Master thesis : Coupled data assimilation on a low-order ocean-atmosphere model

Maxime Tondeur

June 2017

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisors Dr Alberto Carrassi from the NERSC, Dr Stephane Vannitsem from RMI and Dr Marc Bocquet from Ponts ParisTech. This was my first professional experience and they make sure that everything goes well. They guide me during the internships. They give me some scientific and technical skills. I thank them for their patience and for their listening. Step by step, I try to progress in my autonomy.

I share with them great human experience too. They give me a good overview of what is science. Thanks to this internship I meet great people. I meet around twenty nationalities and I traveled in three (and soon four) countries. I am very thankful for these human experiences.

I thank the NERSC for the welcoming. The teams of Data Assimilation welcomes me, answered always to my questions and integrate me as a family. I thank Laurent Bertino, Patrick N. Raanes, Matthias Rabatel, Colin Grudzien and Abhishek Shah. I thank the members of the other team, François Counillon, Madlen Kimmritz, Yiguo Wang and Jiping Xie. I thank obviously to everybody else I met in NERSC for their welcoming.

In RMI, I thank Jonathan Demaeyer and Lesley De Cruz for their advice and their welcoming in Brussels. They help me for precise questions, and they share with me their experiences. I thank to everybody else in RMI for their welcoming too.

Finally I thank my teacher in my school Mines ParisTech Hans Wackernagel for initiating the contacts for the internship. It makes that possible. I thank him and the team of Geostatistics for their support through my scolarity.

I leave the academic research with some sadness. I have matured my professional project and this internship was a great step of it.

Contents

List of figures

Introduction

My research internship has been done half in the Nansen Center in Bergen in Norway with Dr Alberto Carrassi and half in the Royal Meteorological Institute at Brussels in Belgium with Dr Stephane Vannitsem. The goal of the internship is to study the behavior of data assimilation in multiscale coupled (ocean-atmosphere) systems.

Data assimilation is a technique to combine the forecasts of a model and observations [1]. It helps to reduce drastically the forecast error in meteorology. Approximate solutions are developed with the Ensemble Kalman Filter (EnKF - [2]) which combines a Monte Carlo approach with linearity and gaussianity hypothesis. In the present work Ensemble Kalman Filter [3] will be used that allows to provide state dependent error covariance matrix. Yet the coupling is not present in the current models. The main difficulty in coupled model is the time-scale differences between ocean and atmosphere, here ocean quantities evolve slower than the atmosphere ones. The goal of the internship is to study the properties of a data assimilation system when dealing with coupled multiscale systems. A Lorenz 63 model [4] with two compartments and MAOOAM, [5] a quasi-geostrophic model which describes temperatures and streamfunctions of the ocean and the atmosphere expanded on a Fourier basis are used. Stability studies are first done to study the dynamical properties of these models for different coupling physical parameters to test several regimes. Then a benchmark of data assimilation experiments is chosen by changing the size of the ensemble or the observations for example. The goal is to try to relate the dynamical properties to the performances of the data assimilation.

This report is organized in three parts. The first part describes the coupled atmosphere ocean model MAOOAM . It describes the equations, the physical parameters, some runs and the implementation. The second part study the stability properties of the model MAOAM and Lorenz 63 with 2 coupled comparitments. It describes the theory and the computation of Lyapunov exponents, backward Lyapunov vectors, forward Lyapunov vectors and some covariant Lyapunov vectors. There are some tests of the code are performed, and results on MAOOAM and Lorenz 63 are provided. Finally the third part exposes the data assimilation results on MAOOAM. At the appendix, there is 30 pages of documention of the python version of MAOOAM.

1 Ocean-atmosphere model

1.1 Atmosphere model : two layers quasi-geostrophic equation

To work on the atmosphere and ocean equations, we refer to the course of Stéphane Vannitsem [6] and the book of Geoffrey K. Vallis [7].

The model is a coupled atmosphere-ocean model. It is composed of a two layer quasi-geostrophic (QG) atmosphere, coupled both thermally and mechanically to a QG shallow-water ocean layer, in the β-plane approximation. The model has been published in De Cruz, Demaeyer and Vannitsem 2016 [8]. The

atmospheric component is an extension of the QG model, first developed by Charney and Strauss [9] and further refined by Reinhold and Pierrehumbert [10].

1.1.1 Referential and coordinates

The referential is tied to the earth surface, it is the terrestrial referential. It is not Galilean because it is in rotation in the geocentric referential. The coordinates are Cartesian. Forces applied on the atmosphere are superficial and volumetric forces. Superficial force is the gravity. Volumetric forces are hydrostatic pressure and friction.

1.1.2 Primitive equations

1.1.2.1 Momentum equations

In the absolute geocentric referential, the equation of motion in the atmosphere is :

$$\boxed{\frac{d\boldsymbol{v}}{dt} = \boldsymbol{g} - \frac{1}{\rho}\boldsymbol{\nabla} p - \frac{1}{\rho}\boldsymbol{\nabla}\Sigma} \tag{1.1}$$

where $\boldsymbol{v}$ is the speed, $\boldsymbol{g}$ the gravity vector, ρ the density, p the pressure and Σ the friction tensor.
We note the geopotential :

$$\nabla\phi = -\boldsymbol{g}^* = (-\boldsymbol{g} + \Omega^2\boldsymbol{R}) \tag{1.2}$$

with $\Omega = 7 * 10^{-5}s^{-1}$ the rotational velocity of the earth and $\boldsymbol{R}$ the earth radius.

We get in the terrestrial referential due to its rotation :

$$\boxed{\frac{d\boldsymbol{v}}{dt} = -2\boldsymbol{\Omega} \times \boldsymbol{v} - \boldsymbol{\nabla}\phi - \frac{1}{\rho}\boldsymbol{\nabla} p - \frac{1}{\rho}\boldsymbol{\nabla\Sigma}} \tag{1.3}$$

1.1.2.2 Continuity equation

The continuity equation is :

$$\boxed{\frac{1}{\rho}\frac{d\rho}{dt} = -\boldsymbol{\nabla}\cdot\boldsymbol{v}} \tag{1.4}$$

with $\boldsymbol{v}$ the speed in the terretrial referential.

1.1.2.3 Ideal gas law

Primitive ideal gas law is :

$$\boxed{p = \rho RT} \tag{1.5}$$

where $R = 8.3144598 Jmol^{-1}K^{-1}$ is the ideal gas constant and T the temperature.

1.1.2.4 Thermodynamics equation

With the first principle of thermodynamics, the continuity equation and the state equation we get :

$$\boxed{c_p \frac{dT}{dt} - \frac{1}{\rho}\frac{d}{dt}p = Q} \tag{1.6}$$

where c_p is the atmosphere massic heat capacity at constant pressure and Q is the heat fluxes.

We have to compute Q. We have to do an energy balance. The atmosphere receives long-wave radiative and heat fluxes from the ocean, short-wave radiative fluxes from the sun and heat from the atmosphere to space and ocean.

The radiative fluxes from the ocean expresses thanks to Newton relaxation law :

$$Q_{ocean,conv} = -\lambda(T_a - T_o) \tag{1.7}$$

where λ is the heat transfer coefficient of the atmosphere, T_a the temperature of the atmosphere and T_o the temperature of the ocean.
The radiative fluxes from the ocean expresses thanks to grey-body approximation :

$$Q_{ocean,rad} = \epsilon_a \sigma_B T_o^4 \tag{1.8}$$

where ϵ_a is the emissivity of the grey-body atmosphere, σ_B is the Stefan-Boltzmann constant.
We neglect the seasonal effect. Then the short-wave radiation flux from the sun expresses thanks to the latitude y :

$$R_a = \frac{C_0}{3}(1 + cos(y)) \tag{1.9}$$

Finally the atmosphere loses heat as a grey-body :

$$Q_{atm,rad} = -2\epsilon_a \sigma_B T_a^4 \tag{1.10}$$

Finally we have

$$\boxed{Q = -\lambda(T_a + T_o) + \epsilon_a \sigma_B T_o^4 + \frac{C_0}{3}(1 + cos(y)) - 2\epsilon_a \sigma_B T_a^4} \tag{1.11}$$

1.1.2.5 Primitive equations

Here are the primitive equations of the atmosphere in the terrestrial referential :

Conservation of the momentum :	$\frac{d\boldsymbol{v}}{dt} = -2\boldsymbol{\Omega} \times \boldsymbol{v} - \boldsymbol{\nabla}\phi - \frac{1}{\rho}\boldsymbol{\nabla}p - \frac{1}{\rho}\boldsymbol{\nabla\Sigma}$	(1.12)
Conservation of mass / Continuity equation :	$\frac{1}{\rho}\frac{d\rho}{dt} = -\boldsymbol{\nabla} \cdot \boldsymbol{v}$	(1.13)
Ideal gas law :	$p = \rho RT$	(1.14)
Thermodynamic equilibrium : :	$c_p\frac{dT}{dt} - \frac{1}{\rho}\frac{d}{dt}p = Q$	(1.15)

1.1.2.6 Geostrophic and Hydrostatic approximations

There are two typical approximations : the geostrophic one and the hydrostatic one. We define two numbers to compare the quantities in the conservation of the momentum equation : the Rossby number and Eckman number.

$$R_O = \frac{|\ \frac{d\boldsymbol{v}}{dt}\ |}{|\ 2\boldsymbol{\Omega} \times \boldsymbol{v}\ |} \tag{1.16}$$

$$E = \frac{|\ \frac{1}{\rho}\boldsymbol{\nabla\Sigma}\ |}{|\ 2\boldsymbol{\Omega} \times \boldsymbol{v}\ |} \tag{1.17}$$

With a scale analysis we get :

$$R_O \approx \frac{O(\frac{U^2}{L}, \frac{U}{\tau})}{O(2\Omega U)} \approx O(\frac{U}{2\Omega L}) \approx 0.1 \tag{1.18}$$

$$E \approx \frac{O(\frac{\nu U}{L^2})}{O(2\Omega U)} \approx O(\frac{\nu}{2\Omega L^2}) \approx 10^{-13} \tag{1.19}$$

This allows to neglect $\frac{d\boldsymbol{v}}{dt}$ and $\frac{1}{\rho}\boldsymbol{\nabla\Sigma}$ in the momentum equation. This leads us to the geostrophic and hydrostatic approximations and gives the geostrophic wind through the diagnostic equation :

$$2\boldsymbol{\Omega} \times \boldsymbol{v} = \boldsymbol{g^*} - \frac{1}{\rho}\boldsymbol{\nabla}p \tag{1.20}$$

It leads us to the hydrostatic equation and the geostrophic velocity ($\boldsymbol{\nabla_h}$ is horizontal)

$$\begin{aligned} &\frac{\partial p}{\partial z} = -\rho g^* \\ &\boldsymbol{v_g} = \mathbf{1_z} \times \frac{1}{\rho_s f}\boldsymbol{\nabla_h}p \end{aligned} \tag{1.21}$$

with $f = 2\Omega sin(\phi)$ the Coriolis parameter.

To have a prognostic equation we reintroduce the acceleration ($\boldsymbol{v_h}$ is the horizontal velocity) :

$$\boxed{\begin{array}{l}\dfrac{\partial p}{\partial z} = -\rho g^* \\ \dfrac{d\boldsymbol{v_h}}{dt} = -f\mathbf{1_z} \times \boldsymbol{v_h} - \dfrac{1}{\rho}\boldsymbol{\nabla_h} p\end{array}} \tag{1.22}$$

1.1.2.7 Pressure as vertical coordinate

We change $p(x, y, z, t)$ in $z(x, y, p, t)$. The new system is :

Conservation of the momentum :	$\dfrac{d\boldsymbol{v_h}}{dt} = -f\mathbf{1_p} \times \boldsymbol{v_p} - \dfrac{1}{\rho}\boldsymbol{\nabla_p}\phi$	(1.23)
Continuity equation :	$\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} + \dfrac{\partial w}{\partial p} = 0$	(1.24)
Hydrostatic equation :	$(\dfrac{\partial p}{\partial z})_p = -\dfrac{1}{\rho}$	(1.25)
Ideal gas law :	$p = \rho RT$	(1.26)
Thermodynamic equilibrium :	$c_p \dfrac{dT}{dt} + \dfrac{\partial p}{\partial z}\dfrac{d}{dt}p = Q$	(1.27)

The total derivative in pressure coordinates is :

$$\boxed{\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} + \frac{dp}{dt}\frac{\partial}{\partial p}} \tag{1.28}$$

1.1.3 Quasi-Geostrophic (QG) Model

1.1.3.1 QG Momentum equations

The velocity can be decomposed in a divergent and a rotational part :

$$\boldsymbol{v} = \mathbf{1_p} \times \boldsymbol{\nabla}\psi + \boldsymbol{\nabla}\chi \tag{1.29}$$

And we know that the geostrophic wind is quasi non-divergent :

$$\boldsymbol{v} \approx \mathbf{1_p} \times \boldsymbol{\nabla}\psi \tag{1.30}$$

We introduce a new quantity, the vorticity ξ :

$$\begin{array}{l}\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{v} \\ \boldsymbol{\omega_p} = \mathbf{1_p}(\dfrac{\partial v}{\partial x} - \dfrac{\partial u}{\partial y}) \\ \xi = \mathbf{1_p} \cdot \boldsymbol{\nabla} \times \boldsymbol{v} = \dfrac{\partial v}{\partial x} - \dfrac{\partial u}{\partial y} \\ \xi \approx \dfrac{1}{f}\nabla_p^2 \approx \dfrac{1}{f_0}\nabla_p^2\phi \text{ with } \psi = \phi/f_0 \text{ because the wind is geostrophic}\end{array} \tag{1.31}$$

So :

$$\begin{array}{l}\boldsymbol{v_g} = \mathbf{1_p} \times \boldsymbol{\nabla_p}\psi \\ \xi = \nabla_p^2\psi\end{array} \tag{1.32}$$

With some calculus scale analysis approximations we obtain :

$$\frac{\partial \xi}{\partial t} + (\boldsymbol{v_p} \cdot \boldsymbol{\nabla})(\xi + f) = (\xi + f)\frac{\partial w}{\partial p} \tag{1.33}$$

With the QG approximation we obtain $\boldsymbol{v_p} = \boldsymbol{v_g}$ and delete some terms :

$$\boxed{\frac{\partial \nabla^2 \psi}{\partial t} + (\boldsymbol{v_g} \cdot \boldsymbol{\nabla})(\nabla^2 \psi + f) = f_0 \frac{\partial w}{\partial p}} \tag{1.34}$$

1.1.3.2 QG Momentum equations : Two-layers Form

We divide the atmosphere in two isobaric layers ψ_1 (in the upper layer) and ψ_3 (in the lower layer). We have :

- at the top : $p_0 = 0$hPa and $\omega_0 = 0$
- in the upper layer : $p_1 = 250$hPa
- at the boundary : $p_2 = 500$hPa and ω_2
- in the lower layer : $p_3 = 750$hPa
- at the bottom : $p_4 = 1000$hPa and $\omega_4 \approx 0$

We approximate the Coriolis parameter (β plan):

$$f = f_0 + \beta y \tag{1.35}$$

We discretise the derivatives :

$$\begin{array}{l} \frac{\partial \omega}{\partial p}|_1 \approx \frac{\omega_2 - \omega_0}{p_2 - p_0} \approx \frac{\omega_2}{\delta p} \\ \frac{\partial \omega}{\partial p}|_3 \approx \frac{\omega_4 - \omega_2}{p_4 - p_0} \approx \frac{-\omega_2}{\delta p} \\ \frac{\partial \psi}{\partial p}|_2 \approx \frac{\psi_3 - \psi_1}{\delta p} \\ \psi = (\psi_3 + \psi_1)/2 \end{array} \tag{1.36}$$

We obtain the momentum equations :

$$\boxed{\begin{array}{l} \frac{\partial \nabla^2 \psi_1}{\partial t} + (\boldsymbol{v_{g,1}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_1) + \beta \frac{\partial \psi_1}{\partial x} = f_0 \frac{\omega_2}{\delta p} \\ \frac{\partial \nabla^2 \psi_3}{\partial t} + (\boldsymbol{v_{g,3}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_3) + \beta \frac{\partial \psi_3}{\partial x} = -f_0 \frac{\omega_2}{\delta p} \end{array}} \tag{1.37}$$

1.1.3.3 QG Momentum equations : Friction terms

We approximate the terms by their means and fluctuations e.g. $\boldsymbol{v} = \overline{\boldsymbol{v}} + \boldsymbol{v'}$.

When we study a motion on x for example and introduce these approximations, we get three terms of the Reynolds tensor. We only keep $\overline{u'w'}$.

Mixing length theory gives :

$$\begin{aligned} u' &= -l' \frac{\partial u}{\partial z} \\ w' &= -l' \frac{\partial w}{\partial z} \end{aligned} \tag{1.38}$$

Then :

$$\overline{u'w'} = l'^2 \frac{\partial u}{\partial z} \frac{\partial w}{\partial z} = K \frac{\partial u}{\partial z} \tag{1.39}$$

We get the general equation :

$$\begin{aligned} &\frac{\partial \nabla^2 \psi_1}{\partial t} + (\boldsymbol{v_{g,1}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_1) + \beta \frac{\partial \psi_1}{\partial x} = f_0 \frac{\omega_2}{\delta p} - \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) \\ &\frac{\partial \nabla^2 \psi_3}{\partial t} + (\boldsymbol{v_{g,3}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_3 + \frac{f_0 \rho g H}{\delta p}) + \beta \frac{\partial \psi_3}{\partial x} = -f_0 \frac{\omega_2}{\delta p} + \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) - C_D \nabla^2 \psi_3 \end{aligned} \tag{1.40}$$

We write it with the Jacobian :

$$\boxed{\begin{aligned} &\frac{\partial \nabla^2 \psi_1}{\partial t} + J(\psi_1, \nabla^2 \psi_1) + \beta \frac{\partial \psi_1}{\partial x} = f_0 \frac{\omega_2}{\delta p} - \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) \\ &\frac{\partial \nabla^2 \psi_3}{\partial t} + J(\psi_3, \nabla^2 \psi_3 + \frac{f_0 \rho g H}{\delta p}) + \beta \frac{\partial \psi_3}{\partial x} = -f_0 \frac{\omega_2}{\delta p} + \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) - C_D \nabla^2 \psi_3 \end{aligned}} \tag{1.41}$$

1.1.3.4 QG Thermodynamic equation : Temperature Form

We remind the thermodynamic equation in pressure coordinates :

$$c_p \frac{dT}{dt} + \frac{\partial p}{\partial z} \frac{d}{dt} p = Q \tag{1.42}$$

We develop the total derivative ($\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} + \frac{dp}{dt} \frac{\partial}{\partial p}$)

$$c_p (\frac{\partial T}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} T + \frac{dp}{dt} \frac{\partial T}{\partial p}) + \frac{\partial p}{\partial z} \frac{d}{dt} p = Q \tag{1.43}$$

We note that $\omega = \frac{dp}{dt}$:

$$c_p (\frac{\partial T}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} T) + \omega (c_p \frac{\partial T}{\partial p} + \frac{\partial p}{\partial z} p) = Q \tag{1.44}$$

We remind the hydrostatic relation $(\frac{\partial p}{\partial z})_p = -\frac{1}{\rho}$

$$c_p (\frac{\partial T}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} T) + \omega (c_p \frac{\partial T}{\partial p} - \frac{1}{\rho} p) = Q \tag{1.45}$$

We note $\sigma = -\frac{R}{p}(\frac{\partial T}{\partial p} - \frac{1}{\rho c_p})$:

$$\boxed{c_p (\frac{\partial T}{\partial t} + \boldsymbol{v_p} \cdot \boldsymbol{\nabla_p} T) - \omega c_p \sigma \frac{p}{R} = Q} \tag{1.46}$$

We can have the equivalent equation thanks to the Jacobian :

$$\boxed{c_p(\frac{\partial T}{\partial t} + J(\psi_{atm}, T) - \omega\sigma\frac{p}{R}) = Q} \tag{1.47}$$

We remind that we know Q.

$$Q = -\lambda(T_a + T_o) + \epsilon_a\sigma_B T_o^4 + C_0(1 + cos(y)) - 2\epsilon_a\sigma_B T_a^4 \tag{1.48}$$

Knowing Q we can write :

$$\boxed{c_p(\frac{\partial T}{\partial t} + J(\psi_{atm}, T) - \omega\sigma\frac{p}{R}) = -\lambda(T_a + T_o) + \epsilon_a\sigma_B T_o^4 + C_0(1 + cos(y)) - 2\epsilon_a\sigma_B T_a^4} \tag{1.49}$$

1.1.3.5 QG Thermodynamic equation : Vorticity Form

Thanks to the ideal gas law and the hydrostatic equation :

$$T = -\frac{p}{R}\frac{\partial\phi}{\partial p} \tag{1.50}$$

Then QG thermodynamic equation becomes :

$$c_p(\frac{\partial}{\partial t}(-\frac{p}{R}\frac{\partial\phi}{\partial p}) + \boldsymbol{v_p}\cdot\boldsymbol{\nabla_p}(\frac{-p}{R}\frac{\partial\phi}{\partial p}) - \frac{p}{R}\omega\sigma) = Q \tag{1.51}$$

p is independent from time and $\boldsymbol{\nabla_p}$ is horizontal :

$$\frac{\partial}{\partial t}(\frac{\partial\phi}{\partial p}) + \boldsymbol{v_p}\cdot\boldsymbol{\nabla_p}(\frac{\partial\phi}{\partial p})\omega\sigma = -\frac{RQ}{pc_p} \tag{1.52}$$

We know that the vorticity ψ is linked to the geopotential ϕ : $\phi = \psi * f_0$:

$$\boxed{\frac{\partial}{\partial t}(\frac{\partial\psi}{\partial p}) + \boldsymbol{v_p}\cdot\boldsymbol{\nabla_p}(\frac{\partial\psi}{\partial p})\omega\sigma = -\frac{RQ}{f_0pc_p}} \tag{1.53}$$

1.1.3.6 QG Thermodynamic equation : Two-layers Form

We remind the two-layer model. We divided the atmosphere in two isobaric layers ψ_1 (in the upper layer) and ψ_3 (in the lower layer). We had :

- at the top : $p_0 = 0hPa$ and $\omega_0 = 0$
- in the upper layer : $p_1 = 250hPa$
- at the boundary : $p_2 = 500hPa$ and ω_2
- in the lower layer : $p_3 = 750hPa$
- at the bottom : $p_4 = 1000hPa$ and $\omega_4 \approx 0$

We discretise the derivative :

$$\begin{aligned} &\frac{\partial \psi}{\partial p}|_2 \approx \frac{\psi_3 - \psi_1}{\delta p} \\ &\psi = (\psi_3 + \psi_1)/2 \\ &\boldsymbol{v}_{g,2} = (\frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial x}) \end{aligned} \tag{1.54}$$

Then we obtain the thermodynamic equation :

$$\boxed{\frac{\partial}{\partial t}\frac{\psi_1 - \psi_3}{\delta p} + \boldsymbol{v_{g,2}} \cdot \boldsymbol{\nabla_p} \frac{\psi_1 - \psi_3}{\delta p} + \frac{w_2 \sigma}{f_0} = -\frac{RQ}{f_0 p c_p}} \tag{1.55}$$

1.1.3.7 QG equations

Mom. Eq. $$\frac{\partial \nabla^2 \psi_1}{\partial t} + (\boldsymbol{v_{g,1}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_1) + \beta \frac{\partial \psi_1}{\partial x} = f_0 \frac{\omega_2}{\delta p} - \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) \tag{1.56}$$

Mom. Eq. : $$\frac{\partial \nabla^2 \psi_3}{\partial t} + (\boldsymbol{v_{g,3}} \cdot \boldsymbol{\nabla})(\nabla^2 \psi_3 + \frac{f_0 \rho g H}{\delta p}) + \beta \frac{\partial \psi_3}{\partial x} = -f_0 \frac{\omega_2}{\delta p} + \frac{K}{\Delta p^2}(\nabla^2 \psi_1 - \nabla^2 \psi_3) - C_D \nabla^2 \psi_3 \tag{1.57}$$

Thermo. Eq. : $$\frac{\partial}{\partial t}\frac{\psi_1 - \psi_3}{\delta p} + \boldsymbol{v_{g,2}} \cdot \boldsymbol{\nabla_p} \frac{\psi_1 - \psi_3}{\delta p} + \frac{w_2 \sigma}{f_0} = -\frac{RQ}{f_0 p c_p} \tag{1.58}$$

1.2 Ocean model : the shallow water equations

1.2.1 Single shallow layer

We remind that we refer to to the course of Stéphane Vannitsem [6] and the book of Geoffrey K. Vallis [7].
Shallow water dynamics apply to a fluid layer of constant density in which the aspect ratio is very small. This hypothesis gives the hydrostatic relation. So the dynamics is well described by the momentum equations and the mass continuity equation. The ocean is considered as a single layer (a few hundred meters) on top of a deep layer and beneath a fluid of negligible mass. We suppose that the surface on the top is rigid because the variations of height are small.

We take the notations :

- $\rho \approx 0$ the atmosphere density
- η_0 (constant) the height of the free-surface
- ρ_1 and p_1 are the density and pressure of the single layer
- η_1 the depth of the free-surface
- ρ_2 and p_2 are the density and pressure of the lower rigid layer

1.2.2 Mass continuity equation

The mass convergence in a column of water is given by :

$$F_m = \text{mass flux in} = -\int_S \rho \boldsymbol{u} \cdot d\boldsymbol{S} \tag{1.59}$$

where S is the area of the vertical boundary of the column. We transform $d\boldsymbol{S}$ with $h\boldsymbol{n}\delta l$ where δl is a line element circumscribing the column and $\boldsymbol{n}$ is a unit vector perpendicular to the boundary, pointing outwards.

$$F_m = -\oint \rho h \boldsymbol{u} \cdot \boldsymbol{n} dl \tag{1.60}$$

Divergence theorem gives :

$$F_m = -\int_A \nabla_z \cdot (\rho h \boldsymbol{u})\, dA \tag{1.61}$$

With another definition of F_m :

$$F_m = \frac{d}{dt}\int \rho dV = \frac{d}{dt}\int_A \rho h dA = \int_A \rho \frac{\partial h}{\partial t} dA \tag{1.62}$$

We deduce globally and then locally :

$$\boxed{\frac{dh}{dt} + h\boldsymbol{\nabla_z} \cdot \boldsymbol{u} = 0} \tag{1.63}$$

1.2.3 Momentum equations

Here is the momentum equations :

$$\boxed{\begin{aligned} &\frac{\partial p}{\partial z} = -\rho g \\ &\frac{d\boldsymbol{u}}{dt} + \boldsymbol{f} \times \boldsymbol{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p_1 + \frac{1}{\rho}\frac{\partial \tau}{\partial z} \end{aligned}} \tag{1.64}$$

with p the pressure, ρ the density, u the speed, f the Coriolis parameter and τ the surface stress.

We integrate the hydrostatic equation assuming ρ constant with the condition $p(z = \eta_0) = p_{atm}$:

$$\begin{aligned} p_1 &= -\rho g z + p_{atm} \\ p_2 &= -\rho_1 g \eta_1 + \rho_2 g(\eta_1 - z) + p_{atm} \end{aligned} \tag{1.65}$$

We differentiate p_1 :

$$\boldsymbol{\nabla} p_1 = \boldsymbol{\nabla} p_{atm} \tag{1.66}$$

Knowing that the pressure gradient of the lower layer is zero because there is no motion, we deduce :

$$g(\rho_2 - \rho_1)\boldsymbol{\nabla} h = \nabla p_{atm} = \boldsymbol{\nabla} p_1 \tag{1.67}$$

We can change the horizontal equation :

$$\boxed{\frac{d\boldsymbol{u}}{dt} + \boldsymbol{f} \times \boldsymbol{u} = -g'\boldsymbol{\nabla_z} h + \frac{1}{\rho}\frac{\partial \tau}{\partial z}} \tag{1.68}$$

1.2.4 Equations through the vorticity

By the same operation (applying the rotational ie multiply by curl $= \mathbf{1_z} \cdot \boldsymbol{\nabla}\times$) of the atmosphere we have :

$$\begin{aligned} \frac{d}{dt}(\frac{\xi+f}{h}) &= (\frac{1}{h}\frac{d}{dt}(\xi+f) - \frac{\xi+f}{h^2}\frac{dh}{dt}) \\ h\frac{d}{dt}(\frac{\xi+f}{h}) &= (\frac{d}{dt}(\xi+f) - \frac{\xi+f}{h}\frac{dh}{dt}) \\ h\frac{d}{dt}(\frac{\xi+f}{h}) &= (\boldsymbol{\nabla}\times\frac{1}{\rho}\frac{\partial\tau}{\partial z}) \end{aligned} \tag{1.69}$$

That is :

$$\frac{d}{dt}(\frac{\xi+f}{h}) = \frac{1}{h}\text{curl}(\frac{1}{\rho}\frac{\partial\tau}{\partial z}) \tag{1.70}$$

1.2.5 Approximation of $\frac{d}{dt}(\frac{\xi+f}{h})$, the potential vorticity

We know that $\eta = h - H$ with H constant and $\eta \ll H$. We can develop $(\xi+f)/h$:

$$\frac{d}{dt}(\frac{\xi+f}{h}) = \frac{d}{dt}(\frac{\xi+f}{H}(1-\frac{\eta}{H})) \tag{1.71}$$

We assume $f \approx f_0 \gg \xi$:

$$\begin{aligned} \frac{d}{dt}(\frac{\xi+f}{h}) &= \frac{1}{H}\frac{d}{dt}(\xi+f-f_0\frac{\eta}{H}) \\ \frac{d}{dt}(\frac{\xi+f}{h}) &= \frac{1}{H}\frac{d}{dt}(\xi+f-f_0^2\frac{\psi}{g'H})) \end{aligned} \tag{1.72}$$

With $L_R = \sqrt{(g'H/f_0)}$ the Rossby deformation radius, we get :

$$\boxed{\frac{d}{dt}(\frac{\xi+f}{h}) = \frac{1}{H}\frac{d}{dt}(\xi+f-\frac{\psi}{L_R^2}))} \tag{1.73}$$

1.2.6 Calculus of the friction

We get :

$$\frac{1}{H}\frac{d}{dt}(\xi+f-\frac{\psi}{L_R^2})) = \frac{1}{h}\text{curl}(\frac{1}{\rho}\frac{\partial\tau}{\partial z}) \tag{1.74}$$

We integrate both side of the dynamical equation from -H to 0. The left term does not depend on the vertical coordinates :

$$\begin{aligned} \frac{d}{dt}(\xi+f-\frac{\psi}{L_R^2})) &= \int_{-H}^{0}\frac{1}{h}(\text{curl}(\frac{1}{\rho}\frac{\partial\tau}{\partial z}))dz \\ \frac{d}{dt}(\xi+f-\frac{\psi}{L_R^2})) &= \frac{1}{\rho}\int_{-H}^{0}\frac{1}{h}(\text{curl}(\frac{\partial\tau}{\partial z}))dz \end{aligned} \tag{1.75}$$

$h \approx H$ leads us to :

$$\begin{aligned} \frac{d}{dt}(\xi + f - \frac{\psi}{L_R^2}) &= \frac{1}{\rho H}\text{curl}(\int_{-H}^{0}(\frac{\partial \tau}{\partial z})dz) \\ \frac{d}{dt}(\xi + f - \frac{\psi}{L_R^2}) &= \frac{1}{\rho H}\text{curl}(\tau_{top} - \tau_{bottom}) \end{aligned} \tag{1.76}$$

As in the friction between the two layers of the atmosphere, τ_{top} is modeled by $K(\frac{\partial u}{\partial z}, \frac{\partial v}{\partial z})$

And then we apply the rotational and get $\text{curl}\tau_{top} = K\nabla^2(\psi_3 - \psi)$.

For τ_{bottom} we model the surface friction with $C\boldsymbol{u}$ and get $\text{curl}\tau_{bottom} = -C\nabla^2\psi$

Finally we get for momentum equation :

$$\frac{d}{dt}(\xi + f - \frac{\psi}{L_R^2}) = \frac{K}{\rho H}\nabla^2(\psi_3 - \psi) - \frac{C}{\rho H}\nabla^2\psi \tag{1.77}$$

We can write it as :

$$\boxed{\frac{\partial}{\partial t}(\nabla^2\psi - \frac{\psi}{L_R^2}) + J(\psi, \nabla^2\psi) + \beta\frac{\partial \psi}{\partial x} = \frac{K}{\rho H}\nabla^2(\psi_3 - \psi) - \frac{C}{\rho H}\nabla^2\psi} \tag{1.78}$$

1.3 MAOOAM

1.3.1 6 fields

The model has been published in De Cruz, Demaeyer and Vannitsem 2016 [8].

The 6 fields will be reduced further to four variables.

- ψ_a^1 (at 250 hPa) and ψ_a^3 (at 750 hPa) the streamfunctions of the atmosphere, further resumed in $\psi_a = (\psi_a^1 + \psi_a^3)/2$
- ψ_o the streamfunction of the ocean
- T_a the temperature of the atmosphere which will be decomposed as $T_a^0 + \delta T_a$, with δT_a proportional to $\dfrac{\psi_a^1 - \psi_a^3}{2}$
- T_o the temperature of the ocean which will be decomposed as $T_o^0 + \delta T_o$
- $\omega = \dfrac{dp}{dt}$ the vertical velocity which will be eliminated from the equation further

1.3.2 Motion for the atmospheric streamfunction fields ψ_a^1 and ψ_a^3

$$\frac{\partial}{\partial t}(\nabla^2\psi_a^1) + J(\psi_a^1, \nabla^2\psi_a^1) + \beta\frac{\partial \psi_a^1}{\partial x} = -k_d'\nabla^2(\psi_a^1 - \psi_a^3) + \frac{f_0}{\Delta p}\omega \tag{1.79}$$

$$\frac{\partial}{\partial t}(\nabla^2\psi_a^3) + J(\psi_a^3, \nabla^2\psi_a^3) + \beta\frac{\partial \psi_a^3}{\partial x} = k_d'\nabla^2(\psi_a^1 - \psi_a^3) - \frac{f_0}{\Delta p}\omega - k_d\nabla^2(\psi_a^3 - \psi_o) \tag{1.80}$$

For these equations, we have :

- J is the Jacobian operator
- the Coriolis parameter f is linearized around a value f_0 estimated at latitude $\phi_0 = 45°N$, $f = f_0 + \beta y$ with $\beta = df/dy$
- the parameters k'_d and k_d that quantify the friction between the two atmospheric layers and between the ocean and the atmosphere, respectively
- $\Delta p = 500hPa$ is the pressure difference between the atmospheric layers.

1.3.3 Motion for the ocean streamfunction fields ψ_o

$$\frac{\partial}{\partial t}(\nabla^2\psi_o - \frac{\psi_o}{L_R^2}) + J(\psi_o, \nabla^2\psi_o) + \frac{\partial\psi_o}{\partial x} = -r\nabla^2\psi_o + \frac{C}{\rho h}\nabla^2(\psi_a^3 - \psi_o) \quad (1.81)$$

For this equation we have :

- L_R the reduced Rossby deformation radius
- ρ the density
- h the depth
- r the friction at the bottom of the active ocean layer
- the rightmost term represents the impact of the wind stress, and is modulated by the drag coefficient of the mechanical ocean-atmosphere coupling, $d = C/(\rho h)$

1.3.4 Time evolution of the atmosphere and ocean temperature T_a and T_o

$$\gamma_a(\frac{\partial T_a}{\partial t} + J(\psi_a, T_a) - \sigma\omega\frac{p}{R}) = -\lambda(T_a - T_o) + \epsilon_a\sigma_B T_o^4 - 2\epsilon_a\sigma_B T_a^4 + R_a \quad (1.82)$$

$$\gamma_a(\frac{\partial T_o}{\partial t} + J(\psi_o, T_o) = -\lambda(T_o - T_a) - \sigma_B T_o^4 + \epsilon_a\sigma_B T_a^4 + R_o \quad (1.83)$$

For these equations we have :

- γ_a and γ_o are the heat capacities of the atmosphere and the active ocean layer
- $\psi_a = (\psi_a^1 + \psi_a^3)/2$ is the atmospheric barotropic streamfunction
- λ is the heat transfer coefficient at the ocean-atmosphere interface
- σ is the static stability of the atmosphere (constant)
- the quartic terms represent the long-wave radiation fluxes between the ocean, the atmosphere, and outer space
- ϵ_a is the emissivity of the grey-body atmosphere
- σ_B is the Stefan-Boltzmann constant

- T_a and T_o are decomposed as $T_a = T_a^0 + \delta T_a$ and $T_o = T_o^0 + \delta T_o$, then the quartic terms are linearized around spatially uniform temperatures T_a^0 and T_o^0.
- R_a and R_0 are the short-wave radiation fluxes entering the atmosphere and the ocean that are also decomposed as $R_a = R_a^0 + \delta R_a$ and $R_o = R_o^0 + \delta R_o$. These short-wave radiation or insolation are will be determined by $\delta R_a = C_a F_1$ and $\delta R_o = C_o F_1$ where F_1 will be the first Fourier basis function.

1.4 Manipulation on the equations

1.4.1 Simplification of the equations

The hydrostatic relation in pressure coordinates is :

$$\frac{\partial \Phi}{\partial p} = -\frac{1}{\rho_a} \tag{1.84}$$

Where the geopotential height is :

$$\Phi^i = f_0 \psi_a^i \text{ for } i \in \{1, 3\} \tag{1.85}$$

The ideal gas relation is :

$$p = \rho_a R T_a \tag{1.86}$$

where R is the ideal gas relation.
This ideal gas relation allows to write the temperature anomaly :

$$\delta T_a = 2 f_0 \theta_a / R \tag{1.87}$$

where $\theta_a \equiv (\psi_a^1 - \psi_a^3)/2$ is the baroclinic streamfunction.
This is used to eliminate the vertical velocity ω from Eqs. (1.80)-(1.81) and (1.83). This reduces the independent dynamical fields to the streamfunction fields ψ_a and ψ_o and the spatially dependent temperatures anomalies δT_a and δT_o.

1.4.2 Adimensionalization

The prognostic equations for the four fields are non-dimensionalized by dividing time by f_0^{-1}, pressure by the difference Δp, temperature by $f_0^2 L^2 / R$ and the streamfunction by $L^2 f_0$.
The model approximates the earth in a plane with a latitude ϕ_0. The adimensional coordinates are :

$$\begin{cases} x' &= x/L \text{ (the zonal extent)} \\ y' &= y/L \text{ (the meridional extent)} \end{cases} \tag{1.88}$$

where L is a characteristic length scale (related to the meridional channel width $L_y = \pi L = 5 \times 10^3 km$). We define $n = \dfrac{2L_y}{L_x} = 1.5$ which is the aspect ratio between the meridional and zonal extent.

We define the domain by :

$$\begin{cases} 0 \leq x' \leq \dfrac{2\pi}{n} \\ 0 \leq y' \leq \pi \end{cases} \tag{1.89}$$

There is a more detailed discussion of the model equations and their non-dimensionalization in Vannitsem and De Cruz 2014 [5].

1.4.3 Boundary conditions

The boundary conditions are Neumann limit conditions. The atmospheric flow is defined in a zonally periodic channel with no-flux boundary conditions in the meridional direction :

$$\frac{\partial \cdot_a}{\partial x'} \equiv 0 \text{ at } y' = 0, \pi \tag{1.90}$$

The oceanic flow is confined within an ocean basin by imposing no-flux boundaries in both the meridional and zonal directions :

$$\begin{array}{l} \dfrac{\partial \cdot_o}{\partial x'} \equiv 0 \text{ at } y' = 0, \pi \\ \dfrac{\partial \cdot_o}{\partial y'} \equiv 0 \text{ at } x' = 0, 2\pi/n \end{array} \tag{1.91}$$

1.4.4 The modes

We project the physical variables on Fourier basis that we truncate. The variables are expanded in the orthonormal eigenfunctions, F_i (solutions of the boundary conditions) following the nomenclature of Cehelsky and Tung 1987 [9] :

In this nomenclature for the atmosphere there are three types of modes A, K and L :

$$\begin{cases} F_P^A(x', y') &= \sqrt{2}\cos(Py') \\ F_{M,P}^K(x', y') &= 2\cos(Mnx')\sin Py' \\ F_{H,P}^L(x', y') &= 2\sin(Hnx')\sin Py' \end{cases} \tag{1.92}$$

For the ocean there is only one type of modes :

$$\begin{cases} \phi_{H_o,P_o}(x', y') &= 2\sin(\dfrac{H_o n x'}{2})\sin(P_o y') \end{cases} \tag{1.93}$$

Then we can truncate the Fourier modes by choosing the numbers P,M and H for the atmosphere and H_o P_o. They belongs to $\mathbf{N}^*$.Their range are :

$$\begin{cases} 1 \leq H \leq H^{max} \\ 1 \leq M \leq M^{max} \\ 1 \leq P \leq P^{max} \end{cases}$$

Actually $M^{max} = H^{max}$.

$$\begin{cases} 1 \leq H_o \leq H_o^{max} \\ 1 \leq P_o \leq P_o^{max} \end{cases}$$

We have to be care to not **confuse** these integer numbers with A, K, L ! These letters (P,M,H,H_o and P_o) give us the type of the basis functions of the atmosphere.

It gives us the total number of functions : n_a basic atmosphere functions and n_o basic atmosphere functions :

$$\left\{ \begin{array}{rcl} n_a & = & P^{max}(2H^{max}+1) \\ n_o & = & P_o^{max} H_o^{max} \end{array} \right.$$

We compute it by enumerating the possibilities for each one :

- There are P^{max} possibilities for the A basic functions.
- There are $P^{max}H^{max}$ possibilities for the K basic functions.
- There are $P^{max}H^{max}$ possibilities for the L basic functions.
- There are $P_o^{max}H_o^{max}$ possibilities for the K basic functions.

VDGG is used, it is the third version of the model (defined in Vannitsem et al. 2015). $H^{max} = 2$, $P^{max} = 2$, $P_o^{max} = 4$ and $H_o^{max} = 2$. So we have 10 modes for the atmosphere and 8 modes for the ocean :

We get these basis functions :

$$\left\{ \begin{array}{rcl} F_1(x',y') & = & \sqrt{2}\cos(y') \\ F_2(x',y') & = & 2\cos(nx')\sin y' \\ F_3(x',y') & = & 2\sin(nx')\sin y' \\ F_4(x',y') & = & \sqrt{2}\cos(2y') \\ F_5(x',y') & = & 2\cos(nx')\sin 2y' \\ F_6(x',y') & = & 2\sin(nx')\sin 2y' \\ F_7(x',y') & = & 2\cos(2nx')\sin y' \\ F_8(x',y') & = & 2\sin(2nx')\sin y' \\ F_9(x',y') & = & 2\cos(2nx')\sin 2y' \\ F_{10}(x',y') & = & 2\sin(2nx')\sin 2y' \\ \phi_1(x',y') & = & 2\sin(\frac{nx'}{2})\sin(y') \\ \phi_2(x',y') & = & 2\sin(\frac{nx'}{2})\sin(2y') \\ \phi_3(x',y') & = & 2\sin(\frac{nx'}{2})\sin(3y') \\ \phi_4(x',y') & = & 2\sin(\frac{nx'}{2})\sin(4y') \\ \phi_5(x',y') & = & 2\sin(nx')\sin(y') \\ \phi_6(x',y') & = & 2\sin(nx')\sin(2y') \\ \phi_7(x',y') & = & 2\sin(nx')\sin(3y') \\ \phi_8(x',y') & = & 2\sin(nx')\sin(4y') \end{array} \right.$$

To summarize for the atmosphere streamfunction there are 10 Fourier modes F_i, for the atmosphere temperature there are 10 Fourier modes F_i. For the ocean streamfunction there are 8 Fourier modes ϕ_i and for the ocean temperature there are 8 Fourier modes ϕ_i.

We can write it as :

$$\psi_a(x',y',t) = \sum_{i=1}^{10} \psi_{a,i}(t) F_i(x',y') \tag{1.94}$$

$$\delta T_a(x',y',t) = \sum_{i=1}^{10} \delta T_{a,i}(t) F_i(x',y') \tag{1.95}$$

$$\psi_o(x',y',t) = \sum_{j=1}^{8} \psi_{o,i}(t)(\phi_j(x',y') - \overline{\phi_j}) \tag{1.96}$$

$$\delta T_o(x',y',t) = \sum_{j=1}^{8} \delta T_{o,j} \phi_j(x',y') \tag{1.97}$$

The short-wave radiation or insolation is determined by $\delta R_a = C_a F_1$ and $\delta R_o = C_o F_1$. A term $\overline{\phi_j}$ is added to the oceanic basis function in order to give it a vanishing spatial average. It does not affect the dynamics.

1.4.5 The integration

Substituting the fields in the equations and projecting on the different basis functions yields 36 ordinary differential equations (ODEs) for 36 variables. Due to the linearization of the quartic temperature fields, these equations are at most bilinear in the four variables referred by η_i components of $\boldsymbol{\eta}$. So it gave us 36 bilinear equations. We summarize it with it a tensor T_{ijk} :

$$\frac{d\eta_i}{dt} = \sum_{j=0}^{N} \sum_{k=0}^{N} T_{ijk} \eta_j \eta_k \tag{1.98}$$

when $(i,j) = (0,0)$, we have the constant terms, when $i = 0$ and not j or when $j = 0$ but not i this corresponds to the linear terms.

The tensor is sparse and upper triangular. We integrate with a Heun method (RK2).

1.5 The physical parameters

Here are the main parameters according to the table 1 of Vannitsem 2015 [11] :

Dynamic atmosphere	Dynamic ocean	Geometry	Coupling
$k_d = gC/\Delta p \ s^{-1}$	$L_R = (g'H)^{1/2}/f_0 \ m$	$L_y = \pi L = 5000km$	$\epsilon_a = 0.7$
$k'_d = k_d$	$r = 10^{-7} \ s^{-1}$	$n = (2L_y)/L_x = 1.5$	$\lambda = c_{p,a} C \ Wm^{-2}K^{-1}$
$\sigma = 2.1610^{-6} \ Jkg^{-1}Pa^{-2}$	$\gamma_O = c_{p,o}\rho_o H \ Jm^{-2}K^{-1}$	$f_0 = 0.0001032 \ s^{-1}$	$d = C/\rho_o H) \ s^{-1}$
$\gamma_a = 10^7 \ Jm^{-2}K^{-1}$		$\beta = 1.6210^{-11} \ m^{-1}s^{-1}$	
		$\Delta p = 500hPa$	

Note : $c_{p,a}$ and σ_B are the usual specific heat at constant pressure of the air and the Stefan-Boltzmann constant, fixed to 1004 $Jkg^{-1}K^{-1}$ and 5.610^{-8} $Wm^{-2}K^{-4}$ respectively. The density, ρ_o, and the specific heat at constant pressure,$c_{p,o}$,for the ocean layer are fixed to 1000 kgm^{-3} and 4000 $Jkg^{-1}K^{-1}$. g and g' are the gravity and reduced gravity fixed to 10 and 0.031 ms^{-2}, respectively.

The program needs essentially the list below of physical parameters. In the next table, here are two non-exhaustive lists computed thanks to Vannitsem 2015 [11]:

param.	weakly coupled - chaotic	strongly coupled - less chaotic	Description
C_o	$310 W m^{-2}$	$310 W m^{-2}$	Net short wave radiation input for the ocean
C_a	$103.3333 W m^{-2}$	$103.3333 W m^{-2}$	Radiation input for the atmosphere
γ_a	$10^7 J m^{-2} K^{-1}$	$10^7 J m^{-2} K - 1$	Specific heat capacity of the atmosphere
H	$165 m$	$165 m$	Depth of the ocean layer
k	0.01	0.0145	Friction coefficient at the bottom of the atmosphere (adim.)
kp	0.02	0.0290	Internal friction between the atmosphere layers (adim.)
d	$6 * 10^{-8}\ s^{-1}$	$9 * 10^{-8}\ s^{-1}$	Friction coefficient between the ocean and the atmosphere
λ	$10\ W m^{-2} K^{-1}$	$15.06\ W m^{-2} K^{-1}$	Heat exchange between the ocean and the atmosphere
γ_o	$6.6 * 10^8 J m^{-2} K^{-1}$	$6.6 * 10^8 J m^{-2} K^{-1}$	Specific heat capacity of the ocean
$To0$	$301 K$	$299.35 K$	Stationary solution for the 0th order ocean temperature
$Ta0$	$289 K$	$290.2 K$	Stationary solution for the 0th order atmospheric temperature

These are calculated thanks to the formula of Vannitsem 2015 [11]. The first set of parameters corresponds to the chaotic case. And the second set of parameters corresponds to a less chaotic case where a low frequency variability appears. The model is more coupled in the second set because the friction coefficient is larger and so k, d and λ are larger.

1.6 Six regimes

We defines six regimes with two set of parameters (one chaotic and one less chaotic) and three set of dimensions :

- **36wk** is the regime with 36 dimensions and weak coupled physical parameters
- **52wk** is with 52 dimensions, adding some ocean modes compared to the 36 case.
- **56wk** is with 56 dimensions, adding some atmosphere modes compared to the 36 case.
- **36st** is with 36 dimensions and strong coupled physical parameters
- **52st** is with 52 dimensions and strong coupled physical parameters
- **56st** is with 56 dimensions and strong coupled physical parameters

1.7 Chaotic regimes

The chaotic regime is obtained with the first set of parameters. It corresponds to the cases 36wk, 52wk and 56 wk. Figure 1 is the attractor of the regime 36wk on the three important modes $\psi_{o,2}$, $\theta_{o,2}$ and $\psi_{a,1}$ respectively the second

Fourier mode of the ocean streamfunction, the second Fourier mode of the ocean temperature and the first Fourier mode of the atmosphere streamfunction. The attractor has no particular form. Figure 2 shows the first mode of the streamfunction. This is the chaotic form for $\psi_{a,1}$. The ranges of the three variables are

$$\psi_{o,2} \in [-7.5829 * 10^{-5}, 6.2429 * 10^{-5}]$$
$$\theta_{o,2} \in [0.1584, 0.2147]$$
$$\psi_{a,1} \in [0.0285, 0.0546]$$

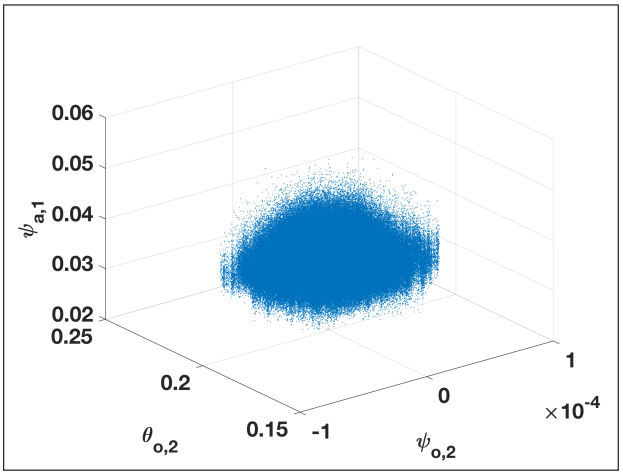


Figure 1: Chaotic attractor - Regime 36wk

Chaotic attractor of a run of 10 millions days of the ocean-atmosphere model MAOOAM in the regime 36 weakly coupled. X-axis is $\psi_{o,2}$, Y-axis is $\theta_{o,2}$, Z-axis is $\psi_{a,1}$.

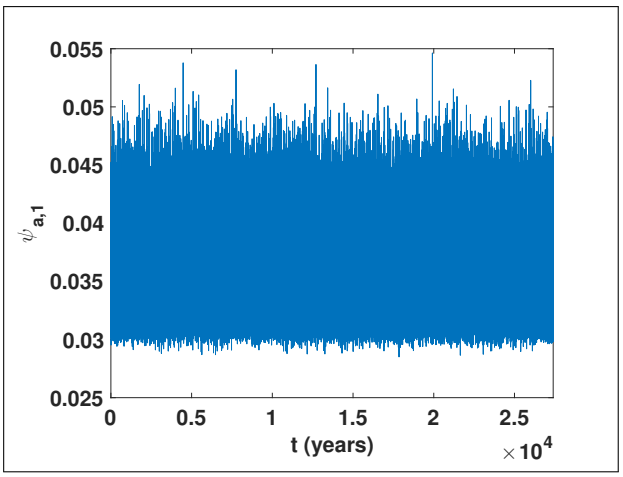


Figure 2: Chaotic behavior of $\psi_{a,1}$- Regime 36wk

$\psi_{a,1}$ is plotted with a run of 10 millions days of the ocean-atmosphere model MAOOAM in the regime 36 weakly coupled. X-axis is time in years.

1.8 "Less chaotic" regimes

The "less chaotic" regime is obtained with the second set of parameters. It corresponds to the cases 36st, 52st and 56st. "Less chaotic" adjective is to say that it is less chaotic than the previous regime. Figure 4 displays the first mode of the atmosphere streamfunction. It displays a low frequency of a period of 16.43 years. Figure 3 displays the attractor with the three important modes $\psi_{o,2}$, $\theta_{o,2}$ and $\psi_{a,1}$. Two regimes are visible : one active for larger value of $\psi_{a,1}$ and one passive for smaller values of $\psi_{a,1}$. They will be referred as 36-active and 36-passive.

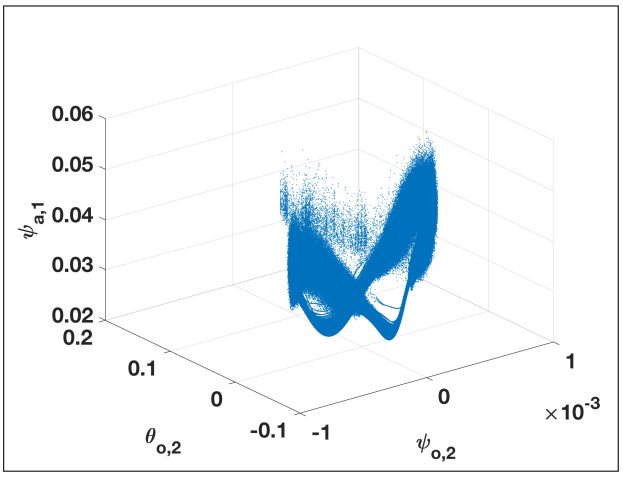


Figure 3: "Less chaotic" attractor - Regime 36st

Less chaotic attractor of a run of 10 millions days of the ocean-atmosphere model MAOOAM in the regime 36 strongly coupled. X-axis is $\psi_{o,2}$, Y-axis is $\theta_{o,2}$, Z-axis is $\psi_{a,1}$.

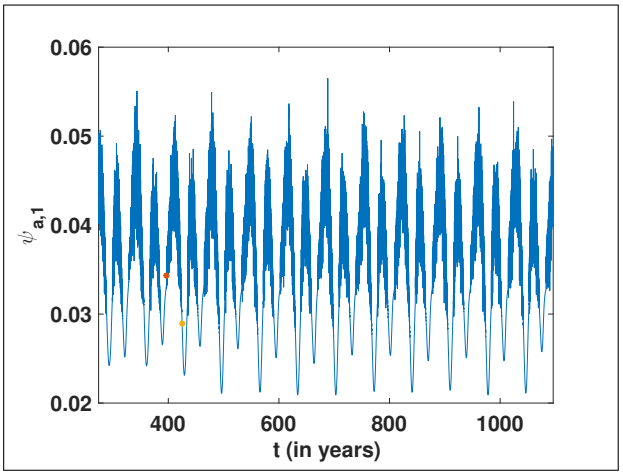


Figure 4: Regime 36st - $\psi_{a,1}$

$\psi_{a,1}$ is plotted with a run of 10 millions days of the ocean-atmosphere model MAOOAM in the regime 36 weakly coupled. X-axis is time in years.

Red point is the active area and Yellow point is the passive area.

1.9 Fortran implementation

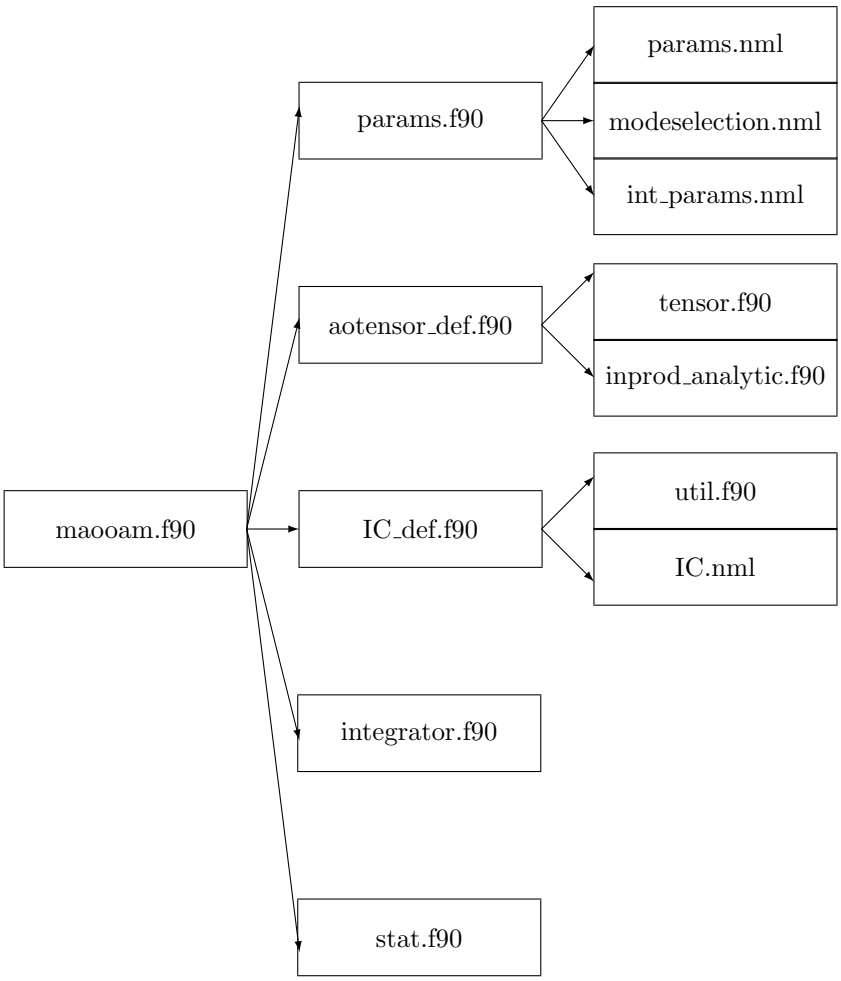
1.9.1 Organization of the code

The model is coded in Fortran 90 by Lesley De Cruz and Jonathan Demaeyer. There is a lua version too. The GitHub repository is : https://github.com/Climdyn/MAOOAM.

Here are the main files :

params.nml	Physical parameters
int_params.nml	Integral parameters
modeselection.nml	Help to generate the Fourier modes
IC_def.f90	Generate initial condition file
IC.nml	Initial condition file
inprod_analytic.f90	Compute the inner-products needed for the tensor
aotensor_def.f90	Compute the sparse tensor
integrator.f90	RK2 integration with the tensor
maooam.f90	main program
util.f90	Help functions
stat.f90	Statistics functions
param.f90	Compute the physical parameters
tensor.f90	Help functions

Here are the dependencies :


maooam.f90
params.f90
aotensor_def.f90
IC_def.f90
integrator.f90
stat.f90
params.nml
modeselection.nml
int_params.nml
tensor.f90
inprod_analytic.f90
util.f90
IC.nml

1.9.2 Parameters files

1.9.2.1 param.nml

It defines the fundamental physical parameter for the model. In the directory params, there are different regimes of MAOOAM.

1.9.2.2 int_params.nml

It represents the integration parameters. There are the transient period, the effective running time after the transient, the time step, the time step of writing the data.

Time units are not seconds,the time in the model is adimensional. In order to dimensionalize it, it has to be divided by f_0 (the Coriolis parameter) to have it in seconds. One adimensional time-unit with the usual Coriolis parameter at midlatitude corresponds thus approximately to 2.7 hours or 0.1122 days.

1.9.2.3 modeselections.nml

It defines the number of Fourier modes. The physical variables will be expanded on these Fourier modes.

In the classic regime, the dimension is 36, there are 10 basic functions F_i for the atmosphere streamfunction ψ_a , 10 basic functions F_i for the atmosphere temperature T_a, 8 basic functions ϕ_i for the ocean streamfunction ψ_o and 8 basic functions ϕ_i for the ocean temperature T_o.

A,K,L are for the type of the basic functions and P,M,H,H_o,P_o are the number to change the modes depending of the type of basic functions.

Nx,Ny are the quantities in front of x and y regardless the other quantities. It can be P or M ...For the ocean, $Nx = H_o$ and $Ny = P_o$ are the only quantities in front of x' and y'. For the atmosphere, $Nx = M$ or H or 0 and $Ny = P_o$.

We represent it by the matrices OMS and AMS. Each line is represents the couple (Nx,Ny).

$$AMS = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{pmatrix} \tag{1.99}$$

$$OMS = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \\ 2 & 4 \end{pmatrix} \tag{1.100}$$

Warning : for OMS coefficient, Nx accounts for half-integer wavenumber. 1 in the matrix represents 1/2 in front of x'.

AMS and OMS are computed in the file modeselections.nml. There are the numbers NBOC and NBATM that represents the number of blocs / lines. Here : $NBOC = 8$ and $NBATM = 4$.

1.9.3 Initial condition files

1.9.3.1 IC_def.f90

This file create IC.nml if it does not exist with the good dimensions.

1.9.3.2 IC.nml

It initializes the 36 component of the basic functions. It gives us the structure of the vector X that represents the state of the model.

- there are 10 psi variables which are components of ψ_a the streamfunction of the atmosphere.
- there are 10 theta variables which are components of T_a the temperature of the atmosphere
- there are 8 A variables which are components of ψ_o the streamfunction of the ocean
- there are 8 T variables which are components of T_o the temperature of the atmosphere

With random initial conditions, the system will converge toward a chaotic attractor. We can choose random variables between say -1e-2 and +1e-2 to leave the quasi-periodic solution and to the chaotic one.

1.9.4 Compute the tensor

1.9.4.1 inprod_analytic.f90

Compute the inner products needed to compute the tensor.

1.9.4.2 aotensor.f90

Compute the tensor for the bilinear equations system.

1.9.5 Step function

1.9.5.1 integrator.f90

Compute RK2 integration.

1.10 Python translation

1.10.1 Organization of the code

I have translated the fortran code in python. The last version is on Git : https://github.com/Climdyn/MAOOAM/tree/master/python. The documentation is in appendix.
Here are the main files, the organisation is the same as the fortran code.

params_maooam.py	Physical,integral, dimensional parameters
ic_def.py	Generate initial condition file
ic.py	Initial condition file
inprod_analytic.py	Compute the inner-products needed for the tensor
aotensor.py	Compute the sparse tensor
integrator.py	RK2 integration with the tensor
maooam.py	main program

1.10.2 Parameters file params_maooam.py

It defines the fundamental physical parameter for the model. In the directory params, there are different regimes of MAOOAM.
It represents the integration parameters. There are the transient period, the effective running time after the transient, the time step, the time step of writing the data.

It defines the number of Fourier modes. You just have to choose the dimensions and the matrices needed will be generated.

1.10.3 Initial condition files

1.10.3.1 ic_def.py

This file create ic.py if it does not exist with the good dimensions.

1.10.3.2 IC.nml

It initializes the 36 component of the basic functions. It gives us the structure of the vector X that represents the state of the model.

- there are 10 psi variables which are components of ψ_a the streamfunction of the atmosphere.
- there are 10 theta variables which are components of T_a the temperature of the atmosphere
- there are 8 A variables which are components of ψ_o the streamfunction of the ocean
- there are 8 T variables which are components of T_o the temperature of the atmosphere

1.10.4 Compute the tensor

1.10.4.1 inprod_analytic.py

Compute the inner products needed to compute the tensor.

1.10.4.2 aotensor.py

Compute the tensor for the bilinear equations system.

1.10.5 Step function

1.10.5.1 integrator.py

Compute RK2 integration. It is possible to call a fortran step function.

2 Instability study

2.1 Theory and computation of Lyapunov exponents and vectors

2.1.1 Lyapunov exponents

Chaos is dynamical system is characterized by sensitivity to initial conditions. The system is chaotic, there is an exponential separation from initial conditions. Lyapunov exponents is a tool to measure the growth rate of perturbations applied. We refer to the PhD of Dr Alberto Carrassi [12].

Let's consider a dynamical system :

$$\frac{dx^i}{dt} = V^i(x^1, ..., x^n) \qquad \forall i \in [1, n] \tag{2.1}$$

Let's perturb the initial conditions on the attractor $\boldsymbol{x_0}$, $\boldsymbol{x_0} + \boldsymbol{\Delta x_0}$.
We note the perturbations $\boldsymbol{y}(t) = \boldsymbol{\Delta x}(\boldsymbol{x_0}, t)$. They verify the equation in the tangent space :

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{J}(\boldsymbol{x}(t))\boldsymbol{y}(t) \tag{2.2}$$

Where $\boldsymbol{J}$ is the Jacobian :

$$J_{ij} = \frac{\partial V_i}{\partial x_j} \tag{2.3}$$

It can be solved formally as :

$$\boldsymbol{y}(t) = \boldsymbol{M}(t_0, t)\boldsymbol{y}(t_0) \tag{2.4}$$

where $\boldsymbol{M}(t_0, t) = e^{\int_0^t \boldsymbol{J}(\boldsymbol{x}(t'))\,\mathrm{d}t'}$ is the fundamental matrix.

Oseledec theorem gives the convergence of the quantities λ_i that will be called Lyapunov exponents. For an initial conditions $\boldsymbol{x_0}$, there exist a set of independent vectors $(\boldsymbol{e_i}(t_1))$ such as :

$$\lambda_i = \lim_{t \to +\infty} \frac{1}{t - t_0} ln \|\boldsymbol{M}(t_0, t)\boldsymbol{e_i}(t)\| \tag{2.5}$$

They measure the chaoticity of the system. If there is one positive exponent, the system is chaotic. They are ordered by decreasing values.

With these exponents, it is possible to compute other quantities that characterize the chaoticity of the system : the Kaplan-Yorke dimension and the Kolmogorov entropy. The Kaplan-Yorke dimension represents the size of the attractor, its formula is :

$$d = d^- + \frac{\sum_{i=1}^{d^-} \lambda_i}{\mid \lambda_{d^+} \mid} \tag{2.6}$$

where d^- and d^+ are defined such that : d^- is the last index such that $\sum_{i=1}^{d^-} \lambda_i > 0$ and $\sum_{i=1}^{d^-+1} \lambda_i < 0$ and $d^+ = d^- + 1$

The Kolmogorov entropy is the sum of the positive exponents :

$$K = \sum_{\lambda_i > 0} \lambda_i \tag{2.7}$$

2.1.2 Backward Lyapunov vectors (BLVs)

These exponents are not easily computed. Bennetin [13] gives a method to compute it with the Backward Lyapunov vectors. These vectors are computed with Gram-Schmidt orthogonalizations.

The general idea is to generate some perturbations and to propagate them with the fundamental matrix. The exponents will be the time averaged growth rate. Backward Lyapunov vectors will converge to the directions of instabilities. In practice, Gram-Schmidt orthogonalizations are needed through the process to contain the numerically exponential growth.

Algorithm 1 Computation of the BLVs - Bennetin 1980 [13]

Require: Model and Tangent Linear Model equations.

1: Generate initial conditions on the attractor after a transient time : $\boldsymbol{x_0}$
2: Compute the fundamental matrix $\boldsymbol{M}_{k:k+1}$ $\forall k = k_0, ..., k_f$ of the tangent linear equation.
3: Initialize some perturbations $\boldsymbol{perts}_0 = \boldsymbol{I_n}$. Propagate them with $\boldsymbol{M}_{k:k+1}$: $\boldsymbol{perts}'_{k+1} = \boldsymbol{M}_{k:k+1} * \boldsymbol{perts}_k$
4: Compute a QR decomposition on $\boldsymbol{perts}'_{k+1}$: $\boldsymbol{perts}'_{k+1} = \boldsymbol{Q} * \boldsymbol{R}$ where $\boldsymbol{Q}$ is an orthogonal matrix and R the upper triangular matrix.
5: The next perturbations, $\boldsymbol{perts}_{k+1}$, is $\boldsymbol{Q}$.
6: Compute the local Lyapunov exponents with the logarithm of the absolute values of the diagonal of $\boldsymbol{R}$.
7: The Lyapunov exponents are the time averaged local exponents.

2.1.3 Forward Lyapunov vectors (FLVs)

The Forward Lyapunov vectors are the equivalent of the Backward Lyapunov vectors with a propagation in the past of the perturbations.

Perturbations and initial conditions are generated and then propagated in the past with the transpose of the fundamental matrix.

Algorithm 2 Computation of FLVs - Bennetin 1980 [13]

Require: Model and Tangent Linear Model equations.

1: Generate initial conditions on the attractor after a transient time : $\boldsymbol{x_0}$
2: Compute the fundamental matrix $\boldsymbol{M}_{k:k+1}$ $\forall k = k_f, ..., k_0$ of the tangent linear equation.
3: Initialize some perturbations $\boldsymbol{perts}_0 = \boldsymbol{I_n}$. Propagate them backward with $\boldsymbol{M}_{k:k-1} : \boldsymbol{perts}'_{k-1} = \boldsymbol{M}^T_{k-1:k} * \boldsymbol{perts}_k$
4: Compute a QR decomposition on $\boldsymbol{perts}'_{k-1} : \boldsymbol{perts}'_{k-1} = \boldsymbol{Q} * \boldsymbol{R}$ where $\boldsymbol{Q}$ is an orthogonal matrix and R the upper triangular matrix.
5: The next perturbations, $\boldsymbol{perts}_{k-1}$, is $\boldsymbol{Q}$.
6: Compute the local Lyapunov exponents with the logarithm of the absolute values of the diagonal of $\boldsymbol{R}$.
7: The Lyapunov exponents are the time averaged local exponents.

2.1.4 Covariant Lyapunov vectors (CLVs)

Backward and Forward Lyapunov vectors are easy to compute because QR steps are very stable. Yet we impose that they stay orthogonal each other, so they are not "covariant" with the trajectories in a sense that they are not the natural directions of instabilities and they are not invariant under time reversal. But the subspaces that they represent are invariant.

Covariant Lyapunov vectors γ_i are the vectors that are invariant under time reversal and covariant with the trajectories. They respect for any t_1 and $t \to +\infty$

$$\|\boldsymbol{M}(t_1, t_1 \pm t)\boldsymbol{\gamma}_i(t_1)\| \approx exp(\pm\lambda_i t) \tag{2.8}$$

They can be computed by intersecting the unstable spaces generated by the BLVs and the FLVs. Several methods were implemented such as Ginelli 2007 [14], Wolf and Samelson 2007 [15] and Kuptsov and Parlitz 2012 [16]. We will focus on Kuptsov one in the section 5.2 of the article Theory and Computation of Covariant Lyapunov Vectors in 2012. Let $\boldsymbol{\Gamma}(t) = [\boldsymbol{\gamma}_1(t), \boldsymbol{\gamma}_2(t), ..., \boldsymbol{\gamma}_n(t)]$ be the covariant vectors, it respects :

$$\boldsymbol{\Gamma}(t) = \boldsymbol{\phi}^+(t)\boldsymbol{A}^+(t) = \boldsymbol{\phi}^- t\boldsymbol{A}^-(t) \tag{2.9}$$

where $\boldsymbol{A}^+(t)$ is a lower triangular matrix and $\boldsymbol{A}^-(t)$ is an upper triangular matrix.

If we note $\boldsymbol{P}(t) = [\boldsymbol{\phi}^+(t)]^T\boldsymbol{\phi}^-(t)$, it respects :

$$\boldsymbol{P}(t) = \boldsymbol{A}^+(t)(\boldsymbol{A}^-(t))^{-1} \tag{2.10}$$

In theory, a LU decomposition is enough. But the implementation of it in numpy package changes the order of the vectors. So we have to write explicitly

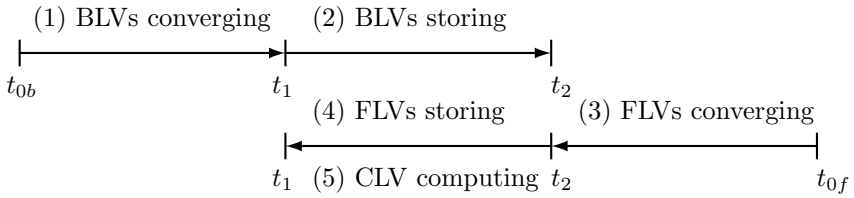
the algorithm. With the properties of the matrices, Kuptsov proves that for the j-th CLV we have to solve :

$$\boldsymbol{P}(1:j-1,1:j)\boldsymbol{A}^{-}(1:j,j)=\mathbf{0} \tag{2.11}$$

In other works, we have to find the null space of a specific matrix.

2.1.5 Algorithm of Covariant Lyapunov Vectors

Here is an algorithm to compute the CLVs. We generate one trajectory from t_{0b} to t_{0f} and the fundamental matrix. Then we generate BLVS and FLVs. BLVs and FLVs have to converge to compute properly the CLVs, this corresponds to the step 1 and 3. Then we have to solve the previous equation to find the CLVs.



Algorithm 3 Computation of CLVs - Kuptsov and Parlitz, 2012, Section 5.2 [16]

Require: The fundamental matrix $M_{k:k+1}$ $\forall t = t_{0b}, ..., t_{0f}$ of the tangent linear equation.

1: Compute the BLVs, ϕ_k^-, with QR steps : $\forall t = t_{0b}, ..., t_1 : M_{k+1:k}\phi_k^- = \phi_{k+1}^- * R_{k+1}$
2: Compute and store the BLVs : $\forall t = t_1, ..., t_2 : M_{k+1:k}\phi_k^- = \phi_{k+1}^- * R_{k+1}$
3: Compute the FLVs, ϕ_k^+, with QR steps : $\forall t = t_{0f}, ..., t_2 : M_{k:k-1}\phi_k^+ = \phi_{k-1}^+ * R_{k-1}$
4: Compute and store the FLVs : $\forall t = t_2, ..., t_1 : M_{k:k-1}\phi_k^+ = \phi_{k-1}^- * R_{k-1}$
5: Compute the CLVs, Γ_k, by solving : $\forall t = t_1, ..., t_2 : \Gamma_k = \phi_k^+ A^+ = \phi_k^- A^-$

2.2 Implementation of the CLVs

2.2.1 Code

The code used to compute the BLVs, the FLVs and their respective exponents has been done by Dr Colin Grudzien. Then I have coded the computation of the CLVs, debug it and test on different models.
Here is the python code to compute it :

```
from sympy import Matrix
# Computation of the CLVs at one moment thanks to FLVs and BLVs
def clv(FLVs,BLVs):
    [m,foo]=np.shape(FLVs)
    P=FLVs.T.dot(BLVs)
    Am=np.zeros((m,m))
    for j in range(1,m+1):
        extr_P=Matrix(P[0:j-1,0:j])
        extr_Am=extr_P.nullspace()
        Am[:,j-1]=np.concatenate([extr_Am[0],[0]*(m-j)])
    Ap=P.dot(Am)
    return BLVs.dot(Am)

# Compute the CLVs between x1 and x2. back_fms2 is used
def clv_kl(FLV_evo,BLV_evo,M_kl,rescale=1):
    [sys_dim, foo, steps] = np.shape(M_kl)
    #exponents
    LLEs=np.zeros((sys_dim,steps))
    C=np.zeros((sys_dim,sys_dim))
    expos=np.zeros(sys_dim)

    G=np.zeros((sys_dim,sys_dim,steps))

    for i in range(steps):
        #computation of CLV with BLV and FLV
        G[:,:,i]= clv(FLV_evo[:,:,i],BLV_evo[:,:,i])

        #normalize the CLV
        for j in range(sys_dim):
            G[:,j,i]=G[:,j,i]/np.linalg.norm(G[:,j,i])

        #copy to compute the local Lyapunov exponents
        Gbis=np.copy(G[:,:,i])

        #propagate the copy of CLV
        Gbis = M_kl[:, :, i].dot(Gbis)

        #compute the changes in the amplitudes
        for j in range(sys_dim):
            C[j,j]=np.linalg.norm(Gbis[:,j])

        #compute the local exponents
        C_diag = np.log(np.abs(np.diagonal(C)))
        LLEs[:, i] = C_diag[:]/rescale
        expos = expos + C_diag

    # the average value over the current run
    expos /= (steps * rescale)

    return [G,expos]
```

2.2.2 Debugging

Here is the list of property we want the CLVs to check :

- the global exponents given by the CLVs should be the same than the BLVs and FLVs
- the first BLV is the first CLV
- the last FLV is the last CLV
- the CLV corresponding to the null exponent has to be parallel to the tangent of the trajectory
- the sum of the exponents has to be equal to the divergence of the flow, ie the trace of the Jacobian (not yet implemented systematically).
- they have to be not orthogonal

2.2.3 Tests on different models

2.2.3.1 First example of Kuptsov and Parlitz

The article of Kuptsov and Parlitz [16] tests the algorithm to compute CLVs with this first example. This example take a constant matrix for the Jacobian :

$$\begin{pmatrix} 1 & -2 & 0 \\ 0 & -1 & 0 \\ 0 & 2 & -3 \end{pmatrix}$$

Parameters

Dimension :	3
Model step time h :	0.01
Tangent Linear Model step time τ :	0.1
Transient time T :	1000
Simulation time T :	1000

We **check numerically** the exponents with the three type of LVs : **1,-1,3**. the propagator corresponds to the one in the article : $\begin{pmatrix} e^{\tau} & e^{-\tau}(1-e^{2\tau}) & 0 \\ 0 & e^{-\tau} & 0 \\ 0 & e^{-3\tau}(e^{2\tau}-1) & e^{-3\tau} \end{pmatrix}$

We **check numerically** the BLVs : $\begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{1/2} & -\sqrt{1/2} \\ 0 & -\sqrt{1/2} & \sqrt{1/2} \end{pmatrix}$ and FLVs : $\begin{pmatrix} -\sqrt{1/2} & \sqrt{1/2} & 0 \\ \sqrt{1/2} & \sqrt{1/2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and CLVs : $\begin{pmatrix} 1 & -\sqrt{1/3} & 0 \\ 0 & -\sqrt{1/3} & 0 \\ 0 & -\sqrt{1/3} & 1 \end{pmatrix}$ given in the article.

2.2.3.2 Lorenz 96 model

Our first test is with the model Lorenz 96 [17] with 8 dimensions and the forcing parameter $f = 8$. The model equations are :

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + f \qquad \forall i = 1, ..., 8 \tag{2.12}$$

It is assumed that $x_{-1} = x_{N-1}$, $x_0 = x_N$ and $x_{N+1} = x_1$.

Parameters

Dimension :	8
Forcing f :	8
Model step time h :	0.01
Tangent Linear Model step time :	0.1
Transient time T :	10 000
Simulation time T :	10 000
Integration scheme :	RK4

Notation : at time k, the i-th CLV is $\vec{v}_k^{i,CLV}$, the i-th BLV is $\vec{v}_k^{i,BLV}$ and $\vec{tang}_k$ is the vector which is tangent to the trajectory.

Results

Equality between the first CLV and first BLV :	$\overline{\langle \vec{v}_k^{1,BLV} \mid \vec{v}_k^{1,CLV} \rangle} = 1.0$
Equality between the last CLV and last FLV :	$\mid \overline{\langle \vec{v}_k^{8,FLV} \mid \vec{v}_k^{8,CLV} \rangle} \mid = 1.0$
Parallelism between the tagent vector and zero exponent CLV	$\mid \overline{\langle \vec{tang}_k \mid \vec{v}_k^{3,CLV} \rangle} \mid = 0.9999$
Differences between the second CLV and second BLV :	$\overline{\langle \vec{v}_k^{2,BLV} \mid \vec{v}_k^{2,CLV} \rangle} = 0.72$

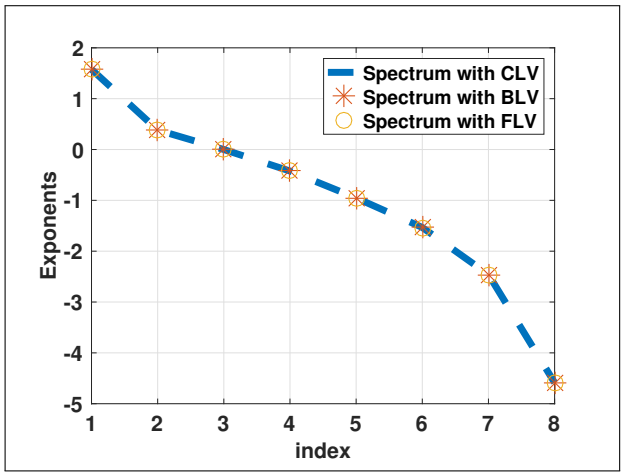


Figure 5: Lyapunov spectrum of Lorenz 96 model

The Lyapunov spectrum is computed with the three types of Lyapunov vectors.

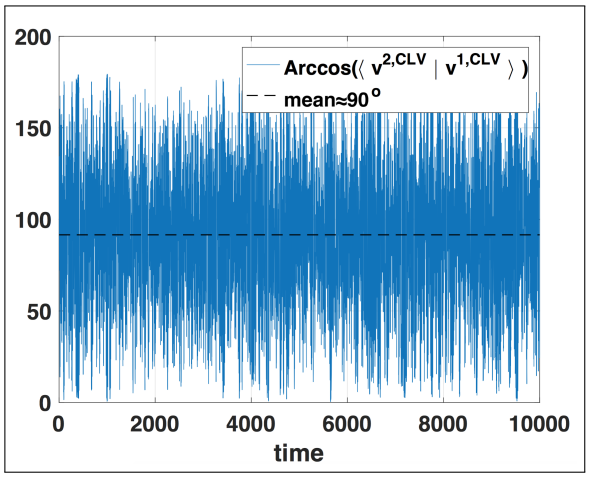


Figure 6: Projection of $\vec{v}_k^{2,CLV}$ on $\vec{v}_k^{1,CLV}$ for Lorenz 96 model

It is the time serie of the projections of the first CLV on the second CLV to check that they are different and they are not orthogonal as the first BLV and second BLV

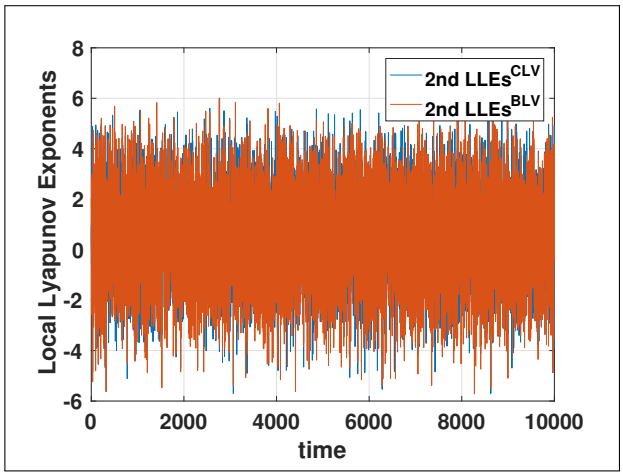


Figure 7: 2nd local exponent of Lorenz 96 model

It is the time serie of the projections of the second local Lyapunov exponents computed with the CLVs and the CLVs. They are different instantaneously but have the same mean and variance.

Interpretation

From the inner products we deduce that the first CLV and first BLV are the same. The last FLV and last CLV are the same. The third CLV is parallel to the tangent vector as expected. And the second BLV is different to the second CLV.
From the figures, we clearly see that the exponents given by the BLVs, FLVs and CLVs are the same. Absolute difference between the spectra is in the order of 10^{-3}. First CLV and second CLV are different and not orthogonal.

2.2.3.3 Generalized Henon map

This is the second example in the Kuptsov and Parlitz's article [16].

$$\begin{array}{rcl} x_1^{n+1} & = & a - [x_2^n]^2 - bx_3^n \\ x_2^{n+1} & = & x_1^n \\ x_3^{n+1} & = & x_2^n \end{array} \tag{2.13}$$

Parameters

Dimension :	3
Model step time h :	0.01
Tangent Linear Model step time :	0.1
Transient time T :	3500
Simulation time T :	3500

The BLVs, FLVs and CLVs give the spectrum **0.223, 0.186, -2.712**. The spectrum in the article is 0.225, 0.188 and -2.716. The differences between the

spectra of the CLV and FLV is 1e-5 and between the CLV an BLV is 1e-5.

Equality between the first CLV and first BLV :	$\langle \vec{v}_k^{1,BLV} \mid \vec{v}_k^{1,CLV} \rangle = 1.0$
Equality between the last CLV and last FLV :	$\overline{\mid \langle \vec{v}_k^{3,FLV} \mid \vec{v}_k^{3,CLV} \rangle \mid} = 0.988$

2.2.3.4 Lorenz 63 model

Lorenz 63 model [4] is another classical model in data assimilation.

$$\begin{aligned} \frac{dx_1}{dt} &= -\sigma x_1 + \sigma x_2 \\ \frac{dx_2}{dt} &= -x_1 x_3 + \rho x_1 - x_2 \\ \frac{dx_3}{dt} &= x_1 x_2 - \beta x_3 \end{aligned} \tag{2.14}$$

Parameters

Dimension :	3
σ :	10
ρ :	28
β :	8/3
Model step time h :	0.01
Tangent Linear Model step time :	0.1
Transient time T :	5000
Simulation time T :	5000

The BLVs, FLVs and CLVs give the spectrum : **0.878, 0. (1e-4), -12.54** . The spectrum in the article of Wolfe is 0.91, 0 and -14.58. The differences between the spectra of the CLV and FLV is 1e-3 and between the CLV an BLV is 1e-3 at the maximum.

Equality between the first CLV and first BLV :	$\overline{\langle \vec{v}_k^{1,BLV} \mid \vec{v}_k^{1,CLV} \rangle} = 1.0$
Equality between the last CLV and last FLV :	$\overline{\mid \langle \vec{v}_k^{3,FLV} \mid \vec{v}_k^{3,CLV} \rangle \mid} = 1.0$
Parallelism between the tagent vector and zero exponent CLV	$\overline{\mid \langle \vec{tang}_k \mid \vec{v}_k^{2,CLV} \rangle \mid} = 0.9999$

2.2.3.5 Lorenz 63 model with 2 compartments

$$\begin{aligned}
\frac{dx_t}{dt} &= a(y_t - x_t) - c(SX + k) \\
\frac{dy_t}{dt} &= rx_t - y_t - x_t z_t + c(SY + k) \\
\frac{dz_t}{dt} &= x_t y_t - bz_t + c_z Z \\
\frac{dX}{dt} &= \tau(a(Y - X) - c(x_t + k)) \\
\frac{dY}{dt} &= \tau(rX - Y - SXZ + c(y_t + k)) \\
\frac{dZ}{dt} &= \tau(SXY - bZ - c_z z_t)
\end{aligned} \tag{2.15}$$

Parameters

Dimension :	6
a :	10
r :	28
b :	8/3
S :	1
k :	0
c_z :	1
τ :	0.1
Model step time h :	0.01
Tangent Linear Model step time :	0.1
Transient time T :	5000
Simulation time T :	5000
Integration scheme :	RK4

The BLVs, FLVs and CLVs give the spectrum : The differences between the spectra of the CLV and FLV is and between the CLV an BLV is at the maximum.

2.3 Results on Lorenz 63 with 2 compartments

We test different values of τ to see how the different time scales affect the spectra. If we increase τ, the Kolmogorov entropy increases but the Kaplan Yorke dimension decreases. If we look the number of positive, zero and negative exponents, if we increase τ there more positive exponents, so the regime is more chaotic.

model configurations	$\tau = 0.1$	$\tau = 0.25$	$\tau = 0.5$	$\tau = 1$
Positive $\lambda_i \in [10^{-3}; 2]$	2	2	3	3
Zero Exponent ($[-10^{-5}; 10^{-3}]$)	1	1	1	1
Negative $\lambda_i \in [10^{-3}; -10^{2}]$	3	3	2	2
Kolmogorov entropy	0.9910	1.1261	1.3481	1.7965
Kaplan Yorke dimension	4.6798	4.3091	4.1851	4.1246

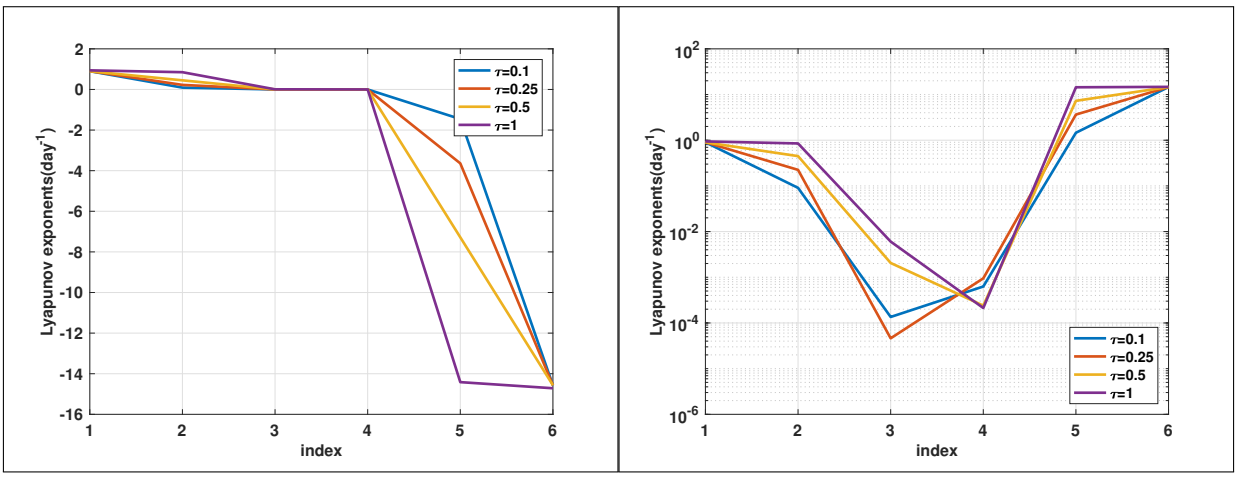


Figure 8: Lyapunov spectrum of Lorenz 63 with 2 compartments with different τ The Lyapunov spectra are computed with different values of τ the time scale difference in the system

We compute the CLVs in different cases. The difference is clearly visible when the model is uncoupled.

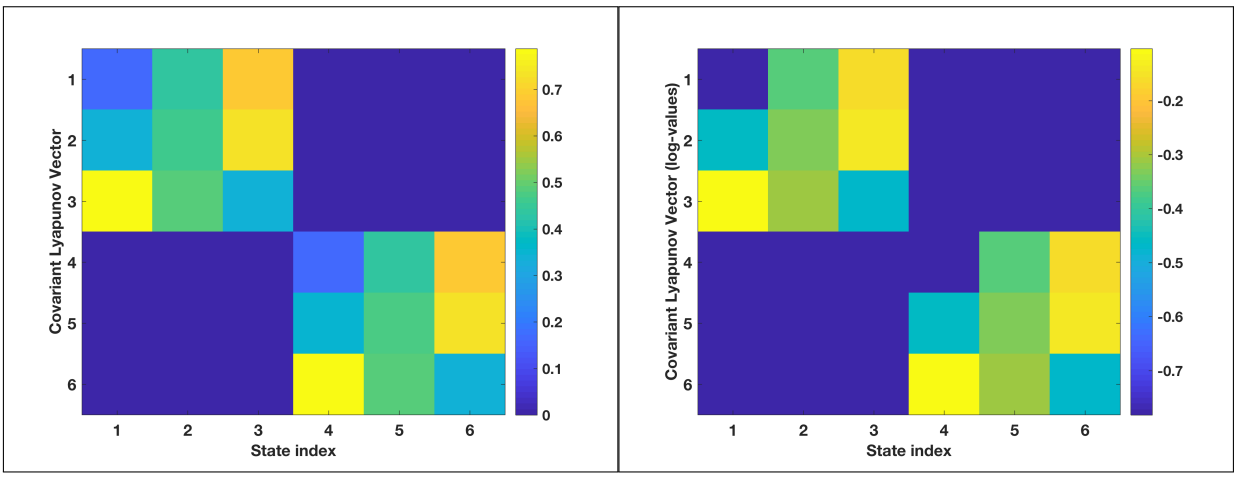


Figure 9: Lyapunov CLVs of Lorenz 63 - 2 comp. - c,tau=0,1

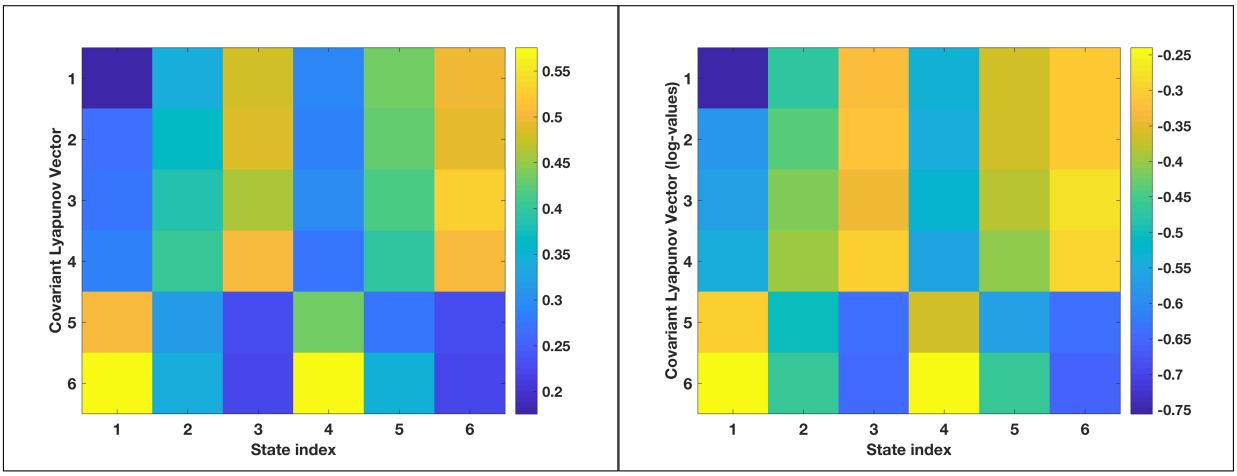


Figure 10: Lyapunov CLVs of Lorenz 63 - 2 comp. - c,tau=0.15,1

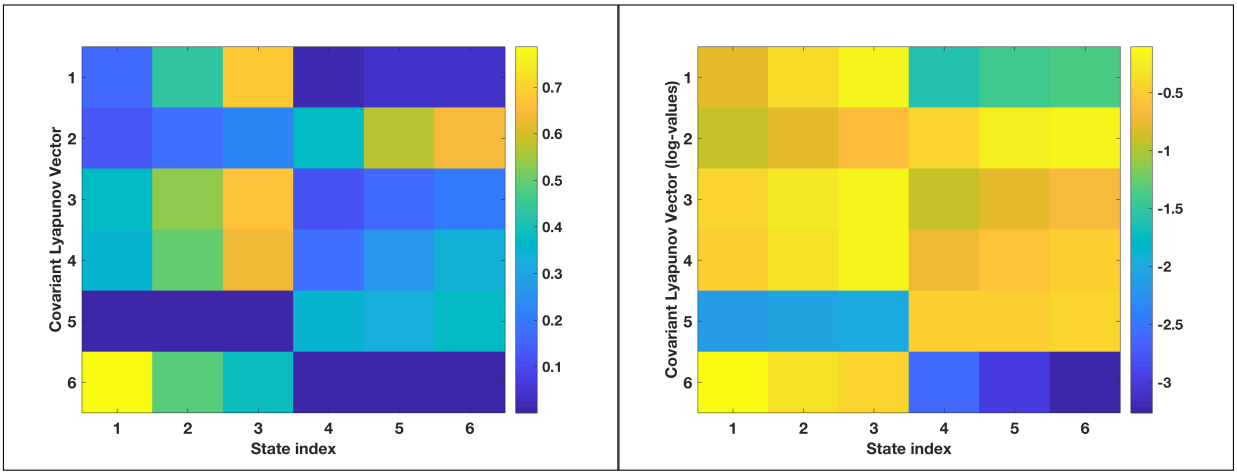


Figure 11: Lyapunov CLVs of Lorenz 63 - 2 comp. - c,tau=0.15,0.25

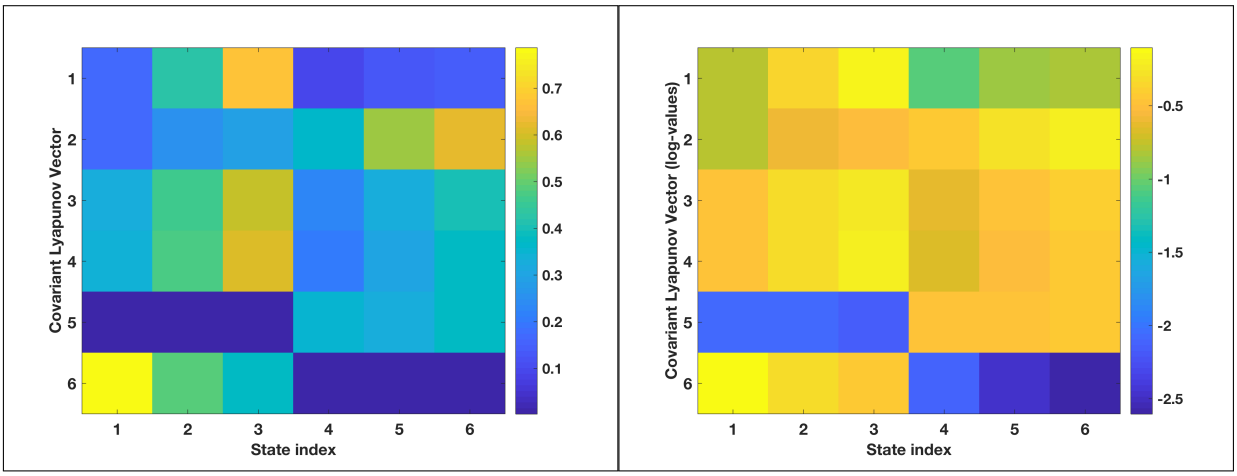


Figure 12: Lyapunov CLVs of Lorenz 63 - 2 comp. - c,tau=0.15,0.5

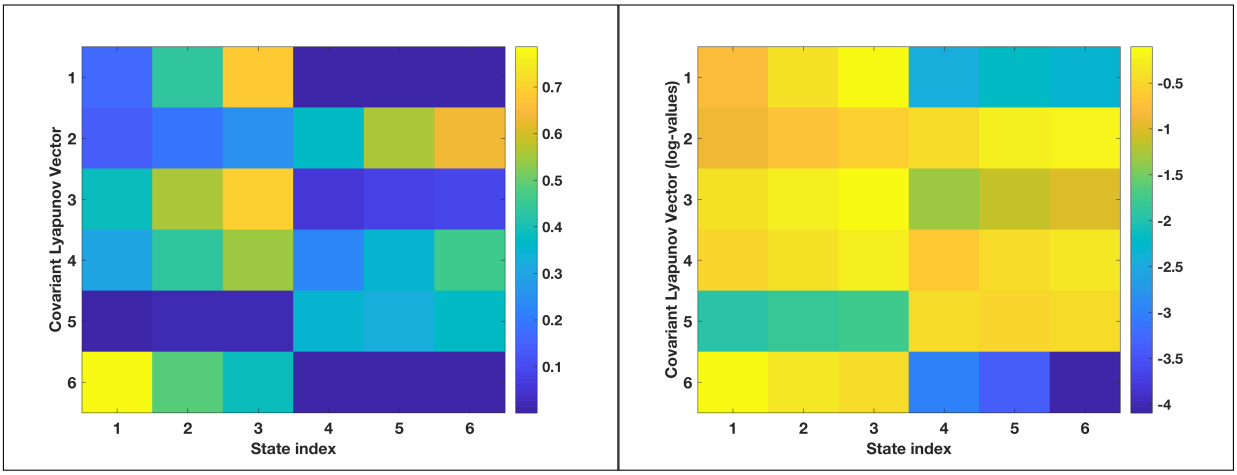


Figure 13: Lyapunov CLVs of Lorenz 63 - 2 comp. - c,tau=0.15,0.1

2.4 Results on MAOOAM

2.4.1 Study of the Lyapunov spectrum

There are six regimes :

- 36 modes with weakly or strongly coupling,
- 52 modes, with new ocean modes, with weakly or strongly coupling,
- 56 modes, with new atmosphere modes, with weakly or strongly coupling.

We compute Lyapunov spectrum with the Fortran code.

model configuration	36 weakly	52 weakly	56 weakly	36 strongly	52 strongly	56 strongly
Positive $\lambda_i \in [10^{-2}; 1]$	3	3	5	2	2	1
Small Positive $\lambda_i \in [10^{-2}; 10^{-5}]$	3	7	3	2	4	4
Zero Exponent ($[-10^{-5}; 10^{-5}]$)	1	1	2	1	1	1
Small negative $\lambda_i \in [-10^{-2}; -10^{-5}]$	13	25	12	11	25	13
Negative $\lambda_i \in [-1; -10^{-2}]$	16	16	34	20	20	37
Kolmogorov entropy	0.498	0.528	0.459	0.139	0.060	0.029
Kaplan Yorke dimension	25.06	41.03	28.42	20.29	33.35	19.32

Notation λ_i is a Lyapunov exponent.
Figure 14 shows the Lyapunov spectrum of 6 configurations of the system.

- As expected, the weakly coupled configurations are more chaotic than the strongly coupled configurations. It is true in terms of values of the positive Lyapunov exponents, in terms of number of positive exponents and in terms of Kolmogorov entropy. The weakly coupled configurations have more neutral exponents than the strongly ones. Strongly configurations have a little more negative exponents.

- 36 and 52 configuration have approximately the same structure of spectrum for positive exponents. As expected, the ocean modes that are added participate to new neutral exponents (two times more). 36 and 52 weakly configurations have the same Kolmogorov entropy as one can expect. For the strongly configurations, 52 have half of the entropy of the 36. So, the new ocean modes have larger effect on the stability of the system in the strongly.

- 56 configurations, where atmosphere modes are added, have more positive exponents than the 36 configurations in the weakly case but not clearly in the strongly case. The Kolmogorov entropy is the same or smaller. It is not coherent with the statement that new atmosphere modes should create more chaoticity...

- According to the Figure 2, with a threshold value at $10^{-5} day^{-1}$, there are 1 real null exponent for all regimes, except for the 56 weakly coupled regime where there are 2 real null exponents. The variance of these null exponents are the smallest.

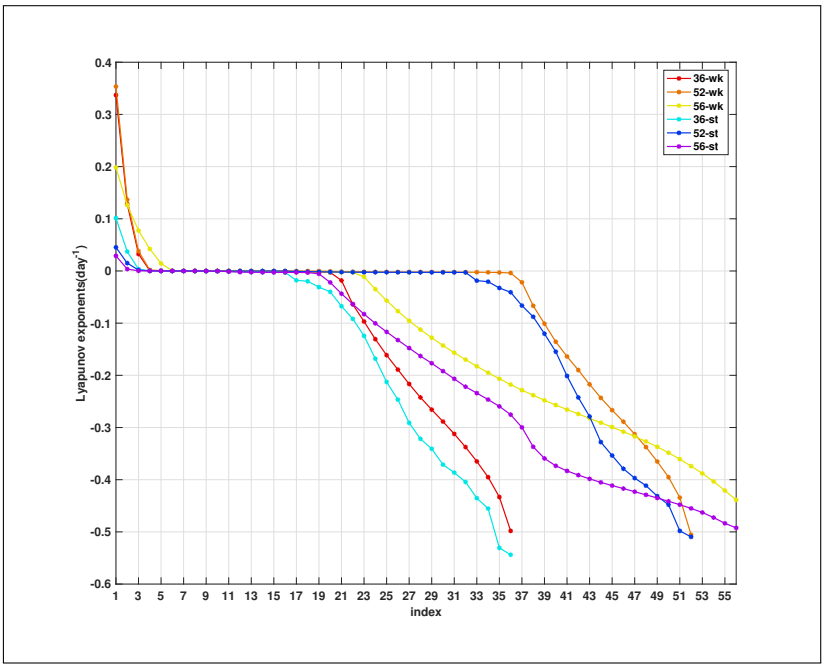


Figure 14: Lyapunov spectra of MAOOAM in different regimes

The Lyapunov spectra are computed in the different regimes 36wk, 52wk and 56wk correspond to weakly coupled configurations and 36st, 52st and 56st correspond to strongly coupled configurations. The number corresponds to the dimension. The 52 configurations have more neutral modes because there are more Fourier ocean modes. The 56wk have more positive exponents, it has more atmosphere modes.

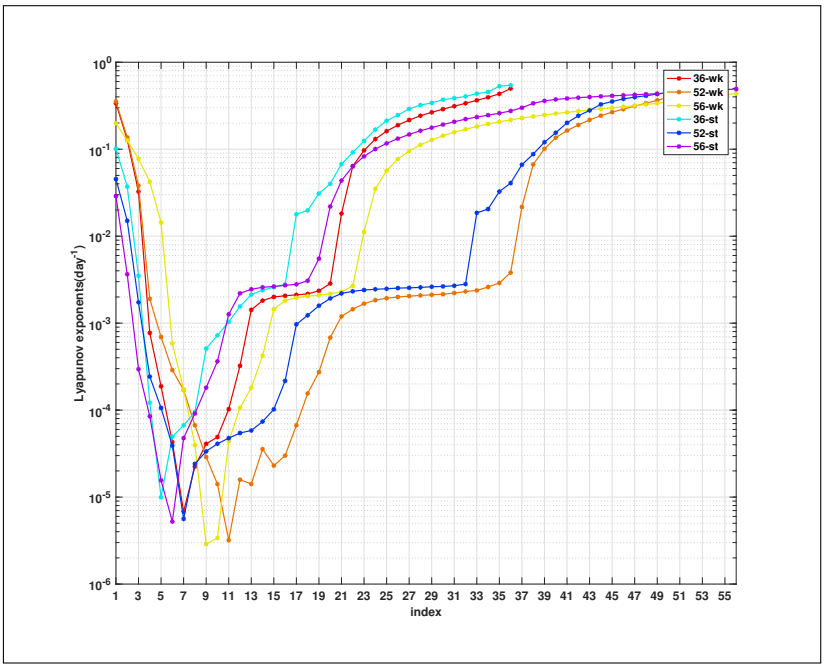


Figure 15: Lyapunov spectra of MAOOAM - log of absolute values

The Lyapunov spectra are computed in the different regimes 36wk, 52wk and 56wk correspond to weakly coupled configurations and 36st, 52st and 56st correspond to strongly coupled configurations. With the log-values, it is easier to see the differences of values and to identify the zero with a threshold.

2.4.2 CLVs

The CLVs of MAOOAM computed with python have the same pattern as in the article Vannitsem and Lucarini 2016 [18]. The CLVs have their biggest projections on the atmosphere. Then there are projections on the ocean temperature.

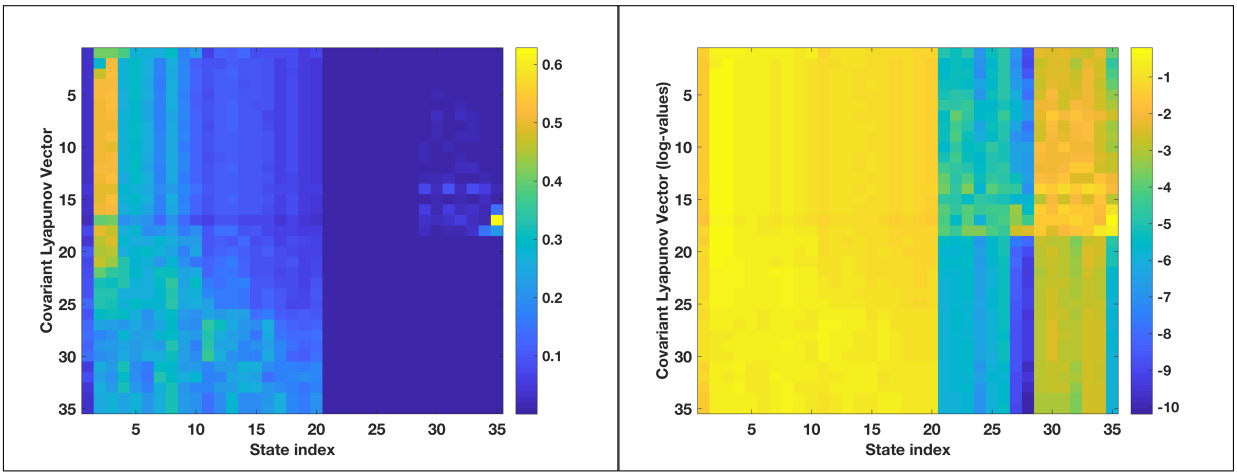


Figure 16: Lyapunov CLVs for MAOOAM - Regime 36wk

The Lyapunov CLVs are computed in the different regimes 36wk, 52wk and 56wk correspond to weakly coupled configurations and 36st, 52st and 56st correspond to strongly coupled configurations. With the log-values on the right panel, it is easier to see the differences of values. The CLVs projections are mainly on the atmosphere (1-20) which is more chaotic, they have projections on the ocean temperatures too (29-36).

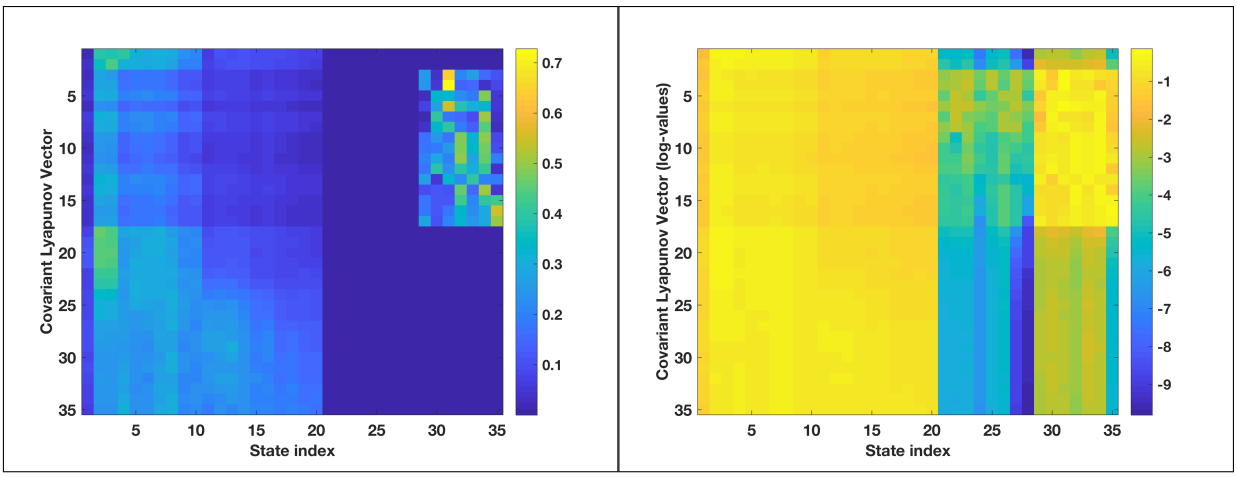


Figure 17: Lyapunov CLVs for MAOOAM - Regime 36st

The Lyapunov CLVs are computed in the different regimes 36wk, 52wk and 56wk correspond to weakly coupled configurations and 36st, 52st and 56st correspond to strongly coupled configurations. With the log-values on the right panel, it is easier to see the differences of values. The difference with Figure 16 is that the projections on the ocean temperature are bigger. It may be dued to the bigger coupling.

Note : we compute only the first 35 variables of the system because a bug is induced by this variable (the last one). The results are still relevant because

this variable is always null.

3 Coupled Data Assimilation

Data assimilation's goal is to predict the future step with the physical model and the observations. It uses in particular a different version of the Kalman Filter, the Ensemble Kalman Filter.

We will refer to the book of Dr Marc Bocquet [1].

3.1 Theory of Data Assimilation

3.1.1 The Extended Kalman Filter (EKF)

In the article in 1960 [19], Kalman defines the filter. There are two steps : the forecast step and the analysis step. The forecast step is the estimation of the state by the model. The analysis step is the optimal estimation of the state thanks to the model and the observations. The hypothesis is that we represent our estimation error with a Gaussian noise, the goal is then to estimate the mean and the variance. The extended Kalman filter is a version of the Kalman filter that does not require the evolution and observations models to be linear.

The state system has the dimension m and the observation state has the dimension p.

Truth and observations equations

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{\mathcal{M}}_k \boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}^q_{k-1} \\ \boldsymbol{y}_k &= \boldsymbol{\mathcal{H}}_k \boldsymbol{x}_k + \boldsymbol{\epsilon}^o_k \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^m$ is the true state of the system at time t_k, $\boldsymbol{y}_k \in \mathbb{R}^p$ is the observation state, $\boldsymbol{\mathcal{M}}_k$ is the evolution model, $\boldsymbol{\mathcal{H}}_k$ is the observation model, $\boldsymbol{\epsilon}^q_k$ is the model error which is unbiased, uncorrelated in time (white noise) and of model error covariance $\boldsymbol{Q}_k$, $\boldsymbol{\epsilon}^o_k$ is the observation error with the same hypotheses and $\boldsymbol{R}_k$ as error covariance matrix. There are no correlation between $\boldsymbol{\epsilon}^q_k$ and $\boldsymbol{\epsilon}^o_k$.

Forecast step equations

$$\begin{aligned} \boldsymbol{x}^f_k &= \boldsymbol{\mathcal{M}}_k \boldsymbol{x}^f_{k-1} + \boldsymbol{\epsilon}^q_{k-1} \\ \boldsymbol{P}^f_k &= \boldsymbol{M}_k \boldsymbol{P}^a_{k-1} \boldsymbol{M}^T_k + \boldsymbol{Q}_{k-1} \end{aligned} \tag{3.2}$$

where $\boldsymbol{P}^f_k$ is the forecast error covariance matrix, $\boldsymbol{P}^a_k$ is the analysis error covariance matrix, $\boldsymbol{M}_k$ is the tangent linear model of $\boldsymbol{\mathcal{M}}_k$.

Analysis step equations

$$\begin{aligned} \boldsymbol{x}^a_k &= \boldsymbol{x}^f_k + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{\mathcal{H}}_k \boldsymbol{x}^f_k) \\ \boldsymbol{P}^f_k &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}^f_k \end{aligned} \tag{3.3}$$

where $\boldsymbol{x}^a_k$ the analysis state, $\boldsymbol{K}_k = \boldsymbol{P}^f_k \boldsymbol{H}^T_k (\boldsymbol{H}_k \boldsymbol{P}^f_k \boldsymbol{H}^T_k + \boldsymbol{R}_k)^{-1}$ is the gain and $\boldsymbol{\delta}_k = \boldsymbol{y}_k - \boldsymbol{\mathcal{H}}_k \boldsymbol{x}^f_k$ is called the innovation.

The size of operational meteorological models is huge. The computational cost to compute the matrices is then not affordable.

3.1.2 The Ensemble Kalman Filter (EnKF)

The ensemble Kalman filter is one solution to the problem of computational cost. The algorithm takes the idea of Monte Carlo method by sampling some members to evaluate the state. It was proposed by G. Evensen [2]. It proves its performances for the last 20 years and has become very popular. There is a stochastic and a deterministic formulation.

The algorithm generates an ensemble of size N, $\boldsymbol{x}_{1:N} = \{\boldsymbol{x}_i; i = 1, ..., N\}$ to estimate the true state. We focus on one analysis at a given time k, index k is dropped. Subscript index i now correspond to ensemble index. These ensemble "members" are gathered in an ensemble matrix :x

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N] \tag{3.4}$$

The analysis is performed on each member of the analysis :

$$\boldsymbol{x}_i^a = \boldsymbol{x}_i^f + \boldsymbol{K}(\boldsymbol{y}_i - \mathcal{H}(\boldsymbol{x}_i^f)) \tag{3.5}$$

where $i = 1, ..N$ is the member index.
We have to compute the gain $\boldsymbol{K} = \boldsymbol{P}^f\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}^f\boldsymbol{H}^T + \boldsymbol{R})^{-1}$. So we have to compute the covariance matrix. We define the mean :

$$\overline{\boldsymbol{x}}^{f,a} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i{}^{f,a} \tag{3.6}$$

We define the anomaly matrices $\mathbf{X}^{f,a}$ which is difference between the ensemble and its mean along the members :

$$[\boldsymbol{X}^{f,a}]_{ij} = \boldsymbol{X}_j^i - \overline{x}_j \tag{3.7}$$

where j index is the coordinate index and i the member index.
We can estimate the forecast and analysis error covariance matrix (and then the gain) as :

$$\boldsymbol{P}^{f,a} = \frac{1}{N-1}\boldsymbol{X}^{f,a}(\boldsymbol{X}^{f,a})^T \tag{3.8}$$

3.1.3 The stochastic formulation vs the deterministic EnKF

With the previous formulation, the error covariance matrices are underestimating the error. It leads to filter divergence. A nice solution is to perturb the observation with a white noise. This is the stochastic formulation. Observations become :

$$\boldsymbol{y}_i = \boldsymbol{y} + \boldsymbol{u}_i \tag{3.9}$$

with $i = 1, ..., N$ and $\boldsymbol{u}_i$ following a normal law with $\mathbf{0}$ mean and covariance $\boldsymbol{R}$.

Yet this formulation introduces some numerical noise that can affect the performances. An alternative idea is to follow the square root approach. It is called deterministic ensemble Kalman Filter because there are no perturbations. Both formulations are equivalent.

The gain is then computed with $\boldsymbol{P}^f = \boldsymbol{X}^f(\boldsymbol{X}^f)^T$ and a known pre-determined $\boldsymbol{R}$. We have to find ω^a the optimal coefficient vectors in the analysis step.

$$\boldsymbol{x}^a = \boldsymbol{x}^f + \boldsymbol{X}^f\omega^a \tag{3.10}$$

We find :

$$\omega^a = (\boldsymbol{I}_N + (\boldsymbol{Y}^f)^T\boldsymbol{R}^{-1}\boldsymbol{Y}^f)^{-1}(\boldsymbol{Y}^f)^T\boldsymbol{R}^{-1}\delta \tag{3.11}$$

and :

$$\begin{aligned}\boldsymbol{P}^a &= \boldsymbol{X}^f(\boldsymbol{I}_N - (\boldsymbol{Y}^f)^T(\boldsymbol{Y}^f(\boldsymbol{Y}^f)^T + \boldsymbol{R})^{-1}\boldsymbol{Y}^f(\boldsymbol{X}^f)^T \\ \boldsymbol{X}^a &= \boldsymbol{X}^f(\boldsymbol{I}_N + (\boldsymbol{Y}^f)^T\boldsymbol{R}^{-1}\boldsymbol{Y}^f)^{-1/2}\boldsymbol{U}\end{aligned} \tag{3.12}$$

with $\boldsymbol{U}$ an arbitrary orthogonal matrix

3.1.4 Localization and Inflation

The filter is cheaper than the EKF. To represent a system with a few members imply some problems of sampling and rank-deficiency. Two tricks are done to overcome these difficulties.

The idea of localization is that in geophysical systems, two distant observations are weakly correlated. It is then possible to localize the analysis.

The error covariance matrices are evaluated with an ensemble of limited size. This often leads to sampling errors and spurious correlations. However small are the residual errors, they will accumulate and they will carry over to the next cycles of the sequential EnKF scheme. As a consequence, there is always a risk that the filter may ultimately diverge. One way around is to inflate the error covariance matrix by a factor λ^2 slightly greater than 1 before or after the analysis. For instance, after the analysis,

$$\boldsymbol{P}^a \to \lambda^2\boldsymbol{P}^a \tag{3.13}$$

It is possible to inflate the ensemble instead.

$$\boldsymbol{x}_i^a \to \overline{\boldsymbol{x}}^a + \lambda(\boldsymbol{x}_i^a - \overline{\boldsymbol{x}}^a) \tag{3.14}$$

3.1.5 The finite-size EnKF (EnKF-N)

This version of EnKF automatically computes the optimized inflation at each observation step.

Algorithm 4 EnKF-N :algorithm - Bocquet 2012 [3]

Require: The forecast ensemble $\{\boldsymbol{x_k}\}_{k=1,\dots,N}$, the observations $\boldsymbol{y}$ and error covariance matrix $\boldsymbol{R}$

1: Compute the mean $\overline{\boldsymbol{x}}$ and the anomalies $\boldsymbol{A}$ from $\{\boldsymbol{x_k}\}_{k=1,\dots,N}$.
2: Compute $\boldsymbol{Y} = \boldsymbol{HA}$, $\boldsymbol{\delta} = \boldsymbol{y} - \boldsymbol{H}\overline{\boldsymbol{x}}$
3: Find the minimum :
4: $\omega_a = min_w\{(\delta - \boldsymbol{Y\omega})^T\boldsymbol{R}^{-1}(\delta - \boldsymbol{Y\omega}) + N * ln(\epsilon_N + \boldsymbol{\omega}^T\boldsymbol{\omega})\}$
5: Compute $\boldsymbol{x}^a = \overline{\boldsymbol{x}} + \boldsymbol{A\omega}_a$.
6: Compute $\boldsymbol{\Omega}_a = (\boldsymbol{Y}^T\boldsymbol{R}^{-1}\boldsymbol{Y} + N\frac{(\boldsymbol{\epsilon}_N + \boldsymbol{\omega}_a^T\boldsymbol{w}_a)\boldsymbol{I}_N - 2\boldsymbol{\omega}_a\boldsymbol{\omega}_a^T}{(\boldsymbol{\epsilon}_N + \boldsymbol{\omega}_a\boldsymbol{\omega}_a^T)^2})^{-1}$
7: Compute $\boldsymbol{W}^a = \{(N-1)\boldsymbol{\Omega}_a\}^{1/2}\boldsymbol{U}$
8: $\boldsymbol{x}_k^a = \boldsymbol{x}^a + \boldsymbol{AW}_k^a$

3.1.6 Coupled Data Assimilation formulation

We try to formalize the problem of coupled data assimilation.

Consider the following system of coupled Ordinary Differential Equations (ODEs) as a toy example of geophysical systems with multiple temporal scales :

$$\begin{aligned} \boldsymbol{x}^{\cdot} &= \epsilon F(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{y}^{\cdot} &= G(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \tag{3.15}$$

where $\boldsymbol{x} \in \mathbb{R}^{m_x}$, $\boldsymbol{y} \in \mathbb{R}^{m_y}$, $F : \mathbb{R}^{m_x} \to \mathbb{R}^{m_x}$ and $G : \mathbb{R}^{m_y} \to \mathbb{R}^{m_y}$ are both smooth functions bounded as $O(1)$.

The vector $\boldsymbol{x}$ describes the slow sub-system (say the ocean), with $\epsilon << 1$, while y describes the fast one (say the atmosphere).

3.2 Formalism in MAOOAM

We give the formalism for the version of MAOOAM with 36 dimensions.

We define the spread $SPD^{f,a}$ as the square root of the mean of the diagonal of $\mathbf{P}^{f,a}$. We choose to split the spread on the 4 different quantities : atmosphere streamfunction and temperature, ocean streamfunction and temperature :

$$SPD^{f,a}_{st,at} = \sqrt{\frac{1}{10}\sum_{j=1}^{10}(\mathbf{P}^{f,a})_{jj}} = \sqrt{\frac{1}{10}\sum_{j=1}^{10}\frac{(\sum_{i=1}^{N}(\mathbf{X}_j^i - \overline{x}_j))^2}{N-1}}$$

$$SPD^{f,a}_{T,at} = \sqrt{\frac{1}{10}\sum_{j=11}^{20}(\mathbf{P}^{f,a})_{jj}} = \sqrt{\frac{1}{10}\sum_{j=11}^{20}\frac{(\sum_{i=1}^{N}(\mathbf{X}^i_j - \overline{x}_j))^2}{N-1}}$$

$$SPD^{f,a}_{st,oc} = \sqrt{\frac{1}{8}\sum_{j=21}^{28}(\mathbf{P}^{f,a})_{jj}} = \sqrt{\frac{1}{8}\sum_{j=21}^{28}\frac{(\sum_{i=1}^{N}(\mathbf{X}^i_j - \overline{x}_j))^2}{N-1}}$$

$$SPD^{f,a}_{T,oc} = \sqrt{\frac{1}{8}\sum_{j=29}^{36}(\mathbf{P}^{f,a})_{jj}} = \sqrt{\frac{1}{8}\sum_{j=29}^{36}\frac{(\sum_{i=1}^{N}(\mathbf{X}^i_j - \overline{x}_j))^2}{N-1}}$$

Then we define the Root Mean Square Error (RMSE), which is the quadratic mean of the "real" error between the the mean of the ensemble and the truth.

$$RMSE^{f,a}_{st,at} = \sqrt{\frac{1}{10}\sum_{j=1}^{10}(\overline{x}^{f,a}_j - x^t_j)^2}$$

$$RMSE^{f,a}_{T,at} = \sqrt{\frac{1}{10}\sum_{j=11}^{20}(\overline{x}^{f,a}_j - x^t_j)^2}$$

$$RMSE^{f,a}_{st,oc} = \sqrt{\frac{1}{8}\sum_{j=21}^{28}(\overline{x}^{f,a}_j - x^t_j)^2}$$

$$RMSE^{f,a}_{T,oc} = \sqrt{\frac{1}{8}\sum_{j=29}^{36}(\overline{x}^{f,a}_j - x^t_j)^2}$$

Finally we define σ^{obs} as the observation error standard deviation. We split it on the four physical quantities. $\mathbf{cv} \in \mathbb{R}^{36}$ is the climate variance :

$$\sigma^{obs}_{st,at} = \sqrt{\frac{1}{10}\sum_{j=1}^{10}\mathbf{cv}_j}$$

$$\sigma^{obs}_{T,at} = \sqrt{\frac{1}{10}\sum_{j=11}^{20}\mathbf{cv}_j}$$

$$\sigma^{obs}_{st,oc} = \sqrt{\frac{1}{8}\sum_{j=21}^{28}\mathbf{cv}_j}$$

$$\sigma^{obs}_{T,oc} = \sqrt{\frac{1}{8}\sum_{j=29}^{36}\mathbf{cv}_j}$$

3.3 Protocol for MAOOAM

Data assimilation on the previous regimes (36-weakly, 36-strongly, 52-weakly, 52-strongly, 56-weakly, 56-strongly) will be compared.

Our integral parameters are a step time of 16 minutes and a simulation time of one year. The method is twin experiments. The truth are initialized on the attractor. The climate variance $\mathbf{cv} \in \mathbb{R}^{36}$, is computed, it corresponds to the natural variance of the system. For the observations, the standard deviation error is 1% of $\sqrt{\mathbf{cv}}$. A finite-size Ensemble Kalman filter, termed EnKF-N [3], is applied with an initial conditions variance of the ensemble of 1% of $\mathbf{cv}$. A standard data assimilation scheme is studied. Then the size of the ensemble is changed changing which box is observed. Then global observation frequency is changed. And finally observation frequency of the atmosphere and then the ocean is changed separately.

3.4 Results with MAOOAM

3.5 General results

The default set of parameters is an ensemble with 15 members and a observation step time of 24 hours.

3.5.0.1 Remarks about the size of the ensemble

We change the size of the ensemble if the ocean or atmosphere or both are changed. If only atmosphere is observed, atmosphere and ocean need more members to be corrected but the patterns are the same. So ocean can handle with information only from the atmosphere. If only ocean is observed, atmosphere generally saturates. Atmosphere correction needs absolutely information from the atmosphere to prevent from saturating.

3.5.0.2 Remarks about the observation frequency

We change the observation frequency and if the ocean or atmosphere or both are changed. There are different behaviors for the ocean and the atmosphere regarding the observation frequency. It may be due to difference of time scales. Sometimes, the RMSE is not growing with the observation frequency. It could stay constant, if information is given less frequently, the system does not see the difference. And sometimes too much information too frequently is not good. For example with full observation and 1h observation frequency atmosphere RMSE is larger than with observations only from the atmosphere. It means that for high frequency of observation, ocean could disturb the atmosphere.

3.5.0.3 Remarks about the ocean observation frequency

The ocean does not need as much information as the atmosphere, it is possible to decrease the frequency of observation of the ocean keeping the atmosphere one constant because the ocean is more stable.

3.5.0.4 Strongly coupled and weakly coupled data assimilation : two schemes of data assimilation

Two schemes of data assimilation can be chosen in data assimilation : the strongly coupled one and the weakly coupled one. The first one is when the assimilation is done one the whole system (ocean and atmosphere here). The second one update the atmosphere and the ocean separately, and the effects of the assimilation are visible after theses analysis because the system is coupled.

3.6 Time series

3.6.0.1 Regime 36wk

We show different global behaviors of the Data Assimilation by changing the size of the ensemble in the Regime 36wk. The RMSE, the SPREAD and the initial observation error σ are plotted. The goal is that the SPREAD estimates well the RMSE and that they are both under the initial error σ. We clearly see the difference. The first figure is a data assimilation RMSE time serie with only 5 members, the RMSE saturates and the SPREAD does not see it. The second figure is with 15 members, the SPREAD estimates well the RMSE and they are below the initial error.

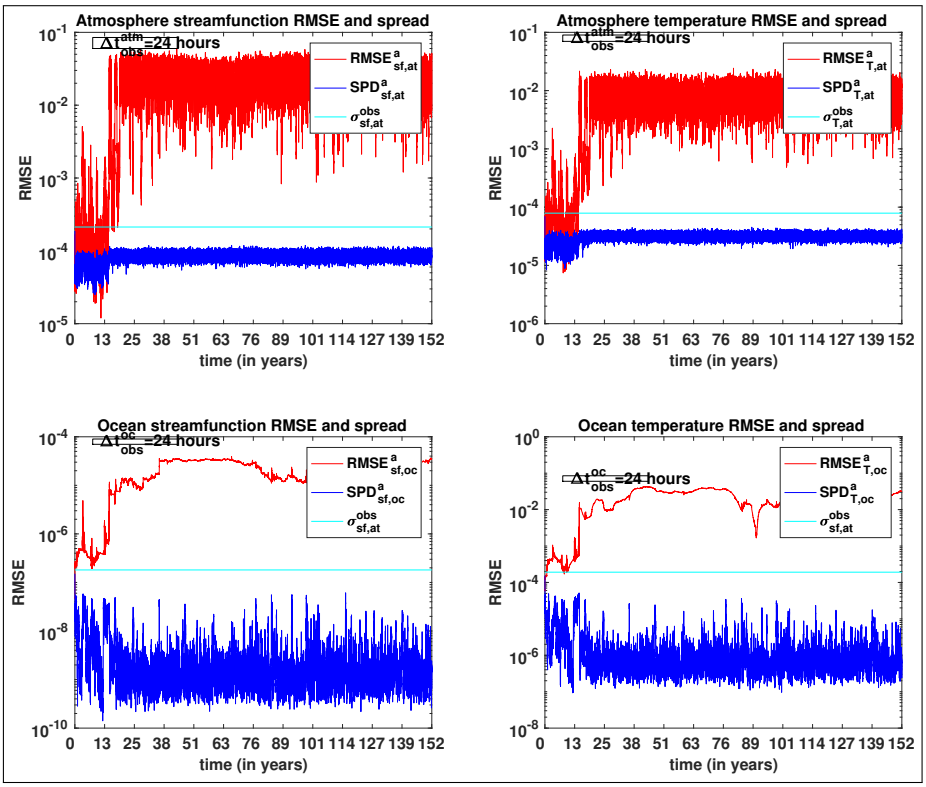


Figure 18: Data Assimilation of MAOOAM - Time series of RMSE and SPREADS - Regime 36wk - 5 members

This is a 152 years assimilation of MAOOAM in the regime with 36 dimensions and weak coupling between ocean and atmosphere. The assimilation uses EnKF-N with a size of 3, that is why the error (RMSE) saturates and the spread underestimates it. The system needs more members.

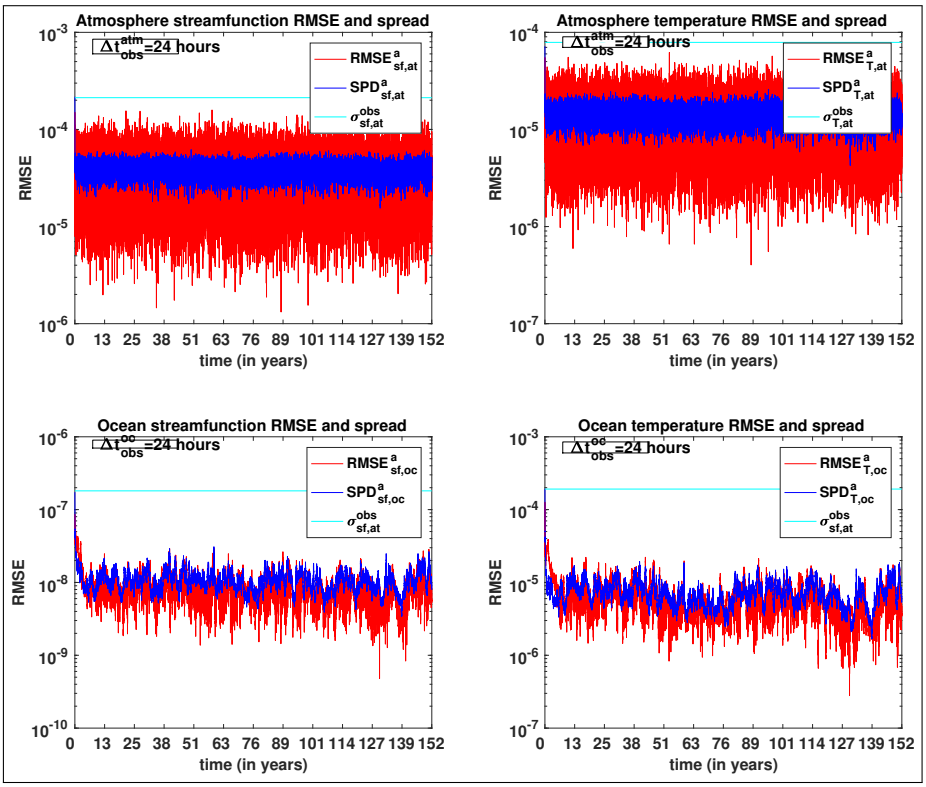


Figure 19: Data Assimilation of MAOOAM - Time series of RMSE and SPREADS - Regime 36wk - 16 members

This is a 152 years assimilation of MAOOAM in the regime with 36 dimensions and weak coupling between ocean and atmosphere. The assimilation uses EnKF-N with a size of 16, that is why the error (RMSE) estimates well the spread and both are below the initial error. The system has enough members to assimilate.

3.6.0.2 Regime 36st

We have the same global behaviors as the regime 36wk. The low-frequency variability is visible in the figures.

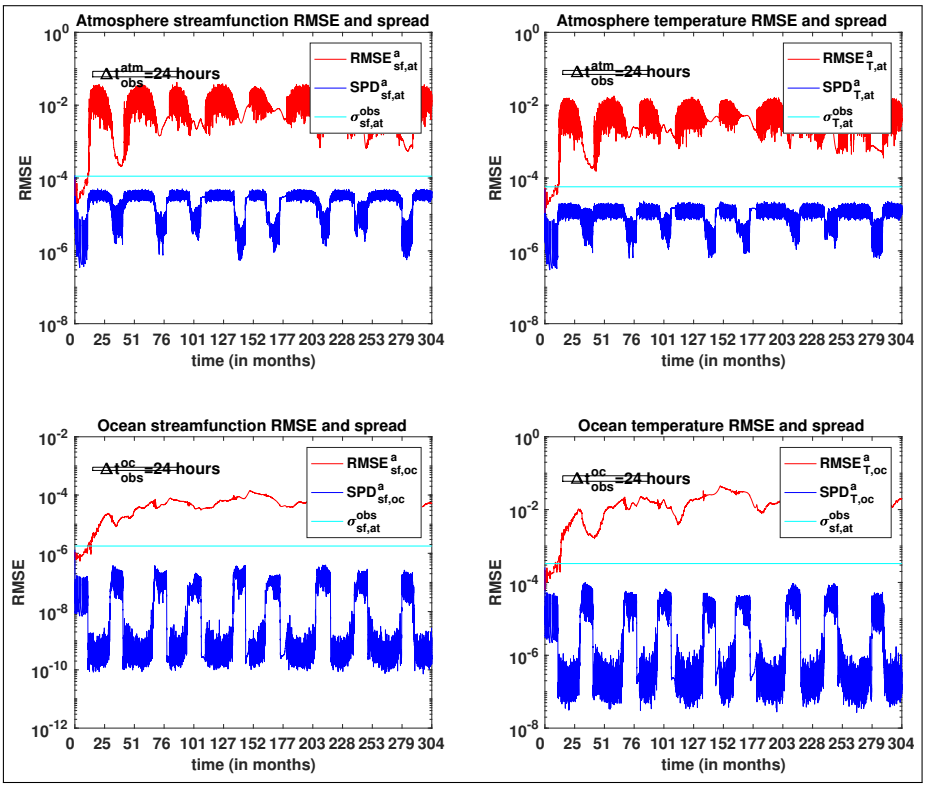


Figure 20: Data Assimilation of MAOOAM - Time series of RMSE and SPREADS - Regime 36st - 3 members

This is a 304 years assimilation of MAOOAM in the regime with 36 dimensions and strong coupling between ocean and atmosphere. The assimilation uses EnKF-N with a size of 3, that is why the error (RMSE) saturates and the spread underestimates it. The system needs more members. The low frequency variability is visible, there are stable and unstable parts.

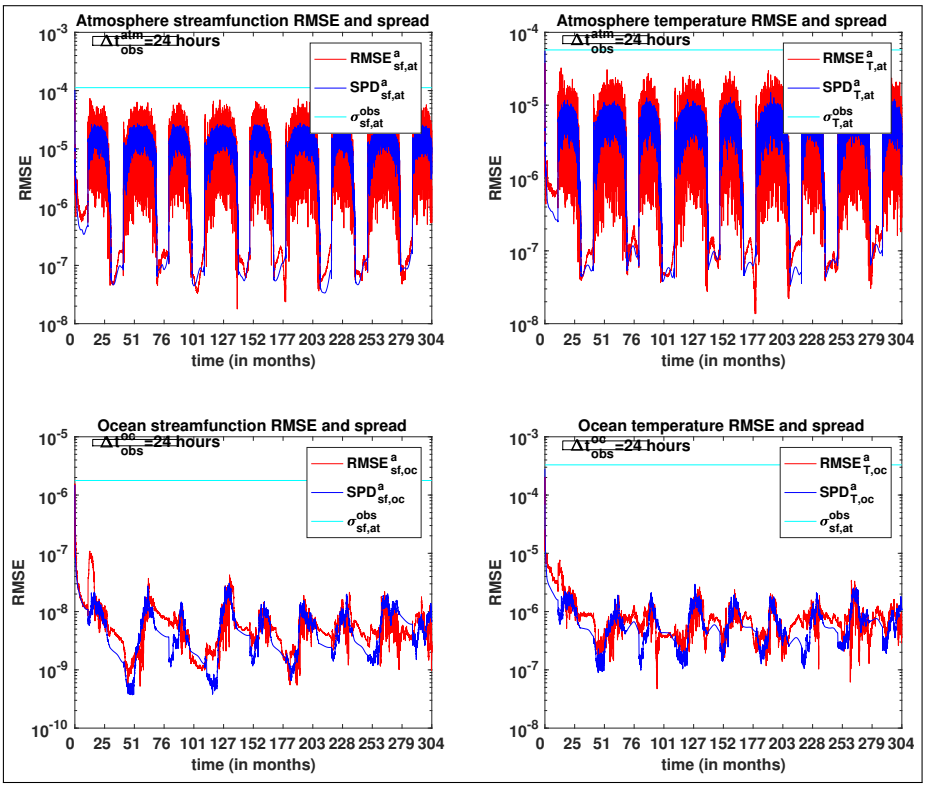


Figure 21: Data Assimilation of MAOOAM - Time series of RMSE and SPREADS - Regime 36st - 16 members

This is a 304 years assimilation of MAOOAM in the regime with 36 dimensions and strong coupling between ocean and atmosphere. The assimilation uses EnKF-N with a size of 16, that is why the error (RMSE) estimates well the spread and both are below the initial error. The system has enough members to assimilate. The low frequency variability is visible, there are stable and unstable parts.

3.7 RMSE vs size of the ensemble

We plot the RMSE in function of the size of the ensemble in the six regimes. There should be a sharp decrease after N_0 the number of positive and zero exponents. This number is represented in the same color as the corresponding curve.

The RMSE is averaged over 150 years for 36wk, 52wk, 56wk and 56st and 300 years for 36st and 52st. We have chosen 300 years to have robuster statistics.

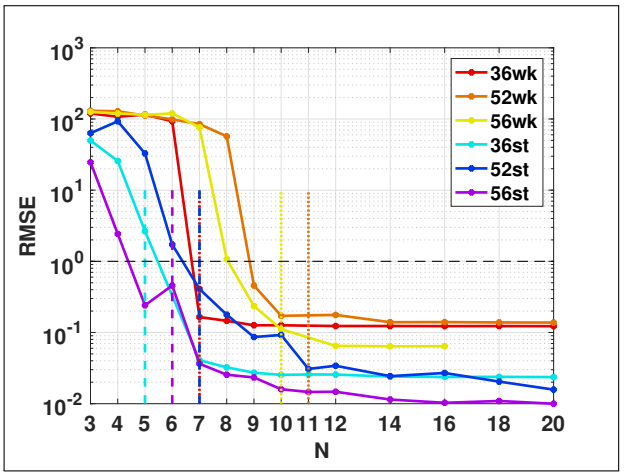


Figure 22: Data Assimilation of MAOOAM - RMSE vs size of the ensemble - Regime 36wk

Data assimilations are computed on the six regimes : weakly and strongly coupled with different dimensions. And RMSE are averaged over time and the dimensions of the state. The vertical bars represent the number N_0 of positive and zero Lyapunov exponents of the corresponding regime. We observe that the sharp decrease is around this number N_0.

Here is the same figure as above where the 4 fields (streamfunction and temperature of atmosphere and ocean). There are some differences between the atmosphere and the ocean. The ocean RMSE continue to decrease because it needs more dimensions because the ocean contains the neutral Lyapunov exponents.

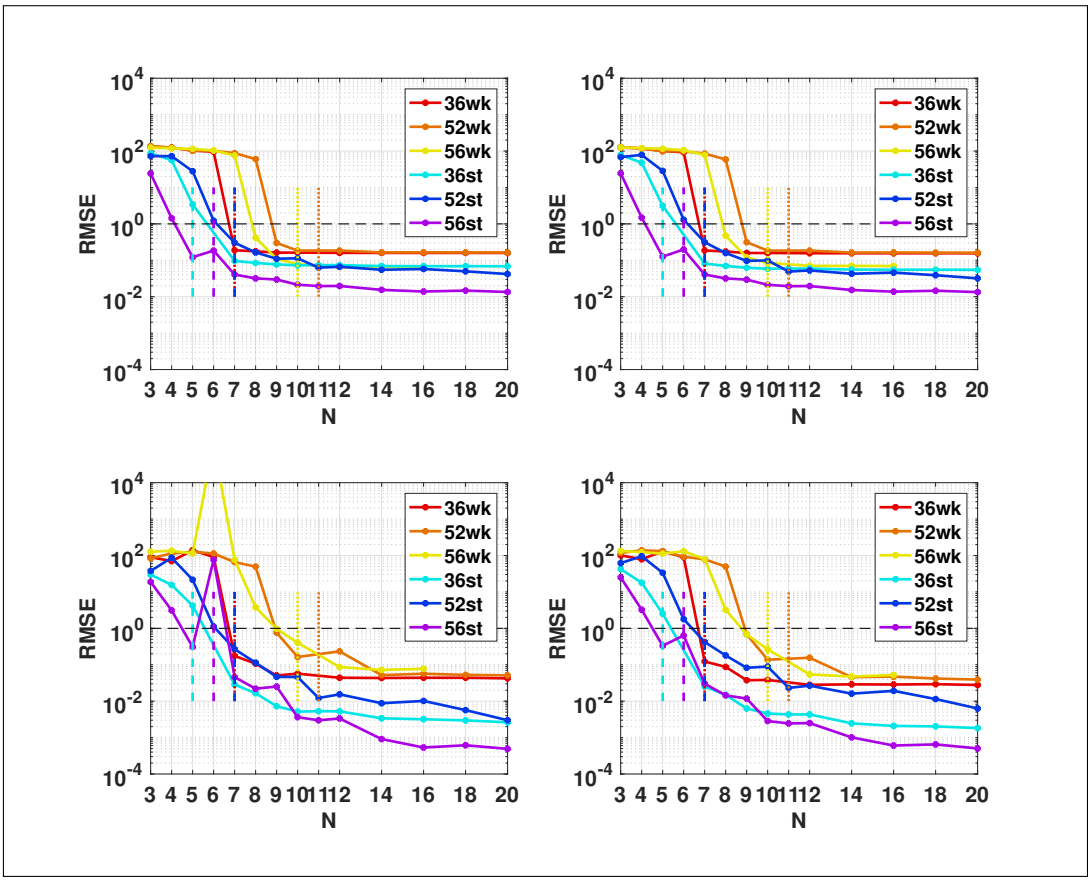


Figure 23: Data Assimilation of MAOOAM - RMSE vs size of the ensemble - Regime 36wk - 4 fields

Same figure as the Figure 22 but it is splitted in the four physical fields : atmosphere and ocean streamfunctions and temperatures. We want to see the different of scale in the values and the difference between the ocean and the atmosphere. The ocean needs more members than the atmosphere because it corresponds to the neutral exponents.

Conclusion

We have succeeded to implement a data assimilation and we have got good performances. Several experiments have been performed in order to test the impact of the size of the ensemble of the Ensemble Kalman Filter, the observation frequency, the importance of observing only the ocean or the atmosphere and the differences between the weakly and strongly data assimilation schemes.

Our results suggest that a strong link exists between some dynamical properties of the system at hand to the performances of the data assimilation. Specifically the dimension of positive, zero and negative sets of Lyapunov exponents has some links with the size of the ensemble of the data assimilation filter. The number of positive and zero exponents corresponds to the size of the ensemble needed to decrease sharply the error. The neutral modes linked to the ocean fields play also a role, we have to increase the size of the ensemble to continue to decrease the error in the ocean. It has been tested in cases with different level of chaoticity and cases with different state dimensions for MAOOAM.

Implementation

The original code of the model MAOOAM [5] is in Fortran in the GitHub repository : https://github.com/Climdyn/MAOOAM. We have succeeded in translating in python and validate it. We validate with tests on the values of the inner products, the tensor and some runs. The integration of the model was slow in python, but we have succeeded to wrap a Fortran 90 function and use it in python to accelerate it. The python code is : https://github.com/Climdyn/MAOOAM/tree/master/python and the documentation is in the appendix.

We have used some python implementation of the forward and backward Lyapunov vectors done by Dr Colin Grudzien from the Nansen Center. Thanks to it we have implemented the covariant Lyapunov vectors following the article of Kuptsov and Parlitz (REFERENCE).

For the data assimilation, we have used the python package DAPPER of Dr Patrick N. Raanes from the Nansen Center : https://github.com/nansencenter/DAPPER. Some minor changes have be done to compute the weakly coupled assimilation schemes. The code used is in the branch : https://github.com/nansencenter/DAPPER/tree/max1

Further work

We have to check the same relations observed in the model MAOOAM in the model Lorenz 63 with 2 compartments. The first step is the relation between the Lyapunov spectrum and the size of the ensemble. Then we can project the anomalies produced in the assimilation on the Covariant Lyapunov Vectors as it is done in Bocquet and Carrassi 2017 [20]. The goal is to analyze the properties of the anomalies along the unstable CLVs. The same analyze should be done for MAOOAM.

References

[1] Mark Asch, Marc Bocquet, and Maëlle Nodet. Data assimilation: methods, algorithms, and applications, 2016.

[2] Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[3] M Bocquet, PN Raanes, and A Hannart. Expanding the validity of the ensemble kalman filter without the intrinsic need for inflation. *Nonlinear Processes in Geophysics*, 22(6):645, 2015.

[4] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[5] S Vannitsem and L De Cruz. A 24-variable low-order coupled ocean–atmosphere model: Oa-qg-ws v2. *Geoscientific Model Development*, 7(2):649–662, 2014.

[6] S. Vannitsem. Météorologie dynamique - physf450.

[7] Geoffrey K Vallis. *Atmospheric and oceanic fluid dynamics: fundamentals and large-scale circulation*. Cambridge University Press, 2006.

[8] L. De Cruz, J. Demaeyer, and S. Vannitsem. A modular arbitrary-order ocean-atmosphere model: Maooam v1.0. *Geoscientific Model Development*, 03 2016.

[9] Priscilla Cehelsky and Ka Kit Tung. Theories of multiple equilibria and weather regimes-a critical reexamination. part ii: Baroclinic two-layer models. *Journal of the atmospheric sciences*, 44(21):3282–3303, 1987.

[10] Brian B Reinhold and Raymond T Pierrehumbert. Dynamics of weather regimes: Quasi-stationary waves and blocking. *Monthly Weather Review*, 110(9):1105–1145, 1982.

[11] Stéphane Vannitsem. The role of the ocean mixed layer on the development of the north atlantic oscillation: A dynamical system's perspective. *Geophysical Research Letters*, 42(20):8615–8623, 2015.

[12] Alberto Carrassi. *Esponenti di Lyapunov e Dimensionalitá Locale dell'attrattore di Lorenz*. PhD thesis, 2001.

[13] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 2: Numerical application. *Meccanica*, 15(1):21–30, 1980.

[14] Francesco Ginelli, P Poggi, A Turchi, H Chaté, R Livi, and A Politi. Characterizing dynamics with covariant lyapunov vectors. *Physical review letters*, 99(13):130601, 2007.

[15] Christopher L Wolfe and Roger M Samelson. An efficient method for recovering lyapunov vectors from singular vectors. *Tellus A*, 59(3):355–366, 2007.

[16] Pavel V Kuptsov and Ulrich Parlitz. Theory and computation of covariant lyapunov vectors. *Journal of nonlinear science*, 22(5):727–762, 2012.

[17] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[18] Stéphane Vannitsem and Valerio Lucarini. Statistical and dynamical properties of covariant lyapunov vectors in a coupled atmosphere-ocean model—multiscale effects, geometric degeneracy, and error dynamics. *Journal of Physics A: Mathematical and Theoretical*, 49(22):224001, 2016.

[19] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[20] Marc Bocquet and Alberto Carrassi. Four-dimensional ensemble variational data assimilation and the unstable subspace. *Tellus A: Dynamic Meteorology and Oceanography*, 69(1):1304504, 2017.

Appendices

Here is the documentation of the python version of model MAOOAM (https://github.com/Climdyn/MAOOAM/tree/master/python).

MAOOAM Documentation

Release

Maxime Tondeur, Jonathan Demayer

Jun 13, 2017

CHAPTER

ONE

SYNOPSIS

This repository provides the code of the model MAOOAM in python. It is a low-order ocean-atmosphere model with an arbitrary expansion of the Fourier modes of temperatures and streamfunctions. The code in Python is a translation of the Fortran code available in the main Git repository : https://github.com/Climdyn/MAOOAM.

CHAPTER

TWO

MOTIVATION

The code has been translated in Python to be used with the Data Assimilation module DAPPER : https://github.com/nansencenter/DAPPER .

CHAPTER

THREE

INSTALLATION

The program can be run with python 2.7 or 3.5. Please note that python 3.5 is needed by the Data Assimilation module DAPPER. Optionally, F2py is also needed to compile the optimized fortran part.

To install, unpack the archive in a folder or clone with git:

```
>>> git clone https://github.com/Climdyn/MAOOAM.git
```

and run f2py (optional):

```
>>> f2py -c sparse_mult.pyf sparse_mult.f90
```

CHAPTER

FOUR

GETTING STARTED

The user first has to fill the params_maooam.py according to their needs. See its documentation for more information about this file. Some examples related to already published articles are available in the params folder.

Finally, the ic.py file specifying the initial condition should be defined. To obtain an example of this configuration file corresponding to the model you have previously defined, simply delete the current ic.py file (if it exists) and run the program:

```
>>> ipython maooam.py
```

It will generate a new one and start with the 0 initial condition. If you want another initial condition, stop the program, fill the newly generated file and restart:

```
>>> ipython maooam.py
```

The code will generate a file evol_field.dat containing the recorded time evolution of the variables.

CHAPTER

FIVE

DESCRIPTION OF THE FILES

- maooam.py : main program.
- params_maooam.py : a module for the parameters of the model (dimensional, integral and physical).
- ic.py : initial conditions file.
- ic_def.py a module that generate the initial conditions if it does not exist.
- inprod_analytic.py : a module that compute the inner product needed for the tensor computation.
- aotensor.py : a module that compute the tensor.
- integrator.py: a module that compute one step of the model and RK2 integration.
- sparse_mult.f90 and sparse_mult.pyf : fortran and f2py files to call fortran module in python.

CHAPTER

SIX

CONTENTS

6.1 Parameters module

This module defines the parameters for the model.

Note: The python code is available here : params_maooam.py .

Example

```
>>> from params_maooam import ndim,natm,noc
>>> from params_maooam import oms,nboc,ams,nbatm
>>> from params_maooam import *
```

There are three types of parameters :

- integration parameters : simulation time (transient and effective), time step, writeout and write step time
- dimensional parameters : dimensions of the truncation of fourier for the atmosphere and the ocean
- physical parameters : they are used in the tensor for the integration

6.1.1 Integration parameters

Warning: Time is adimensional. If t_real is in seconds, then t_model = t_real * f_0 where f_0 is the Coriolis parameter at 45 degrees latitude (1.032e-4)

- **t_trans** : the transient simulation time of the model to be on the attractor. The states vectors are not written on evol_field.dat.
- **t_run** : the running simulation time of the model. The states vectors are written on evol_field.dat every tw.
- **dt** : the step time.
- **writeout** : boolean value to decide if the module produces evol_field.dat.
- **tw** : the step time to write on evol_field.
- **f2py** : boolean to activate the f2py optimization.

6.1.2 Dimensional parameters

- **oms** and **ams** : the matrices that gives the possible values of the modes Nx and Ny.
- **nboc** and **natm** : the numbers of oceanic and atmospheric blocs.
- **natm** and **noc** : the numbers of functions available.
- **ndim** : the total dimension.

Example

```
>>> oms =get_modes(2,4)# ocean mode selection
>>> ams =get_modes(2,2)# atmosphere mode selection
>>> nboc,nbatm = 2*4,2*2       # number of blocks
>>> (natm,noc,ndim)=init_params(nboc,nbatm)
>>>
>>> # Oceanic blocs
>>> #( x block number accounts for half-integer wavenumber e.g 1    => 1/2 , 2 => 1,␣
↪etc...)
>>> OMS[0,:] = 1,1
>>> OMS[1,:] = 1,2
>>> OMS[2,:] = 1,3
>>> OMS[3,:] = 1,4
>>> OMS[4,:] = 2,1
>>> OMS[5,:] = 2,2
>>> OMS[6,:] = 2,3
>>> OMS[7,:] = 2,4
>>> #Atmospheric blocs
>>> AMS[0,:] = 1,1
>>> AMS[1,:] = 1,2
>>> AMS[2,:] = 2,1
>>> AMS[3,:] = 2,2
```

Typical dimensional parameters

- atmosphere 2x,2y ; ocean 2x,4y ; ndim = 36
- atmosphere 2x,2y ; ocean 4x,4y ; ndim = 52
- atmosphere 2x,4y ; ocean 2x,4y ; ndim = 56

6.1.3 Physical parameters

Some defaut parameters are presented below. Some parameters files related to already published article are available in the params folder.

Scale parameters

- **scale = 5.e6** : characteristic space scale, L*pi
- **f0 = 1.032e-4** : Coriolis parameter at 45 degrees latitude
- **n = 1.5e0** : aspect ratio (n = 2Ly/Lx ; Lx = 2*pi*L/n; Ly = pi*L)
- **rra = 6370.e3** : earth radius
- **phi0_npi = 0.25e0** : latitude exprimed in fraction of pi

Parameters for the ocean

- **gp = 3.1e-2** : reduced gravity
- **r = 1.e-8** : frictional coefficient at the bottom of the ocean
- **h = 5.e2** : depth of the water layer of the ocean
- **d = 1.e-8** : the coupling parameter (should be divided by f0 to be adim)

Parameters for the atmosphere

- **k = 0.02** : atmosphere bottom friction coefficient
- **kp = 0.04** : atmosphere internal friction coefficient
- **sig0 = 0.1e0** : static stability of the atmosphere

Temperature-related parameters for the ocean

- **Go = 2.e8** : Specific heat capacity of the ocean (50m layer)
- **Co = 350** : Constant short-wave radiation of the ocean
- **To0 = 285.0** : Stationary solution for the 0-th order ocean temperature

Temperature-related parameters for the atmosphere

- **Ga = 1.e7** : Specific heat capacity of the atmosphere
- **Ca = 100.e0** ; Constant short-wave radiation of the atmosphere
- **epsa = 0.76e0** : Emissivity coefficient for the grey-body atmosphere
- **Ta0 = 270.0** : Stationary solution for the 0-th order atmospheric temperature

Other temperature-related parameters/constants

- **sc = 1.** : Ratio of surface to atmosphere temperature
- **lambda = 20.00** : Sensible+turbulent heat exchange between oc and atm
- **rr = 287.e0** : Gas constant of dry air
- **sb = 5.6e-8** : Stefan-Boltzmann constant

Key values

- **k** is the friction coefficient at the bottom of the atmosphere. Typical values are 0.01 or 0.0145 for chaotic regimes.
- **kp** is the internal friction between the atmosphere layers. kp=2*k
- **d** is the friction coefficient between the ocean and the atmosphere. Typical values are 6*10^{-8} s^{-1} or 9*10^{-8} s^{-1}.
- **lambda** is the heat exchange between the ocean and the atmosphere. Typical values are 10 W m^{-2} K^{-1} or 15.06 W m ^{-2} K^{-1}.

6.1.4 Dependencies

```
>>> import numpy as np
```

6.1.5 Fonctions

Here are the functions to generate the parameters.

params_maooam.**get_modes**(*nxmax*, *nymax*)
Computes the matrix oms and ams with nxmax and nymax

params_maooam.**init_params**(*nboc*, *nbatm*)
Computes the dimensions of the system

6.2 Initial conditions generator module

This module generates the initial conditions for the model if it doesn't exist with the good dimensions.

The dimensions of the system can be changed in the parameters file params_maooam.py . Then delete ic.py and ic_def.py will regenerates it.

Note: The python code is available here : ic_def.py .

Example

```
>>> from ic_def import load_IC
>>> load_IC()
```

6.2.1 Global file

The file ic.py in the same directory.

6.2.2 Dependencies

Uses the following modules to know the dimensions :

```
>>> from params_maooam import natm, noc, ndim, t_trans, t_run
>>> from inprod_analytic import awavenum, owavenum, init_inprod
```

6.2.3 Functions

Functions in the module :

ic_def.**load_IC**()
Check if ic.py exists, if not creates it with good dimensions and zero initial conditions

6.3 Initial conditions file

This file defines the initial conditions of the model. To be deleted if the dimensions are changed.

Example

```
>>> from ic import X0
```

6.3.1 Global variables (state vectors)

- X0 random (non-null) initial conditions.

6.3.2 Dependencies

```
>>> import numpy as np
```

6.4 Inner products module

Inner products between the truncated set of basis functions for the ocean and atmosphere streamfunction fields.

Note: These are calculated using the analytical expressions from De Cruz, L., Demaeyer, J. and Vannitsem, S.: A modular arbitrary-order ocean-atmosphere model: MAOOAM v1.0, Geosci. Model Dev. Discuss. and from Cehelsky, P., & Tung, K. K. : Theories of multiple equilibria and weather regimes-A critical reexamination. Part II: Baroclinic two-layer models. Journal of the atmospheric sciences, 44(21), 3282-3303, 1987.

Note: The python code is available here : inprod_analytic.py .

Example

```
>>> from inprod_analytic import init_inprod
>>> init_inprod()
```

6.4.1 Global variables

```
>>> awavenum = np.empty(natm, dtype=object)
>>> owavenum = np.empty(noc, dtype=object)
>>> atmos = atm_tensors(natm)
>>> ocean = ocean_tensors(noc)
```

6.4.2 Dependencies

it uses the modules :

```
>>> import numpy as np
>>> from scipy.sparse import csr_matrix
```

```
>>> from params_maooam import nbatm
>>> from params_maooam import nboc
>>> from params_maooam import natm
>>> from params_maooam import noc
>>> from params_maooam import n
>>> from params_maooam import oms
>>> from params_maooam import ams
>>> from params_maooam import pi
```

6.4.3 Classes

- atm_wavenum(typ,P,N,H,Nx,Ny)
- ocean_wavenum(P,H,Nx,Ny)
- atm_tensors(natm)
- ocean_tensors(noc)

class inprod_analytic.**atm_tensors**(*natm*)

Class which contains all the coefficients a,c,d,s,b,g needed for the tensor computation :

Attributes :

- $a_{i,j}$
- $c_{i,j}$
- $d_{i,j}$
- $s_{i,j}$
- $b_{i,j,k}$
- $g_{i,j,k}$

Return :

- The object will be name *atmos*.

calculate_a()

$$a_{i,j} = (F_i, \nabla^2 F_j)$$

Note: Eigenvalues of the Laplacian (atmospheric)

calculate_b()

$$b_{i,j,k} = (F_i, J(F_j, \nabla^2 F_k))$$

Note: Atmospheric g and a tensors must be computed before calling this routine.

calculate_c_atm()

$$c_{i,j} = (F_i, \partial_x F_j)$$

Note: Beta term for the atmosphere Strict function !! Only accepts KL type. For any other combination, it will not calculate anything.

calculate_d(*ocean*)

$$d_{i,j} = (F_i, \nabla^2 \eta_j)$$

Note: Forcing of the ocean on the atmosphere. Atmospheric s tensor and oceanic M tensor must be computed before calling this routine !

calculate_g()

$$g_{i,j,k} = (F_i, J(F_j, F_k))$$

Note: This is a strict function: it only accepts AKL, KKL and LLL types. For any other combination, it will not calculate anything.

calculate_s()

$$s_{i,j} = (F_i, \eta_j)$$

Note: Forcing (thermal) of the ocean on the atmosphere.

class inprod_analytic.**atm_wavenum**(*typ*, *P*, *M*, *H*, *Nx*, *Ny*)

Class to define atmosphere wavenumbers.

Attributes :

- typ (char) = 'A','K' or 'L'.
- M (int)
- P (int)
- H (int)
- Nx (int)
- Ny (int)

inprod_analytic.**init_inprod**()

creates and computes the inner products.

class inprod_analytic.**ocean_tensors**(*noc*)

Class which contains all the coefficients k,m,n,w,o,c needed for the tensor computation :

Attributes :

- $K_{i,j}$
- $M_{i,j}$
- $N_{i,j}$
- $W_{i,j}$
- $O_{i,j,k}$
- $C_{i,j,k}$

Return :

- The object will be name ocean

calculate_C_oc()

$$C_{i,j,k} = (\eta_i, J(\eta_j, \nabla^2 \eta_k))$$

Note: Requires $O_{i,j,k}$

and $M_{i,j}$ to be calculated beforehand.

calculate_K(*atmos*)

Forcing of the atmosphere on the ocean.

$$K_{i,j} = (\eta_i, \nabla^2 F_j)$$

Note: atmospheric a and s tensors must be computed before calling this function !

calculate_M()

Forcing of the ocean fields on the ocean.

$$M_{i,j} = (\eta_i, \nabla^2 \eta_j)$$

calculate_N()

Beta term for the ocean

$$N_{i,j} = (\eta_i, \partial_x \eta_j)$$

calculate_O()

Temperature advection term (passive scalar)

$$O_{i,j,k} = (\eta_i, J(\eta_j, \eta_k))$$

calculate_W(*atmos*)
Short-wave radiative forcing of the ocean.

$$W_{i,j} = (\eta_i, F_j)$$

Note: atmospheric s tensor must be computed before calling this function !

class inprod_analytic.**ocean_wavenum**(*P*, *H*, *Nx*, *Ny*)
Class to define ocean wavenumbers

Attributes :

- P (int)
- H (int)
- Nx (int)
- Ny (int)

6.5 Tensor computation module

The equation tensor for the coupled ocean-atmosphere model with temperature which allows for an extensible set of modes in the ocean and in the atmosphere.

Note: These are calculated using the analytical expressions from De Cruz, L., Demaeyer, J. and Vannitsem, S.: A modular arbitrary-order ocean-atmosphere model: MAOOAM v1.0, Geosci. Model Dev. Discuss. And the Fortran Code

Note: The python code is available here : aotensor.py .

Example

```
>>> aotensor, Li, Lj, Lk, Lv =aotensor.init_aotensor()
```

6.5.1 Help Functions

There are ndim coordinates that correspond to 4 physical quantities. These functions help to have the i-th coordinates of a quantity.

- psi(i) -> i
- theta(i) -> i + natm
- A(i) -> i + 2*natm
- T(i) -> i + 2*natm + noc
- kdelta(i,j) -> (i==j)

6.5.2 Global variables

- real_eps = 2.2204460492503131e-16
- t=np.zeros(((ndim+1),(ndim+1),(ndim+1)),dtype=float)

6.5.3 Dependencies

```
>>> from params_maooam import *
>>> from inprod_analytic import *
>>> from scipy.sparse import dok_matrix
>>> from scipy.sparse import csr_matrix
>>> import os
```

6.5.4 Functions

- compute_aotensor
- coeff(i, j, k, v)
- simplify
- init_aotensor

aotensor.**coeff**(*i*, *j*, *k*, *v*)

Affects v for $t_{i,j,k}$ making that tensor[i] upper triangular. Used in compute_aotensor.

Parameters

- **i** (*int in [1,37]*) – first coordinates
- **j** (*int in [1,37]*) – second coodinates
- **k** (*int in [1,37]*) – third coordinates
- **v** (*float*) – value

Returns change the global tensor

Return type void

Example

```
>>> coeff(i, j, k, v)
```

aotensor.**compute_aotensor**()

Computes the three-dimensional tensor t

Takes the inner products of inprod_analytic and computes the tensor

Parameters **t** (*array((37,37,37),float)*) – tensor t is a global variable of aotensor

Returns change the global tensor

Return type void

Example

```
>>> compute_aotensor()
```

Warning: Needs the global variable aotensor and the global inner products to be initialized.

aotensor.**init_aotensor**()
Initialize the tensor.

Returns aotensor, Li, Lj, Lk, Lv

Example

```
>>> aotensor, Li, Lj, Lk, Lv = init_aotensor()
```

aotensor.**simplify**()
Make sure that tensor[i] is upper triangular. To do after compute_aotensor().

Parameters **t** (*array((ndim+1,ndim+1,ndim+1),float)*) – tensor t is a global variable of aotensor

Returns change the global tensor

Return type void

Example

```
>>> simplify()
```

6.6 Integration module

This module actually contains the Heun algorithm routines.

Note: The python code is available here : integrator.py .

Example

```
>>> from integrator import step
>>> step(y,t,dt)
```

6.6.1 Global variables

- **aotensor** tensor with the format (int i, int j, int k, float v) in list
- **Li** first list of index of tensor
- **Lj** second list of index of tensor
- **Lk** third list of index of tensor
- **Lv** list of tensor values

6.6.2 Dependencies

```
>>> import numpy as np
>>> from params_maooam import ndim,f2py
>>> import aotensor as aotensor_mod
>>> if f2py:
>>>     import sparse_mult as mult
>>>     sparse_mul3_f2py = mult.sparse_mult.sparse_mul3
```

6.6.3 Functions

- sparse_mul3
- tendencies
- step

integrator.**sparse_mul3**(*arr*)
Calculate for each i the sums on j,k of the product

$$tensor(i, j, k) * arr(j) * arr(k)$$

Note: Python-only function

integrator.**sparse_mul3_py**(*arr*)
Calculate for each i the sums on j,k of the product

$$tensor(i, j, k) * arr(j) * arr(k)$$

Note: Python-only function

integrator.**step**(*y*, *t*, *dt*)
RK2 method integration

integrator.**tendencies**(*y*)
Calculate the tendencies thanks to the product of the tensor and the vector y

6.7 Principal module

Python implementation of the Modular Arbitrary-Order Ocean-Atmosphere Model MAOOAM

Note: The python code is available here : maooam.py .

Example

```
>>> from maooam import *
```

6.7.1 Global variable

- **ic.X0** : initial conditions
- **X** : live step vector
- **t** : time
- **t_trans**, **t_run** : respectively transient and running time
- **dt** : step time
- **tw** : step time for writing on evol_field.dat

6.7.2 Dependencies

```
>>> import numpy as np
>>> import params_maooam
>>> from params_maooam import ndim,tw,t_run,t_trans,dt
>>> import aotensor
>>> import time
>>> import ic_def
>>> import ic
```

class maooam.**bcolors**
to color the instructions in the console

CHAPTER

SEVEN

CONTRIBUTORS

Maxime Tondeur, Jonathan Demaeyer

CHAPTER

EIGHT

LICENSE

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

i

m

p