A Multilevel Preconditioner for Data Assimilation with 4D-Var

Alison Ramage*, Kirsty Brown*, Igor Gejadze†

∗ Department of Mathematics and Statistics, University of Strathclyde

† IRSTEA, Montpelier, France

Four-dimensional Variational Assimilation (4D-Var)

4D-Var aims to find the solution of a numerical forecast model that best fits sequences of observations distributed in space over a finite time interval.

Minimise cost function

$$J(\mathbf{v}_0) = (\mathbf{v}_0 - \mathbf{v}_0^B)^T B^{-1} (\mathbf{v}_0 - \mathbf{v}_0^B) + \sum_{i=0}^{n} (\mathcal{H}(\mathbf{v}_i) - \mathbf{y}_i)^T R^{-1} (\mathcal{H}(\mathbf{v}_i) - \mathbf{y}_i)$$

with constraint $\mathbf{v}_i = \mathcal{M}^{i,0}(\mathbf{v}_0)$.

analysis	$\mathbf{v}_0$
background (short-term forecast)	$\mathbf{v}_0^B$
observations	$\mathbf{y}$
observation operator	$\mathcal{H}$
model dynamics	$\mathbf{v}_{i+1} = \mathcal{M}(\mathbf{v}_i)$
background error covariance matrix	B
observation error covariance matrix	R

Incremental 4D-Var

- Linearise $\mathcal{H}$, $\mathcal{M}$ and solve resulting unconstrained optimisation problem iteratively:

$$\bar{H}^i_{k-1} \equiv \left.\frac{\partial \mathcal{H}^i}{\partial \mathbf{v}}\right|_{\mathbf{v}=\mathbf{v}_{k-1}}, \qquad \bar{M}^{i,0}_{k-1} \equiv \left.\frac{\partial \mathcal{M}^{i,0}}{\partial \mathbf{v}}\right|_{\mathbf{v}=\mathbf{v}_{k-1}}$$

- Hessian of the cost function is

$$\mathbb{H} = B^{-1} + \widehat{H}^T \widehat{R}^{-1} \widehat{H}$$

where

$$\begin{aligned} \widehat{H} &= [(\bar{H}^0)^T, (\bar{H}^1 \bar{M}^{1,0})^T, \ldots, (\bar{H}^N \bar{M}^{N,0})^T]^T \\ \widehat{R} &= \mathtt{bldiag}(R_i), \quad i = 1, \ldots, N. \end{aligned}$$

- Cannot store $\mathbb{H}$ as a matrix: action of applying $\mathbb{H}$ to a vector is available, but expensive (involves both forward and backward model solves).

Hessian system

- Hessian linear system (within a Gauss-Newton method):

$$\mathbb{H}(\mathbf{u}_k)\delta\mathbf{u}_k = G(\mathbf{u}_k)$$

- Solve using Preconditioned Conjugate Gradient iteration (needs only $\mathbb{H}\mathbf{v}$).

Hessian system

- Hessian linear system (within a Gauss-Newton method):

$$\mathbb{H}(\mathbf{u}_k)\delta\mathbf{u}_k = G(\mathbf{u}_k)$$

- Solve using Preconditioned Conjugate Gradient iteration (needs only $\mathbb{H}\mathbf{v}$).

- Precondition $\mathbb{H}$ based on the background covariance matrix:

$$H = (B^{1/2})^T \mathbb{H} B^{1/2} = I + (B^{1/2})^T \widehat{H}^T \widehat{R}^{-1} \widehat{H} B^{1/2}$$

- Eigenvalues of H are more clustered, in a narrow band above one, with few eigenvalues distinct enough to contribute noticeably to the Hessian value.
  Haben et al. (2011), Tabeart et al. (2018)

Limited-memory approximation

- H amenable to limited-memory approximation.
- Find n_e leading eigenvalues and orthonormal eigenvectors using the Lanczos method (needs only $H\mathbf{v}$).
- Construct approximation

$$H \approx I + \sum_{i=1}^{n_e} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

Limited-memory approximation

- H amenable to limited-memory approximation.
- Find n_e leading eigenvalues and orthonormal eigenvectors using the Lanczos method (needs only $H\mathbf{v}$).
- Construct approximation

$$H \approx I + \sum_{i=1}^{n_e} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

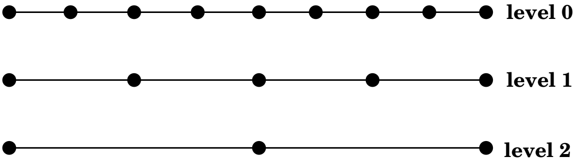
- Easy to evaluate matrix powers:

$$H^p \approx I + \sum_{i=1}^{n_e} (\lambda_i^p - 1)\mathbf{u}_i\mathbf{u}_i^T$$

- IDEA: Build a limited-memory approximation to H^{-1} (or $H^{-1/2}$) for use as a preconditioner in PCG.

Multilevel preconditioning

- Storage/working with H still expensive.
- IDEA: Construct a multilevel approximation to H^{-1} based on a sequence of nested grids.
- Discretise evolution equation on a grid with $m+1$ nodes (level 0) to represent Hessian H_0
- Grid level k contains $m_k = m/2^k + 1$ nodes.



- Identity matrix I_k on grid level k.

Grid transfers with "correction"

- Grid transfer based on piecewise cubic splines:
  - Restriction matrix R_c^f from $k = f$ to $k = c$.
  - Prolongation matrix P_f^c from $k = c$ to $k = f$.

- Construct new operators which transfer a matrix between a course grid level c and a fine grid level f.

  - From coarse to fine:

$$[A_c]_{\to f} = P_f^c(A_c - I_c)R_c^f + I_f$$

  - From fine to coarse:

$$[A_f]_{\to c} = R_c^f(A_f - I_f)P_f^c + I_c$$

Why should this work?

- Conjecture: the eigenvalues of

$$[H_{0\to k+1}^{-1/2}]_{\to k} H_{0\to k} [H_{0\to k+1}^{-1/2}]_{\to k}$$

should be clustered around 1.

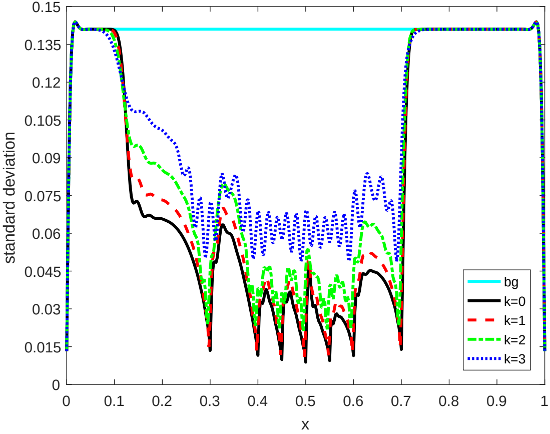
- Test using 1D Burgers' equation:
  - 1D uniform grid with 7 sensors located at 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, and 0.7 in $[0, 1]$.
  - Multilevel preconditioning with four grid levels:

k	0	1	2	3
grid points	401	201	101	51

Correlation matrix V

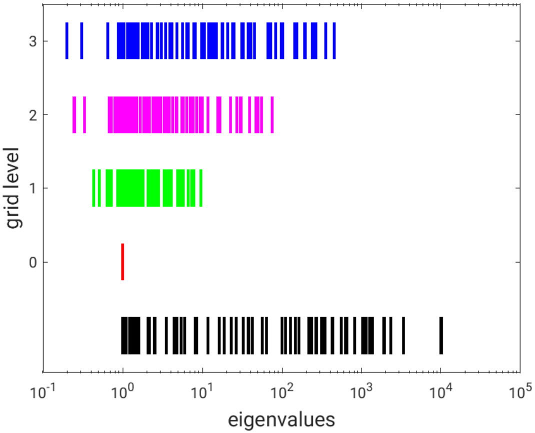
- Inverse Hessian matrix scaled to have unit diagonal:

$$V_{ij} = \frac{H_{ij}^{-1}}{H_{ii}^{-1/2} H_{jj}^{-1/2}}.$$



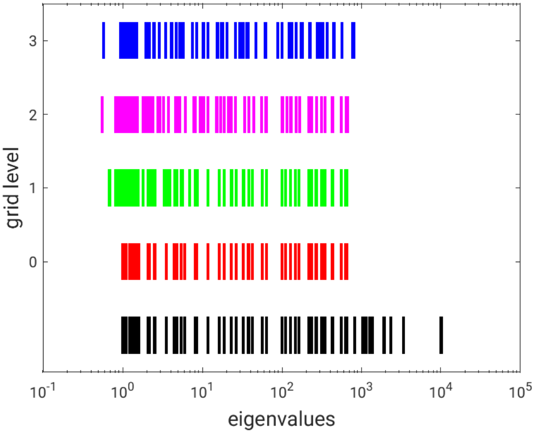
Idea behind preconditioning

- Eigenvalues of $[H_{0\to k}^{-1/2}]_{\to 0}\ H_0\ [H_{0\to k}^{-1/2}]_{\to 0}$.



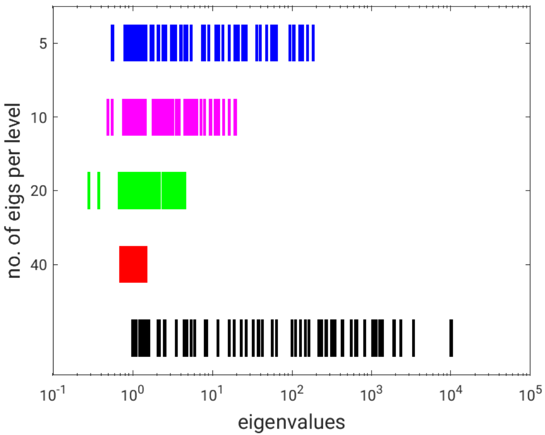
Replace with limited-memory approximations

- Use limited-memory form with 10 eigenvalues per level.



Idea: use all levels

- Build recursive preconditioner using information from all levels.



Outline of multilevel concept

Given a symmetric positive definite operator H_0 available on the finest grid level in matrix-vector product form:

1. represent H_0 on the coarsest grid level as $H_{0\to k}$;
2. use a local preconditioner B_k^{k+1} to obtain

$$\tilde{H}_{0\to k} = (B_k^{k+1})^T H_{0\to k} B_k^{k+1}$$

   with improved eigenvalue clustering;
3. build a limited memory approximation $\tilde{H}_{0\to k}^{-1/2}$ from n_k eigenvalues of $\tilde{H}_{0\to k}$ found using the Lanczos method;
4. project this to the level above to be used as local preconditioner at the next coarsest level;
5. move up one grid level and repeat.

Algorithm

- use $N_e = (n_0, n_1, \ldots, n_c)$ eigenvalues at each level

```
[Λ,U]=mlevd(H_0,N_e)
for   k = k_c, k_c − 1, ..., 0
   compute by the Lanczos method
   and store in memory
      {λ_k^i, U_k^i}, i = 1, ..., n_k of H̃_{0→k}
   using preconditioner B_k^{k+1}
end
```

- storage:

$$\begin{aligned}\Lambda &= \left[\lambda^1_{k_c}, \ldots, \lambda^{n_{k_c}}_{k_c}, \lambda^1_{k_c-1}, \ldots, \lambda^{n_{k_c-1}}_{k_c-1}, \ldots, \lambda^1_0, \ldots, \lambda^{n_0}_0\right], \\ \mathcal{U} &= \left[U^1_{k_c}, \ldots, U^{n_{k_c}}_{k_c}, U^1_{k_c-1}, \ldots, U^{n_{k_c-1}}_{k_c-1}, \ldots, U^1_0, \ldots, U^{n_0}_0\right].\end{aligned}$$

Assessing approximation accuracy

- Riemannian distance:

$$\delta(A,B) = \left\| ln(B^{-1}A) \right\|_F = \left(\sum_{i=1}^{n} ln^2 \lambda_i \right)^{1/2}$$

- Compare eigenvalues of H^{-1} and $\tilde{H}^{-1}$ on the finest grid level $k = 0$ using

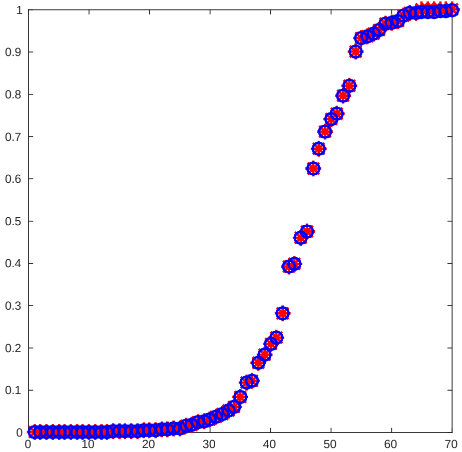
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

$$N_e = (n_0, n_1, n_2, n_3)$$

Eigenvalues of the inverse Hessian

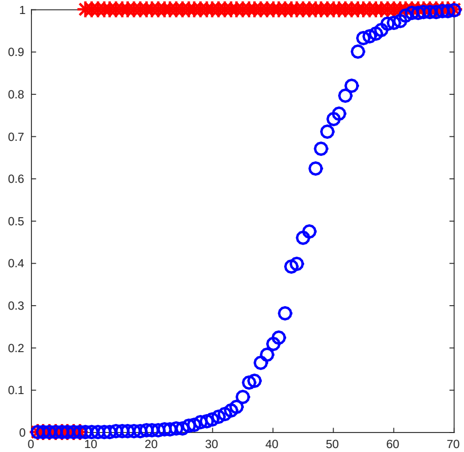
- Exact (blue circles), approximated (red stars)



$$N_e = (64, 0, 0, 0)$$
$$D = 2.98e - 4$$

Eigenvalues of the inverse Hessian

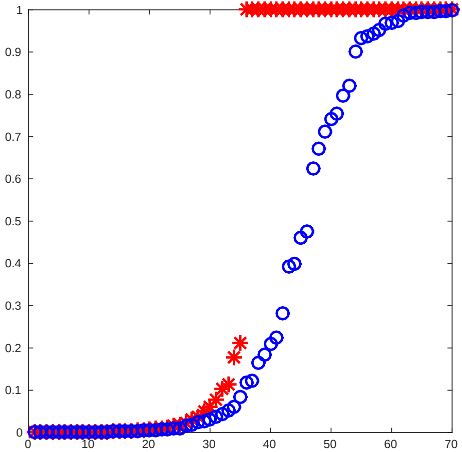
- Exact (blue circles), approximated (red stars)



$$N_e = (8, 0, 0, 0)$$
$$D = 7.71e - 1$$

Eigenvalues of the inverse Hessian

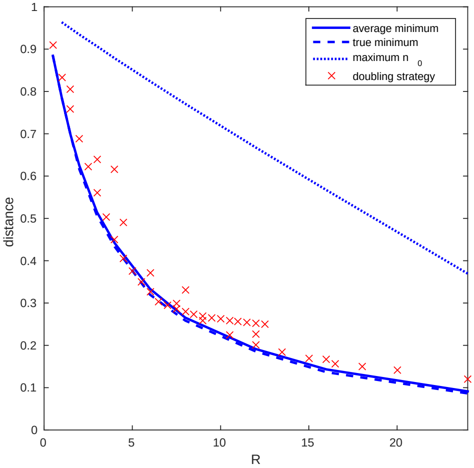
- Exact (blue circles), approximated (red stars)



$$N_e = (0, 0, 29, 6)$$
$$D = 3.39e-1$$

Fixed memory ratio

- Fixed memory ratio $R = \sum_{k=0}^{k_c} \frac{n_k}{2^k}$

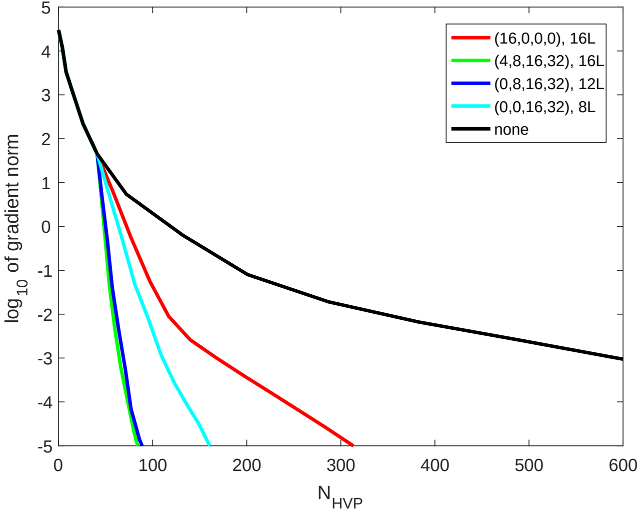


PCG iteration for one Newton step

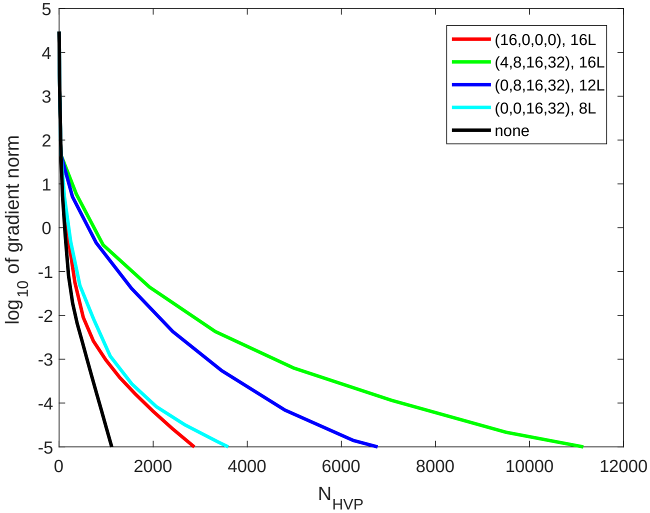
- measurement units
  - memory: length of vector on finest grid   L
  - cost: cost of HVP on finest grid   HVP

Preconditioner	# CG iterations	storage	solve cost
none	57	0 L	57 HVP
MG(400,0,0,0)	1	400 L	402 HVP
MG(4,8,16,32)	4	16 L	34 HVP
MG(0,8,16,32)	5	12 L	14 HVP
MG(0,0,16,32)	8	8 L	10 HVP

Solve cost measured in number of HVPs



Cost including building preconditioner



Hessian decomposition

- partition domain into S subregions and compute local Hessians H^s such that

$$H(\mathbf{v}) = I + \sum_{s=1}^{S} (H^s(\mathbf{v}) - I)$$

- fewer eigenvalues required for each local limited-memory approximation $\hat{H}^s \simeq H^s$

- local Hessians can be computed
  - in parallel;
  - using local rather than global models;
  - at any grid level k.

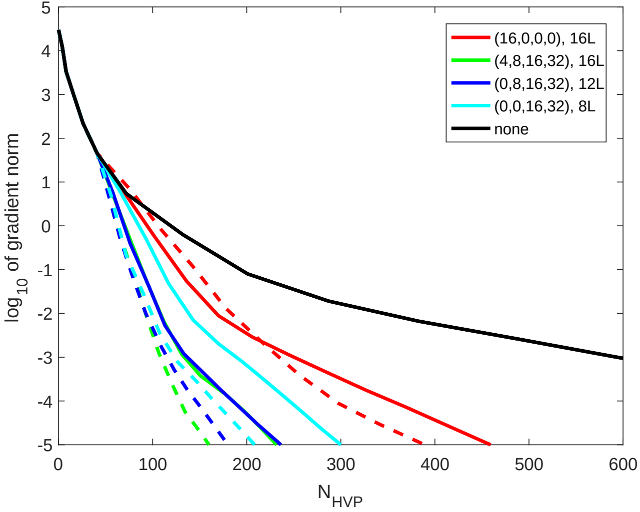
$$H_k(\mathbf{v}_k) = I_k + \sum_{s=1}^{S} (\hat{H}_k^s(\mathbf{v}_k) - I_k)$$

Practical approach: Version 1

- Compute limited-memory approximations to local sensor-based Hessians on level k using n_k eigenpairs.
- Assemble these to form H_a, then apply mlevd to H_a based on a fixed N_e.
- Local Hessians cheaper to compute.
- Additional user-specified parameter(s) k, n_k needed.
- More memory required as local Hessians must also be stored.

Version 1: cost including building preconditioner

- Local Hessians with 8 eigenvalues at level 0 (solid lines) or level 1 (dashed lines).

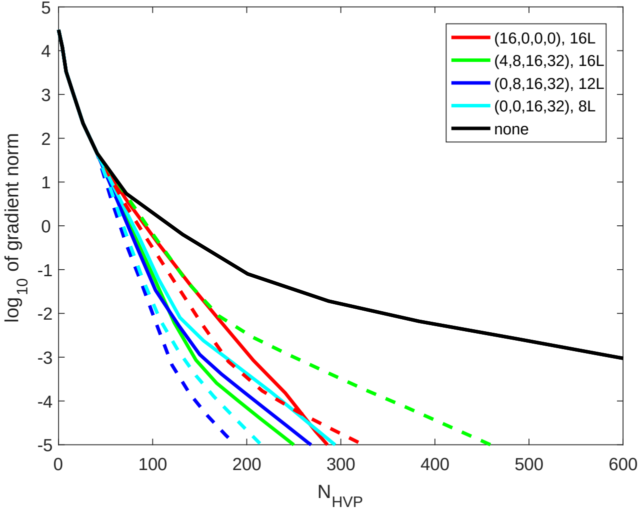


Practical approach: version 2

- Can reduce memory requirements further by using a multilevel approximation of each limited-memory local Hessian on level k using n_k eigenpairs.
- Approximate local Hessians by applying mlevd to local inverse Hessians based on N_e^k.
- Assemble these to form a reduced-memory assembled Hessian H_a^{rm}.
- Use mlevd again on H_a^{rm} based on N_e.

Version 2: cost including building preconditioner

- Local Hessians with 8 eigenvalues at level 0 (solid lines) or level 1 (dashed lines) with (8,4,0,0) MG approx.



Conclusions and next steps

- Similar results with other configurations (e.g. moving sensors, different initial conditions).
- Multilevel preconditioning looks promising for constructing a good limited-memory approximation to H^{-1}.
- The balance between restrictions on memory/cost limitations may vary between particular applications.
- Identifying globally appropriate values for (n_0, n_1, n_2, n_3) and other parameters is tricky, but "rules of thumb" can be developed.

- Current investigation: application to shallow water equations.

- Future investigations:
  - problems in higher dimensions;
  - applications for other sensor systems.

Multilevel algorithm for H^{-1}

- Represent H_0 at a given level (k, say):

$$H_{0\to k} = R_k^0 (H_0 - I_0) P_0^k + I_k$$

- Precondition to improve eigenvalue spectrum:

$$\tilde{H}_{0\to k} = (B_k^{k+1})^T H_{0\to k} B_k^{k+1}$$

- Find n_k eigenvalues/eigenvectors of $\tilde{H}_{0\to k}$ using the Lanczos method.

- Approximate $\tilde{H}_{0\to k}^{-1/2}$:

$$\tilde{H}_{0\to k}^{-1/2} \approx I_k + \sum_{i=1}^{n_k} \left(\frac{1}{\sqrt{\lambda_i}} - 1 \right) \mathbf{u}_i \mathbf{u}_i^T$$

Preconditioners

- Construct B_k^{k+1} on level $k+1$, apply on level k.
- On coarsest grid, level $k+1$ does not exist so set $B_k^{k+1} = I_k$.
- For other levels, construct preconditioners recursively:

$$B_k^{k+1} = \left[B_{k+1}^{k+2} \tilde{H}_{0\to k+1}^{-1/2} \right]_{\to k}, \quad {B_k^{k+1}}^T = \left[\tilde{H}_{0\to k+1}^{-1/2} {B_{k+1}^{k+2}}^T \right]_{\to k}$$

- Square brackets represent projection to the correct grid level using "corrected" grid transfers, e.g.

$$[A_{k+1}]_{\to k} = R_k^{k+1}(A_{k+1} - I_{k+1})P_{k+1}^k + I_k$$

Four-dimensional Variational Assimilation (4D-Var)

- 4D-Var aims to find the solution of a numerical forecast model that best fits sequences of observations distributed in space over a finite time interval.

Minimise cost function

$$J(\mathbf{v}) = \frac{1}{2}[\mathbf{v} - \mathbf{v}^b]^T B^{-1}[\mathbf{v} - \mathbf{v}^b] + \frac{1}{2}\sum_{i=0}^{N}[\mathcal{H}_i(\mathcal{M}_{i,0}(\mathbf{v})) - \mathbf{y}_i^o]^T R_i^{-1}[\mathcal{H}_i(\mathcal{M}_{i,0}(\mathbf{v})) - \mathbf{y}_i^o]$$

analysis $\mathbf{v}$, background $\mathbf{v}^b$, observations $\mathbf{y}^o$

background and observation error covariance matrices B, R_i

observation operators $\mathcal{H}_i$

model propagator $\mathcal{M}_{i,0} \equiv \mathcal{M}(t_i, t_0) \equiv \prod_{k=i}^{1} \mathcal{M}(t_k, t_{k-1})$

Motivation

Motivation

It is sometimes nice in Scotland. . .