Current work

The Problem we are trying to solve

Solve for $\tilde{x}$:

$$A(x)\tilde{x} = b(x) \tag{1}$$

with $A(x) = \left(B^{-1} + (HM)^T R^{-1}(HM)\right)$

- Learn a low-rank approximation using DNN (to use as preconditioner):
  - $x \mapsto (U(x), S(x)) \in \mathbb{R}^{n\times r} \times \mathbb{R}^r$ where $A(x) \approx U(x)\mathrm{diag}(S(x))U(x)^T$
  - Approximate the norm using random vectors
  - Look for an approximation in the dual space instead ?
- Better architecture for neural network (CNN, UNet, attention layers?), instead of MLP
- (Technical stuff): DVC and MLflow for reproductibility, data versioning and workflow management

Variational DA and ML

Using ML-based preconditioners in VarDA problems

Victor Trappler

April 3, 2023

Table of contents

Introduction

Notation and setting

- $x \in \mathbb{R}^n$ state vector: Variables that describe a physical system ($n = \mathcal{O}(10^{6-9})$)

Different sources of information on a state vector x:

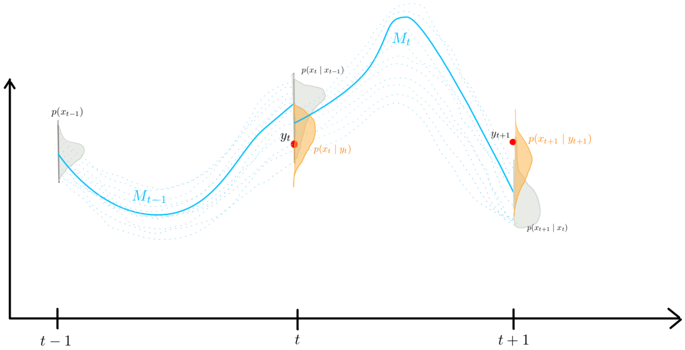
- A priori information $p(x)$
    - Historical data
    - Balance equations
- Observations y, obtained more or less indirectly
- Numerical model which maps the state to the observations $\mathcal{G} = \mathcal{H} \circ \mathcal{M}$

How to combine them ? Bayes theorem

Data Assimilation and Bayesian Inference

Bayes' theorem : update information on x using y

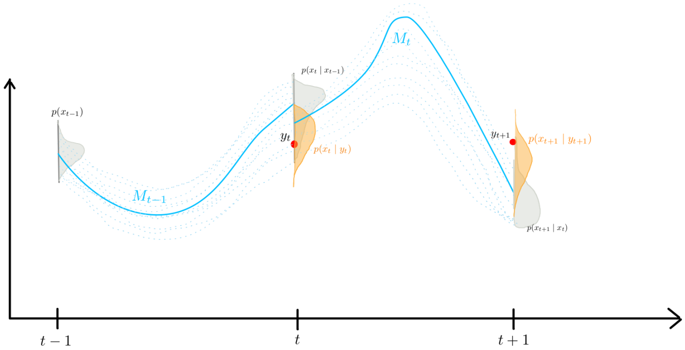
$$p(x \mid y) = p(x)\frac{\overbrace{p(y \mid x)}^{\text{likelihood}}}{\underbrace{p(y)}_{\text{evidence}}} \tag{2}$$



Data Assimilation and Bayesian Inference

Bayes' theorem sequentially: update information on x_t using y_t

$$p(x_t \mid y_t) = \underbrace{p(x_t \mid x_{t-1})}_{\textit{prior/forecast}} \frac{\overbrace{p(y_t \mid x_t)}^{\text{likelihood}}}{\underbrace{p(y_t)}_{\text{evidence}}} \tag{2}$$



Variational Data Assimilation

Point estimates

We are interested in a point estimate of the posterior distribution: Maximum A Posteriori

$$\min_x\{-\log p(x \mid y)\} = \min_x\{-\underbrace{\log p(y \mid x)}_{\text{log-lik=misfit}} - \underbrace{\log p(x)}_{\text{regularization}}\} \tag{3}$$

- $Y \mid x$ encodes the relation between the forward model and the observations
- X encodes the knowledge we gathered so far on x

Gaussian Assumptions

- $y = \mathcal{G}(x) + \varepsilon$, then $Y \mid x \sim \mathcal{N}(\mathcal{G}(x), R)$
- $x \sim \mathcal{N}(x^b, B)$

Standard formulation of the objective function

Using the Gaussian assumptions, we have $p(x \mid y) \propto e^{-J(x)}$ with

$$J(x) = \frac{1}{2}\|\mathcal{G}(x) - y\|^2_{R^{-1}} + \frac{1}{2}\|x - x^b\|^2_{B^{-1}} \tag{4}$$

Which can be simplified to

$$J(x) = \frac{1}{2}\|\mathcal{G}(x) - y\|^2 \tag{5}$$

Analysis

The analysis is the MAP (point estimate)

$$x^a = \min_x J(x) = \min_x \frac{1}{2}\|\mathcal{G}(x) - y\|^2 \tag{6}$$

which is a non-linear least square problem

$\rightarrow$ $\mathcal{G}$ is a numerical model, expensive to evaluate. How to minimize J ?

Incremental formulation

$$J_{\text{inc}}(\delta x; x) = \underbrace{J(x) + \nabla J^T \delta x + \frac{1}{2}(\delta x)^T H(\delta x)}_{\text{quadratic wrt } \delta x} \approx J(x + \delta x) \tag{7}$$

with

$$G = \nabla \mathcal{G} = \text{Tangent Linear of } \mathcal{G} \text{ at } x \tag{8}$$

$$\nabla J(x) = G^T(\mathcal{G}(x) - y) = G^T d \tag{9}$$

$$H(x) = G^T G + \underbrace{\cancel{Q(x)}}_{\frac{\partial^2}{\partial x^2}} \approx G^T G = H_{\text{GN}} \tag{10}$$

To minimize the quadratic approx., the increment δx is the solution to the linear system

$$H_{\text{GN}} \delta x = -\nabla J \iff (G^T G)\delta x = -G^T d \tag{11}$$

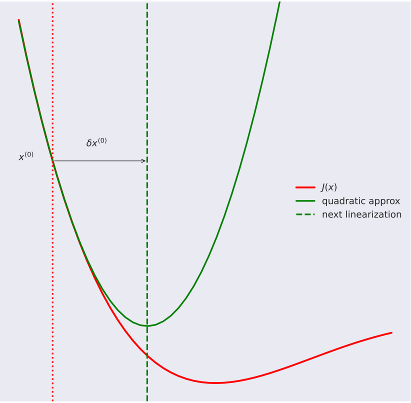
Nested Loops

Set $k = 0$

Repeat until convergence/computational budget spent

Outer Loop

- Evaluate
  - Forward $\mathcal{G}(x^{(k)})$
  - Tangent Linear G
  - Objective $J(x^{(k)})$
  - Gradient $G^T d = G^T(\mathcal{G}(x^{(k)}) - y)$
- Inner Loop
  - Solve $(G^T G)\delta x^{(k)} = -G^T d$
- $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$



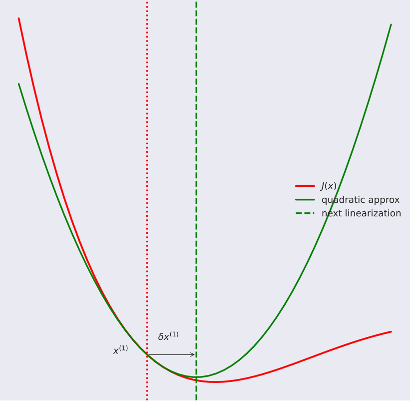
Nested Loops

Set $k = 0$

Repeat until convergence/computational budget spent

Outer Loop

- Evaluate
  - Forward $\mathcal{G}(x^{(k)})$
  - Tangent Linear G
  - Objective $J(x^{(k)})$
  - Gradient $G^T d = G^T(\mathcal{G}(x^{(k)}) - y)$
- Inner Loop
  - Solve $(G^T G)\delta x^{(k)} = -G^T d$
- $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$



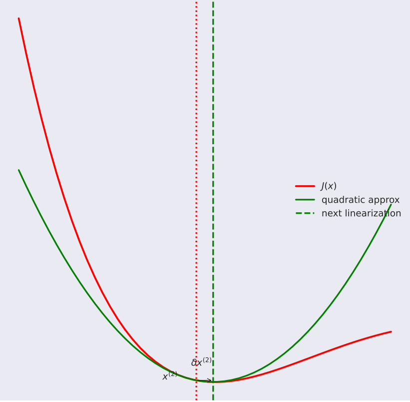
Nested Loops

Set $k = 0$

Repeat until convergence/computational budget spent

Outer Loop

- Evaluate
  - Forward $\mathcal{G}(x^{(k)})$
  - Tangent Linear G
  - Objective $J(x^{(k)})$
  - Gradient $G^T d = G^T(\mathcal{G}(x^{(k)}) - y)$
- Inner Loop
  - Solve $(G^T G)\delta x^{(k)} = -G^T d$
- $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$



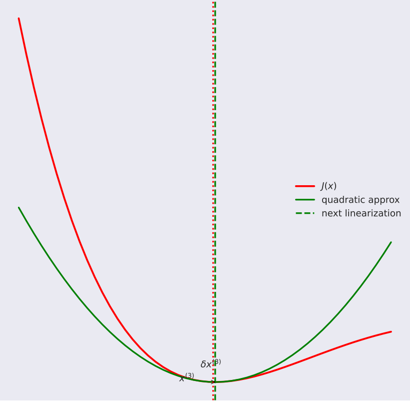
Nested Loops

Set $k = 0$

Repeat until convergence/computational budget spent

Outer Loop

- Evaluate
  - Forward $\mathcal{G}(x^{(k)})$
  - Tangent Linear G
  - Objective $J(x^{(k)})$
  - Gradient $G^T d = G^T(\mathcal{G}(x^{(k)}) - y)$
- Inner Loop
  - Solve $(G^T G)\delta x^{(k)} = -G^T d$
- $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$



ML philosophy

- Trying to learn the forward operator $\mathcal{G} = \mathcal{H} \circ \mathcal{M}$ is not worth the effort
  - Very costly, complex, high-dimensional
  - $d = \mathcal{G}(x) - y$ is central to compute the gradient (ie the direction of descent)
- Use ML to speed up inner loop ?
  - Need to solve $(G^T G)\delta x = -G^T d$
  - How to speed-up iterative methods ?
    - Reduce the number of iterations to convergence
    - Reduce error for constant number of iterations
- Dimension reduction
  - Find a lower dimensional representation of the state space $x \in \mathcal{X}$
  - Lower-dimensional Manifold on which the optimization can take place
  - Non-linear dimension reduction with diffeomorphism

Data-driven preconditioning

Solving linear systems

The increment δx verifies

$$\underbrace{(G^T G)}_{A}\,\delta x = \underbrace{-G^T d}_{b} \tag{12}$$

and using Conjugate Gradient, the error at the kth iteration is bounded according to

$$\|e_k\| \leq 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|e_0\| \tag{13}$$

where $\kappa(A) = |\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}}| \geq \kappa(I_n) = 1$

$\Rightarrow$ Smaller κ = better rate of convergence

(More generally, depends on the whole distribution of the eigenvalues)

[Haben et al., 2011, Gürol et al., 2014, Tabeart et al., 2021, Robert et al., 2006]

Desired properties of preconditioners

Let δx be a solution of

$$A\delta x = b \tag{14}$$

Left Preconditioner

Let H^{-1} be an invertible matrix

δx is also a solution of

$$(H^{-1}A)\delta x = H^{-1}b \tag{15}$$

and we hope that the new linear system is easier to solve

Desired properties:

- H^{-1} symmetric and invertible
- $H^{-1}A$ should be close to I_n
- $\kappa(H^{-1}A) < \kappa(A)$

State dependent preconditioner

One-size-fits-all preconditioners do not exist (or are very simplistic).

Recalling that $A = G^T G = G(x)^T G(x)$ is state-dependent (G TL of the forward model)

State-dependent preconditioner

$$x \longmapsto \text{prec}(x) = H^{-1}(x) \tag{16}$$

which exploits the fact that $G(x)^T G(x)$ is not arbitrary

- Defined according to a numerical model
- Positive-definite matrix and symmetric

Training philosophy

We want $H^{-1}A$ close to I_n

What loss to choose ?

- $\|H^{-1} - A^{-1}\|$
  $\rightarrow$ A^{-1} is what we are trying to avoid computing...
- $\|I_n - H^{-1}A\|$
  $\rightarrow$ quite "unstable" objective in practice, especially for badly conditioned system (local minimum is $H^{-1} = 0$)
- $\|H - A\|$
  $\rightarrow$ but we need to train using H, and use H^{-1} as a preconditioner

Neural Network architecture

Low-rank updates

Let $S = (s_1, \dots, s_r)$ be r vectors of $\mathbb{R}^n$, $r \ll n$

$$H_{\text{LR}}(S) = I_n + SS^T \tag{17}$$

$$H_{\text{LR}}^{-1}(S) = I_n - S \underbrace{\left(I_r - S^T S\right)^{-1}}_{\text{inv of dim } r} S^T \tag{18}$$

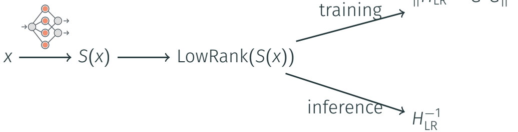


Figure 1: Flowchart for prec. training using low rank matrices

$\rightarrow$ Did not give results (for a lack of structure ?)

Deflation-like preconditioner

$S = (s_1, \ldots, s_r)$ has r orthonormal columns ($S^T S = I_r$), and $(\lambda_1, \ldots, \lambda_r) \in \mathbb{R}^r$, $\lambda_i > 0$

$$H_{\text{defl}}(S, \lambda) = I_n + \sum_{i=1}^{r} (\lambda_i - 1) s_i s_i^T \tag{19}$$

$$H_{\text{defl}}^{-1}(S, \lambda) = I_n + \sum_{i=1}^{r} \left(\frac{1}{\lambda_i} - 1 \right) s_i s_i^T \tag{20}$$

$$\tag{21}$$

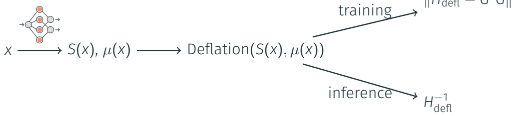


Figure 2: Flowchart for prec. training using deflation

Limited Memory Preconditioners [Tshimanga, 2007, Gratton et al., 2011]

Let $S = (s_1, \dots, s_r)$ be r vectors of $\mathbb{R}^n$, $r \ll n$

$$H_{\mathrm{LMP}}^{-1}(S, AS) = \left(\boxed{I_n} - \boxed{S}\,\boxed{\Sigma}\,\boxed{AS}^T\right)\left(\boxed{I_n} - \boxed{AS}\,\boxed{\Sigma}\,\boxed{S}^T\right) + \mu\,\boxed{S}\,\boxed{\Sigma}\,\boxed{S}^T \tag{22}$$

$$H_{\mathrm{LMP}}(S, AS) = \boxed{I_n} + \boxed{AS}\,\boxed{\Sigma}\,\boxed{AS}^T - \frac{1}{\mu}\boxed{S}\,\boxed{\Gamma}\,\boxed{S}^T \tag{23}$$

Given $S \in \mathbb{R}^{n\times r}$ and $AS \in \mathbb{R}^{n\times r}$, we can construct H_{LMP} and H_{LMP}^{-1} directly, with nice properties (even though Σ and Γ require the inversion of a $r \times r$ matrix)

Flexible preconditioners:

- $(s_1, \dots, s_r)$ eigenvalues of $A \to$ Spectral Preconditioner
- $(s_1, \dots, s_r)$ CG descent vectors $\to$ QN preconditioner

LMP construction using leap-forward NN

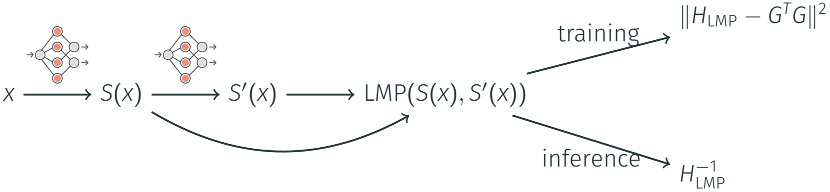


Figure 3: Neural Network architecture for LMP

But: No particular structure imposed for $S' \Rightarrow (H_{\mathrm{LMP}}(S, S'))^{-1} \neq H_{\mathrm{LMP}}^{-1}(S, S')$

However: Good results were still obtained, even though the exact inverse is not used

LMP: self adjoint variation

Force the self-adjointness of the operator $S \mapsto S'$ (which should be $= AS$) by constructing at the same time a low-rank approximation of A

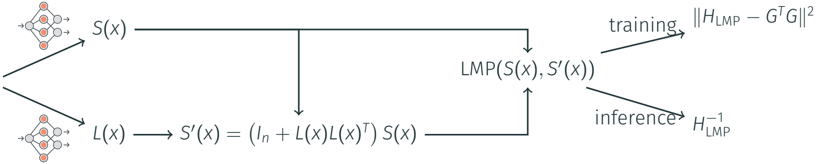


Figure 4: Neural Network architecture for symmetric LMP

with $L(x) \in \mathbb{R}^{n \times n'}$

Training setting: Dataset

For an input x, we assume to have access to $G(x)^T G(x)$,

$\rightarrow$ Training dataset: $\mathcal{D} = \{(\underbrace{x_i}_{\in \mathbb{R}^n}, \underbrace{G(x_i)^T G(x_i)}_{\in \mathbb{R}^{n \times n}})\}$

- Very large in memory when dimension increases
- We access G only as an operator: $\mathrm{TL}(x, z) = G(x) \cdot z$
- Same for adjoint: $\mathrm{Adj}(x, y) = G(x)^T \cdot y$
- Constructing $G(x)$ would require n call to the TL

Less storage intensive solution: Iterable Datasets

Estimate the L_2 norm using random Gaussian vectors:

Matrix norm estimation

For a matrix M, and $z \sim \mathcal{N}(0, I)$

$$\mathbb{E}_Z \left[\|Mz\|_2^2 \right] = \|M\|_{\mathbf{F}}^2 \tag{24}$$

$\rightarrow$ Iterable Dataset: $\mathcal{D} = \{(\underbrace{x_i}_{\in\mathbb{R}^n}, \underbrace{Z_i}_{\in\mathbb{R}^{n\times n_Z}}, \underbrace{G(x_i)^T G(x_i) Z_i}_{\in\mathbb{R}^{n\times n_Z}})_i\}$ where Z_i has n_Z columns iid $\mathcal{N}(0, I)$

The loss for a data point becomes then

$$\mathcal{L}_\theta(x_i) = \sum_{j=1}^{n_Z} \|H_\theta(x_i) z_i^j - G^T(x_i) G(x_i) z_i^j\|_2^2 \tag{25}$$

and we can generate the dataset ”on the fly”, and train the network in an online manner

Regularization

A-conjugacy

Let $S = \left(s_1 \middle| s_2 \middle| \ldots \middle| s_r\right)$. The column vectors of S are A-conjugate when $S^T A S$ is diagonal

Since S' is supposed to be AS

$$K = S^T S' \text{ has general term } K_{i,j} = \langle s_i, s_j' \rangle = s_i^T s_j' \approx s_i^T A s_j \tag{26}$$

$$R_{\text{conjugacy}}(S) \propto \| K \| \tag{27}$$

But we may also add regularization on symmetry of the LMP, orthonormalization of the outputs etc...

Example on Lorenz96 (40D)

Statistics

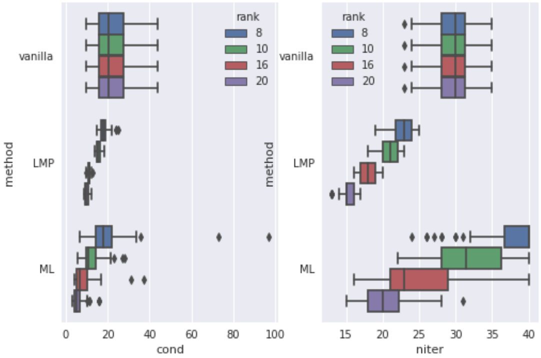
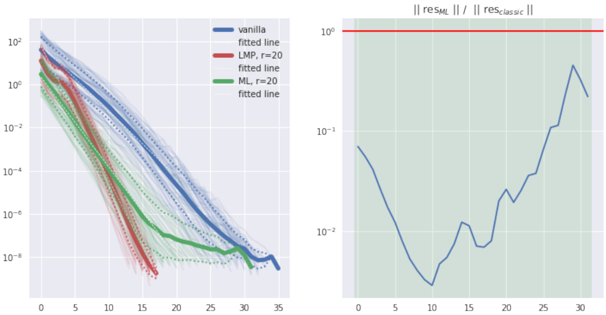


Figure 5: Statistics on inference model

Inner loop CV: leap forward LMP ($r = 20$)

Comparison:

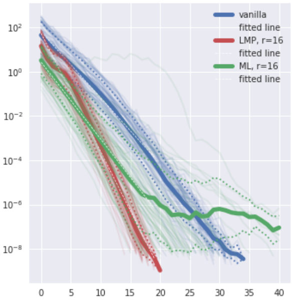
- Baseline: no preconditioner
- Spectral LMP: computation of eigenvectors for every inner loop
- ML-LMP: preconditioned inner loop using ”leap forward” LMP

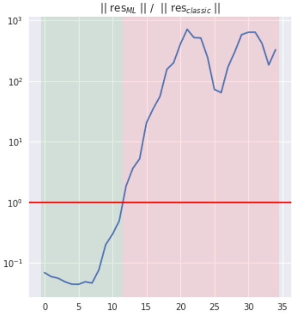


Inner loop CV: leap forward LMP ($r = 16$)

Comparison:

- Baseline: no preconditioner
- Spectral LMP: computation of eigenvectors for every inner loop
- ML-LMP: preconditioned inner loop using "leap forward" LMP

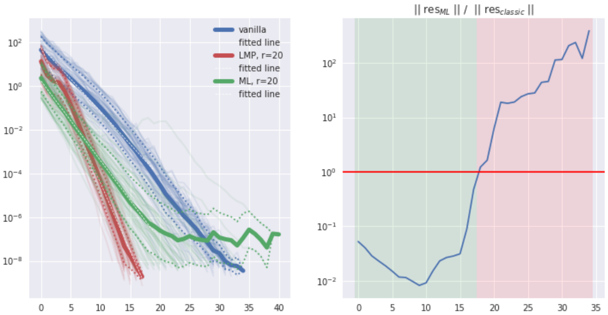




Inner loop CV: self adjoint LMP ($r = 20$)

Comparison:

- Baseline: no preconditioner
- Spectral LMP: computation of eigenvectors for every inner loop
- ML-LMP: preconditioned inner loop using self adjoint LMP



Work in progress / Open questions

- Apply this preconditioner in 4D-Var ✓
- Output both S_θ and AS_θ ✓
- How to train without explicit access to A
    - Online training ?
    - Use information of $G^T(x)G(x)Z$ (Link to REVD [Daužickaitė et al., 2021])
- How to get more consistent results in 4D-Var ?
- Which regularization to use
- Apply to system with worse conditioning (QG) and/or higher dimension (QG / KS)
- Exploit conditioning using the prior (”$(B + G^TG) \rightarrow (I + \tilde{G}^T\tilde{G})$”)
- Extension to weak 4DVar
- Adapt to time-varying observation operators

Daužickaitė, I., Lawless, A. S., Scott, J. A., and van Leeuwen, P. J. (2021).
Randomised preconditioning for the forcing formulation of weak constraint 4D-Var.
Quarterly Journal of the Royal Meteorological Society, 147(740):3719–3734.

Gratton, S., Laloyaux, P., Sartenaer, A., and Tshimanga, J. (2011).
A reduced and limited-memory preconditioned approach for the 4D-Var data-assimilation problem: Reduced 4D-Var using the LMP Preconditioner.
Quarterly Journal of the Royal Meteorological Society, 137(655):452–466.

Gürol, S., Weaver, A. T., Moore, A. M., Piacentini, A., Arango, H. G., and Gratton, S. (2014).
B -preconditioned minimization algorithms for variational data assimilation with the dual formulation: B -preconditioned minimization algorithms.
Quarterly Journal of the Royal Meteorological Society, 140(679):539–556.

Haben, S., Lawless, A., and Nichols, N. (2011).
Conditioning and preconditioning of the variational data assimilation problem.
Computers & Fluids, 46(1):252–256.

Robert, C., Blayo, E., and Verron, J. (2006).
Reduced-order 4D-Var: A preconditioner for the Incremental 4D-Var data assimilation method: A PRECONDITIONER FOR THE 4D-VAR METHOD.
Geophysical Research Letters, 33(18):n/a–n/a.

Tabeart, J. M., Dance, S. L., Lawless, A. S., Nichols, N. K., and Waller, J. A. (2021).
New bounds on the condition number of the Hessian of the preconditioned variational data assimilation problem.

Tshimanga, J. (2007).
On a class of limited memory preconditioners for large-scale nonlinear least-squares problems (with application to variational ocean data assimilation).