Physically consistent global atmospheric data assimilation with machine learning in latent space

Turner Group Meeting PD

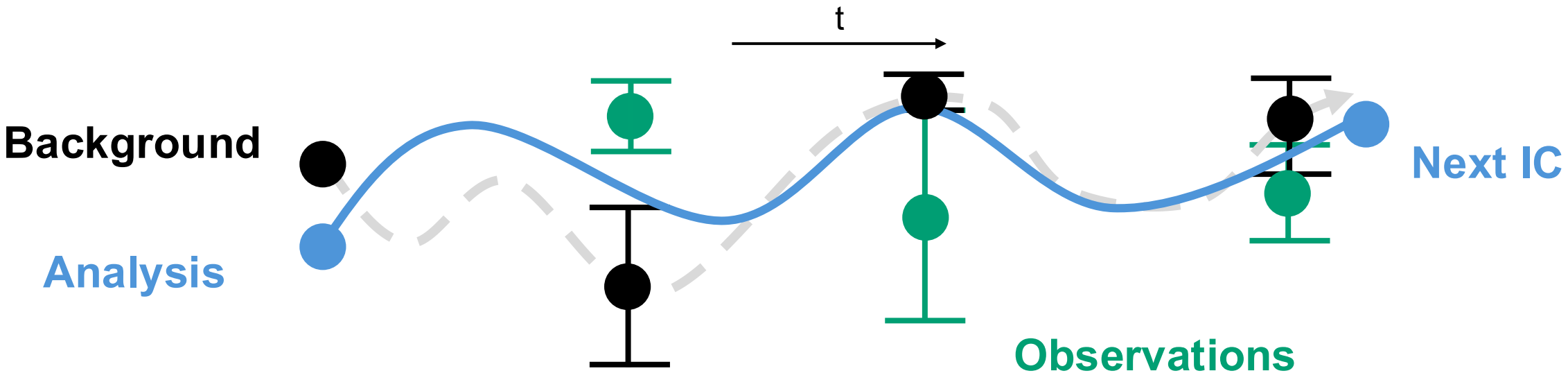
Fan et al. 2026

Eliot Kim

February 11th, 2026

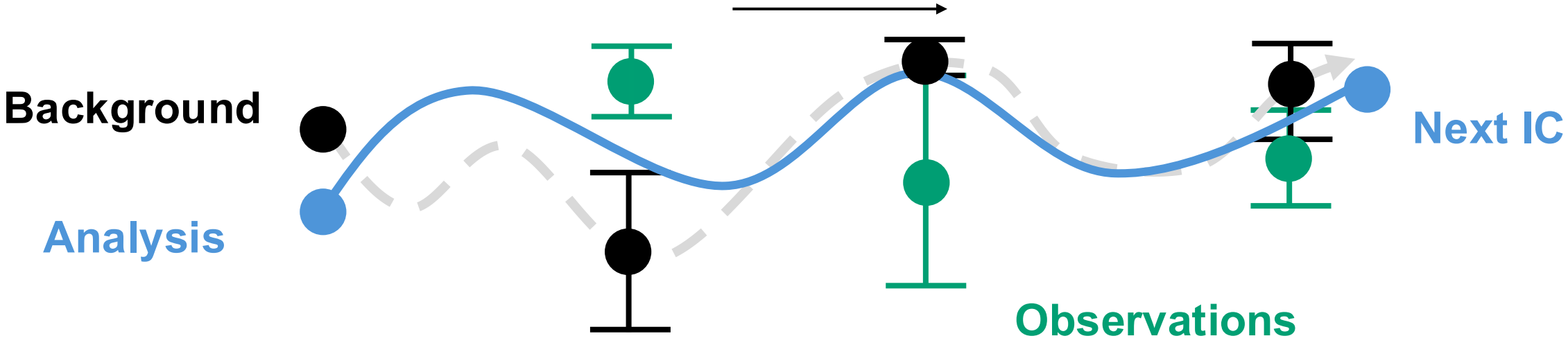
Annotated by Andy Liu, April 16, 2026

What is Data Assimilation (DA)?



Given initial **model estimates,** **observations**, and model and observation errors, what is the **most likely atmospheric state (analysis)**?

Where is DA used?



- Real-time Forecasting: Optimizing initial conditions
  - Ex: Operational weather forecasts @ ECMWF, Met Office, NOAA, etc.
- Re-analyses: Re-constructing best estimate of prior atmospheric states
  - Ex: **ERA5** for weather and climate
  - Ex: **CAMS** for atmospheric composition

Main Prior Approaches

1) Traditional data assimilation (DA)

+ statistical rigor
+ can be physically consistent
– requires lots of computational and labor resources

2) ML-based approaches (e.g. diffusion or end-to-end models)

+ efficient → (1) faster inference speed (matrix ops); (2) Autodifferentiation (Greg's presentation)
– lack "rigorous incorporation of prior information"
– not physically consistent

What's new?

- Latent DA enables near-diagonal background error covariance matrix (**B**)

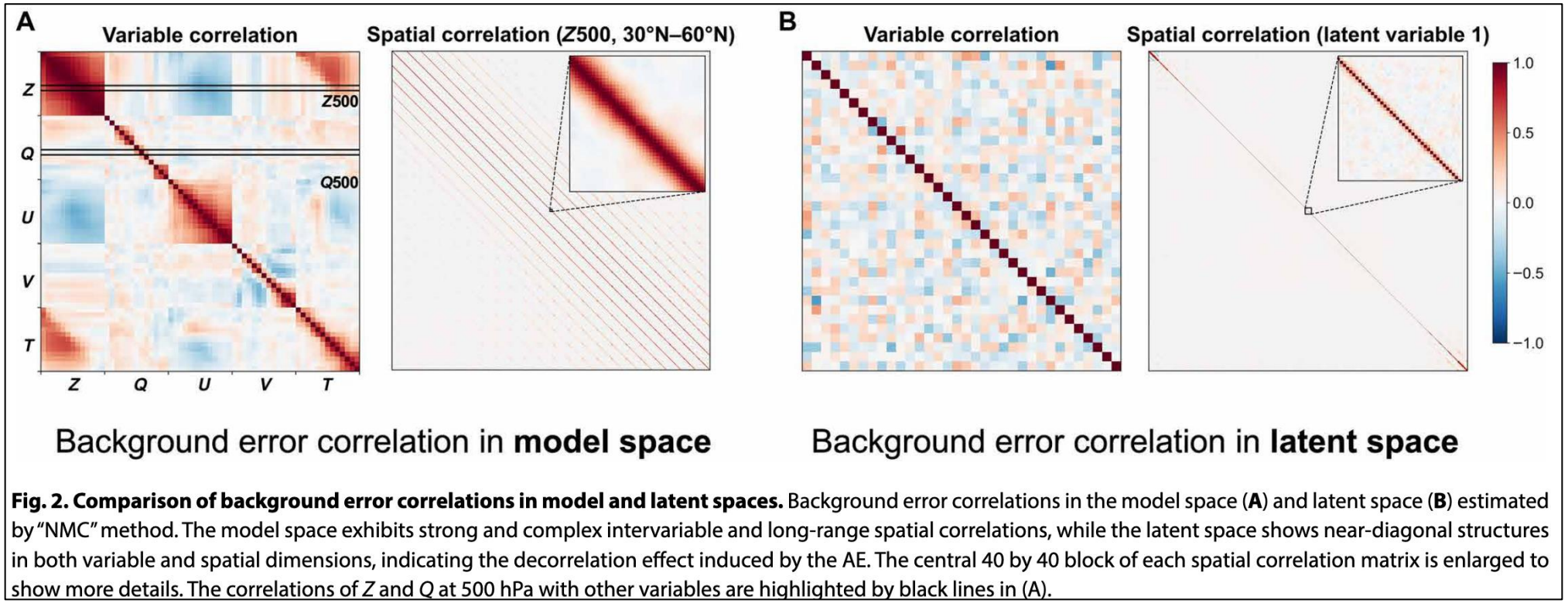


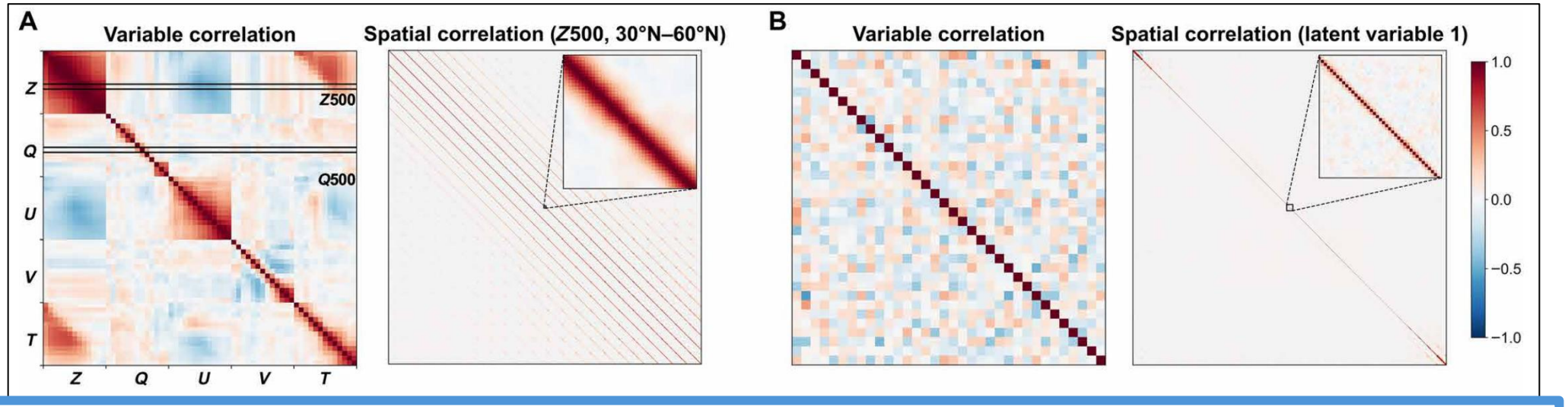
Fig. 2. Comparison of background error correlations in model and latent spaces. Background error correlations in the model space (**A**) and latent space (**B**) estimated by "NMC" method. The model space exhibits strong and complex intervariable and long-range spatial correlations, while the latent space shows near-diagonal structures in both variable and spatial dimensions, indicating the decorrelation effect induced by the AE. The central 40 by 40 block of each spatial correlation matrix is enlarged to show more details. The correlations of *Z* and *Q* at 500 hPa with other variables are highlighted by black lines in (A).

+ Maintains physically consistent covariance
– Huge
– Expensive to compute

+ Much easier to compute **B**
+ Much lower memory to store **B**
+ Faster DA algorithm

What's new?

- Latent DA enables <u>near-diagonal background error covariance matrix (**B**)</u>



$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

This guy

+ Maintains physically consistent covariance
– Huge
– Expensive to compute

+ Much easier to compute **B**
+ Much lower memory to store **B**
+ Faster DA algorithm

What's new?

- Latent DA enables <u>near-diagonal background error covariance matrix (**B**)</u>

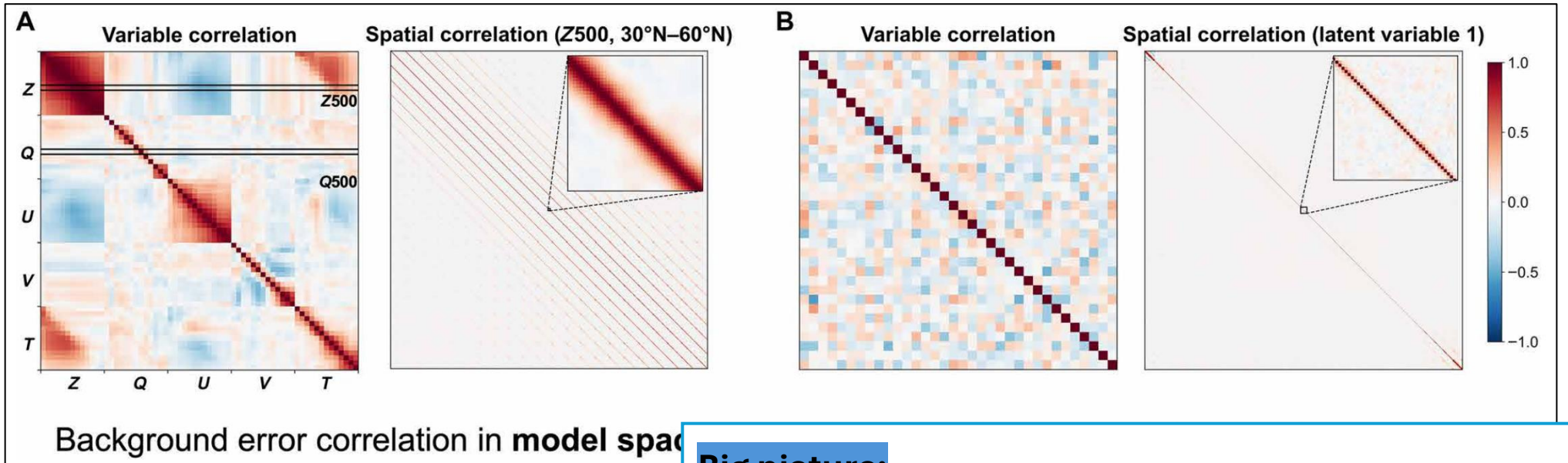


Fig. 2. Comparison of background error correlations in model and late
by "NMC" method. The model space exhibits strong and complex intervari
in both variable and spatial dimensions, indicating the decorrelation effec
show more details. The correlations of *Z* and *Q* at 500 hPa with other varia

Big picture:
Autoencoder capturing the nonlinear physics, and assimilating data in the "simple and nice" latent space.

+ Maintains physically consistent covariance

+ Much easier to compute **B**
+ Much lower memory to store **B**
+ Faster

Also, easier to invert B.
Approximate near-diagonal B as a diagonal matrix, Then inv B is the scalar-inverse of each diagonal entry.

B is a sparse matrix (M<N)
Regular space, $O(N^2)$, N: size of regular space
Latent space O(M), M: size of latent space

What's new?

- Latent DA enables near-diagonal background error covariance matrix (**B**)

- Multivariate latent DA

What's new?

- Latent DA enables near-diagonal background error covariance matrix (**B**)
- Multivariate latent DA
- Analysis is more accurate than traditional 4D-Var

What's new?

- Latent DA enables near-diagonal background error covariance matrix (**B**)
- Multivariate latent DA
- Analysis is more accurate than traditional 4D-Var
- Maintains physically consistent response to perturbations

Decoder is doing the heavy-lifting here

Model Set-Up

Autoencoder

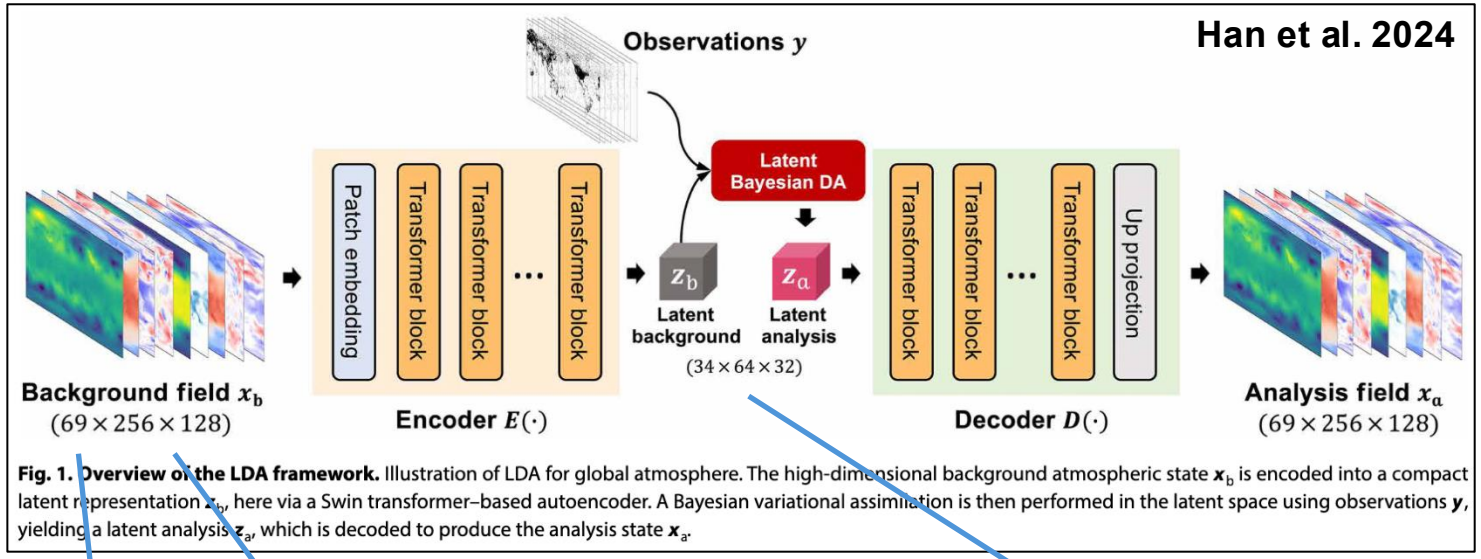


Fig. 1. Overview of the LDA framework. Illustration of LDA for global atmosphere. The high-dimensional background atmospheric state $\boldsymbol{x}_b$ is encoded into a compact latent representation $\boldsymbol{z}_b$, here via a Swin transformer–based autoencoder. A Bayesian variational assimilation is then performed in the latent space using observations $\boldsymbol{y}$, yielding a latent analysis $\boldsymbol{z}_a$, which is decoded to produce the analysis state $\boldsymbol{x}_a$.

5 variables x 13 levels
+ 4 surface variables = 69

1.4º spatial resolution

Compressed to ~5.6º spatial resolution and 34 latent variable-levels

Model Set-Up

Autoencoder

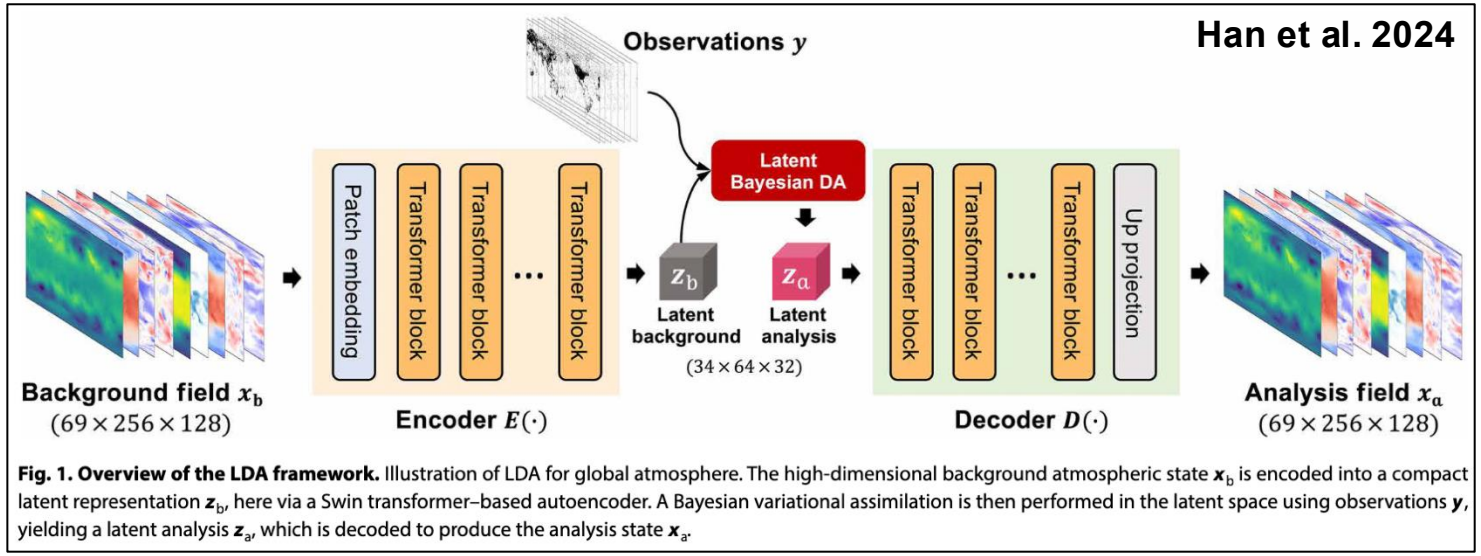
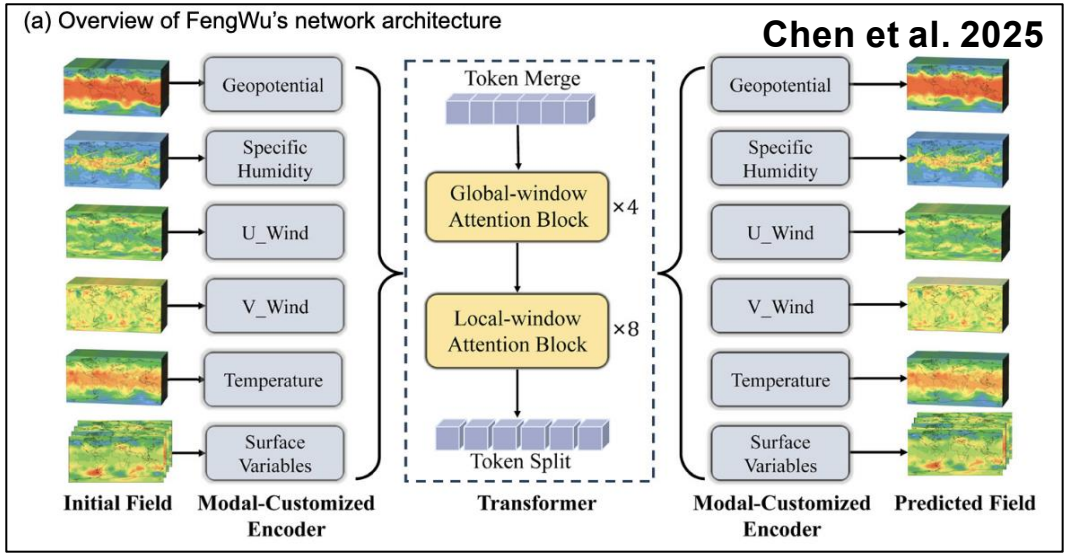


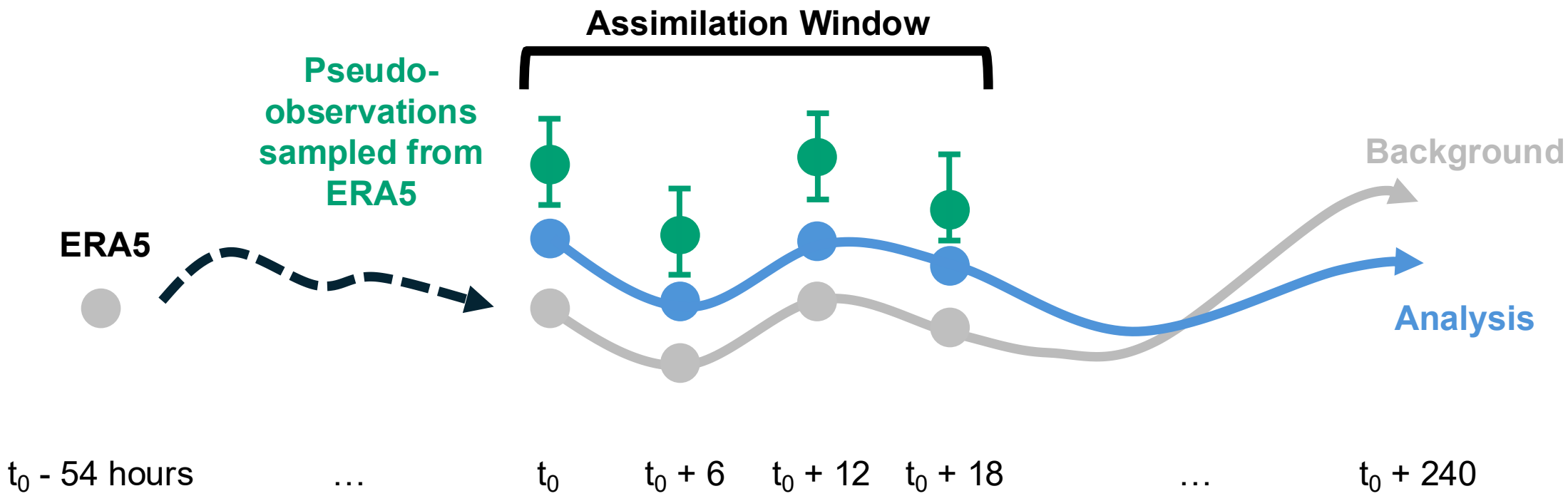
Fig. 1. Overview of the LDA framework. Illustration of LDA for global atmosphere. The high-dimensional background atmospheric state x_b is encoded into a compact latent representation z_b, here via a Swin transformer–based autoencoder. A Bayesian variational assimilation is then performed in the latent space using observations y, yielding a latent analysis z_a, which is decoded to produce the analysis state x_a.

Forecast Model: FengWu



- SOTA DL weather model
- ~750M parameters
- Similar to Aurora
- Both AE and FengWu trained on 1979-2015 ERA5 fields

DA Experiment 1: OSSE with ERA5

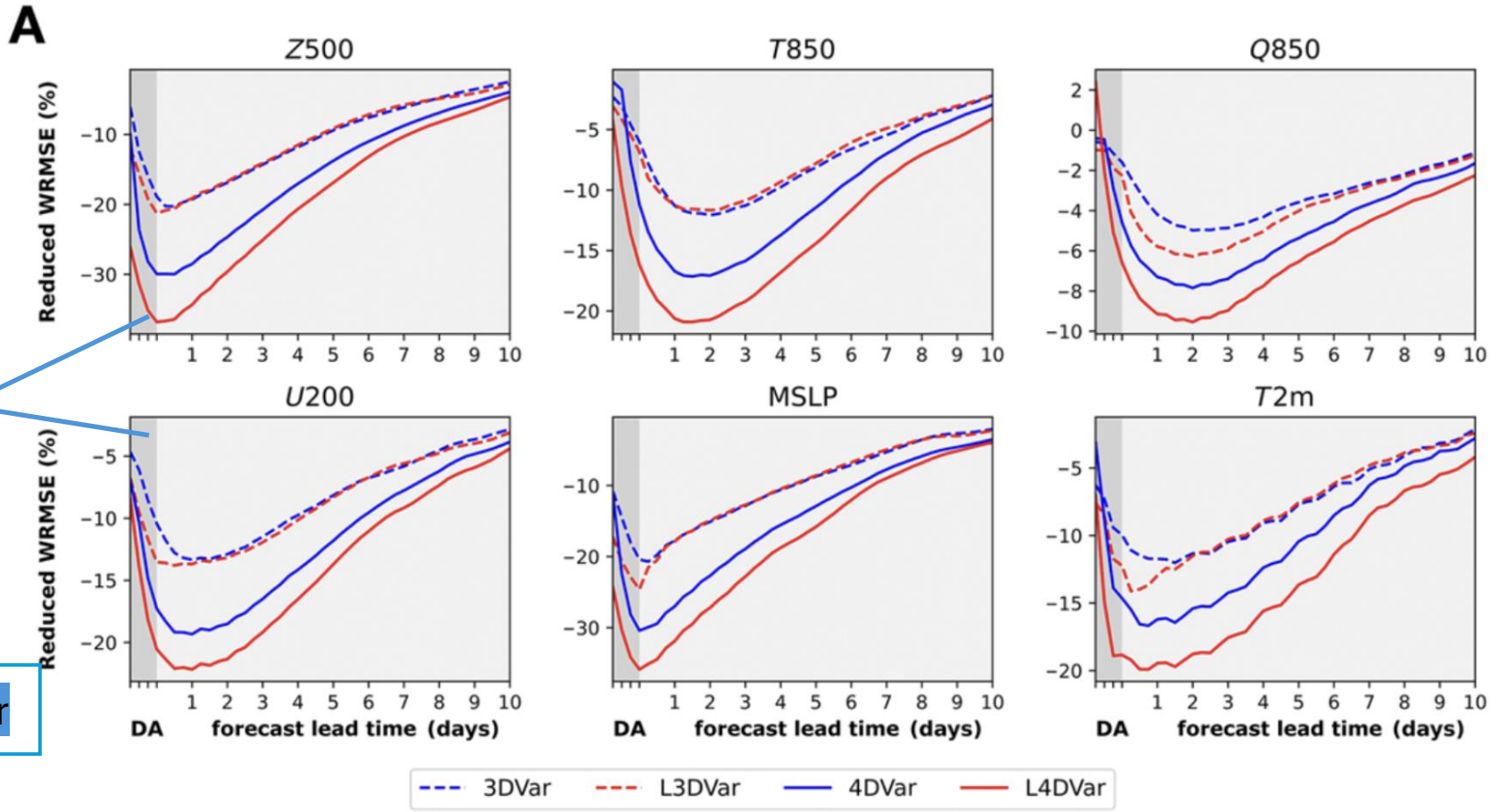


DA Experiment 1: OSSE with ERA5

WRMSE = latitude-weighted RMSE

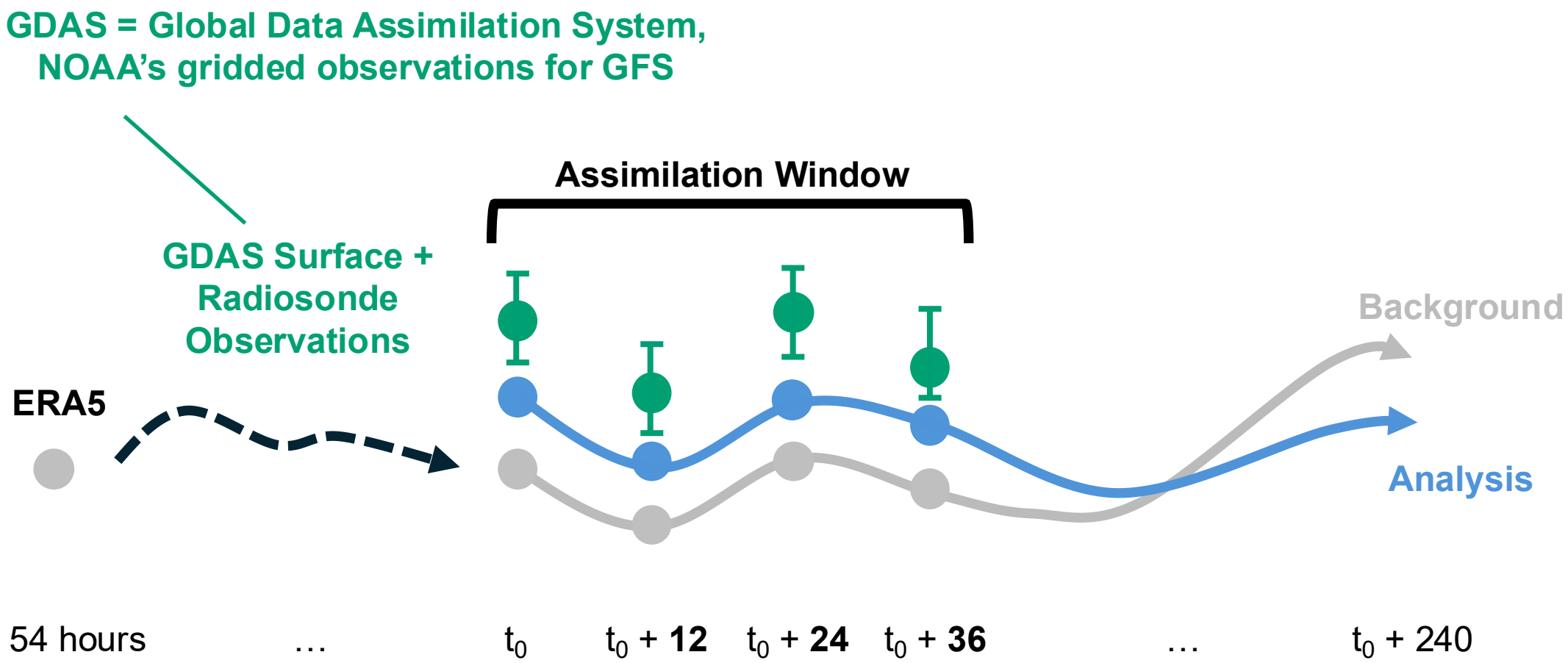
Assimilation Window

Lower is better

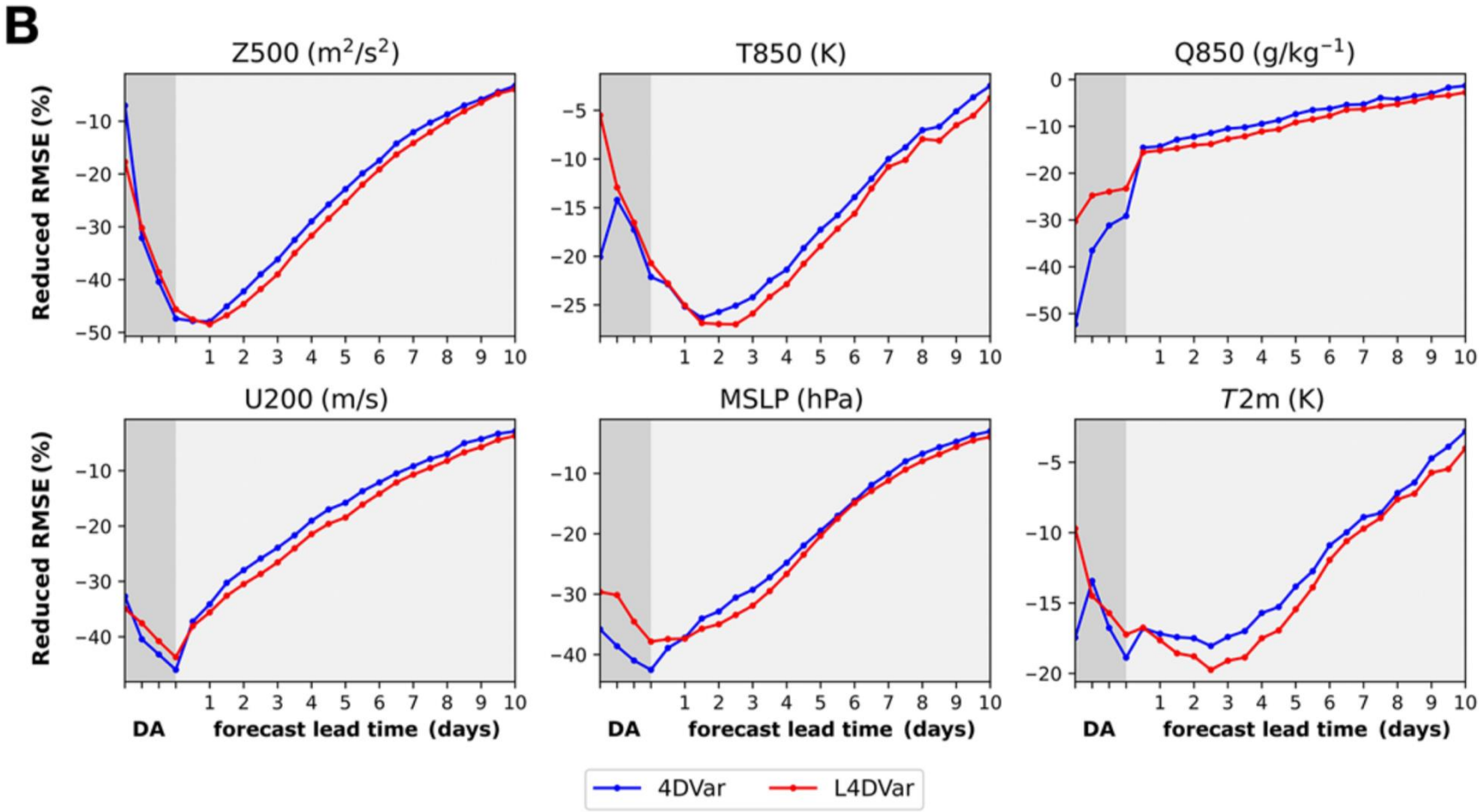


Latent 3D-Var ≈ traditional 3D-Var
but Latent 4D-Var outperforms traditional 4D-Var!
"model dynamics can be effectively used and play a notable role in LDA"

DA Experiment 2: GDAS Assimilation



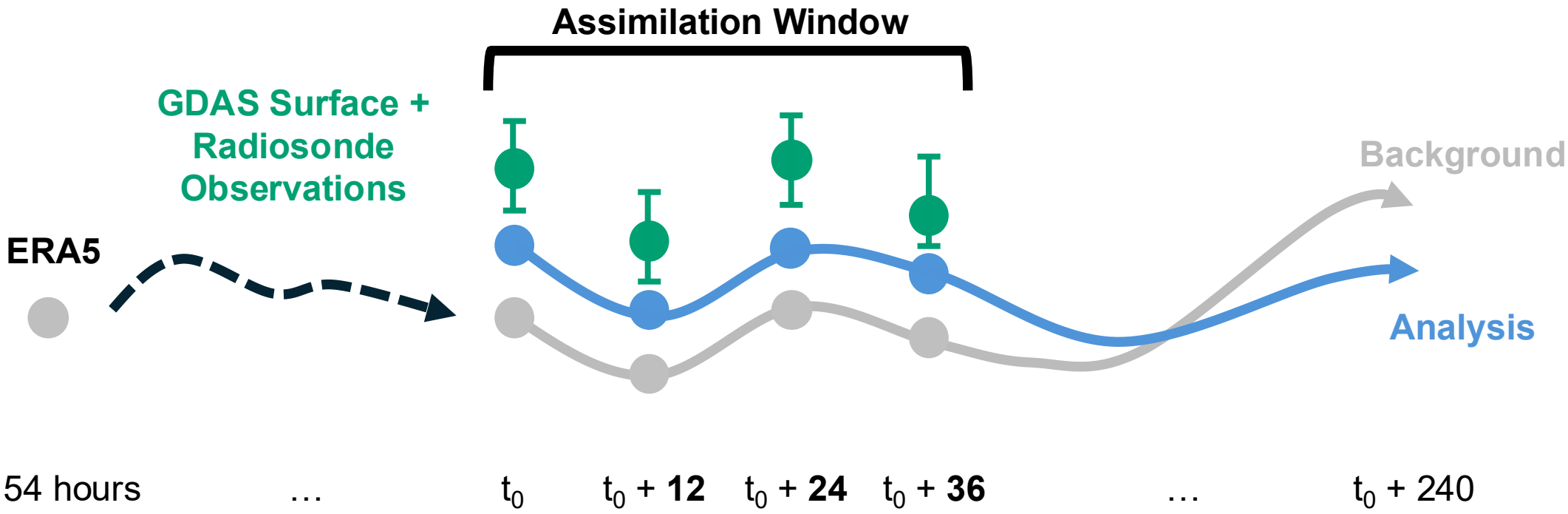
DA Experiment 2: GDAS Assimilation



Latent 4D-Var beats traditional 4D-Var for >1-day lead-times

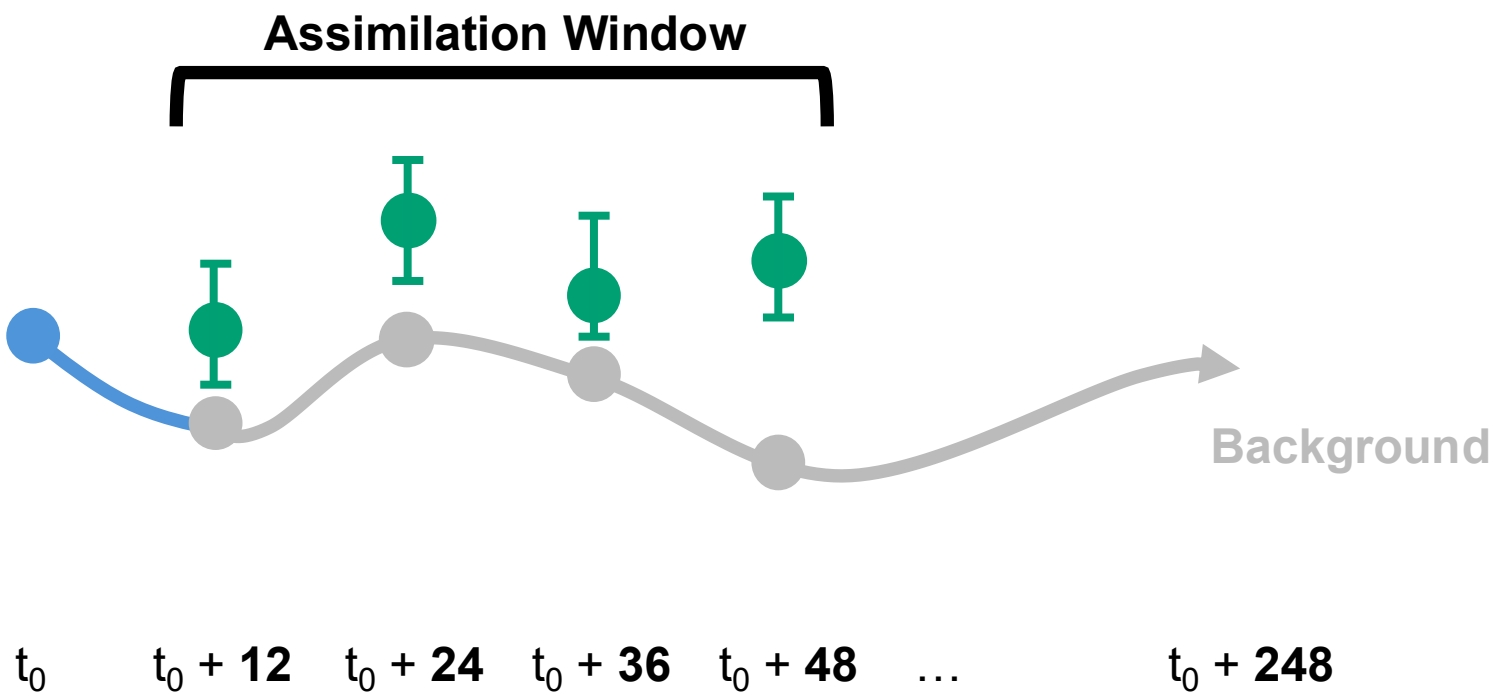
DA Experiment 3: Cycling Assimilation

Cycle 1



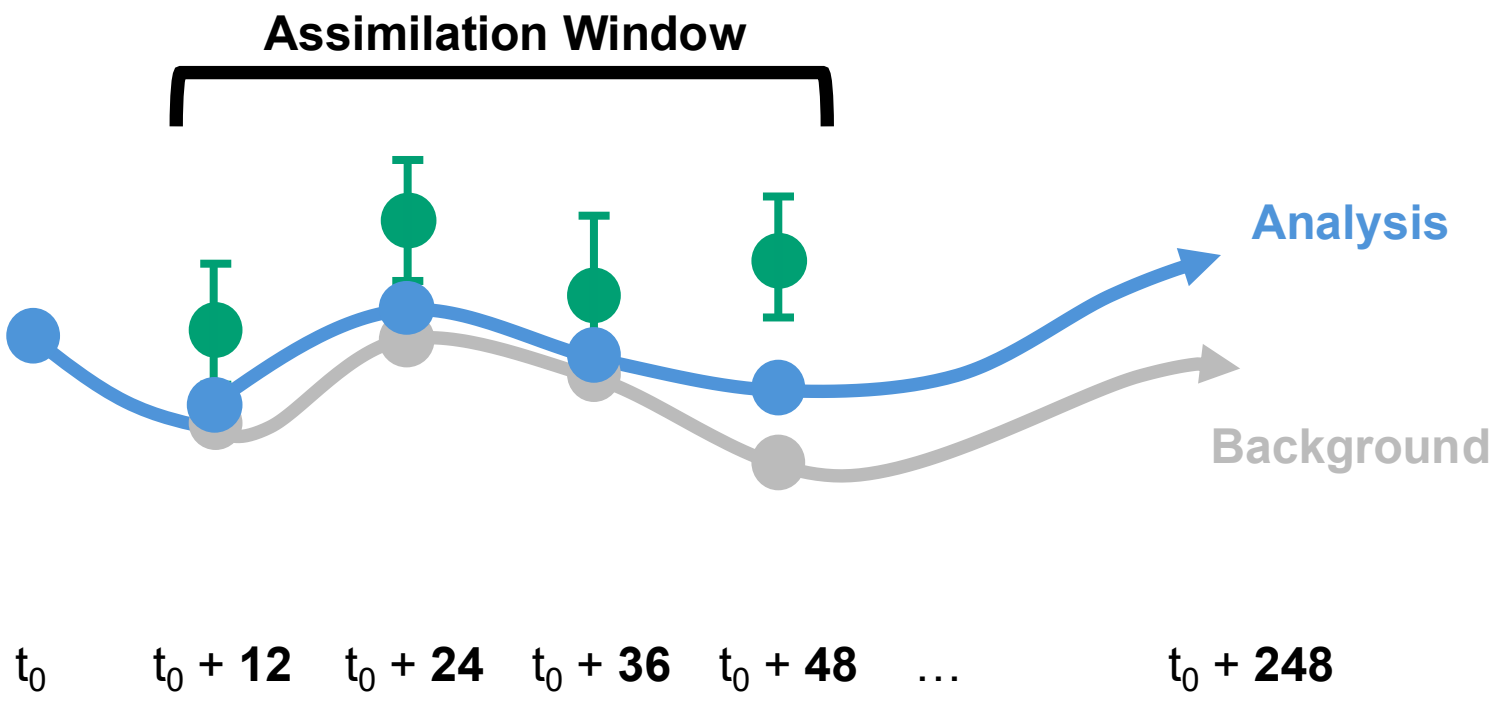
DA Experiment 3: Cycling Assimilation

Cycle 2

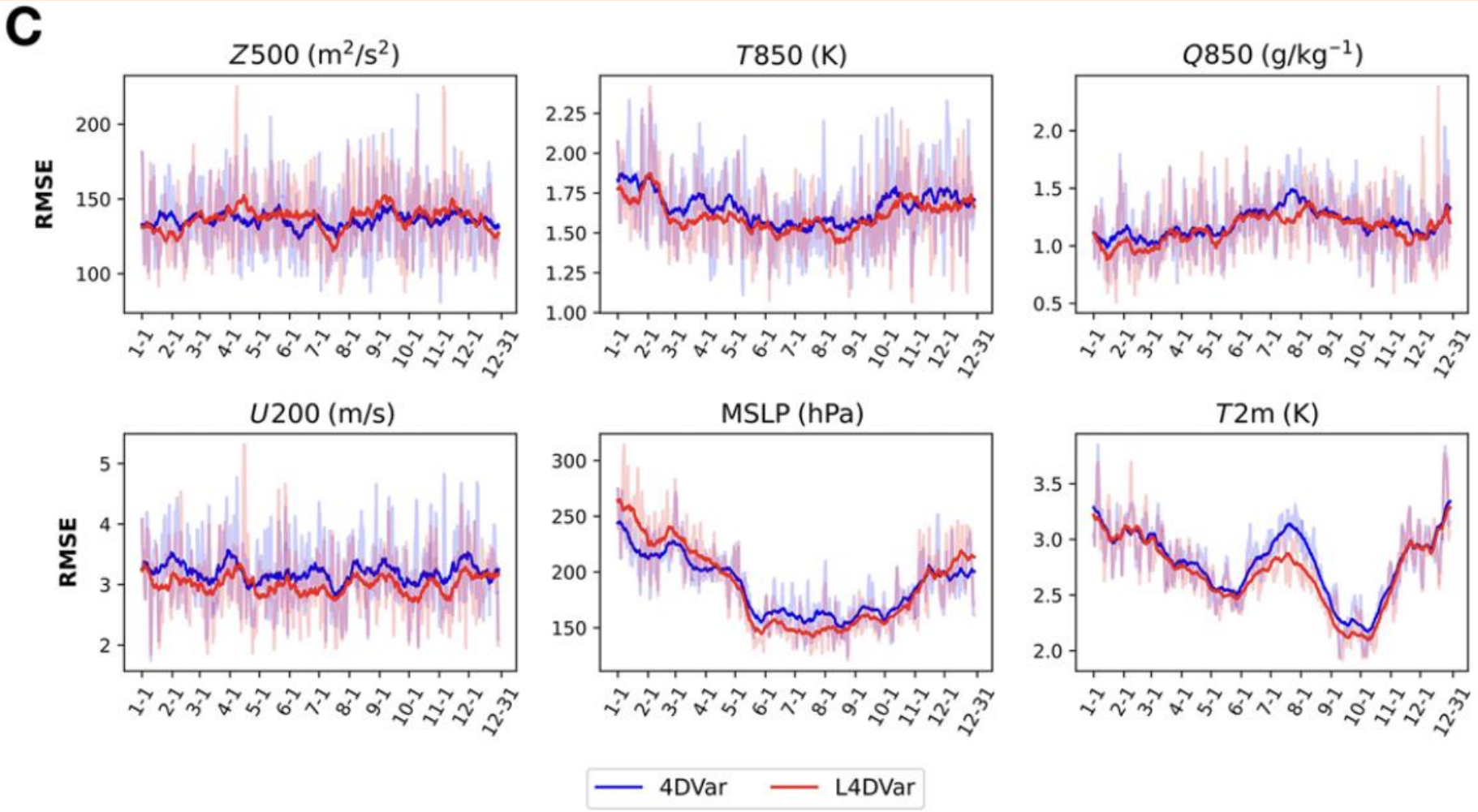


DA Experiment 3: Cycling Assimilation

Cycle 2

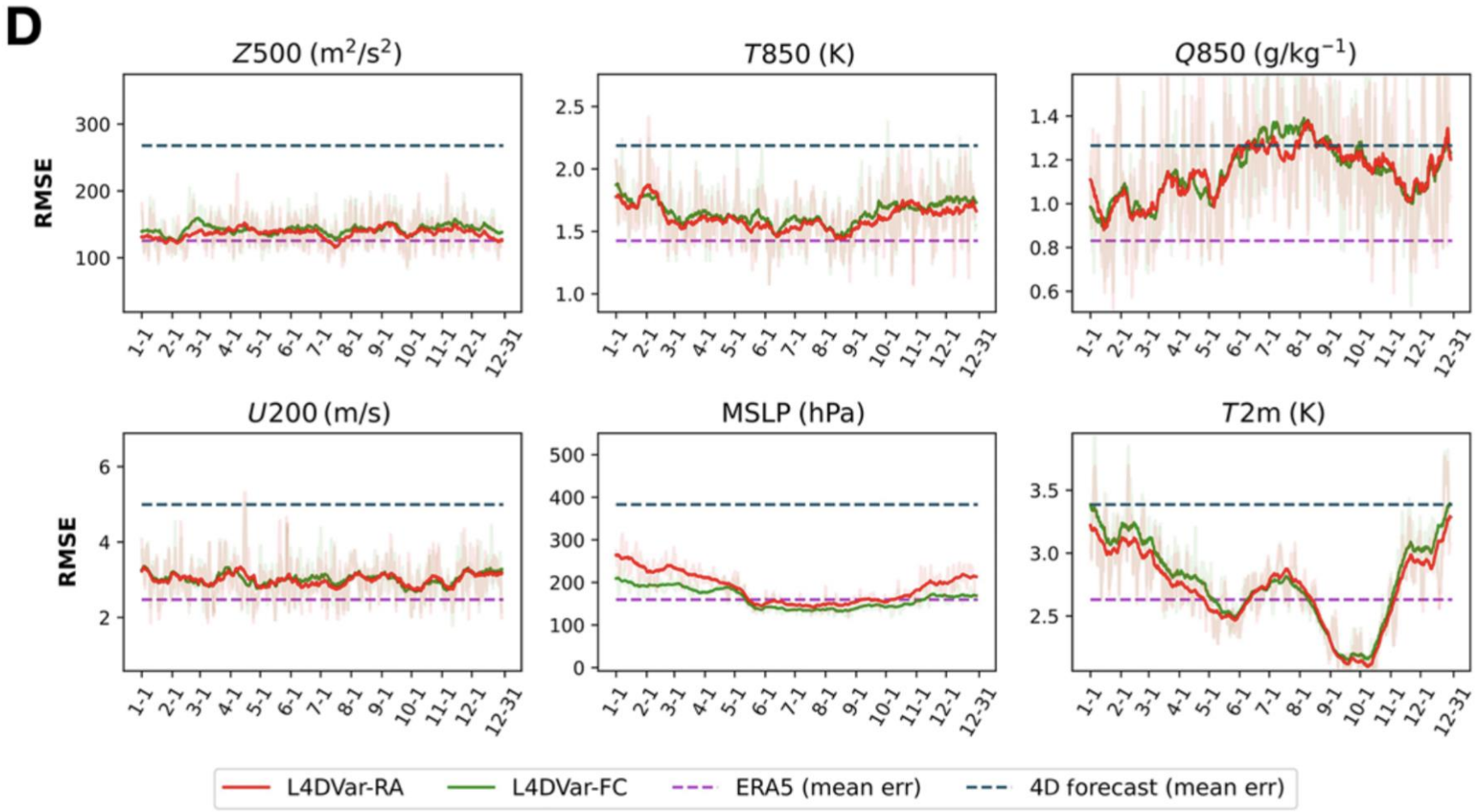


DA Experiment 3: Cycling Assimilation



Latent 4D-Var outperforms 4D-Var against held-out observations

DA Experiment 4: Autoencoder of forecasts, not ERA5



Latent DA with forecast autoencoder nearly matches performance of original ERA5 autoencoder

Physical Responses to Perturbations

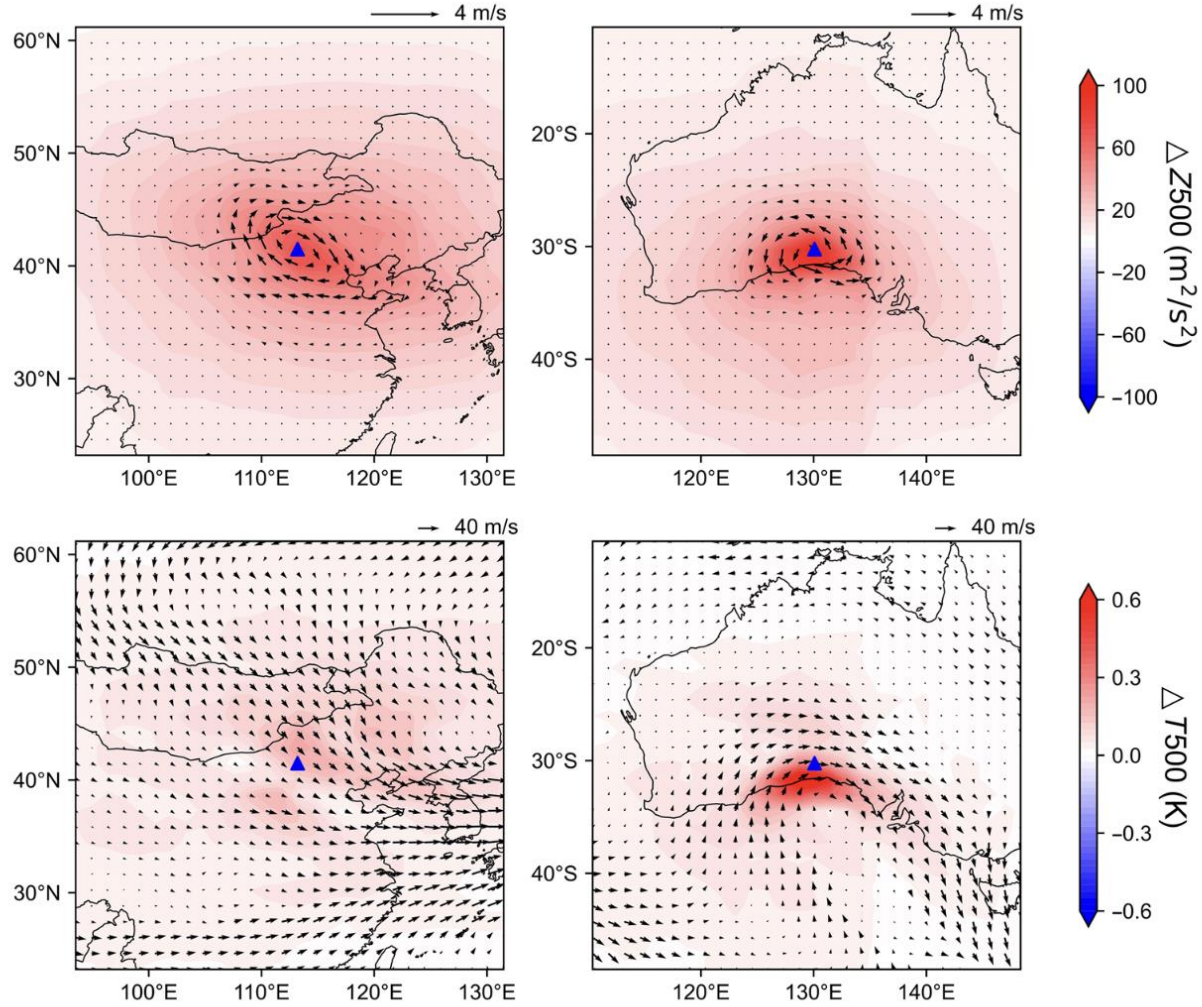


Fig. 4. Single-observation experiments with L3DVar using a static, diagonal B_z. Analysis increments from a perturbation at +200 m^2/s^2 to geopotential at 500 hPa (Z500) over China (left) and Australia (right), using ERA5 reanalysis at 0000 UTC on 1 December 2017 as the background. Top panels show geopotential and wind increments; bottom panels show temperature increments and the background wind field. Perturbation locations are marked with blue triangles. Note that the wind vector scales are different across panels.

Wind and temperature fields respond reasonably to geopotential perturbation

Would 4D-Var give physical responses?

Latent variables influence model-space variables…?

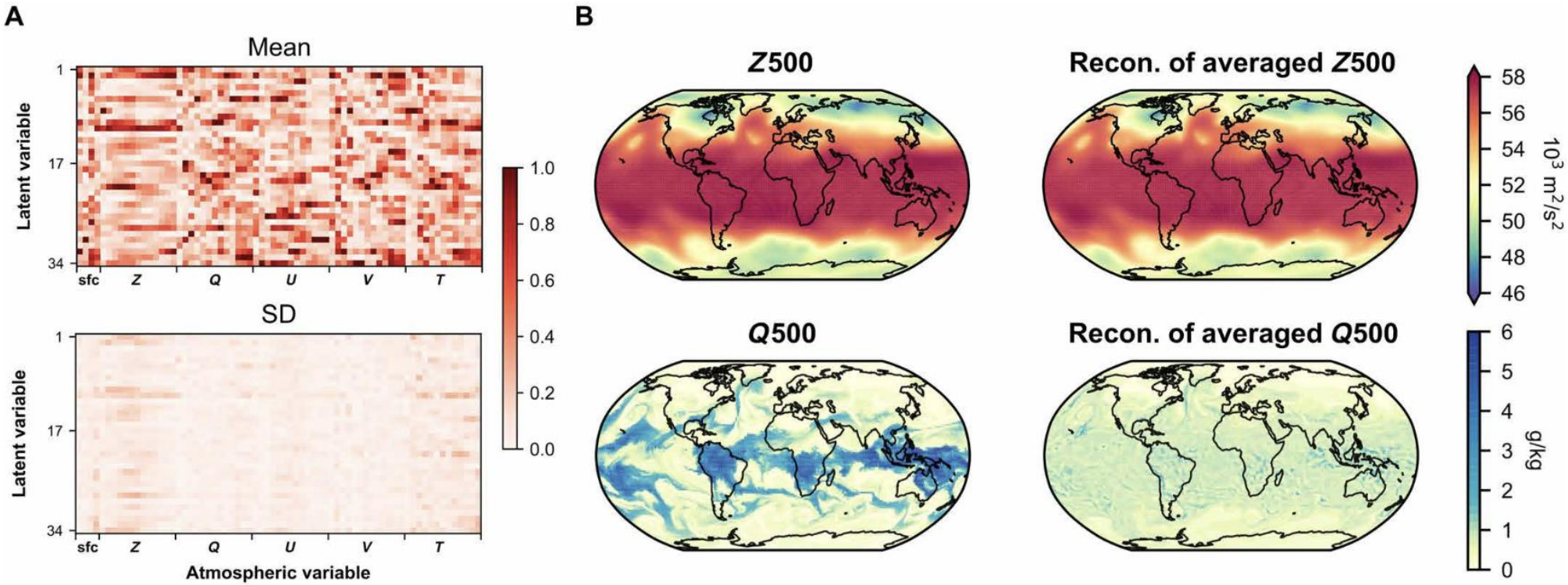


Fig. 5. Evidence of atmospheric multivariate consistency captured in the latent space. (**A**) Influence of latent variables on atmospheric variables, quantified for each atmospheric variable as the spatial mean of the absolute increment in the decoded field induced by perturbing the latent state $\boldsymbol{z}$ along each latent-variable component of $\Delta\boldsymbol{z}$ (first axis). For each sample, we apply min-max normalization per atmospheric variable (columnwise) to highlight its sensitivity across different latent variables. Shown are the mean and variance over 10,000 experiments, with $\boldsymbol{z}$ and $\Delta\boldsymbol{z}$ obtained from randomly selected pairs in the ERA5 test dataset. sfc, surface variables. (**B**) AE reconstruction of physically imbalanced inputs. The fields of *Z*500 and *Q*500 at 0000 UTC on 1 January 2017 are first averaged globally and then passed through the AE.

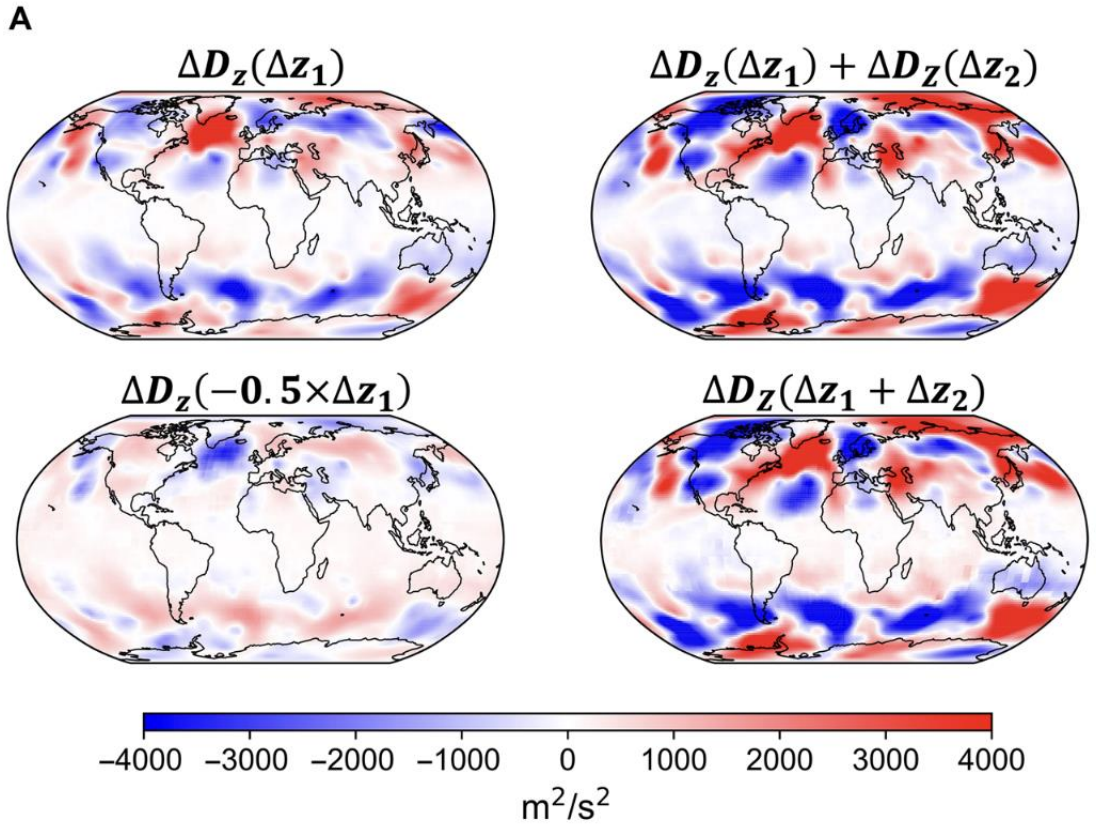
Strong vertical correlation

Weak vertical correlation

What does this figure show?

Uhhh… I don't know…

Linear Responses of Decoder



Latent DA = traditional model-space DA if and only if **decoder is locally linear and error-free**

Linear Responses of Decoder

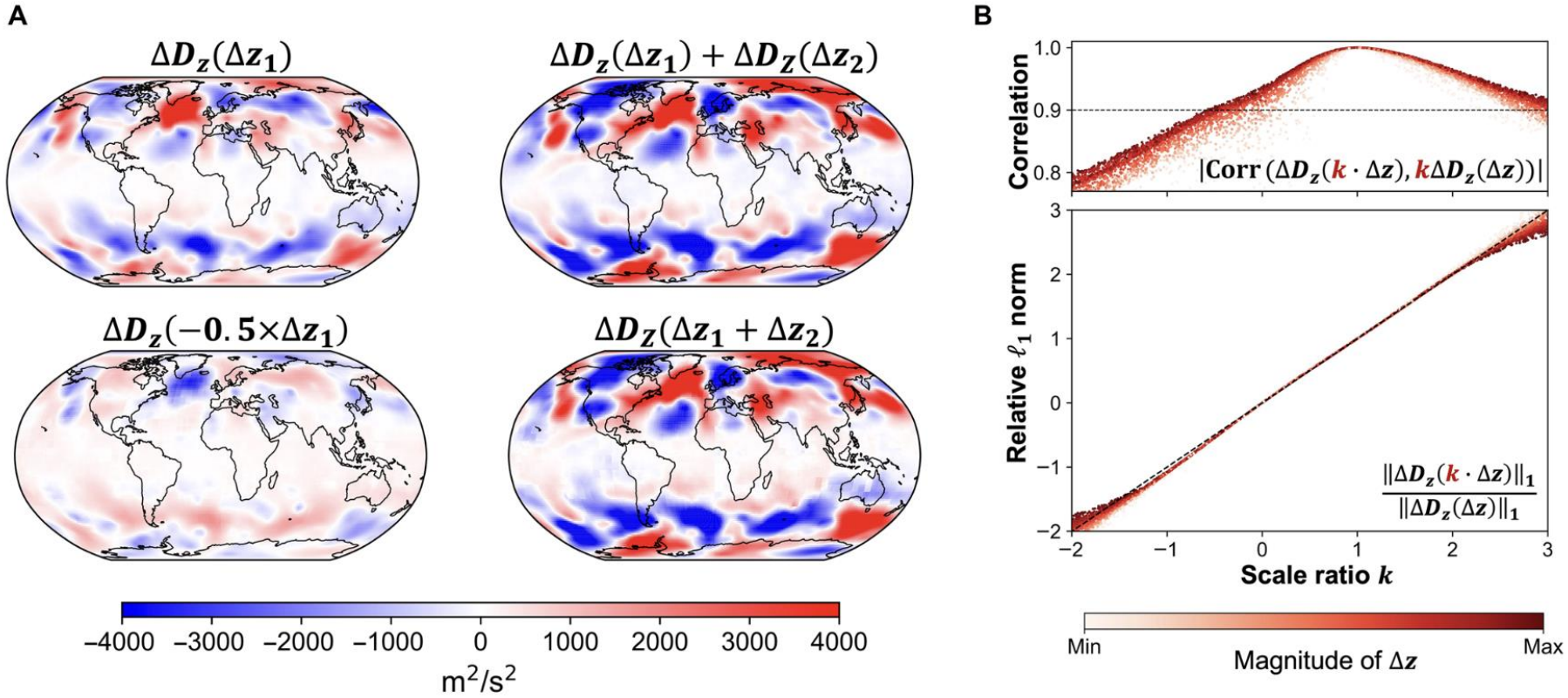
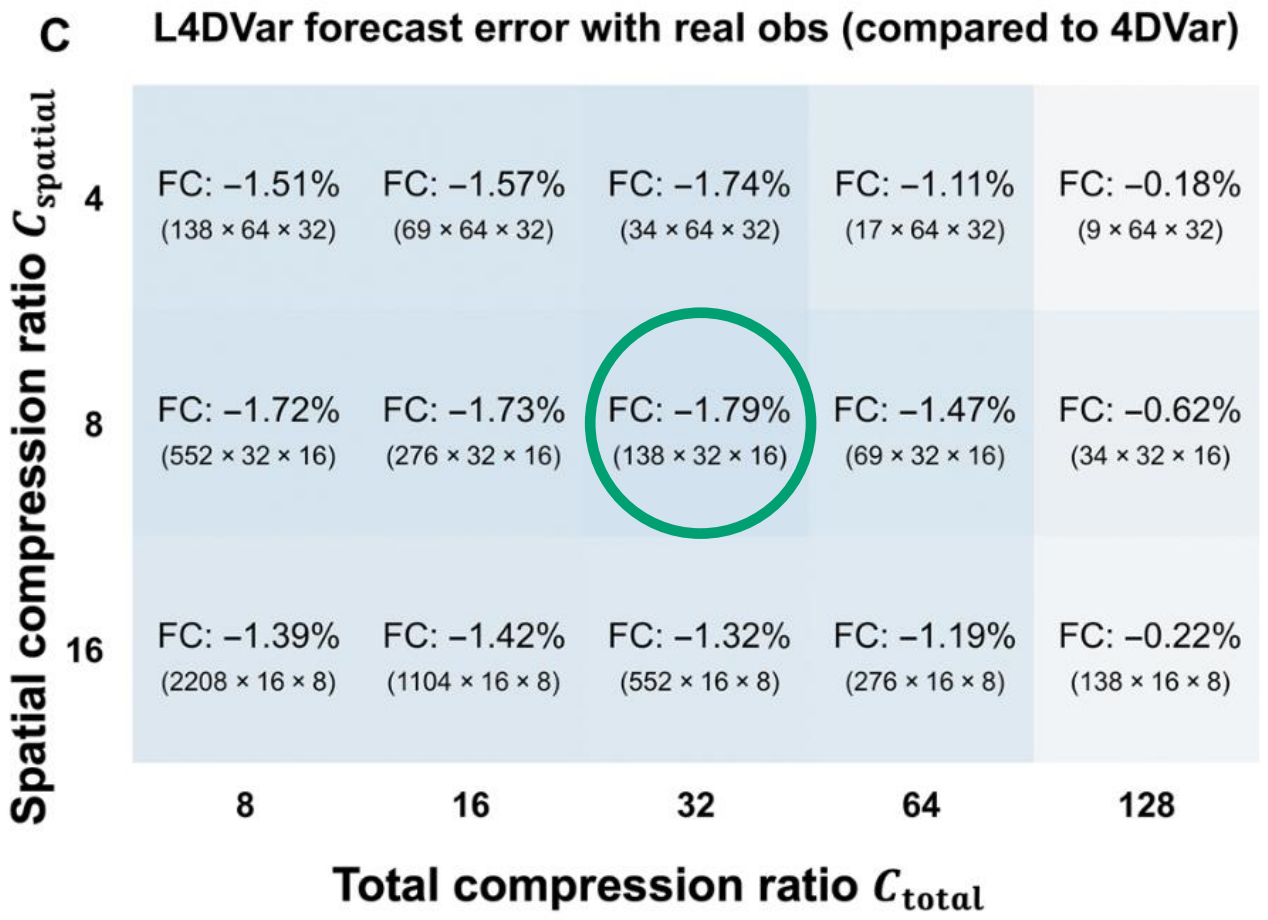


Fig. 6. Approximately affine behavior of the AE decoder along latent directions representative of atmospheric variability. (**A**) The impact of latent-space perturbations $\Delta\boldsymbol{z}$ at $\boldsymbol{z}$ on decoding results, denoted as $\Delta D_z(\Delta\boldsymbol{z})$, is shown using $Z500$. Here, z denotes the latent state corresponding to the ERA5 reanalysis at 0000 UTC on 1 February 2017. The perturbations $\Delta\boldsymbol{z}_1$ and $\Delta\boldsymbol{z}_2$ represent the latent differences between z and the reanalysis at 0000 UTC on 1 January and 1 March 2017, respectively. (**B**) Evaluation of the near-linear response region of the AE decoder. The top panel shows the correlation between $\Delta D_z(k\cdot\Delta\boldsymbol{z})$ and $k\cdot\Delta D_z(\Delta\boldsymbol{z})$ as a function of scale ratio k; the bottom panel shows their relative ℓ_1 norm.

Decoder remains linear across a wide range of scaling factors — can trust as a replacement for model-space DA

Optimal Latent Dimension Size



With more compression, trade-off between greater reconstruction error and a more diagonal **B**

Caveats

- Does not account for autoencoder reconstruction error
- Assumes Gaussian background errors in latent space
- No uncertainty estimation
- May miss extreme events

Why LDA works? lol

Although LDA achieves promising results in both analysis and forecast accuracy, its underlying mechanism remains unclear, raising potential questions about its overall reliability. A central challenge

Caveats

- Does not account for autoencoder reconstruction error
- Assumes Gaussian background errors in latent space
- No uncertainty estimation
- May miss extreme events

I am surprised this doesn't happen sooner.
Perhaps figuring out the physics constraints is the primary bottleneck.

In control theory, developing controllers in a reduced order model (ROM) is pretty common (equivalent to assimilation in latent space). The Linear-Quadratic Regulator (LQR) has a setup similar to 4DVar, but replaces the state with the latent-space state.

$$\min_{u(t)} \quad J = \mathbf{x}^{\mathsf{T}}(t_1)F(t_1)\mathbf{x}(t_1) + \int_{t_0}^{t_1} \left(\mathbf{x}^{\mathsf{T}}Q\mathbf{x} + \mathbf{u}^{\mathsf{T}}R\mathbf{u} + 2\mathbf{x}^{\mathsf{T}}N\mathbf{u}\right)\, dt,$$

e.g. replace x as y, where $y = P^{T}x$, P is the map (linear or non-linear) from regular states to latent states.

Maybe we should be aware and get some inspiration from other fields.

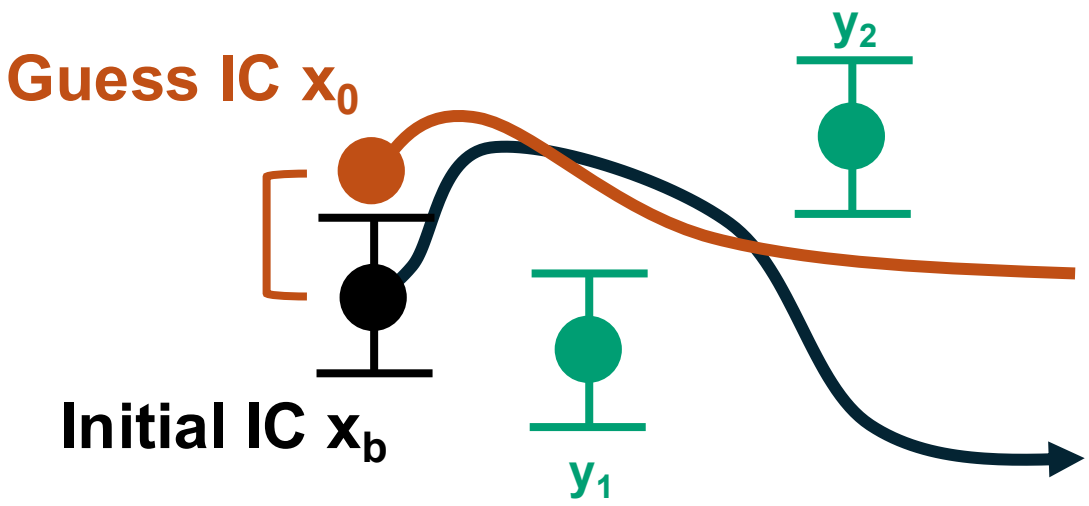
Backup Slides

4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

Model Error Term

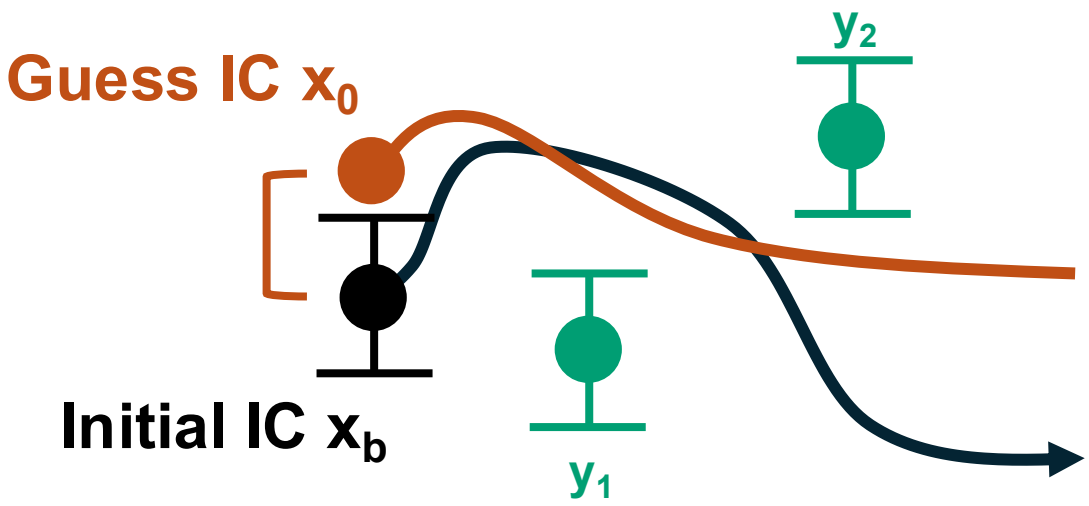


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

Departure from initial guess

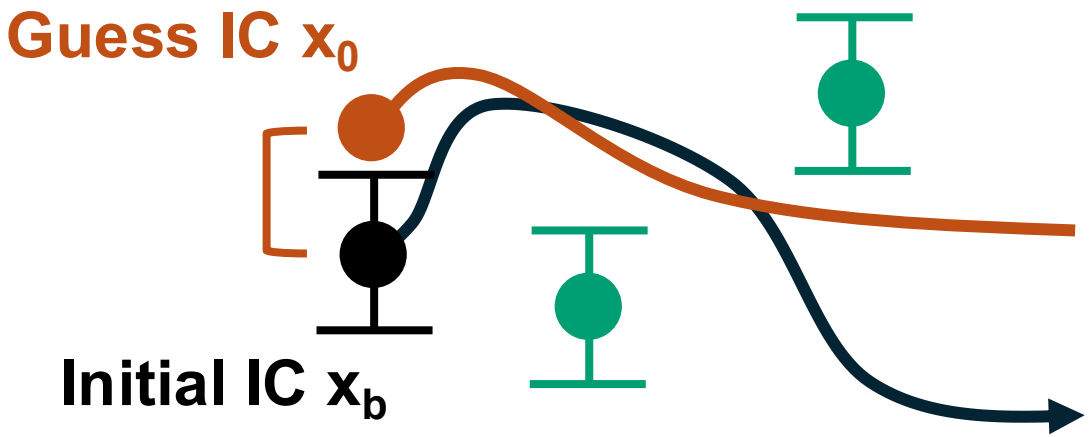


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

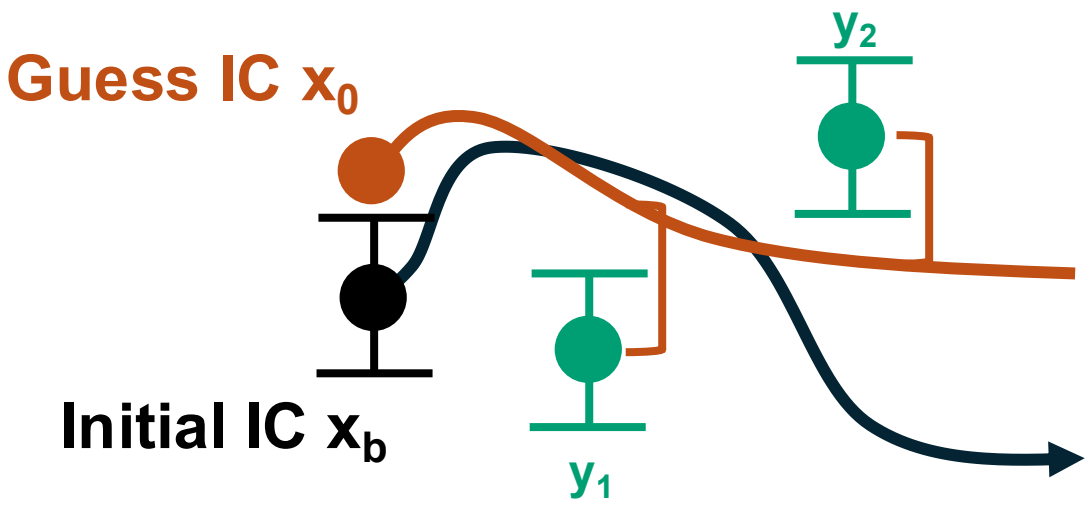
Model Error Covariance Matrix (spreads model departures between variables and across time and space)



4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \underbrace{\frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]}_{\text{Observation Error Term}}$$

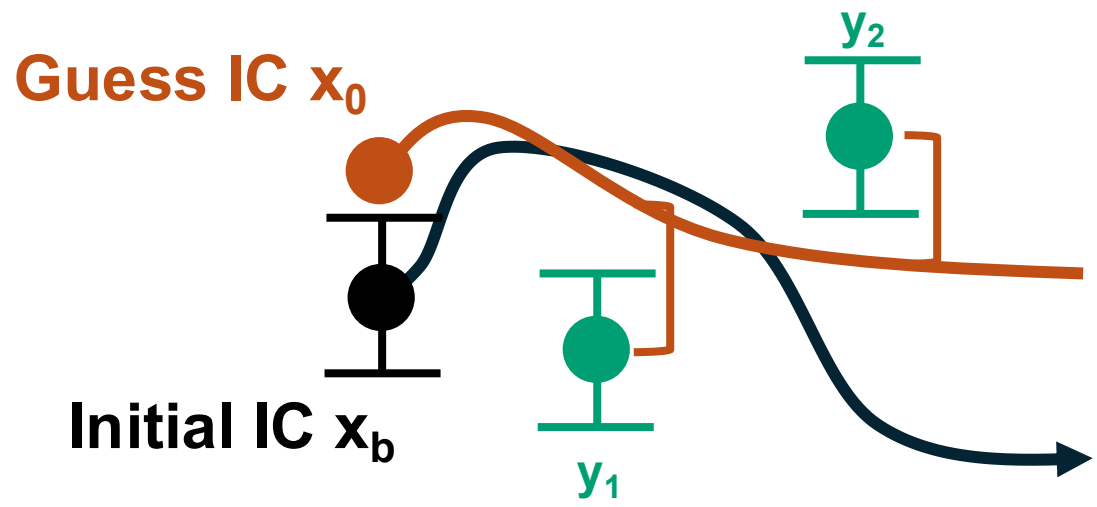


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

Departure from observation

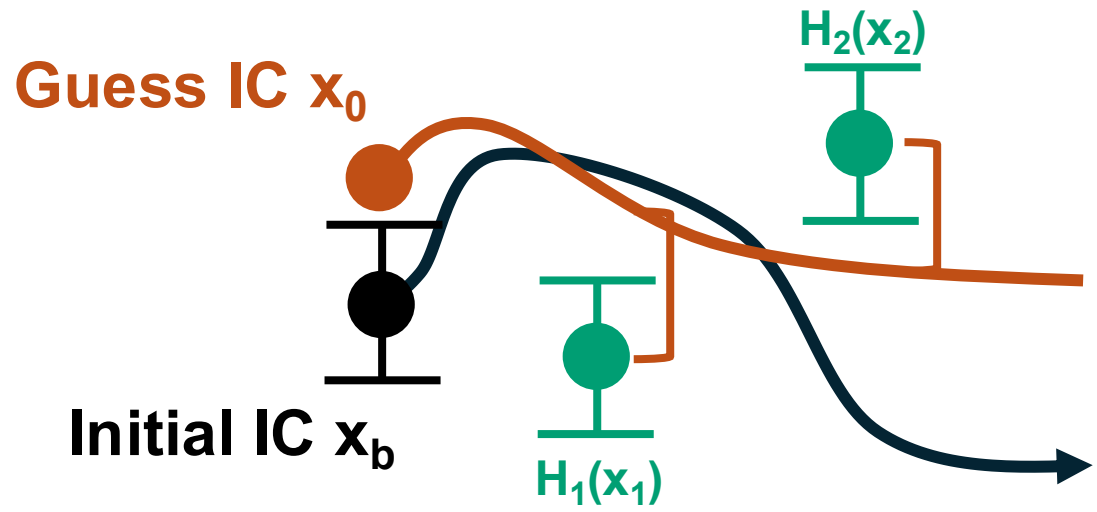


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

"Observation Operator" (interpolate + transform model state to observation space)

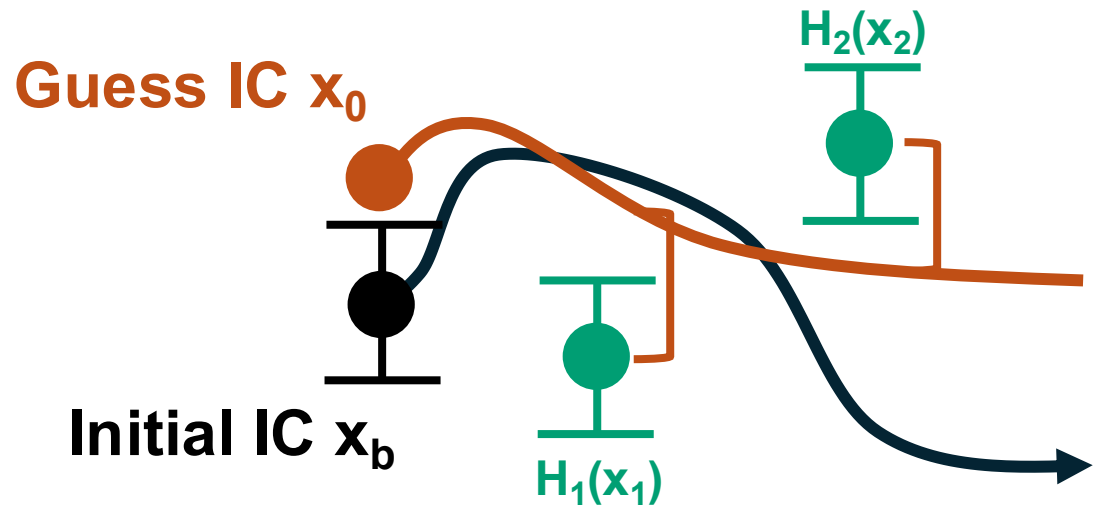


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

Observation Error Covariance Matrix (spreads observation departures between variables and across time and space)

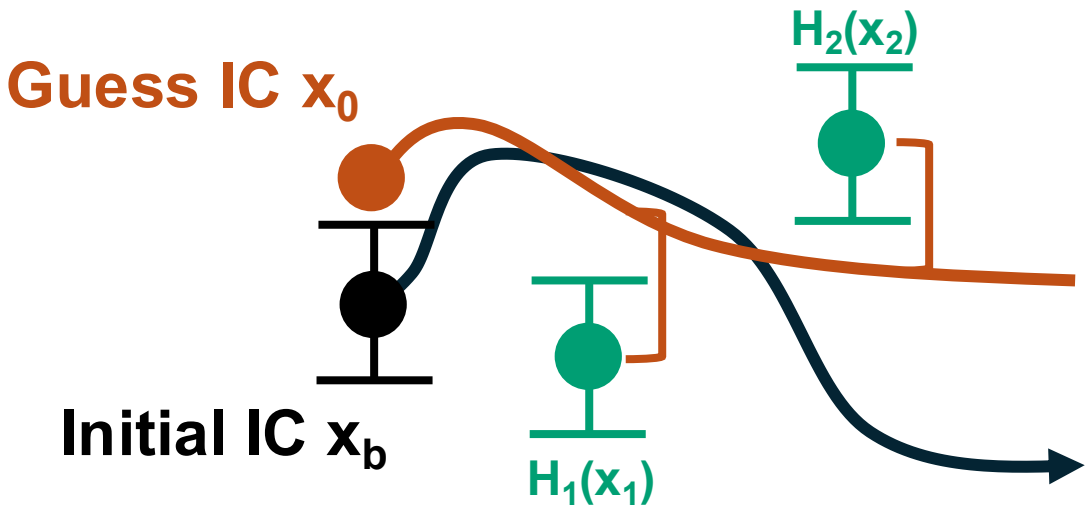


4D-Var Cost Function

Goal: Determine the model state x_a which minimizes cost function J

$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

Sum of observation errors across assimilation window



4D-Var Algorithm

Minimize $$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

→ **Goal: Find x_0 for which $dJ/dx_0 \sim 0$**

$$\nabla J_{x_0} = -B_0^{-1}(x_0 - x_b) - \sum_{i=0}^{n} M_{dt}^T \ldots M_{t-dt}^T M_t^T H_i^T(x_i) R_i^{-1}(y(i) - H_i(x_i))$$

Adjoint: Backwards gradients of model state at time *t* w.r.t model state at time *t-dt*

Why is 4D-Var so dang expensive?

Minimize $$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

→ **Goal: Find x_0 for which $dJ/dx_0 \sim 0$**

$$\nabla J_{x_0} = -B_0^{-1}(x_0 - x_b) - \sum_{i=0}^{n} M_{dt}^T \ldots M_{t-dt}^T M_t^T H_i^T(x_i) R_i^{-1}(y(i) - H_i(x_i))$$

🥵 **B_0** is huge: If $x_0 \sim 10^6$, $B_0 \sim 10^{12}$

😫 **H** must be computed for each new observational product

😵 🛸 **M^T** must be derived for each numerical model

Why is 4D-Var so dang expensive?

Minimize $$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$

→ **Goal: Find x_0 for which $dJ/dx_0 \sim 0$**

$$\nabla J_{x_0} = -B_0^{-1}(x_0 - x_b) - \sum_{i=0}^{n} M_{dt}^T \ldots M_{t-dt}^T M_t^T H_i^T(x_i) R_i^{-1}(y(i) - H_i(x_i))$$

- **B_0** is huge: If $x_0 \sim 10^6$, $B_0 \sim 10^{12}$
- **H** must be constructed for each new observational product
- **M^T** must be derived for each numerical model
- ~100 iterations are needed to optimize x_0 for each assimilation step