Ensemble prediction and strategies for initialization: Tangent Linear and Adjoint Models, Singular Vectors, Lyapunov vectors

Eugenia Kalnay

Lecture 2

Alghero, May 2008

Elements of Ensemble Forecasting

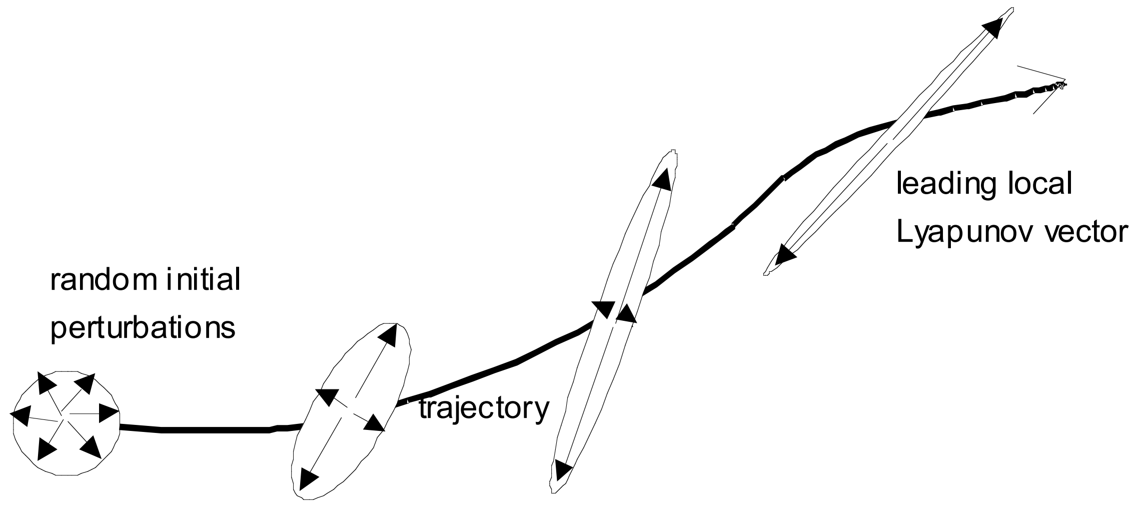
- It used to be that a single control forecast was integrated from the analysis (initial conditions)
- In ensemble forecasting several forecasts are run from slightly perturbed initial conditions (or with different models)
- The spread among ensemble members gives information about the forecast errors
- How to create slightly perturbed initial conditions?
- Basically
  - **Singular Vectors**
  - **Bred Vectors**
  - Ensembles of data assimilation (perturbed obs. EnKF)

In another fundamental paper, Lorenz (1965) introduced (without using their current names) the concepts of:
Tangent linear model, Adjoint model,
Singular vectors, and Lyapunov vectors
for a low order atmospheric model, and their consequences for ensemble forecasting.

He also introduced the concept of
"errors of the day": predictability is not constant:
It depends on the stability of the evolving atmospheric flow (the basic trajectory or reference state).

When there is an instability, all perturbations converge towards the fastest growing perturbation (leading Lyapunov Vector). The LLV is computed applying the linear tangent model on each perturbation of the nonlinear trajectory

Fig. 6.7: Schematic of how all perturbations will converge towards the leading Local Lyapunov Vector



Tangent linear model (TLM) and adjoint models

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}), \quad \mathbf{x} = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix}, \mathbf{F} = \begin{bmatrix} F_1 \\ . \\ . \\ F_n \end{bmatrix}$$

a nonlinear model discretized in space: ODE's

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \Delta t \mathbf{F}\left(\frac{\mathbf{x}^{n} + \mathbf{x}^{n+1}}{2}\right)$$

discretized in time: a set of nonlinear algebraic equations

"Running the model" is integrating the model from time *t_o* to time *t*

$$\mathbf{x}(t) = M(\mathbf{x}(t_0))$$

This is the nonlinear model

We add a small perturbation ***y**(t_o)* and neglect terms *O(**y**^{2})*

$$M(\mathbf{x}(t_0)+\mathbf{y}(t_0)) = M(\mathbf{x}(t_0)) + \frac{\partial M}{\partial \mathbf{x}}\mathbf{y}(t_0) + O(\mathbf{y}(t_0)^2) = \mathbf{x}(t) + \mathbf{y}(t) + O(\mathbf{y}(t_0)^2)$$

$$\mathbf{y}(t) = \mathbf{L}(t_0,t)\mathbf{y}(t_0)$$

This is the tangent linear model (TLM)

$$\mathbf{L}(t_0,t) = \frac{\partial M}{\partial \mathbf{x}}$$

L is an nxn matrix that depends on **x**(*t*) but not on **y**(*t*)

Euclidean norm of a vector: inner product of a vector with itself

$\|\mathbf{y}\|^2 = \mathbf{y}^T\mathbf{y} = <\mathbf{y},\mathbf{y}>$ Size of **y** (Euclidean norm)

Define adjoint of an operator **K**: $<\mathbf{x},\mathbf{K}\mathbf{y}> \equiv <\mathbf{K}^T\mathbf{x},\mathbf{y}>$

In this case of a model with real variables, the **adjoint** of the TLM is simply the **transpose** of the TLM $\mathbf{L}^T(t_0,t)$

Now separate the interval (t_o,t) into two successive intervals

$t_0 < t_1 < t$ $\mathbf{L}(t_0,t) = \mathbf{L}(t_1,t)\mathbf{L}(t_0,t_1)$

Transposing we get the adjoint: $\mathbf{L}^T(t_0,t) = \mathbf{L}^T(t_0,t_1)\mathbf{L}^T(t_1,t)$

the last segment is executed first!

The adjoint of the model can also be separated into single time steps, but they are executed backwards in time, starting from the last time step at t, and ending with the first time step at t_0.

For low order models the tangent linear model and its adjoint can be constructed by repeated integrations of the nonlinear model for small perturbations, as done by Lorenz (1965), and by Molteni and Palmer (1993) with a global quasi-geostrophic model.

For large NWP models this approach is too time consuming, and instead it is customary to develop the linear tangent and adjoint codes from the nonlinear model code following some rules discussed in Appendix B.

A very nice example of the TLM and ADJ code generation for the Lorenz (1963) model is given by Shu-Chih Yang:

Advanced data assimilation methods with evolving forecast error covariance: 4D-Var

Example of TLM and ADJ code with Lorenz (1963) model

Shu-Chih Yang (with EK)

Example with the Lorenz 3-variable model

Nonlinear model

$\mathbf{x}=[x_1,x_2,x_3]$

$$\frac{dx_1}{dt} = -px_1 + px_2$$

$$\frac{dx_2}{dt} = rx_1 - x_1x_3 - x_2$$

$$\frac{dx_3}{dt} = x_1x_2 - bx_3$$

Tangent linear model

$\delta\mathbf{x}=[\delta x_1, \delta x_2, \delta x_3]$

$$\mathbf{L} = \frac{\partial M}{\partial \mathbf{x}} = \frac{\partial M}{\partial x_i} = \begin{bmatrix} -p & p & 0 \\ r - \mathbf{x_3} & -1 & -\mathbf{x_1} \\ \mathbf{x_2} & \mathbf{x_1} & -b \end{bmatrix}$$

Adjoint model

$\delta\mathbf{x}^*=[\delta x^*_1, \delta x^*_2, \delta x^*_3]$

$$\mathbf{L}^T = \left[\frac{\partial M}{\partial x_i}\right]^T = \begin{bmatrix} -p & r - \mathbf{x_3} & \mathbf{x_2} \\ p & -1 & \mathbf{x_1} \\ 0 & -\mathbf{x_1} & -b \end{bmatrix}$$

- The background state is needed in both $\mathbf{L}$ and $\mathbf{L}^T$ (need to save the model trajectory)
- In a complex NWP model, it is impossible to write explicitly this matrix form

Example of tangent linear and adjoint codes (1)

use forward scheme to integrate in time

In tangent linear model

$$\frac{\delta x_3(t+\Delta t)-\delta x_3(t)}{\Delta t} = x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t), \text{ or}$$

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$ forward in time

We will see that in the adjoint model the above line becomes

$$\delta x_3^*(t) = \delta x_3^*(t) + (1-b\Delta t)\delta x_3^*(t+\Delta t)$$
$$\delta x_2^*(t) = \delta x_2^*(t) + (x_1(t)\Delta t)\delta x_3^*(t+\Delta t)$$
$$\delta x_1^*(t) = \delta x_1^*(t) + (x_2(t)\Delta t)\delta x_3^*(t+\Delta t)$$
$$\delta x_3^*(t+\Delta t) = 0$$

backward in time

* Try an example in Appendix B (B.1.15)

Example of tangent linear and adjoint codes (2)

use forward scheme to integrate in time

Tangent linear model,

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

forward in time

$$\begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix} = \begin{bmatrix} 0 & x_2(t)\Delta t & x_1(t)\Delta t & (1-b\Delta t) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix}$$

We have to write for each statement all the “active” variables.

Then we transpose it to get the adjoint model

Example of tangent linear and adjoint codes (3)

Tangent linear model,

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

forward in time

$$\begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix} = \begin{bmatrix} 0 & x_2(t)\Delta t & x_1(t)\Delta t & (1-b\Delta t) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix}$$

$$\begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x_2(t)\Delta t & 1 & 0 & 0 \\ x_1(t)\Delta t & 0 & 1 & 0 \\ (1-b\Delta t) & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix}$$

Adjoint model: transpose of the linear tangent, backward in time

Execute in reverse order

Example of tangent linear and adjoint codes (4)

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

Adjoint model: transpose of the linear tangent, backward in time

Execute in reverse order

$$\begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x_2(t)\Delta t & 1 & 0 & 0 \\ x_1(t)\Delta t & 0 & 1 & 0 \\ (1-b\Delta t) & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix}$$

In adjoint model the line above becomes

$$\delta x_3^*(t) = \delta x_3^*(t) + (1-b\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_2^*(t) = \delta x_2^*(t) + (x_1(t)\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_1^*(t) = \delta x_1^*(t) + (x_2(t)\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_3^*(t+\Delta t) = 0$$

backward in time

There are automatic compilers to create linear tangent and adjoint models (e.g., TAMC)

- It is still a long and painful process…
- Creating and maintaining an adjoint model is the main disadvantage of 4D-Var
- On the other hand, the adjoint provides tremendous possibilities:
  - 4D-Var
  - Sensitivity of the results to any parameter or initial conditions (e.g., sensitivity of forecasts to observations, Langland and Baker 2004)
  - And many others
  - Ensemble Kalman Filter allows doing the with the nonlinear model ensemble, without the need for an adjoint

Singular Vectors: the theory is too long, so we'll just give a basic idea (see section 6.3 of book)

$$\mathbf{y}(t_1) = \mathbf{L}(t_0, t_1)\mathbf{y}(t_0)$$

Remember the TLM **L**(t_o, t_1) evolves the perturbation from t_o to t_1

$$\mathbf{U}^{\mathrm{T}}\mathbf{L}\mathbf{V} = \mathbf{S}$$

Singular Value Decomposition (SVD)

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & . & 0 \\ 0 & \sigma_2 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \sigma_n \end{bmatrix}$$

S is a diagonal matrix whose elements are the **singular values** of **L.**

Singular Vectors: the theory is too long, so we'll just give a basic idea

$$\mathbf{y}(t_1) = \mathbf{L}(t_0, t_1)\mathbf{y}(t_0)$$

Remember the TLM **L**(t$_o$, t$_{1)}$ evolves the perturbation from t$_o$ to t$_1$

$$\mathbf{U}^{\mathbf{T}}\mathbf{L}\mathbf{V} = \mathbf{S}$$

Singular Value Decomposition (SVD)

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & . & 0 \\ 0 & \sigma_2 & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \sigma_n \end{bmatrix}$$

S is a diagonal matrix whose elements are the **singular values** of **L.**

and $\mathbf{U}\mathbf{U}^T = \mathbf{I}; \mathbf{V}\mathbf{V}^T = \mathbf{I}$

Left multiply by **U**:

$$\mathbf{LV} = \mathbf{US}, \quad \text{i.e., } \mathbf{L}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = (\sigma_1 \mathbf{u}_1, \ldots, \sigma_n \mathbf{u}_n)$$

where $\mathbf{v}_i$ and $\mathbf{u}_i$ are the vector columns of **V** and **U**

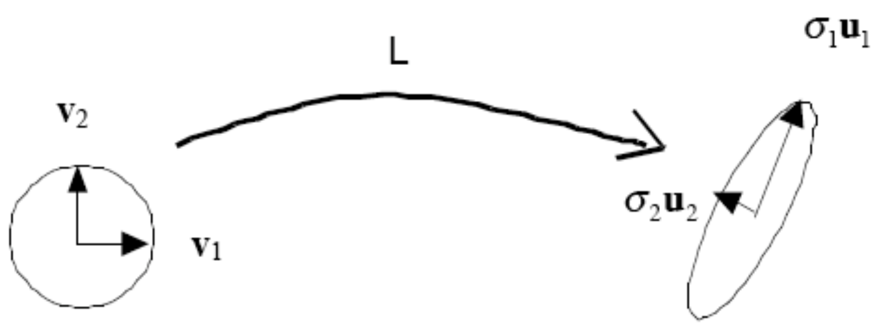
$\mathbf{v}_i$ are the initial singular vectors

$\mathbf{u}_i$ are the final singular vectors

$$\mathbf{L}\mathbf{v}_i = \sigma_i \mathbf{u}_i$$

The initial SV gets multiplied by the singular value when **L** is applied forward in time

Fig. 6.3: Schematic of the application of the TLM to a sphere of perturbations of size 1 for a given interval (t_0,t_1).

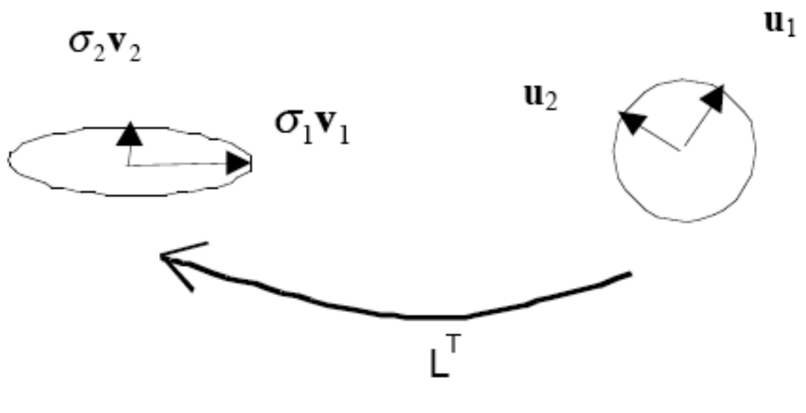


Right multiply by $\mathbf{V}^T$: $\mathbf{U}^{\mathbf{T}}\mathbf{L} = \mathbf{S}\mathbf{V}^{\mathbf{T}}$ and transposing,

$\mathbf{L}^{\mathbf{T}}\mathbf{U} = \mathbf{V}\mathbf{S}, \quad i.e., \quad \mathbf{L}^T(\mathbf{u}_1,...,\mathbf{u}_n) = (\sigma_1\mathbf{v}_1,...,\sigma_n\mathbf{v}_n)$ so that

$\mathbf{L}^T\mathbf{u}_i = \sigma_i\mathbf{v}_i$ The final SV gets multiplied by the singular value when $\mathbf{L}^T$ is applied backward in time

Fig. 6.4: Schematic of the application of the adjoint of the TLM to a sphere of perturbations of size 1 at the final time.

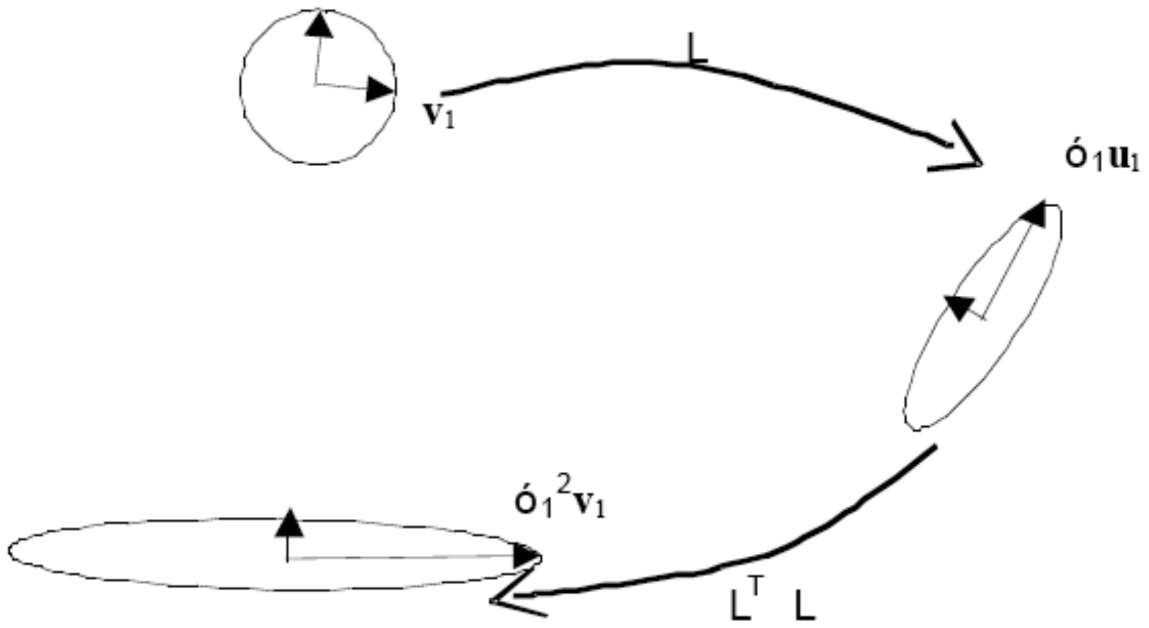


From these we obtain

$$\mathbf{L}^T\mathbf{L}\mathbf{v}_i = \sigma_i\mathbf{L}^T\mathbf{u}_i = \sigma_i{}^2\mathbf{v}_i$$

so that $\mathbf{v}_i$ are the eigenvectors of $\mathbf{L}^T\mathbf{L}$

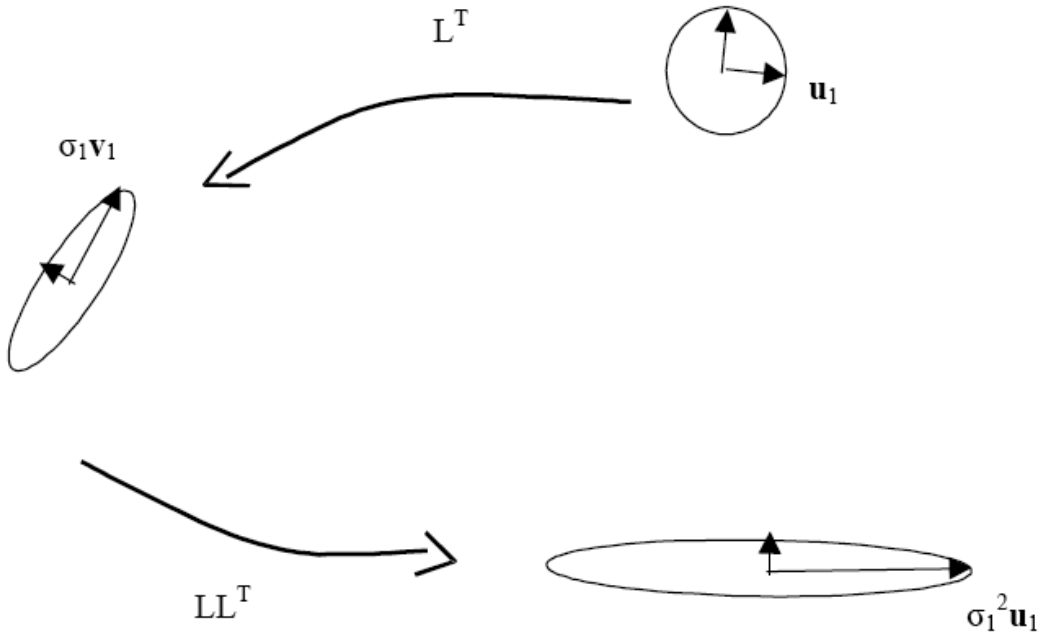
Fig. 6.5: Schematic of the application of the TLM forward in time followed by the adjoint of the TLM to a sphere of perturbations of size 1 at the initial time.



Conversely, the final SVs $\mathbf{u}_i$ are the eigenvectors of $\mathbf{LL}^T$

$$\mathbf{LL}^T\mathbf{u}_i = \sigma_i\mathbf{Lv}_i = \sigma_i^{\,2}\mathbf{u}_i$$

Fig. 6.6: Schematic of the application of the adjoint of the TLM backward in time followed by the TLM forward to a sphere of perturbations of size 1 at the final time.



SV summary and extra properties

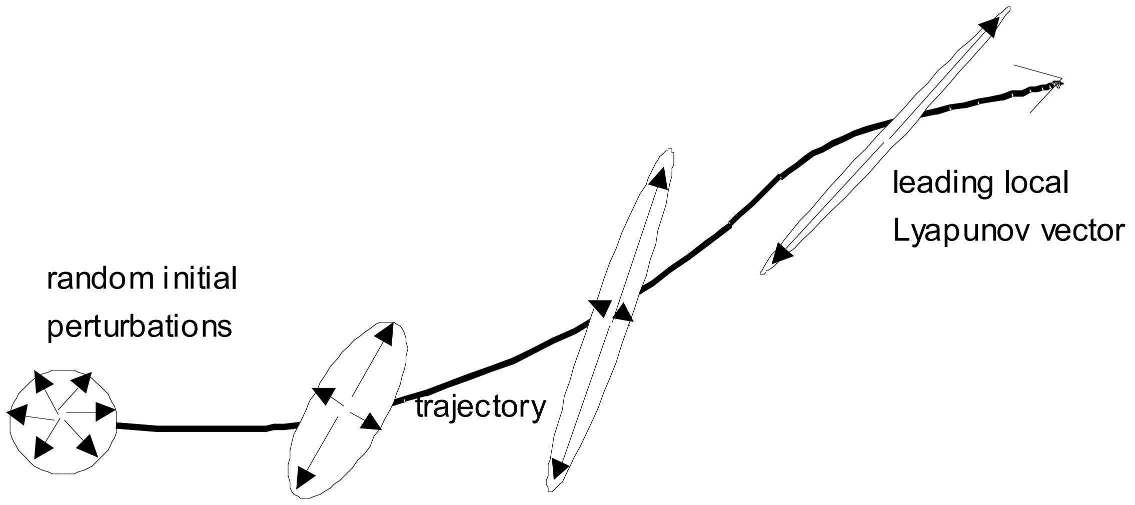
- In order to get SVs we need the TLM and the ADJ models.
- The leading SVs are obtained by the Lanczos algorithm (expensive).
- One can define an initial and a final norm (size), this gives flexibility and arbitrariness$^{(1)}$.
- The leading initial SV is the vector that will grow fastest (starting with the smallest initial norm and ending with the largest final norm).
- The leading SVs grow initially faster than the Lyapunov vectors, but at the end of the period, they look like LVs (and bred vectors~LVs).
- The initial SVs are *very* sensitive to the norm used. The final SVs look like LVs~BVs

Jon Ahlquist theorem

- One can define an initial and a final norm (size), this gives flexibility and arbitrariness$^{(1)}$.
- Given a linear operator ***L,*** a set of arbitrary vectors $\boldsymbol{x}_i$ and a set of arbitrary nonnegative numbers $\boldsymbol{s}_i$ arranged in decreasing order, Ahlquist (2000) showed how to define a norm such that $\boldsymbol{s}_i$ and the $\boldsymbol{x}_i$ are the ith singular value and singular vector of ***L***.
- “Because anything not in the null space can be a singular vector, even the leading singular vector, one cannot assign a physical meaning to a singular vector simply because it is a singular vector. Any physical meaning must come from an additional aspect of the problem. Said in another way, nature evolves from initial conditions without knowing which inner products and norms the user wants to use”

Comparison of SV and BV (LV) for a QG model

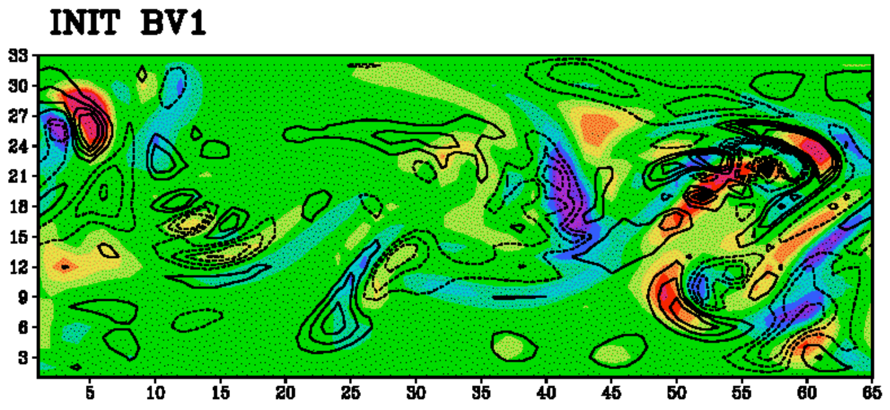
Fig. 6.7: Schematic of how all perturbations will converge towards the leading Local Lyapunov Vector

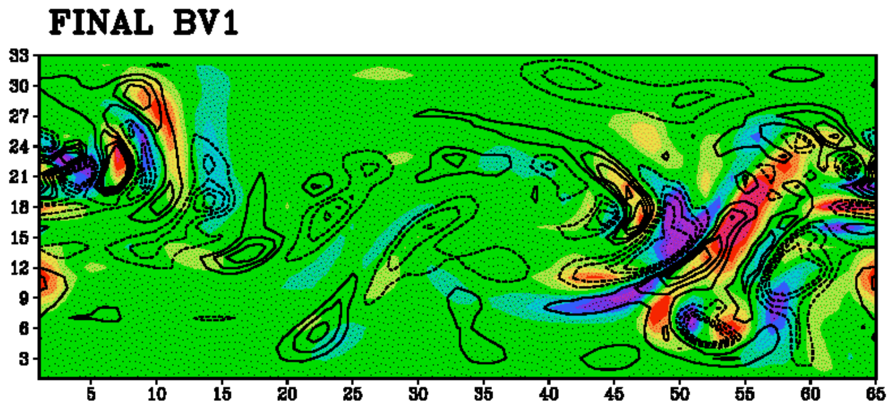


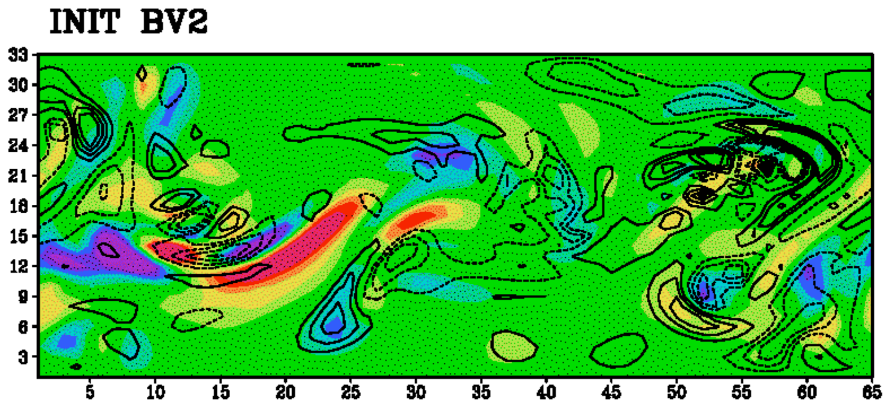
All perturbations (including singular vectors) end up looking like local Lyapunov vectors (i.e., like Bred Vectors).
These are the local instabilities that make forecast errors grow.

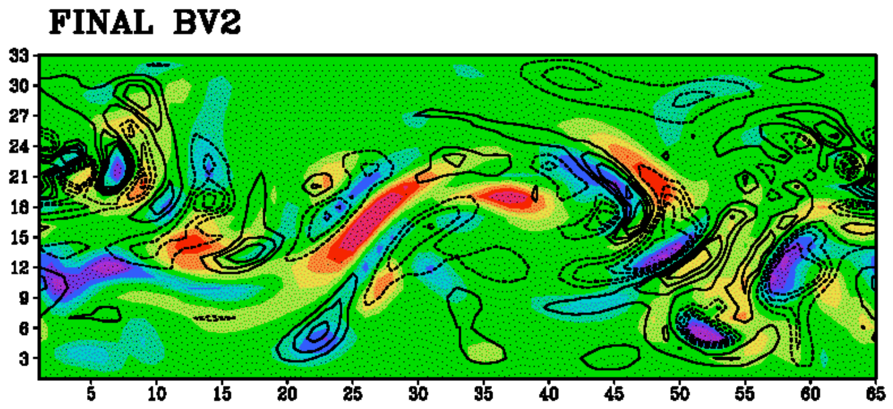
Two initial and final BV (24hr)

contours: forecast errors, colors: BVs





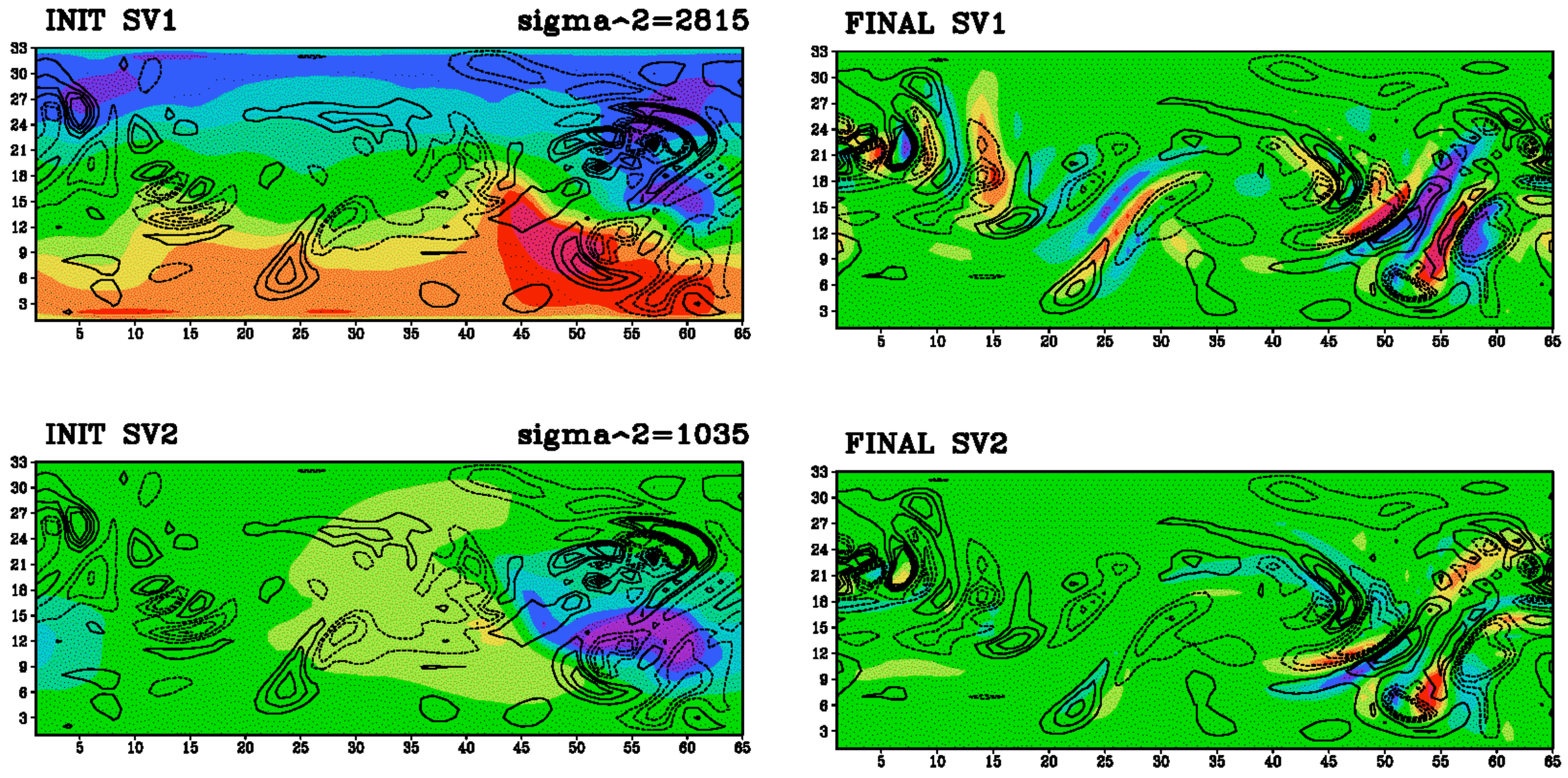




Note that the BV (colors) have shapes similar to the forecast errors (contours)

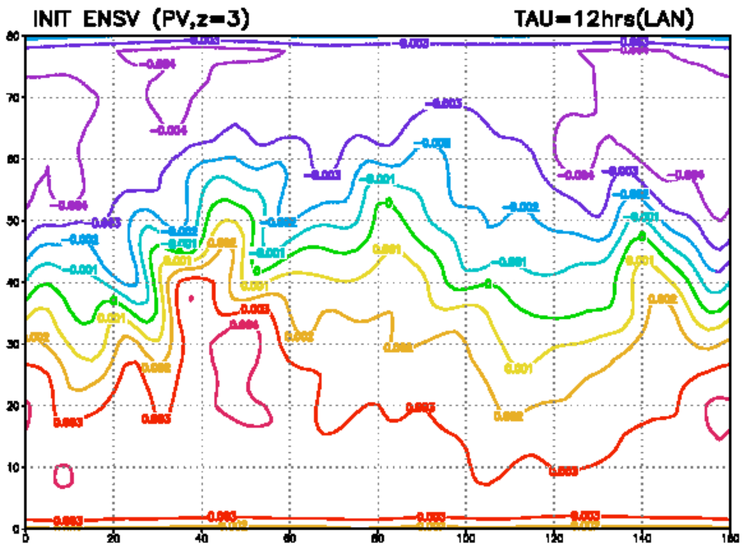
Two initial and final SV (24hr, vorticity norm)

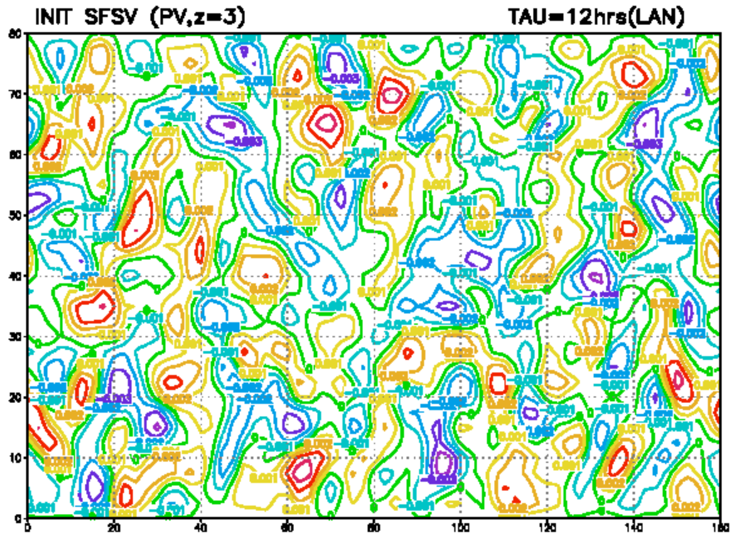
contours: forecast errors, colors: SVs



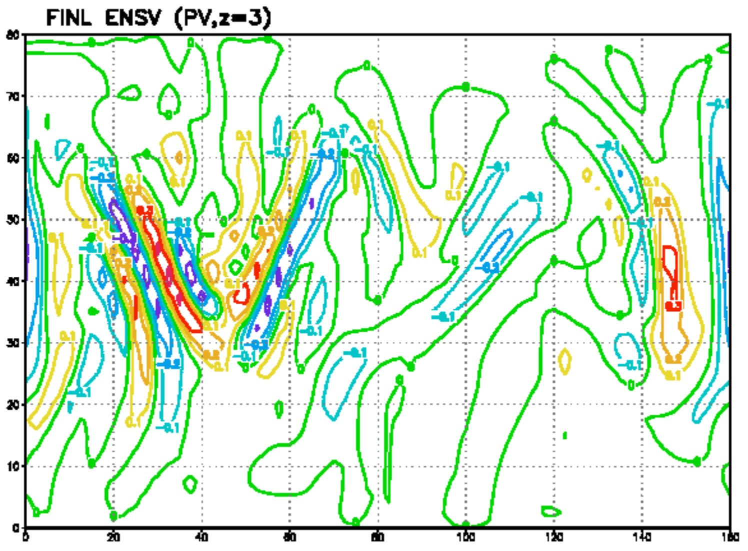
With an enstrophy norm, the initial SVs have large scales (low vorticity). By the end of the"optimization" interval, the final SVs look like BVs (and LVs)

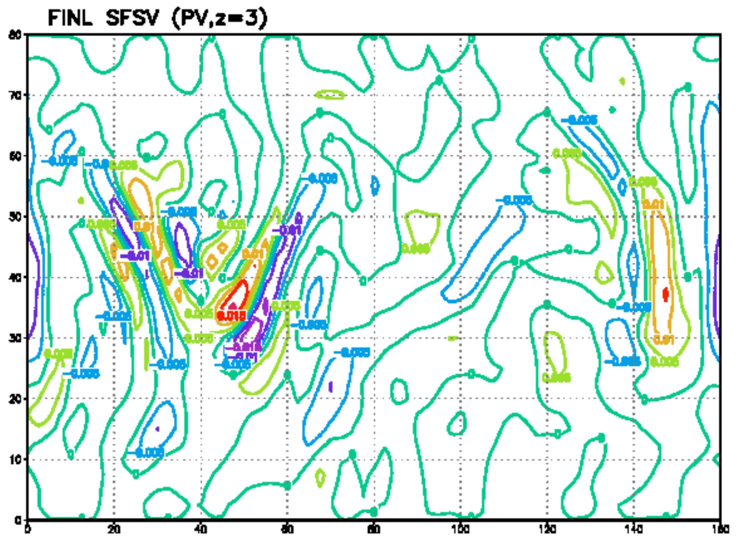
Example of initial and final singular vectors using enstrophy/stream function norms (QG model, 12 hour optimization, courtesy of Shu-Chih Yang)





The initial SVs are very sensitive to the norm





The final SVs look like bred vectors (or Lyapunov vectors)