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:

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

- Hessian of the cost function is

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

where

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

Motivation

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

- $\mathbb{H}^{-1}$ 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.
- $\mathbb{H}^{-1/2}$ can be used in ensemble forecasting.
- $\mathbb{H}^{-1}$, $\mathbb{H}^{-1/2}$ can be used for preconditioning in a Gauss-Newton method.

Approximating the inverse Hessian

- State and observation vectors used in realistic applications can be of length $10^9 - 10^{12}$ and $10^6 - 10^9$, respectively.
- Cannot store $\mathbb{H}$ as a matrix: only action of applying $\mathbb{H}$ to a vector is available.
- Evaluating $\mathbb{H}\mathbf{v}$ is expensive in terms of computing time and memory (involves both forward and backward model solves with a sequence of tangent linear and adjoint problems).
- No such option exists for evaluating $\mathbb{H}^{-1}\mathbf{v}$.
- Aim here is to construct a limited-memory approximation to $\mathbb{H}^{-1}$ using only matrix-vector multiplication.

First level preconditioning

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

- 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).

- For the rest of the talk, we focus on approximating H^{-1}.

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

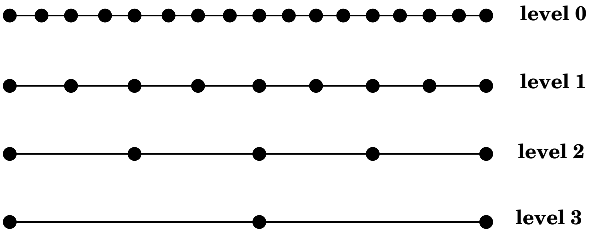
$$H \approx I + \sum_{i=1}^{n_e} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

- IDEA: Build a limited-memory approximation to H^{-1} (or $H^{-1/2}$).
- Easy to evaluate matrix powers in this form:

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

Multilevel approximation

- 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 full Hessian H_0.
- Grid level k contains $m_k = m/2^k + 1$ nodes.



- Identity matrix I_k on grid level k.

Test problem 1

- Model is 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 approximation to H^{-1} with four grid levels:

k	0	1	2	3
grid points	401	201	101	51

- Action of Hessian matrix H_0 available on level 0 (finest grid).
- Need grid transfer operators.
- $[M]_{\to k}$ means "matrix M transferred to 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:

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

  - From fine to coarse:

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

Outline of multilevel concept

Step 1. Start on coarsest grid level.

Step 2. Represent H_0 on grid level k as $H_k = [H_0]_{\to k}$.

Step 3. Precondition this to obtain $\tilde{H}_k = P_k^T H_k P_k$, noting that

$$H_k{}^{-1} = (P_k \tilde{H}_k^{-1/2})(\tilde{H}_k^{-1/2} P_k^T) \quad \equiv \quad \hat{P}_k \hat{P}_k{}^T.$$

Step 4. Build a limited memory approximation to $\tilde{H}_k^{-1/2}$ from n_k eigenvalues using the Lanczos method.

Step 5. Project $\hat{P}_k$ to the level above to be used as preconditioner at the next coarsest level.

Step 6. Move up one grid level and repeat from step 2.

Preconditioners

- On coarsest grid, level $k+1$ does not exist so set $P_k = I_k$.

- For other levels, P_k is constructed on level $k+1$ and applied on level k.

- Preconditioners are constructed recursively:

$$P_k = [\hat{P}_{k+1}]_{\to k} = \left[P_{k+1} \tilde{H}_{k+1}^{-1/2} \right]_{\to k} .$$

- At level 0, inverse Hessian approximation will contain eigenvalue information from all levels.

Algorithm in practice

- use $N_e = (n_0, n_1, \ldots, n_{k_c})$ eigenvalues at each level

```
[Λ,𝒰]=multilevel(H_0,N_e)
for   k = k_c, k_c − 1, ..., 0
   compute by the Lanczos method
       {λ_k^i, U_k^i}, i = 1, ..., n_k of H̃_{0→k}
   using preconditioner P_k
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

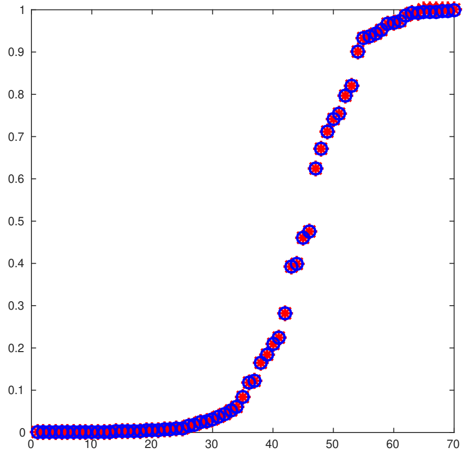
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

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

Eigenvalues of the inverse Hessian

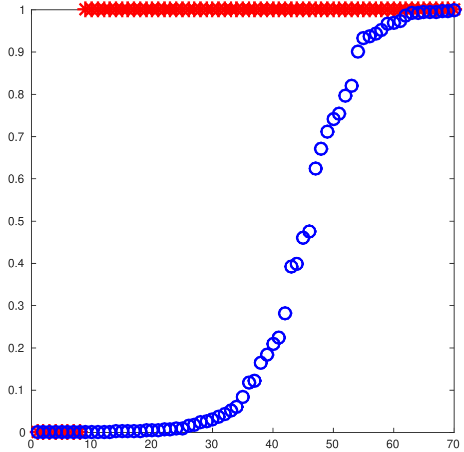
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

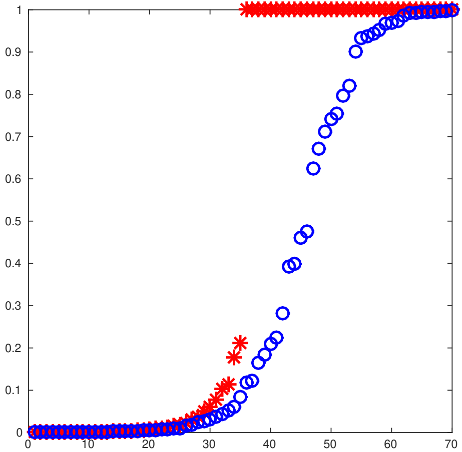
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

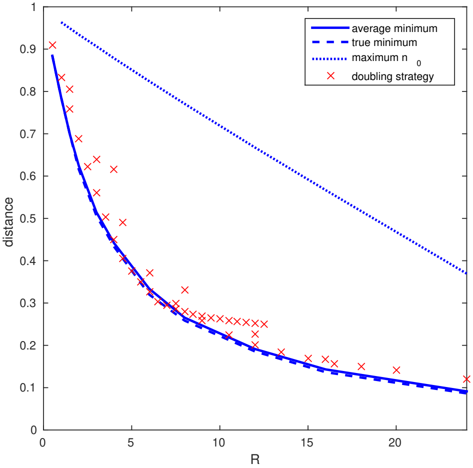
- Exact (blue circles), approximated (red stars)



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

Fixed memory ratio

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



Example: PCG iteration for one Newton step

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

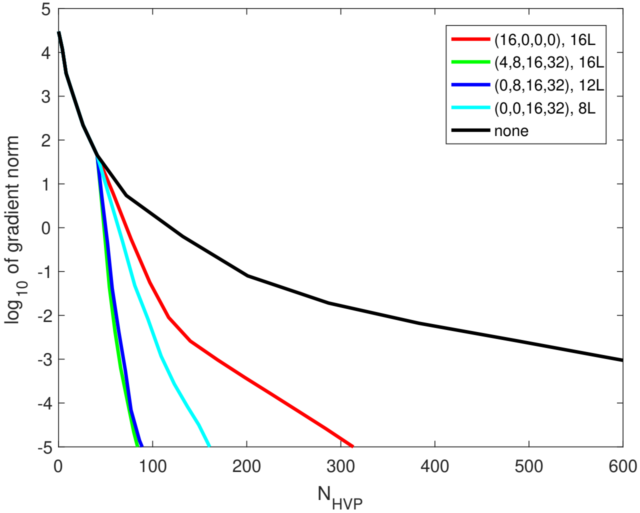
$$\mathbb{H}(\mathbf{u}_k)\delta\mathbf{u}_k = G(\mathbf{u}_k)$$

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

- 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



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:

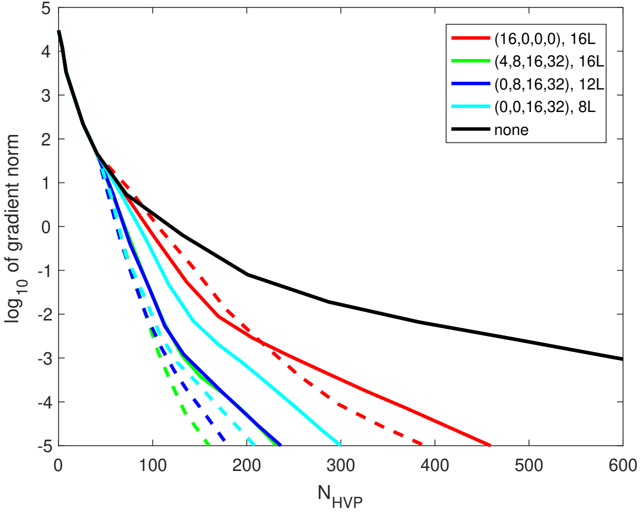
$$H_k^s \approx I + \sum_{i=1}^{n_k} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

2. Assemble these to form H_a.
3. Apply multilevel 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).



Concluding remarks

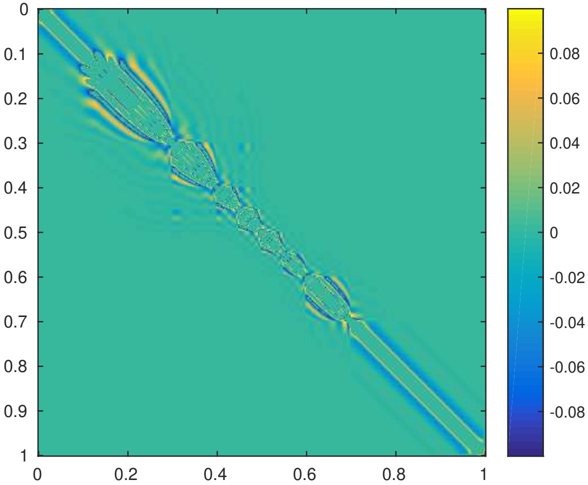
- Algorithm based solely on repeated use of Lanczos at each level (for limited-memory approximations).
- Difficult to identify the correct number of eigenvalues to use at each level: analysis required.
- Full algorithm may not be not practical, but we have developed practical implementations based on Hessian decompositions.
- Also works well for other configurations (e.g. moving sensors, different initial conditions), other equations (shallow water equations).
- Potential for extension to higher dimensions and other applications.

Brown, Gejadze & Ramage,
A Multilevel Approach for Computing the Limited-Memory Hessian and its Inverse in Variational Data Assimilation,
SIAM Journal on Scientific Computing 38(5), 2016.

Additional slides

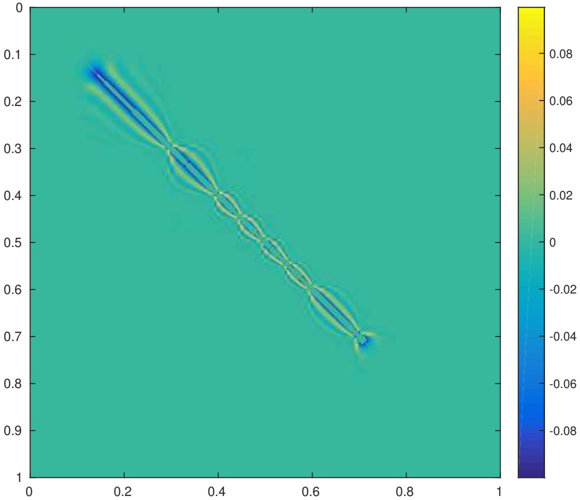
Correlation matrix

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



Preconditioned correlation matrix

- H^{-1} (scaled to have unit diagonal)



Eigenvalues of preconditioned Hessian

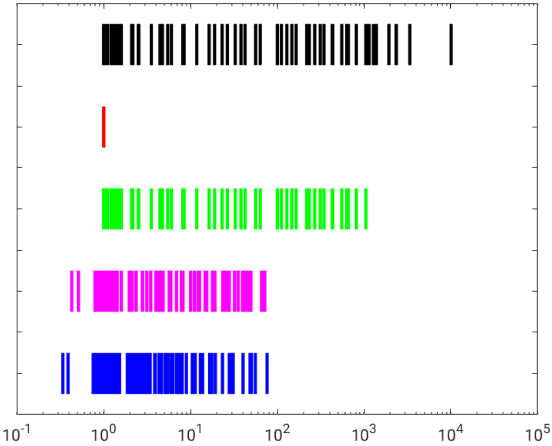
H_0

$(64, 0, 0, 0)$

$(8, 0, 0, 0)$

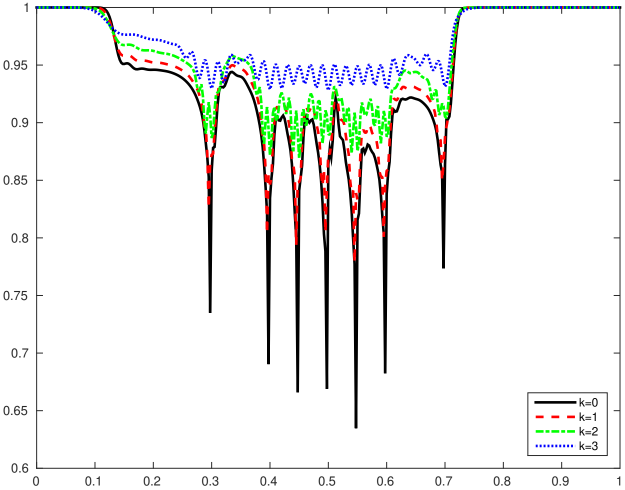
$(2, 4, 8, 16)$

$(0, 0, 29, 6)$



Motivation for multilevel framework

- Diagonal of H^{-1}:



Eigenvalues of Hessian at each level

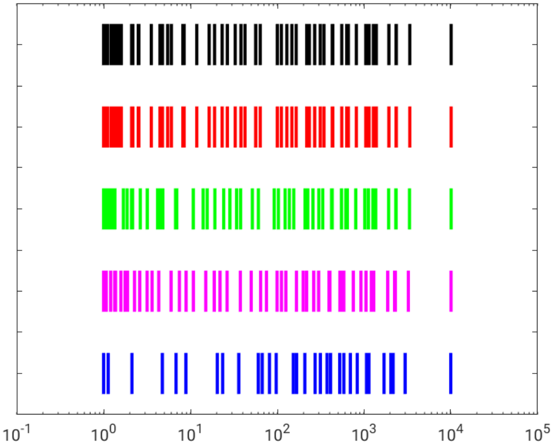
H_0

$H_0 = [H_0]_{\to 0}$

$H_1 = [H_0]_{\to 1}$

$H_2 = [H_0]_{\to 2}$

$H_3 = [H_0]_{\to 3}$



Test problem 2

- Model is 1D shallow water equations.

PCG iteration for one Newton step

- Background covariance matrix B constructed using a Laplacian correlation function.

	# PCG iterations			
Preconditioner	$n = 400$	$n = 800$	$n = 1600$	$n = 3200$
none	308	1302	5,879	25,085
MG(4,0,0,0)	38	34	34	47
MG(1,2,4,8)	31	29	28	37
MG(0,2,4,16)	27	26	24	32
MG(0,0,8,16)	26	25	24	30
MG(0,0,0,32)	23	19	19	24

PCG iteration for one Newton step

- Background covariance matrix B constructed using a Second-Order Auto-Regressive (SOAR) correlation function.

	# PCG iterations			
Preconditioner	$n = 400$	$n = 800$	$n = 1600$	$n = 3200$
none	509	2,277	10,453	43,915
MG(4,0,0,0)	39	35	35	44
MG(1,2,4,8)	28	26	26	34
MG(0,2,4,16)	23	22	21	27
MG(0,0,8,16)	22	21	20	26
MG(0,0,0,32)	19	16	15	20

Practical approach: Version 1

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

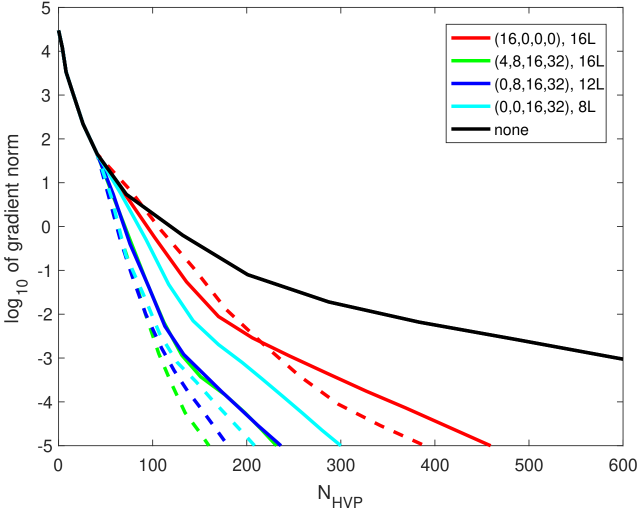
$$H_k^s \approx I + \sum_{i=1}^{n_k} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

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.

Sample costs including building preconditioner

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



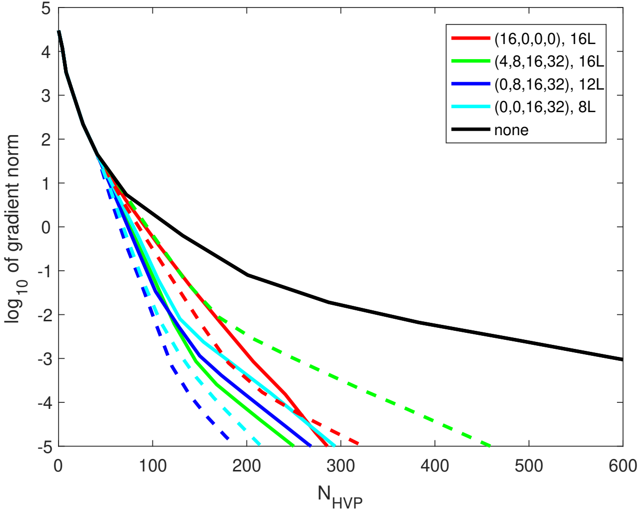
Practical approach: Version 2

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.

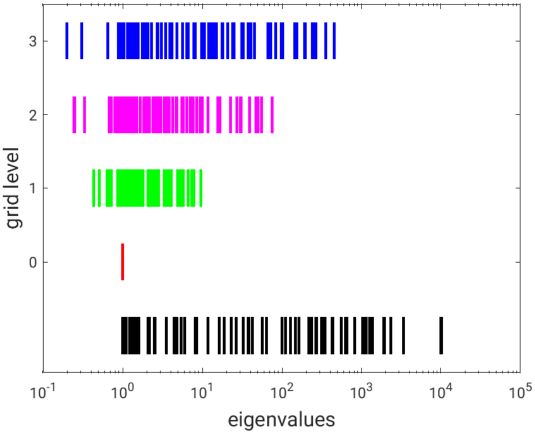
Version 2: cost including building preconditioner

- 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.



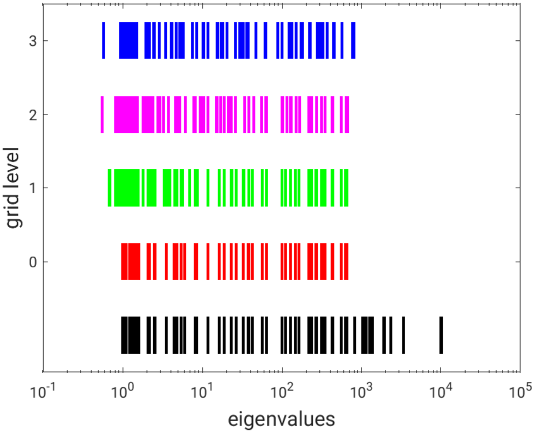
Eigenvalue plots

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



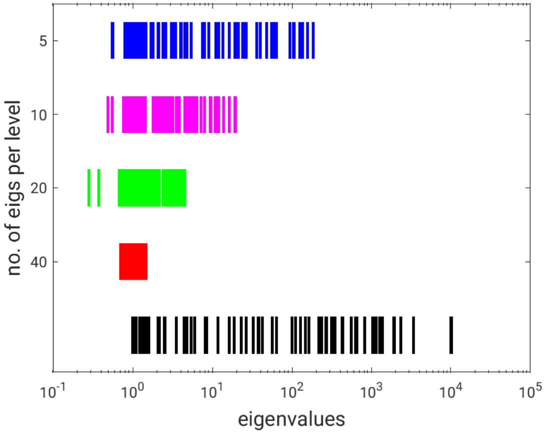
Replace with limited-memory approximations

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



Idea: use all levels

- Build recursive preconditioner using information from all levels.



Modified grid transfer operators: two grid level

Hessian	$H_0 = I_0 + M_0$
prolongation P	restriction $R = P^T$

- Assume identity part is transferred exactly.
- Transfer to coarse grid:

$$\tilde{H}_1 = I_1 + R(H_0 - I_0)P = RH_0P + I_1 - RP$$

- Invert and return to fine grid:

$$\tilde{H}_0^{-1/2} = I_0 + P(\tilde{H}_1^{-1/2} - I_1)R = P\tilde{H}_1^{-1/2}R + I_0 - PR$$

Experiment: transfer of eigenvectors

- Experiment to evaluate the error in grid transfers with P, R corresponding to piecewise cubic spline interpolation.

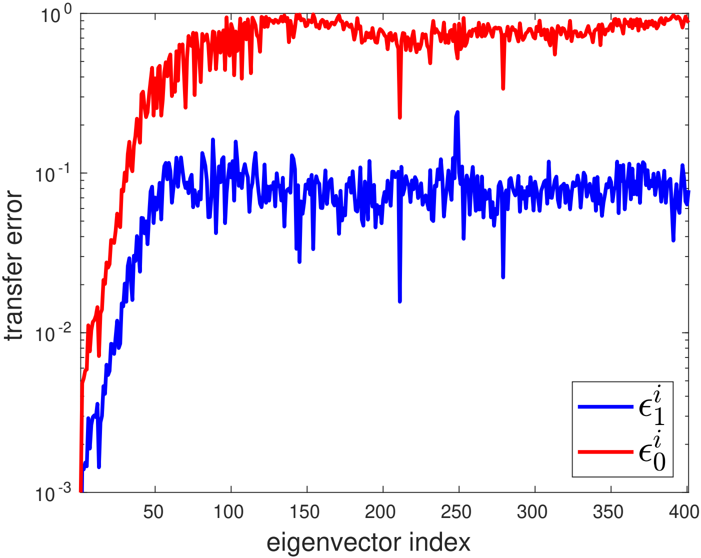
- Order eigenvalues/vectors of H_0 from largest to smallest:

$$\lambda_0^i,\ \mathbf{v}_0^i,\ i = 1, \ldots, n_0$$

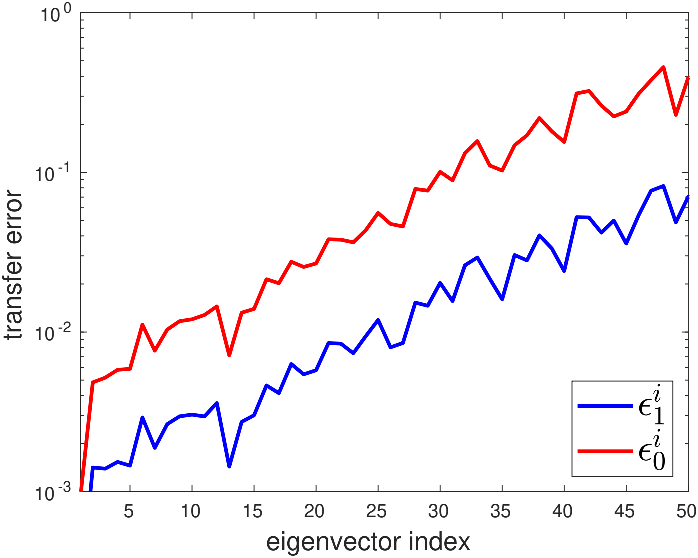
- Calculate two measures of grid transfer error:

$$\epsilon_0^i = \frac{\|(I_0 - PR)\mathbf{v}_0^i\|_2}{\|\mathbf{v}_0^i\|_2}, \qquad \epsilon_1^i = \frac{\|(I_1 - RP)R\mathbf{v}_0^i\|_2}{\|\mathbf{v}_0^i\|_2}$$

Eigenvector transfer



Eigenvector transfer



Motivation

It is sometimes nice in Scotland. . .