Significant Advance in Calculating EOF From a Very Large Data set.

Huug van den Dool

A few 1% / 99% comments

- 99% of the Reanalysis users are happy with 1% of the data
- The other 99% of the data is required (if not demanded) by the remaining 1% of the users.
- That 1% of the users thus has a lot of impact on data storage and data transmission issues
- How do you split data wisely into 1%/99%? One common way: High-Res and Low-Res versions (both time and space).

W.J.A. Kuipers 1970 seminar at Dutch Weather Service (KNMI) about EOF

- This was not about teleconnections, NAO, PNA
- Not about the workings of nature or to 'let the data speak' diagnostics
- It was about reducing a data set to ~0.1-1% of its original size to fit a ~1970 computer
- Curiously, while EOFs are maximally efficient for data compression, it is very costly (CPU, Memory) to calculate EOFs, but, keep in mind: you have to do this only once.
- Curiously, data compression is as desirable now as it was in 1970.

- Almost everybody ….. has calculated EOF
- My guess: They all did it by first calculating the covariance matrix Q or Q^a.
- Evaluating the elements of the covariance matrix requires $n_s * n_s * n_t$ (Q) or $n_t * n_t * n_s$ (Q^a) multiplications where n_s and n_t are the number of points in space and time.
- Example (CFSR) Reanalysis full resolution data:

  space =1152*576 =663552 points, and
  time=32*365*24 =280320

  ($n_s * n_t = 1.86 * 10^{11}$)

➔ It would take $5.22*10^{16}$ ($1.23*10^{17}$) multiplications to fill Q^a (Q) (for one variable). That is before any EOFs are calculated.!!

Here is the good news:

- We can find EOFs without first filling the covariance matrix.
- This advance addresses CPU problems, not so much memory problems (which are also an obstacle, but 2nd to CPU).
- So we can calculate EOFs from CFSR ?

Lay-out talk

- The new(?) method
- Examples to show that it actually works and yields the expected results
- Why does it work?
- Origin of the method

Basics: $T(s, t) = \sum_m \alpha_m(t)\, e_m(s)$ (0)

- Multiply lhs and rhs of (0) by $\alpha_n(t)$ and sum over all times t (n is a specific mode number). Result:

  $e_m(s) = \sum_t \alpha_m(t)\, T(s, t) / \sum_t \alpha^2_m(t)$ (1)

- Multiply lhs and rhs side of (0) by $e_n(s)$ and sum over all space s. Result:

  $\alpha_m(t) = \sum_s w(s)\, e_m(s)\, T(s, t) / \sum_s w(s)\, e^2_m(s)$ (2)

where w(s) are spatial weights, not shown below.

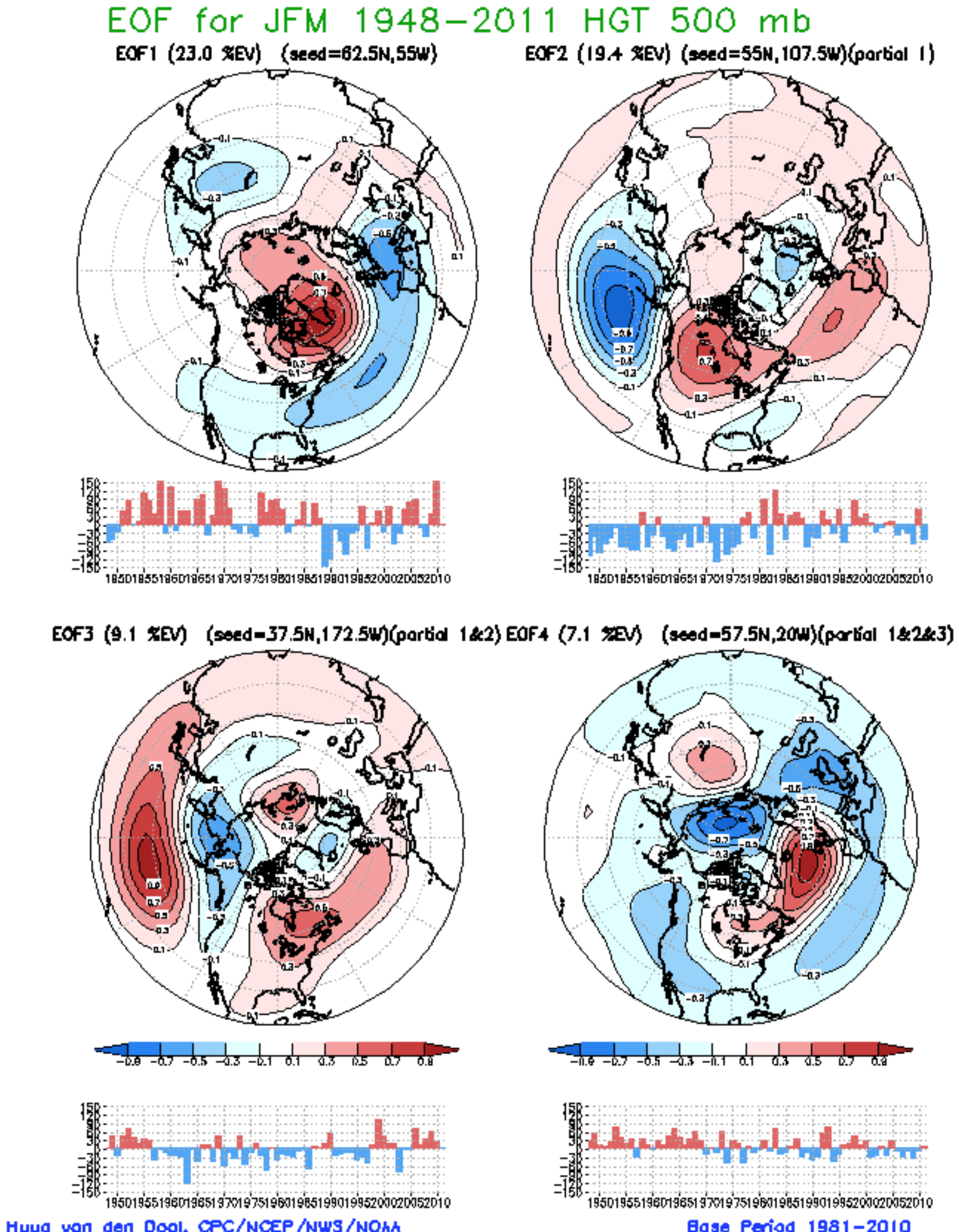
$$e_m(s) = \sum_t \alpha_m(t)\, T(s, t) \,/\, \sum_t \alpha^2_m(t) \quad (1)$$

$$\alpha_m(t) = \sum_s e_m(s)\, T(s, t) \,/\, \sum_s e^2_m(s) \quad (2)$$

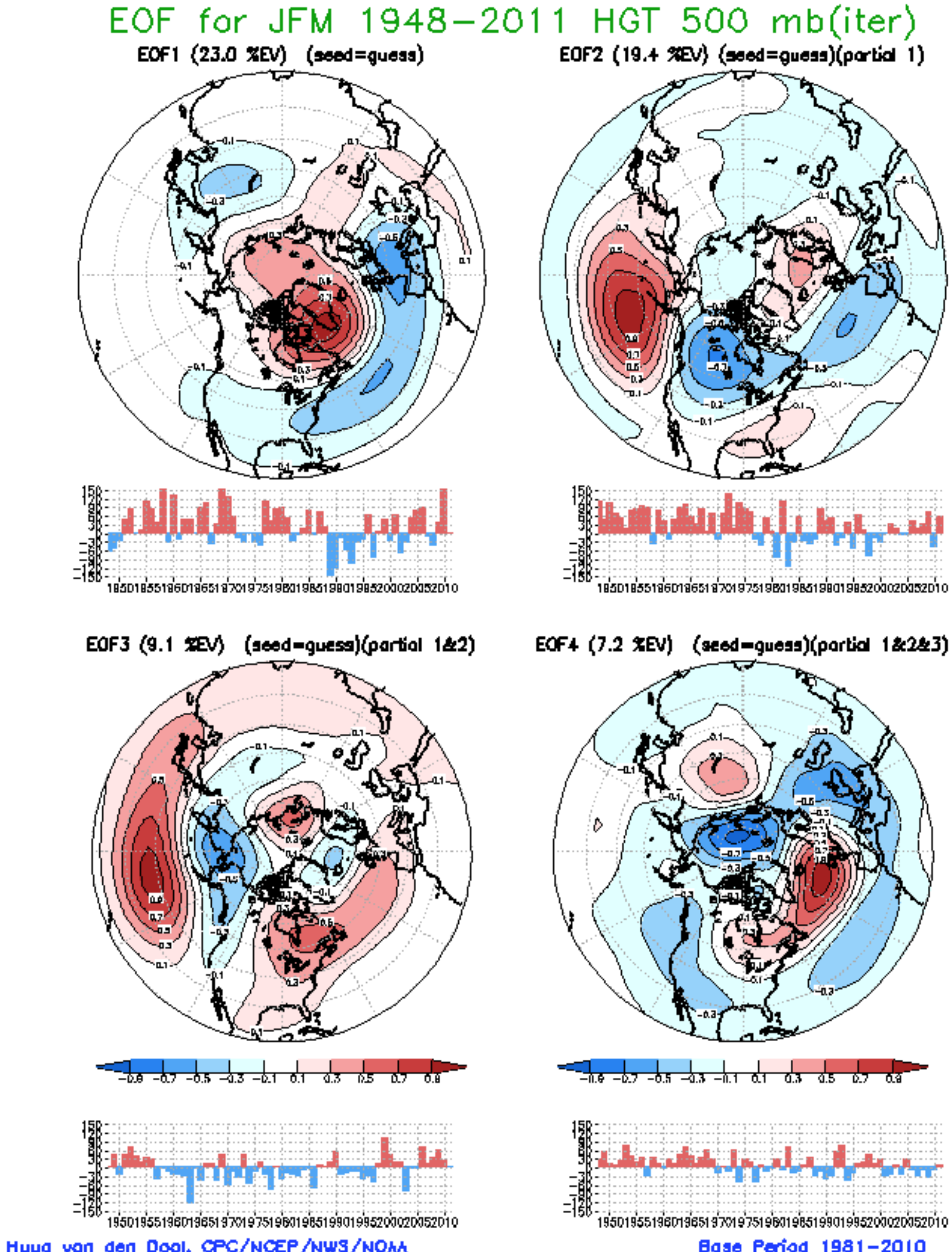
- The above are orthogonality relationships.
- And the basis for an iteration!
- Randomly pick (or make) a time series $\alpha^0(t)$, and stick into (1). This yields $e^0(t)$. Stick $e^0(t)$ into (2). This yields $\alpha^1(t)$. This is one iteration. Etc. This generally converges to the first EOF $\alpha_1(t)$, $e_1(s)$
- $T^{reduced}(s,t) = T(s,t) - \alpha_1(t)\, e_1(s)$ and repeat. One finds mode#2. Etc.

Example to show it works

- This is a lowres example when the cov matrix can be calculated.
- 1948-2011 JFM seasonal Z500 mean at 2.5 degree grid. 64 time levels
- Domain 20N-pole (144*29=4176 gridpoints)

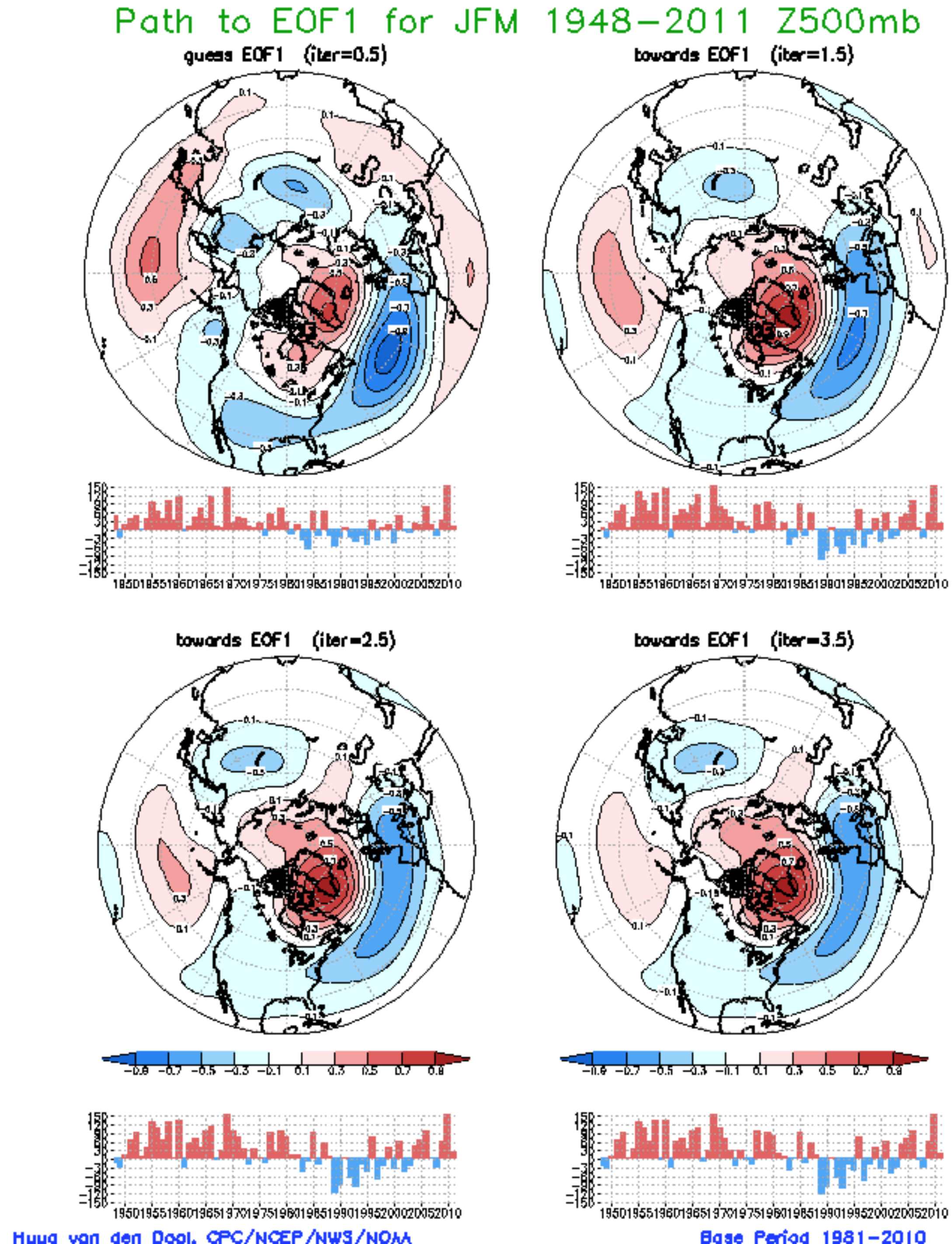


Example using a covariance matrix



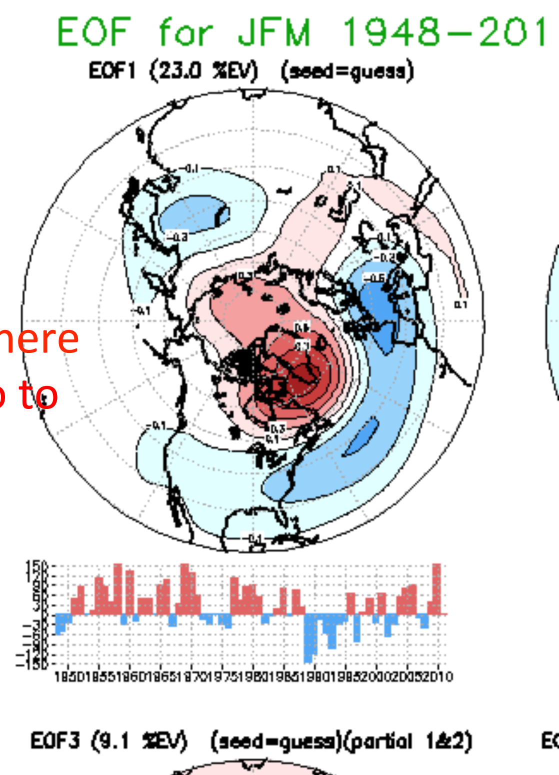
Example using an iterative method (100 iterations)

The route to EOF#1 taken by the iteration from a guess

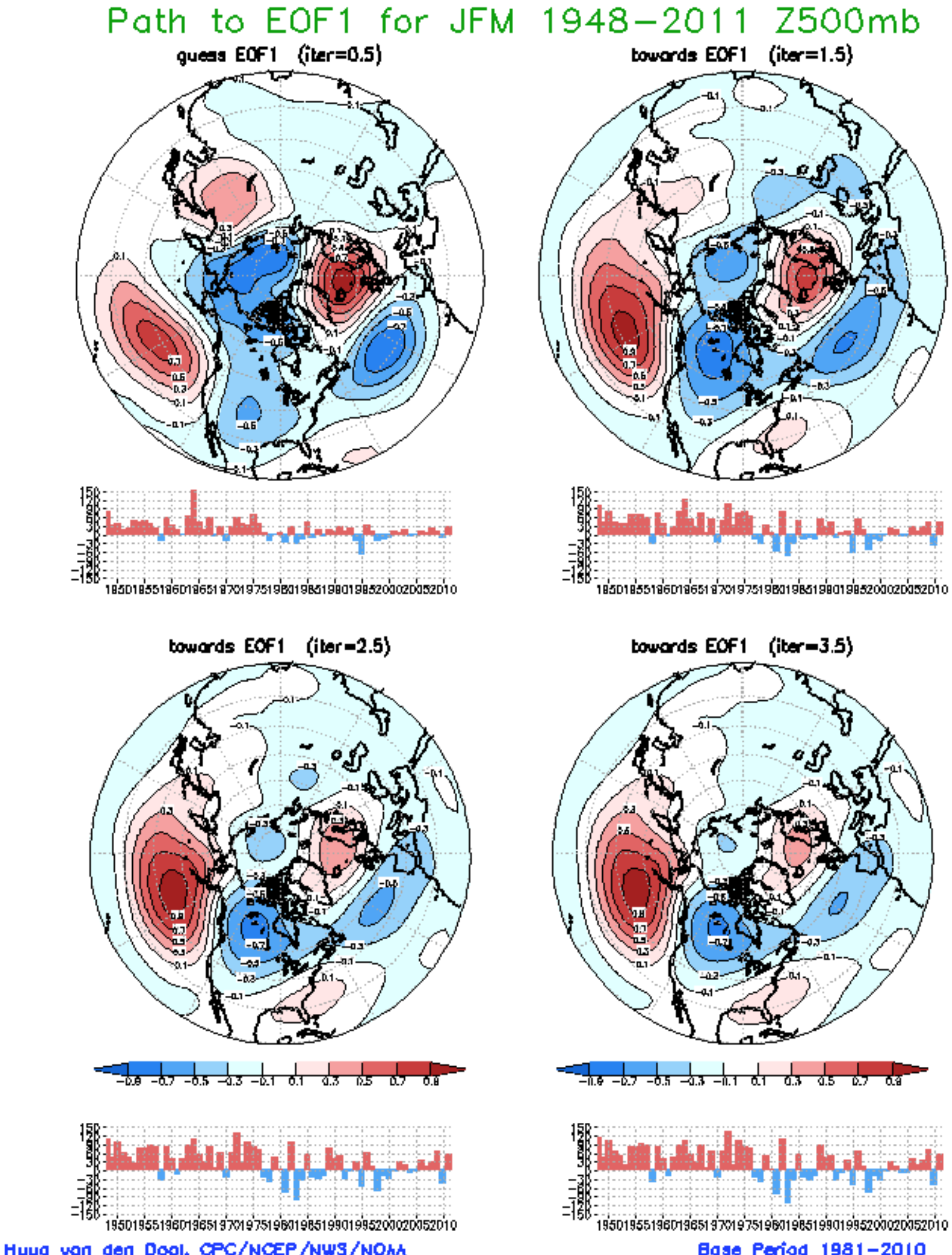


The route to EOF1 from a guess field (pick: (1948+2010)/2)
Iterations 1-4

→ Where we go to

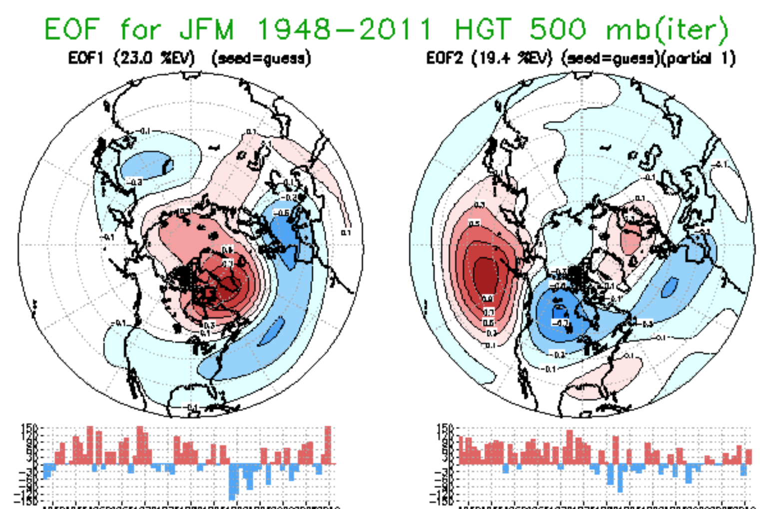


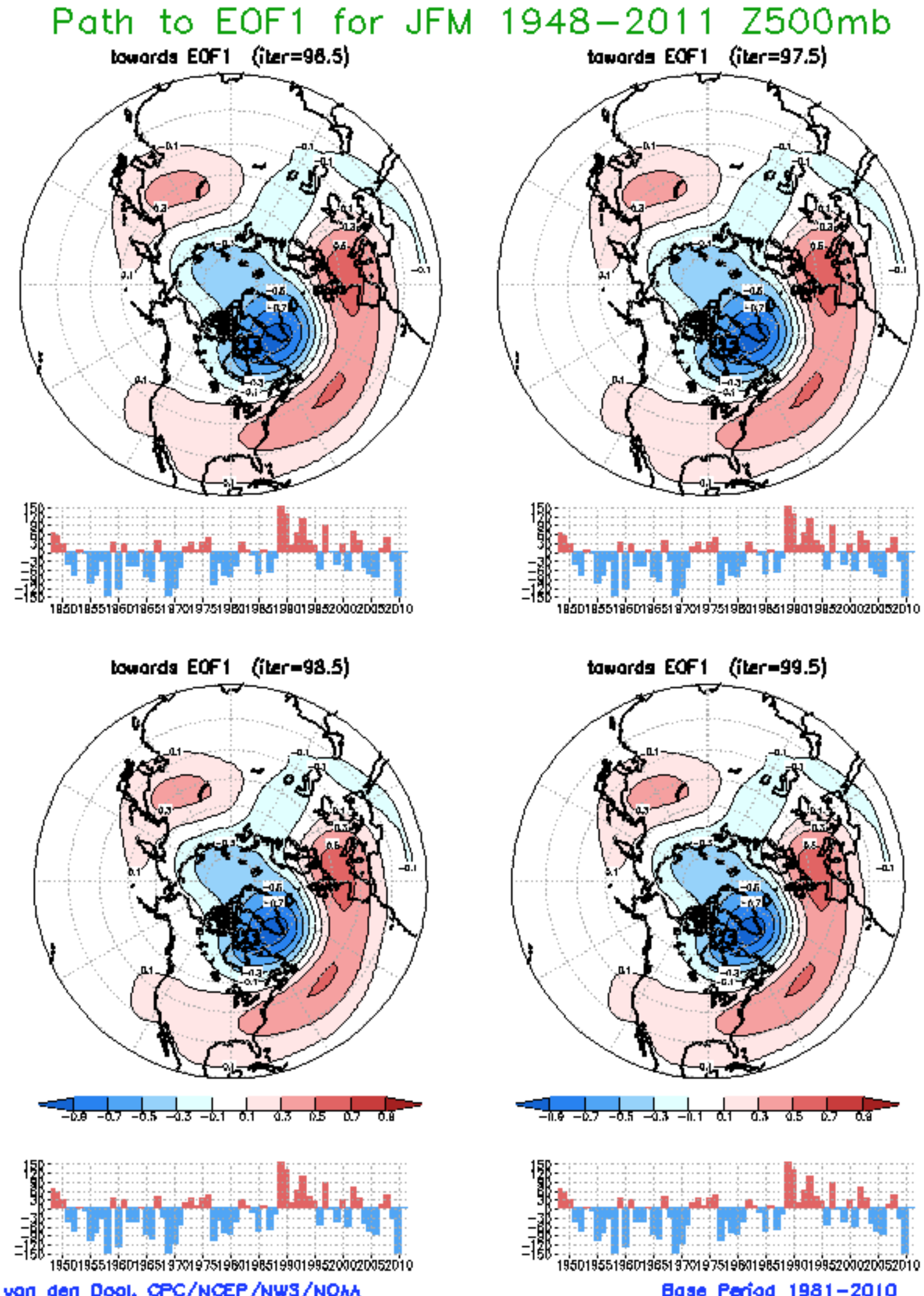
An example of the challenge with convergence:



The route to EOF1 from a very different guess field (pick: 1964)
Iterations 1-4

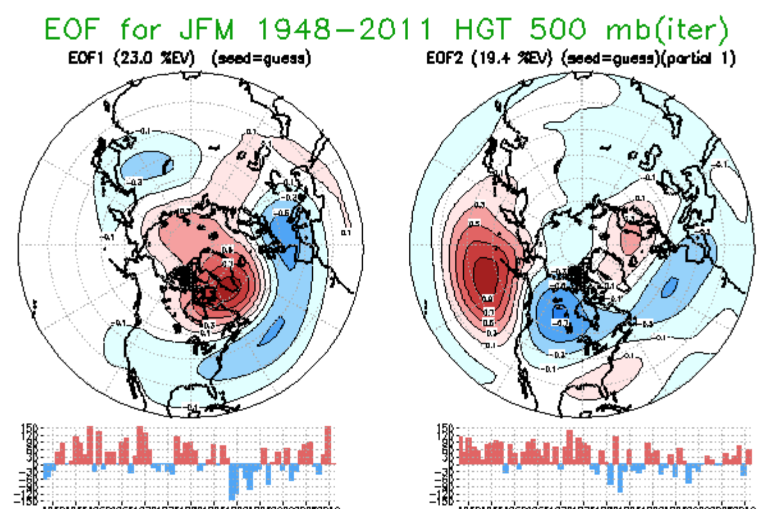
↓ Where we go to



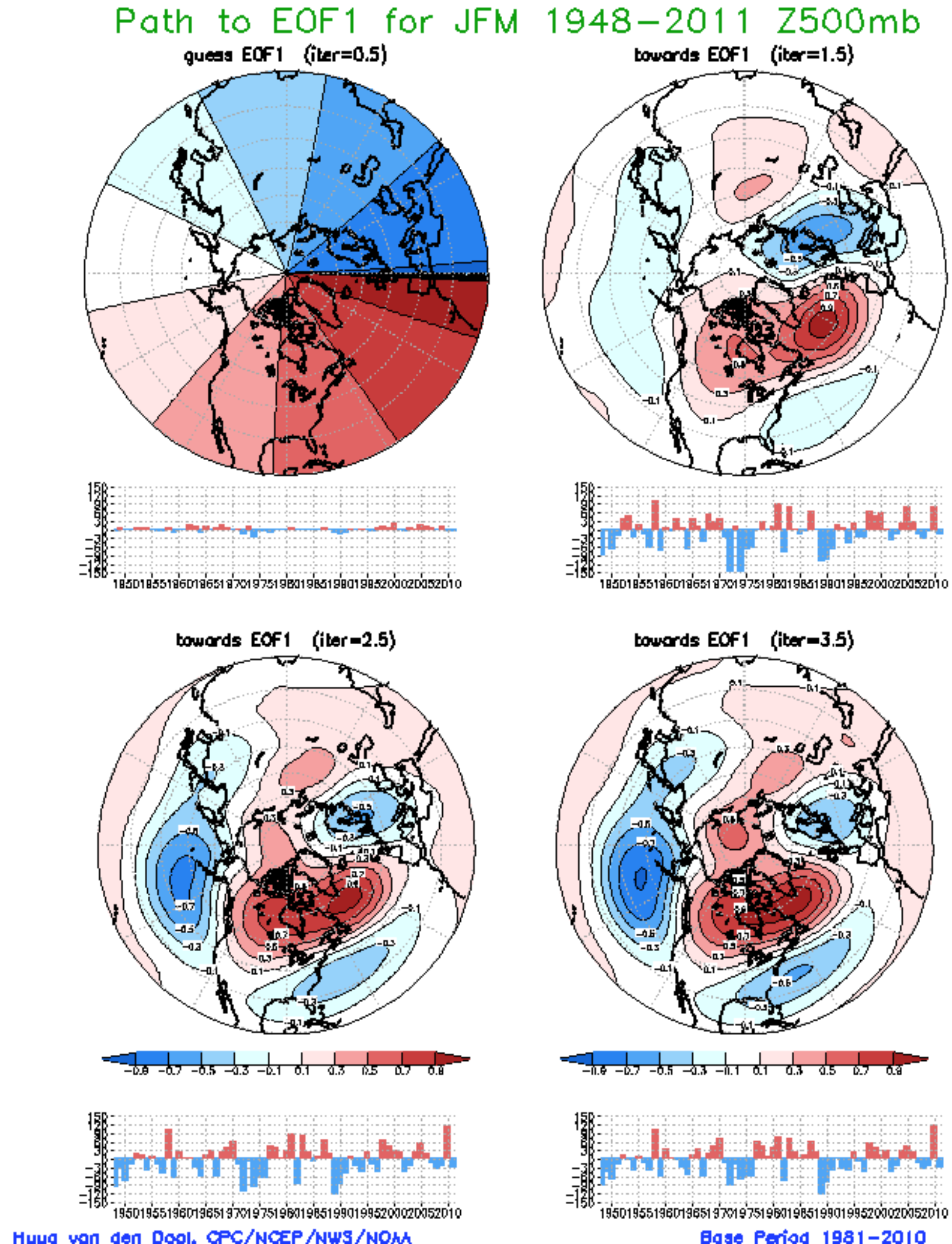


The route to EOF1 from a very different guess field (pick: 1964) Iterations 97-100!

↓ Where we go to

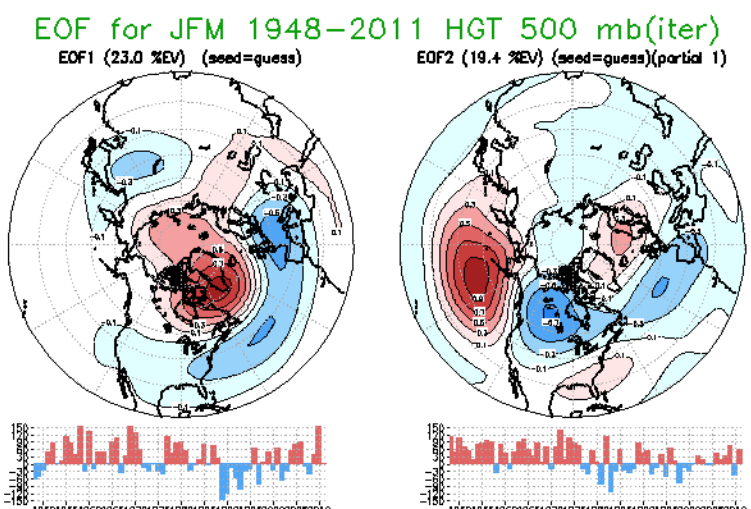


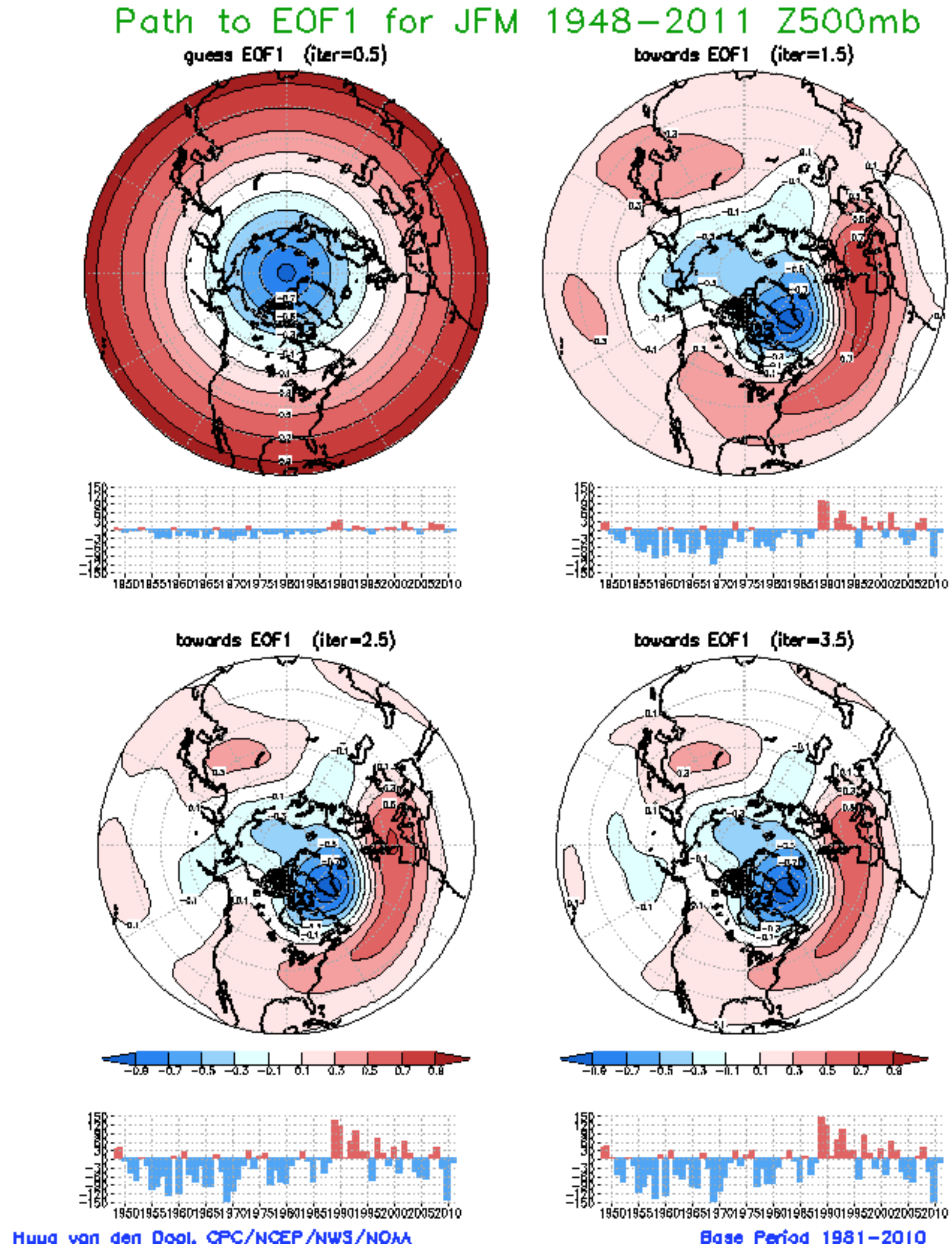
Starting from a ‘ridiculous’ guess



The route to EOF1 from a ridiculous guess field
Iterations 1-4

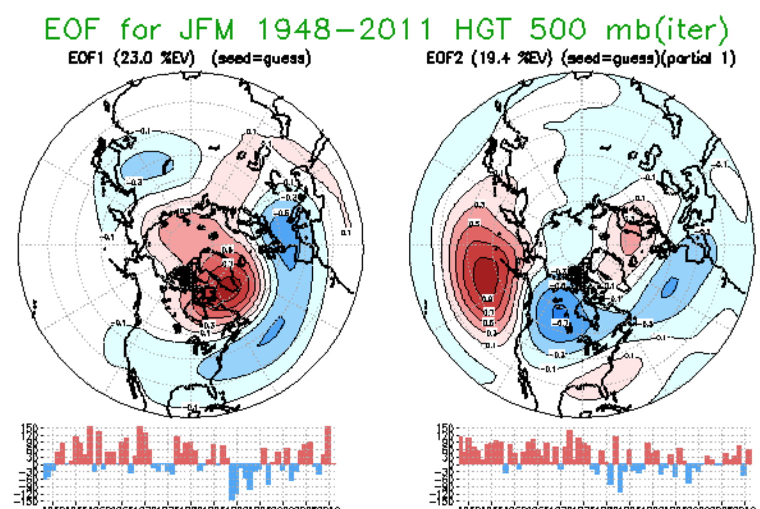
↓ Where we go to





The route to EOF1 from a another ridiculous guess field
Iterations 1-4

↓ Where we go to

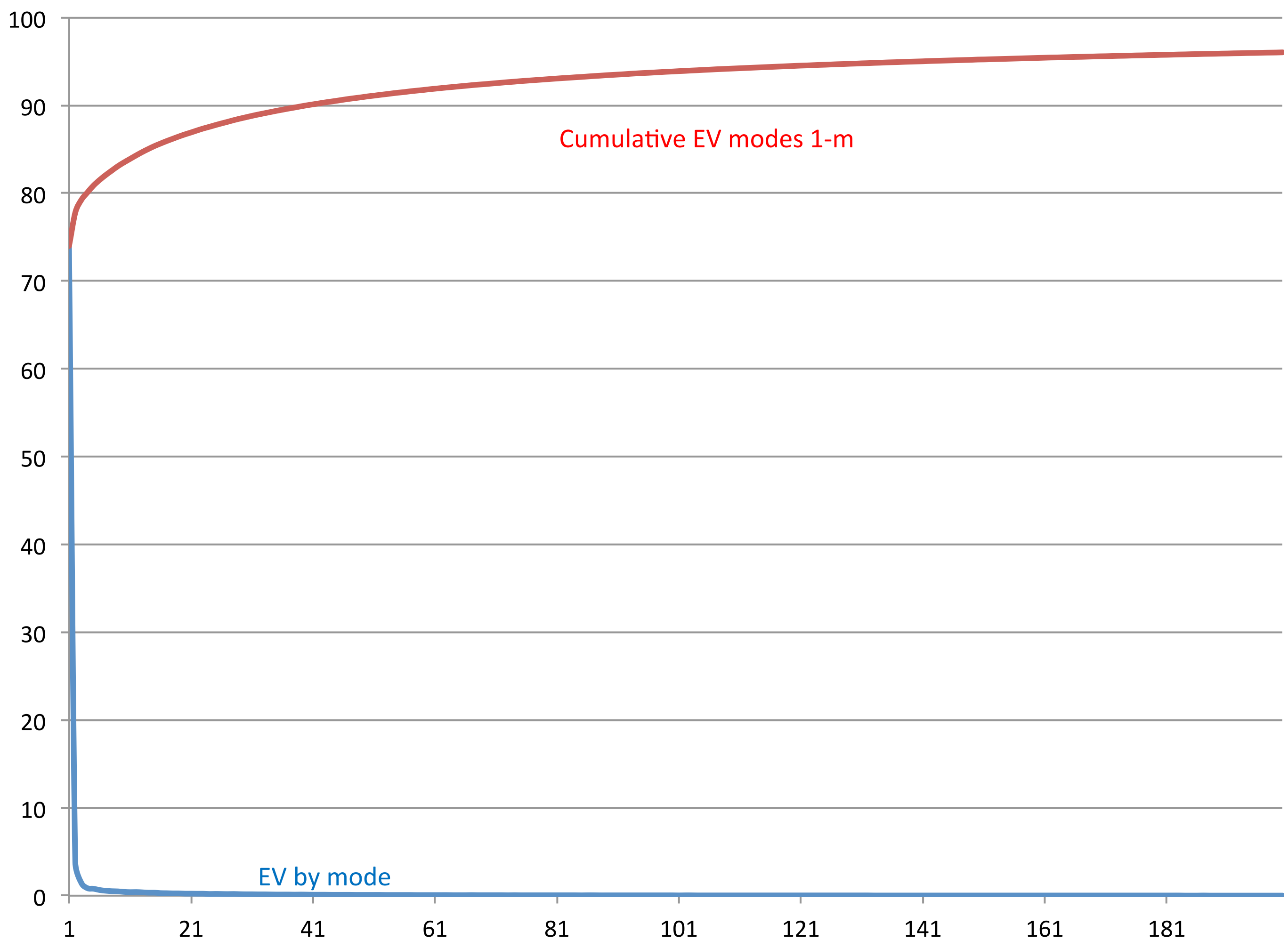


Savings in CPU?

- None (to write home about) for a small problem
- Iteration is factor 30 faster for ns=nt=4000
- When ns and nt go higher (10,000 or 100,000) the cov matrix method quickly becomes 'impossible' (depending on computer and smarts of programmer), while iteration is still possible (although not inexpensive).

Application to highres CFSR daily data

- Something entirely impossible via the cov matrix can actually be done via the iteration method.
- Daily T2m 1979-2010. nt=11688, ns=663552
- It works well, but, but and but



21	0.23
22	0.22
23	0.22
24	0.19
25	0.2
26	0.19
27	0.18
28	0.19
29	0.17
30	0.16
31	0.16
32	0.15
33	0.15
34	0.14
35	0.13
36	0.13
37	0.13
38	0.12
39	0.13
40	0.12
41	0.11
42	0.11
43	0.11
44	0.1

Application to highres CFSR daily data

- Something entirely impossible via the cov matrix can actually be done via the iteration method.
- Daily T2m 1979-2010. nt=11688, ns=663552
- It works well, but, but and but

- Iteration at low spatial res
- Convergence issues (or are these the real EOFs?)
- Pulling teeth on T2m. Even with 200 modes only 96%EV (residual error 1.15K). Compressionability of that data set is not good.

As eigenvalues become smaller the spectrum becomes flatter.

One has to iterate more and more to achieve convergence to the next EOF, when the next two (or more) eigenvalues are almost the same.

The smaller the added EV the longer the iteration. Not a desirable situation. Not (yet) a good approach when 'all' modes are needed.

Baldwin, Mark P., David B. Stephenson, Ian T. Jolliffe, 2009: Spatial Weighting and **<u>Iterative Projection Methods for EOFs</u>**. *J. Climate*, **22**, 234–243

"Another method involving iteration between a spatial pattern and a time series was proposed by Clint and Jennings (1970).

Van den Dool et al. (2000) used a similar approach to find the leading EOF beginning from the leading empirical orthogonal teleconnection (EOT) pattern.

Iteration between a time series and spatial pattern to calculate the leading EOF was ***<u>discovered</u>*** independently by G. Hegerl (2008, personal communication)."

Iteration is described in appendix of van den Dool, H. M., S. Saha, Å Johansson, 2000: Empirical Orthogonal Teleconnections. *J. Climate*, **13**, 1421–1435.

- Did not know or suspect that it was a ‘discovery’
- Did not know why it worked
- Did not realize that the iteration is particularly useful for a very large data set

Other conclusions/comments

- The method works for the same reason as the power method (applied to the cov matrix)
- The method boils down to calculating singular vectors (one by one) from the data matrix.
- The calculation is simple (shorter code)
- No issues of space-time reversal apply
- You can start with a guess in space or in time
- Does it always work? (give me an example where it does NOT)
- Double precision issues
- Iterative EOT?
- Swarming a bunch of guess fields simultaneously

Toumazou, Vincent, Jean-Francois Cretaux, 2001: Using a Lanczos Eigensolver in the Computation of Empirical Orthogonal Functions. *Mon. Wea. Rev.*, **129**, 1243–1250.

The computation of the singular values can be achieved through three different strategies in terms of linear algebra.

The first one is called the SVD strategy
and is based on the singular value decomposition (SVD) algorithm.

The two others (QR and Lanczos) strategies are based on the formulation
as an eigenvalue problem.

Acknowledge Michael Tippett

Appendix (adapted from Wiki)

- Formally, the singular value decomposition of an m×n real matrix M is a factorization of the form $M = U \Sigma V^T$,
- where U is an m×m real matrix, Σ is an m×n diagonal matrix with nonnegative real numbers on the diagonal, and V^T (the conjugate transpose of V) is an n×n matrix. The diagonal entries $\Sigma_{i,i}$ of Σ are known as the singular values of M. The m columns of U and the n columns of V are called the left singular vectors and right singular vectors of M, respectively.
- Singular value decomposition and eigendecomposition are closely related. Namely:
- The left singular vectors of M are eigenvectors of MM^T .
- The right singular vectors of M are eigenvectors of M^TM.
- The non-zero singular values of Σ are the square roots of the non-zero eigenvalues of M^TM or MM^T