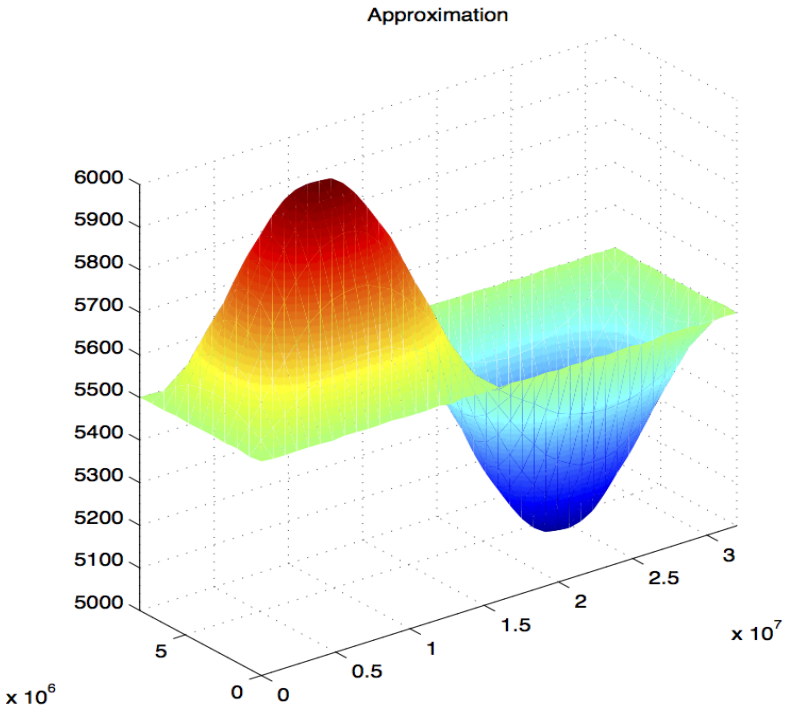
Projet d'Informatique et Mathématiques Appliquées

en

Algèbre Linéaire Numérique

Model reduction approaches for PDE problems (Phase 1)



Contents

1 Introduction and general context

The aim of this project is to illustrate the use of eigenvalues computations in the context of problems involving the resolution of partial differential equations (PDE). Two problems are considered: the forecast of the evolution in space and time of an atmospheric wave without the integration of a numerical model (direct problem) and the estimation of the velocity components (winds) in the initial condition from observations (inverse problem). This study consists of two parts:

- The application: the aim is to understand the use of a reduction technique - building informative lower dimensional subspaces - in data analysis, reconstruction and classification (this is described in Section 2). These tools will be applied (using Matlab) on sets of trajectories of a simplified atmospheric model. A snapshot of the wave height in the domain is illustrated in the figure of the first page.
- Computation of singular values and vectors: computing an informative reduced subspace amounts to computing the dominant singular values/vectors of a matrix derived from the model trajectories, therefore we are interested in efficient methods for the computation of singular values and left singular vectors. We will focus on some variants of the *power method* (this is described in Section 4); the aim will be to understand the numerical properties and the performance issues of these algorithms, and to implement them in Fortran. In the end, these algorithms will be embedded in the application described above.

2 Application context

2.1 Model reduction strategies in PDE problems and an application

The dynamics of geophysical fluids, as the ocean or the atmosphere, can be described by PDE, a.k.a the Navier-Stokes equations. Unfortunately, an analytical solution to this system of nonlinear equations is not known and numerical methods are required to provide an approximation of the solution. It leads to the development of complex *numerical models* based on a spatio-temporal discretization of the continuous problem (e.g. finite difference/volume/element methods). Knowing the state of the system at a given time - called *initial condition* - it is possible to know an approximation of the state of the system at any time simply by running the numerical model forward in time. However, the uncertainties in the initial condition can be large due to our lack of knowledge of the state of the system. Because they propagate with time accordingly to the dynamics of the system, they can significantly degrade the accuracy of the solution for large integration windows. When additional information are available, typically if the system is partially observed, it is possible to reduce the uncertainty in the initial condition. This later one can be *optimized* by properly taking into account the information in the model solution and the observations and their associated uncertainties. This kind of problems are called *inverse problems* and can be addressed with data assimilation methods. However, the computational costs and time can be very large, especially when the resolution of the mesh, and so the dimension of the discret problem, increase. This motivates the development of strategies aiming at reducing the dimension of the problem in order to reduce the computational costs and speed-up the resolution. An approach consists in building lower dimensional subspaces of the discret space that preserve the main features and properties of the variables. Then, the original problem is solved on these subspaces. This approach is expected to be computationally less expensive and faster without compromising too "much" the accuracy of the solution. Indeed, it assumes that such subspaces exist and can be computed.

In this project, we focus on a single level atmospheric model based on the shallow water equations. The configuration represents the evolution of a wave in a channel with periodic boundary conditions in the x-direction. The state vector is made of three components: the zonal velocity component $u(x, y, t)$, the meridional velocity component $v(x, y, t)$, and the Geopotential height $z(x, y, t)$. The evolution of the state vector is given by the following system of equations:

$$\begin{cases} \dfrac{\partial u}{\partial t} - fv + g\dfrac{\partial z}{\partial x} - \nu\Delta u = 0 \\ \dfrac{\partial v}{\partial t} + fu + g\dfrac{\partial z}{\partial y} - \nu\Delta v = 0 \\ \dfrac{\partial z}{\partial t} + z(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}) - \nu\Delta z = 0 \end{cases}$$

with g the gravity, ν the diffusivity coefficient, and f the Coriolis parameter given by the β-plane approximation. The dimensions of the domain are 32000 km in the x-direction (channel length) and 8000 km in the y-direction (channel width). The length of the simulation is 3 days. A finite difference method is used to discretize the continuous model. The spatial grid is uniform (41×11 grid points) and the time step is 1800s. More details regarding the numerical scheme, the grid, and the configuration can be found in the `MATLAB` function `Model.m`.

2.2 Building information-dense subspaces

A possible way to obtain the solution of a discretized PDE would be to build a basis of the discrete solutions space (finite dimension) and find a strategy to estimate the components of the solution given the prior information that we have (estimate of the initial and boundary conditions, observations, etc..). While the solution obtained with this strategy could be uncertain - the accuracy will depend on the prior information - this also requires to compute a basis of a high dimension space (up to billions of variables in some applications). This raises several issues involving the definition of efficient sampling strategies or the computation capacities required to tackle the dimension of the problem. Thus, this approach is not tractable in high dimensional problems.

However, it is still possible to build a dictionary of solutions by running the model with different model inputs, for instance by perturbing the initial condition, and storing the trajectories. This leads to build a matrix $F \in \mathbb{R}^{n_t n_s \times N_{ens}}$, where n_s is the physical dimension (number of grid points), n_t is the temporal dimension and N_{ens} is the number of simulations that have been performed. Our aim is now to approximate the solution of the PDE in the subspace $\mathcal{R}(F)$ defined by the range of the columns of F: $F = [X_1 \cdots X_{N_{ens}}]$ where the $(X_i)_{i=1:N_{ens}} \in (\mathbb{R}^{n_t n_s})^{N_{ens}}$ correspond to N_{ens} solutions of the PDE. However, considering the large size of the ensemble (N_{ens}) that could be required to capture "enough" information about the solution space, a further reduction of the dimension of the problem should be performed. Indeed, this should be done without reducing too much the information content of this "new" subspace, noted S.

It is possible to express the solutions X_i in terms of a mean trajectory $\bar{X}$, given by $\bar{X} = \frac{1}{N_{ens}} \sum_{j=1}^{N_{ens}} X_j$ and deviations from this mean $Z_i = X_i - \bar{X}$. We define the *anomaly matrix* by $Z = [Z_1 \cdots Z_{N_{ens}}]$ and denote by r its rank. Thus, a solution X of the PDE equation in $\mathcal{R}(F)$ can read $X = \bar{X} + Z\alpha$, with $\alpha \in \mathbb{R}^{N_{ens}}$ the components of $X - \bar{X}$ in $\mathcal{R}(Z)$ (the range of the columns of Z). It results that S can be built as a lower dimensional subspace of $\mathcal{R}(Z)$. Fixing $k \leq r$ the dimension of S, we will look for the "best" approximation of $\mathcal{R}(Z)$ among the subspaces of dimension k. This problem can be formulated as:

$$\begin{cases} \text{Find } Z_k \in \mathbb{R}^{n_t n_s \times N_{ens}} \text{ with rank } k \text{ such that} \\ \|Z - Z_k\|_2 = \inf\limits_{\substack{A \in \mathbb{R}^{n_t n_s \times N_{ens}}, \\ \text{rank}(A) = k}} \|Z - A\|_2 \end{cases}$$

This problem has been addressed in your Linear Algebra lectures and a solution is given by:

$$Z_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

where the $(\sigma_i)_{i=1:n}$ are the singular values of Z sorted in decreasing order, and the $(u_i)_{i=1:n}$ and $(v_i)_{i=1:n}$ are the associated left and right singular vectors. They are obtained by applying the singular value decomposition (SVD) to Z: $Z = U\Sigma V^T$. Furthermore, the quality of the approximation is assessed thanks to the following result:

$$\|Z - Z_k\|_2 = \sigma_{k+1}$$

The subspace S on which the solution of the PDE is computed is chosen to be the range of $(u_i)_{i=1:k}$, e.g. the range of the k first columns of the matrix U. The choice of k will be done accordingly to the quality of the subspace approximation, and more, specifically the information content, that we target:

$$1 - \frac{\|Z - Z_k\|_2}{\|Z\|_2} = 1 - \frac{\sigma_{k+1}}{\sigma_1} \tag{1}$$

and choosing k accordingly.

3 Reduced model strategy: data analysis, reconstruction and classification

The method described above consists in decomposing some data in terms of orthogonal basis functions that express the dominant patterns of the data. These basis functions correspond to the left singular vectors of the *anomaly matrix* associated to the data. The aim is to apply such techniques on data corresponding to the wave height in order to build a subspace, or several subspaces that will be used to address two problems:

1. Reconstruction: we have at our disposal an initial condition - a triplet $u(x, y, 0)$, $v(x, y, 0)$, $z(x, y, 0)$ - and we want to solve the PDE, a.k.a obtaining the solution over a given period, without the integration of the numerical model forward in time.
2. Classification: we have at our disposal some outputs of a model simulation - $(z(x, y, t_k))_{k\in\mathbb{N}_n}$ - and a component of the initial condition - $z(x, y, 0)$ - and we want to find the other components of the initial condition - $u(x, y, 0)$, $v(x, y, 0)$ - among a dictionary of potential candidates.

3.1 Data analysis

The first step consists in producing and analyzing the data in order to build the low dimensional subspaces that will be used when solving the reconstruction and classification problems.

0. We assume that the data corresponding to the outputs of the model will be stored in a matrix $F \in \mathbb{R}^{n_t n_s \times N_{ens}}$ where n_s is the physical dimension (number of grid points), n_t is the temporal dimension (number of snapshots that have been saved during the integration of the model) and N_{ens} is the number of simulations that have been performed. We assume that we have a parameter called $PercentInfo$ that corresponds to the quality of the subspace approximation that we want to reach. $PercentInfo$ will influence the number of left singular vectors that have to be kept.

1. Produce the data set by running the model with N_{ens} different initial conditions. This can be done by perturbing one initial condition with a normally distributed random variable

   $$\forall i \in \mathbb{N}_{N_{ens}}, \quad X_0^i = X_0 + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \Sigma),$$

   and storing the associated outputs of the simulation in F. The matrix F contains the *ensemble* of trajectories of the Geopotential height obtained from the model: $F = [F_1, \ldots, F_{N_{ens}}]$, with F_i the i-th column of F.

2. Compute the anomaly matrix $Z \in \mathbb{R}^{n_s n_t \times N_{ens}}$ from F. For each grid point and at each time, the *ensemble mean* is removed from F. It reads $\forall i \in \mathbb{N}_{N_{ens}} Z_i = F_i - \frac{1}{N_{ens}} F \cdot e$, with $e \in \mathbb{R}^{N_{ens}}$ the vector with all elements equal to 1.

3. Compute the dominant singular values and the associated left singular vectors of Z: either use the built-in Matlab `svd` function and filter the dominant singular values and left singular vectors a posteriori (using $PercentInfo$), or call the dominant eigenspace method (cf Algorithm 1). We call $U \in \mathbb{R}^{n_s n_t \times n_d}$ the basis vectors of the subspace (left singular vectors) that have been obtained (n_d vectors in total). We call D the set of singular values ordered by descending order of magnitude; $\frac{D_{n_d+1}}{D_1} \leqslant 1 - PercentInfo$ must hold.

4. Check the quality of the basis: $\frac{\|U \cdot U' \cdot Z - Z\|_2}{\|Z\|_2}$ should be small.

3.2 Reconstruction

In this problem, we have at our disposal an initial condition - a triplet $X_0 = [u(x, y, 0), v(x, y, 0), z(x, y, 0)]$ - and we want to solve the PDE, a.k.a obtaining the solution over a given period, without the integration of the numerical model forward in time.

0. We assume that we performed several simulations with the numerical model starting from different initial conditions and that some outputs of the model have been stored in the matrix F. We assume that we have at our disposal a new initial condition X_0 that differs from the ones used to build F. We want to compute the evolution in time and space of the solution of the PDE starting from X_0. However, we assume that we can no longer run the numerical model (not enough computing resources, not enough time to run the model).

1. We consider the ensemble of trajectories that we have already computed (i.e. data stored in F): compute the reduction strategy applying the same techniques as before. We call U the basis obtained with such strategy (the rank is n_d) and D the associated singular values.

2. We now consider the new initial condition X_0.

    1. Compute the anomaly vector Z_0 associated to X_0 (remove the ensemble mean $\bar{F} = \frac{1}{N_{ens}} F \cdot e$, with $e \in \mathbb{R}^{N_{ens}}$ the vector with all elements equal to 1).
    2. Fit with a least-square approach Z_0 on the basis U: the aim is to find $\hat{Z}_0$ in the range of the columns of $U_{k|t=0} = U(1 : ns, :)$ (the initial time components of the vector k of the basis) such that $\|Z_0 - \hat{Z}_0\|_2$ is minimum. One has: $\hat{Z}_0 = \sum_{k=1}^{n_d} \alpha_k U_{k|t=0}$. Therefore, we want to solve the overdetermined system $\min_{\alpha \in \mathbb{R}^{n_d}} \|U_{|t=0}\alpha - Z_0\|_2$. You can for example use a QR factorization or the normal equations.
    3. Once α is determined, we have expressed the known initial condition in the initial components of the U basis. The reconstruction consists in considering that the same α components are valid at each time of the simulation, i.e. the predicted anomalies of the solution Z_p are computed as $Z_p = U \cdot \alpha$.
    4. We assume now that you can run the model from X_0. Thus, you have access to the solution X^* of the numerical model and you can compare it to the solution obtained from the reconstruction. A measure of the quality of the reconstruction reads $\frac{\|Z_p + \bar{F} - X^*\|}{\|X^*\|}$.

3.3 Classification

The atmospheric model introduces a parameter `GW`$\in \{1, 2, 3\}$ that allows us to choose between different types of velocity fields when initializing the model: geostrophic velocity fields, null velocity fields and velocity fields in the opposite direction of the geostrophic velocity fields. Changes in this parameter, and so in the initial velocity fields, result in changes in the solution of the model, and three physical situations can be associated to the three values of the parameter. Now, assuming that we could partially observe the solution, we would like to be able to specify the value of `GW` that corresponds to this solution. Therefore, we assume that we have at our disposal some outputs of the model simulation - $(z(x, y, t_k))_{k \in \mathbb{N}_{n_t}}$ - and a component of the initial condition $(z(x, y, 0))$. However, we do not know how the other components $u(x, y, 0)$ and $v(x, y, 0)$ have been initialized. Our aim is to find which value of GW has been used to produce the series of wave heights $(z(x, y, t_k))_{k \in \mathbb{N}_{n_t}}$.

0. We assume that the time series of observed wave heights are stored in the vector $Z_{obs} \in \mathbb{R}^{n_s n_t}$. We know the water height component for the initial condition X_0.

1. In order to perform the classification analysis, we need to build an ensemble for each value of the parameter `GW` $\in \{1, 2, 3\}$. Produce three data sets of wave height, each one being associated to a value of `GW` by running the model with N_{ens} different initial conditions. Again, this is done by perturbing the wave height component of the initial condition with a normally distributed random variable. The output of the model are stored in three matrices F_i, with $i \in \{1, 2, 3\}$. Now, compute the reduction strategy applying the same technique as in §3.1 for each value of GW and the same value of *percentInfo*. We note $(U_i)_{i \in \mathbb{N}_3}$ the orthogonal bases of the subspaces associated to the dominant singular values.

2. A strategy to choose `GW` among all the possible values is to measure the distance of Z_{obs} to each subspaces. This can be done by computing the component of Z_{obs} in the orthogonal of U_i:

$$\pi_i = \|(I - U_i U_i^T) Z_{obs}\|, \quad \forall i \in \{1, 2, 3\}.$$

We will choose the value i of the parameter `GW` that corresponds to the lowest π. Compute the $(\pi_i)_{i \in \mathbb{N}_3}$ and suggest a value for the parameter `GW` associated to the observations Z_{obs}.

4 Numerical issues and extension of the power method

Note that in the data analysis described in Section 3.1 we are only interested in the left singular space (more precisely only its most representative or *dominant* part) of matrix Z. One common way of computing this subspace is by computing the dominant eigenspace of a matrix A defined as $A = Z * Z^T$ (note that the eigenvalues of A are the square of the singular values of Z).

The basic *power method*, which was introduced in the ALN lectures, can be extended to compute not only the dominant eigenpair but a complete set of k dominant eigenpairs of a symmetric matrix A. The resulting method is called *Subspace Iteration method*. Specifically, assuming a given fraction $PercentTrace$ of the trace of A, $trace(A)$, we are interested in computing those eigenpairs (v_i, λ_i) such that $(\lambda_1 + ... + \lambda_k)/trace(A) \geq PercentTrace$. The algorithm for achieving this is presented in Algorithm 1. Please note that, for the purpose of our discussion we focus on real matrices even if most results can be extended to complex matrices.

Algorithm 1 Dominant eigenspace method with Raleigh-Ritz projection

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$, tolerance ε, $MaxIter$ (max nb of iterations), p and $PercentTrace$ the target percentage of the trace A
Output: $k \leq m$ dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.

Generate an initial set of m orthonormal vectors $V \in \mathbb{R}^{n \times m}$
$niter = 0$; $converged = 0$; $PercentReached = 0$; $Trace = trace(A)$
repeat
  Compute Y such that $Y = A^p \cdot V$
  $V \longleftarrow$ orthonormalization of the columns of Y
  Compute the Rayleigh quotient $H = V^T A V$
  Compute the spectral decomposition $H = X \Lambda X^T$, where the eigenvalues of H ($diag(\Lambda)$) are arranged in descending order of magnitude
  Compute $V \longleftarrow V X$
  for $i = converged + 1, m$ **do**
    if $\|Av_i - \lambda_i v_i\| / \|A\| < \varepsilon$ **then**
      $converged = converged + 1$, $PercentReached = \sum_{j=1,...,i} \lambda_j / Trace$
    else
      break
    end if
  end for
  $niter = niter + 1$
until ($PercentReached > PercentTrace$ or $niter > MaxIter$)
$V_{out} = V(:, 1 : converged)$; $\Lambda_{out} = \Lambda(1 : converged, 1 : converged)$

Algorithm 1 is derived from the basic power method through the use of the techniques described below.

Orthogonalization The basic idea of the subspace iteration method is to do power iterations using a subspace V of dimension m rather than a single vector. If no special attention is taken, all the components of the subspace, that is, all the columns of V, will converge to the most dominant eigenvector. The orthogonalization operation, executed at every iteration of the method, forces the columns of V to converge to distinct directions.

The Rayleigh-Ritz projection method In a basic subspace iteration method, each iteration simply computes $V = AV$ and then orthogonalizes the columns of V (through a Gram-Schmidt process, for example). In this case the columns of V converge to a dominant eigenspace of A. The actual eigenvectors and the corresponding eigenvalues can be extracted from V using the *Rayleigh-Ritz* projection method.

To keep the introduction of Rayleigh-Ritz method simple we first show, as in [1] (Chapter 4.1) how to find exact eigenpairs; we extend this to computing exact eigenspaces and will then suggest (without giving the proof) a procedure to compute approximate eigenpairs.

Theorem 1 (Rayleigh quotient)**.** *Let $\mathcal{U}$ be a subspace and let the columns of the matrix U be an orthogonal basis for $\mathcal{U}$ (U^T is the left inverse of U, $U^T\ U = I$). We define the Rayleigh quotient to be*

$$H = U^T \cdot A \cdot U\ .$$

If $\mathcal{X} \subset \mathcal{U}$ is an eigenspace of A then there is an eigenpair (λ, w) of H such that $(\lambda, U \cdot w)$ is an eigenpair of A.

With Theorem 1, we have shown that the exact eigenspace contained in $\mathcal{U}$ can be obtained by looking at eigenpairs of the Rayleigh quotient H which is a much smaller matrix of order the dimension of the subspace $\mathcal{U}$.

Furthermore (and we assume it by continuity), we can expect that when $\mathcal{U}$ contains an approximate eigenspace $\tilde{\mathcal{X}}$ of A there would be an eigenpair $(\tilde{L}, \tilde{w})$ of H such that $(\tilde{L}, U \cdot \tilde{w})$ is an approximate eigenpair of A.

This suggests that to approximate an eigenpair of A we can perform the following steps that define the main steps :

1. Find U an orthogonal basis of a subspace $\mathcal{U}$.
2. Form the Rayleigh quotient $H = U^T \cdot A \cdot U$.
3. Compute (M, w) an eigenpair of H.
4. Then $(M, U \cdot w)$ is an approximate eigenpair of A.

One can also extend the previous procedure to compute an approximation of an eigenspace of dimension m that is illustrated here in the case of a symmetric matrix A (this is often referred to as the *Rayleigh-Ritz projection method*: given U an orthogonal basis of a subspace $\mathcal{U}$,

1. Form the Rayleigh quotient $H = U^T \cdot A \cdot U$.
2. H is symmetric : compute the spectral decomposition of H : $X^T \cdot H \cdot X\ =\ \Lambda$
3. X is thus an approximate eigenspace of H corresponding to eigenvalues in $diag(\Lambda)$ so that $U \cdot X$ is an approximate eigenspace of A corresponding to eigenvalues in $diag(\Lambda)$.

Note that in the spectral decomposition of H, the diagonal matrix Λ can be organized in any order (for example in descending order of magnitude) which could then give the idea of studying per column the quality of each approximate eigenpair.

Stopping criterion In those cases where we are only interested in the most dominant eigenpairs of a matrix, it is possible to minimize the cost of the subspace iteration method by interrupting the iterations as soon as the required number of dominant eigenpairs is found; for example, in Algorithm 1, the iterations are stopped when the converged eigenvalues account for a chosen fraction of the trace of A. Therefore, it is necessary to implement a stopping criterion which tests the convergence of single eigenvalues at each iteration. This can be achieved by applying the Rayleigh-Ritz projection method at each iteration to extract the approximate eigenvalues and check their accuracy with the test

$$\frac{\|Av_i - \lambda_i v_i\|}{\|A\|} < \varepsilon$$

Block iterations In order to accelerate the convergence of the subspace iteration method it is possible to merge multiple iterations by executing the operation $Y = AV$ not just once per iteration but p times, which is mathematically equivalent to executing the operation $Y = A^pV$; this technique is commonly known under the name of *block iterations*. Think about the advantages and the disadvantages of this solution and about the effect of the value of p on both the number of iterations and the execution time. Also, think about different ways of implementing the $Y = A^pV$ operation and their implications on both the execution time and the memory consumption. Possibly, at the end of Phase 2 of the project, provide experimental results supporting your conjectures.

5 Matlab codes provided

A prototype of the Reconstruction and Classification algorithm is provided in Matlab. Efficiency is not our main concern in this context and standard Matlab tools have been used. You are provided three Matlab files: `Model.m`, `Reconstruction.m` and `Classification.m`. The first one contains the numerical atmospheric model. It should not be modified. The two other scripts correspond to the reconstruction and classification problems that we want to solve. Regarding the classification problem, you have at your disposal 30 wave height observation files (`observation-G??.mat`) (one per group). These files are stored in the archive file `Observations.tar.gz`. You must provide the classification of the observations stored in the file associated to your group (the group 1 works on the file `observation-G1.mat`, and so on).

6 Deliverables phase 1

The work consists in the development of the Matlab code `Reconstruction.m` and `Classification.m`. The codes provided at the end of the first phase should be well documented and structured. **The readability of the code will be seriously taken into account in the evaluation**. A technical report that includes the following items should be provided as well.

1. Analysis, reconstruction and classification of solution of a simplified atmospheric model
    1. Combine all the informations provided in Section 3 to write a pseudo-code (referred to as **Reconstruction**) that, for a given initial condition and an ensemble of model trajectories, computes a low dimension subspace of the solution (indicate which mathematical ingredients are needed), and reconstruct the solution of the PDE, in the context of the scenario describes in Section 3 (2 pages max).
    2. Extend the Matlab code `Reconstruction.m` to reconstruct the model solution without integrating the model forward in time accordingly to the scenario described in Sections 3.1 and 3.2. Check the quality of your reconstruction and assess the impact of the parameters *percentInfo* and N_{ens} on the results.
    3. Combine all the informations provided in Section 3 to write a pseudo-code (referred to as **Classification**) that, for a given set of observations and a partially known initial condition, computes a low dimension subspace of the solution, and help the user to choose between several values of the model parameter accordingly to the scenario describes in Section 3 (2 pages max).
    4. Extend the Matlab code `Classification.m` to define which model parameter values has been used to build the set of observations associated to your group (for instance, the file `observation-G30.mat` is associated to the group 30). Assess the impact of the parameters *percentInfo* and N_{ens} on the results.
2. You have been using the Matlab function `svd` to compute the singular values and left singular vectors of matrix $Z \in \mathbb{R}^{m\times n}$ (with $m \geq n$) in the Matlab codes `Reconstruction.m` and `Classification.m`. The drawback of this approach is that we must compute the full spectrum of the matrix Z while only few singular values and left singular vectors could be required to reach the targeted accuracy of our subspace approximation. Thus, a reduction of the computation costs and time are expected from the definition of an algorithm that computes only what we need. One

possible solution to this method is presented in Section 4 and consists in computing the dominant eigenspace of a matrix $A = Z * Z^T$.

1. Comment on any possible advantage and drawback of the method proposed in Section 4.
2. Suggest modifications to Algorithm 1 in order to compute the dominant singular values and associated left singular vectors of a matrix $Z \in \mathbb{R}^{m \times n}$ (with $m \geq n$) directly without passing through the computation of $A = Z * Z^T$. Pay special attention to the stopping criterion which has to be modified in order to comply with Equation (1). Describe the algorithm that you obtained (2 page max).
3. Implement a version of this algorithm in the Matlab file `Reconstruction.m` and compare the performance against the Matlab function `svd`.

7 Deliverables phase 2 : developments and numerical experiments

At the end of the first phase, you will be given Fortran codes implementing Algorithm 1. Your modification of this eigensolver in order to process rectangular matrices should be written in Fortran.

Deliverables for the second part of the project will be described later in more details but they should include a technical report, the Fortran codes and material for a short presentation (10mn, pdf format):

- Final report
  1. Summary of the work done in this project (1 page max)
  2. Detailed description of the final eigensolver algorithms. All numerical and algorithmic modifications with respect to Algorithm 1 and will respect to the algorithms proposed in the report at phase 1 should be explained/justified (5 pages max).
  3. A summary of the experiments (maximum of 5 pages, results should be analyzed and commented). The improved version should be compared to the Matlab function `svd` on the applications studied during phase 1 in terms of both efficiency and accuracy. One could analyse the influence of increasing values of the block size on the performance.
  4. Concluding remarks (1 page max).
- A file (**pdf format**, any other format (doc, ppt, odt etc) will not be accepted) of presentation will be used during the oral examination (maximum of 4 slides). This presentation (10 min) should summarize the work done and will be used to illustrate the algorithmic work and the results (precision, performance).
- The Fortran files of the implementation of the block approach and of the deflation approach should be provided.

8 Grading

Your final grade will result from the evaluation of the oral exam (50% of the total) and the deliverables of Phase 1 (25% of the total) and Phase 2 (25% of the total).

9 Important dates

- The first phase of this work (report; see 6 and code) should be sent by mail to Ehouarn Simon (ehouarn.simon@enseeiht.fr) and Alfredo Buttari (alfredo.buttari@enseeiht.fr) by **Thursday May 7th 2015**.
- A more detailed description of the work to be done for the second phase and Fortran source files will be made available on Moodle on May 7th (preliminary description in 7)

- Deliverables for the second phase (codes, technical report and **pdf file for the oral presentation**) should be provided by **May 29th 2015, 8PM** by email to ehouarn.simon@enseeiht.fr.
- Oral examination will start by June 1st 2015.

10 Bibliography

[1] G. W. Stewart. *Matrix Algorithms: Volume 2, Eigensystems.* Society for Industrial and Applied Mathematics (SIAM), 2001.