right)$$

$\mathbf{C}_p$

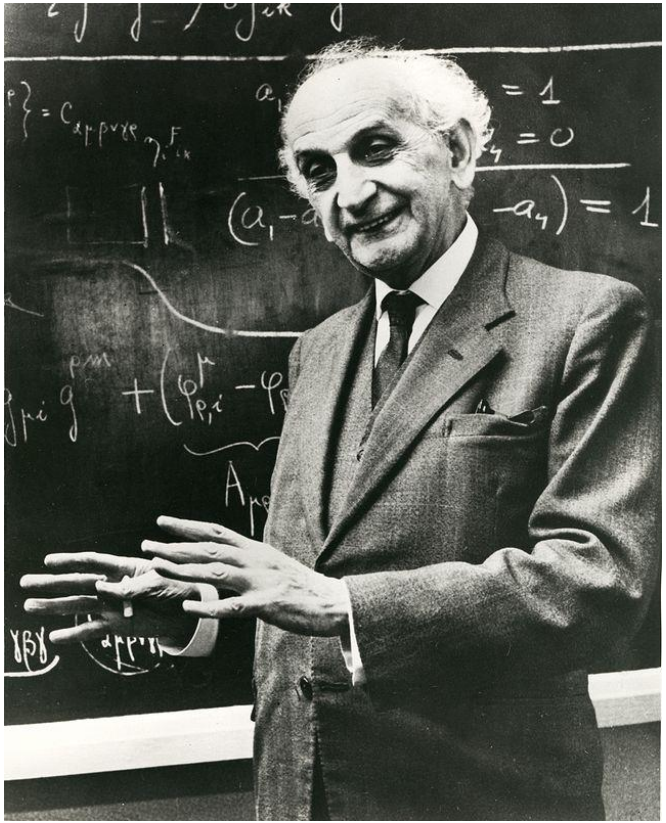
- Symmetric Positive Definite (SPD)
- Rarely diagonal, often spatial Exponential, Gaussian, Poisson, Matérn
- Not sparse, if not localized
- Rarely estimated from data (some nice examples from geostatistical inversion)

$$C_{p,ij}^{\exp} = \boldsymbol{\sigma}^2 \left[\exp\left(-\frac{|r_{ij}|}{L} \right) \right]$$

$$C_{p,ij}^{gauss} = \boldsymbol{\sigma}^2 \left[\exp\left(-\frac{|r_{ij}|^2}{2L} \right) \right]$$

Motivation: Why?

- early work on Nullspace Shuttle in MT (Muñoz & Rath, 2006)
- renewed interest of related methods in hydrogeology and seismology
- movement from 2-D to 3-D (data and modeling) has not improved the situation for uncertainty studies
- approximate methods for uncertainty and resolution analysis even more necessary
- “Big Data” : emergence of randomized linear algebra (“matrix sketching”)



Singular value decomposition – Basics

$$\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^m \quad n < m$$

$$[\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T] = svd(\mathbf{A})$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

Properties:

$\boldsymbol{\Sigma}$ diagonal

$\mathbf{U}$, $\mathbf{V}$ orthogonal(unitary)

$$\mathbf{U}\mathbf{U}^T = \mathbf{I} \quad \mathbf{V}\mathbf{V}^T = \mathbf{I} \quad \mathbf{U}^T, \mathbf{V}^T = \mathbf{U}^{-1}, \mathbf{V}^{-1}$$

$$\mathbf{A}\mathbf{x} = \mathbf{y} \Leftrightarrow \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{x} = \mathbf{y} \Leftrightarrow \boldsymbol{\Sigma}\tilde{\mathbf{x}} = \tilde{\mathbf{y}} \quad \tilde{\mathbf{x}} = \mathbf{V}^T\mathbf{x} \quad \tilde{\mathbf{y}} = \mathbf{U}^T\mathbf{y}$$

Singular value decomposition – Applications

General: Low –rank approximation

- Truncated or damped SVD inversion
- Principal Component Analysis (PCA/ICA)
- Noise-Estimation / reduction (see Poster Kiyan et al.)
- Model order reduction (MOR), e.g. Karhunen-Loeve Expansion (KLE)
- Uncertainty/resolution estimation
- Model-based Experimental design (OED)

Singular value decomposition - Numerics

I. Traditional (e.g., LINPACK, LAPACK and derivates)
   all $k = \min(m,n)$, but computational complexity $\approx \min\{mn^2, m^2n\}$
II. Iterative methods (e.g., EISPACK, ARPACK , PROPACK)
   only k largest values are needed (TSVD)
   SVD of $\mathbf{A}$ can be calculated by any method for eigendecomposition of: $\hat{\mathbf{A}} = \begin{bmatrix} 0 & \mathbf{A} \\ \mathbf{A}^T & 0 \end{bmatrix}$
   well adapted for sparse matrices
III. Randomized SVD. Only k largest values
   Implemented in several machine learning libraries, e.g. scikits.learn (Python, R)
   Very active area of research (”Big Data”)

- First numerical experiments with simple implementation give speedups of more than 2 orders of magnitude for “large” models
- Parallelism needs to be further investigated

Singular value decomposition - randomized

```
[M,N] = size(A);
P = min(2*K,N);
X = randn(N,P);W1 = orth(A*X);
B = W1'*A;
[W2,S,V] = svd(B,'econ');
U = W1*W2;

K=min(K,size(U,2));
U = U(:,1:K);
S = S(1:K,1:K);
V = V(:,1:K);
```

- generate matrix of orthonormal random columns transform M×N matrix to M×2K
- calculate SVD of smaller matrix B
- back transform
- Truncate to standard matlab “econ” form

direct rsvd algorithm
(after Antoine Liutkus 2014)

Nullspace shuttle – basic idea

- Jacobian $\mathbf{J}$: expand nonlinear model at $\mathbf{m}_0$

$$\mathbf{d}(\mathbf{m}_0 + \Delta\mathbf{m}) = \mathbf{d}(\mathbf{m}_0) + \mathbf{J} \cdot (\Delta\mathbf{m})$$

- Calculate Singular Value Decomposition (SVD)
  Numerical bottleneck!
  Truncate SVD, choose nullspace

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \Rightarrow \mathbf{J} = \mathbf{U}_p\boldsymbol{\Sigma}_p\mathbf{V}_p^T$$

- Construct Nullspace Projector $\mathbf{P}$
  Note $\mathbf{R}$ is called Resolution Matrix

$$\mathbf{P} = \mathbf{V}_0\mathbf{V}_0^T = \mathbf{I} - \mathbf{V}_p\mathbf{V}_p^T \quad \mathbf{R} = \mathbf{V}_p\mathbf{V}_p^T$$

- Perturb/deform model
  (e.g. smoothen, sharpen, and random)

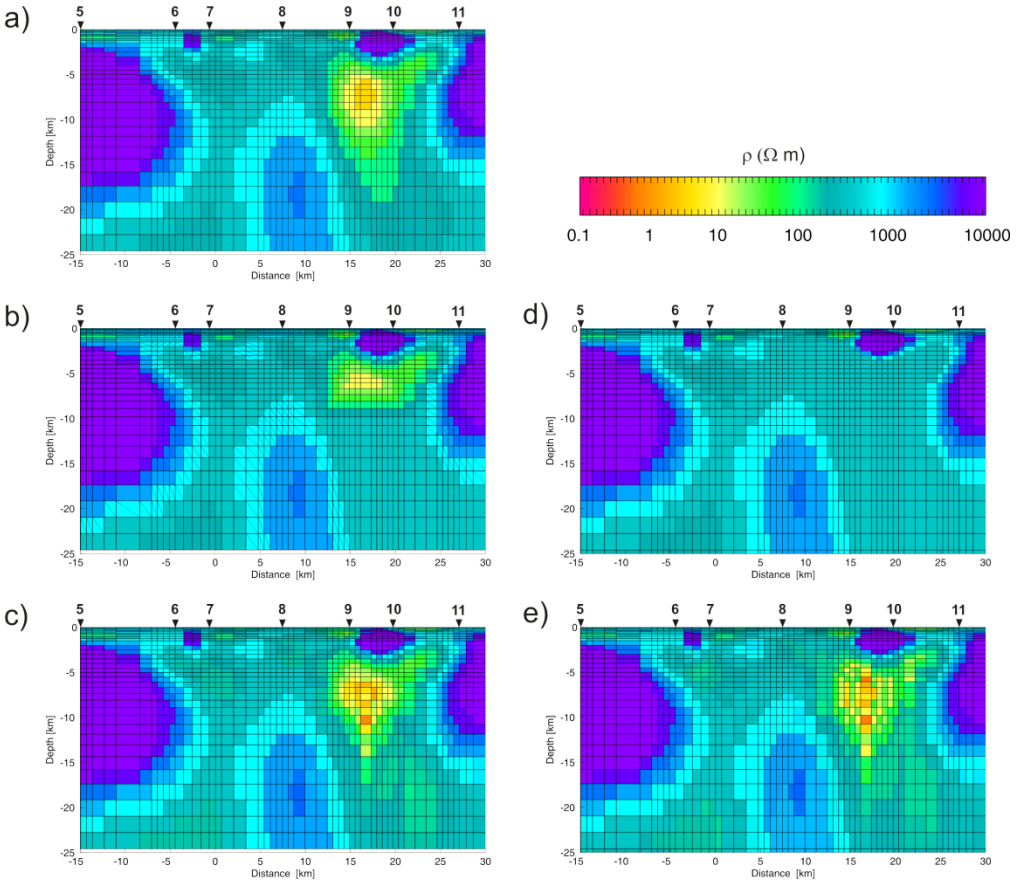
$$\mathbf{m}_f = f(\mathbf{m}_0, \mathbf{a}_f) \Rightarrow \Delta\mathbf{m} = \mathbf{m}_f - \mathbf{m}_0$$

- Project to Nullspace:
  $\mathbf{d}(\mathbf{m}_{null}) \approx \mathbf{d}(\mathbf{m}_0)$

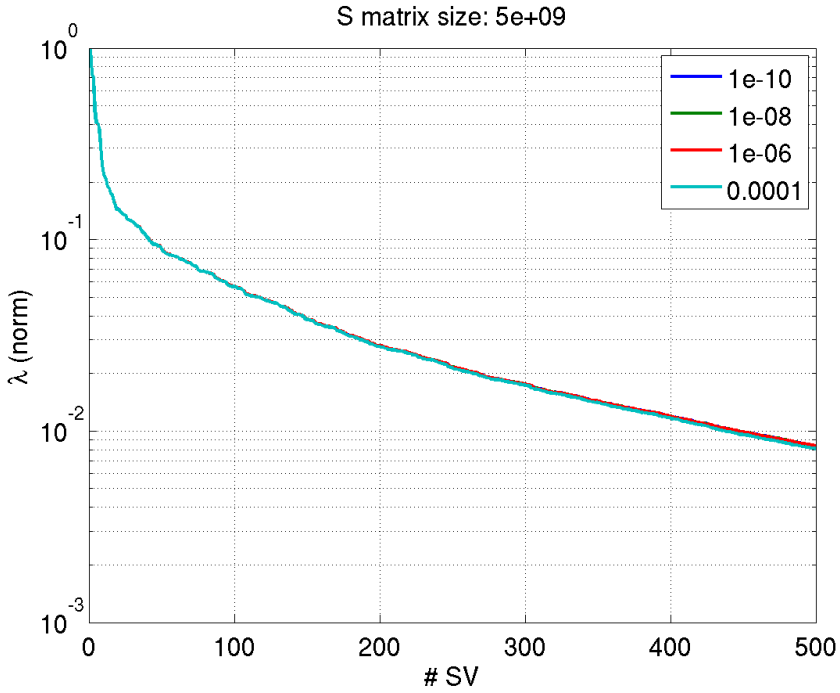
$$\begin{aligned}\mathbf{m}_{null} &= \mathbf{m}_0 + \mathbf{P} \cdot \Delta\mathbf{m} \\ &= \mathbf{m}_0 + \Delta\mathbf{m} - \mathbf{V}_p(\mathbf{V}_p^T\Delta\mathbf{m})\end{aligned}$$

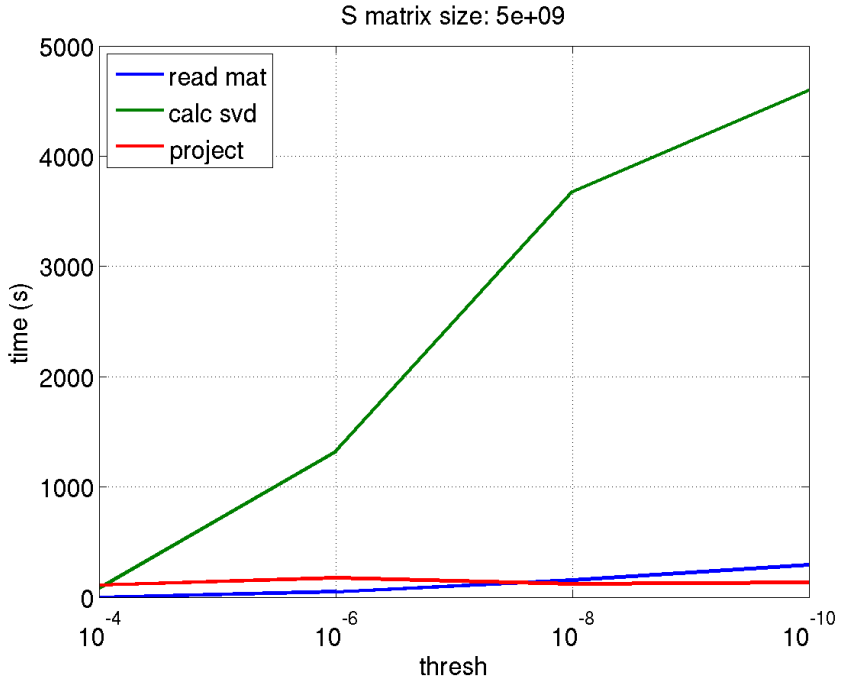
Nullspace shuttle - remarks

- NSP replaces the forward model by 2 matrix-vector products, once SVD is given
- Inverse problems are ill-posed - need to be regularized to be unique! In NSP we relax the regularization as we need to know what changes the data “tolerate”
- Open questions:
  - Numerics: Full SVD prohibitive ⇒Randomized SVD
  - How many singular values?
  - Which domain of validity (linear approximation)
  - Which perturbations or deformations?



Singular value decomposition - Timing





Singular value decomposition – experiments

Accuracy of rsvd results

$$\varepsilon = \frac{\|\mathbf{A}\| - \|\mathbf{A}_p\|}{\|\mathbf{A}\|} \quad \|\mathbf{A}\| = \sqrt{\sum}$$

Singular value decomposition – experiments

rsvd vs MATLAB svds
(Baglama & Reichel, 2005)

sparsity

Null-space Monte Carlo - basic idea

1. Calculate MAP model, Jacobian, and posterior Covariance $\mathbf{m}_0, \tilde{\mathbf{J}}, \mathbf{C}_p^{post}$
2. Calculate truncated singular value decomposition of the Jacobian
3. Generate a new random parameter vector using the parameter covariance matrix
4. Project perturbation vector on the null-space of the Jacobian
5. New parameter vector is then produced by adding to MAP model

(Tonkin and Doherty, 2009)

Null-space Monte Carlo - Posterior Covariance p-space

$$\tilde{\mathbf{C}}_p = \left(\mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} + \mathbf{C}_p^{-1} \right)^{-1}$$

$$= \mathbf{C}_p^{1/2} \left(\mathbf{C}_p^{1/2} \mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} \mathbf{C}_p^{1/2} + \mathbf{I} \right)^{-1} \mathbf{C}_p^{1/2}$$

$$\mathbf{C}_p^{1/2} \mathbf{J}^T \mathbf{C}_d^{-1} \mathbf{J} \mathbf{C}_p^{1/2} \approx \mathbf{V}_k \mathbf{S}_k \mathbf{V}_k$$

$$\mathbf{D}_k \equiv \mathrm{diag}\left(\frac{\lambda_i}{\lambda_i + 1} \right)$$

$$\tilde{\mathbf{C}}_p = \mathbf{C}_p - \mathbf{C}_p^{1/2} \left(\mathbf{V}_k \mathbf{D}_k \mathbf{V}_k \right) \mathbf{C}_p^{1/2}$$

(e.g. Flath et al. 2011)

Null-space Monte Carlo - Posterior Covariance d-space

$$\tilde{\mathbf{C}}_p = \mathbf{C}_p - \mathbf{C}_p \mathbf{J}^T (\mathbf{J}\mathbf{C}_p \mathbf{J}^T + \mathbf{C}_d)^{-1} \mathbf{J}\mathbf{C}_p{}^{-1}$$

$$= \mathbf{C}_p - \mathbf{C}_p \mathbf{J}^T \mathbf{C}_d^{-1/2} (\mathbf{C}_d^{-1/2} \mathbf{J}\mathbf{C}_p \mathbf{J}^T \mathbf{C}_d^{-1/2} + \mathbf{I})^{-1} \mathbf{C}_d^{-1/2} \mathbf{J}\mathbf{C}_p{}^{-1}$$

$$\mathbf{C}_d^{-1/2} \mathbf{J}\mathbf{C}_p^{1/2} \mathbf{C}_p^{1/2} \mathbf{J}^T \mathbf{C}_d^{-1/2} \approx \mathbf{V}_k \mathbf{S}_k \mathbf{V}_k$$

$$\tilde{\mathbf{C}}_p = \mathbf{C}_p - \mathbf{C}_p^{1/2} \left(\mathbf{V}_k \mathbf{D}_k \mathbf{V}_k \right) \mathbf{C}_p^{1/2}$$

$$\mathbf{D}_k \equiv \mathrm{diag}\left(\frac{\lambda_i}{\lambda_i + 1} \right)$$

References Nullspace

Baglama J & Reichel, L: Augmented Implicitly Restarted Lanczos Bidiagonalization Methods, *SIAM J. Sci. Comp.,* **2005***, 27*, 19-42

Bui-Thanh T, Ghattas O, Martin J & Stadler G: A Computational Framework for Infinite-Dimensional Bayesian Inverse Problems Part I: The Linearized Case, with Application to Global Seismic Inversion, *SIAM J. Sci. Comp.*, **2013**, 35, A2494-A2523

Deal M & Nolet G: Nullspace shuttles, *Geophys. J. Int.,* **1996***, 124*, 372-380 Doherty J: Calibration and Uncertainty Analysis for Complex Environmental Models, *Watermark,* **2015**

Flath H, Wilcox L, Akcelik V, Hill J, van Bloemen Waanders B & Ghattas O, Fast Algorithms for Bayesian Uncertainty Quantification in Large-Scale Linear Inverse Problems Based on Low-Rank Partial Hessian Approximations, *SIAM J. Sci. Comp.*, **2011**, 33, 407-432

Halko N, Martinsson P G & Tropp J A: Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *SIAM Review,* **2011***, 53*, 217-288

Kannan R & Vempala S: Randomized algorithms in Numerical Linear Algebra, *Acta Numerica,* **2017***, 26*, 95-135

Kundu A & Drineas, P: A Note on Randomized Element-wise Matrix Sparsification, **2014**, arXiv:1404.0320v1.

Lanczos C: Linear Differential Operators, *Van Nostrand ,* **1961**

Isaac T, Petra N, Stadler G & Ghattas O: Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet, *J. Comp. Phys.*, **2015**, 296, 348-368

Muñoz G & Rath V: Beyond smooth inversion: the use of nullspace projection for the exploration of non-uniqueness in MT, *Geophys. J. Int.,* **2006***, 164*, 301-311

Rowbotham P S & Pratt R G: Improved inversion through use of the null space, *Geophysics,* **1997***, 62*, 869-883

Tavakoli R, Yoon H, Delshad M, ElSheikh A H, Wheeler M F & Arnold B W: Comparison of ensemble filtering algorithms and null-space Monte Carlo for parameter estimation and uncertainty quantification using CO_2 sequestration data, *Water Resour. Res.,* **2013***, 49*, 8108-8127

Tonkin M & Doherty J: Calibration-constrained Monte Carlo analysis of highly parameterized models using subspace techniques, *Water Resour. Res.,* **2009***, 45*, W00B10

Woodruff D P: Sketching as a Tool for Numerical Linear Algebra, *Foundations and Trends in Theoretical Computer Science,* **2014***, 10*, 1-157

Xiang H & Zou J : Regularization with randomized SVD for large-scale discrete inverse problems, *Inverse Problems,* **2013***, 29*, 085008

References Jacobian

C. D. Rodgers: *Inverse Methods for Atmospheric Sounding*. World Scientific, 2000.

A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2005.