

- Full resolution example :
 - Background state and observations
 - B matrix monitoring
 - Innovation vs. guess
 - Cost function
- Changing the resolution :
 - illustration of the change on the background state.
 - Modifications on the guess.
 - Cost function.
- Short conclusion

Parameters of the example

The aim of this presentation is to monitor the output of the code multi 2D and spot the potential problems and reassuring things.

The parameters of the code are fixed to the following :

- outer iterations : 4, inner iterations : 6, no stop criterion.
- no LMP are used.
- $\sigma_{var}^b=0.1$, $L_b = 0.1$ (correlation lengthscale), minimum spectral variance : 10^{-5} .
- $n_{obs} = 2000$, $\sigma^o = 0.01$.
- others : transitive interpolation : True / Projective Bmatrix : True / no orthogonality test.
- method used to compute the increments at outer loop level : "theoretical" (using B^{-1}) – see the notes of Benjamin.

Full resolution example.

The resolution is fixed at $n_x \times n_y = 101 \times 101$ for the example. The problem is modeled in a square domain of size 1, divided in $n_x \times n_y$ pixels.

Full resolution : Background state and Observations

The background and observations generated randomly seems to be good.

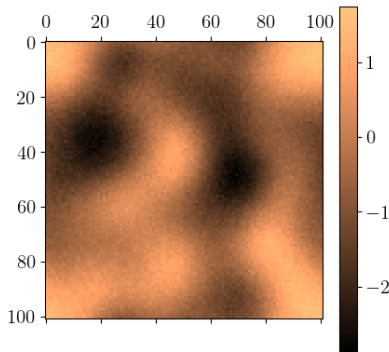


FIGURE – background state x^b .

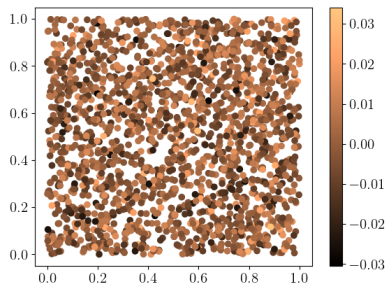


FIGURE – Observations.

Full resolution : B matrix

The B matrix is OK : The correlation lengthscale modifies the figures as expected (not shown), and one can see that the left panel shows correlations consistent with the periodic domain in which we are working.

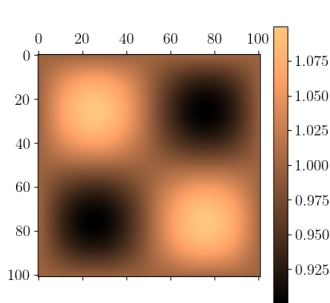


FIGURE – Background error field σ^b .

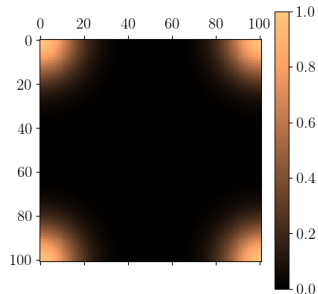


FIGURE – B applied to a "dirac" : $(B(1, 0, 0, \dots, 0)^T)$

Full resolution : guess

With lanczos algorithm :

The problem have been fixed : The first guess is now the background state.

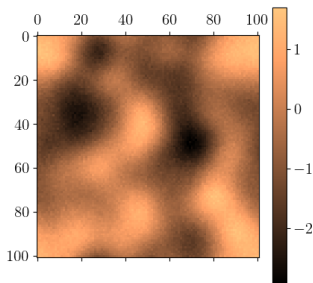


FIGURE – guess (outer iteration 1)

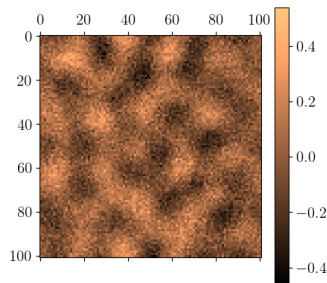


FIGURE – guess (outer iteration 2)

Full resolution : guess

With lanczos algorithm :

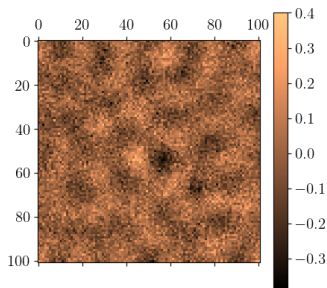


FIGURE – guess (outer iteration 3)

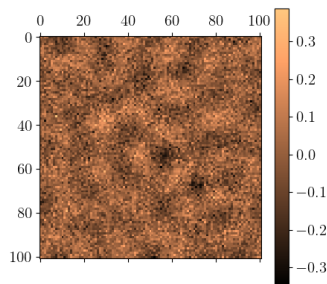


FIGURE – guess (outer iteration 4)

Full resolution : guess

With PlanczosIF algorithm (same results as using lanczos) :

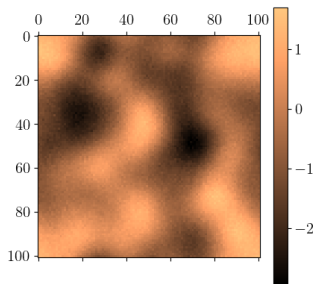


FIGURE – guess (outer iteration 1)

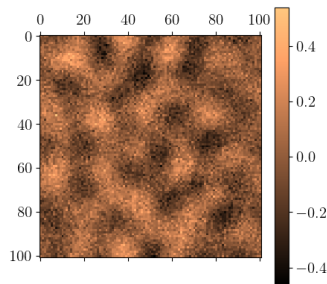


FIGURE – guess (outer iteration 2)

Full resolution : guess

With PlanczosIF algorithm (same results as using lanczos) :

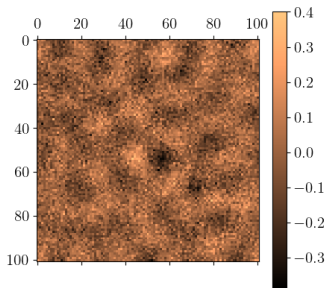


FIGURE – guess (outer iteration 3)

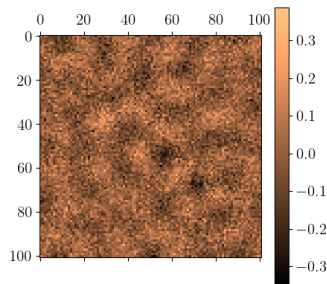


FIGURE – guess (outer iteration 4)

Full resolution : mapping the guess to the obs : Hx^g

lanczos example (identical with PlanczosIF) :

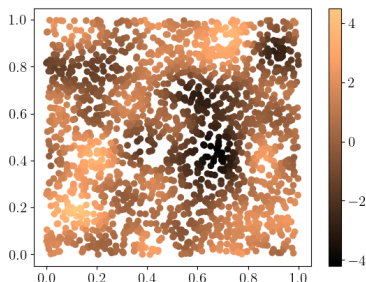


FIGURE – Hx^g (outer iteration 1)

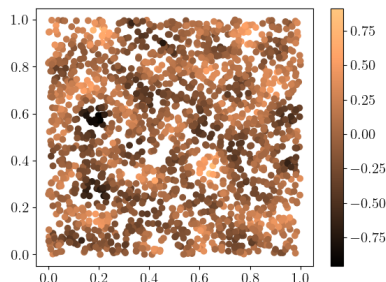


FIGURE – Hx^g (outer iteration 2)

Full resolution : mapping the guess to the obs : Hx^g

Hx^g seems to be close to the background state at the beginning and closer to the obs at the end (but with a higher variance than the observations?).

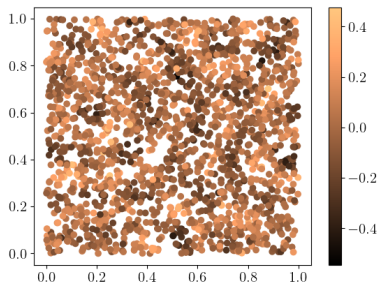


FIGURE – Hx^g (outer iteration 3)

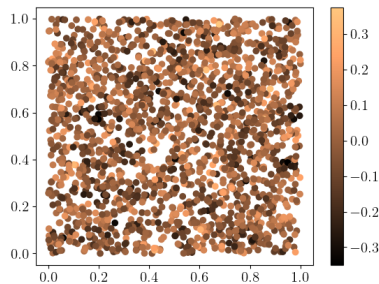


FIGURE – Hx^g (outer iteration 4)

Full resolution : innovations

lanczos example (identical with PlanczosIF) :

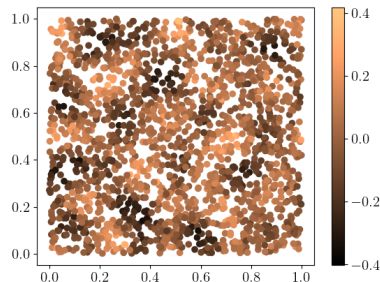
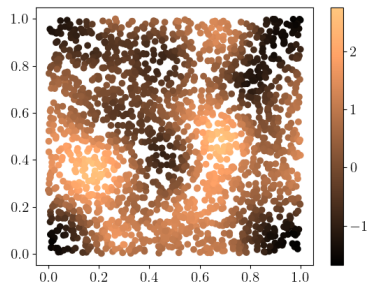


FIGURE – innovation (outer iteration 1) FIGURE – innovation (outer iteration 2)

Are the values of the innovation small enough ?

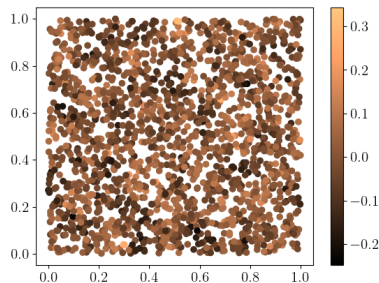
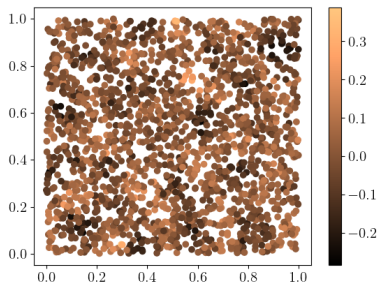


FIGURE – innovation (outer iteration 3) FIGURE – innovation (outer iteration 4)

Full resolution : Cost function

In the following slides, we compare the cost functions obtained with the different methods ("theoretical", "standard", and "alternative") with both lanczos algorithms in control space and PlanczosIF in model space.

The code allows to retrieve the non linear cost functions (departure from the background J_b^{nl} , from the observations J_o^{nl} and there sum J^{nl}) as well as the linearized ones J_b , J_o , and J . The blue dashed lines represent the

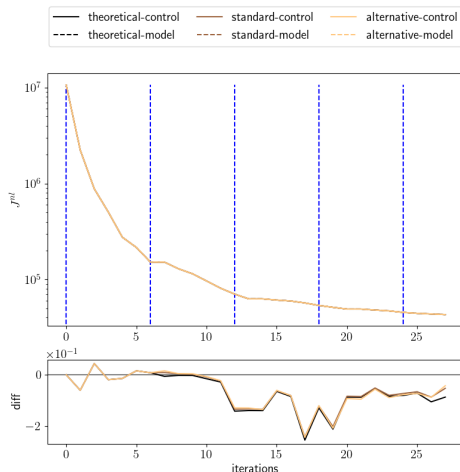
outer iterations. The lower panel illustrates the difference between the result in control space or model space for each method.

Full resolution : Cost function

non linear total cost function
 $J^{nl} = J_o^{nl} + J_b^{nl}$.

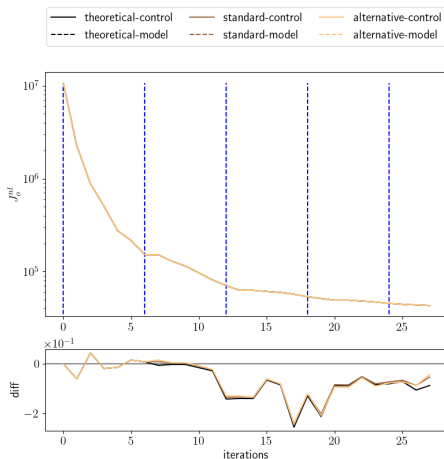
1) The code converges with all the methods.

2) The differences between the methods seems small (as well as the difference between the two algorithms for each methods – lower panel).



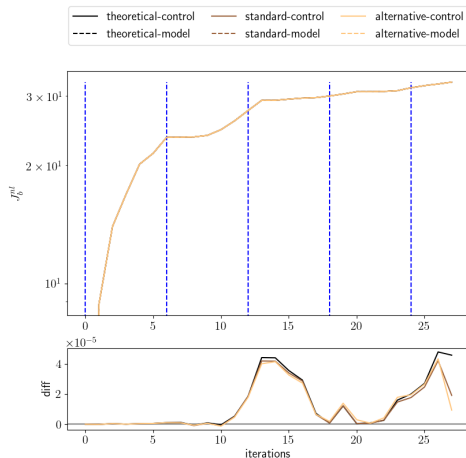
Full resolution : Cost function

The observational part of the cost function seems to be good...



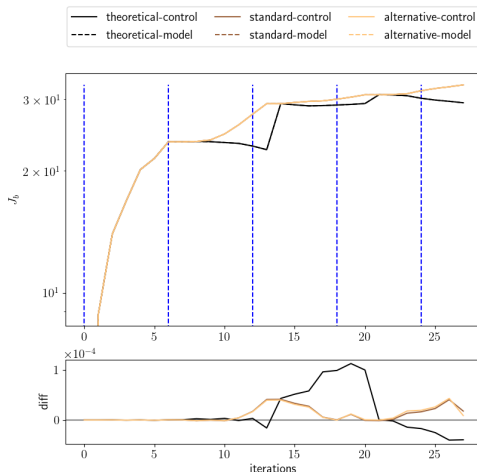
Full resolution : Cost function

... as well as the background contribution.



Full resolution : Cost function

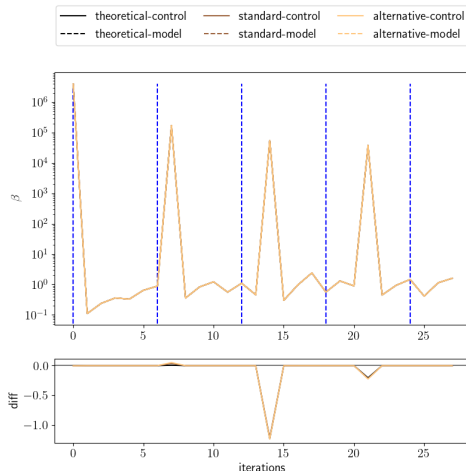
There is a small difference between the linearized cost function and the non-linear one, as expected, but stronger for the "theoretical" method (?).



Full resolution : Cost function

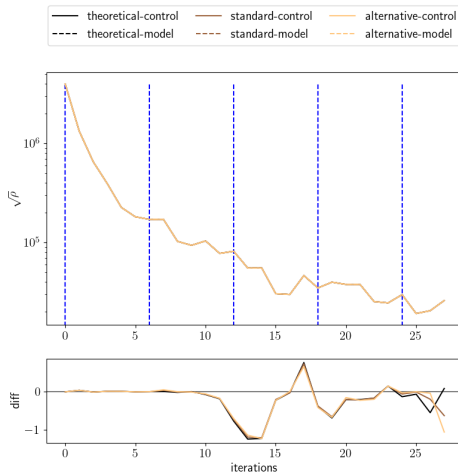
β is the ratio between the norm of the residue at inner iterations i and $i+1$ (see Benjamin's notes).

I think there is a problem in the inner iterations or with the storing of β because we can see the peak is delayed of one inner iteration at each outer iterations. (I will check that issue, it might also be in my plotting code).



Full resolution : Cost function

ρ is the norm of the residue
and seems to converge well.



Differences between the methods

There are small differences between the guesses obtained with the "alternative" and "theoretical" methods as illustrated with this example, whereas there are not between "standard" and "theoretical" methods.

The differences appear after the firsts outer loops.

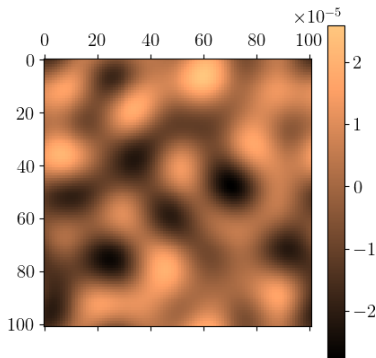


FIGURE – $x_{alt}^g - x_{th}^g$ (lanczos example)

Canging the resolution

In the following, we choose the resolutions for the outer iterations to be :

$$\text{outer 1 : } n_x \times n_y = 101 \times 101,$$

$$\text{outer 2 : } n_x \times n_y = 121 \times 121,$$

$$\text{outer 3 : } n_x \times n_y = 151 \times 151,$$

$$\text{outer 4 : } n_x \times n_y = 201 \times 201.$$

Changing resolution : x^b

The change of resolution is OK for the background (idem for the B matrix – not shown).

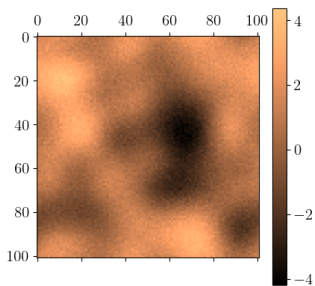


FIGURE – x^b (outer iteration 1)

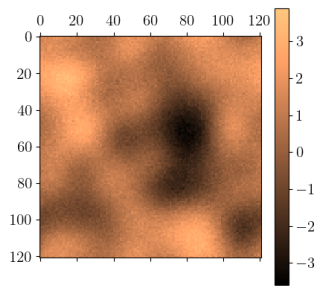


FIGURE – x^b (outer iteration 2)

Changing resolution : x^b

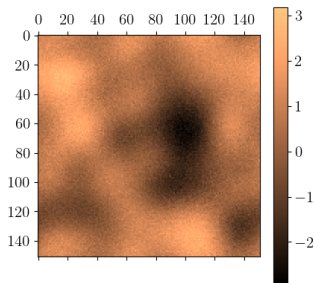


FIGURE – x^b (outer iteration 3)

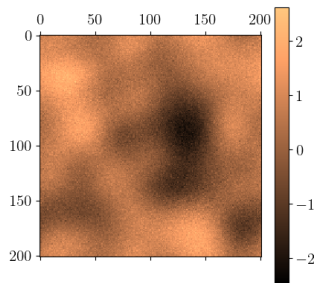


FIGURE – x^b (outer iteration 4)

Changing resolution : guess

With lanczos algorithm (identical with PlanczosIF) : The problem is fixed

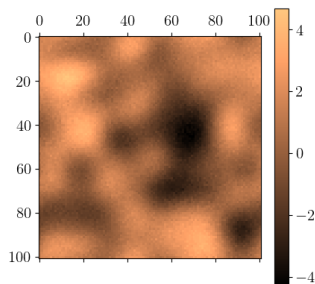


FIGURE – guess (outer iteration 1)

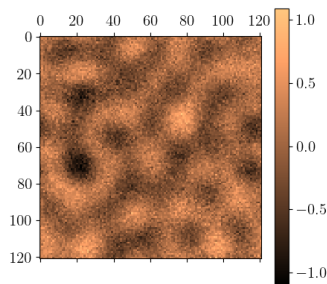


FIGURE – guess (outer iteration 2)

Changing resolution : guess

With lanczos algorithm :

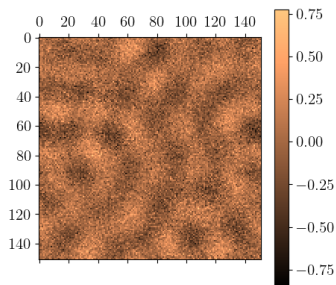


FIGURE – guess (outer iteration 3)

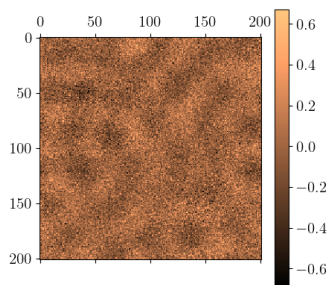
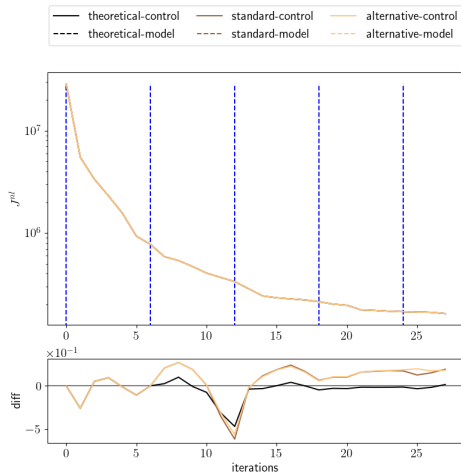


FIGURE – guess (outer iteration 4)

Changing resolution : cost function

The convergence is correct, also for J_o and J_b (in the current state of the code, we do not add new observations at outer iterations level, so there is no artificial deteriorating of the cost function at each outer iterations).



Short conclusion :

- The code seems to converge to a correct solution according to the innovation, cost function and residual norm, both in full resolution or changing it.
- There is no significant difference between the three methods or between the algorithms used in both scenarios for the resolution.
- There is a small difference between non-linear and linearized cost functions.
- There are small differences in the guess appearing after the first outer loops between the alternative and theoretical methods, but not between the standard and theoretical methods.
- Need to check beta (outer loops problem ?)