Multilevel Preconditioning for Data Assimilation with 4D-Var

Alison Ramage*, Kirsty Brown*, Igor Gejadze†

∗ Department of Mathematics and Statistics, University of Strathclyde

† IRSTEA, Montpelier, France

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).

Hessian system

- 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 $\mathbb{H}\mathbf{v}$).

- Precondition $\mathbb{H}$ based on the background covariance matrix:

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

- For detailed information on the eigenspectrum of H, see e.g. HABEN ET AL. (2011), TABEART ET AL. (2018).

- Focus here on solving systems of the form

$$H\mathbf{u} = \mathbf{g}$$

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

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

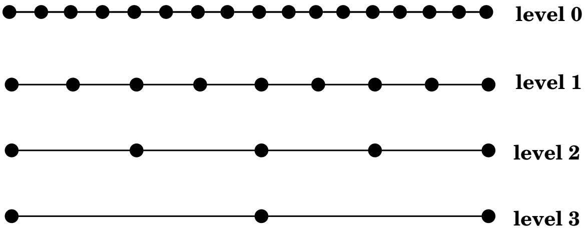
- Easy to evaluate matrix powers:

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

- IDEA: Build a limited-memory approximation to H^{-1} for use as a preconditioner in PCG (and elsewhere).

Multilevel preconditioning

- IDEA: Build 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 top-level Hessian H_0.
- Grid level k contains $m_k = m/2^k + 1$ nodes.



Inter-grid transfers

- Write I_k for identity matrix on grid level k.
- 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:

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

- Write H_k for H_0 restricted to grid level k.

Key ideas

- Build a limited-memory approximation to H^{-1} in multilevel form based on a sequence of nested grids.

- Build upwards from the coarsest level.

If H is preconditioned as $\tilde{H} = P^T HP$, 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.

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

- Action of Hessian matrix H_0 available on level 0 (finest grid).
- For convenience in this talk, we will
  - assume matrices have been transferred to be the right size for multiplying together;
  - use $\stackrel{\text{sim}}{=}$ to denote a similarity transformation between matrices (which have the same eigenvalues).

Level 3 (coarsest level)

- Restrict H_0 to level 3 to obtain H_3.
- Use preconditioner from previous level:

$$P_3 = I_3$$

- Precondition H_3 to obtain $\widetilde{H}_3$:

$$\widetilde{H}_3 = P_3^T H_3 P_3 = H_3$$

- Build $P_3 \widetilde{H}_3^{-1/2}$ to precondition at next level:

$$P_3 \widetilde{H}_3^{-1/2} = H_3^{-1/2}$$

Level 2

- Restrict H_0 to level 2 to obtain H_2.
- Use preconditioner from previous level:

$$P_2 = P_3 \widetilde{H}_3^{-1/2} = H_3^{-1/2}$$

- Precondition H_2 to obtain $\widetilde{H}_2$:

$$\widetilde{H}_2 = P_2^T H_2 P_2 = H_3^{-1/2} H_2 H_3^{-1/2} \stackrel{\text{sim}}{=} H_3^{-1} H_2$$

- Build $P_2 \widetilde{H}_2^{-1/2}$ to precondition at next level:

$$P_2 \widetilde{H}_2^{-1/2} = H_3^{-1/2} H_2^{-1/2} H_3^{1/2} \stackrel{\text{sim}}{=} H_2^{-1/2}$$

Level 1

- Restrict H_0 to level 1 to obtain H_1.
- Use preconditioner from previous level:

$$P_1 = P_2 \widetilde{H}_2^{-1/2} = H_2^{-1/2}$$

- Precondition H_1 to obtain $\widetilde{H}_1$:

$$\widetilde{H}_1 = P_1^T H_1 P_1 = H_2^{-1/2} H_1 H_2^{-1/2} \stackrel{\text{sim}}{=} H_2^{-1} H_1$$

- Use $P_1 \widetilde{H}_1^{-1/2}$ to precondition at next level:

$$P_1 \widetilde{H}_1^{-1/2} = H_2^{-1/2} H_1^{-1/2} H_2^{1/2} \stackrel{\text{sim}}{=} H_1^{-1/2}$$

Level 0 (top level)

- Full top level matrix H_0 already available.
- Use preconditioner from previous level:

$$P_0 = P_1 \widetilde{H}_1^{-1/2} = H_1^{-1/2}$$

- Precondition H_0 to obtain $\widetilde{H}_0$:

$$\widetilde{H}_0 = P_0^T H_0 P_0 = H_1^{-1/2} H_0 H_1^{-1/2} \overset{\text{sim}}{=} H_1^{-1} H_0$$

- Recover final approximation H_0^{-1}:

$$H_0^{-1} = P_0 \widetilde{H}_0^{-1} P_0^T = H_1^{-1/2} H_0^{-1} H_1^{1/2} \quad \left(\overset{\text{sim}}{=} H_0^{-1} \right)$$

Important points

- In practice, algorithm is based on limited-memory approximation of matrices so 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*.

Algorithm

- algorithm:

```
[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_k
   using preconditioner P_k from level k+1
end
```

- storage:

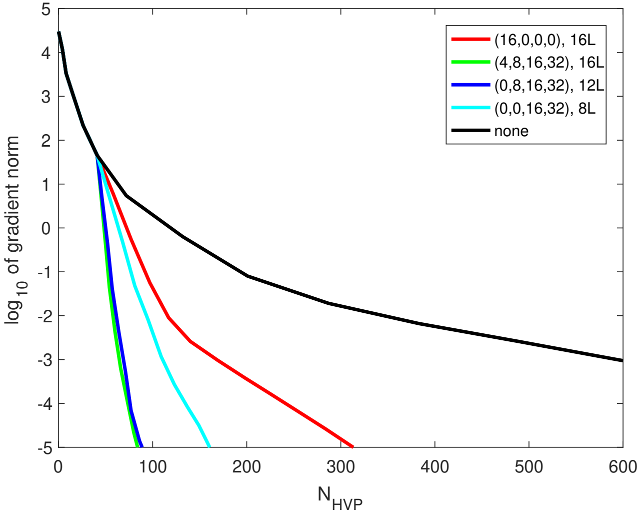
$$\begin{aligned}\Lambda &= \left[\lambda^1_{k_c}, \ldots, \lambda^{n_{k_c}}_{k_c}, \lambda^1_{k_c-1}, \ldots, \lambda^{n_{k_c-1}}_{k_c-1}, \ldots, \lambda^1_0, \ldots, \lambda^{n_0}_0\right], \\ \mathcal{U} &= \left[U^1_{k_c}, \ldots, U^{n_{k_c}}_{k_c}, U^1_{k_c-1}, \ldots, U^{n_{k_c-1}}_{k_c-1}, \ldots, U^1_0, \ldots, U^{n_0}_0\right].\end{aligned}$$

PCG iteration for one Newton step

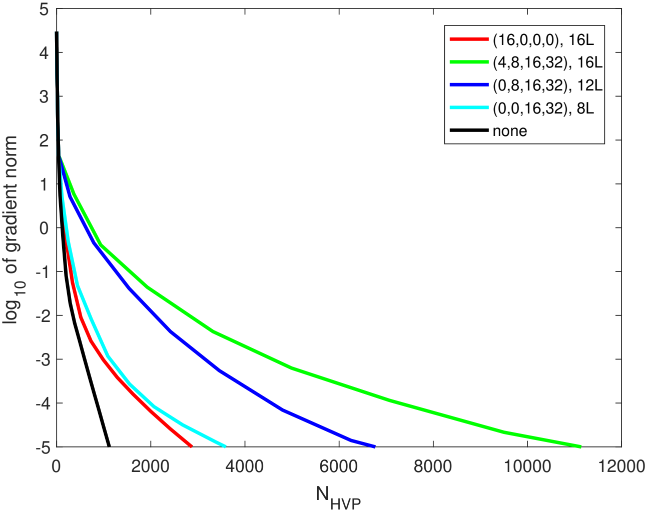
- measurement units
  - memory: length of vector on finest grid L
  - cost: cost of HVP on finest grid HVP

Preconditioner	# CG iterations	storage	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



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:

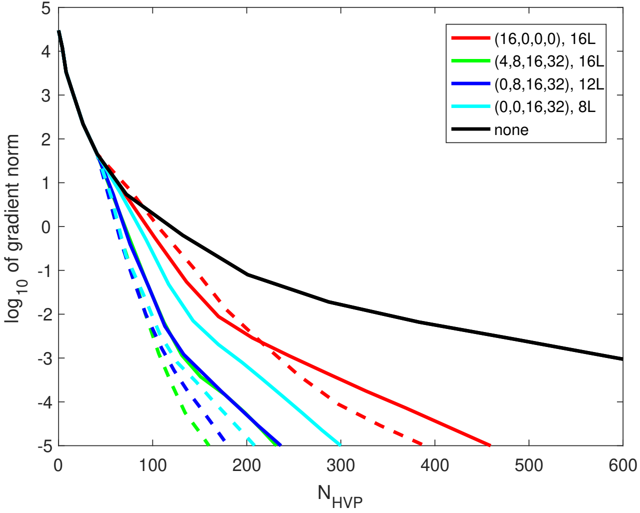
$$H_k^s \approx I + \sum_{i=1}^{n_k} (\lambda_i - 1)\mathbf{u}_i\mathbf{u}_i^T$$

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).



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.
- 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).
- 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.