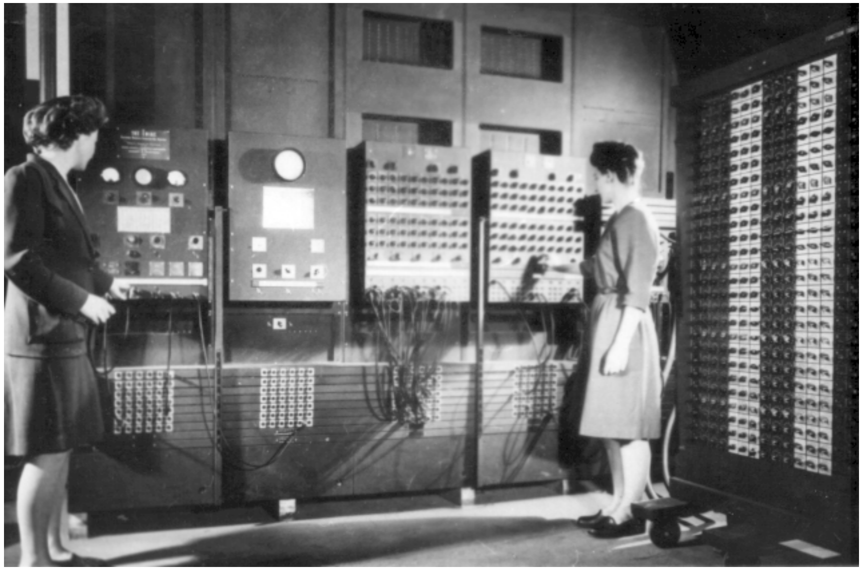
Computational Issues: An EnKF Perspective

Jeff Whitaker

NOAA Earth System Research Lab

ENIAC 1948

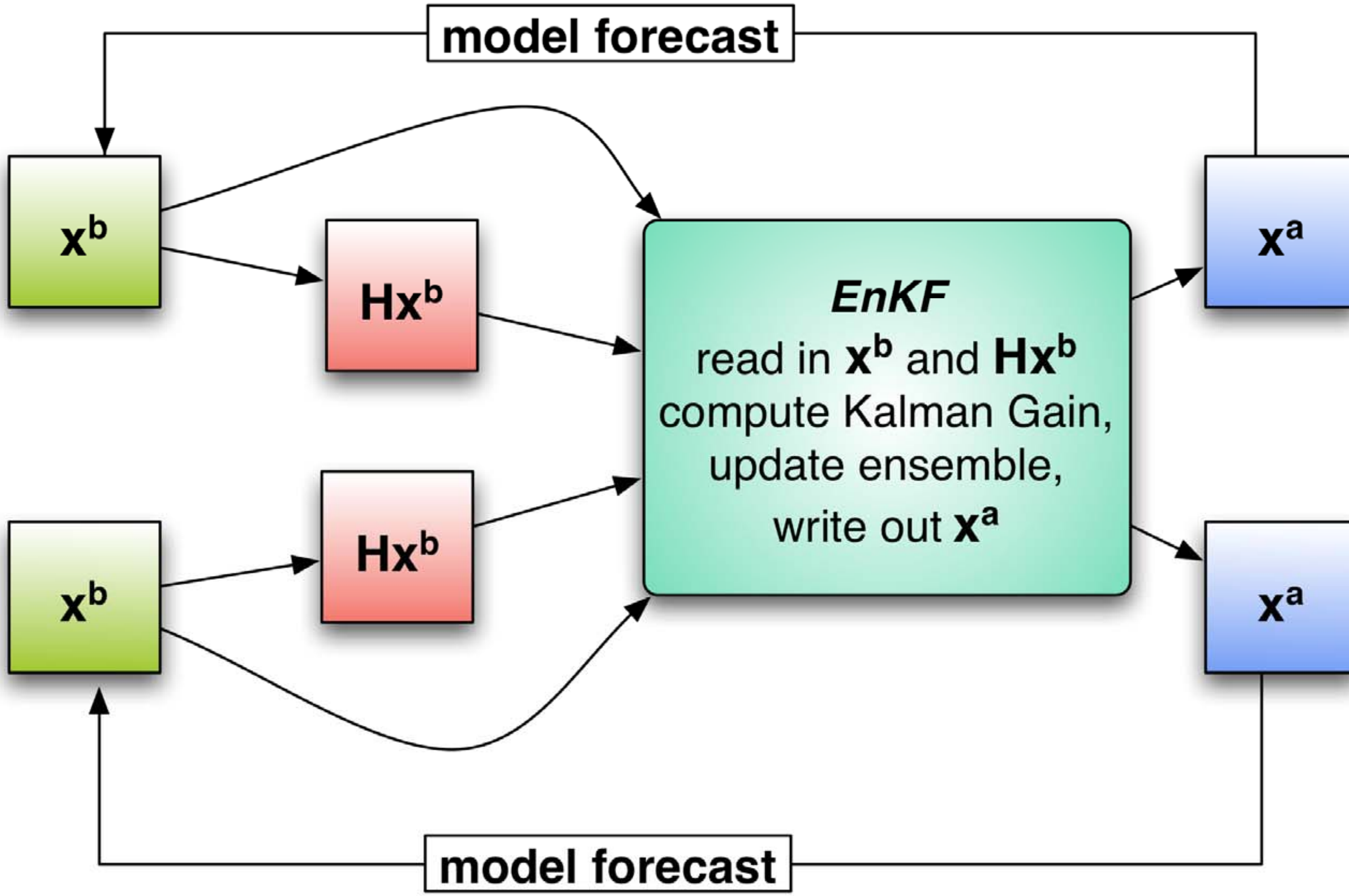


"Roadrunner" 2008

EnKF cycle

1) Run ensemble forecast for each ensemble member to get $\mathbf{x}^b$ for next analysis time.
2) Compute $\mathbf{Hx}^b$ for each ensemble member.
3) Given $\mathbf{Hx}^b$, $\mathbf{x}^b$ compute analysis increment (using LETKF, EnSRF etc)

EnKF Cycle (2)


model forecast
x^b
Hx^b
EnKF
read in x^b and Hx^b
compute Kalman Gain,
update ensemble,
write out x^a
x^a
x^b
Hx^b
x^a
model forecast

Step 1: Background Forecast

- **4DVar** - a single run of the (high-res) non-linear forecast model for each outer loop, many runs of (low-res) TLM/adjoint in inner loop in sequence.
- **EnKF** - N simultaneous runs of the non-linear forecast model (embarassingly parallel).
- **Bottom line** - total cost similar, but EnKF may scale better.

Step 2: Forward operator

- **4DVar** - compute full nonlinear $\mathbf{Hx}^b$ in each outer loop. In each inner loop, use linearized **H** (faster, especially for radiances).
- **EnKF** - compute full nonlinear $\mathbf{Hx}^b$ once for each ensemble member simultaneously. Could use linearized **H** for ensemble perturbations.
- **Bottom line** - total cost similar, but EnKF may be scale better.

Step 3: Calculating the increment

- For EnKF, depends on algorithm
  - Perturbed obs EnKF (Env. Canada - obs processed serially in batches) ?
  - Local Ensemble Transform KF (LETKF - developed at U. of Md, being tested at JMA and NOAA) ✓
  - Serial Ensemble Square-Root Filter (EnSRF - NCAR's DART, NOAA ESRL, UW real-time WRF) ✓

Serial EnSRF algorithm

Whitaker and Hamill, 2002: MWR, **130***, 1913-1924*
Anderson, 2003: MWR, **131***, 634-642*

Assume ob errors uncorrelated (**R** diagonal).

Loop over all L obs (m=1,…L). K = Ens. size

1) Update N_{loc} 'nearby' state variables with this observation. Covariance ($\mathbf{P}^b\mathbf{H}^T$) costs O(K* N_{loc})
2) Update L_{loc}-m 'nearby' observation priors (for obs not yet processed) with this observation. Covariance ($\mathbf{HP}^b\mathbf{H}^T$) costs O(K*($L_{loc}$ -m))

Total cost estimate O(K*L*N_{loc}) + O(K*L*L_{loc})
where L_{loc}=av. # of 'nearby' ob priors and
N_{loc}=av. # of 'nearby state elements (for each ob).

EnSRF parallel implementation

*Anderson and Collins, 2007: Journal of Atmospheric and Oceanic Technology A, **24** 1452-1463*

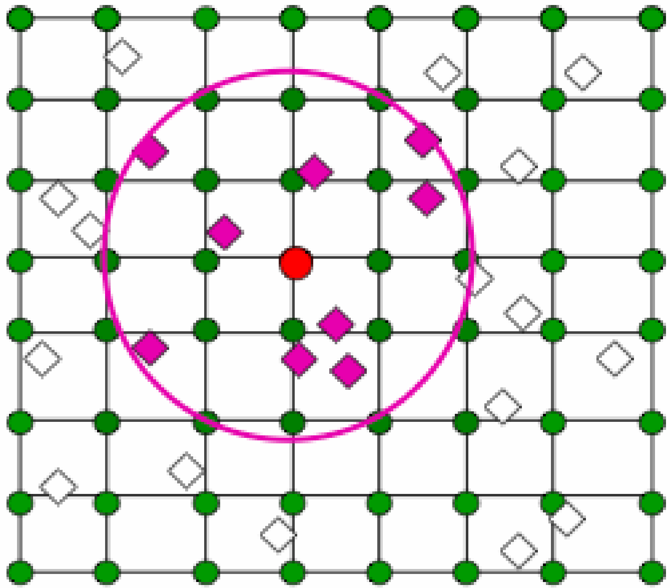
- Update subset of model state and observation priors on each processor.
- Loop over all obs on each processor - get ob priors from processor on which it is updated via MPI_Bcast of K values.

LETKF Algorithm

Perform data assimilation in a local volume, choosing observations

The state estimate is updated at the central grid red dot

All observations (purple diamonds) within the local region are assimilated



Ob error in local volume is increased as a function of distance from red dot, reaching infinity at edge of circle.

LETKF cost estimate

(Szyunogh et al 2008: Tellus, **60A**, *113-130)*

- Each state variable can be updated independently (perfectly parallel, no communication needed). Assume diagonal **R**.
- Most expensive step is $\mathbf{Y}^b\mathbf{R}^{-1}\mathbf{Y}^{bT}$, where **Y** is K x L_{loc} matrix of observation priors. L_{loc} is average number of obs in each local region.
- Cost is $O(K^2*L_{loc}*N)$ vs $O(K*L*N_{loc}) + O(K*L*L_{loc})$ for EnSRF (neglecting communication cost)
  - For L <= N, EnSRF faster
  - For L > K*N, LETKF faster
  - For N~L, LETKF is should be about $O(K*L_{local}/L)$ slower.

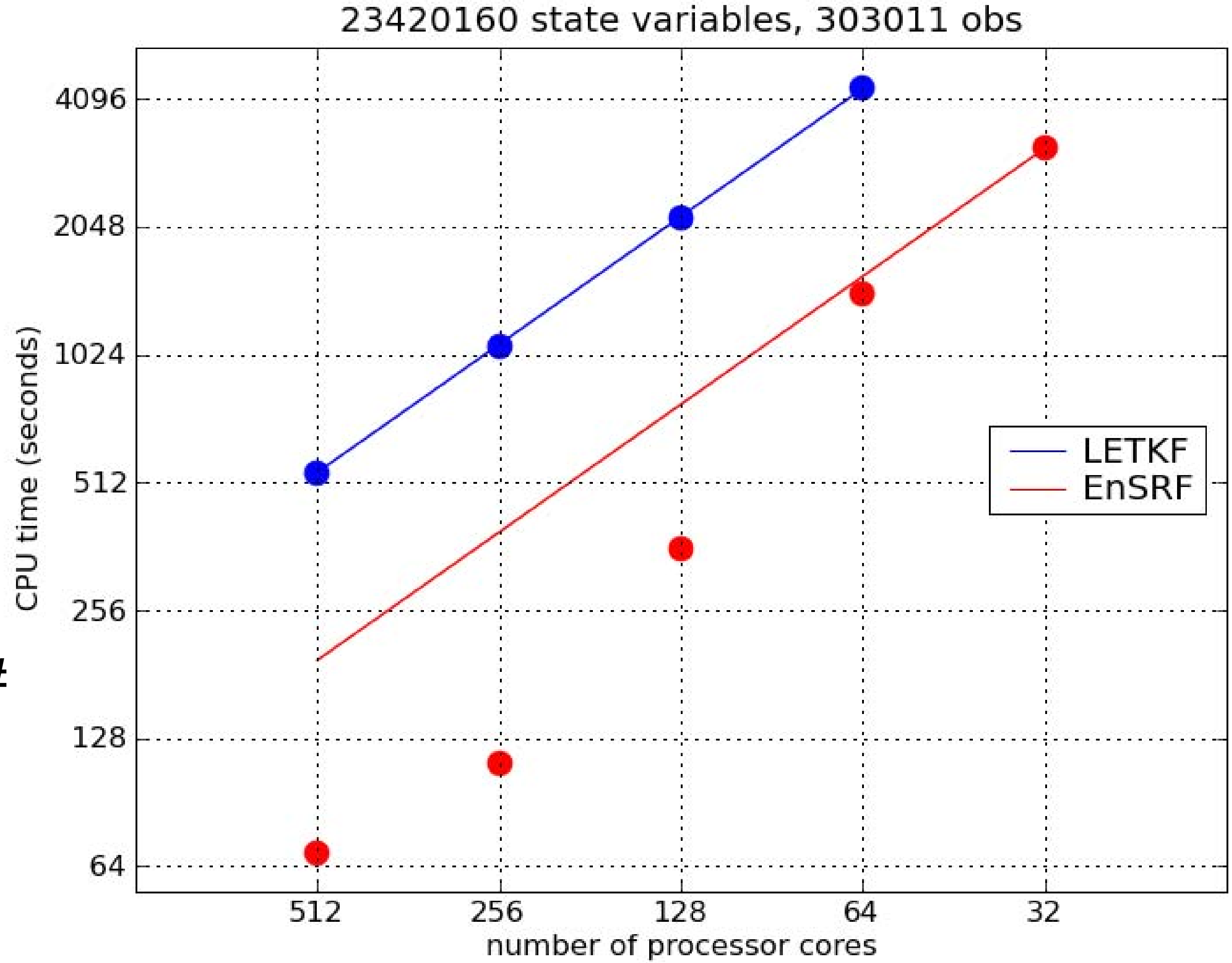
Benchmarks

- Compares only cost of computing increment (no I/O, no forward operator).
- 2100 km, 1.5 scale height localization, K=64 ensemble members. Two cases:
  – 384x190 (T126) analysis grid, two tracers updated. N=23420160, L=33301.
  – 128x64 analysis grid, no tracers updated . N=449820, L=949352.
- 8 core intel cluster, infiniband, mvapich2, intel fortran 10.1/MKL.
- Load balancing using “Graham’s algorithm” - assign each grid pt to processor with least work assigned so far.

Case 1: N = O(100L)

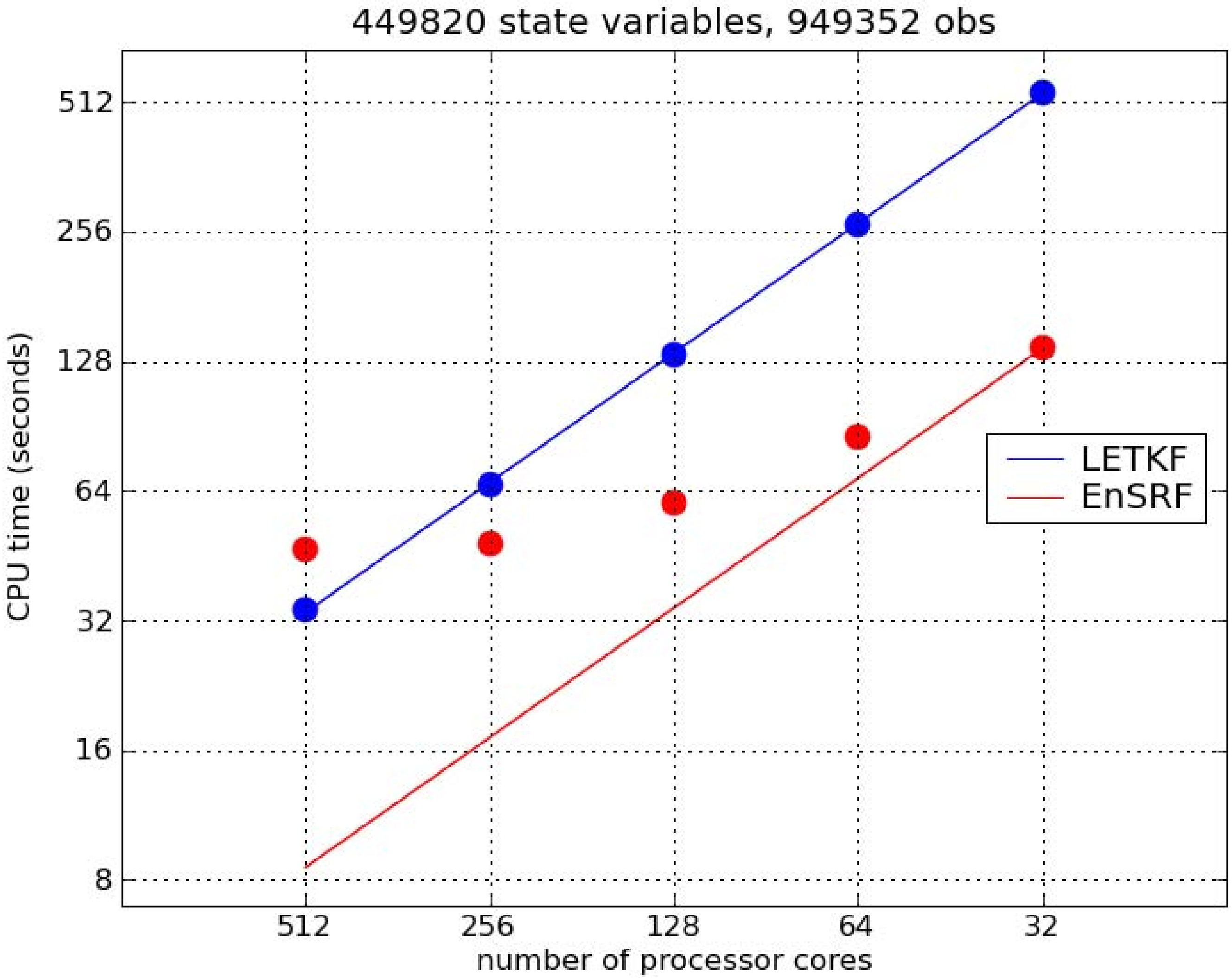
• LETKF scales perfectly, but is 3-7 times slower than EnSRF.

• EnSRF scales better than linear (better cache coherence when # of state vars per proc gets small).



Case 2: N = O(L)

- LETKF scales perfectly.
- EnSRF doesn't scale when # of variables updated on each proc is too small.



Cost of running ensemble dominates as resolution increases

- Because of CFL condition, cost of running model increases by a factor of 8 when horizontal resolution doubles. *This affects calculation of increment in 4D-Var.*
- Calculation of increment in EnKF scales like number of grid points, goes up by a factor of 4.
- Even for modest global resolutions (100-200 km) we find that ensemble forecast step dominates computational cost.
- For EnKF model forecast step scales perfectly, for 4D-Var it depends on model scaling.

Serial EnSRF

- Loop over observations (y_n, n=1..N)
  - n'th observation prior (j'th ens member) $\langle y_{jn} \rangle^b = H \langle \mathbf{x}_j \rangle^b$, $y'^b_{jn} = H\,\mathbf{x}'^b_j$, where $\langle .. \rangle = M^{-1}\Sigma_{j=1..M}$ (1st moment) or $(M-1)^{-1}\Sigma_{j=1..M}$ (2nd moment)
  - Let $d_n = \langle y'^b_{jn}\, y'^b_{jn} \rangle + R_n$, $\alpha_n = (1 + \{R_n / d_n\}^{-1/2})^{-1}$
  - For the i'th state variable $x^b_{ij} = \langle x_{ij} \rangle^b + x'^b_{ij}$
    - $K_{in} = \langle x'^b_{ji}\, y'^b_{jn} \rangle / d_n$ Kalman Gain
    - $\langle x_{ij} \rangle^b = \langle x_{ij} \rangle^b + K_{in}(y_n - \langle y_{jn} \rangle^b)$ update mean for i'th state var
    - $x'^b_{ij} = x'^b_{ij} - \alpha_n K_{in}\, y'^b_{jn}$ update perturbations for i'th state var
  - For the m'th observation prior (y^b_{jm}, m=n..N)
    - $K_{mn} = \langle y'^b_{jm}\, y'^b_{jn} \rangle / d_n$ Kalman Gain
    - $\langle y_{jm} \rangle^b = \langle y_{jm} \rangle^b + K_{mn}(y_n - \langle y_{jn} \rangle^b)$ update mean for m'th ob prior
    - $y'^b_{jm} = y'^b_{jm} - \alpha_n K_{mn}\, y'^b_{jm}$ update perturbation for m'th ob prior
  - Go to (n+1)th observation ($\mathbf{x}^b$ now includes info from obs 1 to n).

Local Ensemble Transform Kalman Filter (LETKF)

Globally:

Forecast step: $\mathbf{x}^b_{n,k} = M_n\left(\mathbf{x}^a_{n-1,k}\right)$

Analysis step: construct $\mathbf{X}^b = \left[\mathbf{x}^b_1 - \overline{\mathbf{x}}^b \mid \ldots \mid \mathbf{x}^b_K - \overline{\mathbf{x}}^b\right];$

$$\mathbf{y}^b_i = H(\mathbf{x}^b_i);\ \mathbf{Y}^b_n = \left[\mathbf{y}^b_1 - \overline{\mathbf{y}}^b \mid \ldots \mid \mathbf{y}^b_K - \overline{\mathbf{y}}^b\right]$$

Locally: Choose for each grid point the observations to be used, and compute the local analysis error covariance and perturbations in ensemble space:

$$\tilde{\mathbf{P}}^a = \left[(K-1)\mathbf{I} + \mathbf{Y}^{bT}\mathbf{R}^{-1}\mathbf{Y}^b\right]^{-1};\ \mathbf{W}^a = [(K-1)\tilde{\mathbf{P}}^a]^{1/2}$$

Analysis mean in ensemble space: $\overline{\mathbf{w}}^a = \tilde{\mathbf{P}}^a\mathbf{Y}^{bT}\mathbf{R}^{-1}(\mathbf{y}^o - \overline{\mathbf{y}}^b)$

and add to $\mathbf{W}^a$ to get the analysis ensemble in ensemble space

The new ensemble analyses in model space are the columns of $\mathbf{X}^a_n = \mathbf{X}^b_n\mathbf{W}^a + \overline{\mathbf{x}}^b$. Gathering the grid point analyses forms the new global analyses. Note that the the output of the LETKF are analysis weights $\overline{\mathbf{w}}^a$ and perturbation analysis matrices of weights $\mathbf{W}^a$. These weights multiply the ensemble forecasts.