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

Data assimilation

- 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).
- Numerical weather prediction is essentially an IVP: given initial conditions, forecast atmospheric evolution.
- 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

Motivation

Four-dimensional Variational Assimilation (4D-Var)

Minimise cost function

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

with constraint $\mathbf{x}_i = \mathcal{M}_{0 \to i} \mathbf{x}_0$.

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

Incremental 4D-Var

- Linearise $\mathcal{H}$, $\mathcal{M}$ using the tangent linear hypothesis and solve resulting unconstrained optimisation problem iteratively.

- Hessian of the cost function

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

  incorporates the tangent linear operator M and its adjoint.

- Action of applying $\mathbb{H}$ to a vector is available, but expensive (involves both forward and backward model solves).

- AIM: construct a limited-memory approximation to $\mathbb{H}^{-1}$ using only matrix-vector multiplication for use as preconditioning in a Gauss-Newton method.

Hessian system

- 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 based on the background covariance matrix:

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

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

  [Haben et al., Computers & Fluids 46 (2011)]

- H amenable to limited-memory approximation.

Limited-memory approximation

- Find n_e leading eigenvalues and orthonormal eigenvectors using the Lanczos method (needs only $\mathbb{H}\mathbf{v}$).

- Construct approximation

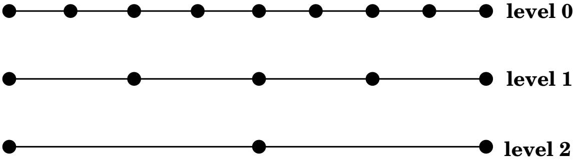
$$H \approx I + \sum_{i=1}^{n_e} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

- Easy to evaluate matrix powers:

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

Second level preconditioning

- 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:

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

Outline of multilevel concept

- Given a symmetric positive definite operator A_0 available on the finest grid level in matrix-vector product form:

1. represent A_0 on the coarsest grid level;
2. use a local preconditioner to improve the eigenvalue distribution;
3. build a limited memory approximation to its inverse using the Lanczos method (which forms the basis of the local preconditioner at the next coarsest level);
4. move up one grid level and repeat.

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.

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

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
```

$$[\Lambda, \mathcal{U}] = mlevd(H_0, N_e)$$

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 preconditioner B_k^{k+1}
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

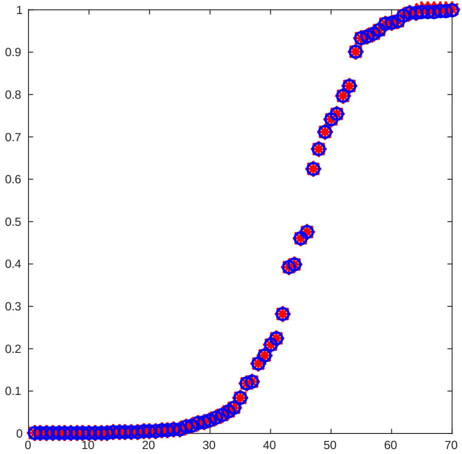
$$D = \frac{\delta(H^{-1}, \tilde{H}^{-1})}{\delta(H^{-1}, I)}$$

- Vary number of eigenvalues chosen on each grid level

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

Eigenvalues of the inverse Hessian

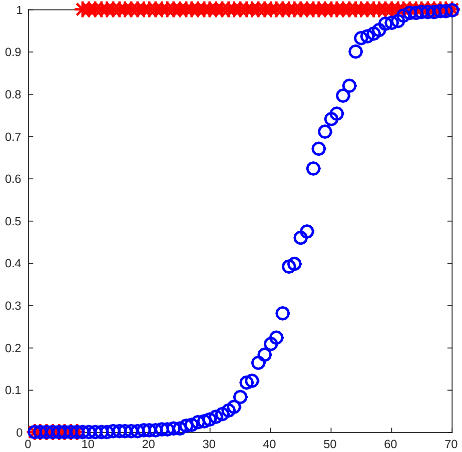
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

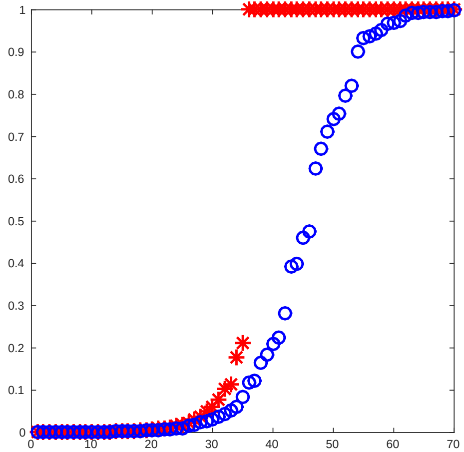
- Exact (blue circles), approximated (red stars)



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

Eigenvalues of the inverse Hessian

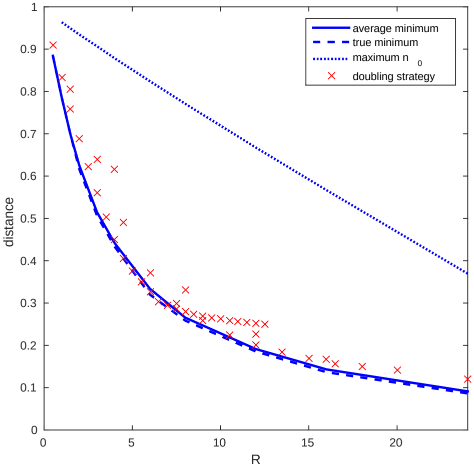
- Exact (blue circles), approximated (red stars)



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

Fixed memory ratio

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

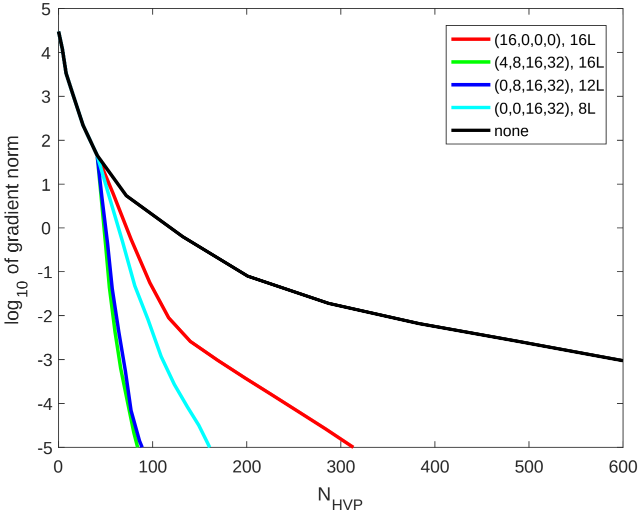


PCG iteration for one Newton step

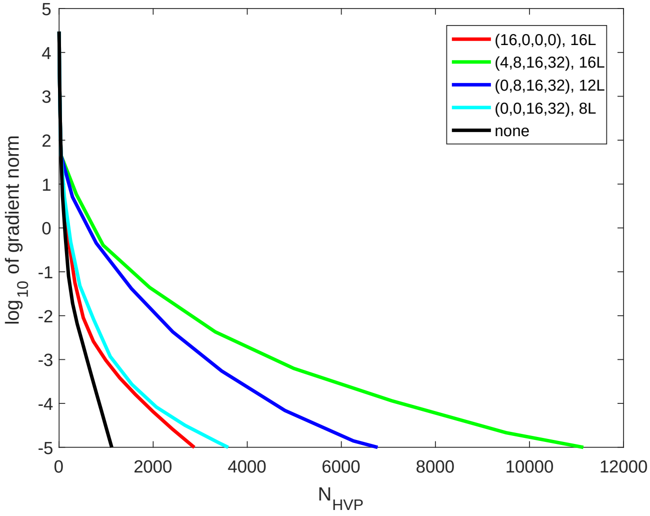
- measurement units
  - memory: length of vector on finest grid L
  - cost: cost of HVP on finest grid HVP

Preconditioner	# CG iterations	storage	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 subregions and compute local Hessians H^l such that

$$H(u) = I + \sum_{l=1}^{L} (H^l(u) - I)$$

- fewer eigenvalues required for limited-memory representation of each H^l
- local Hessians can be computed in parallel
- H^l need not be computed at finest grid level:

$$H_k(u_k) = I_k + \sum_{l=1}^{L} (H_k^l(u_k) - I_k)$$

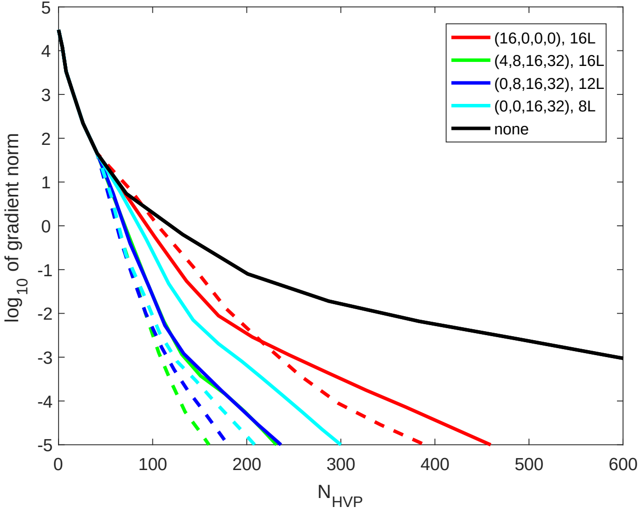
- could run local rather than global model

Practical approach: Version 1

- Compute limited-memory approximations to local sensor-based Hessians on level l using n_l 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) l, n_l 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).

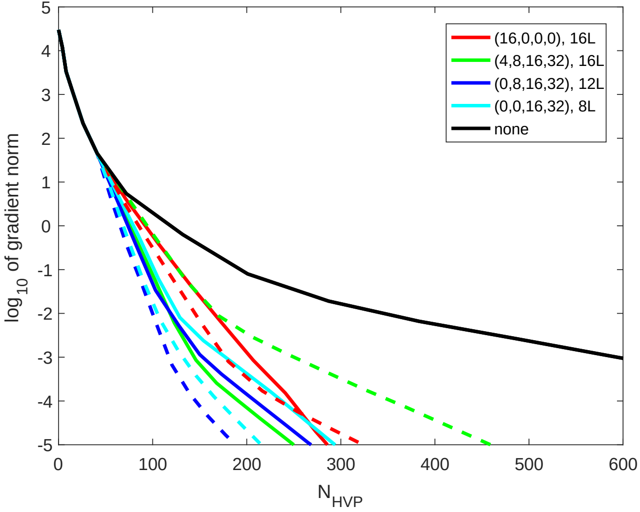


Practical approach: version 2

- Can reduce memory requirements further by using a multilevel approximation of each limited-memory local Hessian on level l using n_l eigenpairs.
- Approximate local Hessians by applying mlevd to local inverse Hessians based on N_e^l.
- 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.

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

It is sometimes nice in Scotland. . .