How low can you go? Reducing the precision of data assimilation to improve weather forecast skill

Sam Hatfield, Peter Düben (2), Tim Palmer (1), Keiichi Kondo (3), Takemasa Miyoshi (3)

(1) Atmospheric, Oceanic and Planetary Physics, University of Oxford

(2) European Centre for Medium Range Weather Forecasts

(3) RIKEN Advanced Institute for Computational Science

UNIVERSITY OF OXFORD

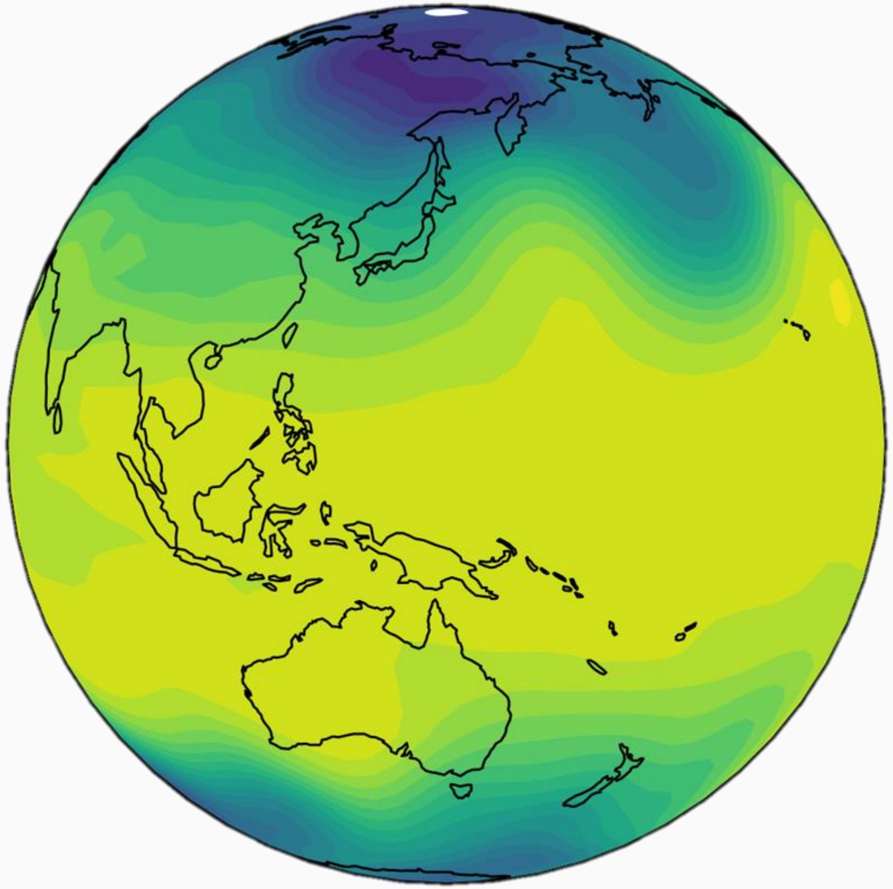
NERC SCIENCE OF THE ENVIRONMENT

Motivation

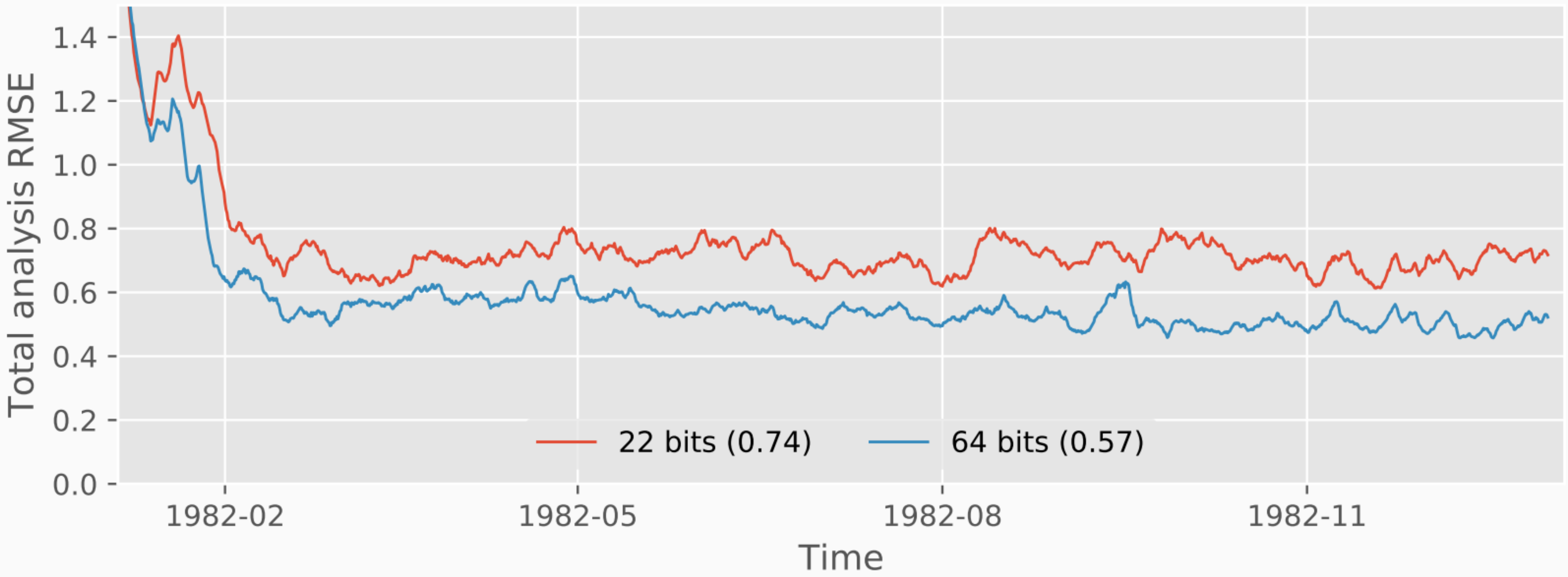
- Studies indicate that precision can be reduced in weather and climate models, without impacting skill scores significantly
- This is because we have **model uncertainty** due to unresolved scales/processes
- Reduced precision models run faster, so we can afford to use higher resolution/larger ensembles etc.
- What about data assimilation?

SPEEDY/LETKF

- First stage of project, Lorenz '96, complete
- Now, move on to SPEEDY/LETKF
- Summer goals:
  1. Add model error to SPEEDY, compare with reduced precision error (using reduced precision emulator)
  2. Reduce precision in LETKF, measure speed-up and analysis quality

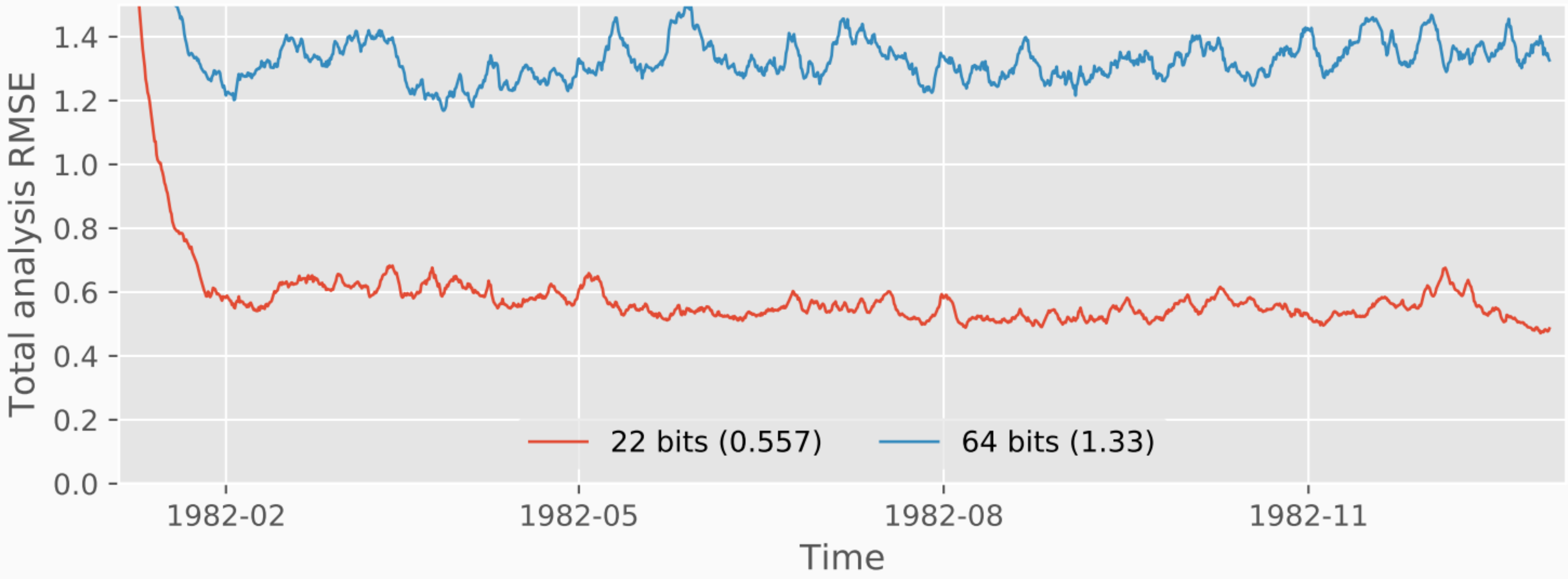


Why do you need model error?



- Error is 25% higher for 22 bit SPEEDY, compared with 64 bit SPEEDY
- **But this experiment is biased towards 64 bit SPEEDY**
- 22 bit model has error (precision error), but 64 bit model does not (nature run is also 64 bits)

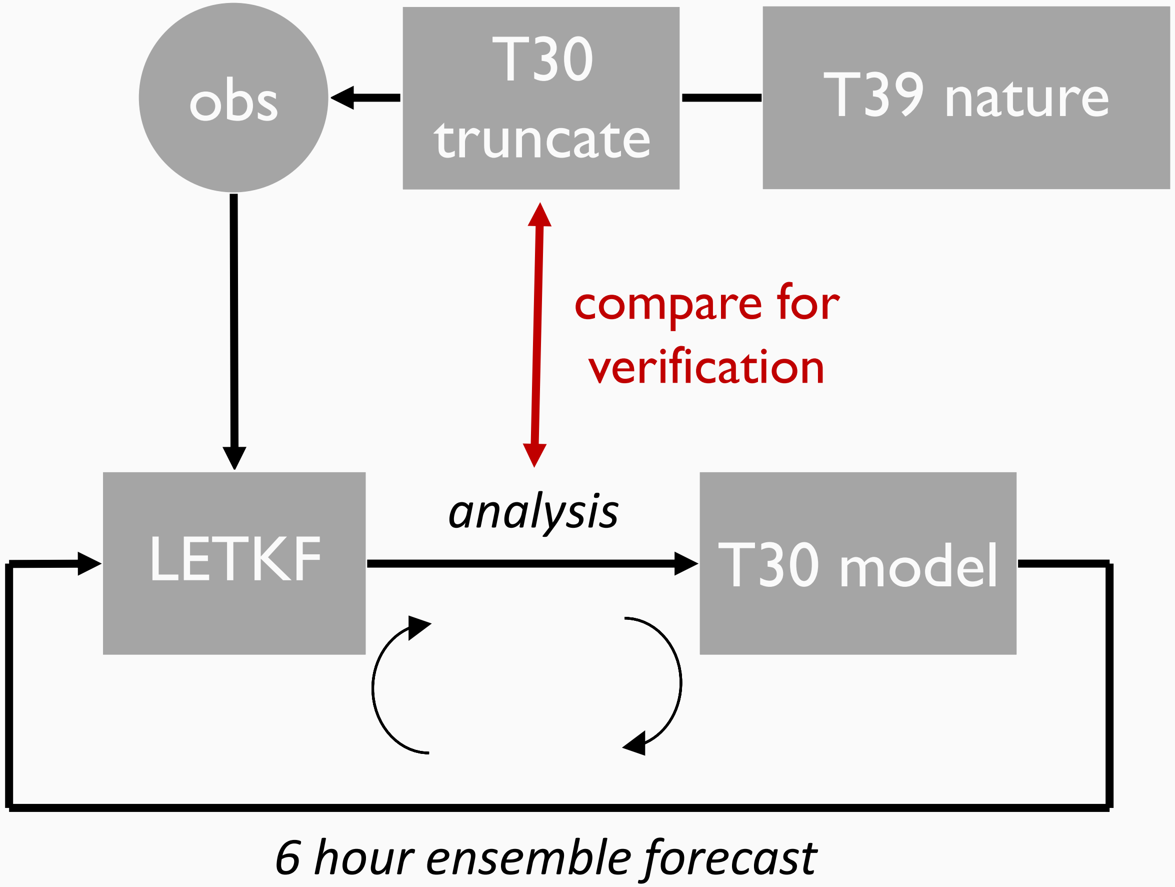
Why do you need model error?



- Same experiment but with **22 bit nature run**
- Now, 64 bits is worse than 22 bits!

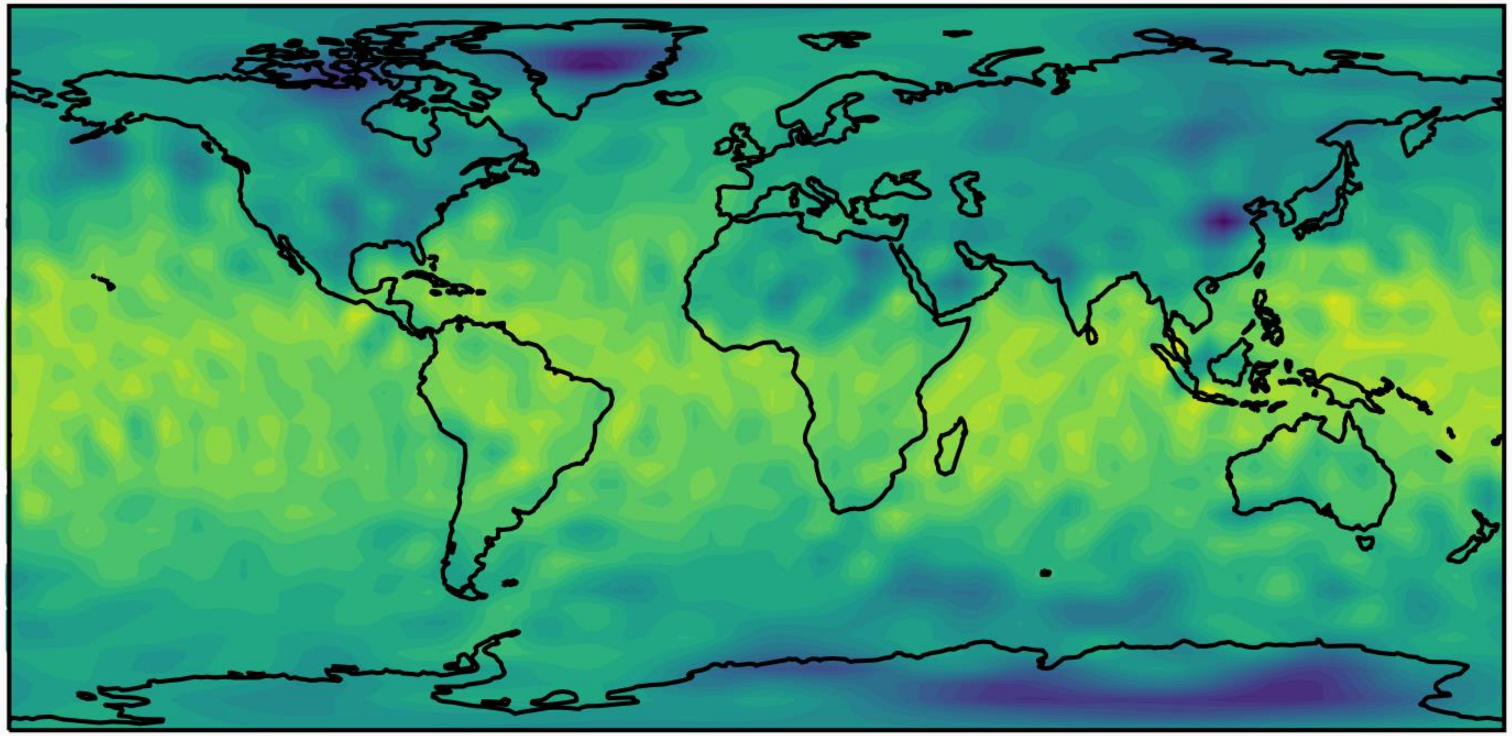
Adding model error (attempt 1)

- First attempt to add model error: **use a higher resolution nature run (T39 nature vs T30 in LETKF)**
- Run model at T39, truncate to T30, generate synthetic observations as before
- Now, there is error due to **unresolved scales**, and **representativity error** in the observations



Adding model error (attempt 1 result)

- It didn't work!
- A strange spherical harmonic pattern appears in Q and T fields, and the model eventually crashes
- I'll come back to this later

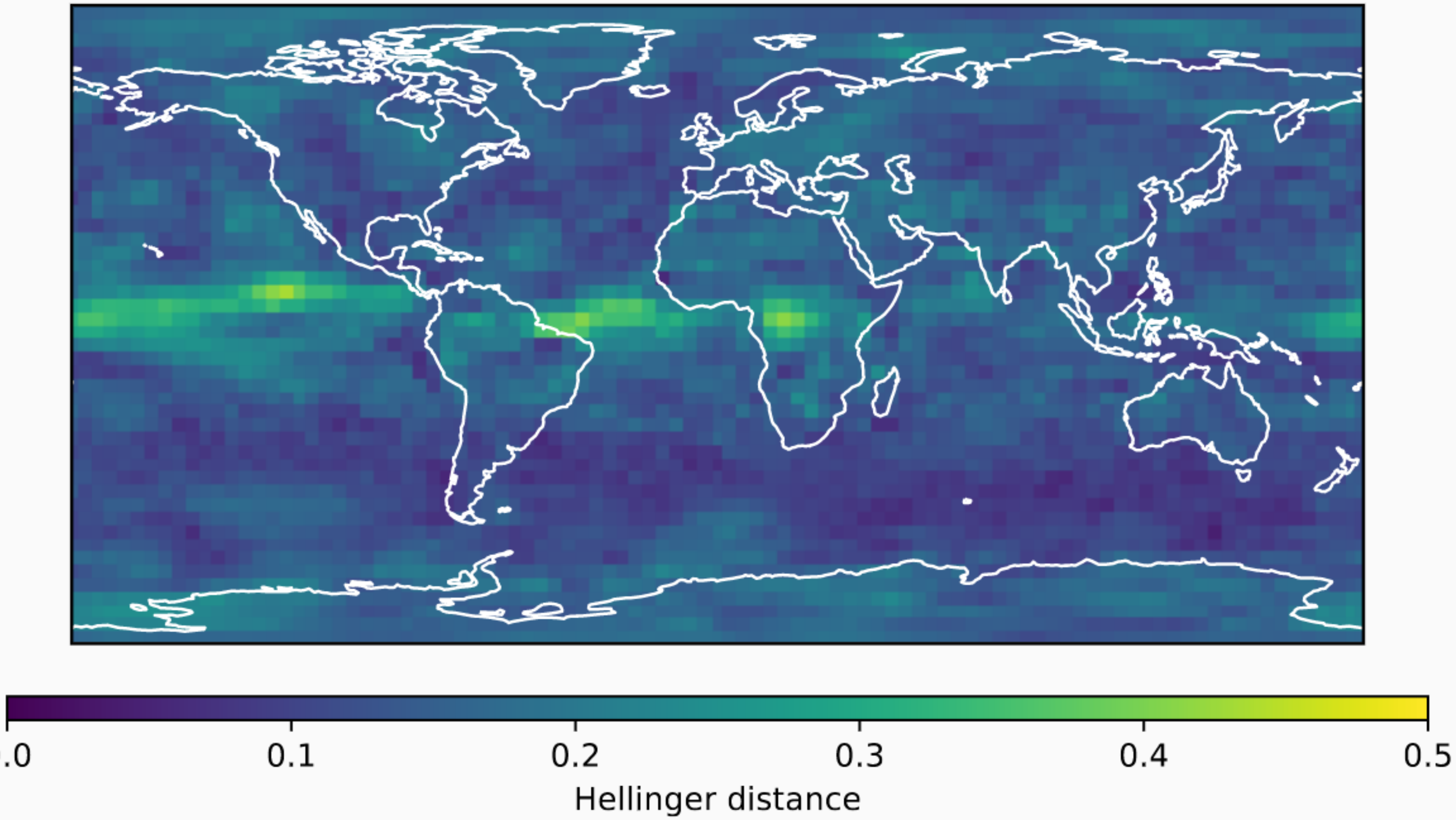


Lowest level specific humidity after 2 months of assimilation

Adding model error (attempt 2)

- Easier way to add model error: perturb parameters
- I perturb some parameters in convection scheme and diffusion and wind drag
- To check climatological changes, I compute **Hellinger distance**
- This measures "distance" between two PDFs
- HD = 0 → identical, HD = 1 → completely different

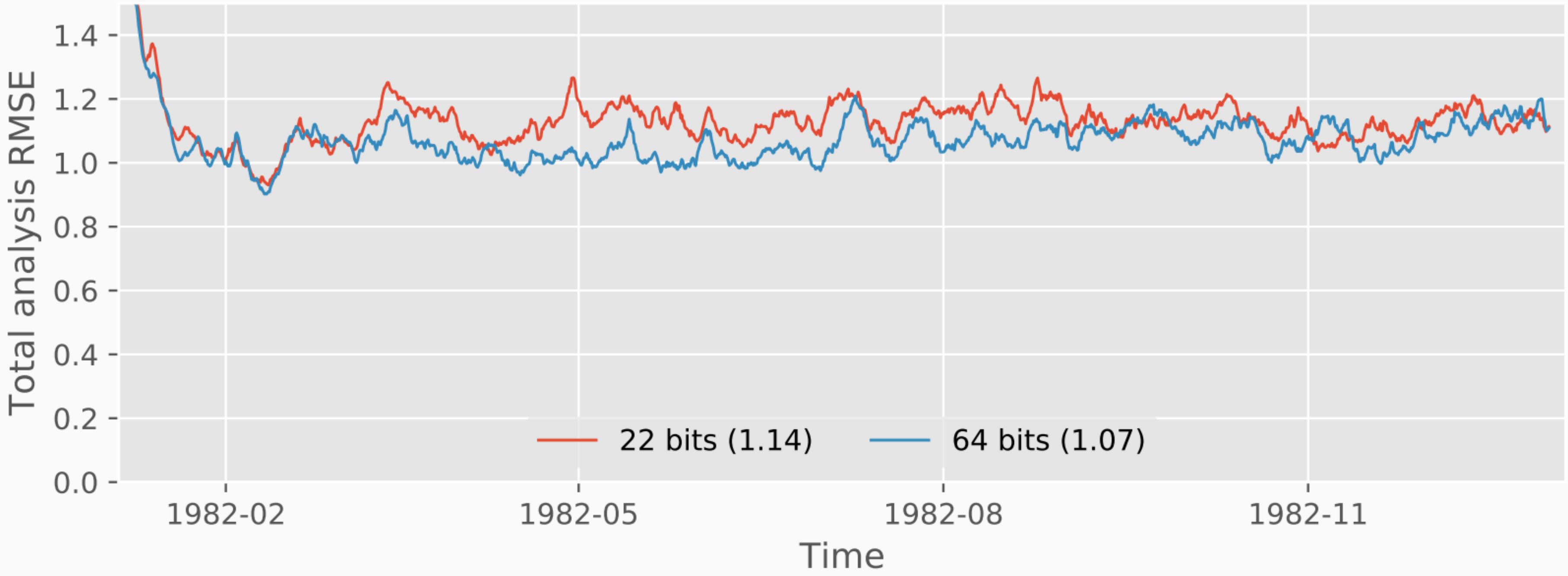
Climatological changes



Lowest level temperature climatological change after perturbing parameters

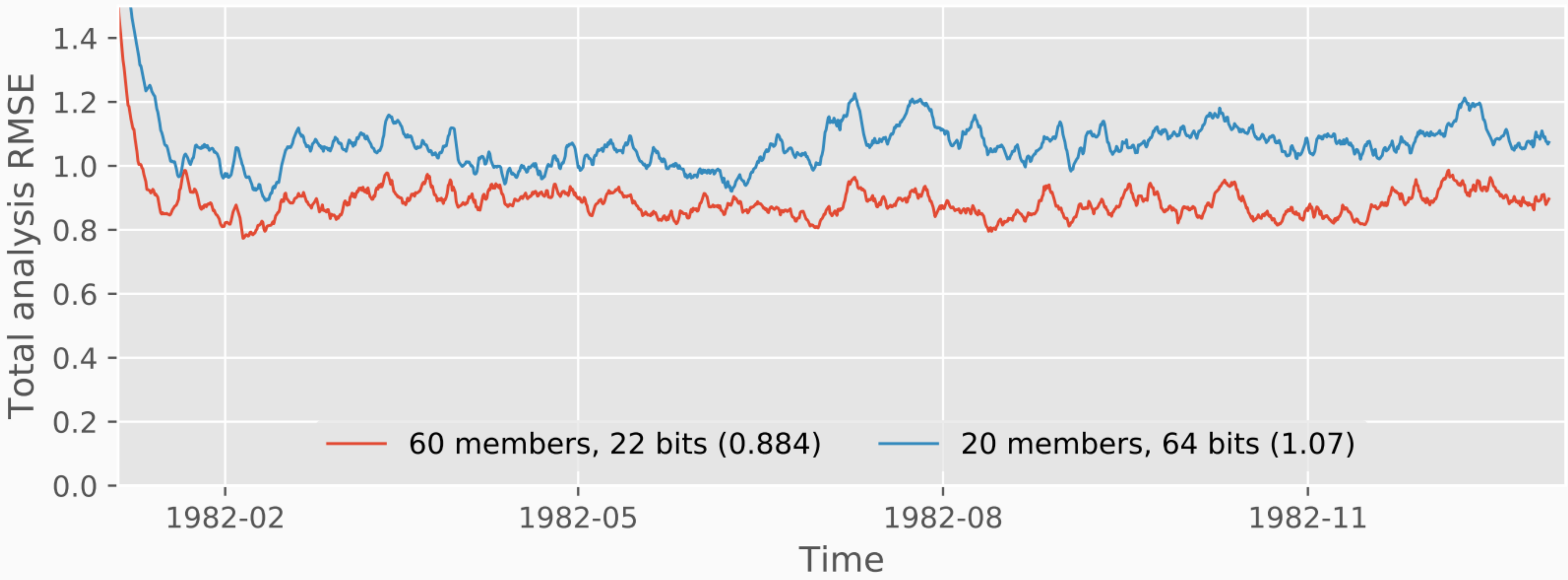
- Biggest difference in tropical convective regions

Adding model error



- After perturbing parameters, 22 bit SPEEDY is only about 5% worse than 64 bit SPEEDY
- **When there is model error, you can reduce precision further**

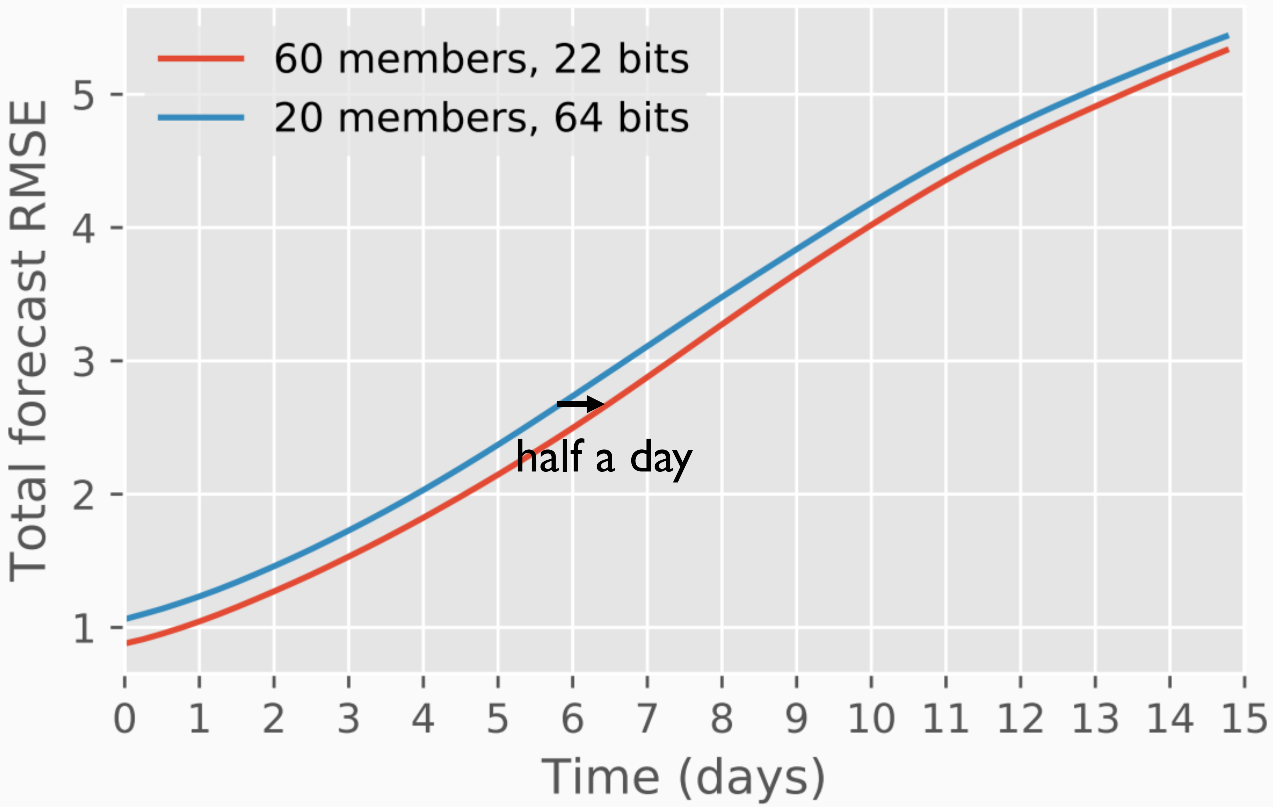
Trading precision for ensemble size



- Cost(22 bits, 60 members) ≈ cost(64 bits, 20 members)
- **Trade precision for ensemble size**
- This reduces error by 17% for (hypothetically) no extra cost

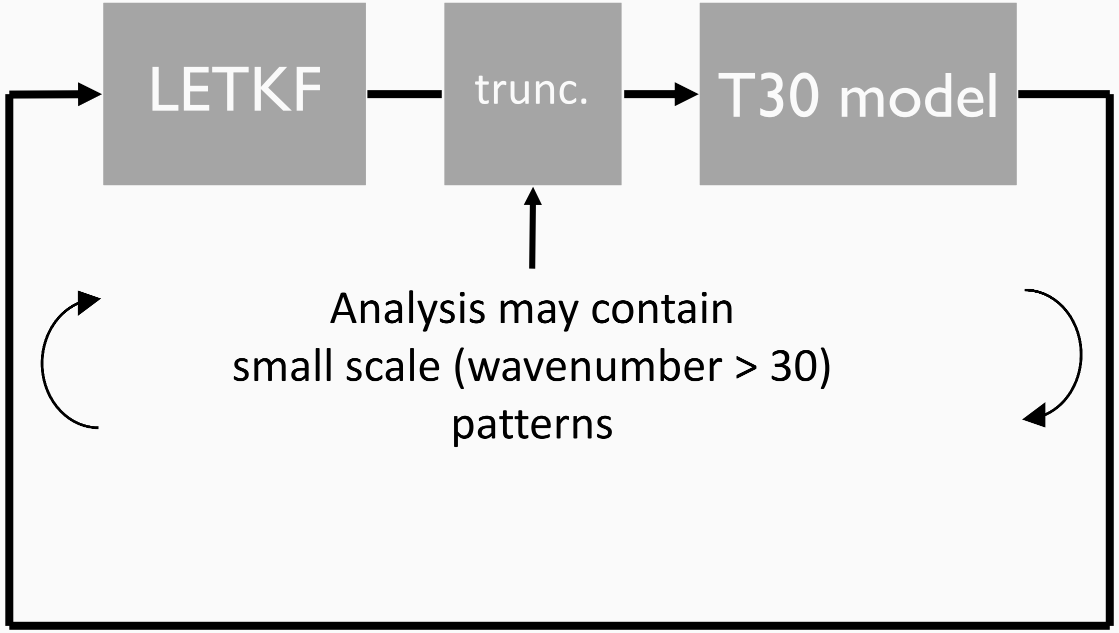
Improved forecasts

- This gives improved forecasts
- Forecast horizon is extended by **half a day**



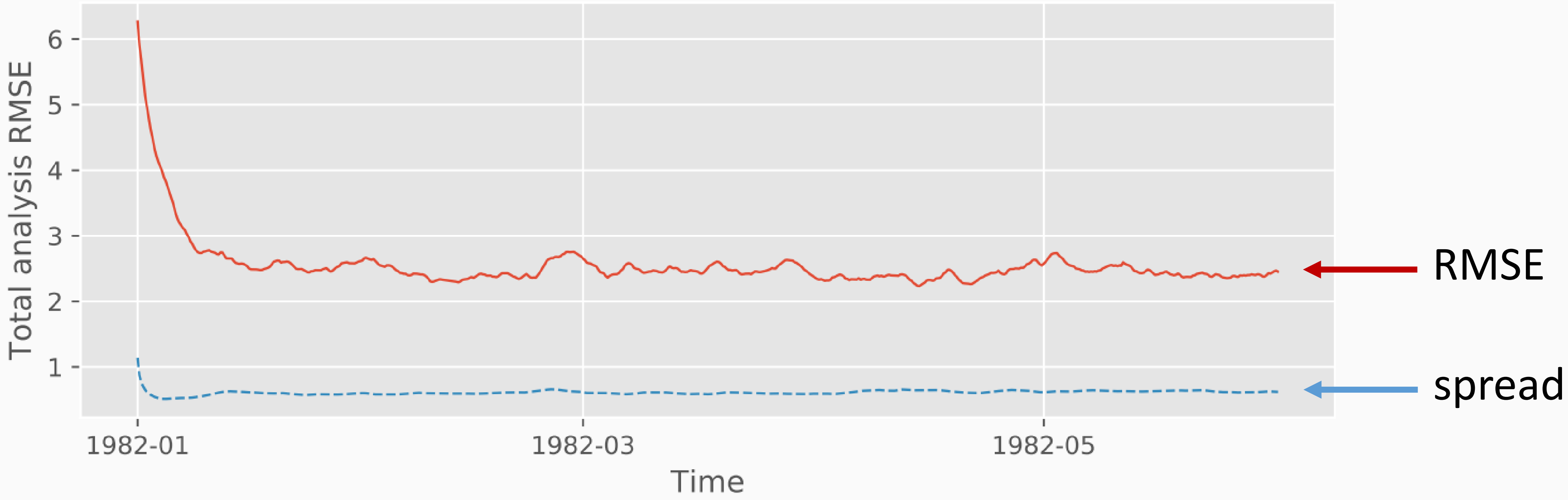
Adding model error (attempt 1 again)

- The strange pattern was caused by a bug
- The analysis fields were not truncated to T30 before running the 6 hour forecast
- Small scale waves amplified and crashed the model
- After fixing this, I can successfully assimilate obs. from the T39 nature run



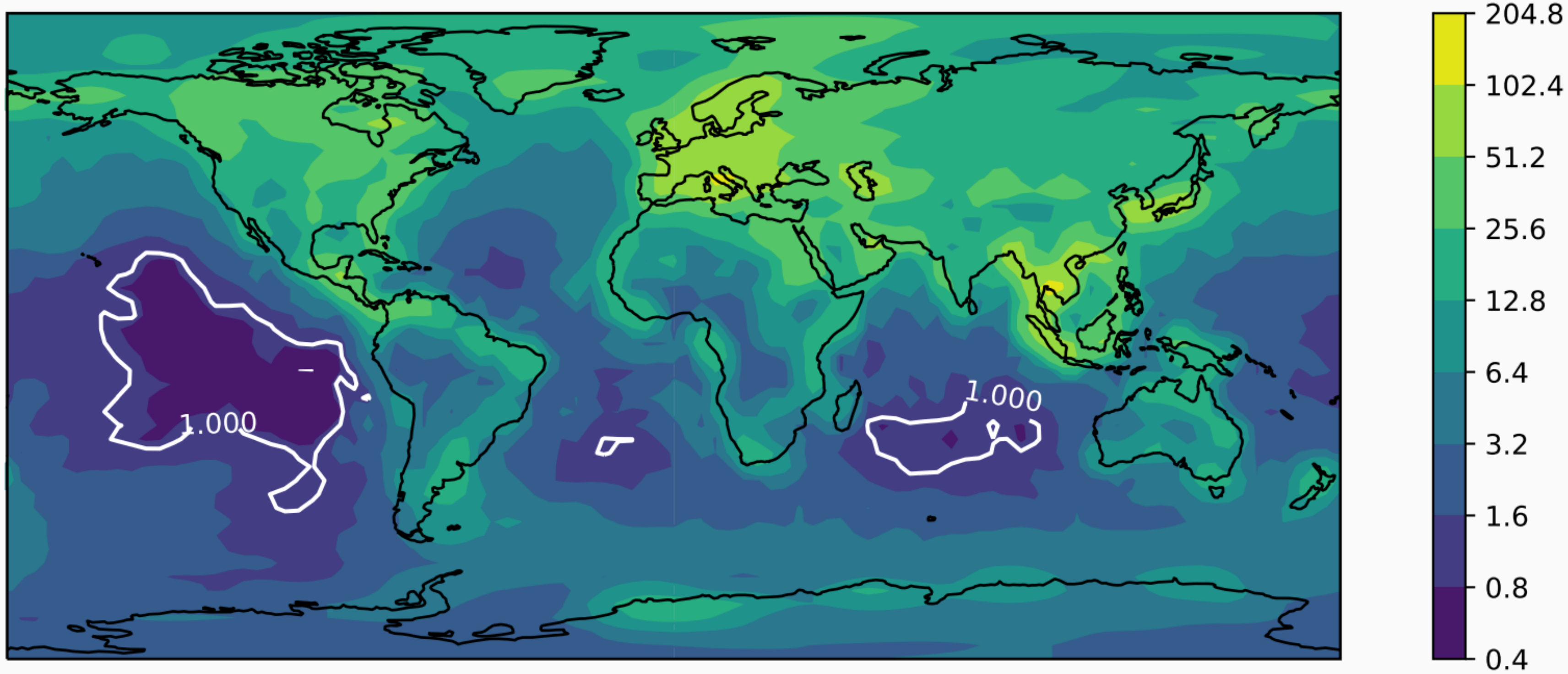
6 hour ensemble forecast

Adding model error (attempt 1 again)



- After truncating the fields, it works!
- But it's very underdispersive, especially over land/sea boundaries
- Work in progress… (RTPP, RTPS?)

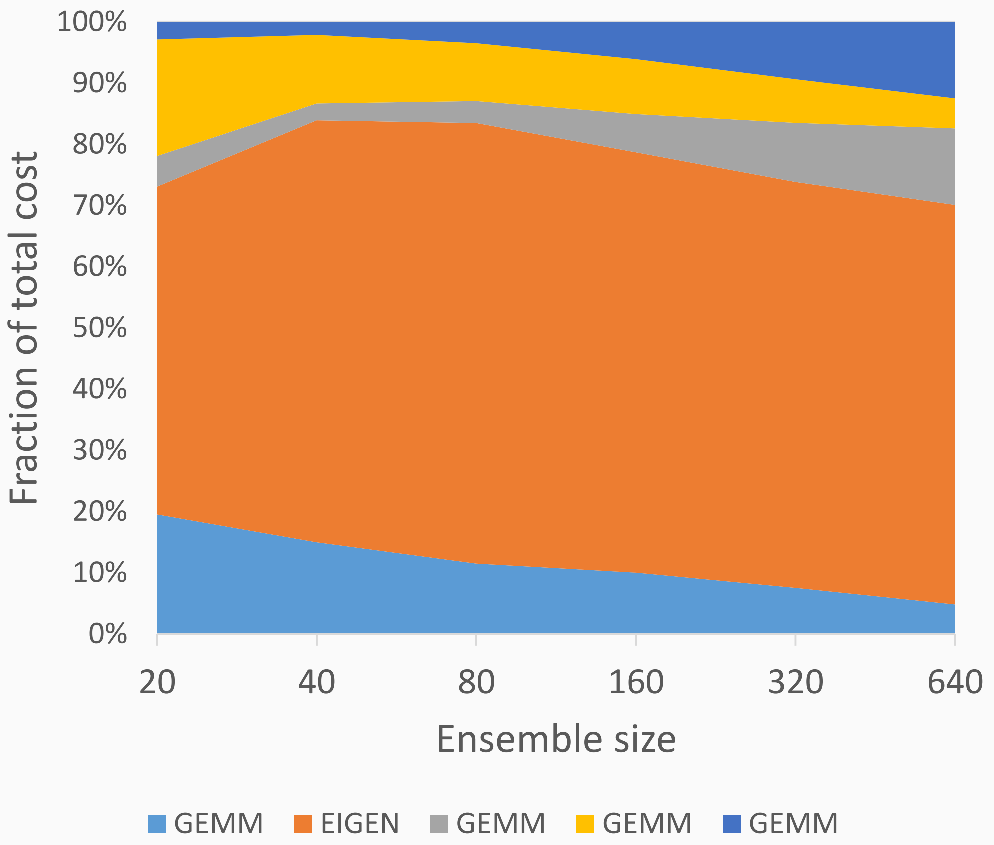
Underdispersiveness



- Time mean RMSE ÷ spread, specific humidity, bottom level
- The pattern doesn’t appear to be correlated with observation density

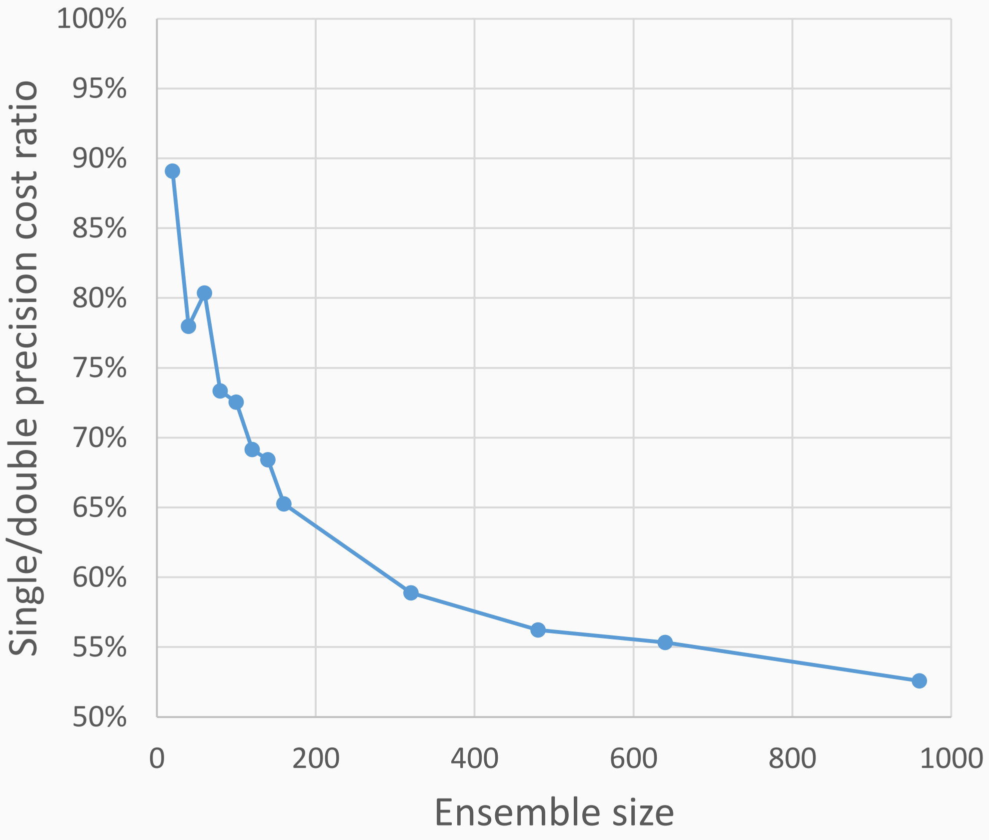
Reducing precision in LETKF

- The LETKF consists of five expensive operations
  - 4 general matrix multiplications (“GEMM”)
  - 1 eigenvalue/vector decomposition
- The cost ratio depends on ensemble size
- The eigendecomposition is ~60% of the total cost



Reducing precision in LETKF

- I measure the total time for all 5 operations at single and double precision (**per gridpoint**)
- Setup: Intel Fortran, MKL library (i.e. optimized BLAS/LAPACK), only 20 nodes
- As ensemble size increases, single/double precision cost ratio **approaches 50%**
- With 240 members, single is ~40% faster than double



Impact on error scores

- No measurable impact on RMSE when reducing precision of LETKF to single precision
- Table shows RMSE averaged over 4 months after 2 months assimilation spin-up
- But, for now, it's still safer to check single and double are the same → especially for larger ensembles

ensemble size	single	double
20	0.335	0.335
40	0.266	0.265
80	0.213	0.221

Conclusion

- If you have model error, you can lower precision further
- Low precision, large ensembles are better than high precision, small ensembles
- LETKF is feasible at single precision
- Up to 50% reduction in wall clock time for large ensembles
- **Future work:**
  - Make high-res nature run experiment work
  - Do assimilation with single precision SPEEDY and LETKF, measure speed up
  - Half precision LETKF?