Efficient high-dimensional variational data assimilation with machine-learned reduced-order models

Eliot Kim

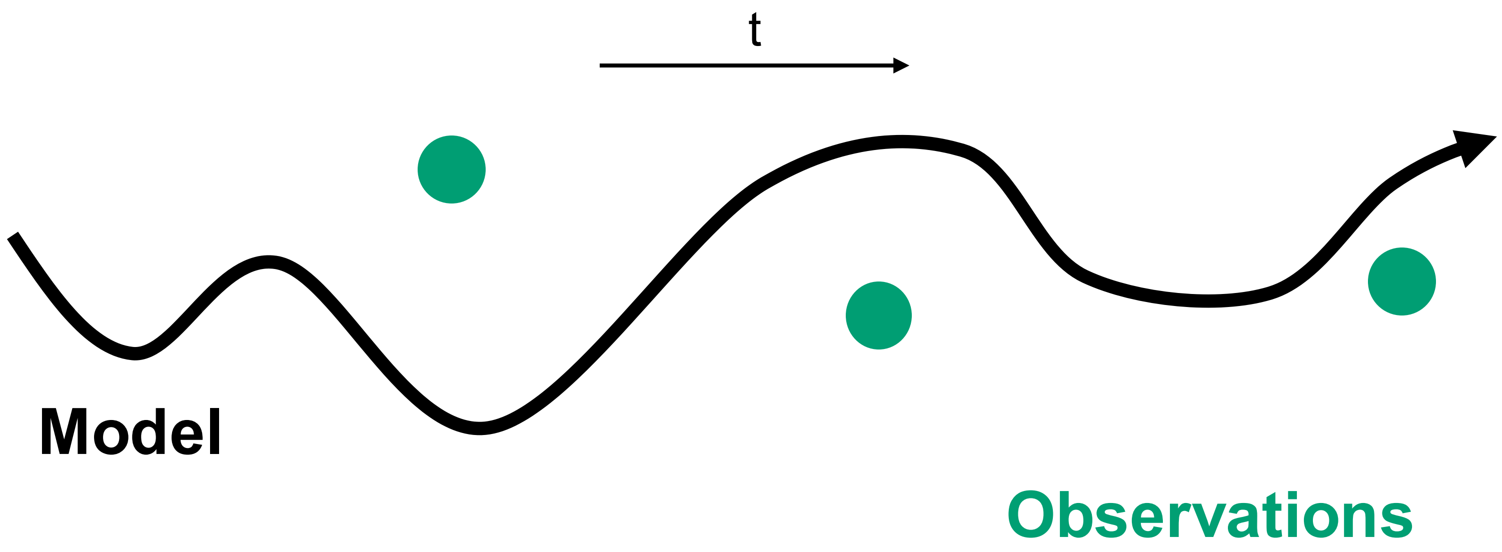
Machine Learning Journal Club

Presenting Maulik et al. 2022

December 5th, 2025

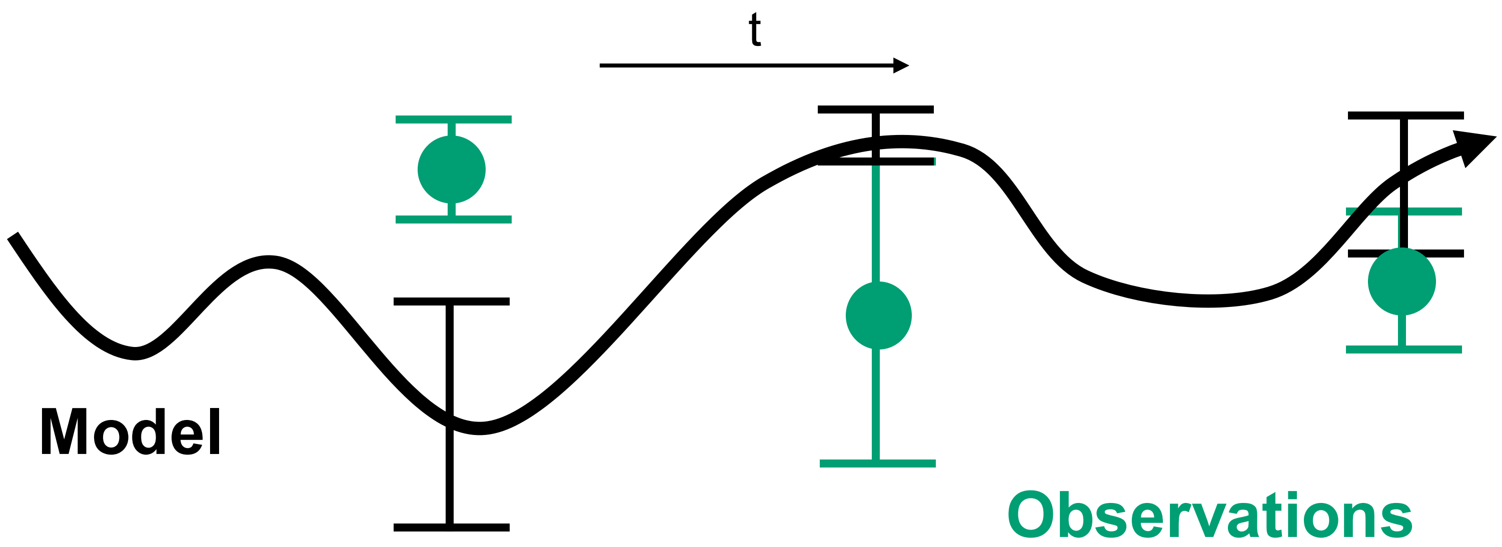
**** Fill out the when2meet for Winter Quarter :] ****

What is "data assimilation"?



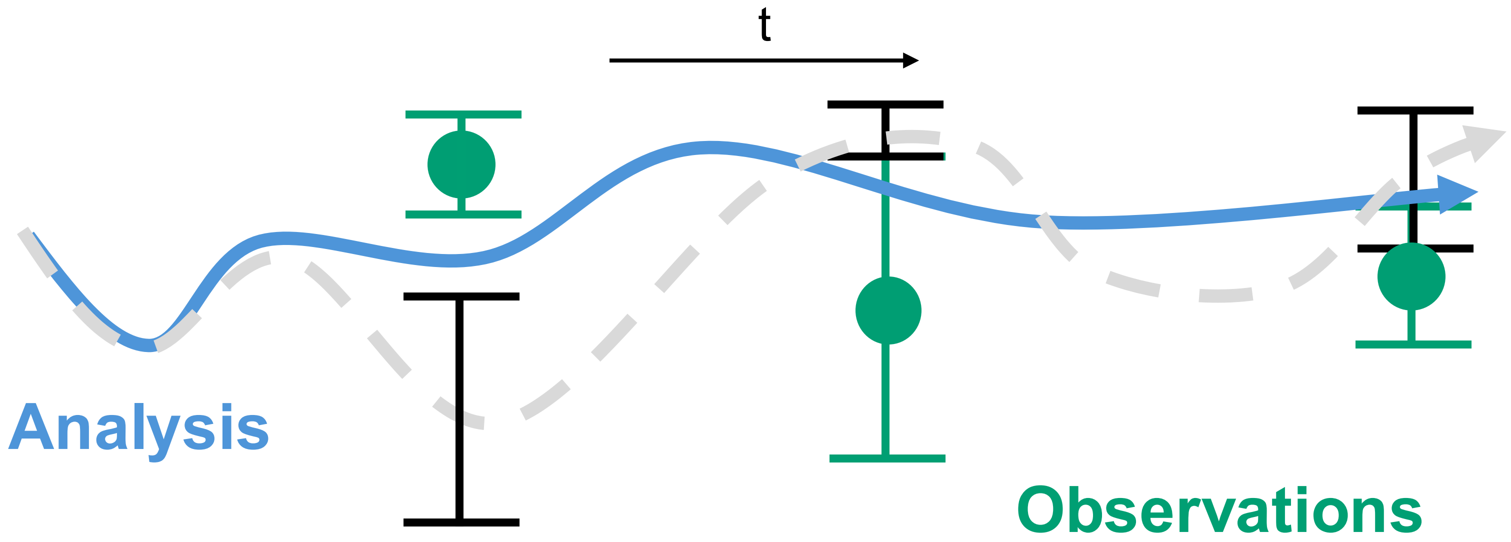
Given **model estimates** and **observations**, what is the most likely atmospheric state?

What is “data assimilation”?



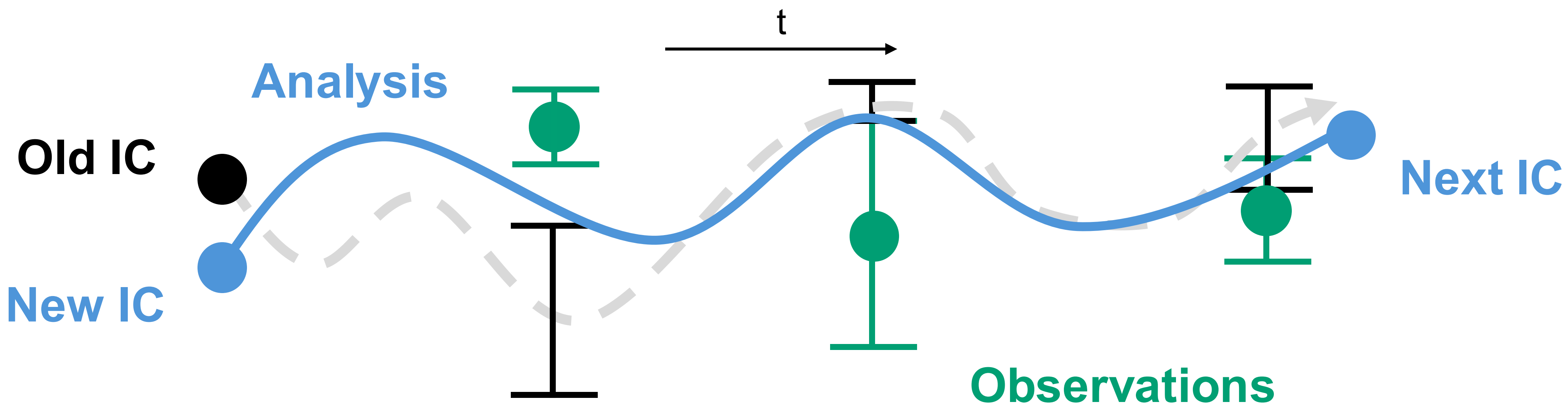
Both model estimates and observations have **errors**

What is “data assimilation”?



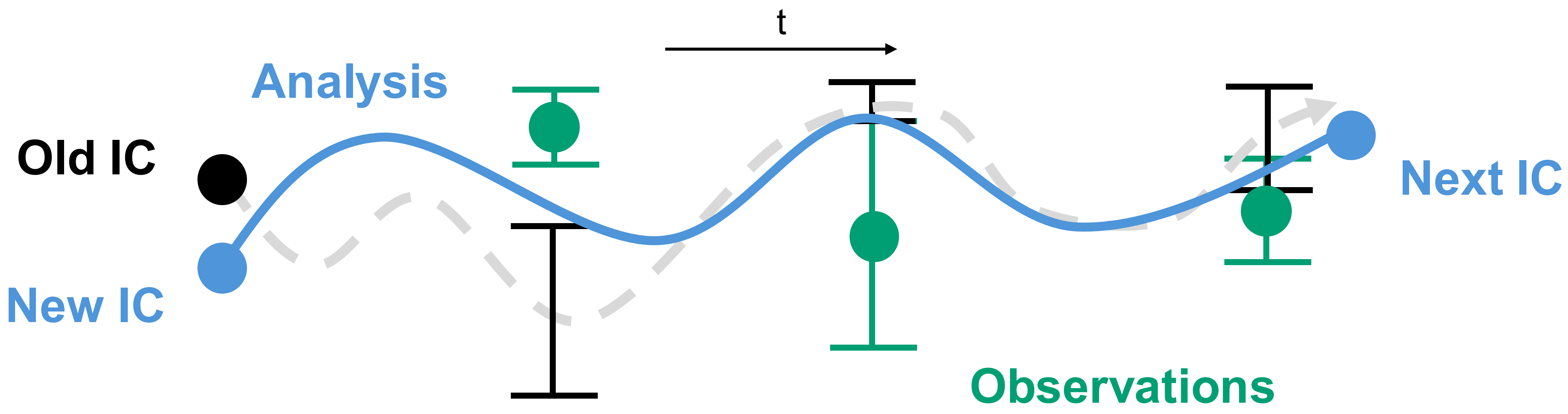
Final **analysis** is determined by error-weighted contributions from observations and model states

Applications of Data Assimilation



- Real-time Forecasting: Optimizing initial conditions
  - Ex: Operational weather forecasts @ ECMWF, Met Office, NCEP, etc.

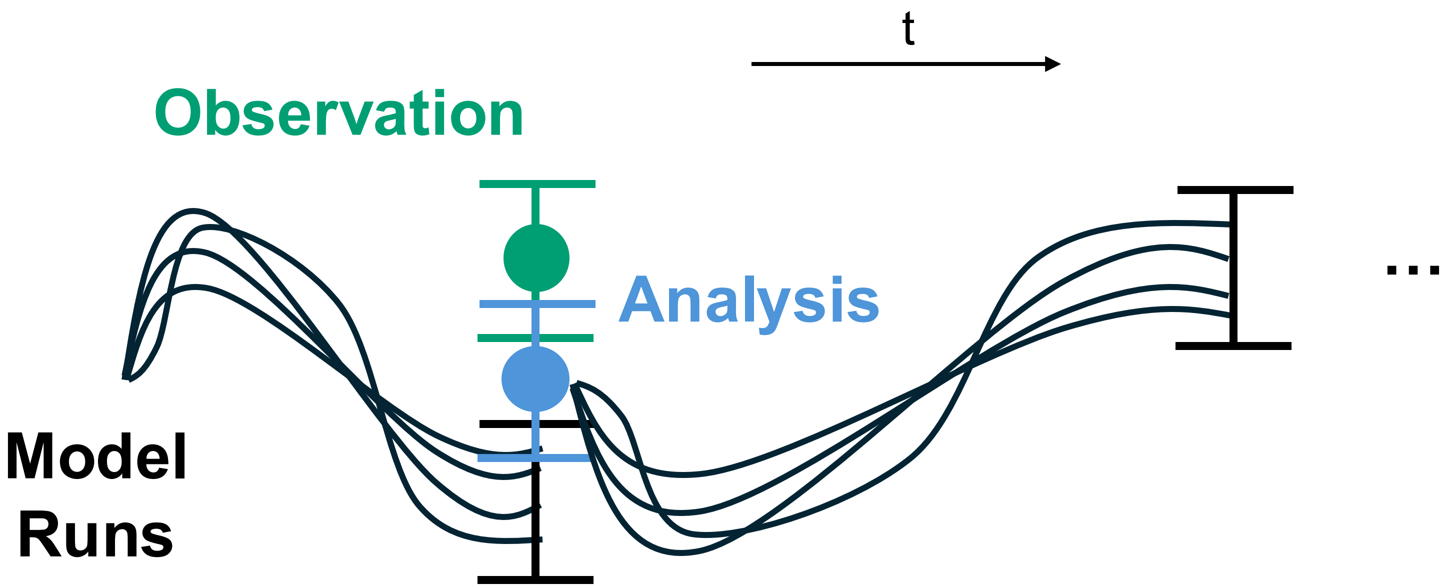
Applications of Data Assimilation



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

DA State-of-the-art

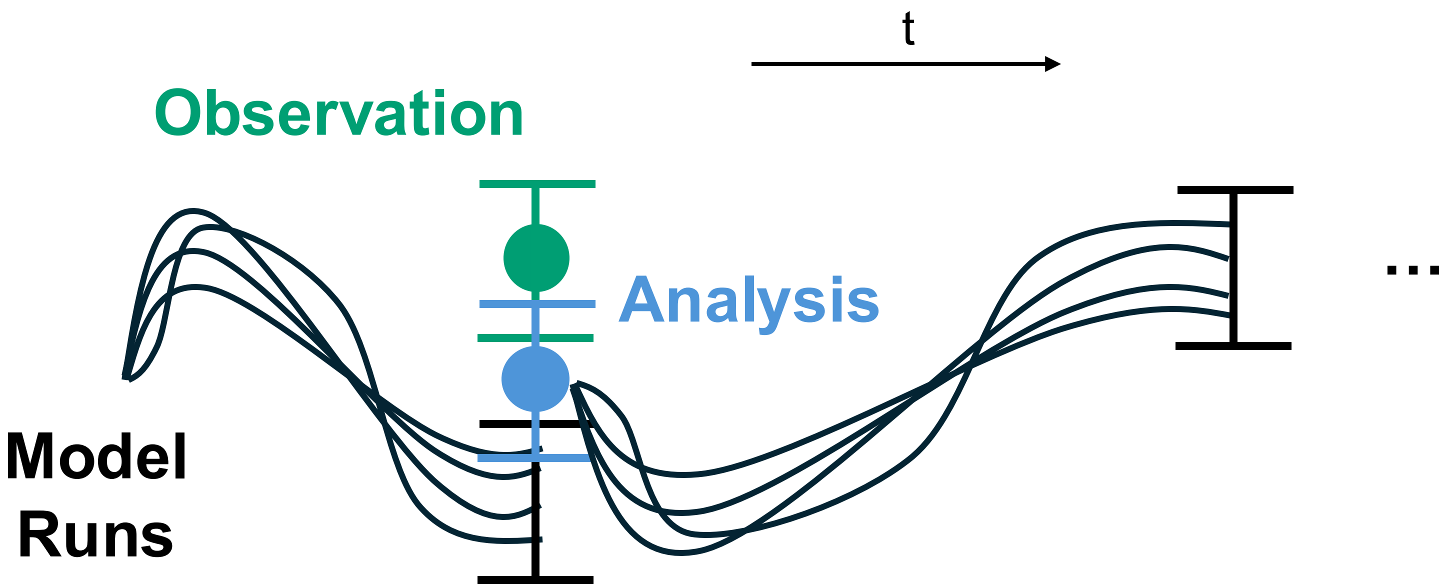
Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step



DA State-of-the-art

Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step

+ Model error updated during assimilation
+ Easy to implement
– Requires 10-100s of expensive model runs
– Outputs discrete analysis trajectory



DA State-of-the-art

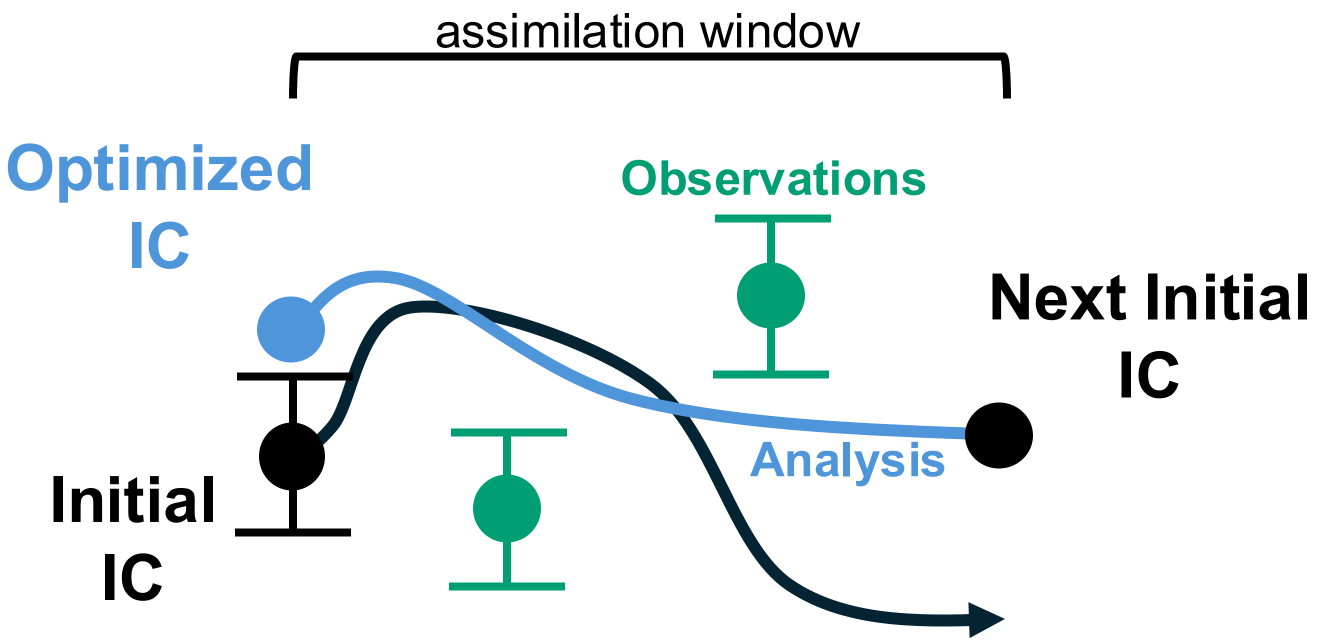
Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step

4D Variational DA (4D-Var)
Determine initial model state which gives best fits to observations in **assim. window***
*3D-Var collapses observations to time of IC

DA State-of-the-art

Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step

4D Variational DA (4D-Var)
Determine initial model state which gives best fits to observations in **assim. window***
*3D-Var collapses observations to time of IC



DA State-of-the-art

Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step

4D Variational DA (4D-Var)
Determine initial model state which gives best fits to observations in assim. window

+ Outputs continuous analysis trajectory which obeys model physics
– Requires backwards model gradients during optimization (expensive!)

DA State-of-the-art

Ensemble Kalman Filters (EnKF)
Run a large ensemble of models to compute model error at each time-step

4D Variational DA (4D-Var)
Determine initial model state which gives best fits to observations in assim. window

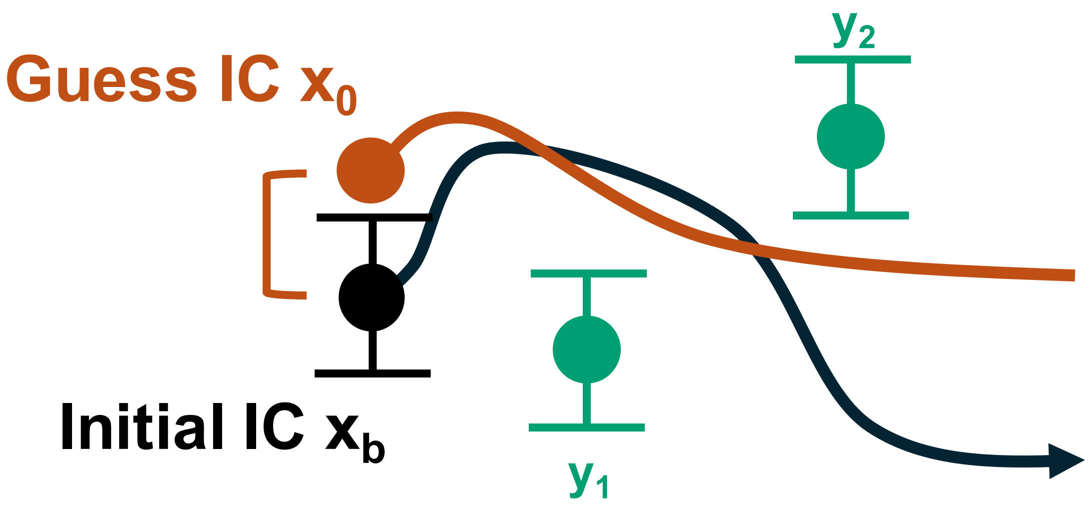
+ Outputs continuous analysis trajectory which obeys model physics
– Requires backwards model gradients during optimization (expensive!)

**** 4D-Var is SOTA, used by ECMWF for IFS and ERA5! ****

4D-Var Cost Function

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

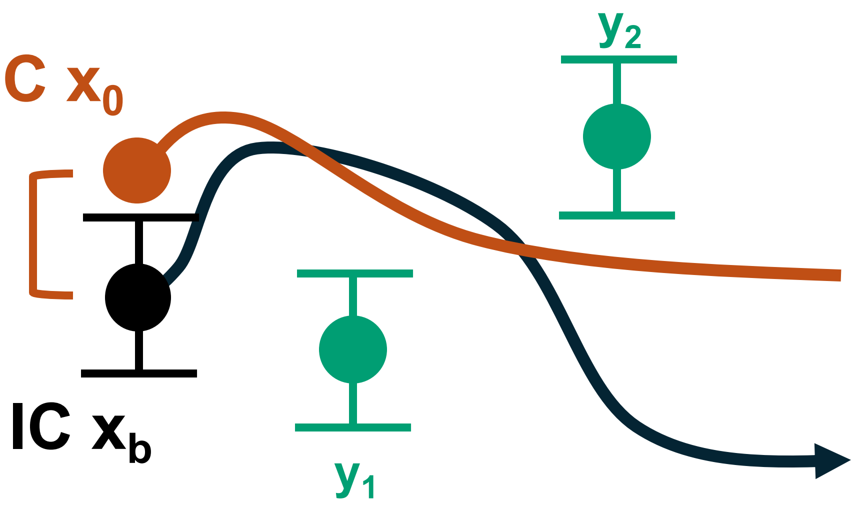
$$J(x_0) = \underbrace{\frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b)}_{\text{Model Error Term}} + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$



4D-Var Cost Function

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

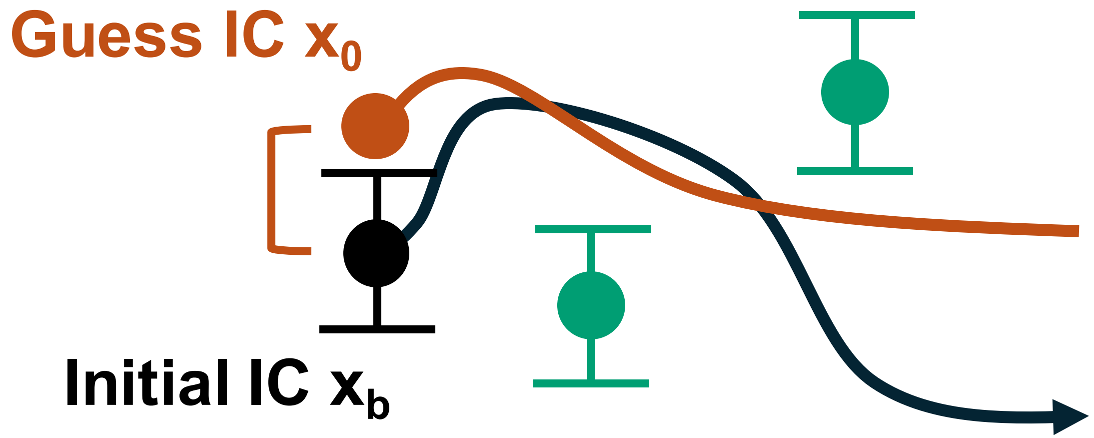
$$J(x_0) = \frac{1}{2}\underbrace{(x_0 - x_b)}_{\text{Departure from initial guess}}{}^T B_0^{-1} \underbrace{(x_0 - x_b)} + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$



4D-Var Cost Function

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

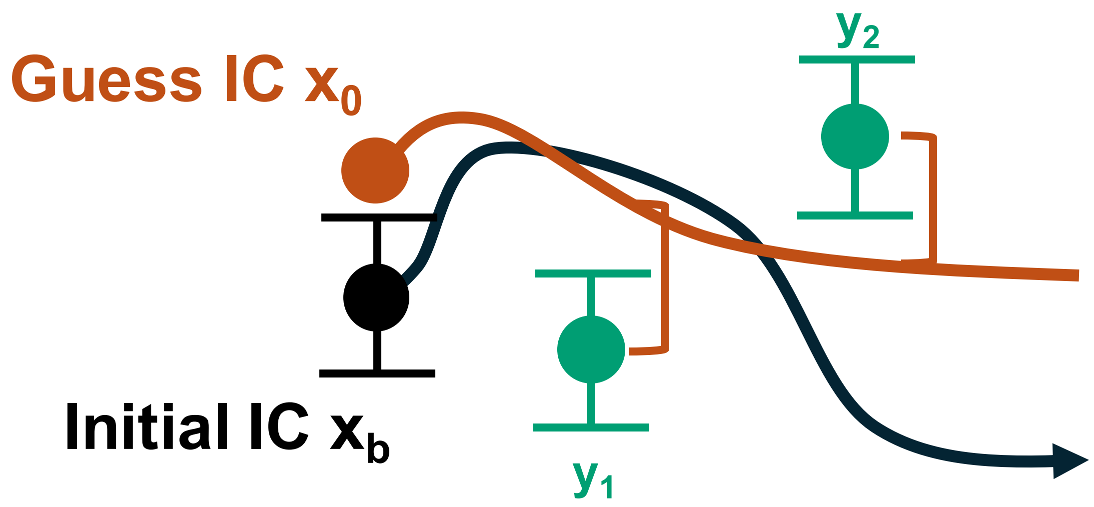
$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T \underbrace{B_0^{-1}}_{\text{Model Error Covariance Matrix (spreads model departures between variables and across time and space)}} (x_0 - x_b) + \frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]$$



4D-Var Cost Function

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

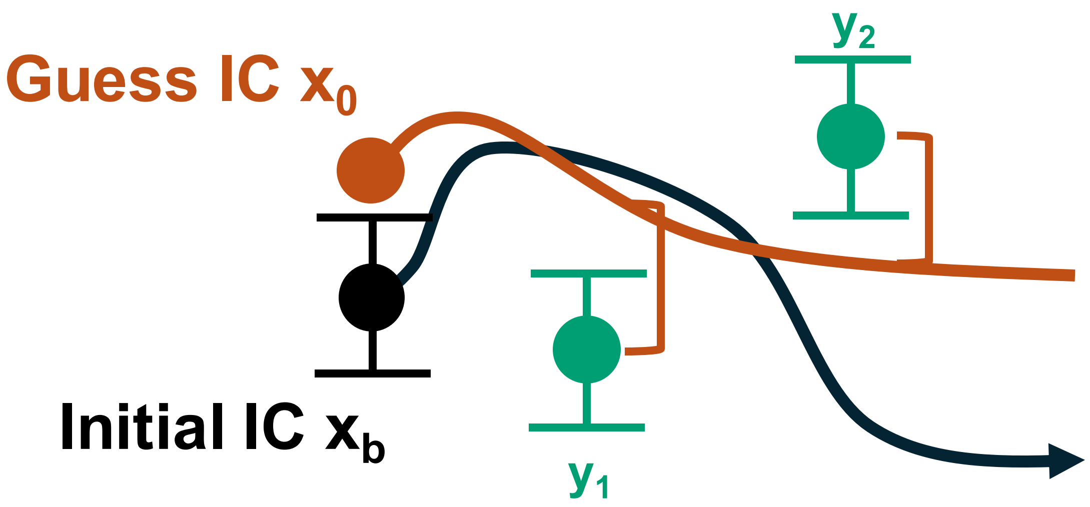
$$J(x_0) = \frac{1}{2}(x_0 - x_b)^T B_0^{-1}(x_0 - x_b) + \underbrace{\frac{1}{2}\sum_{i=0}^{n}[H_i(x_i) - y_i^o]^T R_i^{-1}[H_i(x_i) - y_i^o]}_{\text{Observation Error Term}}$$



4D-Var Cost Function

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

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

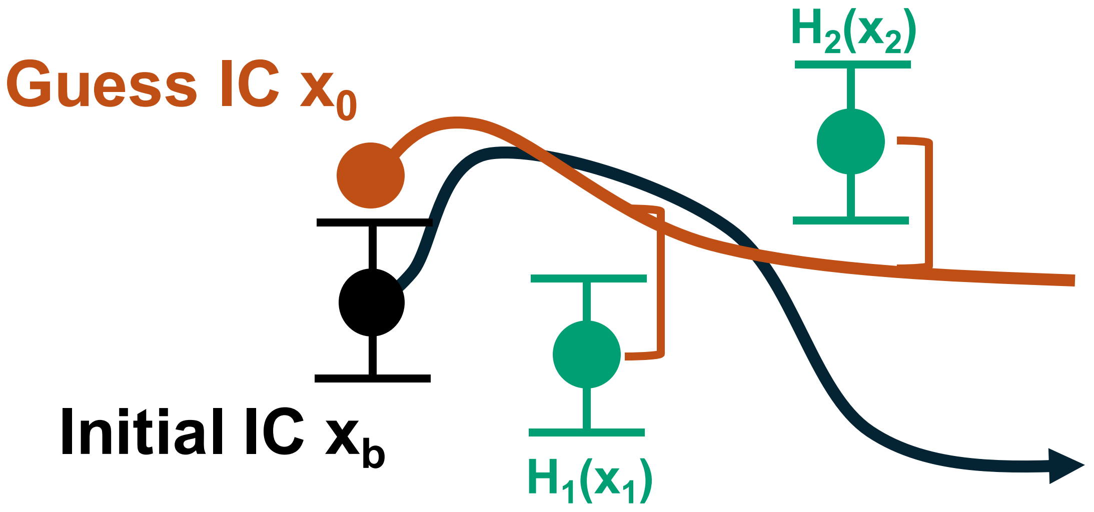


4D-Var Cost Function

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

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

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

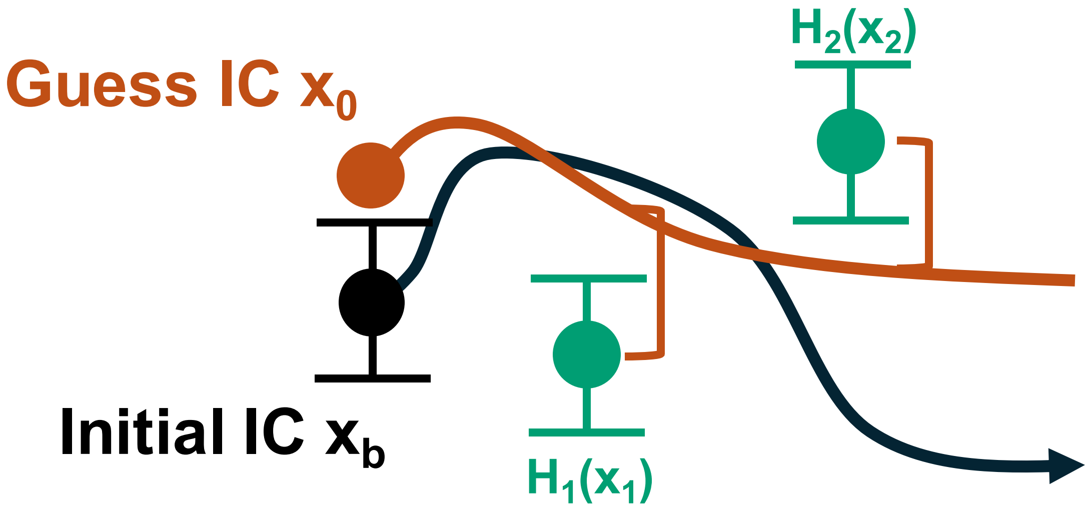


4D-Var Cost Function

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

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

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

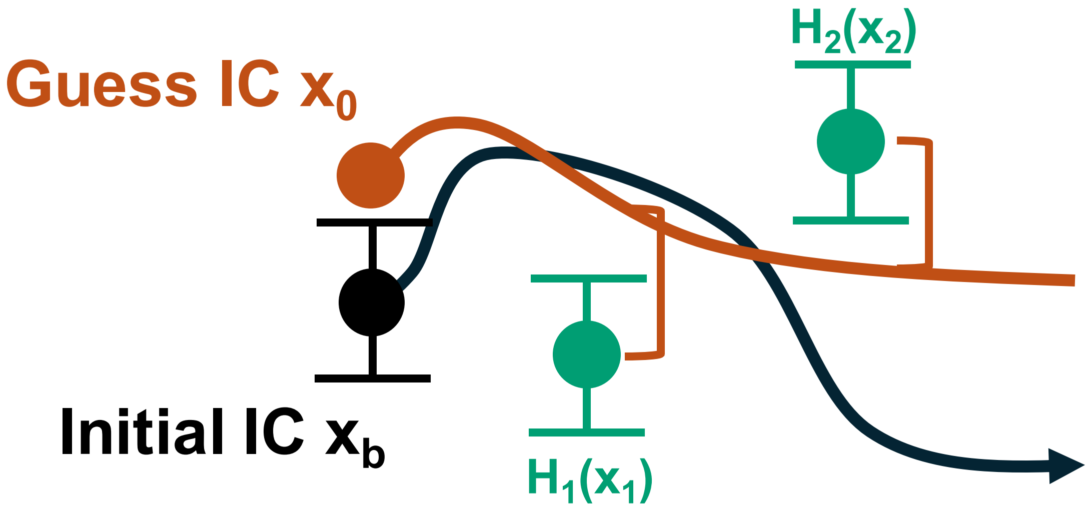


4D-Var Cost Function

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

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

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

Maulik et al. 2022

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

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

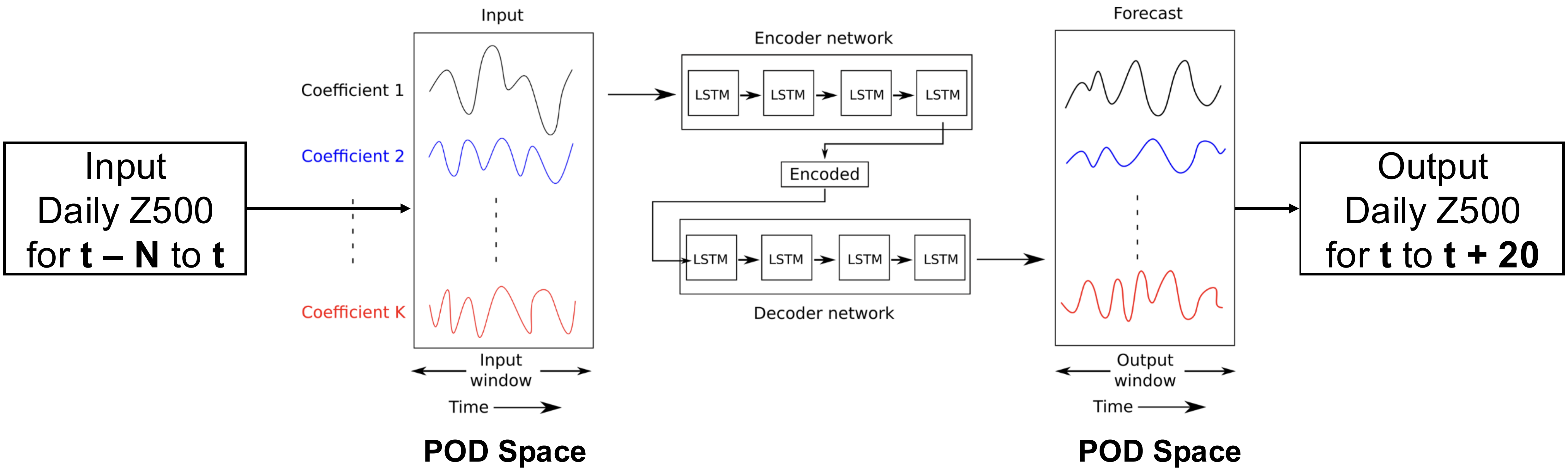
Goal: Reduce burden of $\mathbf{M}^{\mathbf{T}}$ and **B** using fast emulator and dimensionality reduction

→First to conduct “on-the-fly” 4D-Var with an emulator + auto-differentiated adjoint

Experimental Set-Up

- Training Data
  - WRF simulations driven by NCEP-2 Re-analyses
  - Target: Daily Z500 for 20 days lead time **(long!!)**
  - Input Window: 1-4 weeks
- Domain
  - Train: 1984-1989; Test: 1991
  - 60 by 60 km grid cells over North America (102 by 119)
- “Observations”
  - Assumptions: uncorrelated observation errors, observation error standard deviation ~1.5% of mean
  - 5000 grid cells randomly sampled from true state as “sites”
    Q: ~41% of total grid cells in each sample — is this realistic for operational DA?
- Baselines
  - Climatology
  - Persistence

Emulator Set-Up



1) POD reduces size of B

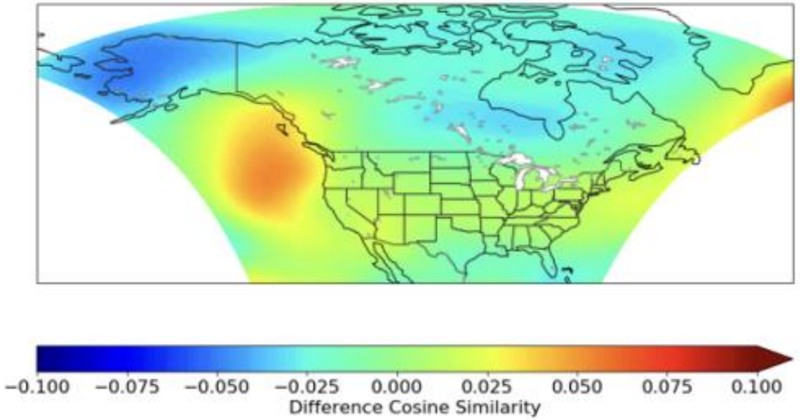
2) LSTM is auto-differentiable → Get the adjoint for free!

- Gradients w.r.t input are computed during back-propagation

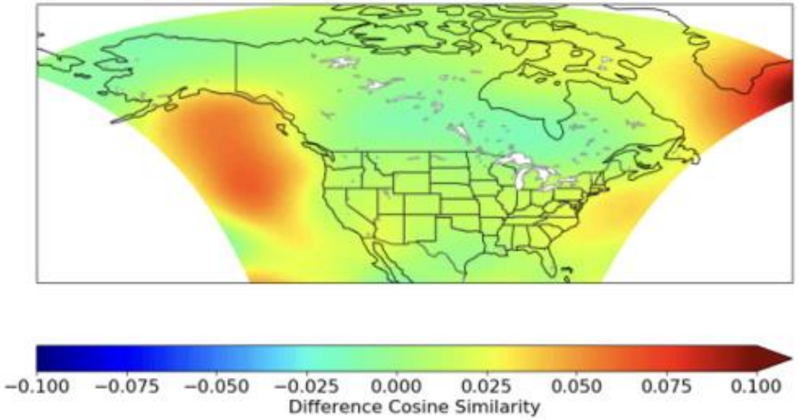
Results

Emulator-Only

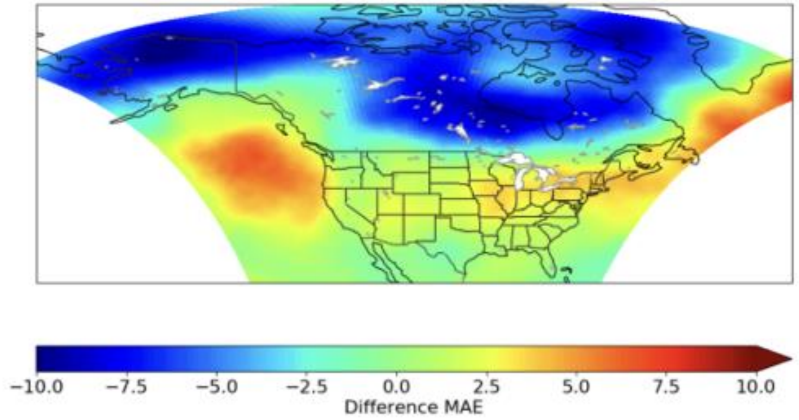
Emulator + DA



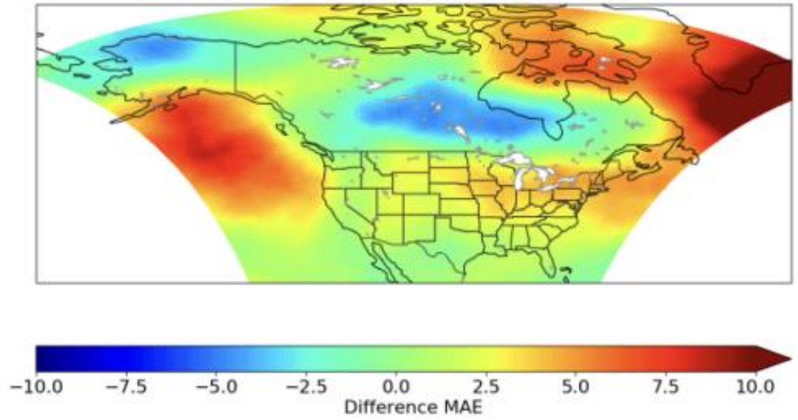
(a) Similarity improvement: Regular emulator



(b) Similarity improvement: Regular emulator + DA

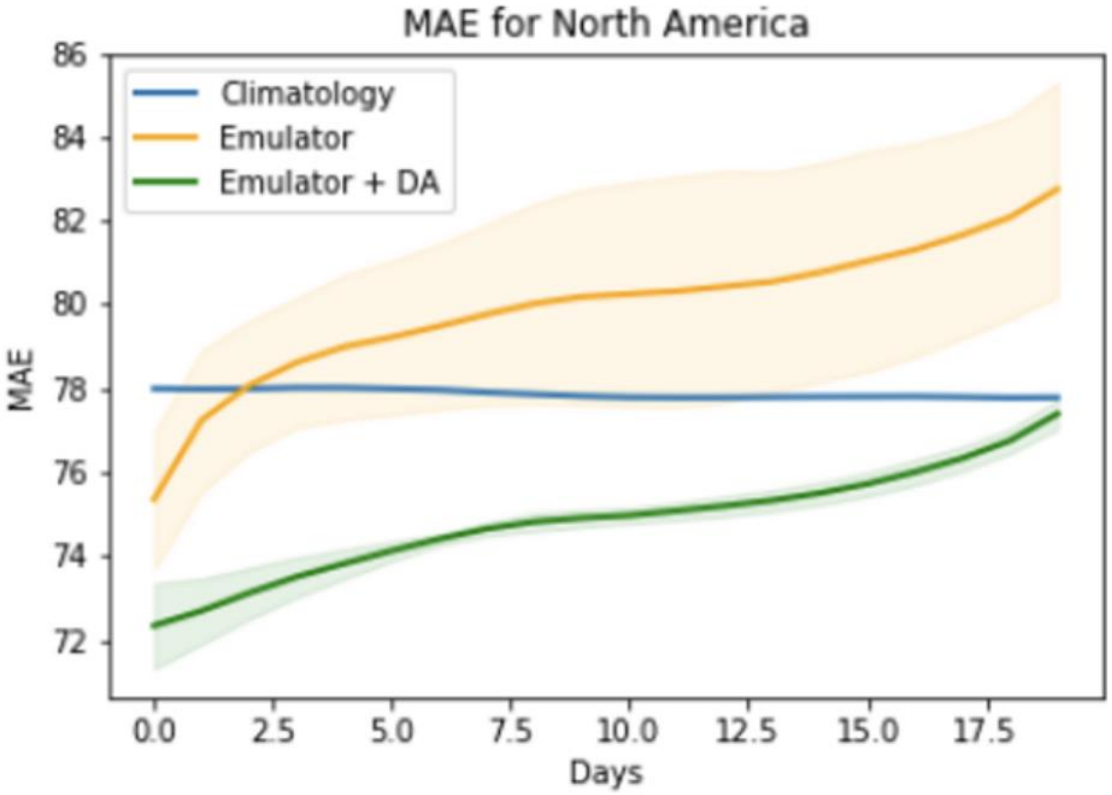


(c) MAE improvement: Regular emulator



(d) MAE improvement: Regular emulator + DA

Results



(i) North America

Speed-Up

Emulator + DA: ~1 hour

Numerical Forecast: ~172 hours

Numerical Forecast + DA: 172 x (~100) hours + adjoint derivation labor cost

Discussion in groups!

~~**Q1:** What are your main takeaways from the paper?~~

Q2: What additional experiments / results would you like to see presented in this work?

~~**Q3:** What are potential impacts of this work's findings for DA applications?~~

Q4: What additional steps or considerations might be necessary for emulator-based 4D-Var in an **operational setting**?

Q5: What questions do you have about data assimilation? Is DA relevant for your research?