# A multilevel preconditioner for data assimilation with 4D-Var

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

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

# Data assimilation

- Combine observational and background data with numerical models to obtain the best estimate of state of a system.
- 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 $\quad \mathbf{u}_{i+1} = \mathcal{M}_{i,i+1}(\mathbf{u}_i), \quad i = 0,\ldots,N-1.$

- Discrete nonlinear evolution operator $\mathcal{M}_{i,i+1}$.
- Incremental 4D-Var: rewrite as an unconstrained minimisation with linearised evolution operator.

# Hessian matrix

- Linear system (Gauss-Newton method):

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

Hessian $\mathcal{H}$, gradient $G(\mathbf{u}_k)$

- PCG convergence depends on conditioning of

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

- Discrete tangent linear operator $R$ and its adjoint.
- $\mathcal{H}$ is usually too large to be stored in memory but all we need for PCG is $\mathcal{H}\mathbf{v}$.
- This is still very expensive to compute, so we also need a good preconditioner.

# First level preconditioning

- Projected Hessian:

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

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

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

# Limited-memory approximation

- Find $n_e$ leading eigenvalues (by $\ln \lambda^2$) 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 the finest grid (level $k = 0$) to obtain Hessian $H \equiv H_0$.
- Grid transfers with "correction" between course grid level $k+1$ and a fine grid level $k$
  - Piecewise cubic splines: $R^k_{k+1}$, $P^{k+1}_k$
  - Coarse to fine:

    $$[M_{k+1}]_{\to k} = P^{k+1}_k (M_{k+1} - I_{k+1}) R^k_{k+1} + I_k$$
  - Fine to coarse:

    $$[M_k]_{\to k+1} = R^k_{k+1} (M_k - I_k) P^{k+1}_k + I_{k+1}$$

# 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 $\tilde{H}_{0\to k}^{-1}$:

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

# Preconditioners

- On coarsest grid, level $k+1$ does not exist so set $B_k^{k+1} = I_k$.

- For other levels, construct preconditioners recursively:

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

- Finest level: recover projected inverse Hessian using

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

# Summary

- Algorithm:

  $[\Lambda,\mathcal{U}]$=*mlpre*$(H_0,n_c,\ldots,n_1,n_0)$

  ```
  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
  ```

  for $k=k_c,k_c-1,\ldots,0$
  compute by the Lanczos method and store in memory $\{\lambda_k^i,U_k^i\}, i=1,\ldots,n_k$ of $\tilde{H}_{0\to k}$ using preconditioners $B_{k,k+1}$ and $B_{k,k+1}^T$
  end

- storage:

$$\begin{aligned}\Lambda &= \left[\lambda_{k_c}^1,\ldots,\lambda_{k_c}^{n_{k_c}},\lambda_{k_c-1}^1,\ldots,\lambda_{k_c-1}^{n_{k_c-1}},\ldots,\lambda_0^1,\ldots,\lambda_0^{n_0}\right],\\ \mathcal{U} &= \left[U_{k_c}^1,\ldots,U_{k_c}^{n_{k_c}},U_{k_c-1}^1,\ldots,U_{k_c-1}^{n_{k_c-1}},\ldots,U_0^1,\ldots,U_0^{n_0}\right].\end{aligned}$$

# 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 |

# 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

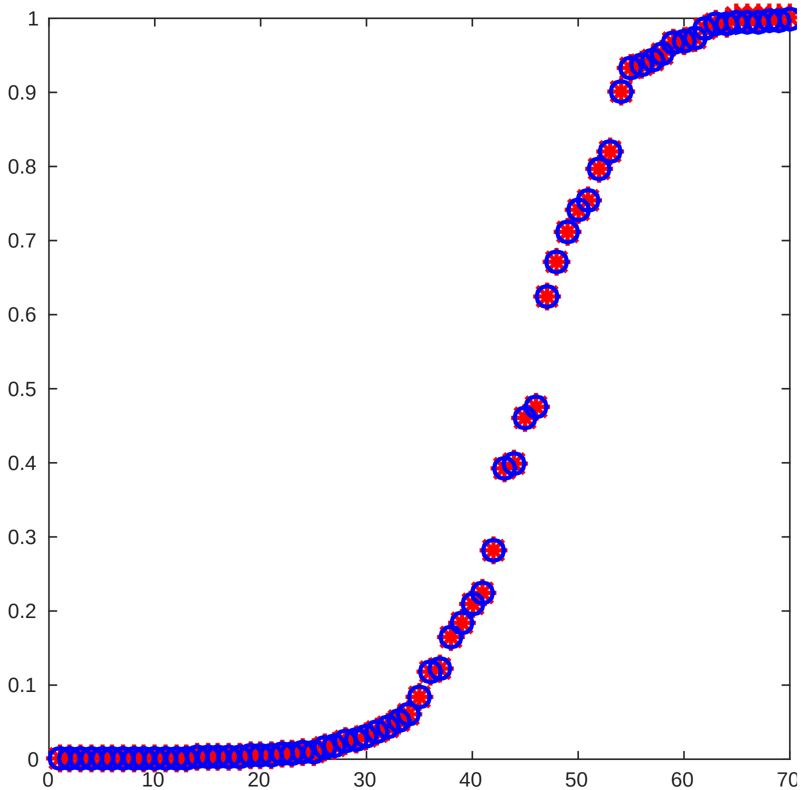
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

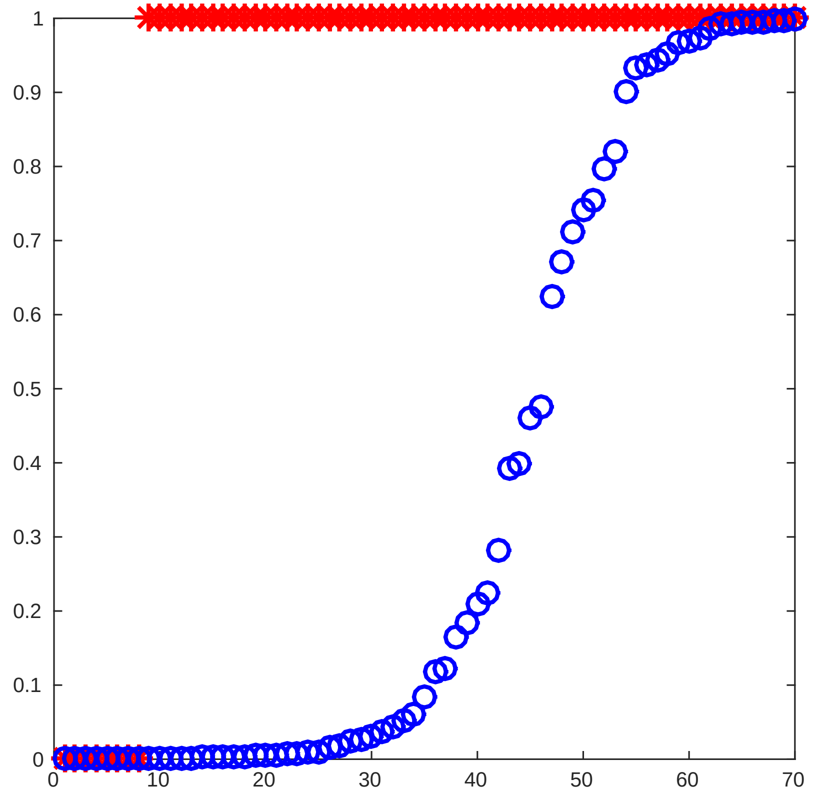
$$N_e = (n_0, n_1, n_2, n_3)$$

# Eigenvalues of the inverse Hessian

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



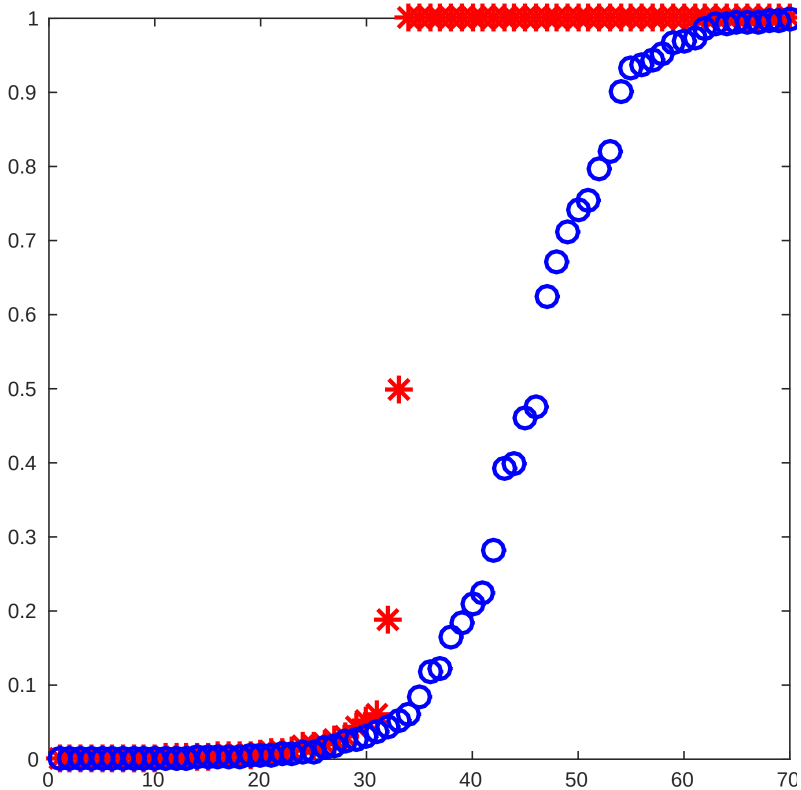

$N_e = (64, 0, 0, 0)$
$D = 2.98e - 4$



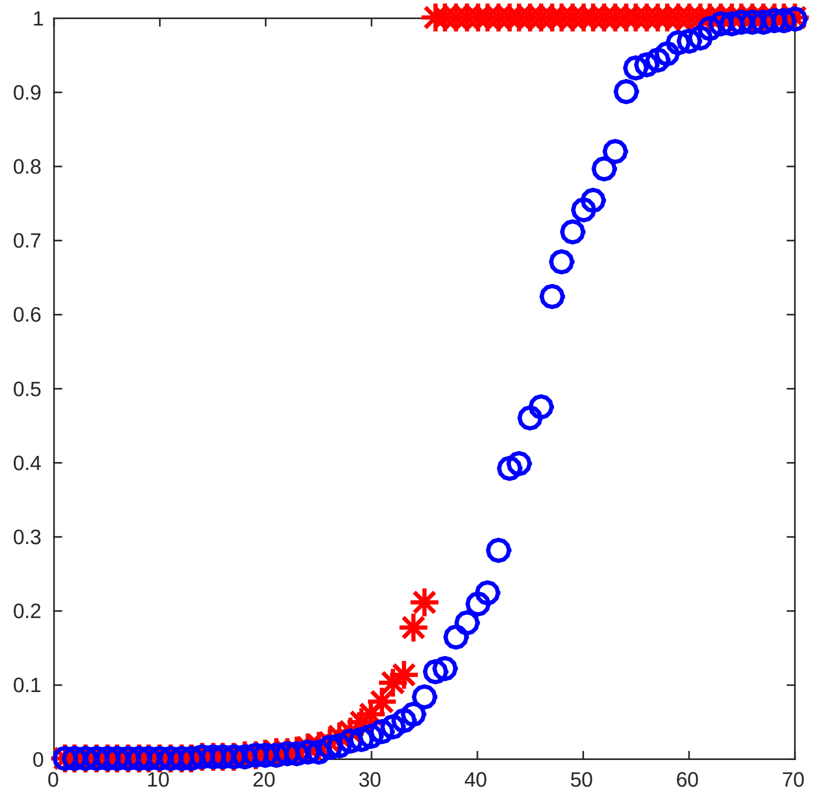

$N_e = (8, 0, 0, 0)$
$D = 7.71e - 1$

# Eigenvalues of the inverse Hessian

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



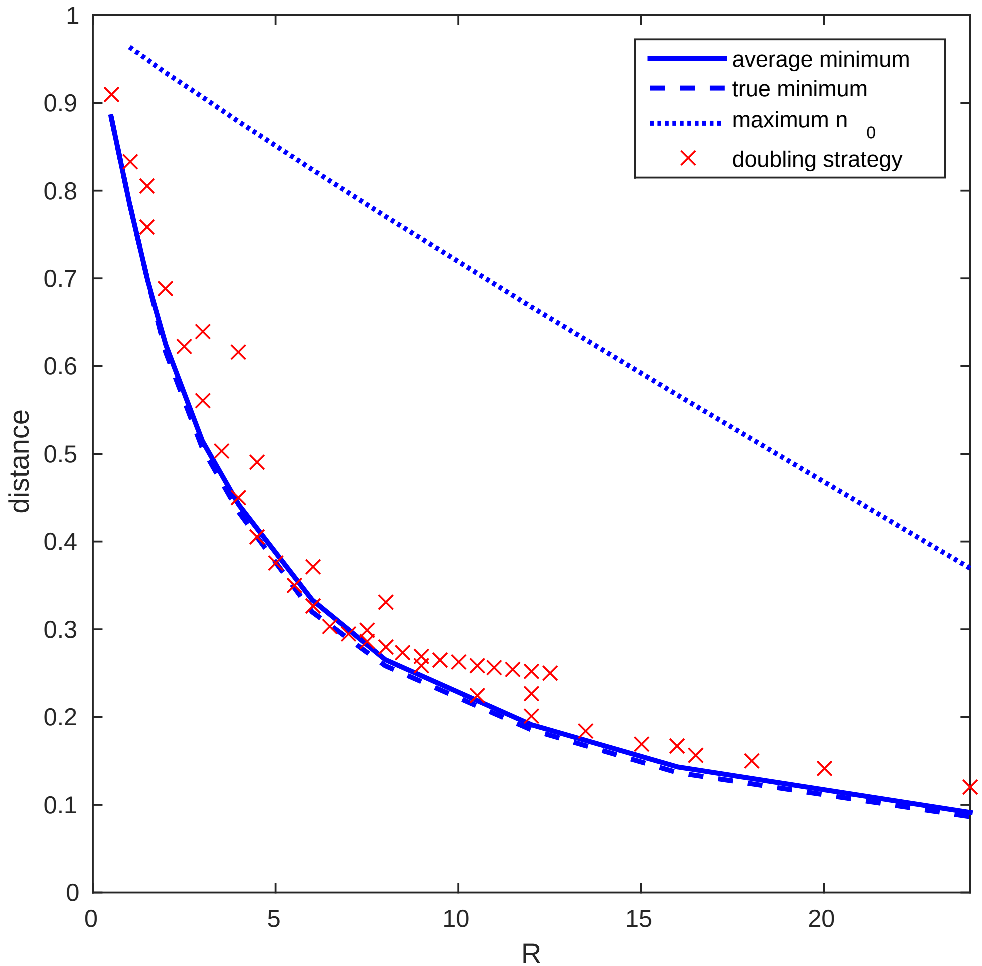

$$N_e = (0, 6, 13, 14)$$
$$D = 3.95e - 1$$




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

# Fixed memory ratio

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



# Practical approach: version 1

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

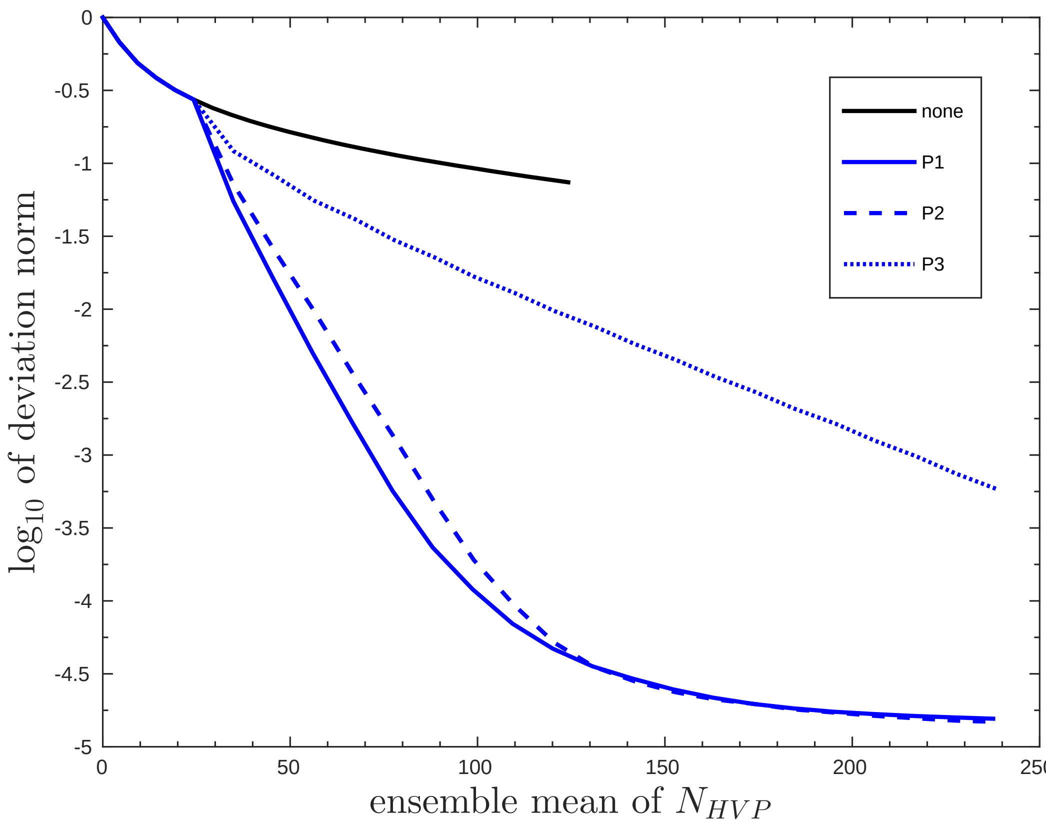
- Local Hessians cheaper to compute:
  - Potentially smaller area of influence.
  - Could run local rather than global model.
  - Compute local Hessians at level $l$.
  - Use limited-memory form with $n_l$ eigenpairs.
  - Can be computed in parallel.

- 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



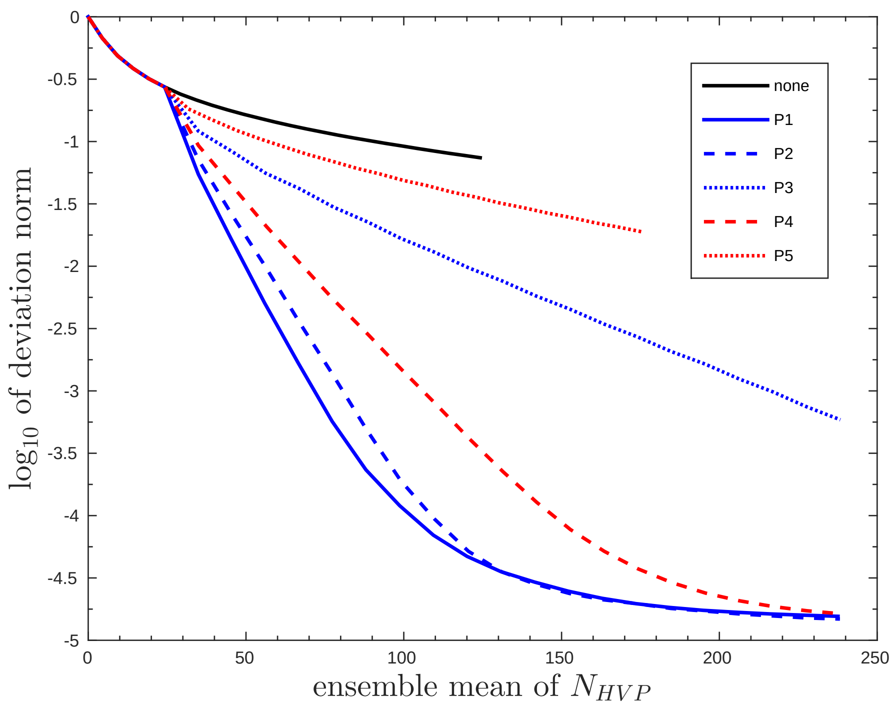
# Practical approach: version 2

- 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.
- Now ready for two dimensions!