Approximating the inverse Hessian in 4D-Var data assimilation

Alison Ramage

Department of Mathematics and Statistics

University of Strathclyde Glasgow

Collaborators: Kirsty Brown (Strathclyde), Igor Gejadze (IRSTEA, France), Amos Lawless (Reading), Nancy Nichols (Reading)

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

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

- 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).

Motivation for approximating $\mathbb{H}^{-1}$

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

- $\mathbb{H}^{-1}$ approximates Posterior Covariance Matrix (used to find confidence intervals and carry out *a posteriori* error analysis).
- $\mathbb{H}^{-1/2}$ can be used in ensemble forecasting.
- $\mathbb{H}^{-1}$, $\mathbb{H}^{-1/2}$ can be used for preconditioning in a Gauss-Newton method.

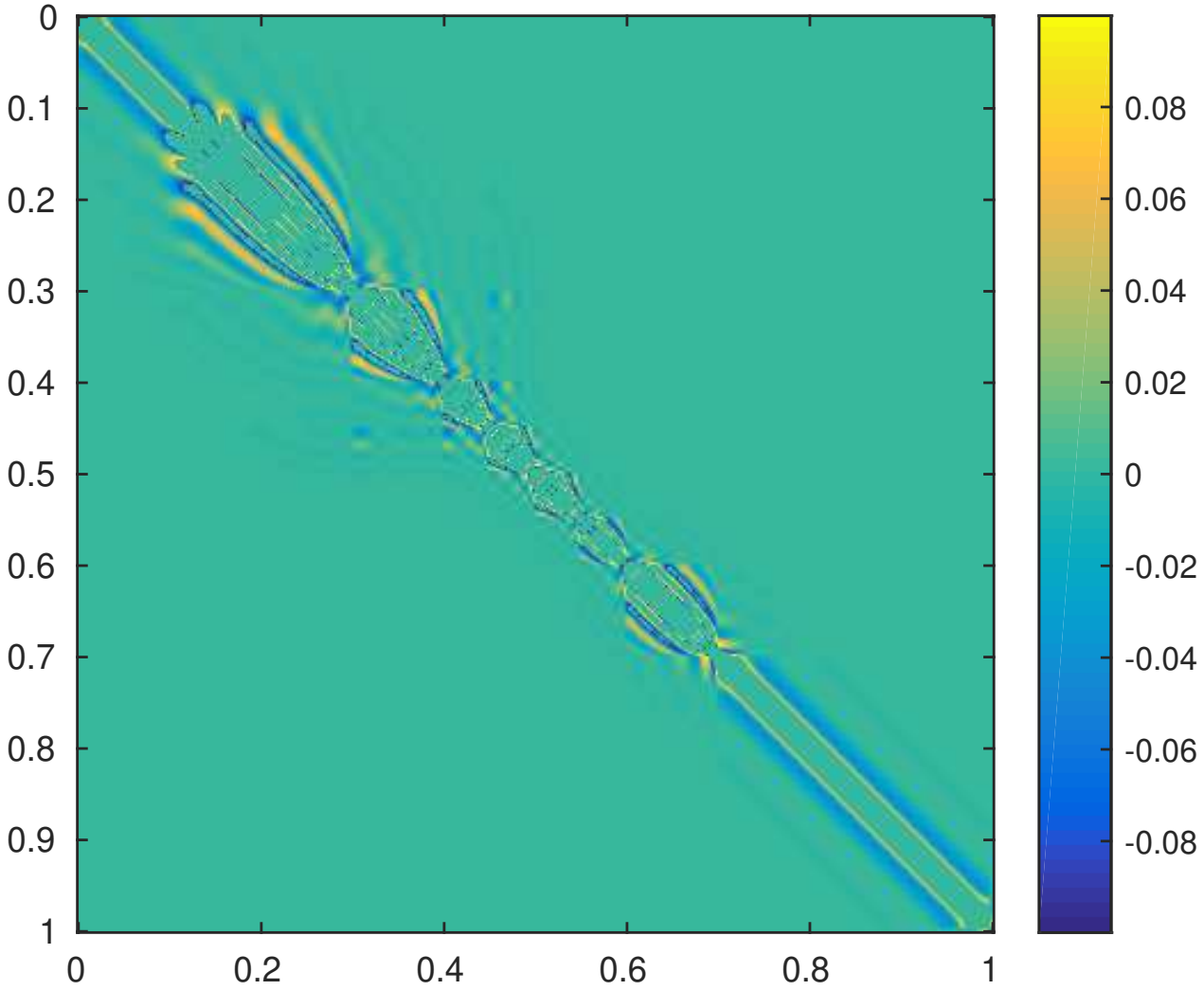
- Control variable transform: precondition $\mathbb{H}$ based on the background covariance matrix

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

- Eigenvalues of H are bounded below by one: more details on the full eigenspectrum can be found in Haben et al. (2011), Tabeart et al. (2018).

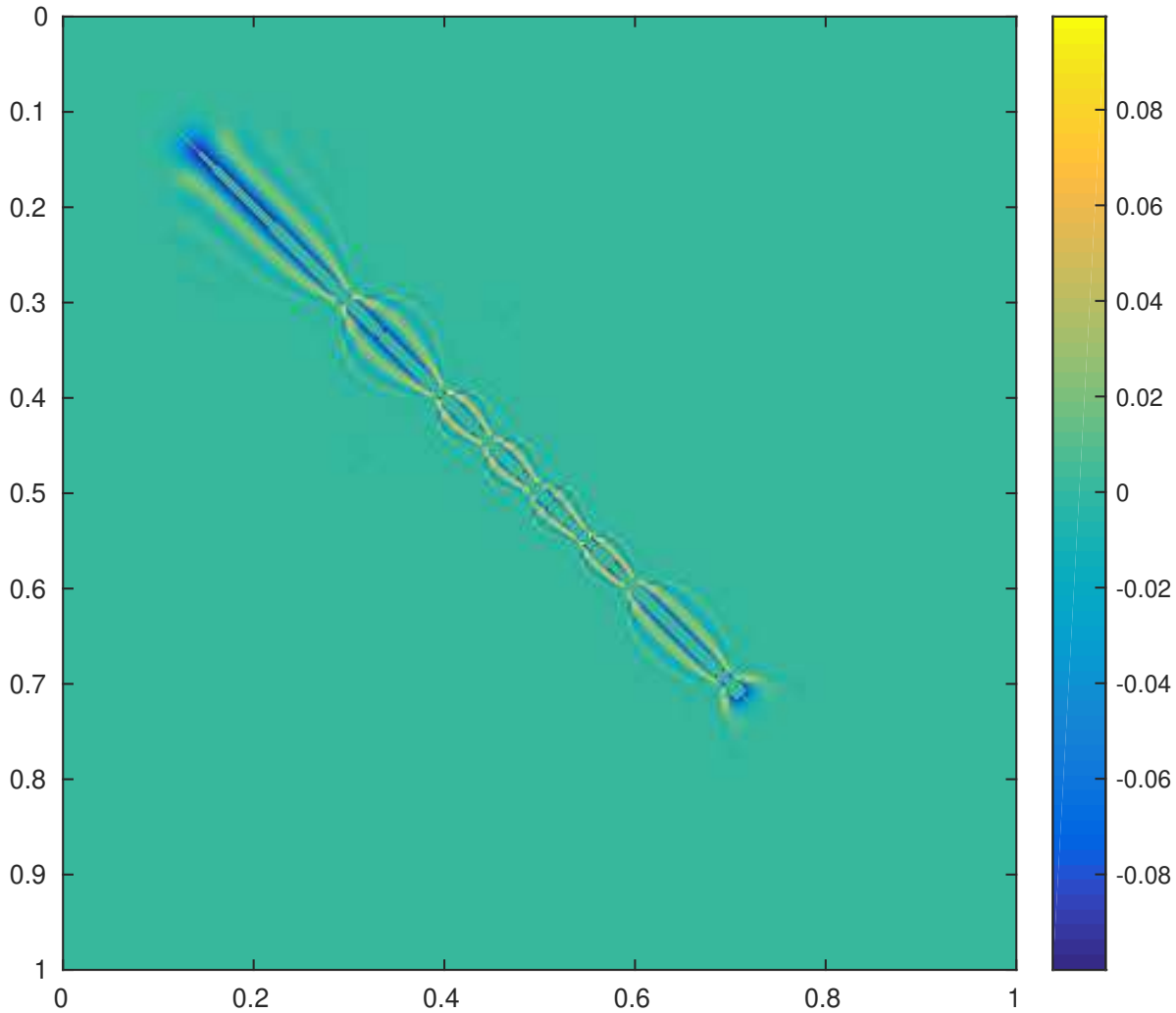
Original inverse Hessian

- $\mathbb{H}^{-1}$ (scaled to have unit diagonal, omitted from plot)



Preconditioned inverse Hessian

- H^{-1} (after control variable transform)



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 $H^{-1/2}$).

Multilevel preconditioning

- Discretise evolution equation on a grid (level k=0) to represent top level Hessian H_0.
- Build a limited-memory multilevel approximation to H^{-1} based on a sequence of nested grids (level k=1,2,...).
- Grid transfer (based on piecewise cubic splines here):
  - Restriction matrix R_c^f from $k = f$ to $k = c$.
  - Prolongation matrix P_f^c from $k = c$ to $k = f$.
- Write H_k for $[H_0]_{\to k}$ ("H_0 restricted to grid level k").
- New operators for matrix transfer:
  - From coarse to fine: $[H_c]_{\to f} = P_f^c(H_c - I_c)R_c^f + I_f$
  - From fine to coarse: $[H_f]_{\to c} = R_c^f(H_f - I_f)P_f^c + I_c$

Key idea

- Build upwards from the coarsest level.
- Assume that H_{k+1} is a good approximation to H_k.

> If H is preconditioned as $\tilde{H} = P^T H P$, then
>
> $$H^{-1} = (P\tilde{H}^{-1/2})(\tilde{H}^{-1/2}P^T) \quad \equiv \quad \hat{P}\hat{P}^T.$$

- Precondition H on one level with $\hat{P}$ from the level below.

$$[H_{k+1}^{-1/2}]_{\to k} H_k [H_{k+1}^{-1/2}]_{\to k} \quad \overset{\text{sim}}{=} \quad [H_{k+1}^{-1}]_{\to k} H_k \quad \simeq \quad I_k$$

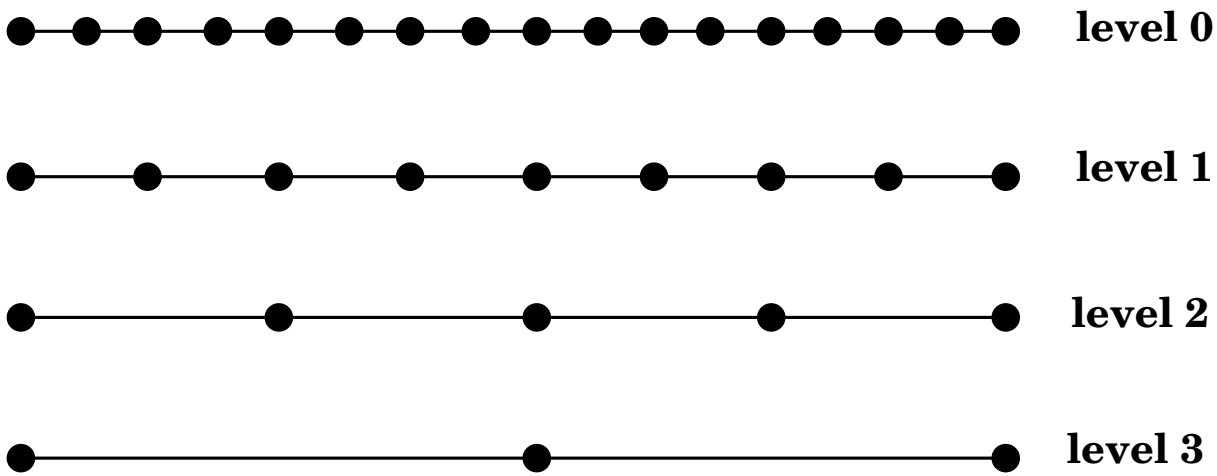
Important points

- In practice, all we need are eigenvalues and eigenvectors of each H_k.
- Limited-memory approximation means matrix powers are easy to calculate.
- Lanczos method used to compute eigenvalues: this is cheaper and requires less storage on coarser grids.
- Choose to retain $N_e = (n_0, n_1, \ldots, n_c)$ eigenvalues at each level.
- Difficult to find good values for N_e *a priori*: we have developed heuristic guidelines from practical experience.

Illustration

- 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



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

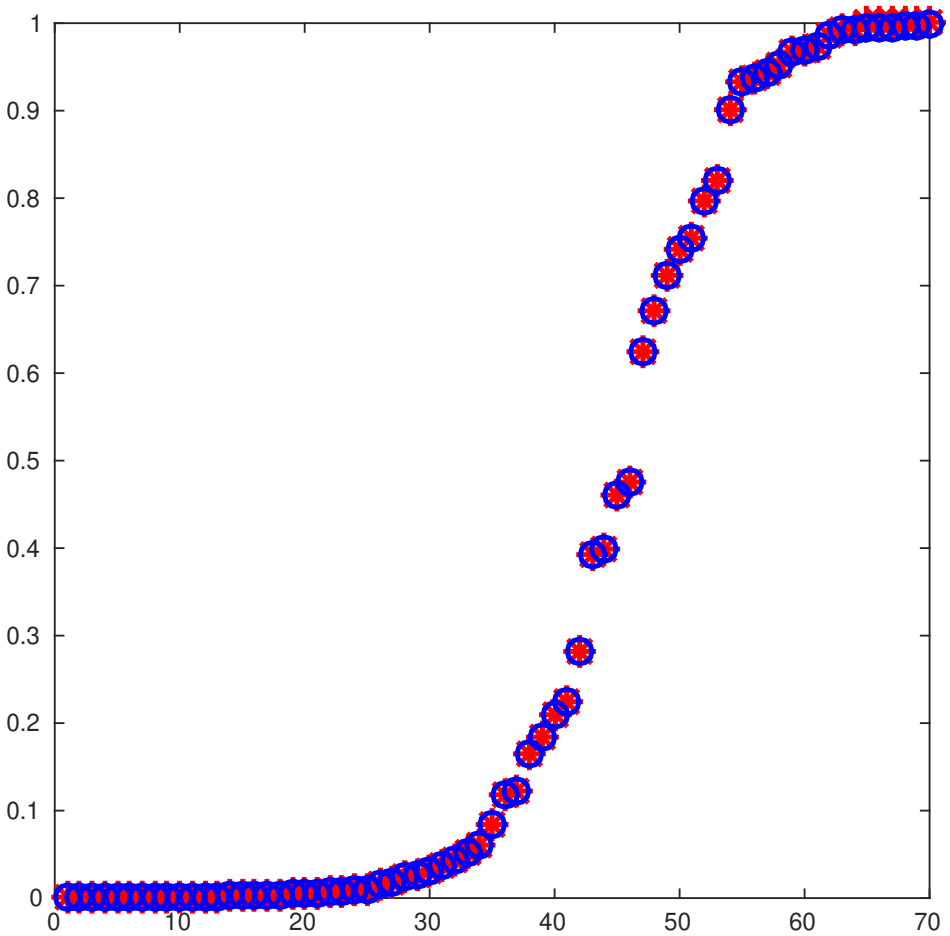
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

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

Eigenvalues of the inverse Hessian

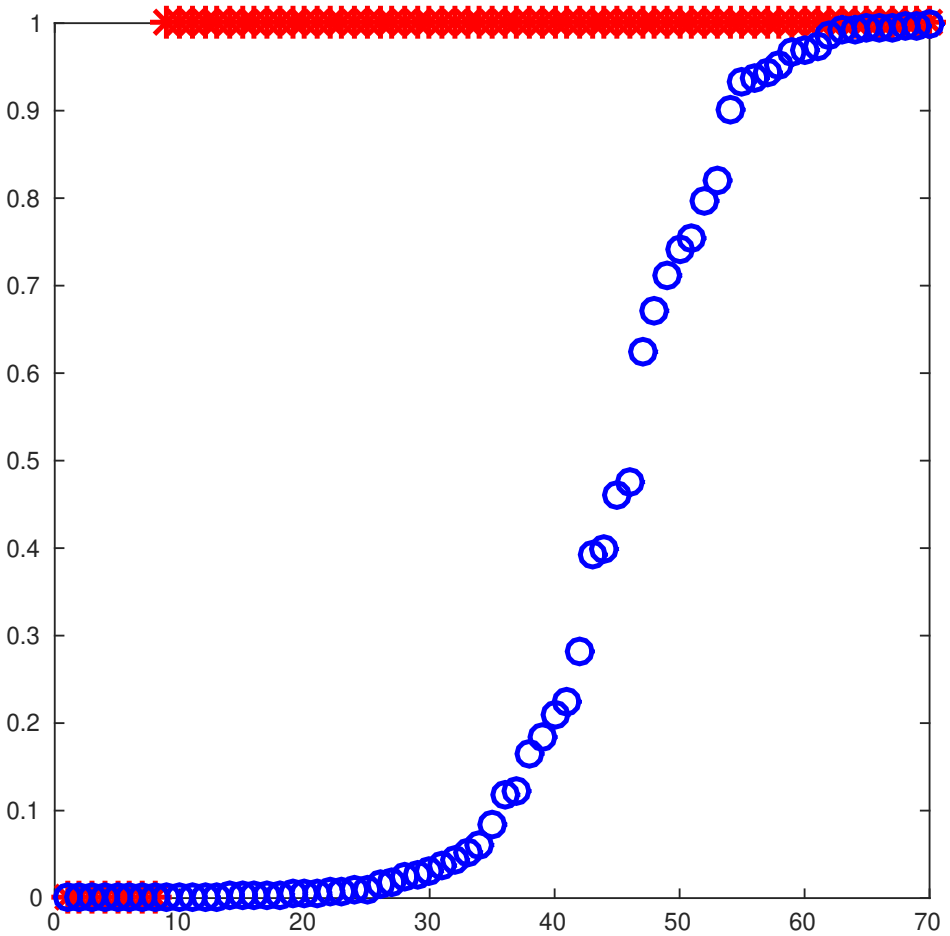
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

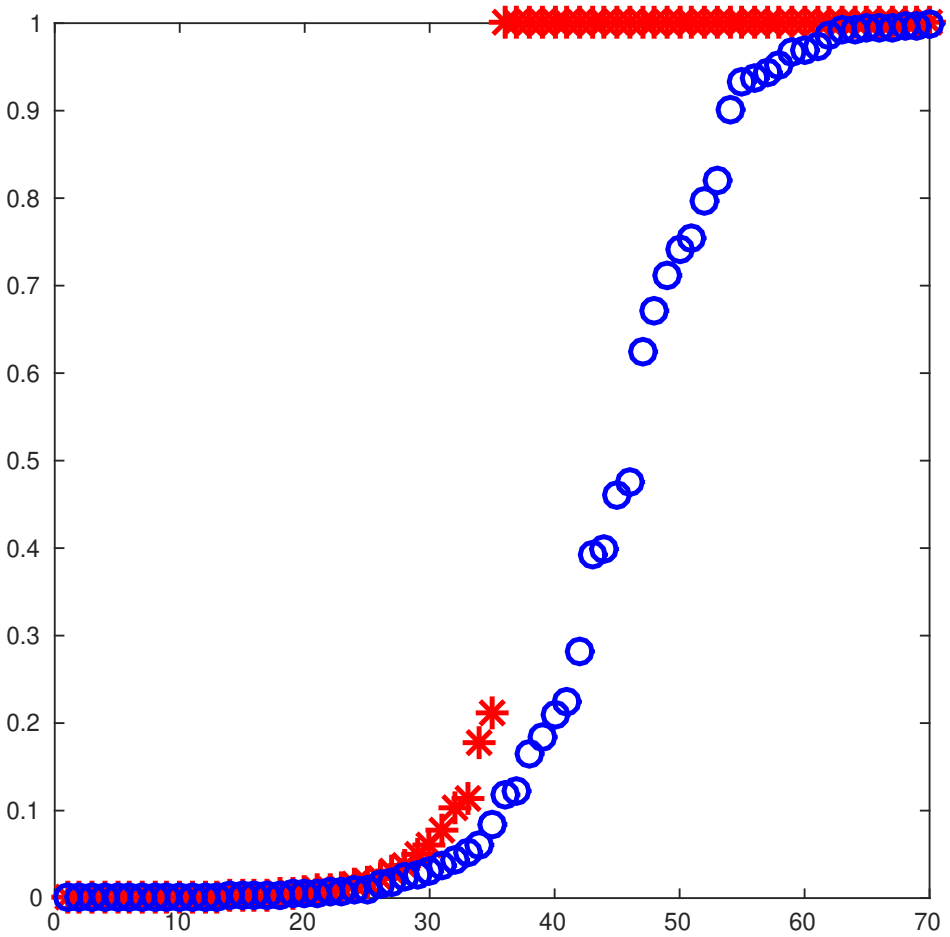
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

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



$$N_e = (0, 0, 29, 6)$$
$$D = 3.82e - 1$$

Example: PCG iteration for one Newton step

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

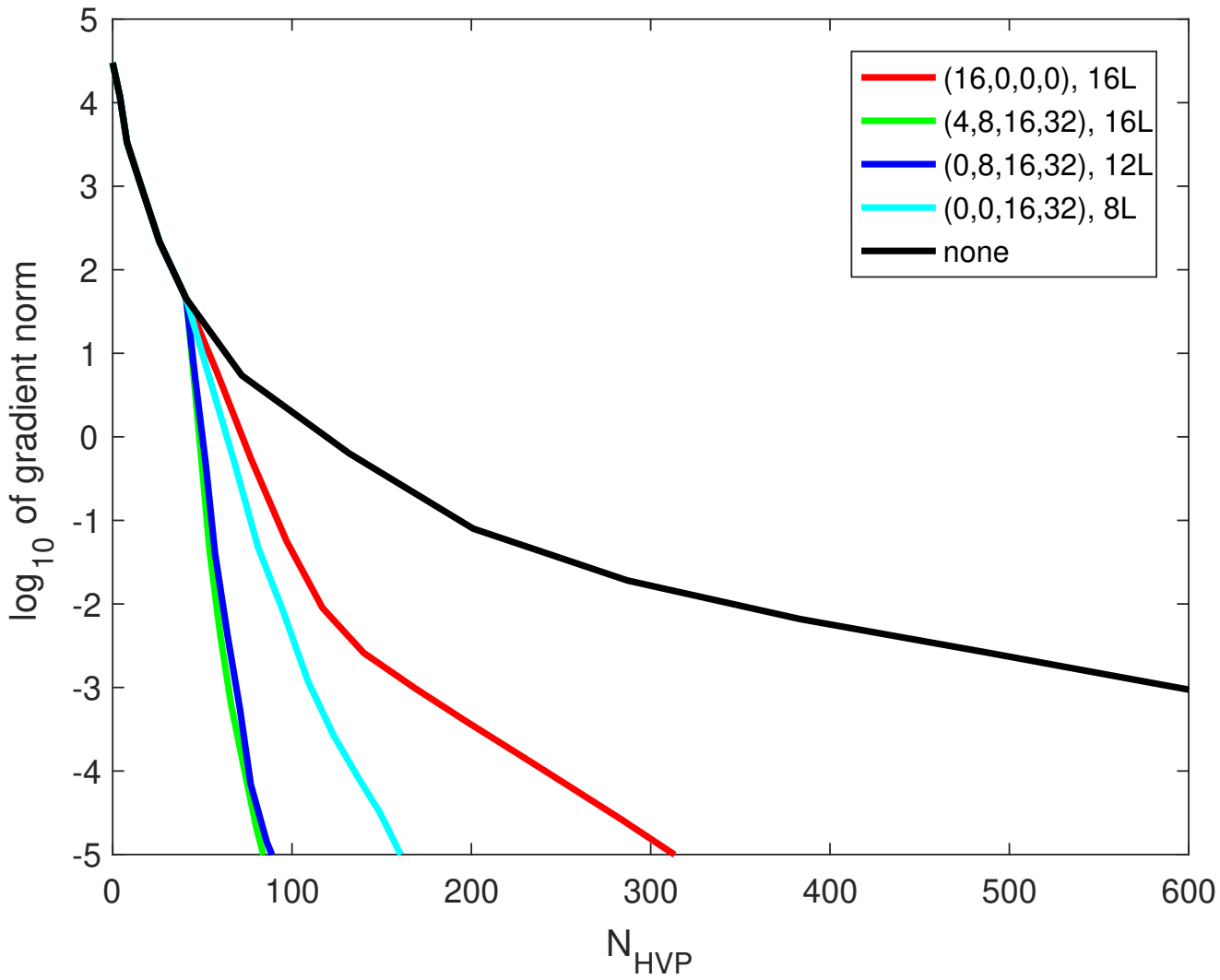
$$H(\mathbf{u}_k)\delta\mathbf{u}_k = G(\mathbf{u}_k)$$

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

- 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

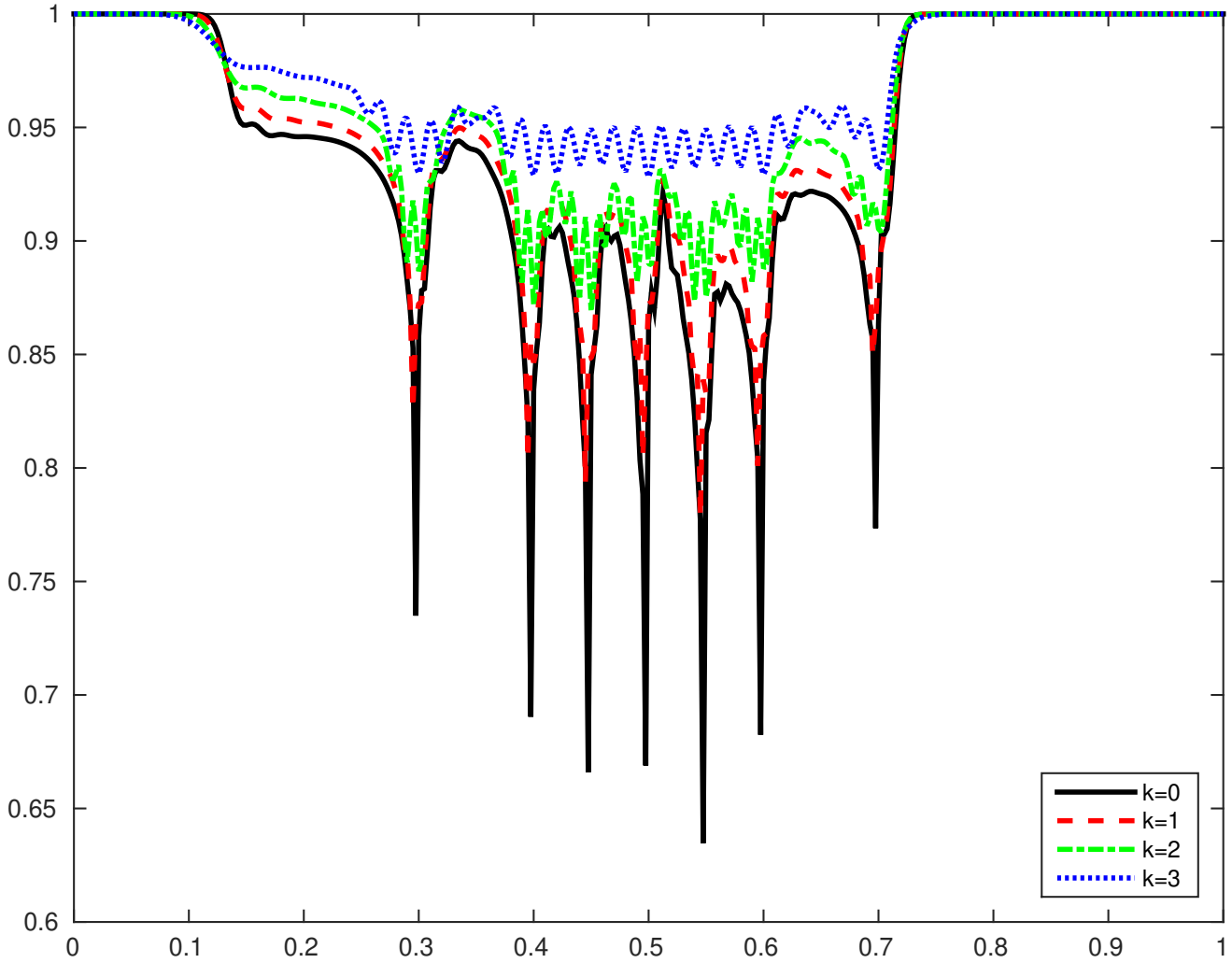


Concluding remarks

- Algorithm based solely on repeated use of Lanczos at each level (for limited-memory approximations).
- Difficult to identify the correct number of eigenvalues to use at each level, but good heuristics available.
- Full algorithm is not practical, but we have developed practical implementations based on Hessian decompositions.
- Also works well for other configurations (e.g. moving sensors, different initial conditions), and other models (e.g. 1D shallow water equations).
- Good potential for extension to higher dimensions and other applications.

Motivation for multilevel preconditioning

- Diagonal of H^{-1} on various grid levels:



Level k

- Restrict H_0 to level k to obtain H_k.
- Use preconditioner from previous level:

$$P_k = [P_{k+1}\widetilde{H}_{k+1}^{-1/2}]_{\to k} = [H_{k+1}^{-1/2}]_{\to k}$$

- Precondition H_k to obtain $\widetilde{H}_k$:

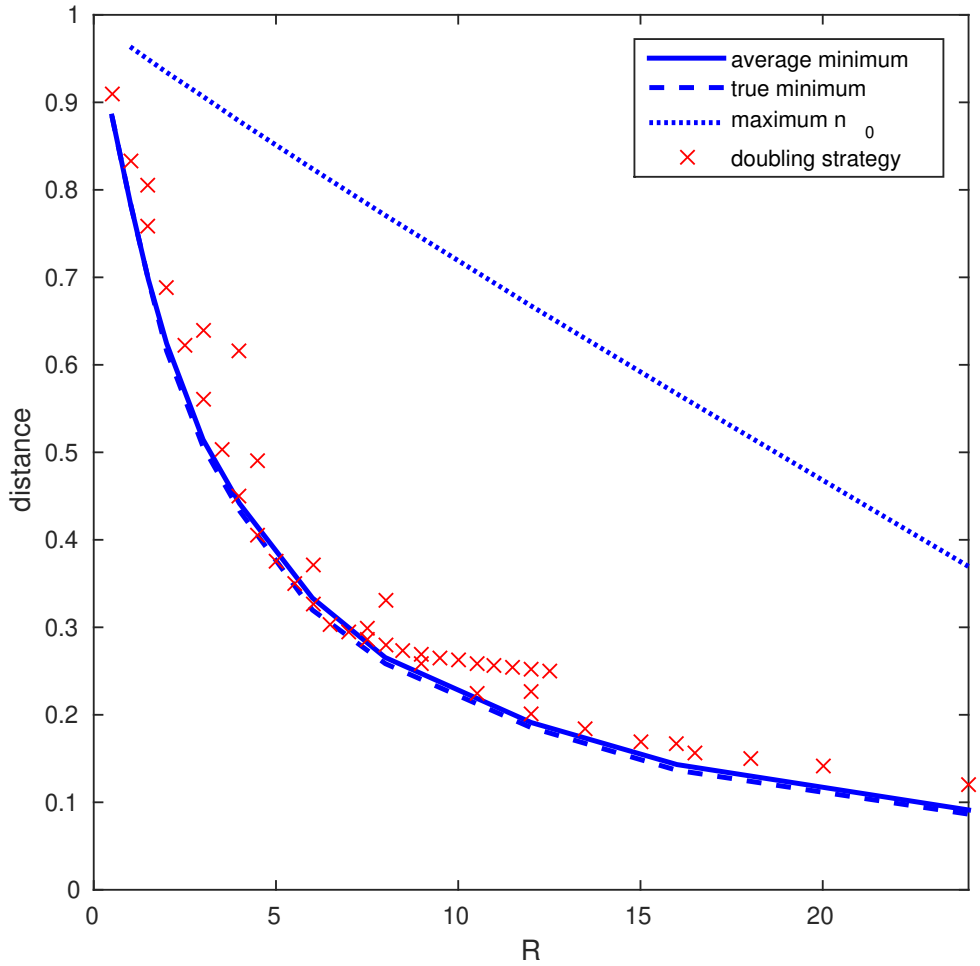
$$\widetilde{H}_k = [H_{k+1}^{-1/2}]_{\to k} H_k [H_{k+1}^{-1/2}]_{\to k} \overset{\text{sim}}{=} [H_{k+1}^{-1}]_{\to k} H_k$$

- Build $P_k \widetilde{H}_k^{-1/2}$ to precondition at next level:

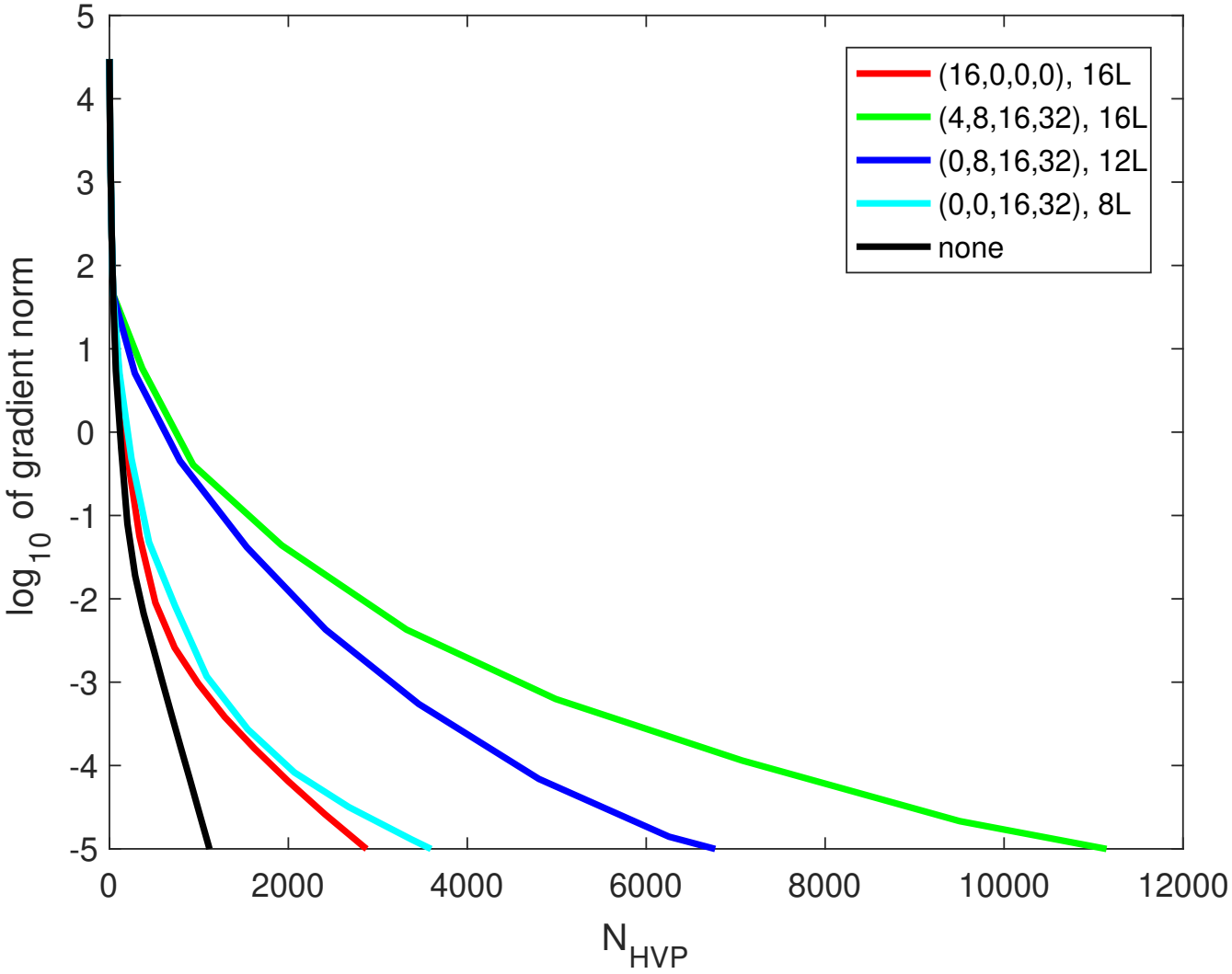
$$P_k \widetilde{H}_k^{-1/2} = [H_{k+1}^{-1/2}]_{\to k} H_k^{-1/2} [H_{k+1}^{1/2}]_{\to k} \overset{\text{sim}}{=} H_k^{-1/2}$$

Fixed memory ratio

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



Cost including building preconditioner



Hessian decomposition

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

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

- computational advantages of local Hessians:
  - fewer eigenvalues required for limited-memory approximation;
  - could be computed in parallel;
  - could use local rather than global models;
  - could be calculated at a coarser grid level.

Practical approach

1. Compute limited-memory approximations to local sensor-based Hessians on level k using n_k eigenpairs:

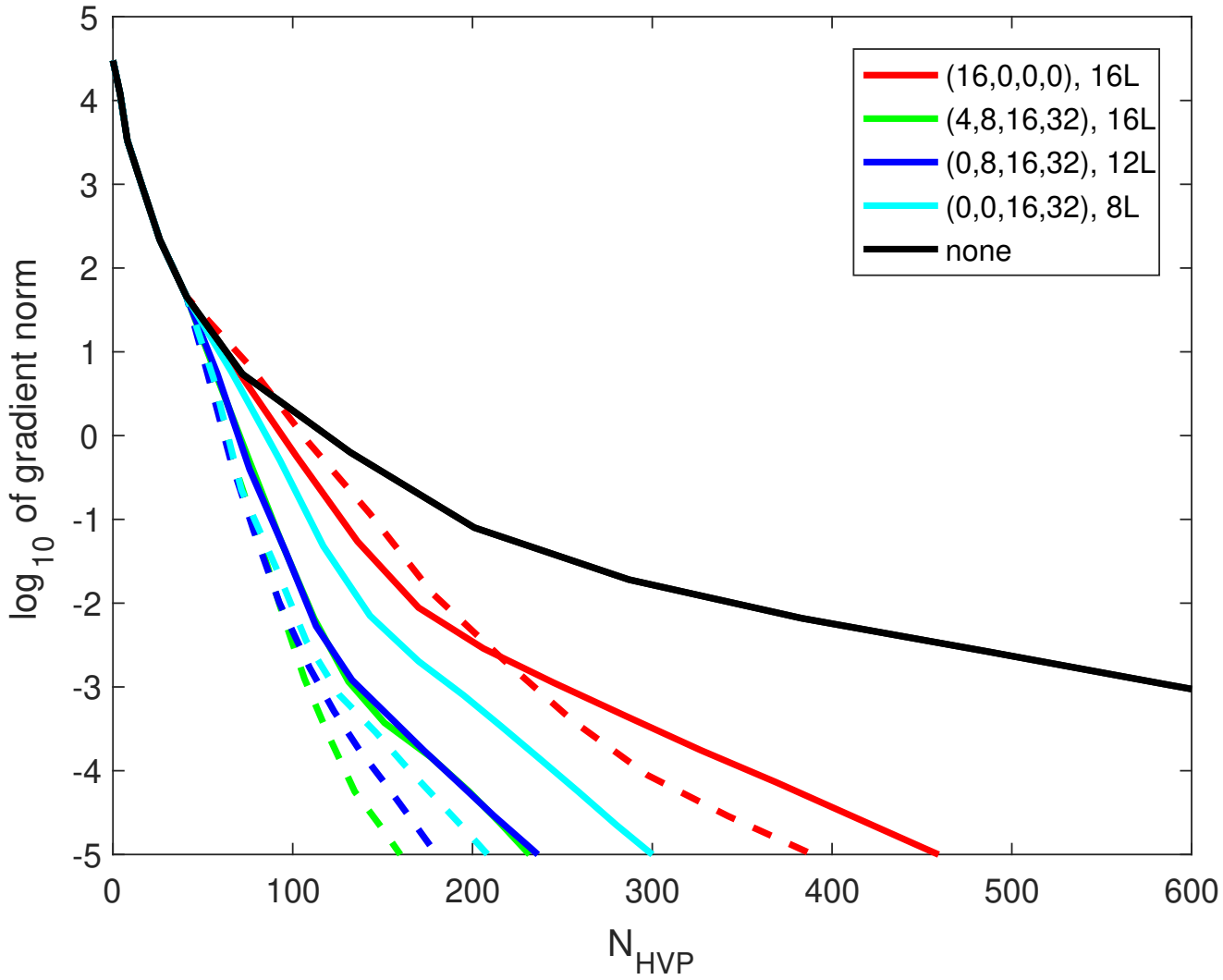
$$H_k^s \approx I + \sum_{i=1}^{n_k} (\lambda_i - 1)\mathbf{u}_i \mathbf{u}_i^T$$

2. Assemble these to form H_a.
3. Apply mlevd to H_a based on a fixed N_e.

- Advantage:
  - Local Hessians cheaper to compute.
- Disadvantages:
  - Additional user-specified parameter(s) k, n_k needed.
  - More memory required as local Hessians must also be stored.
- Can use multilevel approximation of local Hessians to reduce memory costs.

Cost including building preconditioner

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



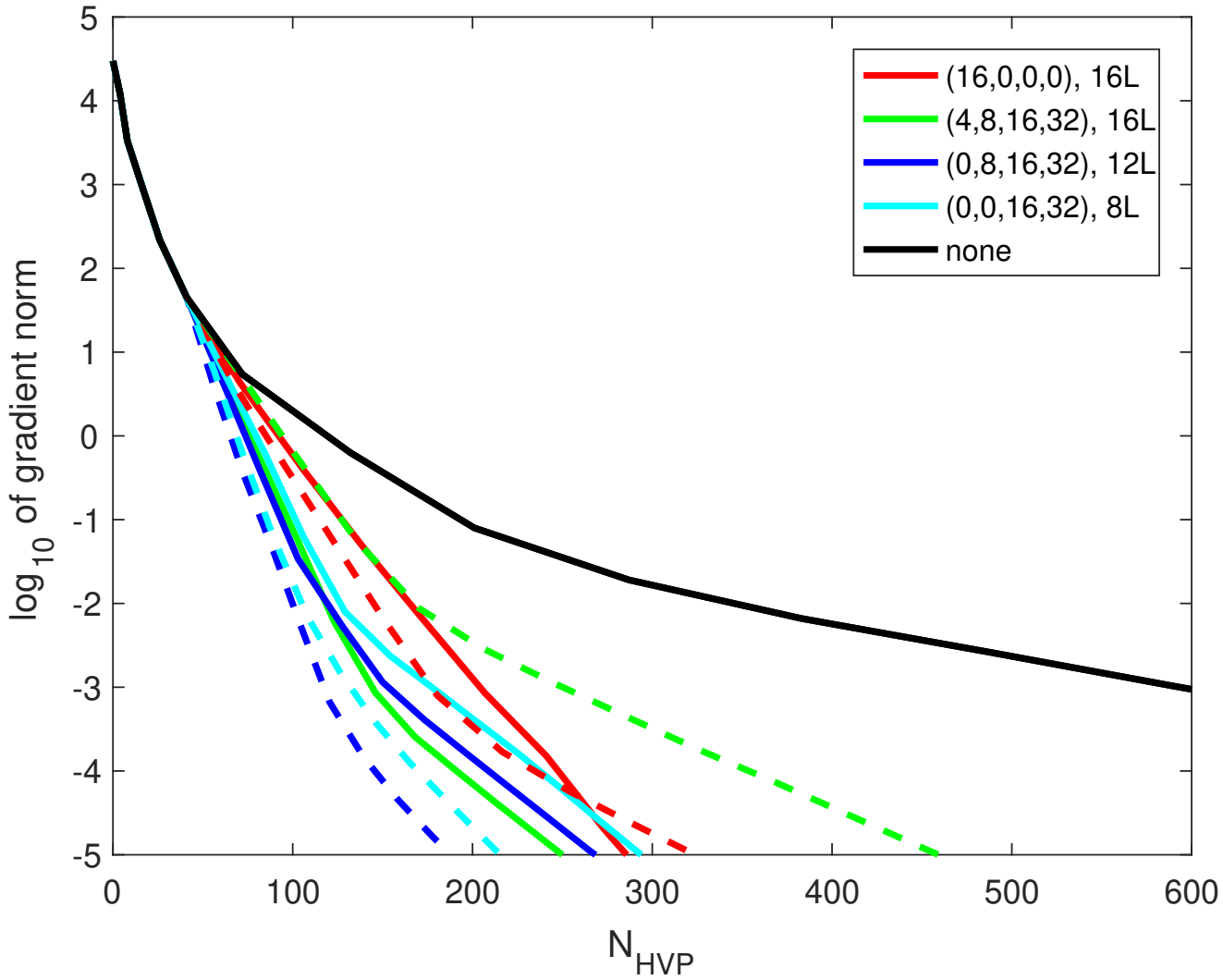
Practical approach: Version 2

1. Approximate each local Hessian H_k^s by applying mlevd to local inverse Hessians based on $N_{e,k}$.
2. Assemble these to form reduced-memory Hessian H_a^{rm}.
3. Use mlevd again on H_a^{rm} based on N_e.

- Advantage:
  - Requires less memory than Version 1.
- Disadvantage:
  - Additional user-specified parameter(s) $N_{e,k}$ needed.

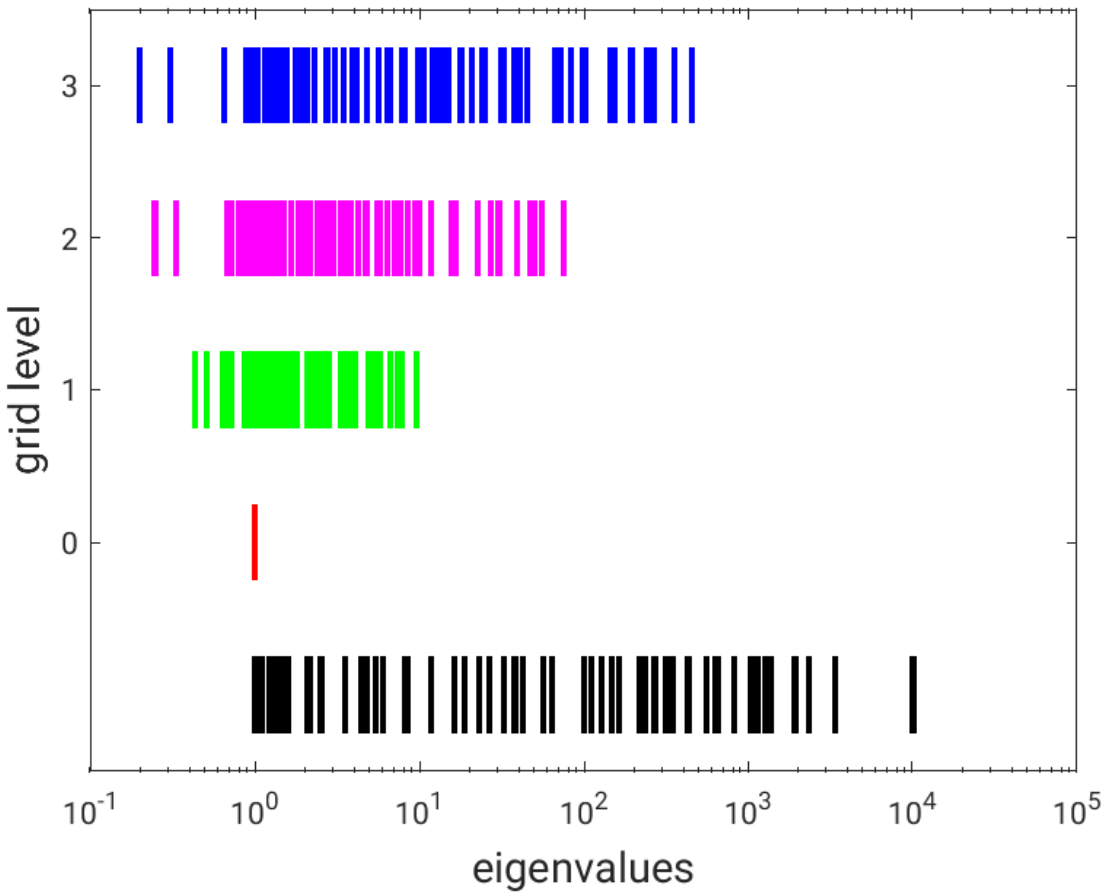
Version 2: cost including building preconditioner

- Local Hessians with 8 eigenvalues at level 0 (solid lines) or level 1 (dashed lines) with $N_{e,k} = (8, 4, 0, 0)$ MG approx.



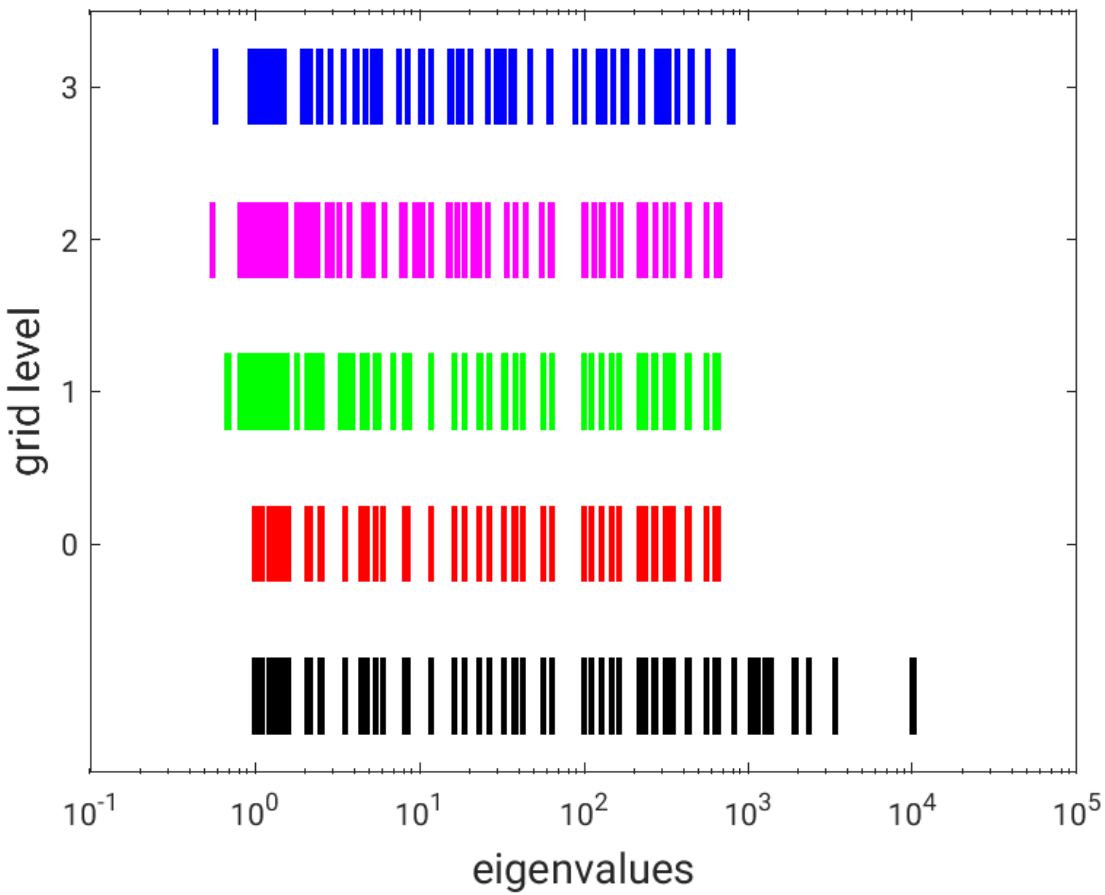
Idea behind preconditioning

- 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.