A multilevel preconditioner for data assimilation with 4D-Var

Alison Ramage and Kirsty Brown,
Mathematics and Statistics,
University of Strathclyde,
Glasgow, Scotland

University of Strathclyde Glasgow

irstea

Igor Gejadze,
National Research Institute of Science and Technology for Environment and Agriculture,
Montpelier, France

Data assimilation

- Numerical weather prediciton is an IVP: given initial conditions, forecast atmospheric evolution.

- Data assimilation is a technique for combining information such as observational and background data with numerical models to obtain the best estimate of state of a system (initial condition).

- Other application areas include hydrology, oceanography, environmental science, data analytics, sensor networks. . .

- Variational assimilation is used to find the optimal analysis that minimises a specific cost function.

Motivation


JST SERVICES

Data assimilation problem

- Evolution process:

$$
\begin{aligned}
\frac{\partial \phi}{\partial t} &= F(\phi) + f, \qquad && t \in (0, T), \\
\phi\big|_{t=0} &= u, && \phi, u \in X, \ \phi \in Y
\end{aligned}
$$

true initial state	$\bar{u}$
true state evolution	$\bar{\phi}$
observation operator	$C_o : Y \to Y_o$
observations	$y = C_o \bar{\phi} + \xi_o$
background function	$u_b = \bar{u} + \xi_b$
background error	ξ_b
observation error	ξ_o

Discrete least-squares problem

- observations distributed within time interval (t_0, t_n)
- find $\mathbf{u}$ which minimises

$$
\begin{aligned}
J(\mathbf{u}) &= \frac{1}{2}(\mathbf{u}-\mathbf{u}_b)^T V_b^{-1}(\mathbf{u}-\mathbf{u}_b) \\
&+ \frac{1}{2}\sum_{i=0}^{N}(C_o(\mathbf{u}_i)-\mathbf{y}_i)^T V_o^{-1}(C_o(\mathbf{u}_i)-\mathbf{y}_i)
\end{aligned}
$$

subject to $\mathbf{u}_i$, $i = 1, \ldots, N$ satisfying

$$
\mathbf{u}_{i+1} = \mathcal{M}_{i,i+1}(\mathbf{u}_i), \qquad i = 0, \ldots, N-1.
$$

- discrete nonlinear evolution operator $\mathcal{M}_{i,i+1}$

Incremental 4D-Var

- Rewrite as an unconstrained minimisation problem using Lagrange's method.
- Incremental approach: linearise evolution operator and solve linearised problem iteratively.
- This involves a tangent linear model (TLM) and its adjoint.
- Each iteration requires one forward solution of the TLM equations and one backward solution of the adjoint equations.

Hessian matrix

- Hessian of the cost function:

$$\mathcal{H} = V_b^{-1} + R^T C_o^T V_o^{-1} C_o R.$$

- Discrete tangent linear operator R and its adjoint.
- $\mathcal{H}$ is often too large to be stored in memory.
- Action of applying $\mathcal{H}$ to a vector is available, but expensive:
  - involves both forward and backward solves with the linearised evolution operator and its adjoint.

Approximating the inverse Hessian

Why approximate $\mathcal{H}^{-1}$?

- $\mathcal{H}^{-1}$ represents an approximation of the Posterior Covariance Matrix (PCM).
- The PCM can be used to find confidence intervals and carry out *a posteriori* error analysis.
- $\mathcal{H}^{-1/2}$ can be used in ensemble forecasting.
- $\mathcal{H}^{-1}$, $\mathcal{H}^{-1/2}$ can be used for preconditioning in a Gauss-Newton method (focus of this talk).

AIM: construct a limited-memory approximation to $\mathcal{H}^{-1}$ using only matrix-vector multiplication.

Return to 4D-Var

- Linear system (within a Gauss-Newton method):

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

Hessian of the cost function $\mathcal{H}$
gradient of the cost function $G(\mathbf{u}_k)$

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

- Convergence depends on eigenvalues of the Hessian

$$\mathcal{H} = V_b^{-1} + R^T C_o^T V_o^{-1} C_o R.$$

- Evaluating $\mathcal{H}\mathbf{v}$ is very expensive, so we need a good preconditoner.

First level preconditioning

- Use the background covariance matrix V_b.

- Projected Hessian:

$$H = (V_b^{1/2})^T \mathcal{H} V_b^{1/2} = I + (V_b^{1/2})^T R^T C_o^T V_o^{-1} C_o R V_b^{1/2}$$

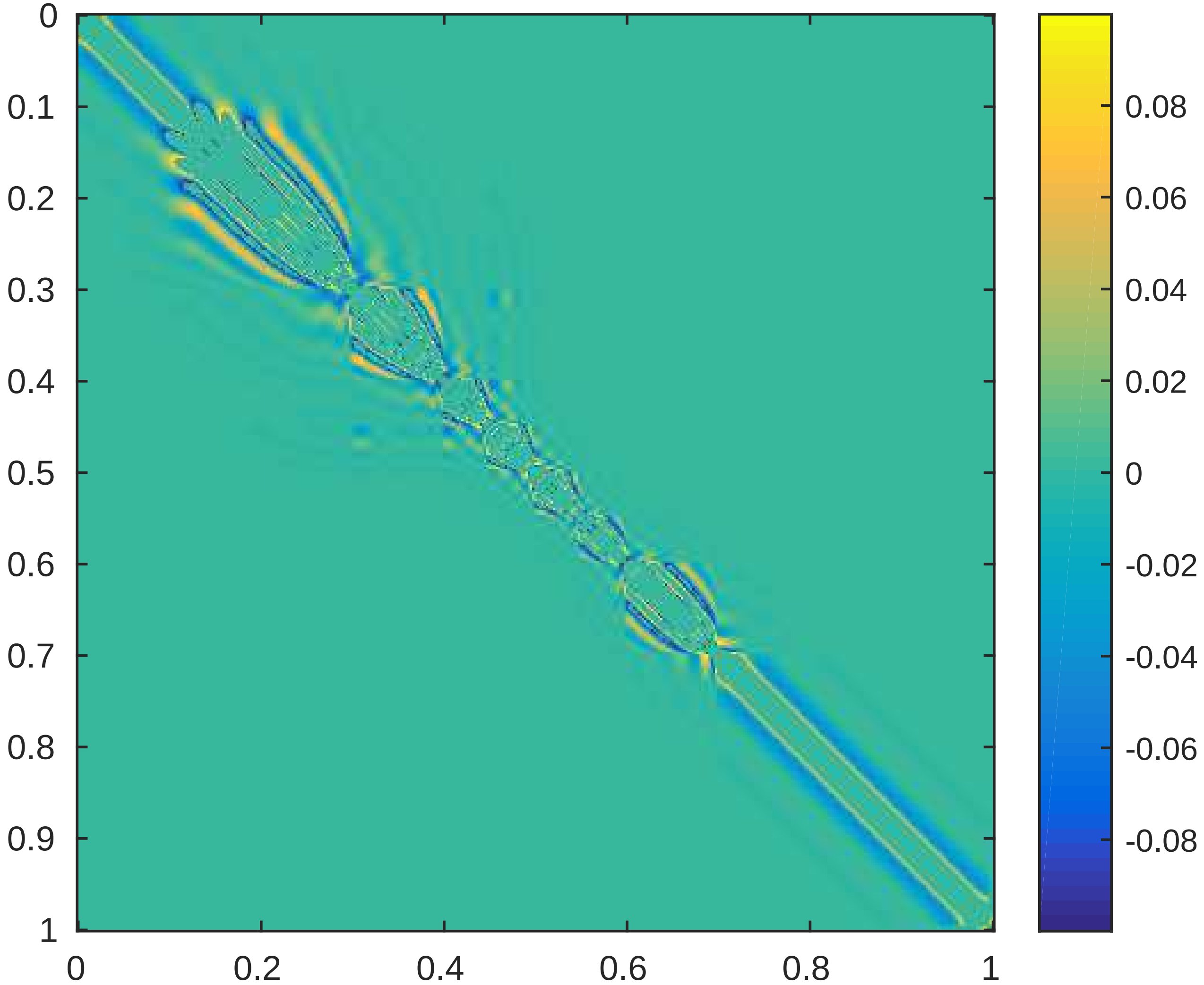
- Easy to recover $\mathcal{H}$ in the original space.

- Eigenvalues of H are usually clustered in a narrow band above one, with few eigenvalues distinct enough to contribute noticeably to the Hessian value.

- This makes $\mathcal{H}$ amenable to limited-memory approximation.

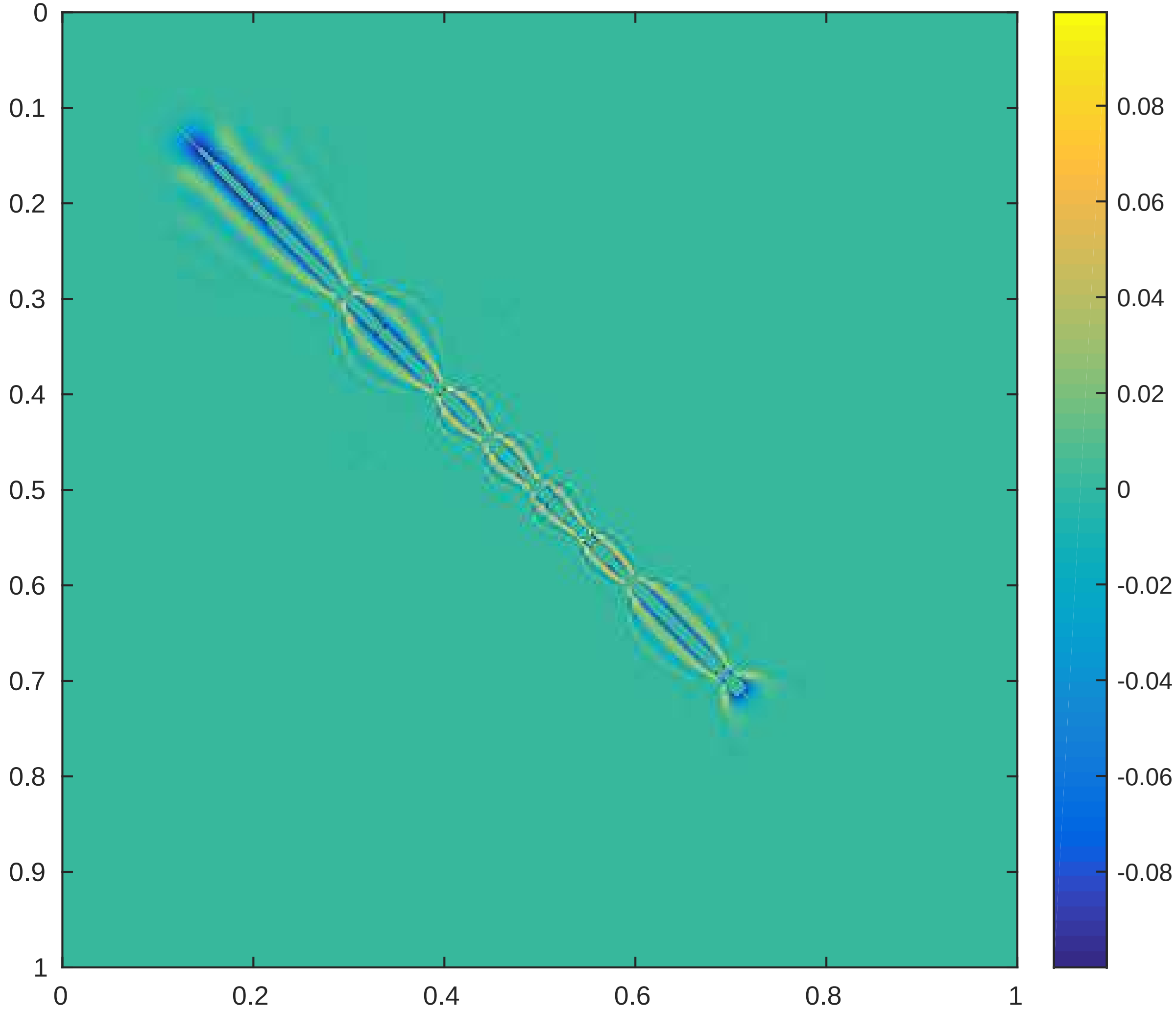
Correlation matrix

- inverse Hessian scaled to have unit diagonal



Preconditioned correlation matrix

- after first level preconditioning has been applied



Limited-memory approximation

- Find n_e leading eigenvalues and orthonormal eigenvectors using the Lanczos method.

- Construct approximation

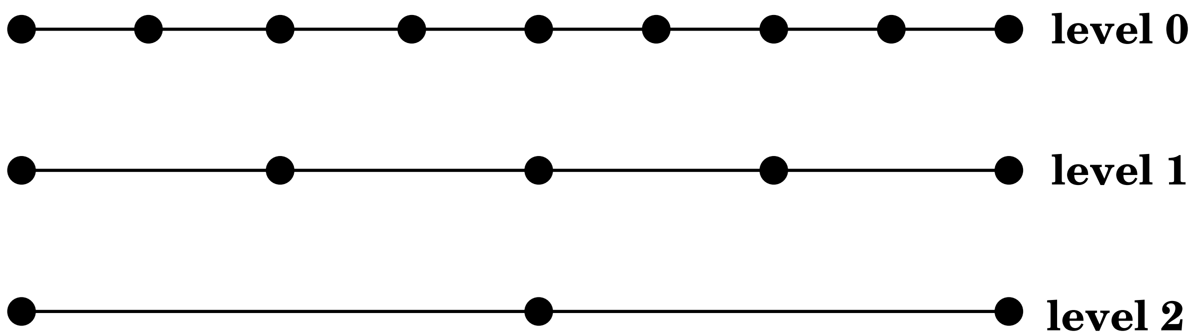
$$H \approx I + \sum_{i=1}^{n_e} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

- Easy to evaluate matrix powers:

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

Second level preconditioning

- Construct a multilevel approximation to H^{-1} based on coarser grids (where it is cheaper to use Lanczos).
- 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:

$$M_{c \to f} = P_f^c (M_c - I_c) R_c^f + I_f$$

  - From fine to coarse:

$$M_{f \to c} = R_c^f (M_f - I_f) P_f^c + I_c$$

Outline of multilevel algorithm

- 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 $B_k^{k+1} = I_k$ 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:

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

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

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

Finest level

- We already have H_0, so precondition to obtain

$$\tilde{H}_0 = {B_0^1}^T H_0 B_0^1$$

- Find n_0 eigenvalues/eigenvectors of $\tilde{H}_0$ using the Lanczos method.

- Approximate $\tilde{H}_0^{-1}$:

$$\tilde{H}_0^{-1} \approx I_k + \sum_{i=1}^{n_0} \left(\frac{1}{\lambda_i} - 1 \right) \mathbf{u}_i \mathbf{u}_i^T$$

- Recover projected inverse Hessian using

$$H_0^{-1} = B_0^1 \tilde{H}_0^{-1} {B_0^1}^T$$

Algorithm

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

```
[Λ,𝒰]=mlpre(H_0,n_0, n_1, ..., n_c)
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 preconditioners B_{k,k+1} and B^T_{k,k+1}
end
```

- storage:

$$\Lambda = \left[\lambda^1_{k_c}, \ldots, \lambda^{n_{k_c}}_{k_c}, \lambda^1_{k_c-1}, \ldots, \lambda^{n_{k_c-1}}_{k_c-1}, \ldots, \lambda^1_0, \ldots, \lambda^{n_0}_0\right],$$

$$\mathcal{U} = \left[U^1_{k_c}, \ldots, U^{n_{k_c}}_{k_c}, U^1_{k_c-1}, \ldots, U^{n_{k_c-1}}_{k_c-1}, \ldots, U^1_0, \ldots, U^{n_0}_0\right].$$

Example

- Test using 1D Burgers' equation with initial condition

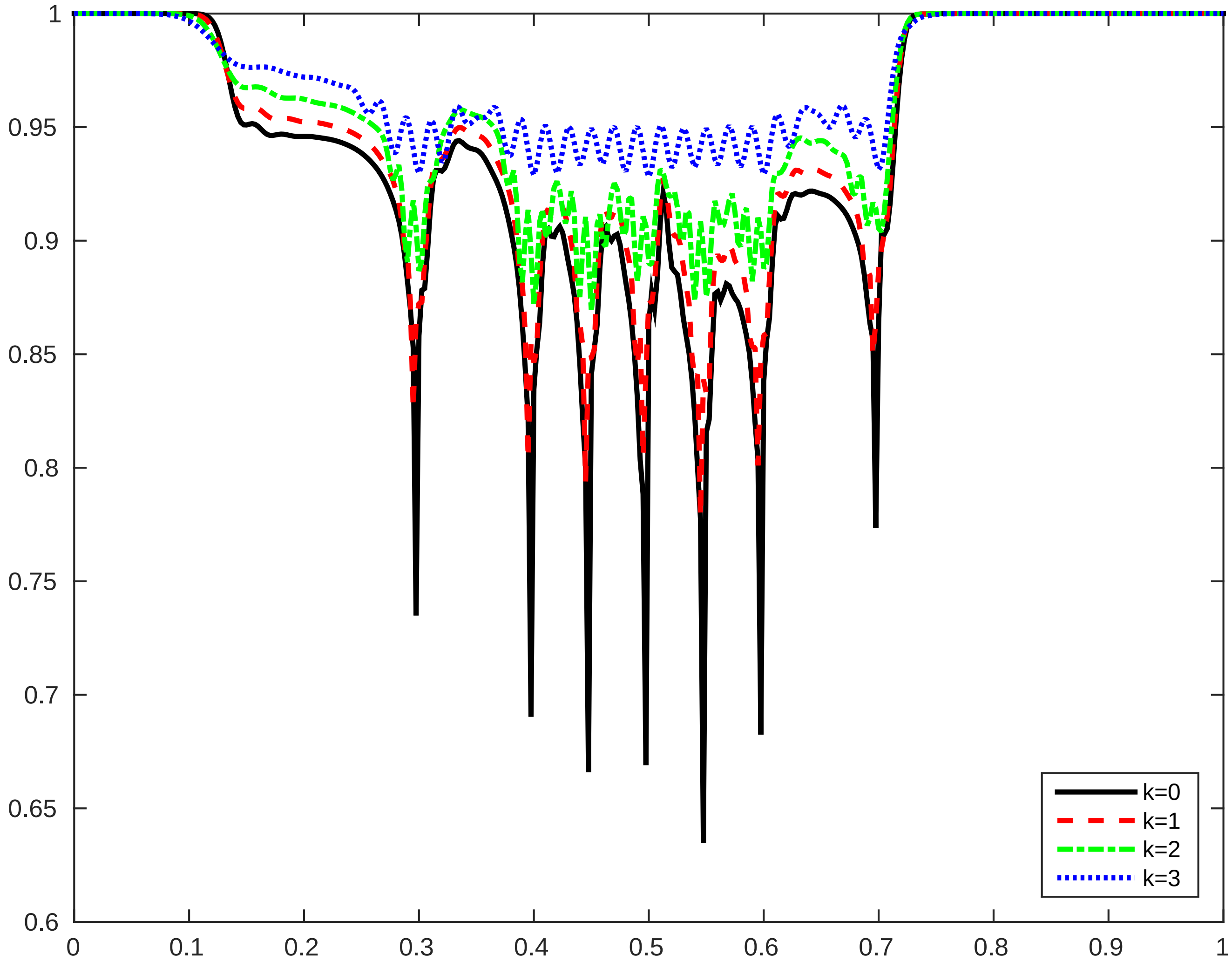
$$f(x) = 0.1 + 0.35\left[1 + \sin\left(4\pi x + \frac{3\pi}{2}\right)\right], \qquad 0 < x < 1$$

- 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

Diagonal of H^{-1}



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

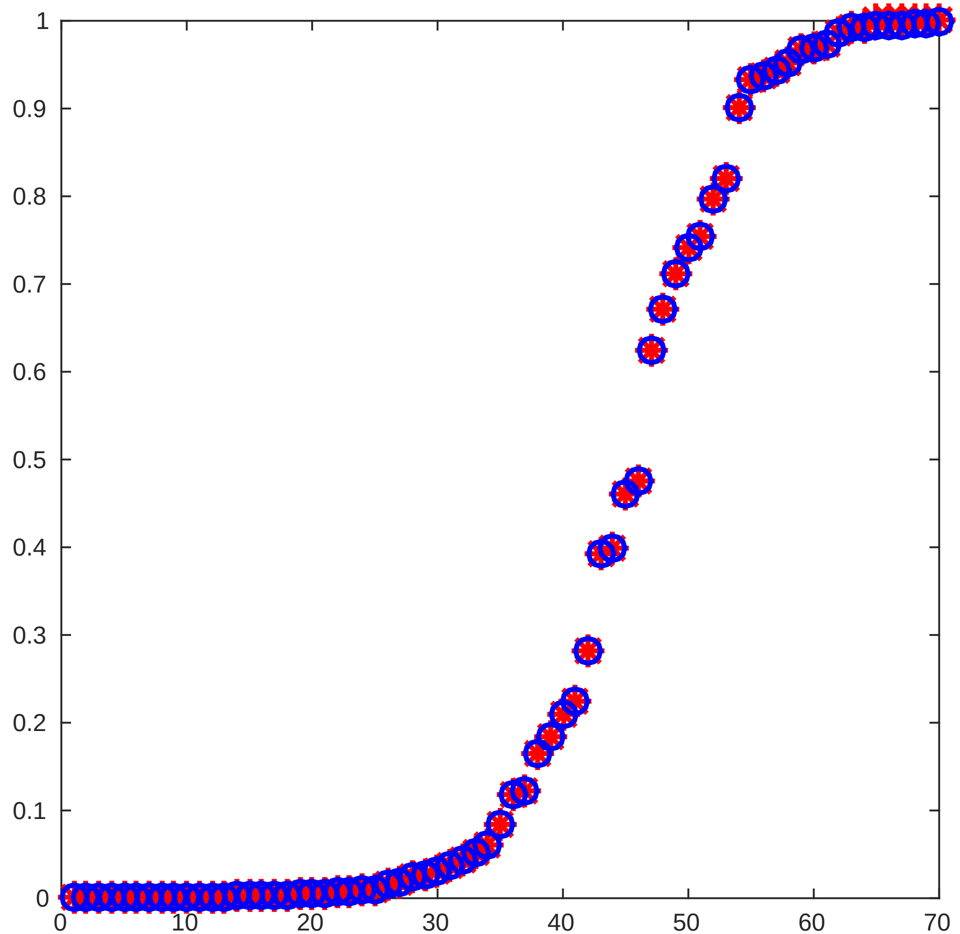
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

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

Eigenvalues of the inverse Hessian

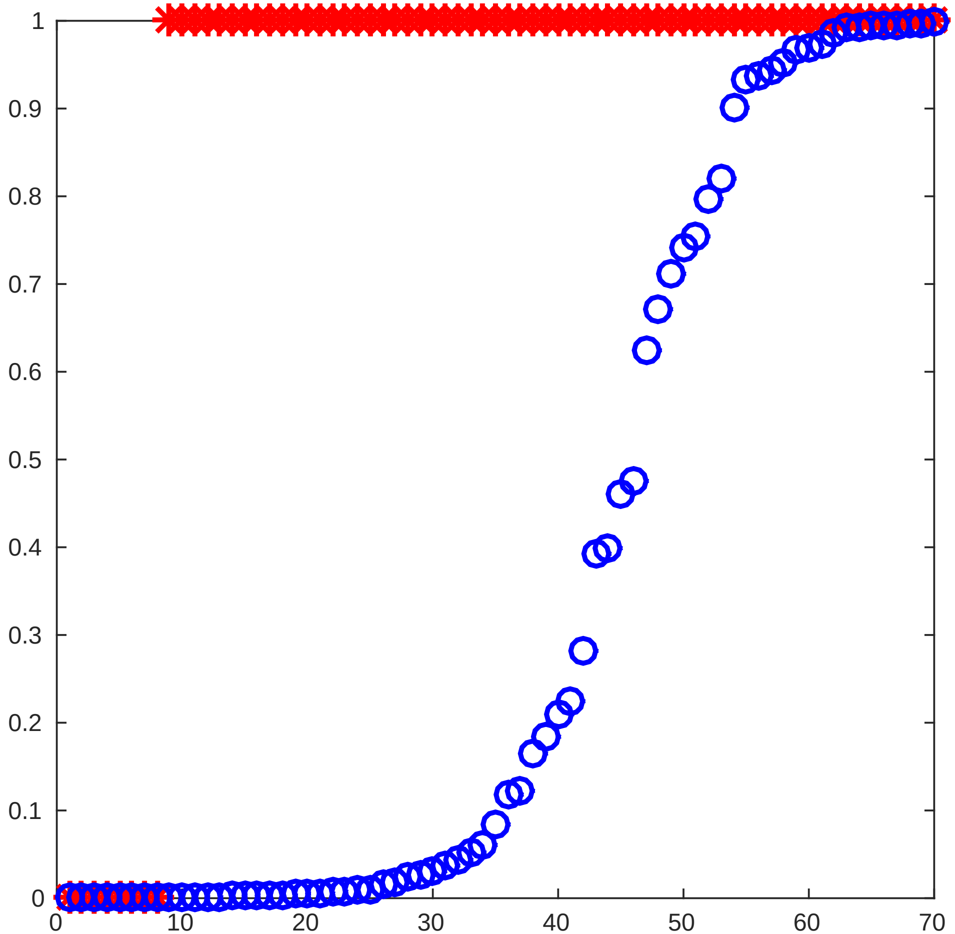
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

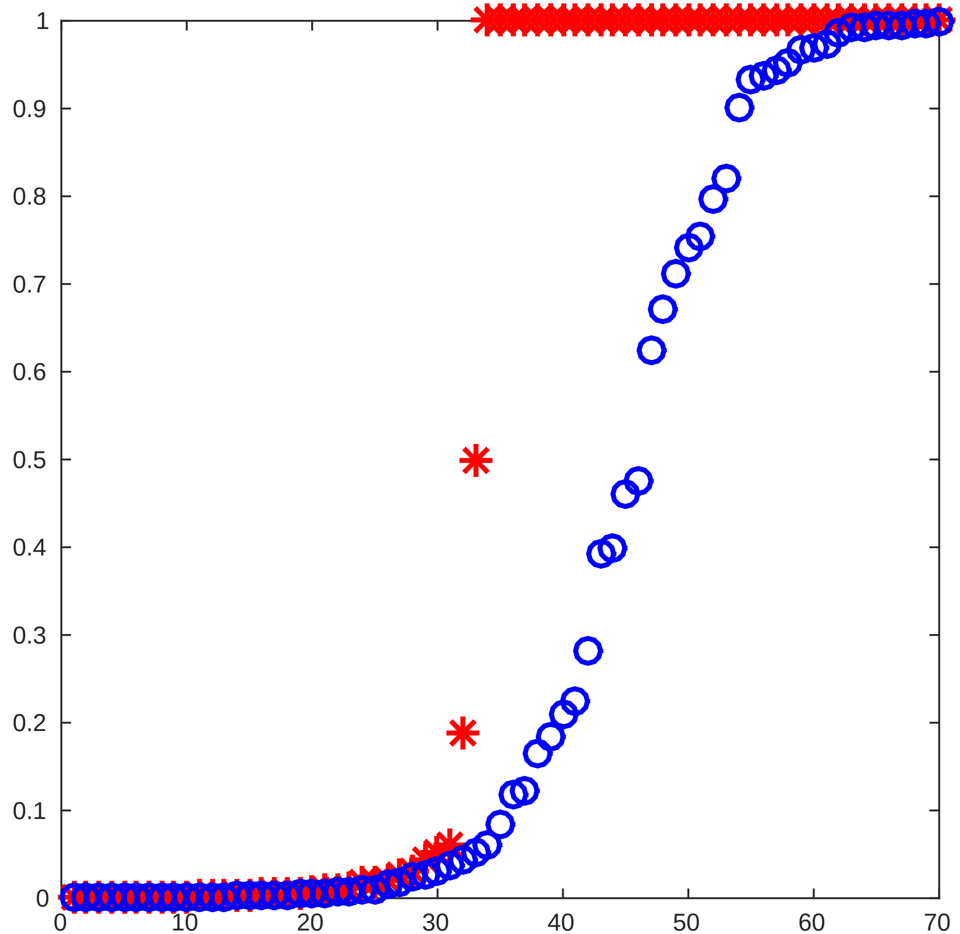
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

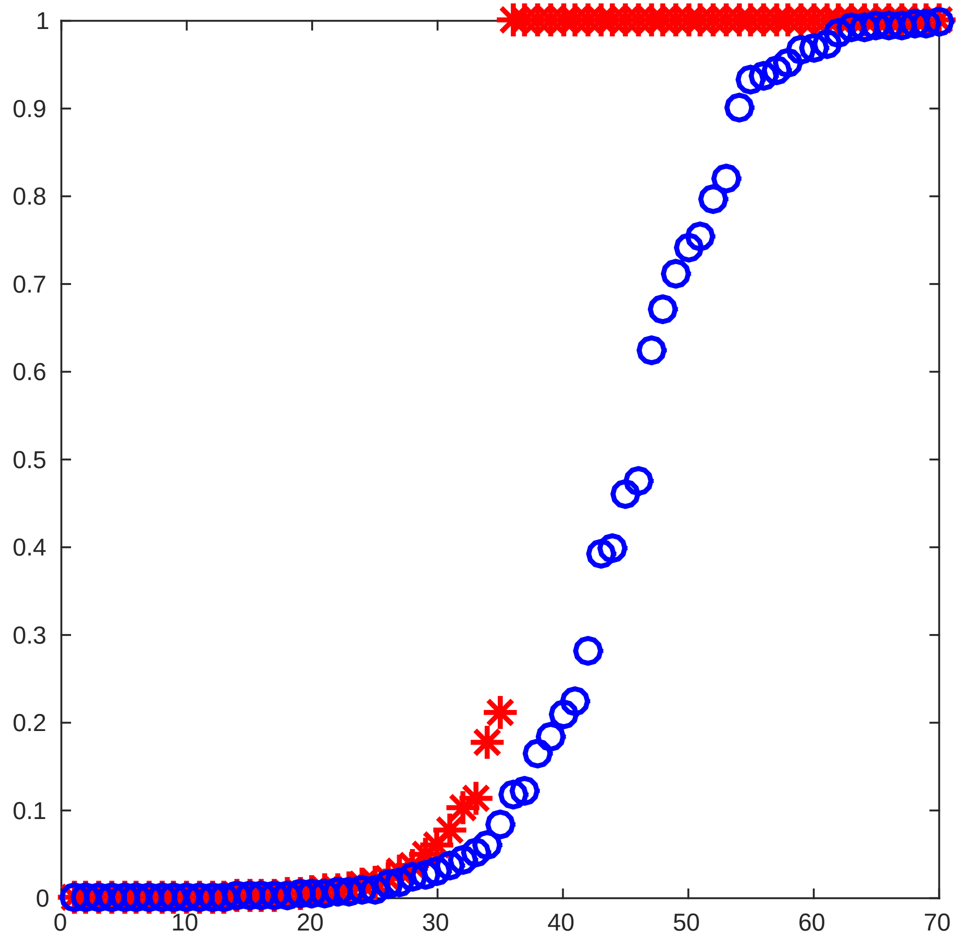
- Exact (blue circles), approximated (red stars)



$$N_e = (0, 6, 13, 14)$$
$$D = 3.95e - 1$$

Eigenvalues of the inverse Hessian

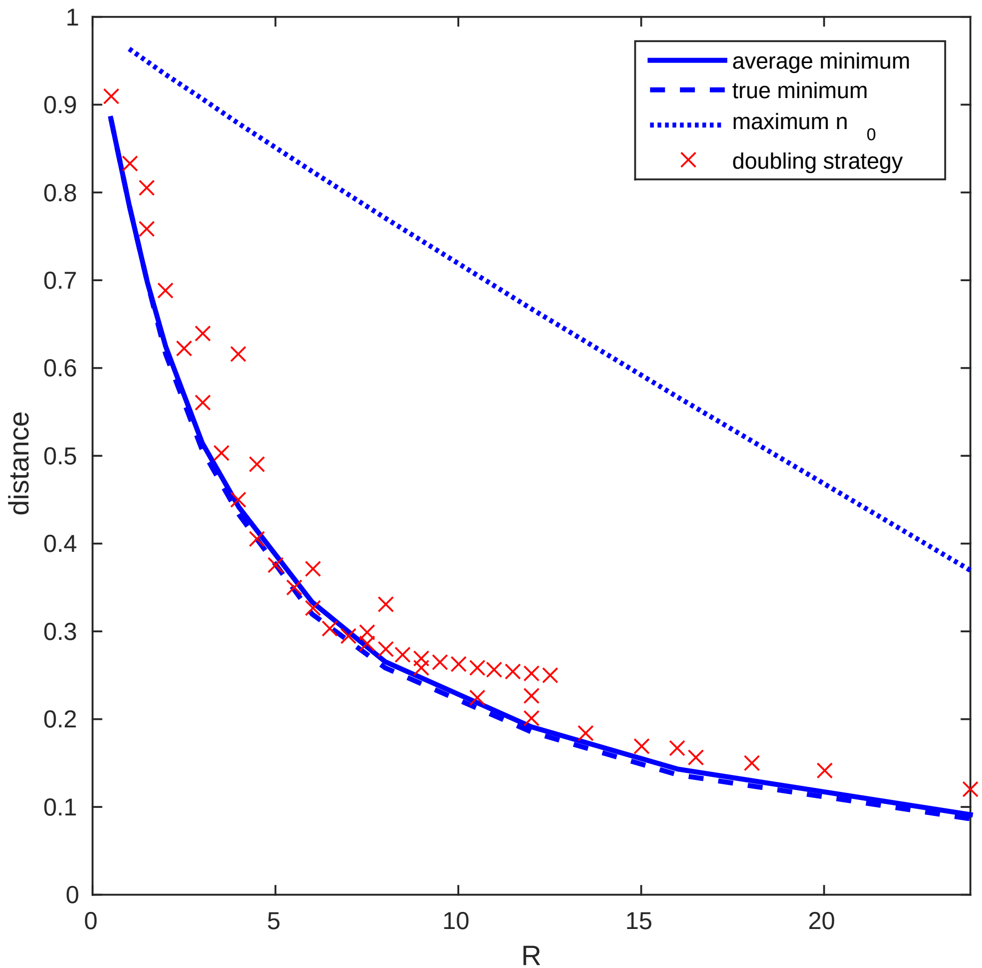
- Exact (blue circles), approximated (red stars)



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

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 MVM on finest grid M

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

Practical approach: version 1

- Assemble local Hessians for each sensor to form H_a, then apply mlpre to H_a.

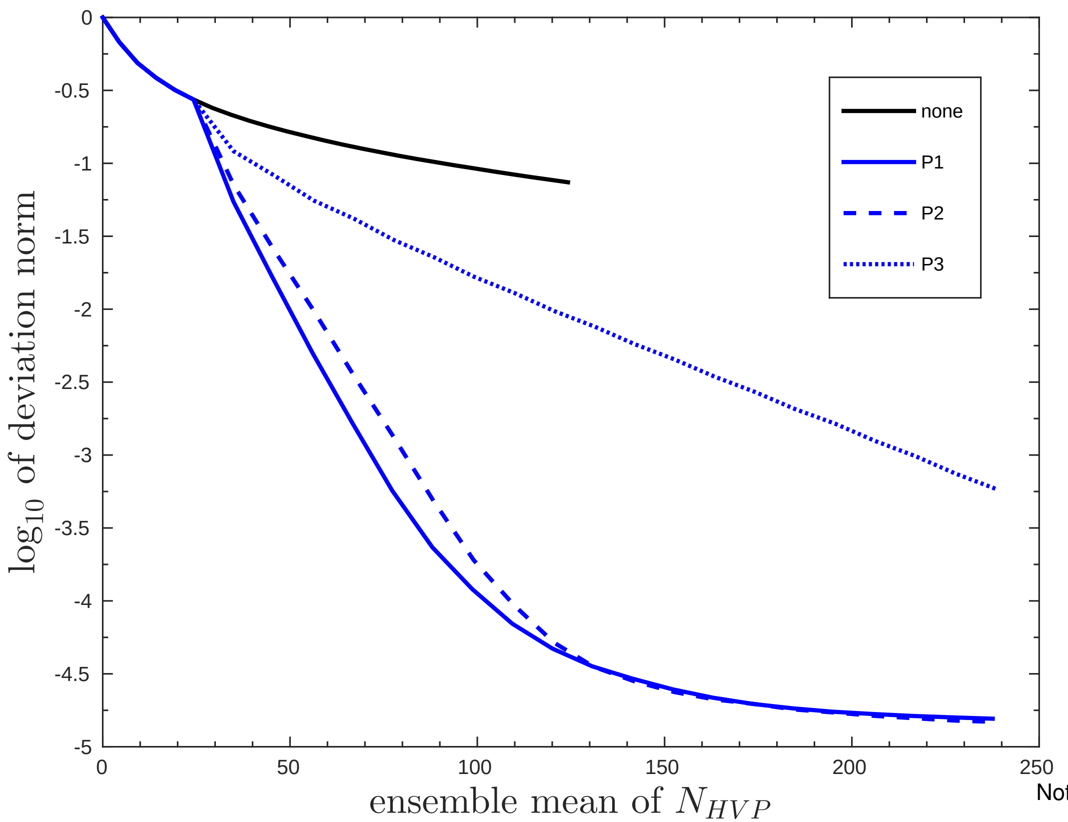
- Local Hessians cheaper to compute:
  - Potentially smaller area of influence.
  - Could run local rather than global model.
  - Compute local Hessians at level l.
  - Use limited-memory form with n_l eigenpairs.
  - Can be computed in parallel.

- More memory required:
  - Need to store additional local Hessians.

Iteration counts

Preconditioner	N_e	l	n_l
P1	(200,0,0,0)	1	8
P2	(0,8,16,32)	1	8
P3	(0,4,8,16)	1	8

log(error) vs number of HVP



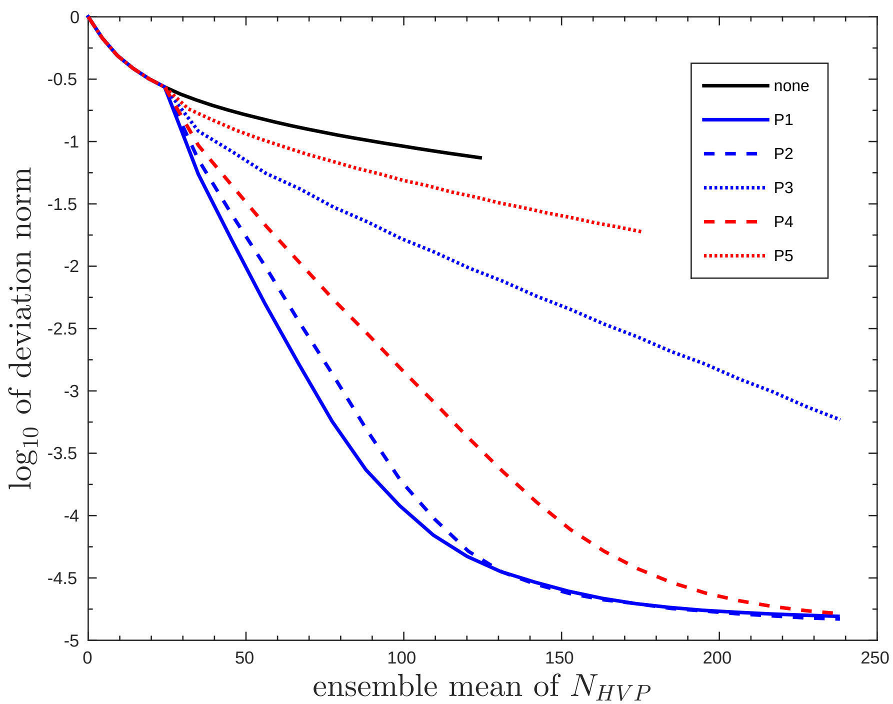
Practical approach: version 2

- Can reduce memory requirements further.
- Approximate local Hessians by applying mlpre to local inverse Hessians using N_e^l.
- Construct a reduced-memory assembled Hessian H_a^{rm}.
- Use mlpre again on H_a^{rm}.

Iteration counts

Preconditioner	N_e	l	n_l	N_e^l
P1	(200,0,0,0)	1	8	-
P2	(0,8,16,32)	1	8	-
P3	(0,4,8,16)	1	8	-
P4	(0,8,16,32)	1	8	(0,0,8,0)
P5	(0,8,16,32)	2	8	(0,0,0,8)

log(error) vs number of HVP



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) is tricky.

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) is tricky.

- Now ready for two dimensions!

It is sometimes nice in Scotland. . .