Image data assimilation with fluid dynamics models: Application to 3D flow reconstruction

Cordelia Robinson

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal et télécommunications

Ecole doctorale Matisse

présentée par

Cordelia Robinson

préparée dans l'équipe FLUMINANCE (Inria - Irstea)

Image data assimilation with fluid dynamics models : Application to 3D flow reconstruction

Thèse soutenue à Irstea
le 18 décembre 2015

devant le jury composé de :

Jörn Sesterhenn
Professeur Dr Sc. Techn. Habil. TU-Berlin / Rapporteur

Olivier Thual
Professeur Université de Toulouse / Rapporteur

Jocelyne Erhel
Directeur de Recherche Inria / Examinateur

Alejandro Gronskis
Professeur Université de Buenos Aires / Examinateur

Roger Lewandowski
Professeur Université de Rennes 1 / Examinateur

Etienne Mémin
Directeur de Recherche Inria / Directeur de thèse

Dominique Heitz
Chargé de Recherche Irstea / Co-Directeur de thèse

Acknowledgments

Firstly, I thank Pr. Jörn Sesterhenn and Pr. Olivier Thual for the time they gave me to examine this thesis. I also thank the rest of my thesis committee : Jocelyn Erhel, Alejandro Gronskis, and Roger Lewandowski, for their encouragement, insightful comments, and interesting questions.

I would like to acknowledge my gratitude to my Ph.D supervisor Etienne Mémin for his guidance and I am very gratful for his welcome to the FLUMINANCE team. I thank him for offering me two wonderful opportunities to travel abroad to work with the department of fluid dynamics in Buenos Aires.

I would also like to thank my supervisor Dominique Heitz for his constant support and our insightful discussions over the entire course of this Ph.D. He showed one of the best examples of leadership and humanism I have ever seen.

I warmly thank Alejandro Gronskis for sharing generously his expertise and helping me to challenge and improve myself. I also thank Guillermo Artana for his warm welcome in his team at the department of fluid dynamics in Buenos Aires.

I express my gratitude to Sylvain Laizet for his expertise and his availability despite the distance. His programming skills and rigor pushed me to improve my own skills.

I thank Charles Deltel and Tina for their precious patience and their technical support. They were always available and willing to help whenever I ask.

I thank my co-worker Yin Yang for our friendly exchanges during this Ph.D. I thank him for our countless discussions and his valuable suggestions. It was a real pleasure to work with him for three years and to

I thank Ioana Barbu for our balcony breaks and for sharing her interesting perspectives in our discussions. I thank her for moral support during the hard times. And I thank all the members of the FLUMINANCE team for the years we spent together.

I would like to thank sincerely all my dear friends from Toulouse who supported me despite the distance and their own issues, including Nadia, Amy, Rémi, Raphael, Jimmy, Matt, Julien and Will. I also thank my partner's family for their support and kind thoughts over the years. And finally, I owe my deepest gratitude to my partner Stéphan who constantly supported me and never failed to cheer me up even I was sometimes about to give up.

I dedicate this thesis to my parents Olivier and Ioly Robinson.

Table of contents

Table of contents

List of figures

List of tables

Introduction

Motivation

The study of turbulent flows plays a prominent role in a wide range of applications. The aeronautics, naval and automobile industries are constantly in need for an accurate representation of the three dimensional wake of an unknown obstacle to build reliable and eco-friendly vehicles that can face the adversity of climate change. At a human scale, the study of turbulent flows can also bring significant improvements to pre-existing systems. For instance, the reconstruction and control of fundamental turbulent flows carried out by the ACTA team (Irstea, Rennes) initiated the development of the FROILOC© system which blows ultra-clean and cold air over a localized work space without altering the ambient air temperature. This system is very attractive for the food and pharmaceutical industries as it ensures the quality of the manufactured products and the well-being of the workers at the same time. All in all, we are most of the time surrounded by turbulent flows which influence human activity at various scales. Therefore, the more we understand the behavior of the turbulence phenomenon in fundamental flows, the more efficiently we can apprehend real life problems and propose efficient technological solutions.

Over the last century, two main fields emerged from the fluid mechanics context to study the behavior of fundamental turbulent flows. In the one hand, the Computational Fluid Dynamics (CFD) field developed a large panel of dynamical models to reconstruct numerically the evolution of steady or unsteady flows from an initial state. On the other hand, the Experimental Fluid Dynamics (EFD) field developed several measurement techniques to observe two dimensional, and more recently three dimensional, observations of each component of the velocity field. During the last decade, the fluid mechanics community witnessed the surge of techniques combining both approaches in order to benefit from the assets and overcome the limitations of each approach (Druault et al., 2004; Suzuki, 2014).

In recent studies, there is a trend towards the implementation of data assimilation techniques in a fluid mechanics context (Cuzol et al., 2007; Cuzol and Mémin, 2009; Kato and Obayashi, 2013; Papadakis and Mémin, 2009; Gronskis et al., 2013). Data assimilation emerged from the weather forecasting and the geophysics fields which are characterized by large-sized systems. This thesis follows the work of Gronskis et al. (2013) and carries out the application of a 4DVar data assimilation technique, to the 3D reconstruction of the characteristics of a given flow from image data and dynamical models. Two different configurations are studies in this thesis.

Free-surface flow configuration

Following the work of Combés et al. (2015) with particle filter, we investigate with a variational data assimilation method the combined use of Kinect depth observations with a shallow water dynamical model to reconstruct free surface flow geometry and dynamics. As mentioned in its name, the Shallow Water Equations (SWE) are used to model a low-depth three dimensional fluid at its surface. The SWE describes the height and the velocity fields at its surface, and are often used as a prototype in geophysics applications. In the framework of this thesis, we built a MATLAB dynamical code from scratch and implemented the 4DVar technique with the help of an automatic differentiation tool. At first, the reconstruction is performed with partial and complete synthetic observations. In a second step, we consider the real measurements of the evolution of the free surface height observed with a Kinect depth-sensor.

In this application, we seized the opportunity to compare the performance of the 4DVar technique with various strategies of an emergent hybrid data assimilation technique. The comparison was published in Computers and Fluids (Yang et al., 2015).

3D cylinder wake at Reynolds 300 configuration

This study is an extention of the work of Gronskis et al. (2013) to the reconstruction of a three dimensional cylinder wake at Reynolds 300 by combining a highly accurate and parallelized code Incompact3d (Laizet and Lamballais, 2009) with a sequence of two dimensional observations. An important part of this application was focused on the construction and validation of the discrete adjoint parallelized code necessary to the implementation of the 4Dvar method. We performed a first reconstruction of a purely synthetic flow generated by Incompact3d, using three dimensional observations. We then performed the reconstruction of a fully three dimensional flow from the alternated synthetic observations of orthogonal stereo PIV like observations (inflow and stream-wise plane). We investigate the possibilities of the reconstruction with the real observations obtained by orthogonal stereo PIV measurements.

This thesis is articulated around four chapters. At first, we overview in chapter 1 the numerical and experimental approaches developed so far in the fluid mechanics context to reconstruct a fundamental turbulent flow. We present the limitations of each approach and shed light on the attempts to combine both methods in the fluid mechanics context. We overview in chapter 2 the data assimilation techniques which are well suited to our problem and focus in particular on the variational data assimilation methods. We describe in chapter 3 the construction and the implementation of the the 4DVar technique to a free surface flow problem. We compare the reconstruction of the free surface characteristics by different data assimilation techniques (Yang et al., 2015) with synthetic and real observations. Finally, we carry out in chapter 4 the implementation of the 4DVar technique to a highly accurate parallel code. The 4DVar code was validated on a purely synthetic experimental case

and puts in perspective the reconstruction of a fully three dimensional flow from a sequence of two dimensional observations.

Chapitre 1

Turbulent 3D flow reconstruction

The study of turbulent flows are crucial for a wide variety of academic and industrial applications. While fluid dynamics problems are often associated to aerodynamics and geophysics, they reveal to have a decisive influence in many other fields. From the modeling of the blood flow within a vessel, to the refrigeration process for the quality and safety of food products ; fluid dynamics has played a pivotal role and opened the door to many applications. The study and modeling of fundamental flows is a cornerstone of industrial applications. By reconstructing the evolution of fundamental fluid flows characteristics (velocity and pressure fields, temperature etc...), we can thereafter enhance complex systems used in the industry. Traditionally, there are two main approaches to the 3D reconstruction of turbulent flows that we discuss in the following.

A first approach, described in § 1.1, consists in modeling the behavior of a set of variables that characterize fundamental flows based on physical laws. The Computational Fluid Dynamics (CFD) field offers a wide range of numerical methods to solve the dynamics equations describing the flow evolution. We also introduce the main techniques used in turbulence modeling. Finally, we discuss the inherent limitations of the CFD methods to model a realistic phenomenon, like the choice of the initial and boundary conditions or the tuning of the model's parameters.

A second approach, described in § 1.2, consists in observing directly the evolution of a fundamental flow in a controlled space. In this approach, many efforts have been devoted to measure a fluid motion in a non-intrusive and accurate manner. The experimental approach was formalized in the beginning of the XXth century and grew along with the technical progress of measurement sensors and computer vision techniques. The recent development of Experimental Fluid Dynamics (EFD) offers a bright perspective for the reconstruction of a full 3D turbulent flow within a square meter volume. Despite the remarkable technological progress, the EFD approach can only provide sparse and often noisy information that require an additional model to deduce the quantities of interests, namely the forces exerted on the fluid.

Considering the inherent limitations of the numerical and experimental approaches, the investigation for a combination of both approaches has aroused a lot of enthusiasm in the fluid dynamics community. We discuss the attempts from each community to combine a dynamics model with a set of observations in § 1.3. The most promising approach is given by the data assimilation techniques which stem from the

weather forecasting field and are well established in every weather forecasting centers and geophysics applications as well. Data assimilation techniques have captured the attention of many other communities, and recent efforts were notably carried out to implement them in the fluid dynamics context. We overview these endeavors and introduce the approach we followed within this PhD work.

1.1 Computational Fluid Dynamics

The simulation of steady and unsteady flow problems rose long before the growth of numerical methods and the advent of computers in the 1970s. Centuries of fluid mechanics research established the governing equations that describe the evolution of an infinitesimal particle in time and space (see § 1.1.1). The latter can be solved exactly by hand for simplified or idealized flows, or approximated by numerical methods. The increasing need to design atmospheric entry vehicles in the 1940s and beyond (for ballistic, then for orbital and lunar motivations), urged for the resolution of complex fluid dynamics problems. In parallel, numerical methods to solve ODEs and PDEs were known and conceived by paper since the time of Newton in the 1700s. Centuries of progress led to a wide range of concepts and numerical methods, however it was a tedious task to apply these methods by hand. Hence, the advent of computers in the 1970s boosted their growth and development ; and while the application of numerical methods go beyond fluid dynamics problem, there was a complete link between the development of numerical methods for PDEs and the simulation of fluid flow problems. It is within this context that the Computational Fluid Dynamics (CFD) field emerged and grew significantly. Nowadays, the literature provides us a wide choice of numerical methods that have been implemented and improved throughout the history of CFD. We briefly overview these methods in § 1.1.3. The development and improvement of the CFD field allowed us to study more and more complex flows, in particular the reconstruction of turbulent flows which is the key to many industrial applications. We briefly discuss the case of turbulence flow modeling principles in § 1.1.2. Finally, we discuss in § 1.1.4 the inherent limitations of the CFD approach.

1.1.1 Governing equations

We generally describe a fluid motion through three conservation laws that link the mass density ρ, the pressure field $p = p(x, y, z)$, the velocity field in each direction $\mathbf{u} = (u, v, w)$ and the energy E. The first conservation law guarantees the conservation of mass, the sum of the mass flow rates entering into a control volume is equal to the sum of the mass flow rates getting out of the control volume. The second conservation law comes from the application of Newton's second law of motion to a given fluid particle, the change of momentum $\rho\mathbf{u} = (\rho u, \rho v, \rho w)$ equals the sum of all the external forces applied to each fluid particle. Finally, the third conservation law, which comes from the first law of thermodynamics, guarantees the conservation of energy. The rate of change of energy equals the sum of the rate of heat and the work done on each fluid particle.

Within the fluid dynamics context, we generally consider the Navier-Stokes equations which describe the relation between the velocity and the pressure of a moving fluid. In an academic context, we generally assume that the variations of pressure doesn't influence the particles' density. This ideal case is specified by the incompressibility hypothesis, which states that $\nabla \cdot \mathbf{u} = 0$. In the Newtonian fluids approximation, we assume that the relation between stress and the corresponding rate of strain is linear. The viscous stress in the fluid is the sum of a diffusion viscous term (proportional to the gradient of velocity) and a pressure term. Turbulence, and the generation of boundary layers, are the result of diffusion in the flow. Under the previous hypotheses, the momentum equation is rewritten into the Navier-Stokes equations for an incompressible fluid. Hence, for an incompressible flow, the fluid dynamics equation reads,

$$\nabla \cdot \mathbf{u} = 0, \tag{1.1}$$

$$\partial_t \mathbf{u} + \frac{1}{2}\left(\nabla(\mathbf{u} \oplus \mathbf{u}) + (\mathbf{u}\nabla)\mathbf{u}\right) - \nu\nabla^2\mathbf{u} + \nabla p = 0. \tag{1.2}$$

The Navier-Stokes equations, coupled with the adequate initial and boundary conditions, are primarily used in fluid dynamics to describe fundamental flows and will be considered within this thesis in chapter 4.

Most of the dynamics fluid equations encountered in many other configurations are generally derived from the latter. We will also consider in this work the shallow water equations (SWE) which are deduced from the depth-integration of the Navier-Stokes equations. These equations are often used in an academic context to model the free surface of a fluid which depth is negligible with respect to its surface dimensions. While a vertical velocity term is not present in the shallow water equations, note that this velocity is not necessarily zero and can be recovered via the continuity equation. The standard 2D SWE are formulated as :

$$\begin{cases} \partial_t h + \partial_x(hu) + \partial_y(hv) & = & S_h, \\ \partial_t(hu) + \partial_x(hu^2) + \partial_y(huv) + g\partial_x h^2 & = & S_{hu}, \\ \partial_t(hv) + \partial_x(huv) + \partial_y(hv^2) + g\partial_y h^2 & = & S_{hv}, \end{cases}$$

where g is the gravity acceleration, S_h, S_{hu} and S_{hv} are the source terms such as, for instance, the Coriolis force $S_{h\boldsymbol{u}} = fh\boldsymbol{u}$, the friction $S_{h\boldsymbol{u}} = c_f \boldsymbol{u}\sqrt{u^2+v^2}$ or the topology at the "bottom" of the fluid.

1.1.2 Introduction to turbulence modeling

While the formal definition of turbulence is still in the center of numerous debates, there is a common ground on the definition of its main characteristics. In the turbulence context, the smallest scales, referred to as the Kolmogorov scales η, have a crucial impact on the characteristic length of the smallest scales of the flow beyond which kinetic energy is transformed into heat (sometime called eddies). Therefore, an ideal simulation of a turbulent flow would consist in representing the contributions of all the spatial and temporal scales of the turbulence spectrum (see figure 1.1). This complete representation can be achieved by the Direct Numerical Simulation

(DNS). Once we define a spatial and temporal discretization of the physical domain that can represent the smallest scales, we solve the Navier Stokes equations by the means of one or a combination of numerical methods presented in § 1.1.3.

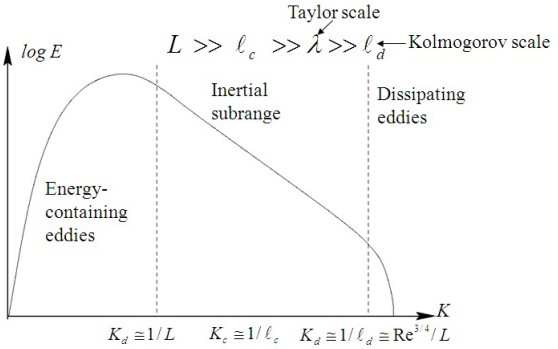

FIGURE 1.1 – Kolmmogorov energy spectrum representing all the scales of a turbulent flow

As one could expect, the DNS requires a tremendous amount of computational resources which increase with the size of the problem. We can assess the computational cost by considering a flow characterized by a given Reynolds number Re, the number of points necessary to describe all the turbulent flow scales depends on Re such as,

$$N_p \propto Re^{9/4}.$$

In addition, the number N_t of time steps Δt necessary for the simulation on a time interval $[0; T]$ is given by

$$N_t \approx \frac{T}{\Delta t} \approx Re^{1/2}.$$

Therefore, the cost of the DNS on a $[0; T]$ time interval is estimated to

$$N_p \times N_t \propto Re^3.$$

The higher the Reynolds number, the more expensive the DNS. In general, engineering applications require high Reynolds number, like for instance $Re = 10^8$ and $Re = 10^9$ in automobile and aerodynamic applications, respectively. While the DNS is clearly inapplicable in these applications, it has greatly contributed to the development of turbulence models that we introduce in this section. We outline three notable turbulence models and refer the reader to Wilcox (1998); Spalart (2000); Lesieur (2012) for a deeper description of these models.

Reynolds Average Navier Stokes (RANS) consists in dividing the velocities into mean components averaged in time and fluctuating components, by introducing them into the Navier Stokes equations, then modeling the new product terms. The RANS model has been implemented among commercial softwares such as ANSYS (former FLUENT). We refer the reader to Spalart (2000) for a review of RANS models.

Large Eddy Simulation (LES) consists in modeling the small eddies by a sub-grid scale model, pioneered by Smagorinsky (1963) and improved by Germano et al. (1991) for near-wall region, while the large eddies are solved directly by a transient calculation. The reader can find in Sagaut (2006) a broader review of LES methods. Mathematically, the LES comes to separating the velocity field into a resolved part, representing the large eddies, and a sub-grid part, representing the small scales.

More recently, Mémin (2014) has proposed an alternative to the classical Reynolds decomposition by separating a differential drift component to a time uncorrelated uncertainty random term. The latter isn't differentiable with respect to time, therefore it is handled through stochastic calculus. The dynamics associated with the differentiable drift component is derived from a stochastic version of the Reynolds transport theorem. It includes in its general form an uncertainty dependent sub-grid bulk formula that cannot be immediately related to the usual Boussinesq eddy viscosity assumption constructed from thermal molecular agitation analogy like in LES subgrid models. This formulation, considering uncertainties into the dynamical turbulence flow model is promising for the coupling with sparse data image. The learning of spatio-temporal model parameters taking into account data uncertainties and model error is potentially of huge interest, as for the moment no satisfying solution exists in complex real world cases encountered in industrial problems.

1.1.3 Numerical methods for CFD

Exact solutions of the Navier Stokes equations (1.1), (1.2) can be found in Crane (1970); Brady and Acrivos (1981); Wang (1992). These solutions are often restricted to a steady-state flow and boundary conditions, and they eventually require additional simplifications of the equations. On the other hand, numerical solvers of ODEs and PDEs were theoretically established in the literature. After over a half-century of research, we have at our disposal a wide range of numerical methods and discretization schemes to solve a set of PDEs at a certain level of accuracy. In the CFD framework, we commonly use the Finite Difference Method (FDM), the Finite Element Method (FEM), the Finite Volume Method (FVM) and the Spectral Element Method (SEM).

Finite Difference Method (FDM) is the first numerical method to be applied to fluid dynamics problems. In the same way as the definition of a derivative, the FDM replaces the partial derivatives of the governing equations by algebraic difference quotients. This yields to a system of algebraic equations from which we deduce the value of the variables of interest at each grid points. The two sources of error in the FDM are round-off errors, the loss of precision due to computer rounding of decimal quantities, and truncation error or discretization error, which is the difference between the exact solution of the finite difference equation and the exact quantity assuming no round-off errors. The easy implementation of the FDM is an important asset which allowed its fast improvement and spread to various domains of application. In the context of turbulent flows, it is necessary to deal with highly accurate schemes which are capable to represent all the ranges of scales while ensuring a

stability. The compact finite difference schemes (Lele, 1992) keeps the versatility of the FDM and provides a spectral-like accuracy. These methods are ideal for DNS as they provide a sufficiently high accuracy and robustness.

Finite element method (FEM) stems from structural mechanics problems which focus on the deformation, deflections and internal stresses within structures. These problems are characterized by complex geometry that can't be represented by a regular structured grid. Therefore, the FEM offers the flexibility to subdivide the domain into cells, called elements, of flexible shape. The elements have typically a triangular form (in 2D) but they can also be rectangular or curved which is an important asset with respect to FDM. The reader can find more description in Szabo and Babu**v**ska (1991). In opposition to the FDM, the FEM doesn't seek for the solution of the PDE itself, but aims at solving the integral form of the PDE instead. This asset leads to more flexibility in terms of boundary conditions implementations and allows the construction of higher order accurate methods. However, the implementation of the FEM itself is a tedious task and is actually worth the effort for a solution that is assumed to depend *a priori* on a complex geometry. In this perspective, FEM is well established in industrial applications and is commonly implemented entirely or partially in most commercial tools such as ANSYS (formely Fluent) or FLOW-3D.

Finite Volume Method (FVM) is well suited to simulate various types (parabolic, elliptic, hyperbolic etc...) of conservation laws and is usually favored to solve CFD problems in particular (Versteeg and Malalasekera, 2007). In opposition to the FDM, the FVM is based on the variational formulation of the PDE. Instead of determining the value at a local grid point, the FVM quantifies the fluxes that move from a discretized cell - referred to as a control volume - to its neighbor. By construction, the FVM ensures the numerical conservation of the flux which is an attractive asset to the fluid dynamics context. Similarly to the FEM, the FVM may be used on complex geometries, using structured or unstructured meshes but requires less implementation effort.

Spectral Method (SM) are established as the most accurate numerical method to solve fluid dynamics PDEs. Spectral methods (Gottlieb and Orszag, 1977) consists in using Fourier or Chebyshev representation and solving the governing equations in the spectral space which guarantees the high accuracy of the method. However, these methods are usually limited to simple geometries and for complex configurations, conventional finite difference methods are used. Nevertheless, the accuracy of the SM is a very attractive asset which pushed the development of hybrid methods. For instance, the Spectral Element Method, introduced by Patera (1984), decomposes the domain into a series of elements similarly to the FEM and the velocity in each element is represented as a high-order Lagrangian interpolant through Chebysev collocation points. Similarly, the Spectral Volume Method (Wang, 2002; Wang and Liu, 2006) decomposes each spectral volume into a series of control volumes. The cell-averaged data from these control volumes are used to reconstruct a high-order approximation in the spectral volume. The Riemann solvers are then used to compute the fluxes at spectral volume boundaries.

In the framework of this PhD, our goal is to perform a DNS model for two fluid flow configurations using the appropriate numerical schemes. In a first academic application, we consider a simplified 2D Shallow Water model defined on a simple geometry form (see chapter 3). We implemented the FVM, as described in detail in § 3.1, which provides an accurate solution for a reasonable implementation effort. On the other hand, we perform a DNS for a 3D turbulent cylinder wake for $Re = 300$ based on the incompressible Navier Stokes equations. The DNS code employed, Incompact3d, combines a spectral method and high order compact FDM as described in Laizet and Lamballais (2009).

1.1.4 Limitations of computational fluid dynamics

As depicted throughout this section, the final goal of CFD methods is to provide a numerical approximation of the actual physical laws that governs a realistic flow. While the CFD field provides accurate methods and tools to represent ideal flows, there are still inherent limitations that may lead to a poor representation of a turbulent realistic flow in three dimension. We discuss in this section the main limitations we can encounter in a purely CFD approach.

Model errors are inherent to the CFD approach. Indeed, we don't always have enough of knowledge of the physical laws governing the turbulent flow of interest itself. Typically, in many fundamental and industrial problems, it is nearly impossible to know the exact boundary conditions. Despite we achieve a perfect representation of the flow within the volume of interest, a bad choice of the **boundary conditions** model might lead to the spread of errors within the whole volume and compromise the final solution of the problem. In a channel flow configuration for instance, it is nearly impossible to model a realistic inflow for a turbulent flow which is crucial to define the evolution of the flow throughout the channel. In a three dimensional cylinder wake configuration in particular, it is custom to impose a periodicity condition along the cylinder axis. Indeed, this choice is based on experimental measurements of (Zdravkovich, 2003), which shows a periodic distribution ($2 < \lambda_z/D < 9$) depending on the Reynolds number.

Additional errors also rise when we consider **turbulence models** such as (LES, RANS). These errors are induced by the model itself and the adjustment of the parameters of the model. These errors can be eventually included in the dynamical model and the parameters can be estimated *in situ* (see § 1.1.2).

The initial condition definition is also a major limitation to the CFD approach. Even if we possess a perfect dynamics model and we input a misinformed initial condition, we might obtain a completely different flow. This comes from the nonlinear terms of the Navier Stokes equations and the deterministic nature of the CFD model. In addition, as we mentioned previously, a turbulent flow is sensible to the finest scales of the flow. Therefore, in a turbulent flow with a high Reynolds number, a slight change in the finest scale structures can impact the large scale structures.

Numerical errors induced by the numerical schemes employed to discretize the dynamics model (§ 1.1.3) can also compromise the representation of realistic turbulent flows. Despite the remarkable progress in the CFD field and the constant growth of computational resources, we are inevitably exposed to round-up and truncation errors. Such errors can be lowered by the representation of the flow on a finer grid, however the latter requires a considerable increase of computation time and storage.

In a nutshell, CFD models provide an essential pillar of the reconstruction of turbulent flows as they provide time continuity and dynamical consistency among the velocity and pressure fields. Nevertheless, these models are only able to reconstruct accurately ideal flows and we still lack knowledge on the physical model of realistic three dimensional turbulent flows. This lack mainly concerns the initial and boundary conditions, and the turbulence model errors and the tuning of their parameters. Additionally, we have to find the best compromise between the computational resources and the accuracy of the numerical schemes used to discretize the dynamics model.

1.2 Experimental fluid dynamics

One of the early and most prominent example of fluid mechanics experiment has been carried out by L. Prandtl in 1904 who observed and studied the aspects of unsteady separated flows behind several objects in a water tunnel(figure 1.2). The flow was visualized by distributing a suspension of mica particles on the surface of water. The latter set up a revolutionary experimental methodology that has systematically been used until today, and paved the way for measurement techniques expansion. Likewise the CFD field, the Experimental Fluid Dynamics (EFD) field expansion has been closely linked with technological progress. Within the fluid dynamics context, the intrusive Hot Wire Anemometry (HWA) and the non intrusive Laser Doppler Anemometry (LDA) were popularly used and developed to measure an accurate and punctual point in space. The estimation of the whole velocity field could be obtained by an additional post-processing technique. While these techniques are well established in the community, they were progressively supplanted by Particle Image Velocimetry (PIV) and later on 3D Particle Image Velocimetry which are briefly discussed throughout this section.

1.2.1 Particle Image Velocimetry for 2D observations

Nowadays, by combining Prandtl's attempt to trace particles, a contemporary laser and image processing techniques, we are able to measure the velocity of micron-sized particles following the flow by the so-called Particle Image Velocimetry (PIV) technique, illustrated in figure 1.3. PIV is a well established non-intrusive 2D measurement technique for measuring velocity fields of fundamental flows from a sequence of instantaneous snapshots (Adrian, 1991). The displacement of the particles is estimated statistically by correlating two sequences of particle images as described in Meinhart et al. (1993). If we consider for instance an airflow at $Re = 300$, i.e.

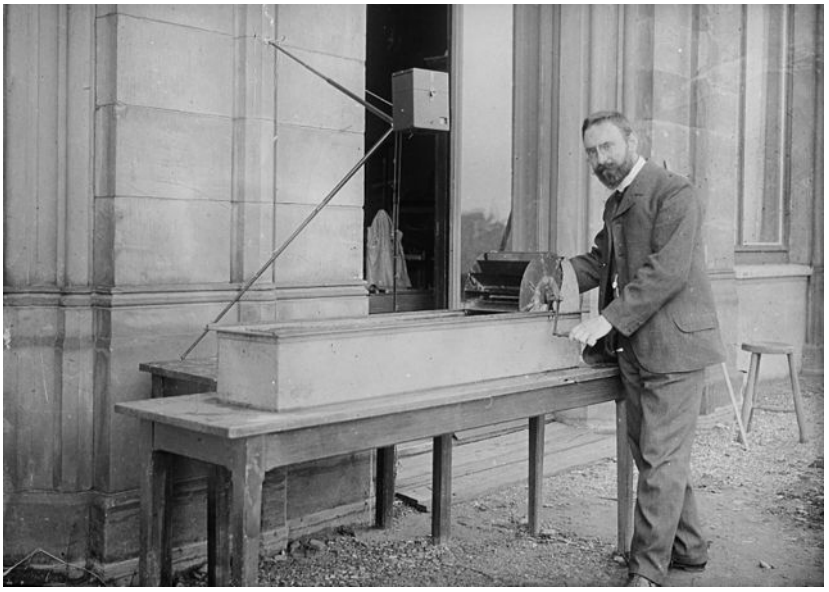

FIGURE 1.2 – Ludwig Prandtl in 1904 showing a test bench, known as the Prandtl channel, to visualize the flow evolution.

$\frac{L}{\eta} = O(Re^{3/4}) \approx 72$, the PIV experiments is able to provide the same spatio-temporal resolution than a DNS. However, if we consider a higher Reynolds number, the corresponding dynamics range is $\frac{L}{\eta} >> 100$ whereas the PIV is limited to a velocity dynamics range of range of 100 :1 (Adrian, 1995). Therefore, at the PIV resolution we can't retrieve the smallest scales yielding a coarse representation of a turbulent flow. The reader can find a complete description of the PIV technical background and applications in Adrian (2005) and in Raffel et al. (2013).

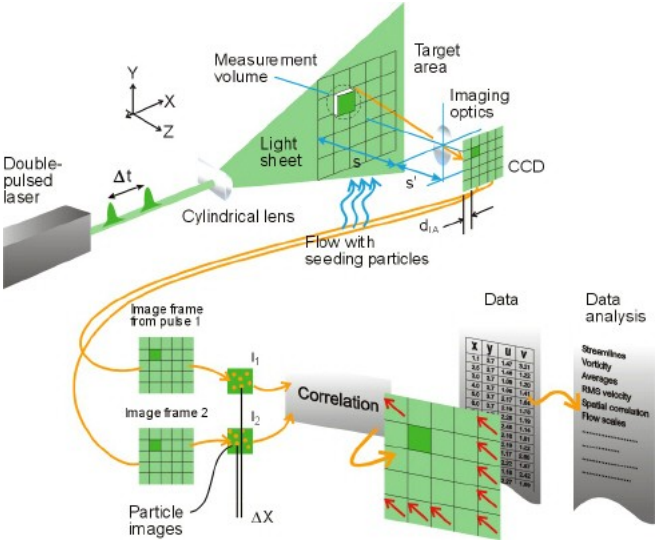

FIGURE 1.3 – The classical PIV system configuration setup is composed of a double-pulsed laser and CCD camera. We deduce the velocity field by performing the analysis of the correlation between a sequence of frames.

The measurement of a two-dimensional three-component velocity field is a major

stake for fluid mechanics and is even indispensable for a good understanding of turbulent flows. By combining PIV and stereoscopy, Arroyo and Greated (1991) developed a set-up which takes two stereoscopic images of the flow simultaneously with only one camera, yielding a measurement of the three components velocity flow in one plane. Kähler and Kompenhans (2000) developed a multi-plane stereo PIV system which consists of a four-pulse laser system and two pairs of progressive-scan cameras in an angular imaging configuration. This system provides the three components velocity field based on a classical PIV setting.

1.2.2 3D Particle Image Velocimetry

The progress of PIV techniques led to satisfying three components reconstruction of the velocity fields. Nevertheless, in order to have a better understanding of turbulent flows, the three dimensional three component velocity field is of crucial need. Several techniques to measure three dimensional quantities were proposed in the literature, such as scanning-PIV (Brücker, 1997), 3D Particle Tracking Velocimentry (3D-PTV) (Virant and Dracos, 1997) or holographic-PIV (Hinsch, 2002; Chan et al., 2004). Tomographic PIV (tomoPIV), pioneered by Elsinga et al. (2006), offers one of the most interesting perspective for three dimensional unsteady flow measurements. In analogy to the PIV technique, tomoPIV consists in seeding the measured 3D volume with tracer particles and illuminate the 3D region by a pulsed light source. The particles are then recorded simultaneously by CCD cameras similar to stereo-PIV as illustrated by figure 1.4. The tomoPIV measurement technique has recently emerged thus we are able to retrieve very small volumes using thin tracer particles. The first tomo-PIV measurements carried out by Elsinga et al. (2006) retrieved the three components of the a cylinder wake velocity field within a 40 mm $\times$ 40 mm $\times$ 10 mm volume in air using $1\mu m$ droplets as tracer particles. Most of the tomo-PIV experiments achieved so far remain in that scale order (Schröder et al., 2008; Humble et al., 2009) in order to provide an accurate representation of the particle motion.

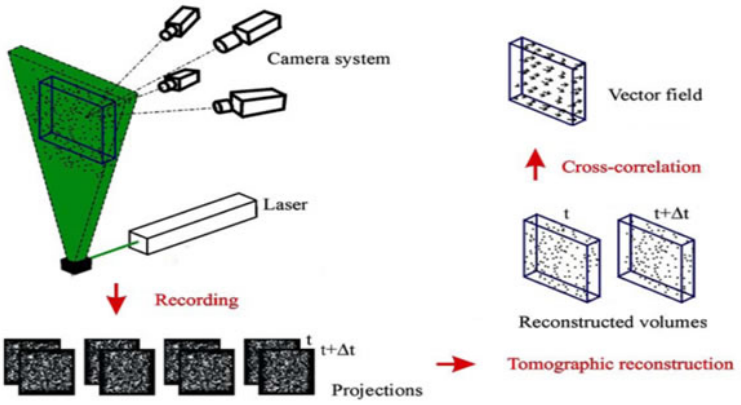

FIGURE 1.4 – The tomographic PIV system configuration setup is composed of a double-pulsed laser and CCD camera. We deduce the velocity field by performing the analysis of the correlation between a sequence of frames.

More recently, there is an emergence of large-scale tomo-PIV systems to retrieve a cubic-meter sized volume by using Helium-Filled Soap Bubbles (HFSB) (Kuhn et al., 2011; Scarano et al., 2015). In this configuration, we seed the turbulent flow with HFSB of which are much larger than the $1\mu m$ particle used so far in PIV and tomoPIV techniques. Due to their size, the particle distribution is smaller and more sparse thus they are esentially able to measure large scale structures. Other measurement techniques also such as tomoPTV and 4D-TomoPIV "Shake the box" also propose to retrieve the three dimension three component velocity fields in for wide volumes (Schanz et al., 2013).

1.2.3 Limitation of Experimental Fluid Dynamics

The EFD techniques presented throughout this section are able to capture phenomena that we might not be able to model yet and that improves our knowledge on the flow of interest. Despite the technological and methodological progress in the measurement techniques, there are still inherent limitations that may restrict severely the representation of the realistic turbulent flows. We discuss in this section the limitations that we can encounter in the 2D and 3D measurement techniques.

2D measurements are first and foremost limited in size, PIV measures are typically limited to $1000 \times 1000\text{px}^2$ which comes to $8 \times 8\text{cm}^2$. In terms of spatial resolution, as mentioned previously the PIV is limited to a velocity dynamics range of $100 : 1$ (Adrian, 1995). Therefore, at a high Reynolds number, we are only able to capture large scales and obtain the same resolution as a LES (see § 1.1.2). In addition to the resolution limitation, 2D measurements are inherently sparse in space, as they only provide the velocity components within planes, and in time. We can eventually set an experimental configuration to retrieve the velocity fields at the inlet and the stream-wise planes alternatively in time (see § 4.4.3), however we still lack the whole 3D contribution of the measured flow.

3D measurements are even more limited in size due to the tremendous cost of energy of the laser used to illuminate the flow. As described in § 1.2.2, tomoPIV measurements are limited to a small volume of a few cm^3 and the velocity dynamics range is smaller than the 2D measurement case. Therefore, similarly to the 2D measurements, we can't retrieve the smaller scales and obtain observations which are LES like. The recent large-scale tomo-PIV techniques can eventually retrieve a larger domain of a few m^3 (Kuhn et al., 2011; Scarano et al., 2015), however the obtained observations are sparse in order to avoid occlusions of the particles.

In a nutshell, EFD techniques provide important characteristics of a realistic flow. However, the measurements size and resolution are limited by the experimental settings and are often sparse in time and in space. Therefore, we obtain a LES-like observations which captures the large scales of the flow of interest. Let us note this LES-like representation is not model-free since strong smoothing functions are used in the estimation process.

1.3 Coupling CFD with EFD

Traditionally, experimental observations are used as a reference for a numerical method result as in Dong et al. (2006) and in Parnaudeau et al. (2008), while the model can be used to deduce non-observable variables characterizing the fluid dynamics. Recently, the CFD and the EFD communities have both shown great interest in combining a dynamical model with observations to obtain a better representation of a given flow motion. We present an overview of the early attempts to introduce measurements into a model and vice versa. A particular focus is brought to the promising data assimilation techniques that couple a model with the observations of a system of interest. We describe the recent attempts to apply a data assimilation technique in the fluid mechanics context and finally present the approach proposed by this thesis.

1.3.1 Early attempts to combine measurements with a model

Early attempts to introduce measurements into a model in the fluid mechanics community stem from the application of the Proper Orthogonal Decomposition (POD). Initially, POD was used in the CFD community to reduce large CFD models. The popular snapshot method proposed by Sirovich (1987) provides a set of modes from a set of instantaneous flow solution obtained from a simulation of the CFD method. The governing equations of the large CFD models are then projected onto the reduced space spanned by the POD modes. The snapshot method was thereafter improved by Everson and Sirovich (1995) in the image processing context, to reconstruct of images - human face - from a sequence of incomplete or gappy images. The first application of the POD in the aerodynamics context was carried out by Bui-Thanh et al. (2004) to reconstruct flow fields corresponding to a given airfoil shape, using a set of incomplete experimental PIV measurements. In the fluid mechanics context, Druault et al. (2004) use a POD model which combines a LES model with HWA and PIV measurements to generate realistic inflow conditions for a numerical simulation. This procedure can also be used as a dynamical data post-processing which aims at improving the experimental data, for instance by completing partial information or restoring the data deterioration occurred during the measurement acquisition. More recently, Suzuki (2014) applied a POD model by combining with time-resolved PIV/PTV measurements for a turbulent flow at Reynolds $Re = 2000$.

Optimal design problems aim at determining the optimal input parameters corresponding to measured forces exerted on the fluid. In this context, we seek for an optimal design parameter, for instance an initial condition (Lundvall et al., 2006) or a material characteristic (Avril et al., 2008), such that the error between the model-based solution and the true measurements reaches a minimum. To that purpose, optimal design problems are formulated as an optimal control problem (Lions, 1971) which is solved by an adjoint-based control optimization, as pioneered by Jameson (1995). In this aspect, optimal design problems are similar to the variational approach of the data assimilation techniques that we present in the following sub-section.

1.3.2 Towards data assimilation techniques

Data assimilation (DA) is a whole branch of mathematical techniques that emerged from the weather forecasting and geophysics fields. One of the most seducing assets of the data assimilation techniques is their ability to retrieve the state of interest in a full spatial and temporal resolution by combining the dynamics model describing the system of interest with low resolution and partial observations. The application of DA techniques to the fluid mechanics context was first investigated in D'Adamo et al. (2007) and in Cuzol et al. (2007) to provide low representations of the flow characteristics. We usually distinguish two main approaches to data assimilation techniques, a sequential data assimilation (SDA) approach which estimates an optimal state from a first guess state and its observation ; and a variational data assimilation (VDA) approach which seeks for an optimal trajectory which is a compromise between the evolution of a system and a set of observations in a given assimilation window.

Application of the SDA approach In the SDA approach, Colburn et al. (2011) employed the Ensemble Kalman Filter (EnKF) technique, introduced by Evensen (2003), to estimate the 3D near-wall flow at a low Reynolds number (Re=100). This application combines a standard spectral second order finite difference approach with measurements of the skin friction and pressure only on the wall. The results are a first step to determine the dynamics law of turbulent near-wall flows with respect to the distance from the wall. Kato and Obayashi (2013) employed the EnKF technique to retrieve the optimal parameters of the Spalart-Allmaras turbulence model. These parameters were initialized by random values within a defined range and were compared to the original values proposed by the Spalart-Allmaras turbulence model. Combés et al. (2015) proposed to reconstruct the 3D geometry dynamics of a free surface flow by coupling Kinect depth measurements with SWE dynamics model via stochastic filtering.

Application of the VDA approach In the VDA approach, a first study was carried out by Papadakis and Mémin (2009) to couple turbulent flow image sequences with DNS. This study was followed by Cuzol and Mémin (2009) who coupled turbulent flow image sequences with vortex particle simulations. Suzuki et al. (2009a,b) propose a hybrid unsteady-flow simulation technique which combines DNS and PTV which is very similar to the variational approach formulation. The latter is used to and is compared to a 2D URANS solution. These first attempts are reviewed in Heitz et al. (2010). The VDA approach has later on been used by Gronskis et al. (2013) to reconstruct a 2D accurate full resolution inflow and initial condition by coupling PIV measurements and a highly accurate DNS model at Reynolds $Re = 400$. This reconstruction was performed by controlling the initial and inlet conditions. In the same vein,

Dovetta et al. (2014) recovered a turbulent pipe flow statistics at high Reynolds ($Re = 37155$) by performing the VDA approach with a RANS model and mean velocity measurement taken from McKeon et al. (2004). In the same vein of Fujisawa et al. (2005) and Van Oudheusden (2005), a variational assimilation technique was

carried by Lemke and Sesterhenne (2013) to reconstruct pressure fields for acoustic fluctuations. The latter combines an adjoint-based formulation of the fully 2D compressible Navier Stokes equation, with synthetic PIV observations to determine an improved full resolution of the velocity fields. The pressure field is thereafter deduced by solving the Poisson equation with the obtained velocity fields. The resulting pressure fields revealed to be more accurate than those obtained directly from solving the Poisson equation on the observed PIV field. Foures et al. (2014) combined a 2D RANS model with several measures of the mean flow for a low Reynolds number ($Re = 100$). These studies laid out the VDA asset to interpolate, reconstruct and extrapolate the flow statistics from partial observations, in regions where measurements are difficult or impossible to obtain. The VDA approach was also used in Mons et al. (2014) to reconstruct the initial kinetic energy spectrum by coupling an eddy-damped quasi-normal Markovian model with the observation of ah homogeneous isotropic turbulence decay.

Hybrid approach More recently, there is an emergence of hybrid methods which combine the SDA and the VDA approaches which is introduced in § 2.5. A complete review of such method can be found in Yang (2014). In chapter 3, we considered the application of the hybrid technique En4DVar and compare its results with a classic VDA approach.

In the framework of this thesis, we carried out two applications of the VDA approach. In the one hand, we reconstruct a free surface height and velocity field by coupling a 2D SWE model and depth map observations provided by the Kinect sensor. On the other hand, we combine a DNS model with stereo-PIV observations to reconstruct the three components three dimension velocity fields of a cylinder wake at $Re = 300$. In extension to Gronskis et al. (2013), we perform the reconstruction by controlling the initial and inflow conditions. One of the biggest challenge of this second application is to retrieve the whole volume in the wake of an unknown obstacle from 2D observations.

Chapitre 2

Data assimilation techniques

The concept of data assimilation (DA) comes from the numerical weather prediction (NWP) field. The goal is to combine an *a priori* knowledge of a given system with observations of the system as formulated in section § 2.1. When we apply data assimilation techniques to a given dynamic system (i.e. we consider the evolution in time), we ought to fulfill three main purposes : improve our knowledge of the present state from the past observations, predict a better state in the future from the past and present observations and provide a better estimate of the evolution of the state, referred to as the trajectory, of the system of interest. Early attempts to combine a model with a set of observations in the NWP context were carried out through the interpolation of the observations to the dynamic numerical grid points. Pioneering works by Bergthorsson and Döös (1955) and Cressman (1959), later combined with a statistical interpretation by Gandin (1965), together gave birth to a methodology called optimal interpolation (OI). The estimation theory offers a wide range of parametric estimators for a state of interest given a set of observations, which is discussed in section § 2.2. Thereafter, two main approaches to data assimilation techniques stood out and were developed mostly independently from one another. In the one hand, the sequential approach, discussed in § 2.3, resorts to the Kalman Filter equations which require the calculation of a single and expensive matrix (called the Kalman gain), and on the other hand, the variational approach, discussed in section § 2.4, is based on the application of optimal control theory elements to the data assimilation problem. Finally, we discuss in § 2.5 the emergence of the hybrid data assimilation techniques which combines the concepts of the sequential and varitional approaches.

2.1 General context

First and foremost we define the mathematical context common to the data assimilation techniques described throughout this chapter. The common goal of data assimilation techniques is to provide an optimal estimate of a true initial state $\mathbf{x}_0^t$ and control parameter $u^t(t,x)$ that are in practice unknown. The obtained optimal state is referred to as the analysis state, denoted $\mathbf{x}_0^a$. Data assimilation techniques require two main ingredients. In the one hand, we need a dynamic model that describes the evolution of the system of interest, named the background trajectory, from an *a*

priori initial state $\mathbf{x}_0^b$. On the other hand, we need a set of measurements $\mathcal{Y}$ of the system. In the fluid mechanics context, the state variables of interest are typically the velocity and pressure fields and the observations are obtained by PIV or tomoPIV measurements. The data assimilation problem is formulated by the following system of equations,

$$\partial_t \mathbf{x}(t,x) + \mathbb{M}(\mathbf{x}(t,x)) = q(t,x), \tag{2.1}$$
$$\mathbf{x}(t_0,x) = \mathbf{x}_0^b(x) + \eta(x), \tag{2.2}$$
$$\mathcal{Y}(t,x) = \mathbb{H}(\mathbf{x}(t,x)) + \varepsilon(t,x). \tag{2.3}$$

The first equation (2.1) describes through a differential operator $\mathbb{M}$ the evolution of the state $\mathbf{x}$ in time and space t, x. The differential operator $\mathbb{M}$ is often non-linear ; in the fluid mechanics context for instance, (2.1) corresponds to the general form of the Navier Stokes equations (1.1),(1.2). Function $q(x,t)$ takes into account potential model errors, such as dynamical parameter errors or errors induced by the discretization of the continuous dynamics equations into the numerical solver. In the framework of this thesis, we assume that the dynamics model is perfect, thus $q(t,x) = 0$.

The second equation (2.2) describes the initial condition for the dynamics equation which is defined by the background state $\mathbf{x}_0^b$ up to the background error $\eta(x)$. The latter is supposed to have a zero mean and is associated to a so-called background covariance matrix,

$$\mathbf{B} = \mathbb{E}((\mathbf{x}_0^b - \mathbf{x})(\mathbf{x}_0^b - \mathbf{x})^T) = \mathbb{E}(\eta\eta^T).$$

The last equation (2.3) links the measurements $\mathcal{Y}$ of the system with the state variable $\mathbf{x}$ for each occurrence of the observation t. This relation is characterized by a differential observation operator $\mathbb{H}$ which can be non-linear. The measurements $\mathcal{Y}$ are often noisy and can be completely or partially defined in time and space at a different scale than the state of interest $\mathbf{x}$. The discrepancy error between the measurement and the state in the observation space, denoted $\varepsilon(t,x) = \mathcal{Y}(t,x) - \mathbb{H}(\mathbf{x}(t,x))$ and is assumed to be a zero mean Gaussian random field associated to the observation covariance tensor $\mathbf{R}$. The observation error $\varepsilon(t,x)$ covers at the same time the measurement errors induced by the measurement sensors and by the observation operator $\mathbb{H}$. In the former case, we usually have an information on the mean and standard deviation corresponding to the adopted measurement sensor. Whereas, in the latter case, it is not trivial to estimate the errors induced directly by the observation operator which depends on a given model. In both applications studied within this thesis, we will consider pseudo observations which are equivalent to the state of interest and which are obtained from a specified pre-process, therefore the observation operator will be set to an identity matrix.

2.2 Linear estimation theory

Estimation theory is a branch of statistics which provides a basis for data assimilation techniques. We consider a noisy observation $\mathcal{Y} \in \mathbb{R}^m$ of a given physical

system, linked to a state variable $\mathbf{x} \in \mathbb{R}^n$ by an observation operator. We will consider throughout this section a linear observation operator noted $\mathbf{H}$ such as

$$\mathcal{Y} = \mathbf{H}\mathbf{x} + \varepsilon.$$

The discrepancy error ε between the observations and the state variable is assumed to be unbiased and is associated to the following covariance matrix

$$R = \mathbb{E}((\mathcal{Y} - \mathbf{H}\mathbf{x})(\mathcal{Y} - \mathbf{H}\mathbf{x})^T) = \mathbb{E}(\varepsilon\varepsilon^T).$$

Estimation theory aims at determining and studying the properties of an optimal estimator $\hat{\mathbf{x}}$ of the true state $\mathbf{x}^t$, based on the knowledge of the observation $\mathcal{Y}$. In the Numerical Weather Prediction (NWP) context, we have an *a priori* knowledge of the data behavior, thus, we are interested into parametric estimators which are very dependent on the adjustment goodness of the data to the density functions. A wide range of estimators can be found in the literature, we present two main approaches to solve the given linear estimation problem.

2.2.1 Linear least squares estimation

Given the estimation problem above, the linear Least Squares (LS) estimation formulation is straightforward. In this approach, we seek for the optimal estimator $\hat{\mathbf{x}}$ that minimizes the following cost function

$$J(\mathbf{x}) = \frac{1}{2}(\mathcal{Y} - \mathbf{H}\mathbf{x})^T(\mathcal{Y} - \mathbf{H}\mathbf{x}).$$

The optimal state is obtained by canceling the cost function gradient :

$$\nabla J = \mathbf{H}^T\mathbf{H}\mathbf{x} - \mathbf{H}^T\mathcal{Y} = 0,$$

supposing $\mathbf{H}^T\mathbf{H}$ is reversible, we deduce the optimal linear LS estimator :

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathcal{Y}.$$

This formulation can be generalized by taking into consideration a weight, described by the observation covariance matrix R, into the cost function :

$$J(\mathbf{x}) = \frac{1}{2}(\mathcal{Y} - \mathbf{H}\mathbf{x})^T\mathbf{R}^{-1}(\mathcal{Y} - \mathbf{H}\mathbf{x}).$$

We deduce, the generalized optimal linear LS estimator is given by

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathcal{Y}.$$

We note that the LS approach is purely deterministic technique, which relies on a Gaussian *a priori* of the discrepancy errors between the data and the state variables. Such a statistic analysis can be pushed forward within a Bayesian context.

2.2.2 Bayesian approach

In the Bayesian estimation, we consider the state vector $\mathbf{x}$ as a random variable associated to the probability distribution $P(\mathbf{x})$. We assume that the observation $\mathcal{Y}$ is also a random vector. In this approach, we seek for an estimation of the true (unobservable) state $\mathbf{x}^t$ knowing the observed data $\mathcal{Y}$ associated to the conditional probability $P(\mathbf{x}|\mathcal{Y})$. By applying the Bayes' rule, the latter is written as

$$P(\mathbf{x}|\mathcal{Y}) = \frac{P(\mathcal{Y}|\mathbf{x})P(\mathbf{x})}{P(\mathcal{Y})}, \tag{2.4}$$

where $P(\mathcal{Y}|\mathbf{x})$ is the distribution of the data given the un-observable state $\mathbf{x}^t$. As the observation data come from imperfect measurements, $P(\mathcal{Y}|\mathbf{x})$ quantifies the distribution of measurement errors, it can reflect possible biases as well as instrument errors. $P(\mathbf{x})$ is the prior distribution which provides an *a priori* understanding of the quantities of interest. Within this PhD framework, the prior distribution will be based on the dynamics model defined by physics laws. The marginal distribution $P(\mathcal{Y})$ is in general not needed as it defines a normalizing constant that can be replaced by the following definition,

$$P(\mathcal{Y}) = \int P(\mathcal{Y}|\mathbf{x})P(\mathbf{x})dx.$$

Given all the previous elements, a given Bayesian estimator $\hat{\mathbf{x}}$ is associated to the conditional probability distribution :

$$P(\mathbf{x}|\mathcal{Y}) = \frac{P(\mathcal{Y}|\mathbf{x})P(\mathbf{x})}{\int P(\mathcal{Y}|\mathbf{x})P(\mathbf{x})}.$$

A Bayesian estimator $\hat{\mathbf{x}}$ is built by minimizing the Bayes risk noted $\mathbb{E}(C(\hat{\mathbf{x}}, \mathbf{x}))$, where $C(\hat{\mathbf{x}}, \mathbf{x})$ is associated to a certain loss function. For instance, the Minimum variance estimator (MV), noted $\mathbf{x}^a_{MV}$, considers the Bayes risk as the mean square error,

$$C(\hat{\mathbf{x}}, \mathbf{x}) = ||\hat{\mathbf{x}} - \mathbf{x}||^2 = Tr\left((\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T\right),$$

where Tr is the trace of a given matrix. Therefore, we aim at minimizing the cost function J_{MV} given by the expectation of $||\mathbf{x} - \hat{\mathbf{x}}||^2$ knowing $\mathcal{Y}$, which reads after further simplification :

$$J_{MV} = \mathbb{E}\left(||\hat{\mathbf{x}} - \mathbf{x}||^2|\mathcal{Y}\right) = \int (\hat{\mathbf{x}} - \mathbf{x})^T(\hat{\mathbf{x}} - \mathbf{x})P(\mathbf{x}|\mathcal{Y})dx. \tag{2.5}$$

The estimator is obtained by nullifying the cost function gradient with respect to $\hat{\mathbf{x}}$,

$$\nabla J_{MV} = \int 2(\hat{\mathbf{x}} - \mathbf{x})P(\mathbf{x}|\mathcal{Y})dx = 0.$$

The unique estimator that minimizes J_{MV} is given by :

$$\mathbf{x}^a_{MV} = \int \mathbf{x} P(\mathbf{x}|\mathcal{Y}) d\mathbf{x} = \mathbb{E}(\mathbf{x}|\mathcal{Y}).$$

2.2.3 Best Linear Unibiased Estimator (BLUE)

We discuss in this section another approach which is based on the knowledge of the error covariance matrices without specifying first the probability densities $P(\mathbf{x}|\mathcal{Y})$ and $P(\mathbf{x})$. We consider the background error $\boldsymbol{\eta} = \mathbf{x}^t - \mathbf{x}^b$ associated with the background error covariance matrix

$$\mathbf{B} = \mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^T), \tag{2.6}$$

and we consider the observation error $\boldsymbol{\varepsilon} = \mathcal{Y} - \mathbf{H}\mathbf{x}^t$ associated with the observation error covariance matrix

$$\mathbf{R} = \mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T), \tag{2.7}$$

We assume that the background and observation error are unbiased, i.e. $\mathbb{E}(\boldsymbol{\eta}) = 0$ and $\mathbb{E}(\boldsymbol{\varepsilon}) = 0$. In this context, we seek for the optimal estimate $\mathbf{x}^a$ defined by the following linear combination :

$$\mathbf{x}^a = \mathbf{L}\mathbf{x}^b + \mathbf{K}\mathcal{Y}. \tag{2.8}$$

First and foremost, we assume that the analysis error $\boldsymbol{\varepsilon}^a = \mathbf{x}^a - \mathbf{x}^t$ is unbiased. By introducing the definition of the background and observation errors into (2.8) we obtain :

$$\mathbb{E}(\boldsymbol{\varepsilon}^a) = E(\mathbf{x}^a - \mathbf{x}^t) = \mathbf{L}E(\boldsymbol{\eta}) + \mathbf{K}E(\boldsymbol{\varepsilon}) + (\mathbf{L} + \mathbf{K}\mathbf{H} + \mathbf{Id})E(\mathbf{x}^t) = 0.$$

As the background and the observation errors are unbiased, we easily establish the value of $\mathbf{L}$ and deduce the expression of the analaysis state $\mathbf{x}^a$:

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{K}(\mathcal{Y} - \mathbf{H}\mathbf{x}^b), \tag{2.9}$$

where the vector $\mathcal{Y} - \mathbf{H}\mathbf{x}^b$ is called the innovation vector and the matrix $\mathbf{K}$ is referred to as the gain matrix. The latter is unknown and has to be defined along with the analysis covariance matrix noted $\mathbf{A}$. If we assume in addition that $\mathbf{x}$ and $\mathcal{Y}$ are both Gaussian random vectors and the background and observation errors are non-correlated, i.e. $E(\boldsymbol{\varepsilon}\boldsymbol{\eta}^T) = 0$, then the analysis state (2.8), referred to as the Best Linear Unbiased Estimator (BLUE), is exactly equivalent to the MVE described in § 2.2.2. Under the previous hypotheses, we seek for the gain matrix $\mathbf{K}$ which minimizes the variance $E(||\boldsymbol{\varepsilon}^a||^2)$ by using the information provided by covariance matrices (2.6) and (2.7). We consider the following cost function by injecting (2.8) into (2.5) :

$$J_{BLUE} = \mathbb{E}(||\mathbf{x}^a - \mathbf{x}^t||^2) = \int (\mathbf{x}^b - \mathbf{x}^t + \mathbf{K}(\mathcal{Y} - \mathbf{H}\mathbf{x}^b))^T (\mathbf{x}^b - \mathbf{x}^t + \mathbf{K}(\mathcal{Y} - \mathbf{H}\mathbf{x}^b)) P(\mathbf{x}|\mathcal{Y}) dx,$$

then we cancel its gradient with respect to the gain matrix $\mathbf{K}$

$$\frac{\partial J_{BLUE}}{\partial \mathbf{K}} = \int (\mathcal{Y} - \mathbf{H}\mathbf{x}^b)^T (\mathbf{x}^b - \mathbf{x}^t + \mathbf{K}(\mathcal{Y} - \mathbf{H}\mathbf{x}^b)) P(\mathbf{x}|\mathcal{Y}) dx = 0.$$

We use the background and observation errors and their proprieties definition to deduce the gain matrix (2.10) and the analysis covariance matrix (2.11). The reader can find the details in the appendice A. The complete procedure to determine the BLUE is summarized in algorithm 1.

$$\mathbf{K} = \mathbf{B}\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}, \tag{2.10}$$

$$\mathbf{A} = E((\mathbf{x}^a - \mathbf{x}^t)(\mathbf{x}^a - \mathbf{x}^t)) = \mathbf{B} - \mathbf{B}\mathbf{H}^T(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{B} \tag{2.11}$$

Algorithm 1 Best Linear Unbiased Estimator (BLUE)

1: Compute the gain matrix

$$\mathbf{K} = \mathbf{B}\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}$$

2: Deduce the analysis and its covariance error matrix

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{K}(\mathcal{Y} - \mathbf{H}\mathbf{x}^b)$$
$$\mathbf{A} = E((\mathbf{x}^a - \mathbf{x})(\mathbf{x}^a - \mathbf{x})) = (\mathbf{Id} - \mathbf{K}\mathbf{H})\mathbf{B}$$

We point out that so far we didn't take into account the state variable dynamics. At this point, two main data assimilation approaches emerged. On the one hand, the sequential approach, presented in § 2.3, couples the results of the Bayesian estimation, in particular the BLUE, with a dynamic model. On the other hand, the variational approach, described in § 2.4, uses the results of optimal control theory by imposing the dynamic model as a constraint and aims at minimizing the least squares cost function written in terms of covariance matrices :

$$J(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^b)\mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b) + (\mathcal{Y} - \mathbb{H}(\mathbf{x}))\mathbf{R}^{-1}(\mathcal{Y} - \mathbb{H}(\mathbf{x})).$$

2.3 Sequential approach

The sequential approach can be seen as an extension of the Bayesian minimum variance estimation which propagates the background state forward in time by a dynamical model. As the name suggests, the goal of sequential data assimilation consists in correcting the background state each time a new observation is available. We discuss throughout this section the Kalman filter and its extensions.

2.3.1 Kalman filter

The Kalman Filter introduced by Kalman (1960) can be seen as the direct extention of the BLUE for dynamics model (see § 2.2.3). In this approach, we seek for

the analysis state $\mathbf{x}^a_{k+1}$ that corrects the background state, that we refer to as the forecast state $\mathbf{x}^f_{k+1}$, given the observation $\mathcal{Y}_{k+1}$ at the instant t_{k+1} as illustrated in figure 2.1.

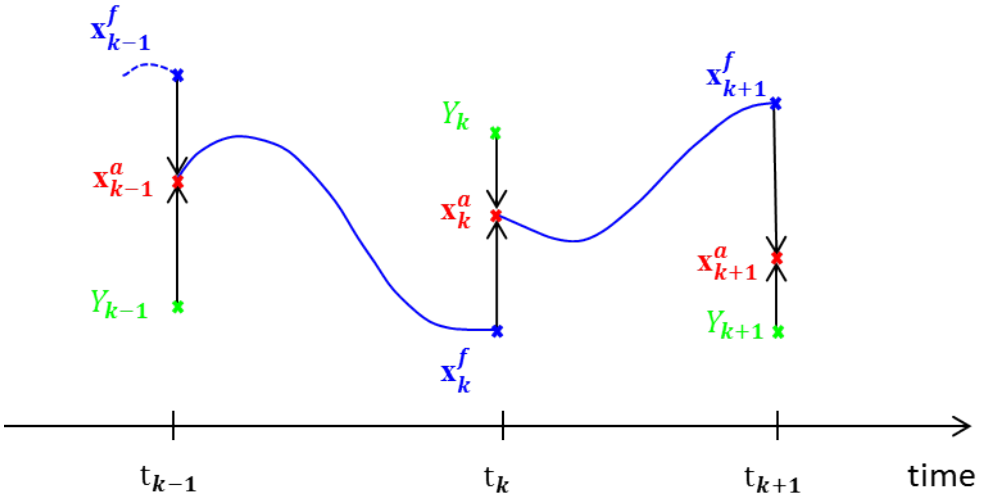

FIGURE 2.1 – Sequential data assimilation representation : we seek for the analysis state $\mathbf{x}^a_{k+1}$ that corrects the forecast state $\mathbf{x}^f_{k+1}$ generated by the dynamics model given the observation $\mathcal{Y}_{k+1}$ at the instant t_{k+1}. The dynamics model start from the analysis state $\mathbf{x}^a_k$ of a previous assimilation process.

We consider the following dynamic linear model and observation defined by equations (2.12) and (2.13),

$$\mathbf{x}^f_{k+1} = \mathbf{M}_{k,k+1}\mathbf{x}^f_k, \tag{2.12}$$
$$\mathcal{Y}_{k+1} = \mathbf{H}_k\mathbf{x}^t_{k+1} + \boldsymbol{\varepsilon}_k, \tag{2.13}$$

where the linear discrete operator $\mathbf{M}_{k,k+1}$ describes the flow operator that links the state evolution between the instant t_k and t_{k+1}. This model imperfections are represented by the model error q_k associated to the error covariance matrix $\mathbf{Q}_k$ such as,

$$\mathbf{x}^t_{k+1} = \mathbf{M}_{k,k+1}\mathbf{x}^t_k + \mathbf{q}_k.$$

In the same way, the observation operator $\mathbf{H}_k$ is assumed linear and the error observation $\boldsymbol{\varepsilon}_{k+1}$ at the instant t_{k+1} is associated to the observation covariance matrix $\mathbf{R}_{k+1}$.

We observe immediately that the forecast error is propagated through the following forecast covariance matrix $\mathbf{P}^f_{k+1}$,

$$\mathbf{P}^f_{k+1} = E((\mathbf{x}^f - \mathbf{x}^t)^T(\mathbf{x}^f - \mathbf{x}^t)) = \mathbf{M}_{k,k+1}\mathbf{P}^a_k\mathbf{M}^T_{k,k+1} + \mathbf{Q}_k.$$

The Kalman filter approach considers the same hypotheses as those of the BLUE given in § 2.2.3, and assumes in addition that the model and observation errors aren't correlated, in other words,

$$E(\boldsymbol{\varepsilon}_k\mathbf{q}^T_k) = 0.$$

The analysis state $\mathbf{x}^a$ is also associated to an error covariance matrix $\mathbf{P}^a$,

$$\mathbf{P}^a = E((\mathbf{x}^a - \mathbf{x}^t)(\mathbf{x}^a - \mathbf{x}^t)^T).$$

Under all the previous hypotheses, the Kalman Filter algorithm, given by algorithm 2, consists into an analysis step, similar to the BLUE algorithm, and an additional forecast step, that invokes the dynamical model $\mathbf{M}_{k,k+1}$ and computes the forecast covariance matrix.

Algorithm 2 Kalman filter

1: **Analysis step** similar to the BLUE algorithm

$$\mathbf{K} = (\mathbf{H}_k\mathbf{P}_k^f)^T \left(\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^f)^T + \mathbf{R}_k\right)^{-1} \tag{2.14}$$

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \mathbf{K}_k(\mathcal{Y} - \mathbf{H}x_k^f) \tag{2.15}$$

$$\mathbf{P}_k^a = (\mathbf{Id} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f \tag{2.16}$$

2: **Forecast step** performed by using the model

$$\mathbf{x}_{k+1}^f = \mathbf{M}_{k,k+1}\mathbf{x}_k^a \tag{2.17}$$

$$P_{k+1}^f = \mathbf{M}_{k,k+1}\mathbf{P}_k^a\mathbf{M}_{k,k+1}^T + \mathbf{Q}_k \tag{2.18}$$

2.3.2 Extended Kalman Filter

The Kalman filter can be extended to non-linear dynamics and observation operators. In this case, we consider the linear tangent operator around the analysis state $\mathbf{x}_k^a$, noted $\partial_{\mathbf{x}}\mathcal{M}_k$, of the non-linear dynamics operator $\mathcal{M}_{k,k+1}$ and the linear tangent operator around $\mathbf{x}_k^f$, noted $\partial_{\mathbf{x}}\mathbb{H}_k$, of the non-linear observation operator $\mathcal{H}_{k+1}$. The tangent linear operators are obtained by the application of the Taylor series :

$$\mathcal{M}_{k,k+1}(\mathbf{x}^a(t_k)) = \mathcal{M}_{k,k+1}(\mathbf{x}^f(t_k)) + \partial_{\mathbf{x}}\mathcal{M}_k(\mathbf{x}^a(t_k) - \mathbf{x}^f(t_k)) + O(\|\boldsymbol{\varepsilon}_k|^2),$$
$$\mathcal{H}_k(\mathbf{x}^a(t_k)) = \mathcal{H}_k(\mathbf{x}^f(t_k)) + \partial_{\mathbf{x}}\mathcal{H}_k(\mathbf{x}^a(t_k) - \mathbf{x}^f(t_k)) + O(\|\boldsymbol{\varepsilon}_k|^2).$$

This extension, called the Extended Kalman Filter (EKF), is illustrated by algorithm 3.

The major drawback of the classic and extended Kalman filter lays in the computation of the Kalman gain filter matrix $\mathbf{K}$ and the error covariance matrices $\mathbf{P}_k^a$ and $\mathbf{P}_k^f$, which requires a significant computation and storage resources of the squared size of the state space vector. Therefore, these techniques are only usable for systems of low size which isn't suitable for NWP, geophysics and in particular, fluid mechanics applications. Besides, we point out that the EKF provides a good estimation for slightly nonlinear models, which is restrictive for many applications. If we consider for instance the Navier Stokes equations as the dynamics model of interest, its associated tangent linear model operator is likely to neglect the non-linear effects and compromise the optimality of the analysis.

Algorithm 3 Extended Kalman filter

1: **Forecast step**
2: Compute the forecast using the nonlinear dynamic operator

$$\mathbf{x}^f_{k+1} = \mathcal{M}_{k,k+1}\mathbf{x}^a_k \tag{2.19}$$

3: Compute the forecast covariance matrix $\mathbf{P}^f_{k+1}$ depending on the tangent linear and adjoint dynamic model

$$\mathbf{P}^f_{k+1} = \partial_{\mathbf{x}}\mathcal{M}_k\mathbf{P}^a_k\partial_{\mathbf{x}}\mathcal{M}^*_k + \mathbf{Q}_k \tag{2.20}$$

4: **Analysis step**
5: Compute the Kalman gain matrix and deduce the analysis state

$$\mathbf{K}_{k+1} = \mathbf{P}^f_{k+1}\partial_{\mathbf{x}}\mathcal{H}^*_k\left(\partial_{\mathbf{x}}\mathcal{H}_k\mathbf{P}^f_{k+1}\partial_{\mathbf{x}}\mathcal{H}^*_k + \mathbf{R}_k\right)^{-1} \tag{2.21}$$

$$\mathbf{x}^a_{k+1} = \mathbf{x}^f_{k+1} + \mathbf{K}_{k+1}(\mathcal{Y}_{k+1} - \partial_{\mathbf{x}}\mathcal{H}\mathbf{x}^f_{k+1}) \tag{2.22}$$

6: Update the analysis error covariance matrix $\mathbf{P}^a_{k+1}$

$$\mathbf{P}^a_{k+1} = (\mathbf{Id} - \mathbf{K}_{k+1}\partial_{\mathbf{x}}\mathcal{H}_{k+1})\mathbf{P}^f_{k+1} \tag{2.23}$$

2.3.3 Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF), first introduced in Evensen (1994), proposes an efficient alternative to the EKF using a Monte-Carlo approach. In this context, we consider a set, called ensemble, of possible state samples, each referred to as an ensemble member, noted $\{\mathbf{x}^{f,(1)}(t+1), \cdots, \mathbf{x}^{f,(N)}(t+1)\}$. This ensemble is generated from the randomization of the non linear dynamics model given the result $\mathbf{x}^a(t)$ of a previous assimilation. Instead of directly computing and storing the error covariance matrices, the EnKF uses an empirical expression of the ensemble mean and covariance matrix to deduce a low-rank approximation of the gain matrix and the covariance matrices $\mathbf{P}^f$ and $\mathbf{P}^a$. The EnKF process is illustrated in figure 2.2.

The more ensemble members are employed, the more accurate the approximations are and the more computational resources is required. A balance policy must be clearly chosen. However, ours is always reduced to choose a number of ensemble members N much smaller than the size of the state of interest, noted n. Similarly to the sequential data assimilation techniques presented throughout this section, the EnKF consists in the following forecast and analysis steps.

Ensemble forecast

In the EnKF context, the ensemble forecast consists in generating and propagating the ensemble of N ensemble members by the nonlinear dynamics. Each ensemble member is obtained by perturbing the given forecast trajectory associated to the initial forecast state $\mathbf{x}^f(t_0)$. As we deal with nonlinear dynamics, each ensemble

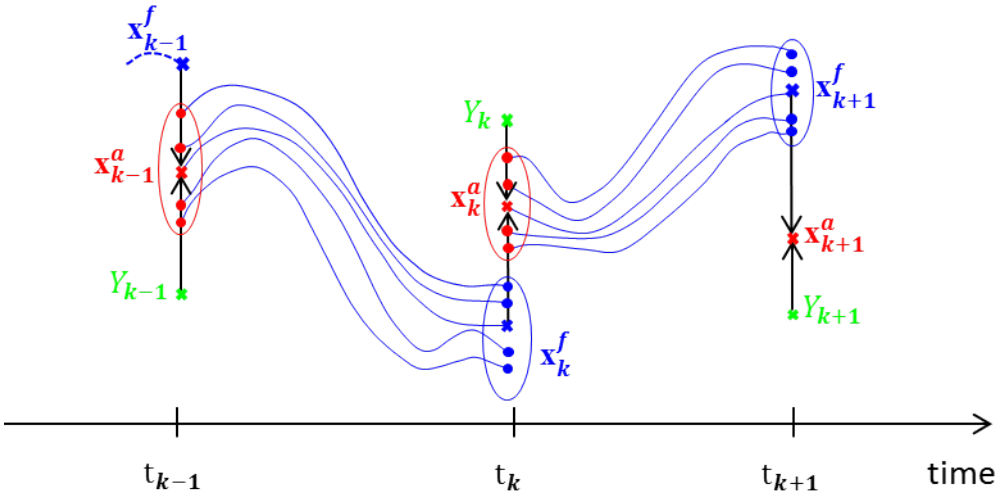

FIGURE 2.2 – Ensemble Kalman Filter representation : we consider a set, called ensemble, of possible state samples, each referred to as an ensemble member, noted $\{\mathbf{x}^{f,(1)}_{k+1}, \cdots, \mathbf{x}^{f,(N)}_{k+1}\}$ (blue dots). The latter is generated from the randomization of the non linear dynamics model given the result $\mathbf{x}^a_k$ of a previous assimilation. The analysis ensemble is obtained by correcting the forecast ensemble (red dots) with the observation Y_{k+1} (green dot) and is associated with the analysis covariance error matrix P^e (red ellipse).

member can be generated by adding a perturbation $\delta^{(i)}$ to the initial forecast, such that

$$\mathbf{x}^{f,(i)}(t_0) = \mathbf{x}^f_0 + \delta^{(i)}, \forall i \in \{1, \cdots, N\},$$

and propagating each $\mathbf{x}^{f,(i)}(t_0)$ by the time integration in $[t_0, t]$ of the nonlinear model operator $\mathcal{M}_t$. We point out however that more and more ensemble systems are generated by perturbing the model parameters, which seems to provide a more accurate ensemble spread as studied in Wei et al. (2013) and in Wu et al. (2008).

The ensemble members are then gathered into the following matrix,

$$\mathbf{X}^f_t := \left[\mathbf{x}^{f,(1)}(t), \cdots, \mathbf{x}^{f,(1)}(t)\right] \in \mathbb{R}^{n \times N},$$

and the empirical ensemble mean of the N ensemble members is noted

$$< \mathbf{X}^f_t >:= \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{f,(i)}_t.$$

The forecast step ends with the estimation of the following ensemble covariance matrix $\mathbf{P}^e$, which implicitly captures the dominant directions of uncertainty of the forecast error,

$$\mathbf{P}^f \approx \mathbf{P}^e = \frac{1}{N}(\mathbf{X}^f_t - < \mathbf{X}t^f >)(\mathbf{X}^f_t - < \mathbf{X}^f_t >)^T. \tag{2.24}$$

Ensemble analysis

We saw previously that the analysis procedure consists in computing the gain matrix $\mathbf{K}_k$, deducing the analysis state $\mathbf{x}_k^a$ and finally updating the analysis covariance matrix $\mathbf{P}_k^a$ from the forecast $\mathbf{P}_k^f$. In the ensemble context, there are two main approach to deduce the analysis estimate and evaluate the analysis errors.

In the same vein as the pioneering work of Evensen (1994) and taking into account the work of Burgers et al. (1998), a first approach consists in generating an ensemble of measurement samples in the same way as the ensemble forecast $(\mathbf{x}_t^{f,(i)})_{i=1,\cdots,N}$. Therefore, each ensemble observation is defined as

$$\mathcal{Y}_t^{(i)} = \mathcal{Y}_t + \boldsymbol{\varepsilon}_t^{(i)}, i = 1, \cdots, N. \tag{2.25}$$

At this point, the most intuitive process consist to replace $\mathbf{P}_k^f$ by the matrix $\mathbf{P}^e$ (2.24) to compute the Kalman gain matrix $\mathbf{K}$,

$$\mathbf{K}_t = \mathbf{P}^e\mathbb{H}(\mathbb{H}\mathbf{P}^e\mathbb{H}^T + \mathbf{R})^{-1}, \tag{2.26}$$

In this context, the observation operator is nonlinear, therefore the terms $\mathbf{P}^e\mathbb{H}$ and $\mathbb{H}\mathbf{P}^e\mathbb{H}^T$ are approximated by

$$\mathbf{P}^f\mathbb{H} = \frac{1}{N-1}(\mathbf{X}_t^f - < \mathbf{X}_t^f >)(\mathbb{H}(\mathbf{X}_t^f) - < \mathbb{H}(\mathbf{X}_t^f) >)^T, \tag{2.27}$$

$$\mathbb{H}\mathbf{P}^f\mathbb{H}^T = \frac{1}{N-1}(\mathbb{H}(\mathbf{X}_t^f) - < \mathbb{H}(\mathbf{X}_t^f) >)(\mathbb{H}(\mathbf{X}_t^f) - < \mathbb{H}(\mathbf{X}_t^f) >)^T. \tag{2.28}$$

We then deduce the analysis state

$$\mathbf{x}_t^{a,(i)} = \mathbf{x}_t^{f,(i)} + \mathbf{K}_t(\mathcal{Y}_t^{(i)} - \mathbb{H}(\mathbf{x}_t^{f,(i)}). \tag{2.29}$$

At last, the update of the analysis matrix (2.23) is replaced by the mean of the variance of the ensemble forecast errors given by

$$\mathbf{P}_t^{a,(i)} = \frac{1}{N-1}(\mathbf{X}_t^a - < \mathbf{X}_t^a >)(\mathbb{H}(\mathbf{X}_t^a) - < \mathbb{H}(\mathbf{X}_t^a) >)^T. \tag{2.30}$$

2.4 Variational approach

The variational approach was first introduced by Sasaki (1958) and expanded rapidly in the 80s with the works of Le Le Dimet and Talagrand (1986) and Courtier and Talagrand (1987). In opposition to the sequential approach, the variational approach considers a given assimilation window in which we have a so-called background trajectory, generated by the dynamics from an initial background state $\mathbf{x}_0^b$, and a set of observations $\mathcal{Y}$. The goal of this approach consists in changing gradually the initial background state $\mathbf{x}_0^b$ to draw the trajectory closer to the observations by solving an optimal control problem as described by Lions (1968, 1971). In practice, this approach comes to minimize an objective functional which measures the mismatch between the model predictions and complete or partial measurements of the

system, weighed by inverse covariance matrices. This optimization is conducted under the constraint that the optimal state obeys to the model dynamics. Therefore, we expect the assimilation results to satisfy two basic requirements. On the one hand, the assimilation trajectory has to be close to the observations, ideally to the accuracy of the observations themselves. On the other hand, the trajectory has to be consistent with the physical relations between each variables and the evolution of the flow imposed by the model.

2.4.1 Variational formulation

The data assimilation problem can be formulated as the minimization of a least square cost function expressed in terms of the background and the observation covariance matrices, respectively $\mathbf{B}$ and $\mathbf{R}$. In this context, we consider the minimization of such function over an assimilation window $[t_0; t_f]$ such as

$$J(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_0^b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b) + \frac{1}{2}\int_{t_0}^{t_f} (\mathbb{H}(\mathbf{x}_t) - \mathcal{Y}(t))^T \mathbf{R}^{-1}(\mathbb{H}(\mathbf{x}_t) - \mathcal{Y}(t)).$$

A complete integration of the dynamics model on a given assimilation widow $[t_0; t_f]$, provides a trajectory of the state, associated to an initial guess $\mathbf{x}_0$ and a flow map $\varphi_t(\mathbf{x}_0)$ such as :

$$\mathbf{x}_t := \varphi_t(\mathbf{x}_0) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbb{M}(\mathbf{x}(s))ds.$$

We place the data assimilation problem within the control optimal theory (Lions, 1968, 1971) framework. Formally, let $\mathcal{V}$ be a Hilbert space identified to his dual space $\mathcal{U}$. We assume that the state variable $\mathbf{x}_t \in \mathcal{V}$ and the observations $\mathcal{Y} \in \mathcal{O}$ are square integrable functions in Hilbert spaces identified to their dual. The norms corresponds to the Mahalanobis distance defined from the inner products $< \mathbf{R}^{-1}., . >_{\mathcal{O}}$, $< \mathbf{B}^{-1}., . >_{\mathcal{V}}$ of the measurements and the state variable spaces respectively. They involve covariance matrices $\mathbf{R}$, and $\mathbf{B}$ related to the measurement error and the error on the initial condition.

The minimization is performed by a gradient-based algorithm which, as its name suggests, requires the evaluation of the cost function gradient $\nabla J(\mathbf{x}_0)$ with respect to the initial state $\mathbf{x}_0$. The optimization algorithm usually requires the computation of the gradient for several values of $\mathbf{x}_0$ to converge to the optimal solution. Therefore one of the major issue of the variational approach lays in the computation of an accurate numerical gradient. There are basically three approaches to obtain numerically such gradient.

The finite difference schemes consist in approximating the gradient by varying each variables of the cost function. Let's consider a first order finite difference scheme is given by

$$\nabla J(\mathbf{x}_0) \approx \frac{J(\mathbf{x}_0 + \alpha \mathbf{d}) - J(\mathbf{x}_0)}{\alpha},$$

where d is an arbitrary direction. In order to meet the accuracy requirements, we need to fix a sufficiently small differentiation step α. Another alternative consists in

increasing the numerical scheme order p which costs $2p-1$ computation and storage of the cost function. This comes to $2p-1$ temporal integration of the nonlinear model for each iteration of the gradient based algorithm. Therefore, this approach is clearly unsuitable to our problem.

The sensitivity equation gives the information on how the flow will react to a given perturbation $d\mathbf{x} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}} \delta\boldsymbol{\eta}$. In opposition to the finite difference approach, we don't need several evaluations of the cost function for different inputs. The sensitivity equation is obtained by the linearization of the cost function J around $\mathbf{x}_0$ in the direction $\delta\boldsymbol{\eta}$. The differentiation of the perfect nonlinear equations (2.1) and (2.2) in the direction $\delta\boldsymbol{\eta}$ reads :

$$\partial_t d\mathbf{x}(t,x) + \partial_{\mathbf{x}}\mathbb{M}(\mathbf{x}(t,x)) d\mathbf{x}(t,x) = 0, \tag{2.31}$$

$$d\mathbf{x}_0(x) = \delta\boldsymbol{\eta}(x), \tag{2.32}$$

where $\partial_{\mathbf{x}}\mathbb{M}(\mathbf{x})$ denotes the tangent linear operator associated to the non-linear operator $\mathbb{M}$ defined as :

$$\partial_{\mathbf{x}}\mathbb{M}(\mathbf{x}) d\mathbf{x} = \lim_{\beta \to 0} \frac{\mathbb{M}(\mathbf{x} + \beta d\mathbf{x}) - \mathbb{M}(\mathbf{x})}{\beta}. \tag{2.33}$$

We deduce the differentiation of the cost function in the direction $\delta\boldsymbol{\eta}$:

$$\left\langle \frac{\partial J}{\partial \boldsymbol{\eta}}, \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} = \left\langle \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b), \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} - \int_{t_0}^{t_f} \left\langle \mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)), \partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}_t) \frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}} \delta\boldsymbol{\eta} \right\rangle_{\mathcal{O}} dt. \tag{2.34}$$

Introducing the adjoint of the linear tangent observation operator $(\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}))^*$, defined as :

$$\forall (u,v) \in (\mathcal{V}, \mathcal{O}), < (\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}))\, u, v >_{\mathcal{O}} = < u, (\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}))^* v >_{\mathcal{V}}, \tag{2.35}$$

the relation (2.34) can be reformulated as :

$$\left\langle \frac{\partial J}{\partial \boldsymbol{\eta}}, \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} = \left\langle \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b), \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} - \int_{t_0}^{t_f} \left\langle (\partial_X \mathbb{H})^* \mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbf{x}_t), \frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}} \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} dt. \tag{2.36}$$

Equation (2.36) describes the functional gradients in the directions $\delta\boldsymbol{\eta}$. For the whole gradient, if we consider p the number of design variables, we have to solve p sensitivity equations. In the framework of our PhD, for instance, we ought to control all three components of the velocity fields in a $p \approx 10^6$ volume, therefore this approach is too expensive.

The adjoint formulation is an elegant alternative based on the work of Lions (1971) and Le Le Dimet and Talagrand (1986). We consider a so-called adjoint variable $\boldsymbol{\lambda} \in \mathcal{V}$ and carry out the integration over the range $[t_0, t_f]$ of the inner product between $\boldsymbol{\lambda}$ and the relation (2.31) such as :

$$\int_{t_0}^{t_f} \Big\langle \frac{\partial d\mathbf{x}}{\partial t}(t), \boldsymbol{\lambda}(t) \Big\rangle_{\mathcal{V}} dt + \int_{t_0}^{t_f} \Big\langle \partial_{\mathbf{x}}\mathbb{M} d\mathbf{x}(t), \boldsymbol{\lambda}(t) \Big\rangle_{\mathcal{V}} dt = 0. \tag{2.37}$$

An integration by parts of the first term yields :

$$-\int_{t_0}^{t_f} \Big\langle -\frac{\partial \boldsymbol{\lambda}}{\partial t}(t) + (\partial_{\mathbf{x}}\mathbb{M})^* \boldsymbol{\lambda}(t), d\mathbf{x}(t) \Big\rangle_{\mathcal{V}} dt = \big\langle \boldsymbol{\lambda}(t_f), d\mathbf{x}(t_f) \big\rangle_{\mathcal{V}} - \big\langle \boldsymbol{\lambda}(t_0), d\mathbf{x}(t_0) \big\rangle_{\mathcal{V}}, \tag{2.38}$$

where the adjoint of the tangent linear operator $(\partial_{\mathbf{x}}\mathbb{M})^* : \mathcal{V} \to \mathcal{V}$ has been introduced. At this point no particular assumptions nor constraints were imposed on the adjoint variable. However, we are free to specify the set of adjoint variables of interest by setting a particular evolution equation or a given boundary conditions simplifying the computation of the functional gradient. As we will see, imposing the adjoint variable λ as the solution of the system

$$\begin{cases} -\partial_t \boldsymbol{\lambda}(t) + (\partial_{\mathbf{x}}\mathbb{M})^* \boldsymbol{\lambda}(t) = (\partial_{\mathbf{x}}\mathbb{H})^* \mathbf{R}^{-1} (\mathcal{Y} - \mathbb{H}(\mathbf{x}(t))) \\ \boldsymbol{\lambda}(t_f) = 0, \end{cases} \tag{2.39}$$

will provide us a simple and accessible solution for the functional gradient.

As a matter of fact, injecting this relation into equation (2.38) with $d\mathbf{x}(t_0) = \delta\boldsymbol{\eta}$ and $d\mathbf{x} = (\partial\mathbf{x}/\partial\boldsymbol{\eta})\delta\boldsymbol{\eta}$ allows us to identify the right hand second terms of the functional gradients (2.36) and we get :

$$\Big\langle \frac{\partial J}{\partial \eta}, \delta\boldsymbol{\eta} \Big\rangle_{\mathcal{V}} = -\Big\langle \boldsymbol{\lambda}(t_0), \delta\boldsymbol{\eta} \Big\rangle_{\mathcal{V}} + \Big\langle \mathbf{B}^{-1}(\delta\mathbf{x}(t_0) - \delta\mathbf{x}_0), \delta\boldsymbol{\eta} \Big\rangle_{\mathcal{V}}. \tag{2.40}$$

From these relations, one can now readily identify the expression of the cost function derivative with respect to the control variable

$$\frac{\partial J}{\partial \boldsymbol{\eta}} = -\boldsymbol{\lambda}(t_0) + \mathbf{B}^{-1}(\delta\mathbf{x}(t_0) - \delta\mathbf{x}_0). \tag{2.41}$$

In the adjoint-based approach, we only require to solve the adjoint equation once. We note that if the nonlinear dynamics equation is time dependent, then the adjoint equation is time dependent and is solved backwards in time.

2.4.2 Standard 4DVar technique

The aim of the standard 4DVar technique consists in determining the initial optimal state $\mathbf{x}^a(t_0)$ from the so called background trajectory, associated to an initial guess $\mathbf{x}^b(t_0)$ and generated by the dynamics model, and a set of observations defined within a given assimilation window. The optimal state leads to an analysis trajectory which is a compromise between the background trajectory and the observations. The standard 4DVar technique is illustrated by figure 2.3.

A complete integration of the dynamics model on a given assimilation widow $[t_0; t_f]$, provides a trajectory of the state, referred to as $\mathbf{x}_t$, associated to an initial guess $\mathbf{x}_0$ and a flow map $\varphi_t(\mathbf{x}_0)$ such as :

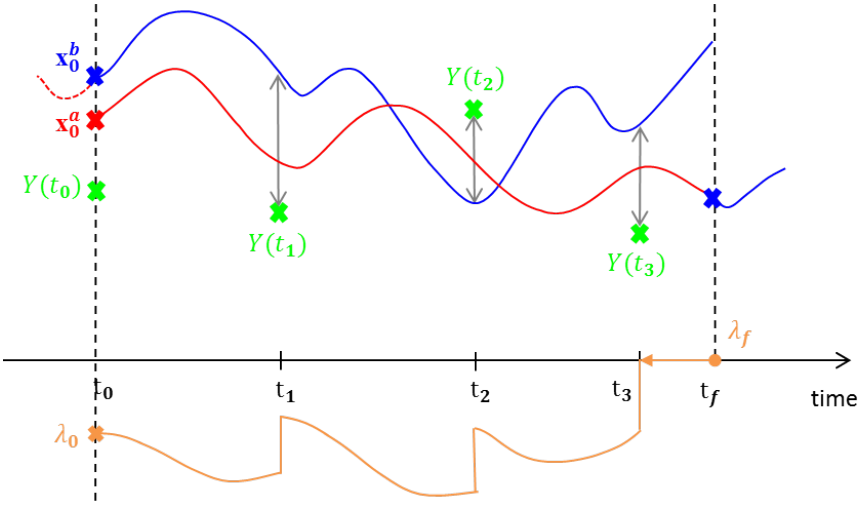

FIGURE 2.3 – Standard 4DVar assimilation representation : we aim at determining the optimal assimilation trajectory (red curve) from the background trajectory (blue curve) and a set of observations (green). We minimize the distance between the background trajectory and the observations by a gradient-descent algorithm. The cost function gradient is deduced by performing an adjoint procedure backwards in time (orange curve) which takes into account the observations correction.

$$\mathbf{x}_t := \varphi_t(\mathbf{x}_0) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbb{M}(X(s))ds.$$

We ought to minimize the distance between the background trajectory defined by (2.1) and (2.2), and the observations (2.3), by controlling the initial state $\mathbf{x}_0$, as formulated by the following cost function :

$$J(\mathbf{x}_0) = \frac{1}{2}\|\mathbf{x}_0^b - \mathbf{x}(t_0, x)\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)\|^2_{\mathbf{R}^{-1}} dt, \tag{2.42}$$

where the $\mathbb{L}^2$ norm is defined as $\|f\|_A^2 = \int_\Omega f(x)A(x,y)f(y)dxdy$. The associated minimization problem is referred to in the literature as the strong constraint variational assimilation formulation.

The background and observation covariance matrices, respectively $\mathbb{B}$ and $\mathbb{R}$, define the variation of the background and observations through the variance and the dependence of each variable to one another through the covariance. The inverted matrices used in the cost function aim at normalizing each term and define the weight of the observation with respect to the background. These matrices guide the minimization process by determining which data should be given more confidence at each mesh point of the grid.

In order to solve the optimal control problem defined by (2.42), (2.1), (2.2) and (2.3), we resort to an iterative optimization method. The literature provides a large range of iterative methods to solve an optimization problem under constraint. In this thesis, we opt for the limited storage gradient-based optimization method LBFGS proposed by Nocedal (1980).

We now have all elements in hand to establish the standard 4DVar assimilation algorithm 4.

Algorithm 4 Standard Variational assimilation technique

Initialization: $k = 0$ and $\mathbf{x}_0^k = \mathbf{x}_0^b$

1: **repeat**
2: Compute $\mathbf{x}_t$ with the forward integration (2.1) and the cost function (2.42)
3: Compute the initial adjoint variable $\boldsymbol{\lambda}(t_0)$ by the backward integration (2.39)
4: Update the initial state $\mathbf{x}_0^{k+1} = \mathbf{x}_0^k + \mathbf{B}\boldsymbol{\lambda}(t_0)$
5: **until** $\|\mathbf{x}_0^{k+1} - \mathbf{x}_0^k\| < tol$
6: From $\mathbf{x}_0^a = \mathbf{x}_0^{k+1}$ deduce the analysis trajectory $\mathbf{x}_t^a$

2.4.3 Incremental Variational Assimilation

As the previous standard 4DVar assimilation cost function (2.42) implies the use of the nonlinear model, there is a chance that the optimization procedure leads to a local minima. Whereas, if we deal with a convex cost function, we are guaranteed to find the global minima. It is in this perspective that Courtier et al. (1994) introduced the incremental strategy to improve the 4DVar technique. This optimization strategy consists in performing a linearization of the dynamics around the current trajectory and minimizes a convex cost function with respect to an increment $\delta\mathbf{x}_0$ instead of the initial state $\mathbf{x}_0$. The optimal increment is then used to update the initial state and we carry out the linearization and optimization again for a fixed number of iterations. The incremental 4DVar strategy is illustrated in figure 2.4. This procedure is similar to a Gauss-Newtoon minimization.

The initial increment $\delta\mathbf{x}_0$ is propagated by the following tangent dynamics equation linearized around a given state $\mathbf{x}$,

$$\partial_t \delta\mathbf{x}(t,x) + (\partial_{\mathbf{x}}\mathbb{M})\delta\mathbf{x}(t,x) = 0, \tag{2.43}$$

$$\delta\mathbf{x}_0(x) = (\mathbf{x}_0^b(x) - \mathbf{x}_0(x)) + \eta(x), \tag{2.44}$$

where $(\partial_{\mathbf{x}}\mathbb{M})$ is the tangent operator associated to the nonlinear dynamics operator $\mathbb{M}$ (2.33). Consequently, the cost function is expressed in terms of the initial increment $\delta\mathbf{x}_0$,

$$J(\delta\mathbf{x}_0) = \frac{1}{2}\|\delta\mathbf{x}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\mathbb{H}(\mathbf{x}_t) - \mathcal{Y}(t)\|^2_{\mathbf{R}^{-1}} dt, \tag{2.45}$$

where we consider the perturbed trajectory

$$\mathbf{x}_t = \varphi(\mathbf{x}_t^b) + \partial_{\mathbf{x}}\varphi_t(\mathbf{x}_t^b)\delta\mathbf{x}_0.$$

The cost function (2.45) can be alternatively formulated as :

$$J(\delta\mathbf{x}_0) = \frac{1}{2}\|\delta\mathbf{x}_0^b - \delta\mathbf{x}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\partial_{\mathbf{x}}\mathbb{H}\delta\mathbf{x}(t,x) - D(t,x)\|^2_{\mathbf{R}^{-1}} dt, \tag{2.46}$$

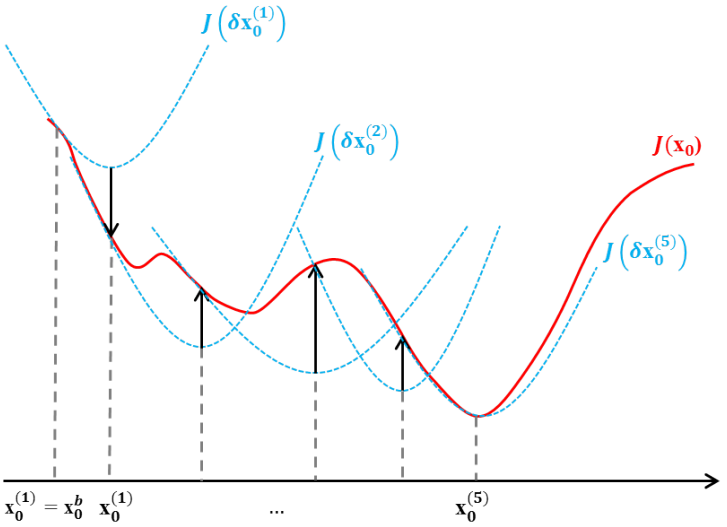

FIGURE 2.4 – Incremental variational assimilation representation : we seek for the optimal increment $\delta \mathbf{x}_0^{(1)}$ of the linearized convex cost function around the initial state $\mathbf{x}_0^{(1)} = \mathbf{x}_0^b$ (blue curve) and unpdate to $\mathbf{x}_0^{(2)} = \mathbf{x}_0^{(1)} + \delta \mathbf{x}_0^{(1)}$. We repeat the procedure until we reach the optimal state $\mathbf{x}_0$ which minimizes the nonlinear cost function J (red curve).

where

$$\mathbf{x}_t = \partial_{\mathbf{x}} \varphi_t(\mathbf{x}_t^b) \delta \mathbf{x}_0,$$

and the innovation vector $D(t, x)$ is defined by

$$D(t, x) = \mathcal{Y}(t, x) - \mathbb{H}(\varphi_t(\mathbf{x}_t^b)).$$

As in the non-incremental case, we perform a gradient descent optimization with the LBFGS algorithm and we compute the incremental cost function gradient with the adjoint equation which reads in this case :

$$\begin{cases} -\partial_t \boldsymbol{\lambda}(t) + (\partial_{\mathbf{x}} \mathbb{M})^* \boldsymbol{\lambda}(t) = (\partial_{\mathbf{x}} \mathbb{H})^* \mathbf{R}^{-1} (\partial_{\mathbf{x}} \mathbb{H} \partial_{\mathbf{x}} \varphi(\mathbf{x}_0) D(t, x)) \\ \boldsymbol{\lambda}(t_f) = 0. \end{cases} \tag{2.47}$$

The gradient is then deduced by :

$$\partial_{\delta \mathbf{x}_0} J(\delta \mathbf{x}_0) = -\boldsymbol{\lambda}(t_0) + \mathbf{B}^{-1} \delta \mathbf{x}_0. \tag{2.48}$$

Once we determine the optimal increment, we update the former initial state $\mathbf{x}_0$ and deduce its corresponding trajectory by solving the nonlinear dynamics equations. The updated trajectory can then be used as a linearization state and we can perform a second outer loop optimization around the new state. The incremental variational assimilation technique proposed by Courtier et al. (1994) is illustrated by algorithm 5.

In the standard case we seek to update directly the background trajectory and in the incremental approach we only use the tangent procedure to update an incremental trajectory. The nonlinear dynamics is used for the outerloop only.

Algorithm 5 Incremental 4DVar assimilation technique (Courtier et al., 1994)

Initialization: $\mathbf{x}_0^{(1)} = \mathbf{x}_0^b$
Forward integration of the nonlinear dynamics (2.1)
for $k = 1, N_k$ **do**
Set the initial increment as : $\delta\mathbf{x}_0^{(k)} = \delta\mathbf{x}_0^b$
repeat
Compute $\delta\mathbf{x}_t$ with the tangent linear dynamics (2.43) and (2.44) and the cost function
Compute the adjoint variable $\boldsymbol{\lambda}(t_0)$ with the backward integration of relation (2.47)
Update the initial increment value $\delta\mathbf{x}_0^{(k)} = \mathbf{B}\boldsymbol{\lambda}(t_0)$
until $\|\mathbf{x}_0^{k+1} - \mathbf{x}_0^k\| < tol$
Update the background : $\mathbf{x}_0^{(k+1)} = \mathbf{x}_0^{(k)} + \delta\mathbf{x}_0^{(k)}$
end for
From $\mathbf{x}_0^a = \mathbf{x}_0^{(k+1)}$ deduce the analysis trajectory $\mathbf{x}_t^a$

2.4.4 Inflow and initial conditions control

In the framework of this thesis, we aim at reconstructing the downstream of a cylinder wake by combining a DNS model with three velocity components measurements of the flow in the inflow and in the stream-wise planes (see § 4.1). In this configuration, noisy or incomplete inlet conditions (with no or erroneous small and even large scales information) may lead to an unrealistic background trajectory which might compromise the convergence of the variational assimilation technique employed. This issue can be tackled by gradually modifying the inflow condition at each time step by including the latter as a control variable in the variational assimilation formulation. This approach has been successfully carried out by Gronskis et al. (2013) to generate inflow and initial conditions of a 2D turbulent flow. The idea of controlling the inflow was also carried out in Habert et al. (2014) with an Extended Kalman Filter in the context of real-time flood forecasting.

Within this context, we introduce the control parameter $u(t)$ which is defined in the space of square integrable function on spatio-temporal domain U_T. Hence, we consider the data assimilation system,

$$\partial_t \mathbf{x}(t,x) + \mathbb{M}(\mathbf{x}(t,x), \mathbf{u}(t)) = 0, \tag{2.49}$$

$$\mathbf{x}_0(x) = \mathbf{x}_0^b(x) + \boldsymbol{\eta}(x). \tag{2.50}$$

In addition to the background error $\boldsymbol{\eta}(x)$ and observation error $\boldsymbol{\varepsilon}(t,x)$, associated to the covariance error matrices $\mathbf{B}$ and $\mathbf{R}$ respectively, we consider the deviation $\theta(t)$ between the control $\mathbf{u}(t)$ an its *a priori* value $\mathbf{u}^c(t)$ associated to the covariance error matrix $\mathbf{B}_c$. Therefore, our goal consists in minimizing the cost function (2.51) with respect to the control variable $\boldsymbol{\gamma}(t,x) = \{\boldsymbol{\eta}(x), \boldsymbol{\theta}(t)\} = \{\mathbf{x}(x,t_0) - \mathbf{x}_0(x), \mathbf{u}(t) -$

$\mathbf{u}^c(t)\}$.

$$J(\boldsymbol{\gamma}) = \frac{1}{2}\int_{t_0}^{t_f} \|\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)\|^2_{\mathbf{R}^{-1}} dt + \frac{1}{2}\|\mathbf{x}(t_0,x) - \mathbf{x}_0^b(x)\|^2_{\mathbf{B}^{-1}} \\ \frac{1}{2}\int_{t_0}^{t_f} \|\mathbf{u}(t) - \mathbf{u}^c(t)\|^2_{\mathbf{B}_c^{-1}} dt, \tag{2.51}$$

with $\|f\|^2_{\mathbf{B}_c^{-1}} = \int_{\mathbf{\Gamma}} f(x)\mathbf{B}_c^{-1}(x,y)f(y)dxdy$ where Γ refers to the inflow section. Given a perturbation $d\mathbf{x} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}}\delta\mathbf{u}(t) + \frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}}\delta\boldsymbol{\eta}$, the differentiation of the equations (2.49) and (2.50) in the direction $\delta\boldsymbol{\gamma} = (\delta\mathbf{u}, \delta\boldsymbol{\eta})$ reads :

$$\partial_t d\mathbf{x}(t,x) + \partial_{\mathbf{x}}\mathbb{M}(\mathbf{x}(t,x),\mathbf{u}(t))d\mathbf{x}(t,x) + \partial_{\mathbf{u}}\mathbb{M}(\mathbf{x}(t,x),\mathbf{u}(t))d\mathbf{u}(t) = 0, \tag{2.52}$$

$$d\mathbf{x}_0(x) = \delta\boldsymbol{\eta}(x). \tag{2.53}$$

The differentiation of the cost function in the direction $\delta\boldsymbol{\gamma}$:

$$\left\langle \frac{\partial J}{\partial \mathbf{u}}, \delta\mathbf{u} \right\rangle_{U_T} = \int_{t_0}^{t_f} \left\langle \mathbf{B}_c^{-1}(\mathbf{u}(t) - \mathbf{u}^c(t)), \delta\mathbf{u}(t) \right\rangle_U \\ - \int_{t_0}^{t_f} \left\langle \mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)), \partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}_t)\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\delta\mathbf{u}(t) \right\rangle_{\mathcal{O}} dt, \tag{2.54}$$

$$\left\langle \frac{\partial J}{\partial \boldsymbol{\eta}}, \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} = \left\langle \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b), \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} \\ - \int_{t_0}^{t_f} \left\langle \mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)), \partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}_t)\frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}}\delta\boldsymbol{\eta} \right\rangle_{\mathcal{O}} dt. \tag{2.55}$$

Considering the linear tangent observation operator $(\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}))^*$ introduced in (2.35), we can reformulate the equations (2.54) and (2.55) by :

$$\left\langle \frac{\partial J}{\partial \mathbf{u}}, \delta\mathbf{u} \right\rangle_{U_T} = \int_{t_0}^{t_f} \left\langle \mathbf{B}_c^{-1}(\mathbf{u}(t) - \mathbf{u}^c(t)), \delta\mathbf{u}(t) \right\rangle_U \\ - \int_{t_0}^{t_f} \left\langle (\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}_t))^*\mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)), \frac{\partial \mathbf{x}}{\partial \mathbf{u}}\delta\mathbf{u}(t) \right\rangle_{\mathcal{V}} dt \tag{2.56}$$

$$\left\langle \frac{\partial J}{\partial \boldsymbol{\eta}}, \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} = \left\langle \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b), \delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} \\ - \int_{t_0}^{t_f} \left\langle (\partial_{\mathbf{x}}\mathbb{H}(\mathbf{x}_t))^*\mathbf{R}^{-1}(\mathcal{Y}(t) - \mathbb{H}(\mathbf{x}_t)), \frac{\partial \mathbf{x}}{\partial \boldsymbol{\eta}}\delta\boldsymbol{\eta} \right\rangle_{\mathcal{V}} dt. \tag{2.57}$$

As it is unfeasible to compute directly the cost function gradients, we resort to a similar adjoint approach than in § 2.4.1. To that end, we proceed to the integration over the range $[t_0, t_f]$ of the inner product between an adjoint variable $\boldsymbol{\lambda} \in \mathcal{V}$ and the relation (2.52) is given by :

$$\int_{t_0}^{t_f} \left\langle \frac{\partial d\mathbf{x}}{\partial t}(t), \boldsymbol{\lambda}(t) \right\rangle_{\mathcal{V}} dt + \int_{t_0}^{t_f} \left\langle \partial_{\mathbf{x}}\mathbb{M}d\mathbf{x}(t), \boldsymbol{\lambda}(t) \right\rangle_{\mathcal{V}} dt + \int_{t_0}^{t_f} \left\langle \partial_{\mathbf{u}}\mathbb{M}\delta\mathbf{u}(t), \boldsymbol{\lambda}(t) \right\rangle_{\mathcal{V}} dt = 0. \tag{2.58}$$

An integration by parts of the first term yields :

$$-\int_{t_0}^{t_f}\Big\langle -\frac{\partial\lambda}{\partial t}(t)+(\partial_{\mathbf{x}}\mathbb{M})^*\boldsymbol{\lambda}(t),d\mathbf{x}(t)\Big\rangle_{\mathcal{V}}dt=\ \big\langle\boldsymbol{\lambda}(t_f),d\mathbf{x}(t_f)\big\rangle_{\mathcal{V}}-\big\langle\boldsymbol{\lambda}(t_0),d\mathbf{x}(t_0)\big\rangle_{\mathcal{V}}+\int_{t_0}^{t_f}\Big\langle\delta\mathbf{u}(t),(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}(t)\Big\rangle_{U}dt. \tag{2.59}$$

As in § 2.4.1, we impose the adjoint variable $\boldsymbol{\lambda}$ as the solution of (2.39) and inject the latter into (2.59) with $d\mathbf{x}(t_0)=\delta\boldsymbol{\eta}$ and $d\mathbf{x}=(\partial\mathbf{x}/\partial\boldsymbol{\eta})\delta\boldsymbol{\eta}$ to identify the right hand second terms of the functional gradients (2.56) and (2.57). Hence, we get

$$\begin{aligned}\Big\langle\frac{\partial J}{\partial\mathbf{u}},\delta\mathbf{u}\Big\rangle_{U_T}&=\int_{t_0}^{t_f}\Big\langle\delta\mathbf{u}(t),\mathbf{B}_c^{-1}(\mathbf{u}(t)-\mathbf{u}^c(t))+(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}(t)\Big\rangle_{U}dt\\&=\Big\langle\mathbf{B}_c^{-1}(\mathbf{u}-\mathbf{u}^c)+(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda},\delta\mathbf{u}\Big\rangle_{U_T}\\\Big\langle\frac{\partial J}{\partial\boldsymbol{\eta}},\delta\boldsymbol{\eta}\Big\rangle_{\mathcal{V}}&=-\Big\langle\boldsymbol{\lambda}(t_0),\delta\boldsymbol{\eta}\Big\rangle_{\mathcal{V}}+\Big\langle\mathbf{B}^{-1}(\mathbf{x}_0-\mathbf{x}_0^b),\delta\boldsymbol{\eta}\Big\rangle_{\mathcal{V}}.\end{aligned}$$

Thus, we deduce the expression of the cost function derivative with respect to $\delta\boldsymbol{\gamma}=(\delta\boldsymbol{\eta},\delta\mathbf{u})$:

$$\frac{\partial J}{\partial\mathbf{u}}=-\boldsymbol{\lambda}(t_0)+\mathbf{B}_c^{-1}(\mathbf{u}-\mathbf{u}^c)+(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}, \tag{2.60}$$

$$\frac{\partial J}{\partial\boldsymbol{\eta}}=-\boldsymbol{\lambda}(t_0)+\mathbf{B}^{-1}(\mathbf{x}(t_0)-\mathbf{x}_0). \tag{2.61}$$

All in all, the optimal control problem defined by (2.51), (2.49), (2.50) and (2.3) is solved by the LBFGS method (see § 2.4.1), the assimilation algorithm is summarized in 7.

Algorithm 6 4DVar assimilation with inflow and initial condition control (Gronskis et al., 2013)

Initialization: $k=0$ and $\mathbf{x}_0^k=\mathbf{x}_0^b$
  repeat
    Compute $\mathbf{x}_t$ with the forward integration (2.49) and the cost function (2.51)
    Compute the initial adjoint variable $\boldsymbol{\lambda}(t_0)$ by the backward integration (2.39)
    Update the initial state $\mathbf{x}_0^{k+1}=\mathbf{x}_0^k+\mathbf{B}\boldsymbol{\lambda}(t_0)$
    Update the parameter model $\mathbf{u}^{k+1}=\mathbf{u}^k+\mathbf{B}_c(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}(t_0)$
  until $\|\mathbf{x}_0^{k+1}-\mathbf{x}_0^k\|<tol$
  From $\mathbf{x}_0^a=\mathbf{x}_0^{k+1}$ deduce the analysis trajectory $\mathbf{x}_t^a$

2.4.5 Incremental variational assimilation with inflow and initial condition control

In the framework of this thesis, we implemented the incremental version of the previous variational formulation which includes the control of the inflow and the

initial conditions (see § 2.4.4). In the same vein as in § 2.4.3, we aim at minimizing the following convex cost function with respect to an increment $\delta\boldsymbol{\gamma} = \{\delta\mathbf{x}, \delta u\}$

$$J(\delta\boldsymbol{\gamma}_0) = \frac{1}{2}\|\delta\mathbf{x}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\delta\mathbf{u}(t)\|^2_{\mathbf{B_c}^{-1}} dt + \frac{1}{2}\int_{t_0}^{t_f} \|\mathbb{H}(\mathbf{x}_t) - \mathcal{Y}(t)\|^2_{\mathbf{R}^{-1}} dt, \quad (2.62)$$

where the increment vector $\delta\boldsymbol{\gamma}$ verify (2.52) and (2.53).

As in the previous case, we call upon the adjoint equation (2.39) to compute the incremental cost function gradient with respect to the increment vector $\delta\boldsymbol{\gamma}$ such that

$$\partial_{\delta\mathbf{x}_0} J = -\boldsymbol{\lambda}(t_0) + \mathbf{B}^{-1}\delta\mathbf{x}_0, \quad (2.63)$$

$$\partial_{\delta\mathbf{u}} J = -\boldsymbol{\lambda}(t_0) + \mathbf{B}_c^{-1}\delta\mathbf{u} + (\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}. \quad (2.64)$$

The present 4DVar technique is described in algorithm 7 and is referred to as the 4DVar technique in the following chapters.

Algorithm 7 Incremental 4DVar assimilation technique with inflow and initial condition control

Initialization: $\mathbf{x}_0^{(1)} = \mathbf{x}_0^b$
  Forward integration of the nonlinear dynamics (2.49)
  for $k = 1, N_k$ **do**
    Set the initial increment as : $\delta\mathbf{x}_0^{(k)} = \delta\mathbf{x}_0^b$
    repeat
      Compute $\delta\mathbf{x}_t$ with the tangent linear dynamics (2.52) and (2.53)
      Compute the cost function (2.62)
      Compute the adjoint variable $\boldsymbol{\lambda}(t_0)$ with the backward integration of relation (2.47)
      Update the initial increment value $\delta\mathbf{x}_0^{(k)} = \mathbf{B}\boldsymbol{\lambda}(t_0)$
    until $\|\mathbf{x}_0^{k+1} - \mathbf{x}_0^k\| < tol$
    Update the background : $\mathbf{x}_0^{(k+1)} = \mathbf{x}_0^{(k)} + \delta\mathbf{x}_0^{(k)}$
    Update the parameter model $\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{B}_c(\partial_{\mathbf{u}}\mathbb{M})^*\boldsymbol{\lambda}(t_0)$
  end for
  From $\mathbf{x}_0^a = \mathbf{x}_0^{(k+1)}$ deduce the analysis trajectory $\mathbf{x}_t^a$

2.4.6 Numerical adjoint construction strategies

As described previously, the variational assimilation techniques employs a gradient-descent method to solve an optimal control problem. The computation of the cost function gradient is deduced from the backward integration of the adjoint equation (2.39). We discuss in this section the construction and implementation of the numerical adjoint equation corresponding to the dynamics.

There are two main strategies for the construction of a numerical adjoint associated to a given dynamics model. On the one hand, the *differentiate-then-discretize* strategy consists in determining the analytic adjoint from the nonlinear dynamics model, then discretizing the adjoint model to obtain a numerical adjoint procedure,

referred to as the continuous adjoint. On the other hand, *discretize-then-differentiate* consists in constructing the discrete adjoint directly from the discretized non-linear dynamics model. The latter can be performed by the mean of an Automatic Differentiation tool (AD), which is described in appendix C.1. Both strategies are illustrated in figure 2.5 and were compared in different contexts (Nadarajah and Jameson, 2000; Griesse and Walther, 2004).

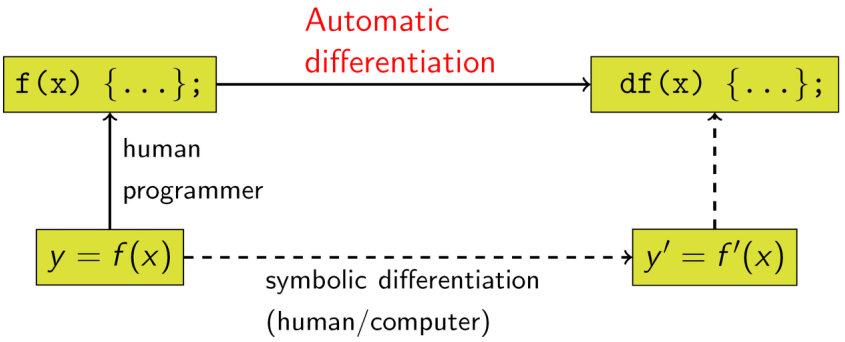

FIGURE 2.5 – Construction of the differentiation code df(x) associated to the analytic function $y = f(x)$ by the discretize-then-differentiate (solid line) and the differentiate-then-discretize (dashed line) strategies.

In general, as iterative gradient-descent algorithms are sensible to numerical errors, we seek for the most accurate numerical adjoint associated to a given problem. The discretize-then differentiate strategy provides an accurate numerical adjoint operator up to the machine precision, whereas the continous adjoint obtained by the differentiate-the-discretize strategy is accurate up to the discretization errors of the numerical scheme on a given mesh. Therefore, if we possess beforehand a sufficiently accurate original code, the discrete adjoint will remain at the same accuracy up to the machine precision. In addition, the continuous case generates boundary terms, commonly by integration by parts, that have to be dealt separately. On the other hand, the construction of discrete adjoint is straightforwardly derived from the original code and doesn't require additional thoughts to discretizations or boundary conditions for either operator.

The variational approach has been applied so far on 2D problems which require a reasonable computational cost, and most of the authors opted for the construction of an continuous adjoint (Lemke and Sesterhenne, 2013; Mons et al., 2014). De Pando et al. (2012) evaluated the direct and adjoint linearized dynamics from compressible flow solvers. In another context, Ponçot et al. (2013) constructed the discrete adjoint to implement a variational technique for xenon dynamical forecasts in neutronic. Gronskis et al. (2013) employed for the first time the discrete adjoint in the implementation of a variational technique which combines the 2D version of Incompact3d and PIV measurements. This strategy has also the great advantage to guarantee that the constructed adjoint is the adjoint of the discrete code.

Within the framework of this thesis, we build the discrete adjoint by applying the AD on the highly accurate 3D DNS code Incompact3d (see § 4.2). Unlike the continuous adjoint, the discrete adjoint takes directly advantage of the Incompact3d's optimal parallelization structure and conserves the high accurate of the original

code. We also we use the same strategy to build the discrete adjoint associated to the 2D SWE model for the reconstruction of the free surface height and velocity fields (see § 3.3). It is indeed difficult to argue this choice as the analytic adjoint is often different from the real one.

The AD implementation can also be eased by the use of a automatic differentiation (AD) tools. In terms of AD implementation, the literature offers several AD tools that employs either the source code transformation (SCT) or the operator overloading (OO) strategy (see appendix C.1.1). In this thesis, we opt for TAPENADE which is an AD tool developed by the Tropics team in Inria Sophia Antipolis (Hascoët, 2004). This tool employs the SCT strategy and was formerly successfully used in the work of Gronskis et al. (2013).

2.4.7 Validation of the adjoint procedure

The discrete tangent and adjoint procedure built by the means of the AD has to be validated by a series of tests. Before presenting the different validation tests carried out, we point out the importance of the choice of the inputs into each validation process. Indeed, in the strictly numerical sense, the validation test should be verified for a random value of the state. However, in practice the dynamics model obey to a physical model, and it has been tested that inputting a random state $\mathbf{x}$ to the original direct model f leads often to a numerical explosion. Therefore, in order to validate a dynamic model on several time steps, we have to input state $\mathbf{x}$ has to obey to the physics. We detail in the following validation tests the degrees of freedom that are adopted to this respect.

Tangent validation The tangent validation is based on the Taylor formula applied to the direct method operator f around a given state $\mathbf{x}_0$ given a direction $\mathbf{d}$,

$$f(\mathbf{x}_0 + \alpha\mathbf{d}) \approx f(\mathbf{x}_0) + \alpha < \nabla f(\mathbf{x}_0), \mathbf{d} > . \tag{2.65}$$

This validation test consists in studying the ratio

$$R_1 := \frac{f(\mathbf{x}_0 + \alpha\mathbf{d}) - f(\mathbf{x}_0)}{\alpha < \nabla f(\mathbf{x}_0), \mathbf{d} >}. \tag{2.66}$$

It is particularly interesting if we obtain the tangent operator $\nabla f(\mathbf{x}_0)$ with an AD tool.

We point out that the ratio accuracy depends on the choice of α. A too large value of α might result to accuracy errors as the dynamics operator f is non-linear, whereas a too small value of may lead to the accumulation of rounding errors. In the applications within this thesis, we study the ratios for α in a range of $\alpha \in [10^{-3}; 10^{-9}]$. We can reasonably expect the ratio R_1 to be more or less close to 1 up to $O(\alpha)$. In the latter case, it would mean that the tangent procedure would have the same accuracy that a finite difference which isn't suitable.

Dot product

The results of the reverse adjoint mode are validated by using the previously validated tangent. We recall the definition of the adjoint operator $(\partial_{\mathbf{x}}\mathbb{M})^*$:

$$< (\partial_{\mathbf{x}}\mathbb{M})\dot{X}, \dot{Y} >=< \dot{X}, (\partial_{\mathbf{x}}\mathbb{M})^*\dot{Y} >, \qquad \forall \dot{X}, \dot{Y} \tag{2.67}$$

We set in particular $\dot{Y} = (\partial_{\mathbf{x}}\mathbb{M})\dot{X}$ and define $\bar{X} := (\partial_{\mathbf{x}}\mathbb{M})^*(\partial_{\mathbf{x}}\mathbb{M})\dot{X} = (\partial_{\mathbf{x}}\mathbb{M})^*\dot{Y}$. We then develop the dot product of $\bar{X}$ and $\dot{X}$:

$$< \dot{X}, \bar{X} >=< \dot{X}, (\partial_{\mathbf{x}}\mathbb{M})^*\dot{Y} >=< (\partial_{\mathbf{x}}\mathbb{M})\dot{X}, \dot{Y} >=< \dot{Y}, \dot{Y} > . \tag{2.68}$$

This comes to reduce the following ratio to 1 :

$$R_2 := \frac{< (\partial_{\mathbf{x}}\mathbb{M})X, (\partial_{\mathbf{x}}\mathbb{M})X >}{< X, (\partial_{\mathbf{x}}\mathbb{M})^*Y >} = \frac{< \dot{Y}, \dot{Y} >}{< \dot{X}, \bar{Y} >} \approx 1. \tag{2.69}$$

The adjoint operator $(\partial_X\mathbb{M}^*)$ has to be the most accurate as possible. In other words, we have to obtain a ration of $R_2 = 1+O(\epsilon_m)$ where ϵ_M represents the machine accuracy.

Adjoint validation

In the 4DVar context, we will particularly evaluate the theoric equality between the adjoint variable $\boldsymbol{\lambda}(t_0)$ at time t_0 and the cost function gradient $\nabla J(\mathbf{x}_0)$ at time t_0 for a given direction $\mathbf{d}$, such that

$$< \nabla J(\mathbf{x}_0), \mathbf{d} >=< \lambda(t_0), \mathbf{d} > . \tag{2.70}$$

We recall that the adjoint variable $\boldsymbol{\lambda}(t_0)$ is obtained after the backwards adjoint model integration.

Similarly to the tangent validation, we apply the Taylor formula to the cost function J and evaluate the ratio (2.71) given by :

$$R_3 = \frac{J(\mathbf{x}_0 + \alpha\mathbf{d}) - J(\mathbf{x}_0)}{\alpha < \nabla J(\mathbf{x}_0), \mathbf{d} >} \approx \frac{J(\mathbf{x}_0 + \alpha d) - J(\mathbf{x}_0)}{\alpha < \boldsymbol{\lambda}(t_0), d >}. \tag{2.71}$$

In analogy to the accuracy discussions for the ratio R_1, the value of the ratio R_3 depends on the value of α. Although, we have to keep in mind that the first order finite difference scheme provides a rough approximation of the cost function gradient. Therefore, we expect don't expect the ratio to be accurate up to $10^{-\alpha}$.

2.5 Hybrid approaches

During the last decade, there is a trend in the development of hybrid data assimilation techniques which combine the advantages of ensemble methods and varitional assimilation approaches. Each technique differs according to the context in which they are developed, therefore there are two main groups of hybrid techniques :

A first group of hybrid techniques, *based on the variational formalism*, aims at incorporating an ensemble-based background covariance into a variational system. Hamill and Snyder (2000) proposed the EnKF-3DVar method which basically combines several 3DVar data assimilation process with the EnKF update process. The background error covariace is set as a linear combination of the ensemble covariance and the static covariance. This idea was extended to the 4DVar formalism by Lorenc (2003) who also introduced a localization procedure. Several authors proposed hybrid methods that express either the background error covariance as a linear combination of the ensemble covariance and the static covariance or the solution as a linear combination of the square root of the ensemble-based covariance (Liu et al., 2009; Buehner et al., 2010a,b; Clayton et al., 2013; Desroziers et al., 2014).

A second group of hybrid techniques, *based on the EnKF formalism*, incorporates the 4DVar formulation to overcome the limitations of the sequential approach. In the one hand, the standard EnKF assimilates sequentially the observation whenever they are available (Evensen, 2003). The hybrid 4D-EnKF technique proposed by Hunt et al. (2004) seeks for an observation operator which could link the state to future or past observations. This idea is then extended in the 4D-LETKF technique introduced by Hunt et al. (2007) and Fertig et al. (2007). Sakov et al. (2010) introduced the AEnKF which aims at finding the explicit form of temporal evolution of the increment by propagating the correction along the forecast system trajectory.

On the other hand, the explicit form of the update equations in EnKF (or EKF) are difficult to calculate when dealing with high dimensional problems. A group of hybrid techniques propose to replace the EnKF update by the minimization of an arbitrary cost function by an iterative gradient descent procedure similarly to the 4DVar technique which handles efficiently large dimension problems (Zupanski, 2005; Sakov et al., 2012; Solonen et al., 2012).

In the framework of this thesis, we compare an ensemble-based 4DVar, referred to as 4DEnVar, a classical 4DVar and the 4D-LETKF techniques to reconstruct a 2D free surface in chapter 3. The 4DEnVar technique is closely related to the strategy proposed by Buehner (2005) and Liu et al. (2008). This technique introduces in its objective function an empirical ensemble-based background error covariance which avoids the use of tangent linear and adjoint model. We describe the 4DEnVar in § 3.2.

Chapitre 3

A first application : reconstruction of a free surface flow

The shallow water equations (SWE) - also called Saint Venant equations - describes the height and the velocity of a thin layer of an inviscid fluid at its free surface. These equations are often used as a prototype in geophysics applications, such as ocean motion, river sedimentation and under certain hypotheses, they can also provide a satisfying atmospheric flow approximation model. The simplicity of both numerical model implementation and experimental setting (for an academical context) is an ideal framework for a first application of the 4DVar technique. In this work, we also seized the opportunity to compare the reconstruction of the dimensional free surface flow characteristics (height and 2D velocity fields) using a classic formulation of the 4DVar and an enhanced version of an ensemble based technique (Yang et al., 2015).

We present in § 3.1 the dynamical model used to simulate the free surface of the flow of interest. The data assimilation techniques used throughout this chapter are overviewed in § 3.2. We describe in particular the construction of the adjoint procedure necessary to implement the classical 4DVar technique in § 3.3. In a first step, we consider in § 3.4.1 the reconstruction of the free surface characteristics from partial and complete synthetic observations. In a second step, we consider the real measurements of the evolution of the free surface height observed with a Kinect depth sensor as described in § 3.4.2. We finally draw the conclusions of the comparison between the assimilation techniques considered for this application and layout some perspectives in § 3.5.

3.1 Dynamics

3.1.1 Hypotheses and governing equations

As its name suggests, the thickness (or height when we consider a flat bottom topology) of the fluid of interest is neglected with respect to its length and width. Under this hypothesis, the mass conservation implies that the vertical velocity of the fluid is small, hence, we only take into consideration the 2D velocity field $\mathbf{u} := (u, v)^T$. In the framework of this chapter, we consider a simplified version of the SWE under

the following hypothesis :

– The bottom of the domain is flat and non tilted.
– We neglect the Coriolis effect and the bottom friction.

As introduced in § 1.1.1, the SWE are deduced by the depth-integration of the Navier-Stokes equations and are used to model the height and the velocity field at the free surface of the fluid. The simplified 2D SWE used throughout this chapter reads

$$\partial_t h + \partial_x(hu) + \partial_y(hv) = 0, \tag{3.1}$$
$$\partial_t(hu) + \partial_x(hu^2) + \partial_y(huv) + g\partial_x h^2 = 0, \tag{3.2}$$
$$\partial_t(hv) + \partial_x(huv) + \partial_y(hv^2) + g\partial_y h^2 = 0. \tag{3.3}$$

The equations (3.1), (3.2) and (3.3) can be written in a conservative form by considering the conserved variables as $\mathbf{x} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}$ and their associated fluxes $F(\mathbf{x}) = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}$ and $G(\mathbf{x}) = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}$. The conservative form is given by :

$$\partial_t \mathbf{x}(t,x) + \partial_x F(\mathbf{x}(t,x,y)) + \partial_x G(\mathbf{x}(t,x,y)) = 0. \tag{3.4}$$

The literature provides many numerical schemes that can be used to solve the SWE model (see § 1.1.3). In this work we created from scratch a numerical code based on the work of Vreugdenhil (1994) who studied numerical solutions for 2D SWE. This code is described thereafter.

3.1.2 Spatial discretization and Roe solver

As described in § 1.1.3, the Finite Volume Method (FVM) is well suited to simulate conservation laws and is usually favored to solve CFD problems. We describe in the present section the discretization of the advection terms $F(\mathbf{x})$ and $G(\mathbf{x})$ of the conservative equation (3.4).

Finite volume formulation

The solution domain Ω is a partition of many control volumes, that are referred to as cells $\mathcal{C}_{ij} = [x_{i-1/2}; x_{i+1/2}] \times [y_{j-1/2}; y_{j+1/2}]$ with $\Delta x_i = x_{i+1/2} - x_{i-1/2}$ and $\Delta y_j = y_{j+1/2} - y_{j-1/2}$ where $i \in \{1, \dots, nx\}$ and $j \in \{1, \dots, ny\}$, as illustrated in figure 3.1.

Within our context, we assume that each cells have the same size, i.e. $\Delta x := \Delta x_i = \Delta y_j$, $\forall i, j$. On a given cell $\mathcal{C}_{ij}$, the integral form of (3.4) reads :

$$\begin{aligned} \frac{d}{dt}\int_{\mathcal{C}_{ij}} \mathbf{x}(t,x) \quad & + \tfrac{1}{\Delta x} F(\mathbf{x}(t, x_{i+1/2}, y_j)) - \tfrac{1}{\Delta x} F(\mathbf{x}(t, x_{i-1/2}, y_j)) \\ & + \tfrac{1}{\Delta y} G(\mathbf{x}(t, x_i, y_{j+1/2})) - \tfrac{1}{\Delta y} G(\mathbf{x}(t, x_i, y_{j-1/2})) = 0. \end{aligned} \tag{3.5}$$

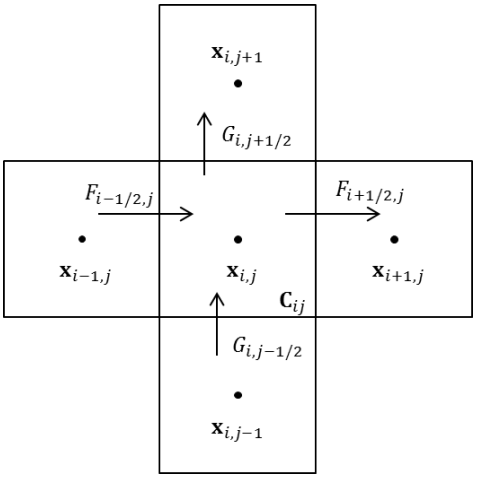

FIGURE 3.1 – Spatial discretization of the FVM grid : the cell $C_{ij} = [x_{i-1/2}; x_{i+1/2}] \times [y_{j-1/2}; y_{j+1/2}]$ is the cell associated to the quantity $\mathbf{x}_{i,j}$. We aim at computing the the flux crossing each inter-cell flux $F^n_{i+1/2,j}$, $F^n_{i-1/2,j}$, $G^n_{i,j+1/2}$ and $G^n_{i,j-1/2}$ in each direction east, west, south and north respectively.

Given the following averages for each cell $\mathcal{C}_{ij}$ at a given time t^n,

$$\mathbf{x}^n_{i,j} = \tfrac{1}{\Delta x \Delta y} \int_{\mathcal{C}_{ij}} \mathbf{x}(t^n, x, y) dx dy,$$
$$F^n_{i+1/2,j} = \tfrac{1}{\Delta t} \int_{t^n}^{t^{n+1}} F(\mathbf{x}(t^n, x_{i+1/2}, y_j)) dt, \quad F^n_{i-1/2,j} = \tfrac{1}{\Delta t} \int_{t^n}^{t^{n+1}} F(\mathbf{x}(t^n, x_{i-1/2}, y_j)) dt,$$
$$G^n_{i,j+1/2} = \tfrac{1}{\Delta t} \int_{t^n}^{t^{n+1}} G(\mathbf{x}(t^n, x_i, y_{j+1/2})) dt, \quad G^n_{i,j-1/2} = \tfrac{1}{\Delta t} \int_{t^n}^{t^{n+1}} G(\mathbf{x}(t^n, x_i, y_{j-1/2})) dt,$$

the finite volume formulation reads

$$\mathbf{x}^{n+1}_{i,j} = \mathbf{x}^n_{i,j} - \frac{\Delta t}{\Delta x} \left(F^n_{i+1/2,j} - F^n_{i-1/2,j} \right) - \frac{\Delta t}{\Delta y} \left(G^n_{i,j+1/2} - G^n_{i,j-1/2} \right). \tag{3.6}$$

Godunov (1959) proposed a conservative numerical scheme which assumes that the finite volume solution is piece-wise constant at each cell, as illustrated in figure 3.2. Under this hypothesis, we seek for an estimation of the inter-cell fluxes $F^n_{i+1/2,j}$, $F^n_{i-1/2,j}$, $G^n_{i,j+1/2}$ and $G^n_{i,j-1/2}$ by solving a local Riemann problem at each inter-cell.

1D Riemann problem

Given a time interval $[t^n; t^{n+1}]$, the 1D Riemann problem at a the inter-cell $(x_{i+1/2}, y_j)$ is formulated as the following initial value problem :

$$\begin{aligned} &\partial_t \mathbf{x} + A.\partial_x \mathbf{x} = 0, \\ &\mathbf{x}(t^n, x, y_j) = \begin{cases} \mathbf{x}_L := \mathbf{x}^n_{i,j} & \text{if } x < x_{i+1/2} \\ \mathbf{x}_R := \mathbf{x}^n_{i+1,j} & \text{if } x > x_{i+1/2} \end{cases}, \end{aligned} \tag{3.7}$$

where $A = \nabla F^n_{i+1/2,j}$ is the Jacobian matrix of $F^n_{i+1/2,j}$.

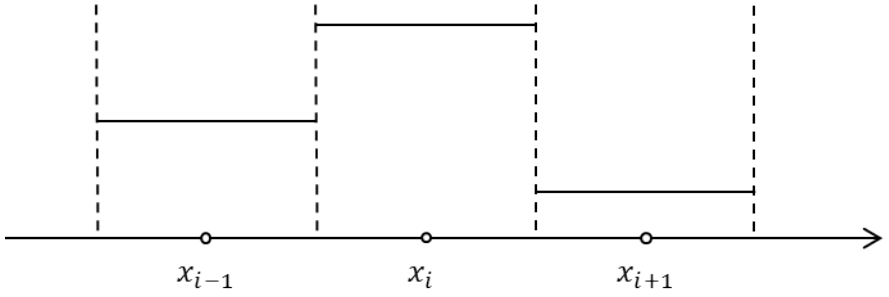

FIGURE 3.2 – 1D representation of the piecewise partition of the solution. The finite volume solution has a constant value within each cell $C_i = [x_{i-1/2}; x_{i+1/2}]$.

If the flux $F^n_{i+1/2,j}$ is **linear**, the solution is straightforward. We perform a diagonalization $A = R^{-1}\Lambda R$, where the eigenvalues $e_1 < \cdots < e_m$ correspond to the linearly independent eigenvectors $r_1, \cdots, r_m$ of the Jacobian A. Each wave $p \in \{1, \cdots, m\}$ carries a jump discontinuity with a velocity propagation e_p, also called characteristic speed. By introducing an auxiliary variable $W := R\mathbf{x}$, the problem (3.7) comes to a system of m scalar hyperbolic problems :

$$\partial_t W^{(p)}(t,x) + \Lambda \partial_x W^{(p)}(t,x) = 0, \text{ for } p = 1, \cdots, m. \tag{3.8}$$

The solution of (3.8) is obtained by introducing the following matrices

$$A^+ = R\Lambda^+ R^{-1}, \quad A^- = R\Lambda^- R^{-1}, \quad |A| = R|\Lambda|R^{-1},$$

where Λ^+ and Λ^- gather the positive and negative characteristic speed respectively, and $|\Lambda|$ contains the absolute value of the characteristic speed. Given these matrices, we apply the first-order upwind method to (3.8) for positive an negative characteristic speeds such that

$$\begin{aligned}(W^+)_i^{n+1} &= (W^+)_i^n - \Lambda^+ \tfrac{\Delta t}{\Delta x}((W^+)_i^n - (W^+)_{i-1}^n),\\ (W^-)_i^{n+1} &= (W^-)_i^n - \Lambda^- \tfrac{\Delta t}{\Delta x}((W^-)_{i+1}^n - (W^-)_i^n).\end{aligned}$$

As W is by the definition the sum of W^+ and W^-, the numerical solution of the hyperbolic equation (3.8) reads :

$$W_i^{n+1} = W_i^n - \Lambda \frac{\Delta t}{\Delta x}(W_i^n - W_{i-1}^n) + |\Lambda| \frac{\Delta t}{\Delta x} \frac{W_{i+1}^n - 2W_i^n + W_{i-1}^n}{2}.$$

Thus, we deduce the numerical solution of the 1D Riemann problem (3.7) for a linear flux F with respect to its Jacobian matrix A :

$$\mathbf{x}^{n+1} = \mathbf{x}_i^n - A \frac{\Delta t}{\Delta x}(\mathbf{x}_i^n - \mathbf{x}_{i-1}^n) + |A| \frac{\Delta t}{\Delta x} \frac{\mathbf{x}_{i+1}^n - 2\mathbf{x}_i^n + \mathbf{x}_{i-1}^n}{2}. \tag{3.9}$$

In the **non-linear** case, the literature provides several solvers that approximate the Riemann solution (Harten et al., 1983; Roe, 1981). The Roe solver, proposed by Roe (1981), is commonly chosen when it comes to solve numerically the SWE. This solver consists in linearizing the Jacobian matrix $A \approx \tilde{A}(\mathbf{x}_L, \mathbf{x}_R)$, called Roe matrix, which has to verify the following proprieties :

– $\tilde{A}(\mathbf{x}_L, \mathbf{x}_R)$ is assumed to be constant between the left and the right cells,
– $\tilde{A}(\mathbf{x}_L, \mathbf{x}_R)$ has to be diagonalizable with real eigenvalues $\{e_p\}_p$ associated to linearly independent eign vectors $\{r_p\}_p$. This propriety is consistent with the exact Jacobian and the conservation of $\tilde{A}(\mathbf{x}_R - \mathbf{x}_L) = F(\mathbf{x}_R) - F(\mathbf{x}_L)$.

Once the linearized Jacobian matrix is obtained, the solution of the 1D Riemann problem follows the linear case (3.9). The Roe matrix associated to the 2D SWE was proposed by Priestley (1987) and also Glaister (1988). We describe the construction of the Roe solver in the following.

Application of the Roe solver to the 2D SWE

In order to construct the Roe matrix, Priestley (1987) introduced the following parameter vectors :

$$\mathbf{w}_L := \begin{pmatrix} w_L^1 \\ w_L^2 \\ w_L^3 \end{pmatrix} = \begin{pmatrix} \sqrt{h_L} \\ u_L\sqrt{h_L} \\ v_L\sqrt{h_L} \end{pmatrix}, \mathbf{w}_R := \begin{pmatrix} w_R^1 \\ w_R^2 \\ w_R^3 \end{pmatrix} = \begin{pmatrix} \sqrt{h_R} \\ u_R\sqrt{h_R} \\ v_R\sqrt{h_R} \end{pmatrix}.$$

At each cell interface we compute the difference between the left flux and the right flux, denoted by the subscripts L an R respectively,

$$F(\mathbf{x}_L) - F(\mathbf{x}_R) = \tilde{A}(\mathbf{x}_L - \mathbf{x}_R) = \sum_{i=1}^{3} \alpha_k e_k r_k.$$

For each variable, we denote $\mathbf{\Delta w} := \mathbf{w}_L - \mathbf{w}_r$ and $\tilde{\mathbf{w}} := \frac{1}{2}(\mathbf{w}_L + \mathbf{w}_R)$. The eigenvalues of the linearized Jacobian $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$ are defined by

$$e_1 = \frac{\tilde{w}_2}{\tilde{w}_1} - \sqrt{\tilde{w}_1^2}, \quad e_2 = \frac{\tilde{w}_2}{\tilde{w}_1}, \quad e_3 = \frac{\tilde{w}_2}{\tilde{w}_1} + \sqrt{\tilde{w}_1^2},$$

and their corresponding eigenvectors are

$$\mathbf{r}_1 = \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 - \tilde{w}_1\sqrt{\tilde{w}_1^2} \\ \tilde{w}_3 \end{pmatrix}, \quad \mathbf{r}_2 = \begin{pmatrix} 0 \\ 0 \\ w_1 \end{pmatrix}, \quad \mathbf{r}_3 = \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 + \tilde{w}_1\sqrt{\tilde{w}_1^2} \\ \tilde{w}_3 \end{pmatrix},$$

with the coefficients

$$\alpha_1 = \Delta w_1 - \frac{\tilde{w}_1\Delta w_2 - \tilde{w}_2\Delta w_1}{2\tilde{w}_1\sqrt{\tilde{w}_1^2}}, \quad \alpha_2 = \frac{\tilde{w}_1\Delta w_2 - \tilde{w}_2\Delta w_1}{\tilde{w}_1}, \quad \alpha_3 = \Delta w_1 + \frac{\tilde{w}_1\Delta w_2 - \tilde{w}_2\Delta w_1}{2\tilde{w}_1\sqrt{\tilde{w}_1^2}}.$$

Hence, the fluxes along the x-direction are described by :

$$F_{i+1/2,j}^n = \frac{1}{2}(F(\mathbf{x}_L^n) + F(\mathbf{x}_R^n)) - \frac{1}{2}\sum_{k=1}^{3} \alpha_k |e_k| \mathbf{r}_k, \text{ for } \mathbf{x}_L = \mathbf{x}_{i,j}^n \text{ and } \mathbf{x}_R = \mathbf{x}_{i+1,j}^n, \tag{3.10}$$

$$F_{i-1/2,j}^n = \frac{1}{2}(F(\mathbf{x}_L^n) + F(\mathbf{x}_R^n)) - \frac{1}{2}\sum_{k=1}^{3} \alpha_k |e_k| \mathbf{r}_k, \text{ for } \mathbf{x}_L = \mathbf{x}_{i-1,j}^n \text{ and } \mathbf{x}_R = \mathbf{x}_{i,j}^n. \tag{3.11}$$

Similarly, in the y-direction, we compute the flux difference between the north flux and the south flux, denoted by the subscripts N an S respectively,

$$G(\mathbf{x}_S) - G(\mathbf{x}_N) = \tilde{A}(\mathbf{x}_N - \mathbf{x}_S) = \sum_{k=1}^{3} \beta_k e_k r_k.$$

For each variable we note $\mathbf{\Delta w} = \mathbf{w}_N - \mathbf{w}_S$ and $\tilde{\mathbf{w}} = \frac{1}{2}(\mathbf{w}_N + \mathbf{w}_S)$. The eigenvalues of the approximated Jacobian $\tilde{A}(\mathbf{x}_N, \mathbf{x}_S)$ are defined by

$$e_1 = \frac{\tilde{w}_3}{\tilde{w}_1} - \sqrt{\tilde{w}_1^2}, \quad e_2 = \frac{\tilde{w}_3}{\tilde{w}_1}, \quad e_3 = \frac{\tilde{w}_3}{\tilde{w}_1} + \sqrt{\tilde{w}_1^2},$$

and their corresponding eigenvectors are

$$\mathbf{r}_1 = \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 - \tilde{w}_1\sqrt{\tilde{w}_1^2} \end{pmatrix}, \quad \mathbf{r}_2 = \begin{pmatrix} 0 \\ 0 \\ w_1 \end{pmatrix}, \quad \mathbf{r}_3 = \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 + \tilde{w}_1\sqrt{\tilde{w}_1^2} \end{pmatrix},$$

with the coefficients

$$\alpha_1 = \Delta w_1 - \frac{\tilde{w}_1\Delta w_3 - \tilde{w}_3\Delta w_1}{2\tilde{w}_1\sqrt{\tilde{w}_1^2}}, \quad \alpha_2 = \frac{\tilde{w}_1\Delta w_3 - \tilde{w}_3\Delta w_1}{\tilde{w}_1}, \quad \alpha_3 = \Delta w_1 + \frac{\tilde{w}_1\Delta w_3 - \tilde{w}_3\Delta w_1}{2\tilde{w}_1\sqrt{\tilde{w}_1^2}}.$$

Finally, the fluxes along the y-direction are given by :

$$G_{i,j+1/2} = \frac{1}{2}(G(\mathbf{x}_L) + G(\mathbf{x}_R)) - \frac{1}{2}\sum_{k=1}^{3} \alpha_k |e_k| \mathbf{r}_k, \text{ for } \mathbf{x}_S = \mathbf{x}_{ij} \text{ and } \mathbf{x}_N = \mathbf{x}_{i,j+1}, \tag{3.12}$$

$$G_{i,j-1/2} = \frac{1}{2}(G(\mathbf{x}_L) + G(\mathbf{x}_R)) - \frac{1}{2}\sum_{k=1}^{3} \alpha_k |e_k| \mathbf{r}_k, \text{ for } \mathbf{x}_S = \mathbf{x}_{i,j-1} \text{ and } \mathbf{x}_N = \mathbf{x}_{ij}. \tag{3.13}$$

3.1.3 Boundary conditions

As we study a fluid contained in a non-porous rectangular tank (see § 3.4.2), we expect the fluid to return in the opposite direction whenever it hits the walls. This comes to impose a reflecting boundary condition in every direction. In this configuration, we impose the values of the height and velocity field at ghost cells $\{C_{1,j}, j = 1, \dots ny\}$, $\{C_{i,1}, i = 1, \dots nx\}$, $\{C_{nx,j}, j = 1, \dots ny\}$ and $\{C_{i,ny}, i = 1, \dots nx\}$ around the boundaries. The height and velocity field in these ghost cells are set to the same height as the boundaries and the same velocity in the opposite direction than the incoming wave respectively. The reflecting boundary conditions are illustrated in figure 3.3.

3.1.4 Time integration

We describe in the present section the formulation of the time integration scheme. The general formulation of Runge-Kutta techniques is given by :

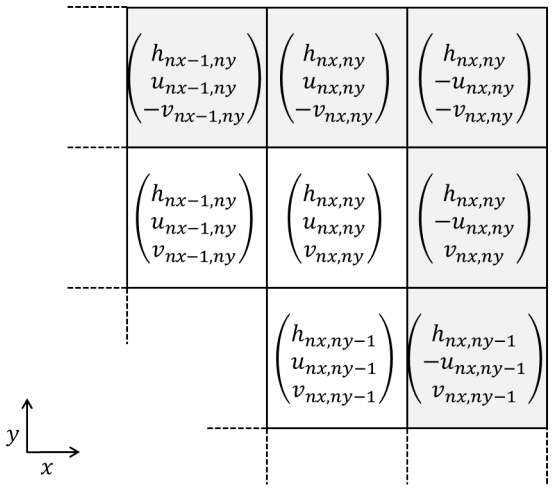

FIGURE 3.3 – Reflecting boundary conditions : we set the value of the flow at ghost cells (in gray) with respect to the values of the height and velocity fields at the boundary cells.

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \sum_{i=1}^{r} b_i k_i,$$

$$k_i = f(t^n + \Delta t c_i, \mathbf{x}_i^n + \Delta t \sum_{j=1}^{r} a_{ij} k_i), i = 1, ..., r$$

where the function f computes the fluxes given a state variable $\mathbf{x}^n$ such that $f(\mathbf{x}^n) = F^n_{i,j+1/2} - F^n_{i,j-1/2} + G^n_{i,j+1/2} - G^n_{i,j-1/2}$.

The 4th order Runge-Kutta method (RK4) is usually privileged in many CFD applications for its robustness and convergence proprieties. The 3rd order method (RK3) also provides a good compromise between accuracy and computation time. Both techniques were implemented in the SWE code and performed very similar results. Thus, we decided to consider the RK3 method for the implementation of the 4DVar code. The RK3 method is defined by fixing $r = 3$ and the coefficients $(c_1, c_2, c_3) = (0, 0.5, 1)$, $(b_1, b_2, b_3) = (1/6, 2/3, 1/6)$ and $(a_{ij})_{1 \leq i,j \leq 3} = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ -1 & 2 & 0 \end{pmatrix}$.

3.1.5 Implementation into MATLAB

We created the `SWE` code based on the dynamical model described previously. The code computes the trajectory of the non-conservative variable state on a given assimilation window $[0; N\Delta t]$ as described in algorithm 8. The auxiliary routines `u2q` and `q2u` perform a simple transformation from a given non conservative state variables $\mathbf{x} = (h, u, v)^T$ to the conservative variables $\mathbf{x}_c := (h, q_x, q_y)^T = (h, hu, hv)^T$ and vice versa. The `RK3` routine performs the time integration procedure which deduces the conservative state $\mathbf{x}_c^{n+1}$ at time $t^{n+1} = (n+1)\Delta t$ from the previous state

$\mathbf{x}_c^n$ at time $t^n = n\Delta t$ (see§ 3.1.4). The latter performs several calls of the boundary conditions routine `BC` defined by § 3.1.3, and the `FVM` routine which computes the fluxes at each inter-cells (see § 3.1.2).

Algorithm 8 `SWE`(h^0,u^0,v^0)

Initialization: $n = 0$, h^0, u^0, v^0
  while $n < N$ **do**
    call `u2q`$(h^n,u^n,v^n,q_x^n,q_y^n)$
    call `RK3`$(h^n,q_x^n,q_y^n,h^{n+1},q_x^{n+1},q_y^{n+1})$
    call `q2u`$(h^{n+1},u^{n+1},v^{n+1},q_x^{n+1},q_y^{n+1})$
    save $(h^{n+1},u^{n+1},v^{n+1})$
    $n = n + 1$
  end while
  return h^N, u^N, v^N

3.2 Ensemble-based 4DVar technique

We compare in this chapter the performance of a purely variational data assimilation technique (see § 2.4.7) and an enhanced version of an ensemble based technique (Yang et al., 2015). The latter is based on the work of Liu et al. (2008) and Liu et al. (2009), and is referred to as the 4DEnVar in the following. This technique is specifically defined within the framework of preconditioned incremental variational formulation (see § 3.2.1) while handling an empirical approximation of the background covariance matrix. The low rank approximation of the background covariance matrix is directly inspired from the Ensemble Kalman filter techniques where the covariance terms are empirically estimated from an ensemble of realizations.

3.2.1 Preconditioning of the incremental 4DVar assimilation

We recall the incremental variational assimilation formalism defined by the assimilation system

$$\partial_t \delta\mathbf{x}(t,x) + \partial_{\mathbf{x}}\mathbb{M}\delta\mathbf{x}(t,x) = 0, \tag{3.14}$$

$$\delta\mathbf{x}(t_0,x) = \mathbf{x}_0^b - \mathbf{x}_0 + \boldsymbol{\eta}, \tag{3.15}$$

$$\boldsymbol{\mathcal{Y}}(t,x) = \mathbb{H}(\varphi_t(\mathbf{x}_0(x)) + \boldsymbol{\varepsilon}(t,x), \tag{3.16}$$

where $\partial_{\mathbf{x}}\mathbb{M}(\mathbf{x})$ denotes the tangent linear operator of the dynamics operator $\mathbb{M}$. The variable $\mathbf{x}_0^b$ denotes an arbitrary background condition, $\mathbf{x}_0$ is the initial guess and $\boldsymbol{\eta}$ is some error of background covariance $\mathbf{B}$. The quantity

$$\varphi_t(\mathbf{x}_0(x)) = \mathbf{x}_t = \mathbf{x}_0 + \int_{t_0}^{t} \mathbb{M}(\mathbf{x}_s)ds,$$

denotes the flow map. The state variables $\mathbf{x}_t$ and the measurements $\boldsymbol{\mathcal{Y}}$ are linked by an observation operator $\mathbb{H}$, up to an observation error $\boldsymbol{\varepsilon}$ that is assumed to be

a zero mean i.i.d. (independent and identically distributed)Gaussian random field with covariance tensor R. In this study, for simplification purpose, we will consider a linear observation operator set to the identity or to an incomplete identity operator when only a part of the state is observable. The optimal increment $\delta\mathbf{x}_0$ at time t_0 is obtained by minimizing the following cost function :

$$J(\delta\mathbf{x}_0) = \frac{1}{2}\|\delta\mathbf{x}_0^b - \delta\mathbf{x}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\partial_{\mathbf{x}}\mathbb{H}\delta\mathbf{x}(t,x) - D(t,x)\|^2_{\mathbf{R}^{-1}}dt, \tag{3.17}$$

where

$$\mathbf{x}_t = \partial_{\mathbf{x}}\varphi_t(\mathbf{x}_t^b)\delta\mathbf{x}_0,$$

and the innovation vector $D(t,x)$ is defined by

$$D(t,x) = \boldsymbol{\mathcal{Y}}(t,x) - \mathbb{H}(\varphi_t(\mathbf{x}_t^b)).$$

In order to minimize the objective function $J(\delta\mathbf{x}_0)$ with respect to $\delta\mathbf{x}_0$ we have to cancel the gradient $\nabla J(\delta\mathbf{x}_0)$. This comes to solve the following linear system :

$$\nabla J(\delta\mathbf{x}_0) = \mathbf{B}^{-1}\delta\mathbf{x}_0 + \int_{t_0}^{t_f} \partial_{\mathbf{x}}\varphi_t^*(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}(\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t\delta\mathbf{x}_0 - D(t))dt = 0, \tag{3.18}$$

$$\Leftrightarrow \left(\mathbf{B}^{-1} + \int_{t_0}^{t_f} \partial_{\mathbf{x}}\varphi_t^*(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t dt\right)\delta\mathbf{x}_0 = \int_{t_0}^{t_f} \partial_{\mathbf{x}}\varphi_t^*(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}D(t)dt, \tag{3.19}$$

which can be compactly written as $\mathcal{H}\delta\mathbf{x}_0 = b$, denoting the left hand operator as the Hessian matrix

$$\mathcal{H} = \mathbf{B}^{-1} + \int_{t_0}^{t_f} (\partial_{\mathbf{x}}\varphi_t)^*(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t dt, \tag{3.20}$$

and the right hand as the vector

$$b = \int_{t_0}^{t_f} (\partial_{\mathbf{x}}\varphi_t)^*(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}D(t,x)dt. \tag{3.21}$$

The possibly ill-conditioned nature of (3.19) depends on the condition number of the Hessian matrix $\mathcal{H}$ (3.20). The condition number is defined by the ratio between the maximal eigenvalue over the minimal eigenvalue of the matrix of interest. The larger the condition number, the more sensitive the system is to errors both in the b vector and in the estimate. A poorly-conditioned variational incremental system can be improved by applying a change of variable with the matrix square-root of the background error covariance matrix $\mathbf{B}$. This change of variable is commonly achieved by the so-called control variable transform (CVT),

$$\delta\mathbf{x}_t = \mathbf{B}^{\frac{1}{2}}\delta\mathbf{z}_t, \tag{3.22}$$

and leads to a new Hessian matrix $\tilde{H} = \mathbf{B}^{\frac{1}{2}}H\mathbf{B}^{\frac{1}{2}}$ possessing a lower condition number (Haben et al., 2011; Yang, 2014).

Thus, we obtain a modified cost function by injecting the CVT (3.22) into the objective function (3.17) such as

$$J(\delta\mathbf{z}_0) = \frac{1}{2}\|\delta\mathbf{z}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t(\mathbf{x}_0)\mathbf{B}^{\frac{1}{2}}\delta\mathbf{z}_0 - D(t,x)\|^2_{\mathbf{R}^{-1}}dt, \tag{3.23}$$

3.2.2 Low rank approximation of the background error covariance matrix

The En4DVar assimilation is designed on such preconditioned incremental variational system and introduces an empirical approximation of the background covariance matrix B. This low rank approximation of the background covariance matrix is directly inspired from the Ensemble Kalman filter where the covariance terms are estimated from the spread of an ensemble of samples.

Denoting $< f(t) >= \frac{1}{N}\sum_{i=1}^{N} f^{(i)}(t)$ as an empirical ensemble mean of a quantity $f(t)$ through N samples, the unbiased empirical background covariance matrix reads

$$\mathbf{B} \approx \frac{1}{N-1}\sum_{i=1}^{N}\left(\mathbf{x}^{(i),b} - < \mathbf{x}^b >\right)\left(\mathbf{x}^{(i),b} - < \mathbf{x}^b >\right)^T. \tag{3.24}$$

Noting $\mathbf{A}'_b := \frac{1}{\sqrt{N-1}}(\mathbf{x}_0^{(1),b} - < \mathbf{x}_0^b >, \cdots, \mathbf{x}_0^{(N),b} - < \mathbf{x}_0^b >)$, the perturbation matrix gathering the N zero mean centered background ensemble members as a low-dimensional approximation of the background matrix. Introducing the background covariance approximation in the preconditioned cost function (3.23), we get

$$J(\delta\mathbf{z}_0) = \frac{1}{2}\|\delta\mathbf{z}_0\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\int_{t_0}^{t_f} \|\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t(\mathbf{x}_0)\mathbf{A}'_b\delta\mathbf{z}_0 - D(t,x)\|^2_{\mathbf{R}^{-1}}dt, \tag{3.25}$$

we define the square root of the background error covariance matrix as the perturbation matrix :

$$\mathbf{B}^{\frac{1}{2}} \approx \mathbf{A}'_b := \frac{1}{\sqrt{N-1}}(\mathbf{x}_0^{(1),b} - < \mathbf{x}_0^b >, \cdots, (\mathbf{x}_0^{(N),b} - < \mathbf{x}_0^b >)).$$

Let us note $\tilde{\mathbf{B}}^{\frac{1}{2}} := \partial_{\mathbf{x}}\varphi_t(\mathbf{x}_0)\mathbf{A}'_b$ the propagation of ensemble perturbation matrix $\mathbf{A}'_b$ which corresponds to a forecast by the dynamical model of the centered square-root background covariance matrix. As we rely here on an empirical description of this matrix from a set of samples, we can observe that integrating these samples in time provides us immediately an empirical expression of a low-rank approximation of the background covariance trajectory and of its square root. This avoids thus the employment of the adjoint operator.

The gradient of the cost function is now given by

$$\nabla J(\delta\mathbf{z}_0) = \delta\mathbf{z}_0 + \int_{t_0}^{t_f}(\tilde{\mathbf{B}}^{\frac{1}{2}T}(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}(\partial_{\mathbf{x}}\mathbb{H}\tilde{\mathbf{B}}^{\frac{1}{2}}\delta\mathbf{z}_0 - D(t,x))dt. \tag{3.26}$$

and its Hessian reads

$$\tilde{\mathcal{H}} = \mathbb{I} + \int_{t_0}^{t_f} \tilde{\mathbf{B}}^{\frac{1}{2}T}(\partial_{\mathbf{x}}\mathbb{H})^*\mathbf{R}^{-1}\partial_{\mathbf{x}}\mathbb{H}\tilde{\mathbf{B}}^{\frac{1}{2}}dt. \tag{3.27}$$

Once the minimizer $\widehat{\delta\mathbf{z}_0}$ estimated, the analysis reads

$$\mathbf{x}_0^a = \mathbf{x}_0^b + \mathbf{A}'_b\widehat{\delta\mathbf{z}_0}. \tag{3.28}$$

Let us emphasize that, as the empirical covariance matrix $\mathbf{B}$ is at most of rank $N-1$, the control variable has at most $N-1$ non null components in the eigenspace. Compared to the full 4DVar approach, the control variable's degrees of freedom are thus considerably lowered and the minimization computational complexity is significantly decreased. Indeed, this ensemble version has a lower computation cost if the ensemble forecasting step is distributed on a grid computing.

3.2.3 Preconditioning matrix update

As mentioned earlier, in this study we focus on situations where the background state is only poorly known. It is hence essential to allow in the estimation process a substantial deviation from the background state. So unlike typical incremental ensemble-based variational methods which keep a fixed background covariance and apply a single outer loop of the Gauss-Newton minimization, we propose to update the approximation of this associated error covariance between two consecutive outer loops. The update of the error covariance can be either derived from the ensemble of analysis based on perturbed observations or by a direct transformation of the background ensemble perturbations. The first method relies on a perturbed ensemble of observations, generated with an additional noise with the same standard deviation than in (3.16) :

$$\boldsymbol{\mathcal{Y}}^j = \boldsymbol{\mathcal{Y}} + \epsilon^j, j = 1, ..., N. \tag{3.29}$$

At the kth outer loop iteration, the innovation vector of the member of the initial ensemble $\mathbf{x}_0^{(j),k}$ is defined as,

$$D^{(j),k}(t,x) = \boldsymbol{\mathcal{Y}}^j(t,x) - \mathbb{H}(\varphi_t(\mathbf{x}_0^{(j),k})), j = 1, ..., N, \tag{3.30}$$

with $\mathbf{x}_0^{(j),0} = \mathbf{x}_0^{(j),b}$. Thus a parallel realization of minimization with regard to each member of initial ensemble is conducted

$$\delta\mathbf{z}_0^{(j),k} = (A_b'^k)^{-1}\delta\mathbf{x}_0^{(j),k}, j = 1, ..., N. \tag{3.31}$$

$$J(\delta\mathbf{z}_0^{(j),k}) = \frac{1}{2}\|\delta\mathbf{z}_0^{(j),k}\|^2 + \frac{1}{2}\int_{t_0}^{t_f} \|\partial_{\mathbf{x}}\mathbb{H}\partial_{\mathbf{x}}\varphi_t(\mathbf{x}_0)A_b'^k\delta\mathbf{z}_0^{(j),k} - D^{(j),k}(t,x)\|^2_{R^{-1}}dt. \tag{3.32}$$

Finally the updated initial ensemble field and its perturbation matrix read :

$$\mathbf{x}_0^{(j),k+1} = \mathbf{x}_0^{(j),k} + A_b'^k\widehat{\delta\mathbf{z}_0}^{(j),k}, j = 1, ..., N. \tag{3.33}$$

$$\mathbf{A}_b'^{(k+1)} = \frac{1}{\sqrt{N-1}}\left(\mathbf{x}_0^{(1),k+1} - < \mathbf{x}_0^{k+1} >, ..., \mathbf{x}_0^{(N),k+1} - < \mathbf{x}_0^{k+1} >\right). \tag{3.34}$$

The direct transformation approach corresponds to a linear transformation of the initial error's $\mathbf{x}_b - \mathbf{x}_t$ covariance. This approach can take many forms as the transformation matrix is not unique, here we opt for a mean preserving transformation

as used in Ensemble Transform Kalman filter, the updated background ensemble perturbations reads,

$$\mathbf{A}_b^{'(k+1)} = \mathbf{A}_b^{'k} \left(\mathbb{I} + \int_{t_0}^{t_f} \tilde{\mathbf{B}}_t^{\frac{1}{2}T} (\partial_{\mathbf{x}} \mathbb{H})^* \mathbf{R}^{-1} \partial_{\mathbf{x}} \mathbb{H} \tilde{\mathbf{B}}^{\frac{1}{2}} dt \right)^{\frac{1}{2},k} \mathbf{V}. \tag{3.35}$$

It corresponds to the Hessian square root computed from previous perturbation matrix at outer loop iteration k. As the minimization algorithm LBFGS relies on an approximation of the inverse Hessian matrix $\mathcal{H}^{-1}$, we can use this byproduct to evaluate (3.35). At the initial time, the background matrix is fixed from the initial random conditions chosen. The arbitrary orthogonal matrix $\mathbf{V}$ is used to center the posterior ensemble on the updated initial condition/analysis. In this approach a single minimization process is conducted with respect to the background state in opposition to previous cases where the minimization has to be done with respect to each member of the ensemble plus the background state. Finally the updated initial ensemble fields are,

$$\mathbf{x}_0^{(j),k+1} = \hat{\mathbf{x}}_0^k + \sqrt{N-1} \mathbf{A}_b^{'(k+1)}, j = 1, ..., N, \tag{3.36}$$

where $\hat{\mathbf{x}}_0^k$ corresponds to the updated initial state at the kth outer loop,

$$\hat{\mathbf{x}}_0^k = \mathbf{x}_0^k + \left(\mathbf{A}_b^{'k} \right)^{-1} \widehat{\delta z_0}^k. \tag{3.37}$$

Both variants of the update will be assessed in the experimental § 3.4.2.

3.2.4 Localization issues

The previous ensemble method relies on a low rank approximation of the background matrix. This empirical approximation built from only very few samples, compared to the state space dimension, leads in practice to spurious correlations between distant points. For ensemble Kalman techniques, it is customary to remove these long distance correlations through the so-called localization procedure. There are generally two methods to filter the pseudo-correlations.

The first approach introduces a Schur element-wise product between the background correlation matrix and a local isotropic correlation function $P_b = C \odot B$. The spatial correlation function can be simply defined as a matrix $C(\|x-y\|/L)$ in which we set $C_{xy} = 0$ when the distance between x and y exceeds the cutoff distance L. Polynomial approximations of a Gaussian function with compact support and a hard cutoff are often employed to that end (Gaspari and Cohn, 1999). They lead to sparse correlation matrices, which is computationally advantageous. In order to incorporate the localized background error matrix into our system, we approximate the square root of P_b by a spectral decomposition of the isotropic correlation function and keep only the r first leading Fourier modes

$$C = E_{n \times r} \lambda_{r \times r}^{1/2}. \tag{3.38}$$

The modified perturbation matrix is then provided by

$$\mathbf{P}_b' = (diag(A_b^{'(1)})C, ..., diag(A_b^{'(N)})C). \tag{3.39}$$

Here the *diag* operator sets the vector $\mathbf{x}'_{b,k}$ as the diagonal of a matrix. This localized perturbation matrix is used to precondition the assimilation system associated with (3.25). Remark that this approach is incompatible with the direct transform update of the background error covariance matrix. This is due to the inconsistency of matrix dimensions when updating the background ensemble based on (3.36). As the dimension of $\mathbf{P}'_b$ is $n \times N \times r$ instead of $n \times N$, the ensemble perturbation matrix cannot be recovered from its localized counterpart.

A variant of covariance localization approach based on Lorenc (2003) is implemented in Desroziers et al. (2014) along with other localization schemes. Desroziers et al. (2014) also highlights the importance of the dimension of the control vector which is directly related to the cost of the minimization algorithm.

Another localization technique proposed by Hunt et al. (2004) employs local ensemble. This approach involves a transformation $\mathbb{M}_l$ from state space $\mathbb{R}^n$ to local space $\mathbb{R}^l$, the local vector is defined as :

$$\mathbf{x}_l = \mathbf{M}_l \mathbf{x}_n. \tag{3.40}$$

Then the analysis process is done in local space $\mathbb{R}^n$ around each grid point only incorporating the model points and observations within a certain range. This certain range, denoted as l, which corresponds to the concept of cut-off distance aforementioned, determines the size of the local space. This localization strategy is ideally compatible with the method of direct transformation approach associated with the update of background error covariance matrix. The great advantage of this combination is its low computational cost when implemented with properly parallelized minimization procedures. However, good performances can only be reached with a small local space.

All these elements (i.e. CVT, localization and incremental/background error covariance matrix update) associated with a LBFGS minimization strategy constitute the proposed ensemble method. The algorithm descriptions of the overall methods are presented in Figs. 2 and 3. We point out that this assimilation system - composed of perturbed observations, one outer loop and localization via a modified covariance matrix - is equivalent to the 4DEnVar method proposed by Liu et al. (2008) and Liu et al. (2009). If direct ensemble transformation update is used, it is hence close to the 4D-LETKF method Fertig et al. (2007) and the IEnKS method Bocquet and Sakov (2014) but with a minimization performed on a variational basis.

3.3 Construction of the SWE discrete tangent and adjoint

The construction of the discrete adjoint corresponding to the SWE model described in algorithm 8 is the cornerstone of the 4DVar implementation. While the choice of constructing the discrete adjoint over the analytic adjoint in this application can be debated, our goal is to explore the implementation of a variational technique in preparation to the 3D reconstruction (see § 4.2). In this application, we need to construct the most accurate as possible adjoint represenetation of the discrete SWE dynamics (i.e. the finite volume scheme with the Roe approximation).

Algorithm 9 Ensemble-based variational data assimilation algorithm : Localize covariance approach

Set an initial state $\mathbf{x}(t_0) = \mathbf{x}^b(t_0)$ and the ensemble $\mathbf{x}_0^{j,b}$ as an arbitrary choice (for the 1st cycle) or as the forecast state an the ensemble forecast derived from the previous assimilation cycle respectively

loop

Define the matrix $\mathbf{X}(t) = [\mathbf{x}(t_0), \mathbf{x}_0^{1,b}, ..., \mathbf{x}_0^{N,b}]$ concatenating the initial ensemble and compute in parallel $\mathbf{X}(t)$ with the forward integration of the nonlinear dynamics (3.14)

Generate ensemble observations (3.29) and innovations (3.30)

Derive the background perturbation matrix localization technique from (3.39)

Initialize the increment matrix $\delta\mathbf{X}_0 = [\delta\mathbf{x}(t_0), \delta\mathbf{x}_0^{1,b}, ..., \delta\mathbf{x}_0^{N,b}]$

Do an inverse control variable transformation $\delta\mathbf{R}_0 = (\mathbf{P}_b')^{-1}\delta\mathbf{X}_0$ if necessary where $(\mathbf{P}_b')^{-1}$ is calculated by SVD

Optimize in parallel $\delta\mathbf{R}_0$ in the inner loop, the cost function and the gradient are calculated based on the modified versions of (3.23)

Update control initial space $\mathbf{R}_0$ and calculate $\delta_0\mathbf{X}_0$ by transforming $\delta\mathbf{R}_0$ to the state space with (3.22)

Update the initial ensemble (3.33) and the ensemble perturbation matrix (3.34)

end loop

Evolve the analysis state $\mathbf{x}^a(t_0)$ to the beginning of the next cycle through the nonlinear dynamics (3.14). The forecast state and the forecast enesemble are used to initialize the next assimilation cycle.

Algorithm 10 Ensemble-based variational data assimilation algorithm : Localize ensemble approach

Set an initial state $\mathbf{x}(t_0) = \mathbf{x}^b(t_0)$ and the ensemble $\mathbf{x}_0^b$ as an arbitrary choice (for the 1st cycle) or as the forecast state an the ensemble forecast derived from the previous assimilation cycle respectively

loop

Define the matrix $\mathbf{X}(t) = [\mathbf{x}(t_0), \mathbf{x}_0^{1,b}, ..., \mathbf{x}_0^{N,b}]$ concatenating the initial ensemble and compute in parallel $\mathbf{X}(t)$ with the forward integration of the nonlinear dynamics (3.14)

Define local space and the transformation from state space to local space (3.40)

Parallelizing minimization computation at each grid point (p, q)

Initialize the increment vector

$$\delta\mathbf{x}_{0,l} = M_{l,s}\delta\mathbf{x}_0$$

Derive the background ensemble perturbation covariance

$$\mathbf{A}'_{b,l} = \frac{1}{\sqrt{N-1}}\left(\mathbf{x}_{0,l}^{(1),b} - < \mathbf{x}_{0,l}^b >, ..., \mathbf{x}_{0,l}^{(N),b} - < \mathbf{x}_{0,l}^b >\right)$$

Apply an inverse control variable transformation in the local space

$$\mathbf{z}_{0,l} = (A'_{b,l})^{-1}\delta\mathbf{x}_{0,l}$$

The cost function minimization is done only once in terms of the background. The updated background is obtained by (3.37)

Update the initial ensemble and ensemble perturbation matrix by (3.36) and (3.35)

Reconstruct the analysis and the ensemble form local space to state space

end loop

Evolve the analysis state $\mathbf{x}^a(t_0)$ to the beginning of the next cycle through the nonlinear dynamics (3.14). The forecast state and the forecast enesemble are used to initialize the next assimilation cycle.

To that end, we build the discrete adjoint procedure associated to the `SWE` code with the help of the AD tool TAPENADE (Hascoët, 2004). We note that we had to perform a conversion of the `SWE` code from MATLAB to FORTRAN 90 in order to implement the AD tool TAPENADE. The reader can find the description of the AD principles and the different implementation strategies in appendix C.1. We also provide in appendix C.1.2 an example of the detailed construction of the tangent and adjoint procedure corresponding to a simple program.

In terms of the notations, we inherit from the formalism imposed by TAPENADE such that :

- the tangent procedures are ended by the subscript `_d`, the differential tangent variables associated to a given variable $\mathbf{x}$ are denoted $\dot{\mathbf{x}}$,
- the adjoint procedures are ended by the subscript `_b`, the differential adjoint variables associated to a given variable $\mathbf{x}$ are denoted $\bar{\mathbf{x}}$.

We describe in § 3.3.1, § 3.3.2 and § 3.3.3 the construction of the adjoint procedure of the boundary conditions, the Roe solver and the time discretization respectively. We deduce the discrete adjoint of the SWE code in § 3.3.4. The tangent and adjoint procedures of each subroutine and the entire code are systematically verified by the validation procedures presented in § 3.3.5.

3.3.1 Adjoint of the boundary conditions

The imposed reflecting boundary conditions consist in adding ghosts cells in every directions, and defining the value of non-conservative variable state according to figure 3.3. In the perspective of the computation of the fluxes F_E, F_W, G_N and G_S in the east, west, north and south directions with respect to a given cell $\mathcal{C}_{ij}$ respectively, the `BC` routine generates the auxiliary variables $\mathbf{x}_E$, $\mathbf{x}_W$, $\mathbf{x}_N$ and $\mathbf{x}_S$ as described in the left side of figure 3.4. The values at the ghost cells for each variables $\mathbf{x}_E$, $\mathbf{x}_W$, $\mathbf{x}_N$ and $\mathbf{x}_S$ are determined according to figure 3.3.

The adjoint routine `BC_b` corresponding to the `BC` routine outputs the adjoint variable $\bar{\mathbf{x}}$ from each adjoint $\bar{\mathbf{x}}_E$, $\bar{\mathbf{x}}_W$, $\bar{\mathbf{x}}_N$ and $\bar{\mathbf{x}}_S$ which are the output of the adjoint `flux\_b` routine (see § 3.3.2). The adjoint variable $\bar{\mathbf{x}}$ is the sum of the values at the ghost cells with the values at the boundaries of the adjoint variable state $\bar{\mathbf{x}}$ for each adjoint $\bar{\mathbf{x}}_E$, $\bar{\mathbf{x}}_W$, $\bar{\mathbf{x}}_N$ and $\bar{\mathbf{x}}_S$. The reader can find in appendix D.1 the complete adjoint routine of the boundary conditions.

3.3.2 Adjoint of the flux terms

Adjoint of the Roe solver

As described in § 3.1.2, we solve a 1D Riemann problem with a Roe solver to determine the flux crossing every inter-cell in a given direction (see § 3.1.2). In practice, the Roe solver computes the flux crossing from the left to the right cell from the input $\mathbf{x}_L$ and $\mathbf{x}_R$. The values of $\mathbf{x}_L$ and $\mathbf{x}_R$ depend on the direction of the flux we aim at computing. We assume beforehand that we computed the quantities $\mathbf{x}_W$, $\mathbf{x}_E$, $\mathbf{x}_N$ and $\mathbf{x}_S$ by the `BC` routine. The horizontal fluxes $F_W := F_{i-1/2,j}$ and $F_E := F_{i+1/2,j}$ require the couple $(\mathbf{x}_W, \mathbf{x})$ and $(\mathbf{x}, \mathbf{x}_E)$ respectively. In the same way,

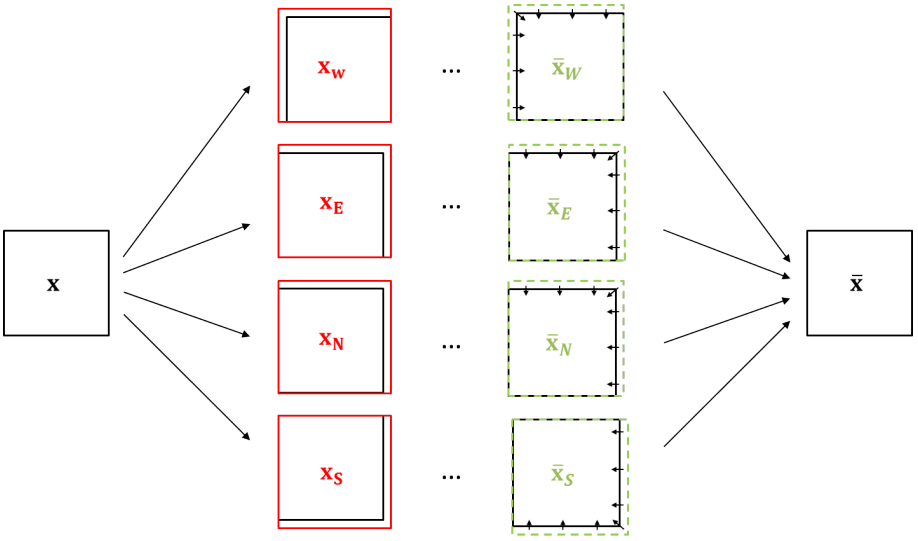

FIGURE 3.4 – Left side : `BC` creates ghost cells from the input state $\mathbf{x}$ (black box) to $\mathbf{x}_E$, $\mathbf{x}_W$, $\mathbf{x}_N$ and $\mathbf{x}_S$ (red box) which are used to compute the horizontal flux F_E, F_W, G_N and G_S and $G_{i+1/2,j}$ respectively. Right side : `BC_b` sums the values at the ghost cells with the values at the boundaries of each adjoint variable state $\bar{\mathbf{x}}$ (black box) for each adjoint $\bar{\mathbf{x}}_E$, $\bar{\mathbf{x}}_W$, $\bar{\mathbf{x}}_N$ and $\bar{\mathbf{x}}_S$ (green box).

the vertical flux $F_S := G_{i,j-1/2}$ and $F_N := G_{i,j+1/2}$ require the couple $(\mathbf{x}_S, \mathbf{x})$ and $(\mathbf{x}, \mathbf{x}_N)$ respectively.

At first, the Roe solver calls the `Priestley` routine which computes the parameter variables $\mathbf{w}_L$ and $\mathbf{w}_R$ from $\mathbf{x}_L$ and $\mathbf{x}_R$. The parameters $\mathbf{w}_L$ and $\mathbf{w}_R$ are then used to compute the eigenvalues, eigenvectors and the coefficient of a linearized Jacobian $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$. These quantities are computed by the `eigenvalue`, `eigenvector` and `coefficient` routines respectively. Finally, the latter are combined if the nonlinear flux $F(\mathbf{x}_L)$ and $F(\mathbf{x}_R)$ or $G(\mathbf{x}_L)$ and $G(\mathbf{x}_R)$ to deduce flux of interest in the given direction. We recall that the nonlinear flux of a given state $\mathbf{x}$ in the horizontal and vertical direction is given by $F(\mathbf{x}) = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}$ and $G(\mathbf{x}) = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}$ respectively. The latter is discretized in the algorithm `fluxH` and `fluxV` respectively.

The adjoint procedure corresponding to the auxiliary routines `Priestley`, `eigenvalue`, `eigenvector`, `coefficient`, `fluxH` and `fluxV` are systematically built with the TAPENDADE tool and are detailed in appendix § D.2. We assume that we determined these discrete adjoint and deduce the discrete adjoint `Roe_b` associated to `Roe` in algorithm 11.

Adjoint of the FVM

The `FVM` routine combines the fluxes terms F_E, F_W, G_S and G_N in each direction according to the FVM formulation (3.6). Each flux term was computed beforehand by the Roe solver described previously. The original `FVM` routine is described in

Algorithm 11 `Roe_b`($\bar{F}$,h_L,u_L,v_L,h_R,u_R,v_R,$flag$)

! Forward step ==========>
! Preprocessing
$\mathbf{w}_L$ = `Priestley` (h_L, u_L, v_L,$flag$)
$\mathbf{w}_R$ = `Priestley` (h_R, u_R, v_R,$flag$)
$\Delta\mathbf{w} = \mathbf{w}_L - \mathbf{w}_R$
$\tilde{\mathbf{w}} = \frac{1}{2}(\mathbf{w}_L + \mathbf{w}_R)$
! Characteristics of the Roe matrix $\tilde{A}(X_R, X_L)$
[e_1, e_2, e_3] = `eigenvalues`($\tilde{\mathbf{w}}$,$flag$)
[r_1, r_2, r_3] = `eigenvectors`($\tilde{\mathbf{w}}$,$flag$)
[α_1, α_2, α_3] = `coefficients`($\tilde{\mathbf{w}}$,$\Delta\mathbf{w}$,$flag$)
! Backward step <==========
! Compute the left and right fluxes
$\bar{F}_L = \bar{F}_L + \frac{1}{2}\bar{F}$
$\bar{F}_R = \bar{F}_R + \frac{1}{2}\bar{F}$
if $flag$ **then**
  [$\bar{h}_L$, $\bar{u}_L$, $\bar{v}_L$] = `fluxH_b` ($\bar{F}_L$,h_L, u_L, v_L)
  [$\bar{h}_R$, $\bar{u}_R$, $\bar{v}_R$] = `fluxH_b` ($\bar{F}_R$,h_R, u_R, v_R)
else
  [$\bar{h}_L$, $\bar{u}_L$, $\bar{v}_L$] = `fluxV_b` ($\bar{F}_L$,h_L, u_L, v_L)
  [$\bar{h}_R$, $\bar{u}_R$, $\bar{v}_R$] = `fluxV_b` ($\bar{F}_R$,h_R, u_R, v_R)
end if
[$\overline{\tilde{\mathbf{w}}}$, $\overline{\Delta\mathbf{w}}$] = `coefficients_b`($\bar{\alpha}1$, $\bar{\alpha}_2$, $\bar{\alpha}_3$,$\tilde{\mathbf{w}}$,$\Delta\mathbf{w}$,$flag$)
$\overline{\tilde{\mathbf{w}}}$ = `eigenvectors_b`($\bar{r}_1$, $\bar{r}_2$, $\bar{r}_3$, $\tilde{\mathbf{w}}$)
$\overline{\tilde{\mathbf{w}}}$ = `eigenvalues_b`($\bar{e}_1$, $\bar{e}_2$, $\bar{e}_3$,$\tilde{\mathbf{w}}$)
$\bar{\mathbf{w}}_L = \bar{\mathbf{w}}_L + \frac{1}{2}\overline{\tilde{\mathbf{w}}}$
$\bar{\mathbf{w}}_R = \bar{\mathbf{w}}_R + \frac{1}{2}\overline{\tilde{\mathbf{w}}}$
$\bar{\mathbf{w}}_L = \bar{\mathbf{w}}_L + \overline{\Delta\mathbf{w}}$
$\bar{\mathbf{w}}_R = \bar{\mathbf{w}}_R - \overline{\Delta\mathbf{w}}$
[$\bar{h}_R$, $\bar{u}_R$, $\bar{v}_R$] = `Priestley_b` ($\bar{\mathbf{w}}_R$,h_R, u_R, v_R)
[$\bar{h}_L$, $\bar{u}_L$, $\bar{v}_L$] = `Priestley_b` ($\bar{\mathbf{w}}_L$,h_L, u_L, v_L)
return $\bar{h}_L$, $\bar{u}_L$, $\bar{v}_L$, $\bar{h}_R$, $\bar{u}_R$, $\bar{v}_R$

algorithm 12.

Algorithm 12 `FVM`(h,q_x,q_y)

call `q2u`(h,u,v,q_x,q_y)
call `BC`$(h,\ u,\ v,\ h_W,\ u_W,\ v_W,\ h_E,\ u_E,\ v_E,\ h_N,\ u_N,\ v_N,\ h_S,\ u_S,\ v_S)$
$F_E =$ `Roe`$(h,\ u,\ v,\ h_E,\ u_E,\ v_E,\ flag)$
$F_W =$ `Roe`$(h,\ u,\ v,\ h_W,\ u_W,\ v_W,\ flag)$
$G_S =$ `Roe`$(h,\ u,\ v,\ h_S,\ u_S,\ v_S,\ flag)$
$G_N =$ `Roe`$(h,\ u,\ v,\ h_N,\ u_N,\ v_N,\ flag)$
$F = \frac{\Delta t}{\Delta x}(F_W - F_E) - \frac{\Delta t}{\Delta y}(G_S - G_N)$
return F

The `q2u` routine transforms the conservative variable $\mathbf{x}_c = (h, q_x, q_y) = (h, hu, hv)$ to the non-conservative variable $\mathbf{x} = (h, hu, hv)$. The adjoint procedure of the latter is described in appendix D.3. Given the adjoint procedure of the boundary condition and the Roe solver discussed previously, we deduce the discrete adjoint `FVM_b` described in algorithm 13.

Algorithm 13 `FVM_b`$(\bar{F})$

! Forward step ==========>
call `q2u`(h,u,v,q_x,q_y)
call `BC`$(h,\ u,\ v,\ h_W,\ u_W,\ v_W,\ h_E,\ u_E,\ v_E,\ h_N,\ u_N,\ v_N,\ h_S,\ u_S,\ v_S)$
! Backward step <==========
$\bar{F}_W = \frac{\Delta t}{\Delta x}\bar{F}$, $\bar{F}_E = -\frac{\Delta t}{\Delta x}\bar{F}$
$\bar{G}_S = \frac{\Delta t}{\Delta y}\bar{F}$, $\bar{G}_N = -\frac{\Delta t}{\Delta y}\bar{F}$
$[\bar{h}, \bar{u}, \bar{v}, \bar{h}_N, \bar{u}_N, \bar{v}_N] =$ `Roe_b`$(\bar{F}_N,1)$
$[\bar{h}_S, \bar{u}_S, \bar{v}_S, \bar{h}, \bar{u}, \bar{v}] =$ `Roe_b`$(\bar{F}_S,1)$
$[\bar{h}_W, \bar{u}_W, \bar{v}_W, \bar{h}, \bar{u}, \bar{v}] =$ `Roe_b`$(\bar{F}_W,0)$
$[\bar{h}, \bar{u}, \bar{v}, \bar{h}_E, \bar{u}_E, \bar{v}_E] =$ `Roe_b`$(\bar{F}_E,0)$
call `BC_b`$(\bar{h},\ \bar{u},\ \bar{v},\ \bar{h}_W,\ \bar{u}_W,\ \bar{v}_W,\ \bar{h}_E,\ \bar{u}_E,\ \bar{v}_E,\ \bar{h}_N,\ \bar{u}_N,\ \bar{v}_N,\ \bar{h}_S,\ \bar{u}_S,\ \bar{v}_S)$
call `q2u_b`$(h,\bar{h},u,\bar{u},v,\bar{v},q_x,\bar{q}_x,q_y,\bar{q}_y)$
return $\bar{h}$, $\bar{u}$, $\bar{v}$

3.3.3 Adjoint of the time integration scheme

The time integration is simply carried out by the RK3 method recalled below,

$$k_1 = f(\mathbf{x}_c^n), \tag{3.41}$$
$$k_2 = f(\mathbf{x}_c^n + 0.5k_1), \tag{3.42}$$
$$k_3 = f(\mathbf{x}_c^n + 2k_1 - k_2), \tag{3.43}$$
$$\mathbf{x}_c^{n+1} = \mathbf{x}_c^n + \frac{\Delta t}{6}(k_1 + k_2 + k_3), \tag{3.44}$$

where $\mathbf{x}_c^n$ is the conservative vector $\mathbf{x}_c^n = (h^n, q_x^n, q_y^n) = (h^n, h^n u^n, h^n v^n)$. The function f calculates the fluxes in each direction as described in § 3.1.2, in other words, $f(\mathbf{x}_c^n) =$ `FVM`(h^n, q_x^n, q_y^n).

Hager (2000) provided the general formulation of the Runge-Kutta methods discrete adjoint procedure and Sandu (2006) showed that the Runge-Kutta discrete adjoint method has the same order of accuracy as the original forward method. Hence, we apply below the general formulation of the discrete adjoint to the RK3 method :

$$k_1 = f(\mathbf{x}_c^n), \tag{3.45}$$
$$k_2 = f(\mathbf{x}_c^n + 0.5k_1), \tag{3.46}$$
$$l_1 = \partial_{\mathbf{x}} f^*(\bar{\mathbf{x}}^n), \tag{3.47}$$
$$l_2 = \partial_{\mathbf{x}} f^*(\bar{\mathbf{x}}^n + 0.5k_1), \tag{3.48}$$
$$l_3 = \partial_{\mathbf{x}} f^*(\bar{\mathbf{x}}^n + 2k_1 - k_2), \tag{3.49}$$
$$\bar{\mathbf{x}}^{n-1} = \bar{\mathbf{x}}^n + \frac{\Delta t}{6}(l_1 + 4l_2 + l_3), \tag{3.50}$$

where $\partial_{\mathbf{x}} f^*$ is the analytic adjoint operator corresponding to f. In our context, this operator corresponds to the adjoint routine `FVM_b` such that $f(\bar{\mathbf{x}}_c^n) = \texttt{FVM}(\bar{h}^n, \bar{q}_x^n, \bar{q}_y^n)$. We deduce the adjoint routine `RK3_b` summarized in algorithm 14.

Algorithm 14 `RK3_b`$(h^{n-1},\bar{h}^{n-1},q_x^{n-1},\bar{q}_x^{n-1},q_y^{n-1},\bar{q}_y^{n-1},h^n,\bar{h}^n,q_x^n,\bar{q}_x^n,q_y^n,\bar{q}_y^n)$

! Forward procedure ==========>
$\mathbf{x}^n = (h^n,q_x^n,q_y^n)$
$k_1 = \texttt{FVM}(h^n,q_x^n,q_y^n)$
$tmp = \mathbf{x}^n + 0.5 * k1$
$k_2 = \texttt{FVM}(tmp)$
! Backward procedure ==========>
$\bar{\mathbf{x}}^n = (\bar{h}^n,\bar{q}_x^n,\bar{q}_y^n)$
$l_1 =$`FVM_b`$(\bar{\mathbf{x}}^n)$
$l_2 =$`FVM_b`$(\bar{\mathbf{x}}^n + 0.5 * k1)$
$l_3 =$`FVM_b`$(\bar{\mathbf{x}}^n - k1 + 2 * k1)$
$\bar{h}^{n-1} = \bar{h}^n + \frac{1}{6}(l_1(1) + 4 * l_2(1) + l_3(1))$
$\bar{q}_x^{n-1} = \bar{q}_x^n + \frac{1}{6}(l_1(2) + 4 * l_2(2) + l_3(2))$
$\bar{q}_y^{n-1} = \bar{q}_y^n + \frac{1}{6}(l_1(3) + 4 * l_2(3) + l_3(3))$

3.3.4 Adjoint of the main code

Given the previous adjoint routines, we deduce the adjoint procedure corresponding to the main `SWE` routine in algorithm 15. As we work with a rather small sized problem, we choose to implement a store-all strategy which consists in saving every quantities h^n, u^n, v^n for each time step t^n and invoke them when needed.

3.3.5 Code validation

We validated previous tangent and adjoint procedures by applying the validation tests presented in § 2.4.7. We recall that the original `SWE` code studied in this chapter

Algorithm 15 SWE_b

Initialization: $n = N$, $\bar{h}^N = 0$, $\bar{u}^N = 0$,$\bar{v}^N = 0$
  while $n > 0$ **do**
    call `q2u_b`(h^n,$\bar{h}^n$,u^n,$\bar{u}^n$,v^n,$\bar{v}^n$,q_x^n,$\bar{q}_x^n$,q_y^n,$\bar{q}_y^n$)
    call `RK3_b`(h^{n-1},$\bar{h}^{n-1}$,q_x^{n-1},$\bar{q}_x^{n-1}$,q_y^{n-1},$\bar{q}_y^{n-1}$,h^n,$\bar{h}^n$,q_x^n,$\bar{q}_x^n$,q_y^n,$\bar{q}_y^n$)
    call `u2q_b`(h^{n-1},$\bar{h}^{n-1}$,u^{n-1},$\bar{u}^{n-1}$,v^{n-1},$\bar{v}^{n-1}$,q_x^{n-1},$\bar{q}_x^{n-1}$,q_y^{n-1},$\bar{q}_y^{n-1}$)
    $n = n - 1$
  end while
  return $\bar{h}^0$, $\bar{u}^0$, $\bar{v}^0$

was implemented in MATLAB, therefore we performed beforehand the conversion to an equivalent FORTRAN code to use the AD tool TAPENADE. The obtained tangent and adjoint procedures are then converted back to MATLAB procedure. Consequently, the following test were performed on the procedures converted to MATLAB.

Tangent process validation

We recall the expression of the tangent validation procedure for the operator f of interest, which represents in this case the original code `SWE`, and its corresponding tangent denoted ∇f,

$$R_1 := \frac{f(\mathbf{x}_0 + \alpha d) - f(\mathbf{x}_0)}{\alpha < \nabla f(x_0), d >}. \tag{3.51}$$

We evaluate the value of $1 - R_1$ for different differentiation step α over a temporal window of $10\Delta t'$ then $100\Delta t'$. The results are summarized in table 3.1. The tangent code accuracy meets our expectation as the ratio R_1 is close to 1 up to $10^{-\alpha}$. We obtain the same accuracy for both temporal windows.

α	$10\Delta t'$	$200\Delta t'$
10^{-3}	4.738510^{-5}	4.195710^{-5}
10^{-4}	2.561910^{-6}	4.116810^{-6}
10^{-5}	4.397310^{-7}	3.567710^{-7}
10^{-6}	4.445610^{-8}	3.537610^{-8}
10^{-7}	4.260110^{-9}	4.368910^{-9}
10^{-8}	4.305410^{-10}	8.092410^{-10}
10^{-9}	1.140810^{-9}	1.369510^{-9}

TABLE 3.1 – Evaluation of $1 - R_1$ results for different values of the differentiation step α over $10\Delta t'$ and $200\Delta t'$. These results are calculated in MATLAB's default accuracy.

Dot product validation

By definition, the adjoint procedure must correspond exactly to the tangent procedure associated to original code. Therefore, this validation test is carried out in double precision $O(10^{-16})$ and we expect the results to reach the same accuracy. We computed the ratio R_2 over $200\Delta t'$ obtained a satisfying result (3.52). Hence, the discrete adjoint corresponds accurately to the previous tangent procedure.

$$R_2 = \frac{< (\partial_X \mathbb{M})X, (\partial_X \mathbb{M})X >}{< X, (\partial_X \mathbb{M}^{\#})Y >} = \frac{< \dot{Y}, \dot{Y} >}{< \dot{X}, \bar{Y} >} = 1.000000000000220. \tag{3.52}$$

Adjoint process validation

Finally, we evaluated the validity of the entire 4DVar assimilation implementation by computing the ratio R_3 for several differential step α,

$$R_3 = \frac{J(\mathbf{x}_0 + \alpha d) - J(\mathbf{x}_0)}{\alpha < \boldsymbol{\lambda}(t_0), d >}. \tag{3.53}$$

We recall that we don't require the latter to be close to 1 as the first order finite difference provides a lowly accurate estimate of the cost function. The results are summarized in table 3.2.

α	R_3
10^{-3}	0.88820
10^{-4}	1.05709
10^{-5}	1.07748
10^{-6}	1.06722
10^{-7}	1.05659
10^{-8}	1.04759
10^{-9}	1.07377

TABLE 3.2 – 4DVar validation result for different values of the differentiation step α over an assimilation window $T' = 200\Delta t'$.

3.4 Application to the reconstruction of a free surface characteristics

In this chapter, we seek for the complete characteristics (height and velocity field) of the free surface of a fluid contained in a rectangular flat bottom tank of size $L_x \times L_y = 250\text{mm} \times 100\text{mm}$. Since we have a numerical code which describes the evolution at the free surface in time (see § 3.1.1) and a set of observations, we apply and compare the following data assimilation strategies :

– An **incremental 4DVar** technique, based on Courtier et al. (1994), which requires the construction of the discrete adjoint code described in § 3.3. We fixed the background error co-variance matrix as a static diagonal matrix $\mathbf{B} = \sigma_b^2 \mathbb{I}$ for both experiments. σ_b was optimally tuned as the standard deviation between the true solution and the *a priori* experimental initial state.
– Several hybrid data assimilation strategies based on the **4DEnVar** developed in Liu et al. (2008), labeled by the suffix "Liu-et-al" in the following figures. The strategies with several outer loops, perturbed observation or direct transformation are indicated by suffix "OL", "OP" and "DT" respectively. Covariance localization and local ensemble are indicated by suffix "LC", "LE" accordingly. In terms of background error covariance matrix, it is crucial that the initial ensemble represents correctly the background errors, therefore the latter are specified in each case. The reader can find complete study of these strategies in Yang (2014).

3.4.1 Synthetic experiment

General context

In this purely synthetic experiment, we define a reference initial condition as a smooth slope tilted along the x-axis and the y-axis by 21% and 10% respectively. The initial reference velocity field was fixed as a Gaussian field with a standard deviation of 1mm/s. The data assimilation techniques are run on a meshgrid of $n_x \times n_y = 101 \times 41$ nodes. The background is set as a smooth 20% slope along the x-axis only. The slight slope in the y direction induced in the reference height contributes to the divergence between the reference and the background trajectory. The initial reference and background configurations are illustrate in figure 3.5.

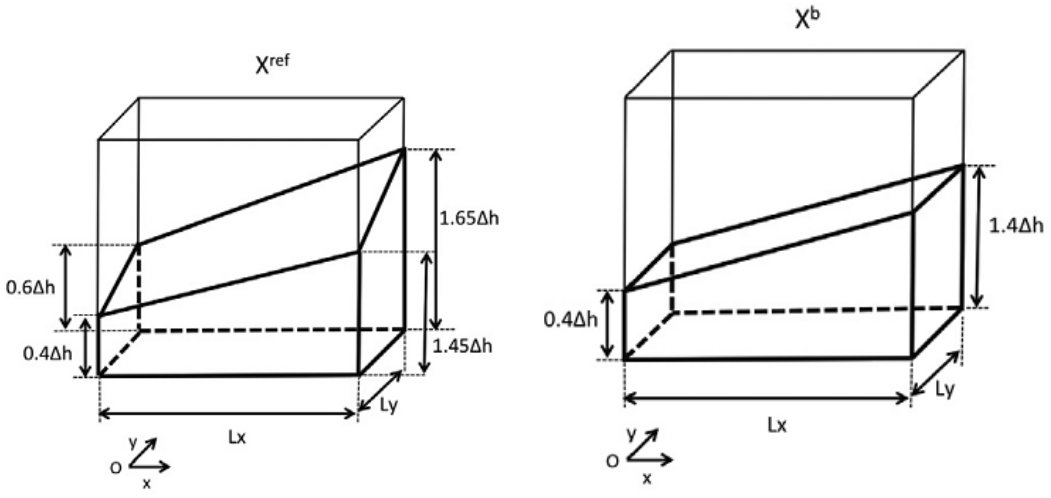

FIGURE 3.5 – Representation of the initial configuration of the reference (on the left) and the background (on the right). The reference is obtained by a smooth 21% slope along the x-axis and a 10% slope along the y-axis. The background height is set as a smooth slope at 20% slope along the x-axis.

In terms of the initialization of the ensemble members, we set up two kind of perturbations of the initial condition. In the one hand, we opt for a Gaussian perturbation of the background. On the other hand, we opt for a parameter perturbation strategy which consists in generating heights with random smooth slope.

The synthetic observations are generated by adding i.i.d Gaussian noise perturbations to the reference free surface height and velocity fields at each grid point. In this experiment, we observe either the free surface height only, or the free surface velocity field only or both at the same time. The observations are acquired every $\Delta t'_{\text{obs}} = 50\Delta t{\cdot}U/L_x$ of the dynamic step $\Delta t = 1e^{-3}$s. The common temporal setup of both experiments is illustrated in figure 3.6. The height and velocity observation errors are specified as 0.25mm and 1mm/s respectively.

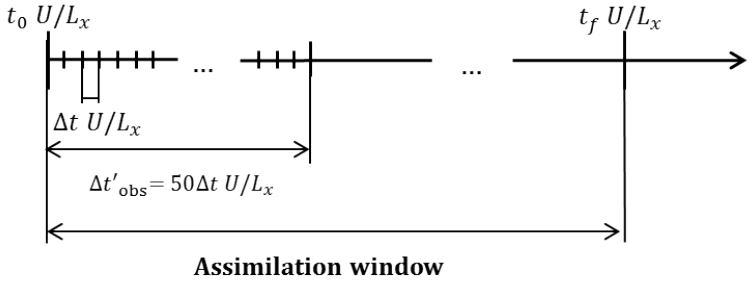

FIGURE 3.6 – Temporal setup for the data assimilation techniques : we perform the assimilation over a period of $t_f{\cdot}U/L_x$. The observations are injected each $\Delta t'_{\text{obs}} = 50\Delta t{\cdot}U/L_x$.

In this experiment, we define the characteristic height $\Delta h = 50$mm as the difference between the maximum and minimum surface height of the initial background state. We also define the characteristic velocity $U = \sqrt{g\Delta h} = 0.34$ m/s as an approximation of the wave phase velocity. For every data assimilation techniques, we performed a single assimilation window of duration $t_f U/L_x = 0.8951$. All the following results are adimensionalized with respect to these characteristic values.

We analyze the following results by computing the Root Mean Square Error (RMSE) defined for the given state of interest $\mathbf{x}$ with respect to the reference state $\mathbf{x}_{\text{ref}}$. We assess display the evolution in time of the RMSE for the background, observation and the resulting analysis states with respect to the values of the reference state. The RMSE function is defined as

$$RMSE(\mathbf{x}) = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\mathbf{x} - \mathbf{x}_{\text{ref}})^2}, \tag{3.54}$$

where N represents the total size of the state $\mathbf{x}$. We also consider the RMSE comparison on a semi-logarithmic graph. In general, a lower RMSE at the end of the assimilation window indicates better performance in terms of the forecast, and a lower temporal RMSE indicates better performance in terms of reconstruction.

Height observations

In this configuration, we only take into account noisy free surface height observations. The En4DVar techniques provided comparable results with the 4DVar for an ensemble of 8 members. Figure 3.7 shows that in terms of the RMSE comparison, the 4DEnVar yields to the best reconstruction of each variables than the standard 4DVar assimilation techniques. Both methods performed strong corrections of the height whereas the standard 4DVar seems to have difficulties to reconstruct the non-observed velocity fields.

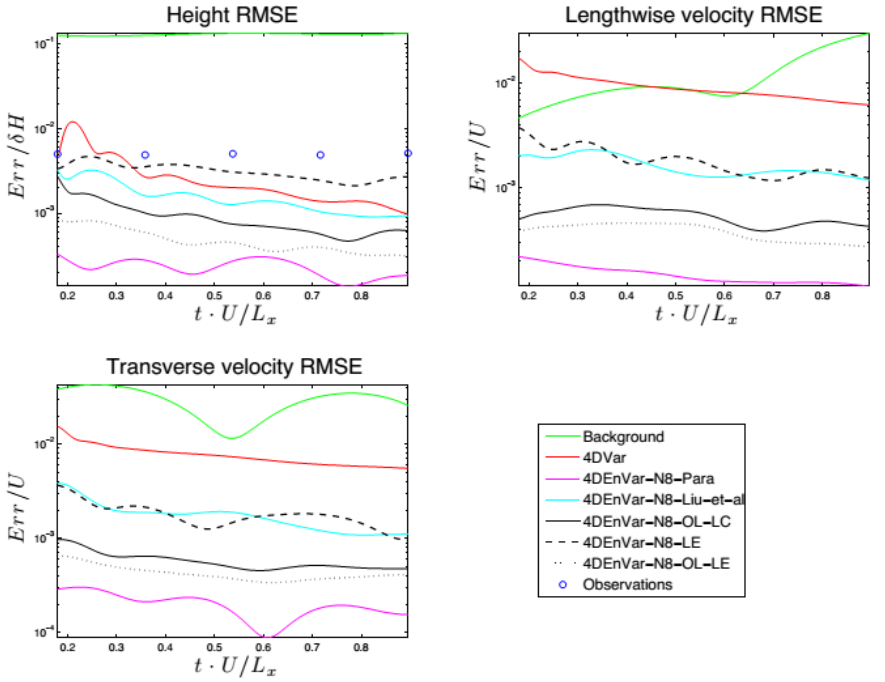

FIGURE 3.7 – RMSE comparison between an incremental 4DVar and 4DEnVar assimilation approaches with a a partial height observation through a noisy free surface height.

Concerning the 4DEnVar methods, the parameter perturbation strategy (4DEnVar-Para) provided the best reconstruction of the complete state. In order to obtain comparable results with the Gaussian perturbation strategy, a localization is necessary and the cutoff distance was set as the optimal cutoff distance minimizing the average RMSE (the same value was used to define the size of local space). The distance increases as the ensemble member increases. We also observe that several outer loops clearly improve the results (OL-LC against Liu-et-al and OL-LE against LE), which highlights the pertinence of the background covariance update. The method with local ensemble (LE) provided slightly better results than the method with localized covariance (LC).

Velocity observations

This time, we only take into account noisy free surface velocity observations. Figure 3.8 reveals a similar behavior than the previous case as the 4DEnVar techniques globally provided a better reconstruction than the standard 4DVar.

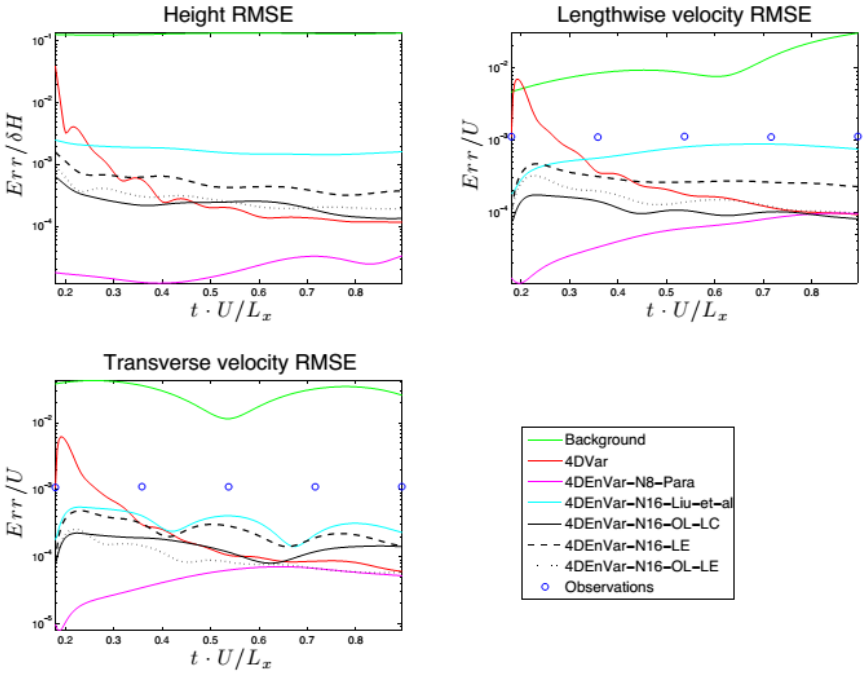

FIGURE 3.8 – RMSE comparison between an incremental 4DVar and 4DEnVar assimilation approaches with a a partial velocity observation through a noisy free surface height.

As in the previous case, the parameter perturbation strategy (4DEnVar-Para) provides the best reconstruction of the complete state for a low ensemble member $N = 8$. On the other hand, the Gaussian perturbation strategy required a moderately higher ensemble number $N = 16$ to obtained better comparable results to the 4DVar. The increase of the ensemble size to $N = 32$ slightly improves the quality of the results at the expense of a higher computational cost.

Height and velocity observations

Finally, we performed the reconstruction with complete observations of the system of interest. The comparison of the each data assimilation techniques provided a very similar behavior than in the previous case. Indeed, the ensemble techniques provided a better reconstruction than the 4DVar in terms of the RMSE evolution. However the advantage of the ensemble methods towards the 4DVar is less enhanced than in the previous case. As we possess the complete observations of the system, we assume that each variable component is mainly corrected by the corresponding observations than indirectly inferred from other observed components.

In terms of the ensemble methods comparison, we obtain globally the same conclusions as previously. The parameter perturbation strategy provided a slightly

better result than the Gaussian perturbation. The localization ensemble (LE) approach generally led to better results than the localization covariance (LC) approach, however the more we increase the ensemble number N the more the results of both approaches are alike.

As the data assimilation techniques are comparable in terms of RMSE, we compare their computational needs in table 3.3. The parameter perturbation approach (4DEnVar-PP), requires the less CPU time among all the data assimilation techniques compared. The 4DVar approach requires the most CPU time as it performs several forward and backward time integrations, however in the upside, this technique requires a smaller memory resources. Concerning the Gaussian perturbation, for an equal number of ensemble member, the local covariance approach costs more in time and in memory than the local ensemble approach.

Data assimilation method	**CPU time**	**Memory demands**
4DVar	3200s	Small
4DEnVar-PP (no localization, N=8)	120s	Small
4DEnVar-LC($N = 32$)	2400s	Huge
4DEnVar-LE($N = 32$)	600s	Small

TABLE 3.3 – Comparison of the computation resources required for the classic 4DVar method (4DVar), the parameter perturbation approach (4DEnVar-PP), Gaussian perturbation with a local covariance approach for $N = 32$ ensemble members (4DVar-LC) and the Gaussian perturbation with a local ensemble approach for $N = 32$ ensemble members.

Conclusion

The ensemble based data assimilation techniques showed great potential in handling incomplete and noisy observations and each 4DEnVAr strategies clearly outperformed the 4DVar reconstruction in terms of RMSE. Regarding the ensemble initialization, a physical noise, in our case the generation of smooth random slopes, is preferred over non-physical noise which requires a localization procedure. However, in practice we often don't have a profound knowledge of both the dynamical model and the observations to generate a good parameter sampling.

When we consider a set of complete observations, the ensemble-based methods provide similar results with respect to the 4DVar technique. Table 3.3 showed that the 4DVar technique requires the highest CPU time for a relatively small memory cost. Although, these results assume that the En4DVar methods are parallelized beforehand which can represent a tedious task more complex dynamical systems.

3.4.2 Real image data experiment

Experimental observations

The experimental setup simply consists in measuring the height of a colored liquid contained in a tank with a Kinect depth sensor placed above the tank as illustrated in figure 3.9. The measurements were acquired on a 222×88 grid. Combes et al. (2011) described in details the measurement technique employed to extract the surface height from the raw data captured by the Kinect sensor.

FIGURE 3.9 – Experimental setup with the Kinect depth sensor

The free surface height observations are characterized by a high level of noise and exhibit large regions of missing data mostly along the boarders which are due to light reflections on the tank's wall. We performed a sequence of procedures that transforms the sequence of spatially sparse height observations into complete height observations defined on 248×98 nodes. However, the computational limitations leads us to run the data assimilation on 124×49 nodes instead. Therefore, the observations are interpolated on the coarse grid. The entire pre-procedure that transforms the height observations acquired on the 222×88 grid into a format suitable to run the assimilation techniques is summarized in algorithm 16.

Algorithm 16 Real observations pre-processing

1: Eliminate singular points whose value exceeds a given threshold
2: Fill the missing boundary areas located besides the long borders with the average of the all the pixels' values located in the same section
3: Extrapolate the height profile to the missing boundary areas located besides the short boarders
4: Fill the missing inner hole areas with pseudo-values computed as the mean of all the adjacent pixel's observation values

Similarly to the synthetic configuration, we consider an adimensionalized system

with respect to the characteristic height, velocity and time defined as $\Delta h = 12$mm, $U = \sqrt{g\Delta h} = 0.34$ m/s and $t_f \cdot U/L_x = 0.4608$ respectively.

We estimate the observation error within an unobserved region with respect to the distance from the closest observed point. Thus, the longer the distance, the larger the error. The observation error is however bounded by a threshold of $60\%\Delta h$.

The observation time step was tuned cautiously to $\Delta t'_{\text{obs}} = 30\Delta t \cdot U/L_x$ such that the observation sequence is synchronized with the dynamics model.

3.4.3 Choice of the background

In this framework, the initial background was built from the partial observations provided by the Kinect measurements. The initial height is deduced by filtering the observation at the initial time which was obtained by the pre-processing described in algorithm 16. As we notice that the observed free surface behaved roughly as an unidirectional wave along the x-axis, we set the initial x-axis velocity field as a smooth linear slope. The velocity at the top of the wave was set as 23% of the wave velocity and the velocity at the bottom of the wave was set to 0. The transverse velocity field was set to $v = 0$. The initial observation and background height and velocity are illustrated in figure 3.10.

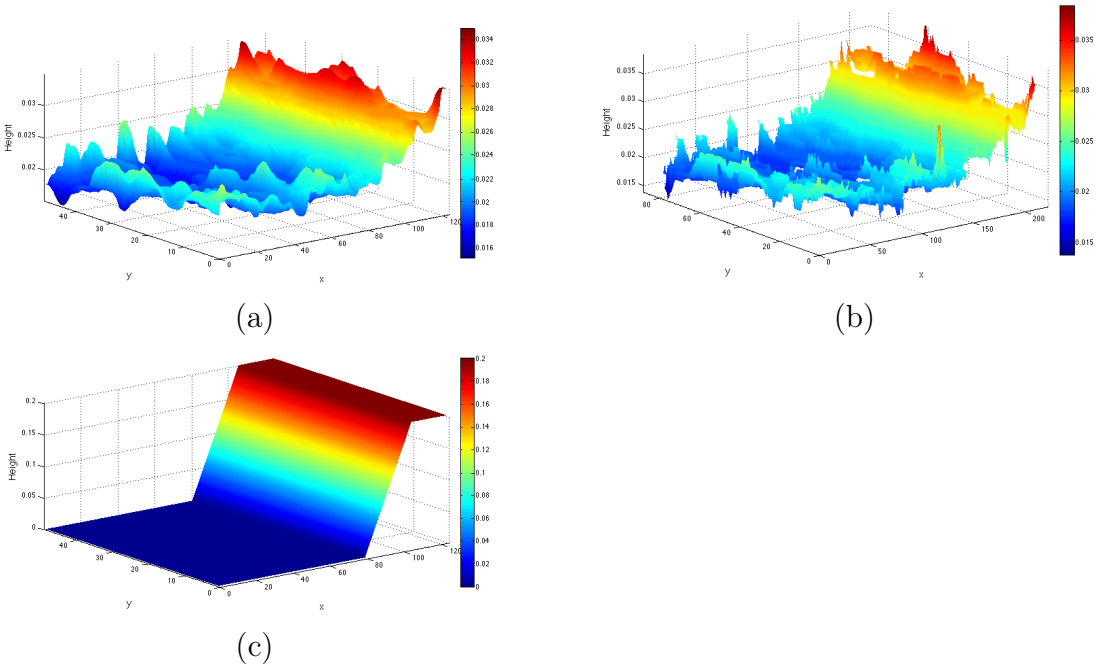

FIGURE 3.10 – Observed free surface height retrieved from the Kinect depth map (a), interpolated free surface height used as the background (b), background velocity set accordingly to the height observation (c)

In terms of ensemble sampling, we initialize the ensemble members by adding isotropic Gaussian perturbation fields with standard deviation $\sigma_b = 3.6\%\Delta h$ to the background state. The cutoff distance and the size of local space are fixed as $15\%L_x$.

The assimilation scheme was adapted to sliding assimilation windows to avoid long range temporal correlations. Each window contains five observations and we adopted five windows over nine observations, as illustrated in figure 3.11.

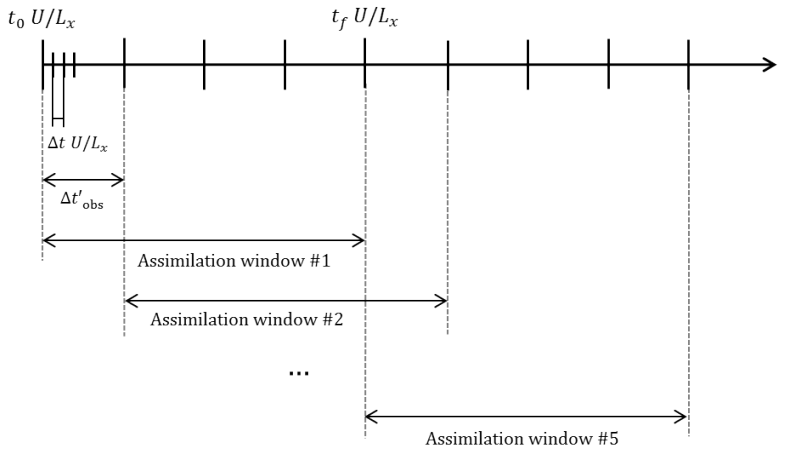

FIGURE 3.11 – Temporal setup for the sliding data assimilation techniques : we perform the assimilation over a period of $t_f \cdot U/L_x$. The observations are injected each $\Delta t'_{\text{obs}} = 30\Delta t \cdot U/L_x$. We then slide the assimilation window to the next observation and perform once again the data assimilation. This procedure is repeated until we reach the ninth observation.

3.4.4 Results

In opposition to the synthetic case, we don't possess a reference state in this case. We can only assume the true solution to lie within an interval around the observation. Therefore, instead of studying the RMSE with respect to the observations, we compare directly the height fields.

Figure 3.12 shows the evolution of the average surface height of the wave crest of the background, the observations and the result of the different data assimilation techniques. As the observations indicate that the free surface behaves globally as a single wave, we particularly focus on the wave crest's region rather than hte other flat regions. While the background diverges from the observations, we observe that the 4DVar and the 4DEnVar can both follow the observation trajectory tendency. The 4DVar tends to understimate the surface height at the beginning of the simulation whereas the ensemble-based methods follow the trend imposed by the 1st to the 4th observation. After the fifth observation, the En4DVar (Liu et al., 2008) diverges from the observation trajectory.

We also compared the free surface in figure 3.13. According to these free surfaces, we can see that the 4DVar solution showed some difficulties to handle the discontinuities at the boundaries of the regions in which the data have been extrapolated. On the other hand, the En4DVar provided a much smoother solution on the borders which corresponded clearly to a better compromise between the observation and the model.

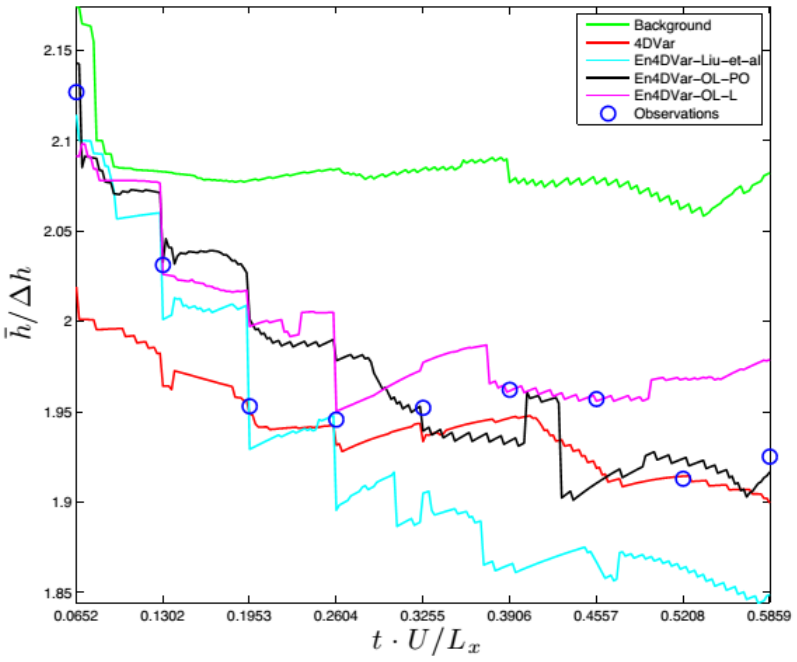

FIGURE 3.12 – Comparison of the evolution of the mean surface height of the wave crest region.a

3.5 Conclusion and perspectives

In this chapter, we evaluated the capacity of a classical incremental method to reconstruct the complete characteristics (height and velocity field) at the free surface of a unidirectional wave from partial and noisy observations. We had the opportunity to compare the results with the ensemble-based method 4DEnVar, proposed by (Liu et al., 2009), and various strategies based on the latter (Yang et al., 2015).

In general, both methods retrieved successfully the complete characteristics of the free surface in the synthetic and the experimental case. In opposition to the varitional approach, the major advantage of the ensemble-base methods lies in the fact that we avoid the tedious task of constructing the adjoint procedure. In general, the ensemble-based methods provided better results in terms of RMSE for partial and complete observations. In return, these techniques require a significant number of ensemble members to have comparable results with the 4DVar technique. A parametric perturbation of the initial ensemble clearly improves the assimilation process and requires the less computational ressources. However, we often don't have enough knowledge on the system of interest, especially in industrial applications, to generate an adequate physical perturbation. The parallelization of the ensemble members is also required to reach a computational comparable to the 4DVar. While the parallelization can be simply implemented for simple dynamical models, this task gets tougher when we consider a DNS or even a LES model, as the latter often require a parallelization by domain decomposition.

Finally, we note that the incremental 4DVar technique used throughout this chapter could be enhanced by applying a preconditioning technique similar to § 3.2.1. We could also refine the background error covariance matrix, fixed as $\mathbf{B} = \sigma_b^2 \mathbb{I}$

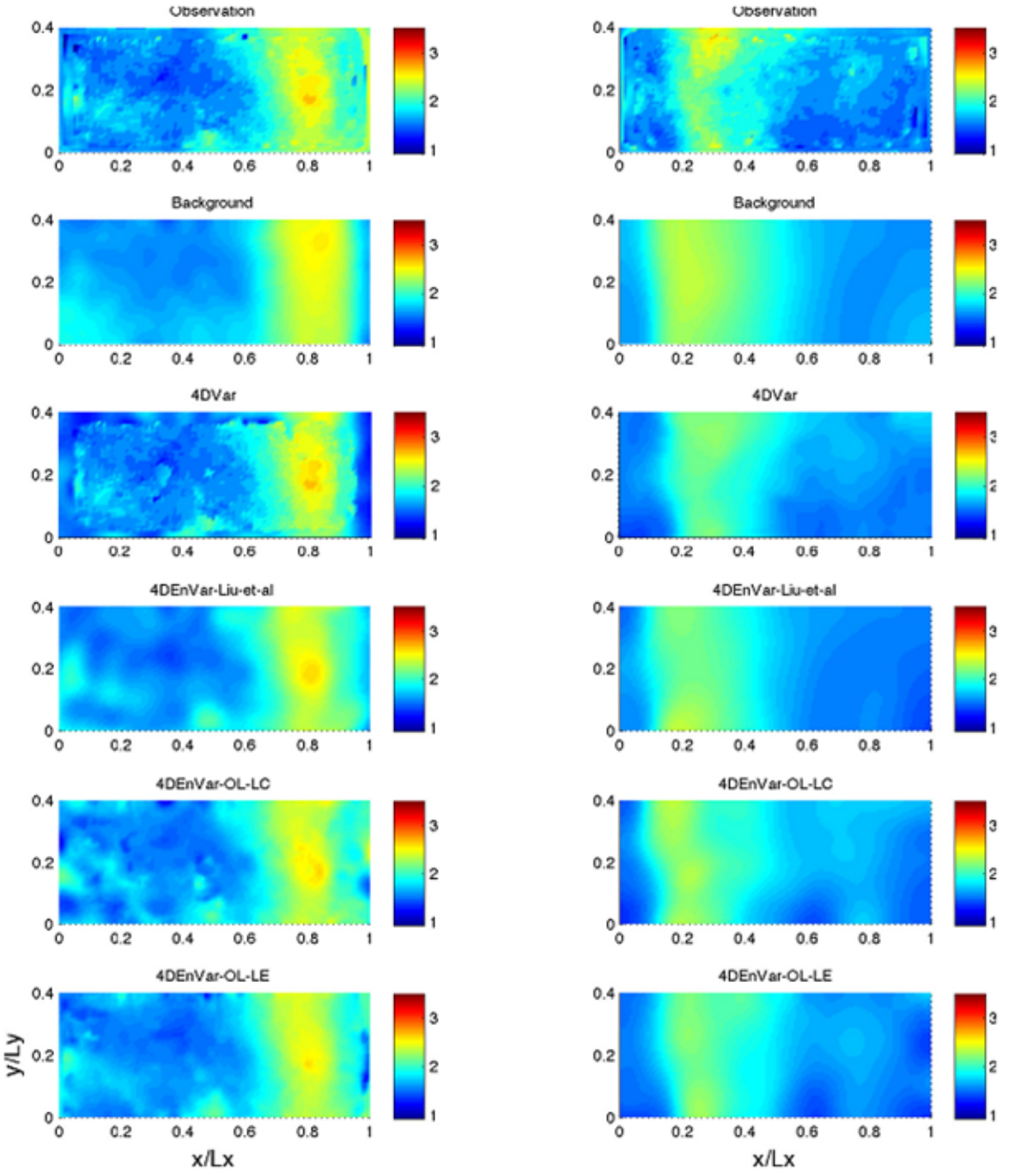

FIGURE 3.13 – Height field comparison at $t{\cdot}U/L_x \approx 0.0652$ (in the left) and at $t{\cdot}U/L_x \approx 0.5859$ (in the right). From top to bottom : observation, background, 4DVar, En4DVar proposed by Liu et al. (2008), the 4DEnVar-OL-LC and the 4DEnVar-OL-LE.

throughout this chapter, by injecting the dependence of each variables, height and velocity, from the spatial correlation.

On the other, as the experimental setup is simple to reproduce, we could also add a sensor that measures the free surface velocity field and lead a comparative

study as in the synthetic case § 3.4.1. Despite these implementations are out of the scope of this present thesis, the latter can pave the way to significant improvements of the results.

In the following chapter, we chose to remain in variational framework as the gain provided by the En4DVar techniques are relatively small with respect to the implementation difficulties.

Chapitre 4

Reconstruction of a turbulent 3D cylinder wake flow

The reconstruction of three dimensional fundamental turbulent flows is a cornerstone for fluid mechanics applications. As discussed in § 1.3.2, several papers hinted towards the use of data assimilation techniques which combine a numerical model with a set of observations. In this perspective, we aim at building and validating an incremental 4DVar code which is capable to reconstruct a turbulent flow at $Re = 300$ by combining a DNS code with 2D or 3D observations.

The CFD field provides a wide range of models and numerical schemes to simulate a three dimensional turbulent flow by solving the Navier-Stokes equations (see § 1.1). In the present application, we seek for an accurate representation of the flow, therefore we employed the parallel DNS code Incompact3d developed by Laizet and Lamballais (2009). In § 4.1, we describe the structure and present briefly the numerical schemes employed in Incompact3d. We describe in § 4.2 the construction and the numerical validation of the adjoint procedure necessary to implement the assimilation technique. We apply and assess the 4DVar code on a ideal experiment in § 4.3 with a set of synthetic 3D observations. Finally, we investigate the possibility to reconstruct a 3D cylinder wake flow with orthogonal-plane stereo PIV observations in § 4.4.

4.1 DNS code Incompact3D

We consider the Navier Stokes equations, recalled below, as the dynamics model to describe the evolution of the incompressible fluids of interest,

$$\nabla.\mathbf{u} = 0, \tag{4.1}$$

$$\partial_t \mathbf{u} + \frac{1}{2}\left(\nabla(\mathbf{u}\otimes\mathbf{u}) + (\mathbf{u}\nabla)\mathbf{u}\right) - \nu\nabla^2\mathbf{u} + \nabla\mathbf{p} = 0. \tag{4.2}$$

In this work, we use the parallel DNS code Incompact3D to solve these equations (Laizet and Lamballais, 2009; Laizet et al., 2010). Incompact3d is a highly accurate code which combines high-order compact schemes with a spectral solver. It was initially designed for serial processors, then converted to vector processors

and more recently converted to parallel platforms (Laizet et al., 2010). The parallel computation, which is achieved by using a 2D decomposition library decomp2d (Li and Laizet, 2010), is a very important asset especially for fluid mechanics applications which usually involve large dimension variables. This section provides a succinct description of the main procedures implemented in Incompact3d. First of all, we present briefly the numerical aspects such as the 2D decomposition library (see § 4.1.1) and the required computational resources (see § 4.1.2). We then present the discretization schemes of the time advancement, the convection and diffusion terms and the pressure treatment in § 4.1.3, § 4.1.4 and § 4.1.5 respectively.

4.1.1 2d decomposition

The 2D decomposition library decomp2d (Li and Laizet, 2010) decomposes the computation domain ($n_x \times n_y \times n_z$) into "pencils" along a given direction x, y or z, respectively referred to as X-pencil, Y-pencil and Z-pencil, as illustrated by figure 4.1. The communication between each processor is taken in charge by "transpose" subroutines provided by 2Ddecomp library. Each processor performs the same instructions bloc to the input data. For instance, let's assume we start from the X-pencil configuration and we aim at computing a gradient $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) \approx \left(f_i', f_j', f_k'\right)$. We compute at first the 1D derivative in the x-axis, f_i', then we swap to the Y-Pencil to compute 1D derivative in the y-axis, f_j', and finally swap to the Z-Pencil to compute the last 1D derivative in the z-axis, f_k'. This procedure is extended to every instruction of Incompact3d with the necessary swap operations. For the sake of clarity, we voluntarily omit to mention the swap operations in the following sections.

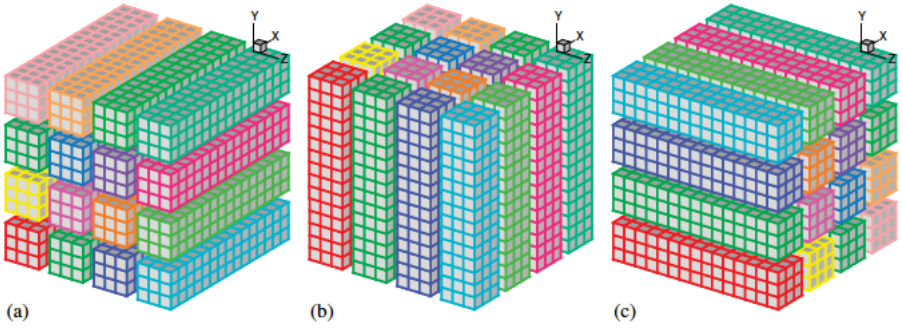

FIGURE 4.1 – 2D domain decomposition example of a 3D computation domain using a 4×4 processor grid : (a) X-pencil, (b) Y-pencil and (c) Z-pencil.

4.1.2 Computational resources

In terms of computational resources, every calculation of this thesis was run on the computer grid IGRIDA which offers a 125 computer nodes (1200 cores). In terms of storage, IGRIDA offers to each member of the research teams at IRISA/INRIA a shared scratch space of 1.7 TB for temporarily storing simulations inputs and outputs. There are also 5 GPU nodes with NVIDIA GPUs which hasn't been used during this thesis. The computation resources provided by each cluster used in the

Name	CPU	REFERENCE	RAM	DISK	Network
Calda	2 x 8 cores Sandy Bridge	Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz	48GB	2 x 600GB	Infiniband + 1GB/s
Panda	2 x 4 cores Clovertown	Intel(R) Xeon(R) CPU L5310 0 @ 1.60GHz	8GB	1 x 73GB	1 GB/s
Gouda	2 x 4 cores Clovertown	Intel(R) Xeon(R) CPU E5345 0 @ 2.33GHz	8GB	73GB	1 GB/s SAS 15k

TABLE 4.1 – List of the clusters used in this thesis

framework of this work are summarized in table 4.1. The computational time and memory resources strongly depend on the size of the domain of study, therefore they will be precised in § 4.3 and § 4.4.

4.1.3 Time advancement

The general formulation of the time advancement of the equation (4.2), which deduces the state u^{k+1} at time t_{k+1} from the previous state u^k at time t_k, can be expressed as

$$\frac{\mathbf{u}^* - \mathbf{u}^k}{\Delta t} = a_k F^k + b_k F^{k-1} + c_k F^{k-2} - g_k \nabla \tilde{p}^k, \tag{4.3}$$

$$\frac{\mathbf{u}^{**} - \mathbf{u}^*}{\Delta t} = g_k \nabla \tilde{p}^k, \tag{4.4}$$

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^{**}}{\Delta t} = -g_k \nabla \tilde{p}^{k+1}, \tag{4.5}$$

with

$$F^k = -\frac{1}{2}\left[\nabla(u^k \otimes u^k) + (u^k.\nabla)u^k\right] + \nu \nabla^2 u^k \tag{4.6}$$

and the time averaged values on a given sub-step $g_k \Delta t$:

$$\tilde{p}^{k+1} = \frac{1}{g_k \Delta t} \int_{t_k}^{t_{k+1}} p(t) dt \tag{4.7}$$

We specify the discretization scheme by fixing the coefficient triplet $\{a_k, b_k, c_k\}_{k=1,\cdots,n_k}$ (where $g_k = a_k + b_k + c_k$ for simplicity) on n_k sub-time steps with $t_1 = t_n$ and $t_{n_k} = t_{n+1}$ ($\Delta t = t_{n+1} - t_n$ being the full time step). The time discretization schemes available in Incompact3d are the Runge Kutta methods at order 3 (RK3) and 4 (RK4), and the Adams-Bashford methods at order 2 (AB2) and 3 (AB3). The values attributed to each coefficient for each technique are summarized in appendix B. In this thesis, we kept the default numerical scheme AB3 for which $n_k = 1$ and (a_1,b_1,c_1) = (23/12, −16/12, 5/12).

4.1.4 Spatial discretization of convective and diffusive terms

The convection and diffusion terms of the Navier-Stokes equations (4.6) are directly solved in the computational domain $L_x \times L_y \times L_z$ discretized on a regular three-dimensional Cartesian mesh of $n_x \times n_y \times n_z$ nodes. Despite this meshing choice, the code is capable to treat complex geometries via an Immersed Boundary Method (Parnaudeau et al., 2004). The discretization is carried out by a sixth-order, and a second-order on the boundaries, compact finite difference scheme. The following description is based on Laizet et al. (2010).

Let us consider a uniform 1D meshgrid consisting of the distribution of n_x nodes $\{x_i\}_{i=1,\cdots,n_x}$ defined on the domain $[0; L_x]$. A given derivative $f'(x_i)$ at the node x_i is approximated by its numerical derivative $f'_i \approx f'(x_i)$ as follow :

$$\alpha f'_{i-1} + f'_i + \alpha f'_{i+1} = a\frac{f_{i+1} - f_{i-1}}{2\Delta x} + b\frac{f_{i+2} - f_{i-2}}{4\Delta x}, \tag{4.8}$$

where the coefficients are given by $\alpha = 1/3$, $a = 14/9$ and $b = 1/9$. The second order derivative is therefore deduced by :

$$\alpha f''_{i-1} + f''_i + \alpha f''_{i+1} = a\frac{f_{i+1} - 2f - i + f_{i-1}}{2\Delta x^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4\Delta x^2} + c\frac{f_{i+3} - 2f_i + f_{i-3}}{9\Delta x^2}, \tag{4.9}$$

where the coefficients are given by $\alpha = 2/11$, $a = 12/11$, $b = 3/11$ and $c = 0$. The latter is a sixth-order like approximation with the same favorable properties than the first derivative in terms of spectral-like resolution.

In practice, Incompact3d offers the possibility to impose a combination of three types of boundary conditions : periodic, free-slip and open boundary conditions.

Periodic condition conserves the order of the numerical scheme and is easy to implement. The values of f, f', f'' at the boundaries are simply deduced by :

$$f_0 = f_{n_x}, f_{-1} = f_{n_x-1}, f'_0 = f'_{n_x}, f''_0 = f''_{n_x}.$$

Free slip condition has the same assets than the periodic condition and corresponds whether to the symmetric condition

$$f_0 = f_2, f_{-1} = f_3, f'_0 = f'_2, f''_0 = f''_2,$$

or the asymmetric condition

$$f_0 = -f_2, f_{-1} = -f_3, f'_0 = f'_2, f''_0 = -f''_2.$$

Open boundary condition doesn't make any assumptions on the flow outside the computational domain. Single sided formulations are used for the approximation of first derivatives and second derivatives for these types of boundaries using relations of the form :

$$f'_1 + 2f'_2 = \frac{1}{2\Delta x}(-5f_1 + 4f_2 + f_3),$$
$$f''_1 + 11f''_2 = \frac{1}{2\Delta x^2}(13f_1 - 27f_2 + 15f_3 - f_4),$$

that are third order accurate (Lele, 1992). At the adjacent nodes, because a three point formulation must be used. Padé schemes are employed with forth-order accurate schemes :

$$\frac{1}{4}f_1' + f_2' + \frac{1}{4}f_3' = \frac{3}{2}\frac{f_3 - f_1}{2\Delta x},$$
$$\frac{1}{10}f_1'' + f_2'' + \frac{1}{10}f_3'' = \frac{6}{5}\frac{f_3 - 2f_2 + f_1}{\Delta x}.$$

The open boundary conditions are thereafter specified at the 2nd step of the time advancement (4.4). In our application, as we aim at modeling the downstream of a cylinder wake without knowing the obstacle geometry. Therefore, the inlet is assumed to be known beforehand and is imposed at each time step Δt of the DNS. The outlet is modeled by a first-order advection model in the streamwise direction that reads :

$$\mathbf{u}_{n_x}^{BC} = \mathbf{u}_{n_x}^* - g_k \frac{\Delta x}{\Delta t} u_{n_x}^* (\mathbf{u}_{n_x}^* - \mathbf{u}_{n_x-1}^*) \tag{4.10}$$

4.1.5 Pressure treatment

The tedious incompressibility condition (4.1) can be verified at the end of each sub-time step

$$\nabla \cdot \mathbf{u}^{k+1} = 0, \tag{4.11}$$

through solving a Poisson equation

$$\nabla^2 \tilde{p}^{k+1} = \frac{\nabla \cdot \mathbf{u}^{**}}{g_k \Delta t}, \tag{4.12}$$

that provides the estimation of the pressure $\tilde{p}^{k+1}$ from the velocity $\mathbf{u}^{**}$ (obtained by (4.4)) required to perform the correction (4.5).

The inversion of a 3D Poisson equation is in itself a tedious task which requires sophisticated and expensive methods such as in Mercier and Deville (1981) who use multidimensional compact high order schemes. On the other hand, the Poisson equation can be easily solved in Fourier space by the means of a Fast Fourier Transform (FFT) library. Laizet et al. (2010) provides a review of the parallel FFT libraries to be implemented in Incompact3d. Initial attempts were carried out to adapt existing FFT packages, however each library has their limitations and aren't always practical in terms of implementation with the decomposition library. Li and Laizet (2010) created from scratch a generic FFT based on the derivation of Glassmans's general N Fast Fourier Transform (Ferguson Jr, 1979) and avalaible at http://www.jjj.de/fft/glassman-fft.f. The latter is written in FORTRAN 77 and provides an interface which supports complex to complex, real to complex/complex to real transforms.

In order to reduce any spurious oscillations on the pressure field, the pressure discretization is performed on a regular staggered mesh grid shifted by a half-mesh in each spatial direction as illustrated in figure figure 4.2. The partial staggering, initially proposed by Wilhelmson and Ericksen (1977) and further developed by Canuto et al. (1988) and Chen et al. (2010) using second-order schemes, is easy

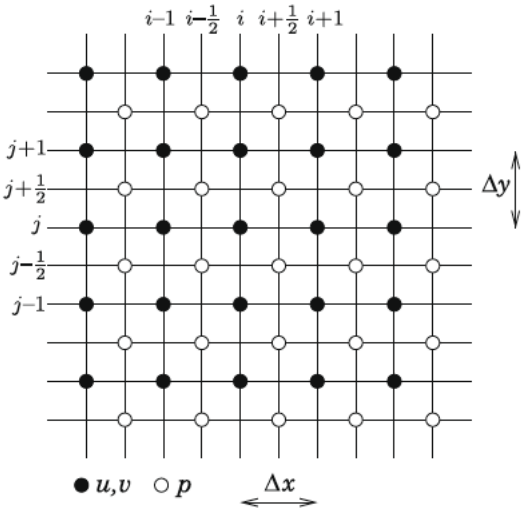

FIGURE 4.2 – Representation of the velocity meshgrid (black points) and the pressure meshgrid (white points). The pressure meshgrid is obtained by shifting by a half-mesh, $\Delta x/2$ and $\Delta y/2$, along the spatial direction x and y respectively.

to implement because it does not require extensive use of mid-point interpolations which are computationally expensive when combined with high-order schemes.

The evaluation of the first order derivative $f'_{i+1/2}$ at the staggered nodes is given by a sixth-order finite difference scheme expressed by :

$$\alpha f'_{i-1/2} + f'_{i+1/2} + \alpha f'_{i+3/2} = a\frac{f_{i+1} - f - i}{2\Delta x^2} + b\frac{f_{i+2} - f_{i-1}}{3\Delta x^2}, \tag{4.13}$$

with $\alpha = 9/62$, $a = 63/62$ and $b = 17/62$. The spectral behaviour of this scheme is better than its collocated counterpart (4.8), however we have to consider in addition the combination with a mid-point intepolation procedure which is given by :

$$\alpha f'_{i-1/2} + f'_{i+1/2} + \alpha f'_{i+3/2} = a\frac{f_{i+1} + f_i}{2} + b\frac{f_{i+2} + f_{i-1}}{2}, \tag{4.14}$$

which provides a sixth-order estimation of $f_{i+1/2}$ for $\alpha = 3/10$, $a = 3/4$ and $b = 1/20$ (Nagarajan et al., 2003).

4.1.6 Initialization from a previous state

In preparation for the 4DVar implementation, we are led to restart a DNS from a pre-existing turbulent flow and the inflow computed or observed at each DNS time step. In order to ensure that the restart procedure after a given time t^k doesn't induce any errors and the flow doesn't diverge from its initial trajectory, Incompact3d generates the restart file `sauve.dat` att^k which is saves the following elements :

- each component of the velocity field at t^k,
- the the pressure field p^k at t^k,
- the gradient of the pressure field ∇p^k at t^k,
- and the previous convection and diffusion terms F^{k-1}, F^{k-2} necessary to the time advancement scheme (4.3).

4.2 Construction of Incompact3d discrete adjoint

In this chapter we implement the 4DVar technique described in § 2.4.5. The construction of the discrete adjoint corresponding to the Incompact3d code is the cornerstone of the 4DVar implementation. In this work, the discrete adjoint code was built with the help of the AD tool TAPENADE (Hascoët, 2004). The reader can find the description of the AD principles and the different implementation strategies in § C.1. We also provide in § C.1.2 an example of the detailed construction of the tangent and adjoint procedure corresponding to a simple program.

In the present chapter, we consider the complete DNS code Incompact3d which is structured into a sequence of subroutines as illustrated in figure 4.3.

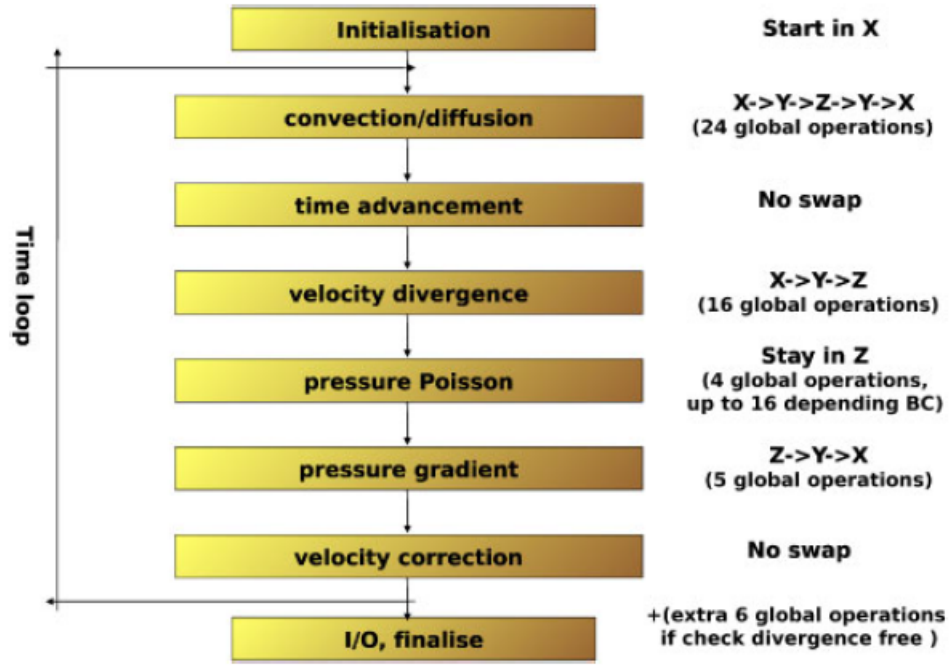

FIGURE 4.3 – General structure of the DNS code Incompact3d. The pencil configuration swaps carried out in each subroutine are specified in the right column.

In terms of the notations, we inherit from the formalism imposed by TAPENADE such that :

- the tangent procedures are ended by the subscript `_d`, the differential tangent variables associated to a given variable x are denoted $\dot{x}$,
- the adjoint procedures are ended by the subscript `_b`, the differential adjoint variables associated to a given variable x are denoted $\bar{x}$.

First and foremost, we present in § 4.2.1 the treatment of the parallelization routines involved in the code. We then describe in § 4.2.2, § 4.2.3, § 4.2.4, § 4.1.5 and § 4.2.6 the discrete adjoint procedures corresponding to each subroutine of the original program depicted in figure 4.3. Most of these procedures involve the first and/or second order derivatives which are defined by high order compact schemes. The latter are a algebraic combination of coefficients and values of the flow at given

points, therefore we deduce their corresponding discrete adjoint by a systematic application of TAPENADE. We refer the reader to appendix E. Finally, the complete discrete adjoint of Incompact3d is summarized in § 4.2.7. The tangent and adjoint procedures of each subroutine and the entire code was systematically verified by the validation procedures presented in § 4.2.8.

4.2.1 Discrete adjoint of parallel-MPI codes

Incompact3d and in particular the decomp2d library are based on MPI subroutines which need to be differentiated in the same way as the other subroutines of the code. The numerical gradient of MPI-codes have been investigated in the literature by Faure and Dutto (1999), Utke et al. (2009) and Schanen et al. (2010). We base the following description on Utke et al. (2012).

In the one hand, we consider the global communicators which carry out operations that involve all the data within all the processors. For instance, the global communicator `MPI_allreduce`(x,y,...,SUM,P,...) sums the values of `x` from each processor into the global variable `y` in the processor `P`. The procedure adjoint corresponding to the global communicators are summarized in figure 4.4, where `P` denotes a specific processor to which the result is sent to for `MPI_bcast`, `MPI_reduce`, `MPI_gather` and `MPI_scatter`.

Original	Adjoint
`MPI_bcast`(x, ... ,P, ...)	`MPI_reduce` ($\bar{x}$, $\bar{t}$, ... , SUM, P, ...) $\bar{x} = 0.0$; on `P` : $\bar{x} = \bar{x} + \bar{t}$
`MPI_reduce`(x,y, ... ,SUM, P, ...)	on P : $\bar{t} = \bar{y}$; on P : $\bar{y} = 0.0$ `MPI_bcast` ($\bar{t}$, ... , P, ...) ; $\bar{x} = \bar{x} + \bar{t}$
`MPI_allreduce`(x,y, ... ,SUM, ...)	`MPI_allreduce` ($\bar{y}$, $\bar{t}$, ... , SUM, ...) $\bar{y} = 0.0$; $\bar{x} = \bar{x} + \bar{t}$
`MPI_gather`(x,y, ... ,P, ...)	`MPI_scatter` ($\bar{y}$, ... ,$\bar{t}$, P, ...) on P : $\bar{y} = 0.0$; $\bar{x} = \bar{x} + \bar{t}$
`MPI_scatter`(y, ... ,t, P, ...)	`MPI_gather` ($\bar{y}$, ... ,$\bar{t}$(:), P, ...) on $\bar{y} = 0.0$; on P : $\bar{x} = \bar{x} + \bar{t}$(:)

FIGURE 4.4 – Discrete adjoints of global MPI communicators

On the other hand, the point-to-point communicators consists in sending or receiving the data between two given processes. The reader can find in Utke et al. (2009) the adjoining of point-to-point communication. In our context, these communicators are exclusively used within the decomp2d library to swap from a pencil to another (Li and Laizet, 2010). It is shown that there exists a duality between forward and adjoint MPI communication routines for several applications (Faure and Dutto, 1999; Cheng, 2006). This duality can be extended in our context with the swap of a configuration to another. In other words, the adjoint of the swap procedure `transpose_x_to_y` which swaps from the X-pencil to the Y-pencil is given by

`transpose_y_to_x` which swaps from the Y-pencil to the X-pencil. By analogy, the adjoint of the swap procedure `transpose_y_to_z` is given by `transpose_z_to_y`.

For the sake of clarity, we avoid to mention the swap procedures in the following sections and we assume that each operation performs the adequate forward and backward swap before each procedure.

4.2.2 Discrete adjoint of the convection and diffusion terms

Discrete adjoint of the convective term

The convective term, namely $\frac{1}{2}\left(\nabla(\mathbf{u}\otimes\mathbf{u}) + (\mathbf{u}\nabla)\mathbf{u}\right)$, requires the knowledge of the first order derivatives of the velocity field $\mathbf{u}$, denoted $\texttt{derx}(\mathbf{u})$, $\texttt{dery}(\mathbf{u})$ and $\texttt{derz}(\mathbf{u})$. Given these derivatives, the computation of the convective term, summarized into the vector $\mathbf{conv} = (conv_x, conv_y, conv_z)$, with respect to the velocity field $\mathbf{u} = (u, v, w)$, is given by algorithm 17.

Algorithm 17 $\texttt{conv}(u,v,w,conv_x,conv_y,conv_z)$

$t_x = u.u$
$t_y = u.v$
$t_z = u.w$
$conv_x = 0.5(\texttt{derx}(t_x)+\texttt{dery}(t_y)+\texttt{derz}(t_z)) + 0.5(u.\texttt{derx}(u) + v.\texttt{dery}(u) + w.\texttt{derz}(u))$
$t_x = v.u$
$t_y = v.v$
$t_z = v.w$
$conv_y = 0.5(\texttt{derx}(t_x)+\texttt{dery}(t_y)+\texttt{derz}(t_z)) + 0.5(u.\texttt{derx}(v) + v.\texttt{dery}(v) + w.\texttt{derz}(v))$
$t_x = w.u$
$t_y = w.v$
$t_z = w.w$
$conv_z = 0.5(\texttt{derx}(t_x)+\texttt{dery}(t_y)+\texttt{derz}(t_z)) + 0.5(u.\texttt{derx}(w) + v.\texttt{dery}(w) + w.\texttt{derz}(w))$

We detail the construction of the adjoint for the first component $conv_x$ the other components $conv_y$ and $conv_z$ are then deduced by analogy.

The tangent of the intermediate terms t_x, t_y, t_z is straightforward and reads

$$\begin{pmatrix} \dot{t}_x \\ \dot{t}_y \\ \dot{t}_z \end{pmatrix} = \begin{pmatrix} 2u & 0 & 0 \\ v & u & 0 \\ w & 0 & u \end{pmatrix} \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}. \tag{4.15}$$

The adjoint is simply deduced by transposing the previous matrix such as

$$\begin{pmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \end{pmatrix} = \begin{pmatrix} 2u & v & w \\ 0 & u & 0 \\ 0 & 0 & u \end{pmatrix} \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \\ \bar{t}_z \end{pmatrix}. \tag{4.16}$$

We assume that we know beforehand the tangent procedures `derx_d`, `dery_d` and `derz_d`, respectively corresponding to each derivative subroutines `derx`, `dery` and `derz`, and we denote their associating Jacobian matrix as D_x, D_y and D_z. Thus, given (4.15), we deduce the tangent of $\dot{conv}$:

$$\begin{aligned}\dot{conv}_x =& \frac{1}{2}\begin{pmatrix} D_x & D_y & D_z \end{pmatrix}\begin{pmatrix} 2u & 0 & 0 \\ v & u & 0 \\ w & 0 & u \end{pmatrix}\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} + \\ & \frac{1}{2}\begin{pmatrix} \text{derx}(u) & \text{dery}(v) & \text{derz}(w) \end{pmatrix}\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} u.D_x & v.D_y & w.D_z \end{pmatrix}\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}.\end{aligned}$$

Given the transposed of the Jacobian matrices D_x^T, D_y^T and D_z^T and (4.16), we deduce the expression of the adjoint variable $\overline{conv}_x$

$$\begin{aligned}\begin{pmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \end{pmatrix} =& \frac{1}{2}\begin{pmatrix} 2u & v & w \\ 0 & u & 0 \\ 0 & 0 & u \end{pmatrix}\begin{pmatrix} D_x^T \\ D_y^T \\ D_z^T \end{pmatrix}\overline{conv}_x + \\ & \frac{1}{2}\begin{pmatrix} \text{derx}(u) \\ \text{dery}(v) \\ \text{derz}(w) \end{pmatrix}\overline{conv}_x + \frac{1}{2}\begin{pmatrix} u.D_x^T \\ v.D_y^T \\ w.D_z^T \end{pmatrix}\overline{conv}_x.\end{aligned}$$

The discrete adjoint of each convective term $conv_y$ and $conv_y$ can easily be deduced by analogy. The discrete adjoint of the convection procedure is finally presented in algorithm 18.

Discrete adjoint of the diffusion term

The diffusion term, namely $-\nu\nabla^2\mathbf{u}$, requires the knowledge of the second order derivatives of the velocity field $\mathbf{u}$, determined by the subroutines `derxx`($\mathbf{u}$), `deryy`($\mathbf{u}$) and `derzz`($\mathbf{u}$) respectively. The computation of the diffusion vector $\mathbf{diff} = (diff_x, diff_y, diff_z)$ is given by algorithm 19.

Similarly to the previous case, we assume that we know beforehand the tangent and discrete adjoint associated to the second order derivative subroutine. Thus the adjoint of the the diffusive terms are given by algorithm 20. Let us note although this operator is formally auto-adjoint, the discretization of its continous adjoint doesn't respect this property. Thus the discrete adjoint has to be specified by AD.

Discrete adjoint of the convection and diffusion terms

Given the previous routines, the convection and diffusion terms are summed into the vector $F := (F_x, F_y, F_z) = \mathbf{conv} + \mathbf{diff}$. We easily deduce the discrete adjoint `convdiff_b` summarized in algorithm 22.

Algorithm 18 `conv_b`$(u,\bar{u},v,\bar{v},w,\bar{w},conv_x,\overline{conv}_x,conv_y,\overline{conv}_y,conv_z,\overline{conv}_z)$

Initialization: $\bar{u}=0$, $\bar{v}=0$, $\bar{w}=0$

! Forward procedure

call `conv`$(u,v,w,conv_x,conv_y,conv_z)$

! Background procedure

$tmp1 =$ `derx_b`$(conv_x, \overline{conv}_x)$

$tmp2 =$ `dery_b`$(conv_y, \overline{conv}_x)$

$tmp3 =$ `derz_b`$(conv_z, \overline{conv}_x)$

$\bar{u} = \bar{u} + 0.5(2u.tmp1 + v.tmp2 + w.tmp3) + 0.5\ \mathrm{derx}(u).\overline{conv}_x + 0.5.u.tmp1$

$\bar{v} = \bar{v} + 0.5u.tmp2 + 0.5\ \mathrm{dery}(v).\overline{conv}_x + 0.5.v.tmp2$

$\bar{w} = \bar{w} + 0.5u.tmp3 + 0.5\ \mathrm{derz}(w).\overline{conv}_x + 0.5.w.tmp3$

$tmp4 =$ `derx_b`$(conv_x, \overline{conv}_y)$

$tmp5 =$ `dery_b`$(conv_y, \overline{conv}_y)$

$tmp6 =$ `derz_b`$(conv_z, \overline{conv}_y)$

$\bar{u} = \bar{u} + 0.5v.tmp4 + 0.5\ \mathrm{derx}(u).\overline{conv}_y + 0.5.u.tmp4$

$\bar{v} = \bar{v} + 0.5(u.tmp4 + 2v.tmp5 + w.tmp6) + 0.5\ \mathrm{dery}(v).\overline{conv}_y + 0.5.v.tmp5$

$\bar{w} = \bar{w} + 0.5v.tmp6 + 0.5\ \mathrm{derz}(w).\overline{conv}_y + 0.5.w.tmp6$

$tmp7 =$ `derx_b`$(conv_x, \overline{conv}_z)$

$tmp8 =$ `dery_b`$(conv_y, \overline{conv}_z)$

$tmp9 =$ `derz_b`$(conv_z, \overline{conv}_z)$

$\bar{v} = \bar{u} + 0.5w.tmp7 + 0.5\ \mathrm{derx}(u).\overline{conv}_z + 0.5.u.tmp7$

$\bar{w} = \bar{v} + 0.5w.tmp8 + 0.5\ \mathrm{dery}(v).\overline{conv}_z + 0.5.v.tmp8$

$\bar{u} = \bar{u} + 0.5(u.tmp7 + v.tmp8 + 2w.tmp9) + 0.5\ \mathrm{derz}(w).\overline{conv}_z + 0.5.w.tmp9$

Algorithm 19 `diff`$(u,v,w,diff_x,diff_y,diff_z)$

Initialization: u, v, w

$diff_x = -\nu($`derxx`$(u)+$`deryy`$(u)+$`derzz`$(u))$

$diff_y = -\nu($`derxx`$(v)+$`deryy`$(v)+$`derzz`$(v))$

$diff_z = -\nu($`derxx`$(w)+$`deryy`$(w)+$`derzz`$(w))$

4.2.3 Discrete adjoint of the time integration scheme

We recall the general formulation of the time advancement procedure described in § 4.1.3,

$$\mathbf{u}^* = \mathbf{u}^k + a_k \Delta t F^k + b_k \Delta t F^{k-1} + c_k \Delta t F^{k-2},$$

where F^k is the result of the `convdiff` routine. The time integration scheme can be written into the following matrix form

$$\begin{pmatrix} u^* \\ F^k \\ F^{k-1} \\ F^{k-2} \end{pmatrix} = \begin{pmatrix} 1 & a_k\Delta t & b_k\Delta t & c_k\Delta t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u^k \\ F^k \\ F^{k-1} \\ F^{k-2} \end{pmatrix}. \tag{4.17}$$

Algorithm 20 `diff_b`$(u,\bar{u},v,\bar{v},w,\bar{w},diff_x,\overline{diff}_x,\ diff_y,\ \overline{diff}_y,diff_z,\overline{diff}_z)$

Initialization: $\bar{u}=0,\ \bar{v}=0,\ \bar{w}=0$
! Forward procedure
Initialization: u,v,w
call `diff`$(u,v,w,diff_x,diff_y,diff_z)$
! Background procedure
$\bar{u}=\bar{u}+$(`derxx_b`$(-\nu.diff_x,-\nu.\overline{diff}_x)$
$\bar{u}=\bar{u}+$`deryy_b`$(-\nu.diff_x,-\nu.\overline{diff}_x)$
$\bar{u}=\bar{u}+$`derzz_b`$(-\nu.diff_x,-\nu.\overline{diff}_x)$
$\bar{v}=\bar{v}+$(`derxx_b`$(-\nu.diff_y,-\nu.\overline{diff}_y)$
$\bar{v}=\bar{v}+$`deryy_b`$(-\nu.diff_y,-\nu.\overline{diff}_y)$
$\bar{v}=\bar{v}+$`derzz_b`$(-\nu.\overline{diff}_y,-\nu.\overline{diff}_y)$
$\bar{w}=\bar{w}+$(`derxx_b`$(-\nu.diff_z,-\nu.\overline{diff}_z)$
$\bar{w}=\bar{w}+$`deryy_b`$(-\nu.diff_z,-\nu.\overline{diff}_z)$
$\bar{w}=\bar{w}+$`derzz_b`$(-\nu.diff_z,-\nu.\overline{diff}_z)$

Algorithm 21 `convdiff`(u,v,w,F_x,F_y,F_z)

call `conv`$(u,v,w,conv_x,conv_y,conv_z)$
call `diff`$(u,v,w,diff_x,diff_y,diff_z)$
$F_x=conv_x+diff_x$
$F_y=conv_y+diff_y$
$F_z=conv_z+diff_z$

Algorithm 22 `convdiff_b`$(u,\bar{u},v,\bar{v},w,\bar{w},F_x,\bar{F}_x,\ F_y,\ \bar{F}_y,F_z,\bar{F}_z)$

Initialization: $\bar{u}=0,\ \bar{v}=0,\ \bar{w}=0$
! Forward procedure
call `conv`$(u,v,w,conv_x,conv_y,conv_z)$
call `diff`$(u,v,w,diff_x,diff_y,diff_z)$
! Backward procedure
$\overline{conv}_x=\bar{F}_x\,;\ \overline{conv}_y=\bar{F}_y\,;\ \overline{conv}_z=\bar{F}_z$
$\overline{diff}_x=\bar{F}_x\,;\ \overline{diff}_y=\bar{F}_y\,;\ \overline{diff}_z=\bar{F}_z$
call `diff_b`$(u,\bar{u},v,\bar{v},w,\bar{w},diff_x,\overline{diff}_x,\ diff_y,\ \overline{diff}_y,diff_z,\overline{diff}_z)$
$tmp1=\bar{u}\,;\ tmp2=\bar{v}\,;\ tmp3=\bar{w}$
call `conv_b`$(u,\bar{u},v,\bar{v},w,\bar{w},conv_x,\overline{conv}_x,conv_y,\overline{conv}_y,conv_z,\overline{conv}_z)$
$\bar{u}=\bar{u}+tmp1\,;\ \bar{v}=\bar{v}+tmp2\,;\ \bar{w}=\bar{w}+tmp3$

The corresponding adjoint procedure is then simply deduced by transposing the latter matrix such that

$$\begin{pmatrix}\bar{u}^k\\ \bar{F}^k\\ \bar{F}^{k-1}\\ \bar{F}^{k-2}\end{pmatrix}=\begin{pmatrix}1 & 0 & 0 & 0\\ a_k\Delta t & 1 & 0 & 0\\ b_k\Delta t & 0 & 1 & 0\\ c_k\Delta t & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\bar{u}^*\\ \bar{F}^k\\ \bar{F}^{k-1}\\ \bar{F}^{k-2}\end{pmatrix}. \tag{4.18}$$

Therefore, the generalized form of the discrete adjoint of the time integration scheme is given by algorithm 24.

Algorithm 23 intt_b($\mathbf{u}^k$,$\bar{\mathbf{u}}^k$, $\bar{F}^k$, $\bar{F}^{k-1}$, $\bar{F}^{k-2}$)

Initialization: $F^k = 0$, $F^{k-1} = 0$ and $F^{k-2} = 0$
$\bar{F}^k = \bar{F}^k + a_k \Delta t \bar{\mathbf{u}}^k$
$\bar{F}^{k-1} = F^{k-1} + b_k \Delta t \bar{\mathbf{u}}^k$
$\bar{F}^{k-2} = F^{k-2} + c_k \Delta t \bar{\mathbf{u}}^k$

4.2.4 Discrete adjoint of the velocity divergence

At this point, we proceed to several operations before computing the pressure field. First and foremost, we impose the boundary conditions on each component of the velocity field and in each directions. A pre-correction treatment is applied to ensure the proper application of the boundary conditions. Finally, the velocity divergence $\nabla \cdot \mathbf{u}$ required to solved the Poisson equation is developped (see § 4.2.5). All these operations are gathered under the designation of velocity divergence in the structure of the code (see figure 4.3). As the velocity divergence $\nabla \cdot \mathbf{u}$ is computed by an elementary combination of the derivative routines, we omit here its description and refer the reader to appendix E. Thus, the present section describes the discrete adjoint of the boundary conditions and the pre-correction routine.

Boundary conditions

In terms of boundary conditions, we constructed the discrete adjoint corresponding specifically to the cylinder wake flow configuration studied within this chapter. In this work, we impose the following boundary conditions model :

- periodic condition along the z-axis,
- free slip condition along the y-axis,
- Dirichlet condition for the inflow and
- a first order advection model for the outflow.

As discussed in § 4.1, the periodic and free-slip conditions are implicitly imposed by the numerical schemes used in the first and second order derivatives subroutines. As the inflow is a control parameter of the 4DVar assimilation technique, we refer the reader to § 2.4.5 for its adjoint. Finally, we recall the first order transport model in the stream-wise direction employed to model the outlet boundary condition,

$$\mathbf{u}_{nx}^{BC} = \mathbf{u}_{nx}^* - c_x(\mathbf{u}_{nx}^* - \mathbf{u}_{nx-1}^*), \tag{4.19}$$

where $\mathbf{u}^*$ is the result of the time integration scheme (4.3) and $c_x = g_k \frac{\Delta t}{\Delta x} u_{nx}^*$ is set such as the CFL condition is verified.

The tangent of outflow model with respect to the differential variables $\dot{\mathbf{u}}^*$ can be written into the following matrix form

$$\begin{pmatrix} \dot{\mathbf{u}}_{nx}^{BC} \\ \dot{\mathbf{u}}_{nx-1}^{BC} \end{pmatrix} = \begin{pmatrix} 1 - c_x & c_x & \mathbf{u}_{nx}^* - \mathbf{u}_{nx-1}^* \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{u}}_{nx}^* \\ \dot{\mathbf{u}}_{nx-1}^* \\ \dot{c}_x \end{pmatrix}.$$

The corresponding adjoint procedure is obtained by transposing the previous matrix :

$$\begin{pmatrix} \bar{\mathbf{u}}^*_{nx} \\ \bar{\mathbf{u}}^*_{nx-1} \\ \bar{c}_x \end{pmatrix} = \begin{pmatrix} 1-c_x & 0 \\ c_x & 1 \\ \mathbf{u}^*_{nx}-\mathbf{u}^*_{nx-1} & 0 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{u}}^{BC}_{nx} \\ \bar{\mathbf{u}}^{BC}_{nx-1} \end{pmatrix}.$$

The adjoint procedure of the outflow boundary is given by algorithm 24.

Algorithm 24 outflow_b($\bar{u}^*$, $\bar{v}^*$, $\bar{w}^*$,$\bar{u}^{BC}$, $\bar{v}^{BC}$, $\bar{w}^{BC}$)

Initialization: $\bar{u}_{nx} = 0$, $\bar{u}^*_{nx-1} = 0$, $\bar{v}^*_{nx} = 0$, $\bar{v}^*_{nx-1} = 0$, $\bar{w}^*_{nx} = 0$, $\bar{w}^*_{nx-1} = 0$,

! Forward procedure

$c_x = g_k \frac{\Delta t}{\Delta x} u_{nx}$

! Background procedure

$\bar{u}^*_{nx} = \bar{u}^{BC}_{nx} + c_x \bar{u}^{BC}_{nx}$

$\bar{v}^*_{nx} = \bar{v}^{BC}_{nx} + c_x \bar{v}^{BC}_{nx}$

$\bar{w}^*_{nx} = \bar{w}^{BC}_{nx} + c_x \bar{w}^{BC}_{nx}$

$\bar{c}_x = \bar{c}_x - (u_{nx} - u_{nx-1})\bar{u}_{nx} - (v_{nx} - v_{nx-1})\bar{v}_{nx} - (w_{nx} - w_{nx-1})\bar{w}_{nx}$

$\bar{u}^*_{nx-1} = \bar{u}^{BC}_{nx-1} + c_x \bar{u}^{BC}_{nx}$

$\bar{v}^*_{nx-1} = \bar{v}^{BC}_{nx-1} + c_x \bar{v}^{BC}_{nx}$

$\bar{w}^*_{nx-1} = \bar{w}^{BC}_{nx-1} + c_x \bar{w}^{BC}_{nx}$

$\bar{u}^*_{nx} = \bar{u}^{BC}_{nx} + g_k \frac{\Delta t}{\Delta x} \bar{c}_x$

Pre-correction

We perform the pre-correction on the boundaries for each component of the velocity field (4.4) recalled below,

$$\mathbf{u}^{**} = \mathbf{u}^* + g_k \Delta t \nabla \tilde{p}^k,$$

where $\mathbf{u}^*$ and $\nabla \tilde{p}^k$ is the velocity field and the average of the pressure gradient respectively and computed at the previous time step. This pre-correction ensures that the boundary condition model is effectively applied after the correction of the pressure gradient deduced by solving the Poisson equation (see § 4.2.6). The discrete adjoint corresponding to this pre-correction is straightforward and reads :

$$\begin{aligned} \bar{\mathbf{u}}^* &= \bar{\mathbf{u}}^* + \bar{\mathbf{u}}^{**}, \\ \overline{\nabla p}^k &= \overline{\nabla p}^k + g_k \Delta t \bar{\mathbf{u}}^{**}. \end{aligned}$$

A second correction on the outflow is performed to ensure that the outflow rate corresponds to the inflow rate. This correction is an elementary operation which requires the use of a global communicator `MPI_AllReduce` in order to compute the sum of the flow on the entire plane. The adjoint associated to routine is described in § 4.2.1.

4.2.5 Discrete adjoint of the Poisson solver

As described in § 4.1.5, the incompressibility condition is verified by solving a Poisson equation,

$$\nabla\cdot\nabla\tilde{p}^{k+1} = \frac{\nabla.\mathbf{u}^{**}}{g_k\Delta t},$$

in the spectral space. In this work, we choose to construct the discrete adjoint of the generic FFT developed by Li and Laizet (2010) which is based on the derivation of Glassmans's general N Fast Fourier Transform (Ferguson Jr, 1979). The FFT routine denoted `SPCFFT` can be found in http://www.jjj.de/fft/glassman-fft.f. The construction of the adjoint procedure corresponding to `SPCFFT` revealed that the routine is autoadjoint in the numerical sense which eases the implementation of the adjoint operator.

The `poisson` routine takes as input the right hand side of the Poisson equation, denoted rhs, and outputs the pressure field p. The procedure is described by algorithm 25. The wave number $kxyz$ is determined by the auxilary procedure `waves`, as it doesn't depend on an active variable of the routine, there is no need to compute its adjoint. The routine `poisson` also depends on two auxiliary routines `assemble`, `disassemble` and `FFT_postproc`. The routines `assemble` and `disassemble` are conjugate routines which simply performs a redistribution of the element of the input matrix. In other words, the discrete adjoint of `assemble` corresponds to `disassemble` and vice versa. The routine `FFT_postproc` performs a one dimensional post processing applied in every direction. The reader can find in appendix F the construction of its corresponding discrete adjoint.

Algorithm 25 `poisson`(rhs,p)

Initialization: call `waves`($kxyz$)
`assemble`(rhs)
`FFT`(rhs,cw,1)
! Post processing in the spectral space
call `FFT_postproc`(cw,cwp,1)
! Solve the poisson equation
$tmp1 = real(kxyz)$; $tmp2 = aimag(kxyz)$
cwp = cmplx$(-cwp/tmp1, -cwp/tmp2)$
! Backward post processing in the spectral space
call `FFT_postproc`(cwp,$cwpp$,-1)
`FFT`(p,$cwpp$,-1)
`assemble`(p)

Given the discrete adjoint of the auxiliary routines, the discrete adjoint corresponding to the `poisson` routine is deduced by algorithm 26.

Algorithm 26 `poisson_b`($\overline{rhs}$,$\bar{p}$)

Initialization: call `waves`($kxyz$)
`disassemble`($\bar{p}$)
`FFT`($\bar{p}$,$\overline{cwpp}$,-1)
! Post processing in the spectral space
call `FFT_postproc_b`($\overline{cwpp}$,$\overline{cwp}$,-1)
! Solve the poisson equation
$tmp1 = real(kxyz)$; $tmp2 = aimag(kxyz)$
$cw1 = \text{cmplx}(-\overline{cwp}/tmp1, -\overline{cwp}/tmp2)$
! Backward post processing in the spectral space
call `FFT_postproc_b`($\overline{cwp}$,$\overline{cw}$,1)
`FFT`($\overline{rhs}$,$\overline{cw}$,1)
`assemble`($\overline{rhs}$)

4.2.6 Discrete adjoint of the pressure gradient and velocity correction

The pressure gradient ∇p^{k+1}, computed by the subroutine `gradp`, is determined by sixth-order finite difference schemes described in § 4.1.5. The numerical schemes are linear combinations of the value of the pressure at different position of the staggered grid. Therefore, the corresponding adjoint procedure `gradp_b` can be determined straightforwardly by using TAPENADE.

We recall the expression of the final correction (4.5) of the velocity field with the pressure gradient computed previously

$$\mathbf{u}^{k+1} = \mathbf{u}^{**} - g_k \Delta t \nabla \tilde{p}^{k+1},$$

its discrete adjoint $\bar{\mathbf{u}}^{k+1}$, implemented to the `corgp_b` routine, is straightforwardly deduced by :

$$\begin{aligned} \bar{\mathbf{u}}^{**} &= \bar{\mathbf{u}}^{**} + \bar{\mathbf{u}}^{k+1} \\ \nabla \bar{p}^{k+1} &= \nabla \bar{p}^{k+1} + -g_k \Delta t \nabla \tilde{\bar{u}}^{k+1}. \end{aligned}$$

4.2.7 Discrete adjoint of the main program

All in all, the discrete adjoint of the original program Incompact3d is summarized in algorithm 25.

4.2.8 Code validation

We applied each validation test presented in § 2.4.7 to each adjoint routine built previously. We show in this section the results run on the discrete adjoint `Incompact3d_b` (see algorithm 27) corresponding to the original Incompact3d code.

Algorithm 27 `Incompact3d_b` ($\mathbf{u}^0$,$\bar{\mathbf{u}}^0$)

Initialization: $k = N$,$\mathbf{u}^0$,$\bar{\mathbf{u}}^N = 0$, $\bar{F}^{N-1} = 0$, $\bar{F}^{N-2} = 0$
  while k>0 **do**
    `corgp_b`($\mathbf{u}^k$,$\bar{\mathbf{u}}^k$,∇p,$\nabla \bar{p}$)
    `gradp_b`(∇p,$\nabla \bar{p}$,p,$\bar{p}$)
    `poisson_b`(p,$\bar{p}$,$\mathbf{u}^{**}$,$\bar{\mathbf{u}}^{**}$)
    `pre_correc_b`(p,$\bar{p}$)
    `intt_b`($\mathbf{u}^*$,$\bar{\mathbf{u}}^*$,F^{k-1},$\bar{F}^{k-1}$,F^{k-2},$\bar{F}^{k-2}$)
    `convdiff_b`($\mathbf{u}^{k-1}$,$\bar{\mathbf{u}}^{k-1}$,F^{k-1},$\bar{F}^{k-1}$,F^{k-2},$\bar{F}^{k-2}$)
    $k = k - 1$
  end while

Tangent process validation

We recall the tangent validation procedure for the operator f, which represents in this case the original code `Incompact3D`, and its corresponding tangent denoted ∇f,

$$R_1 := \frac{f(\mathbf{x}_0 + \alpha d) - f(\mathbf{x}_0)}{\alpha < \nabla f(\mathbf{x}_0), d >}. \tag{4.20}$$

We evaluate the ratio R_1 for different differentiation step α over a temporal window of $10\Delta t'$ then $100\Delta t'$. The results are summarized in table 4.2. The tangent code accuracy meets our expectation as the ratio R_1 is close to 1 up to $10^{-\alpha}$. We obtain the same accuracy for both temporal windows.

α	R_1 for 10 iterations	R_1 for 100 iterations
10^{-3}	1.0006899708371482	1.0000100833054026
10^{-4}	1.0000689567331724	1.0000001285470035
10^{-5}	1.0000068952890127	1.0000000040572325
10^{-6}	1.0000006894438878	1.0000000003169462
10^{-7}	1.0000000695308335	1.0000000000302249
10^{-8}	1.0000000375275666	0.99999999989057253
10^{-9}	1.0000000677486891	0.99999999908904569

TABLE 4.2 – Tangent validation results for different values of the differentiation step α over $10\Delta t'$ and $100\Delta t'$.

Dot product validation

By definition, the discrete adjoint procedure must correspond exactly to the previous tangent procedure associated to original code. Therefore, as the validation test is carried out in FORTRAN double precision ($O(10^{-14})$, we expect the result to reach the same accuracy. We computed the following ratio R_2 for $50\Delta t'$ and obtained

the following result which met our expectations,

$$R_2 = \frac{<(\partial_X \mathbb{M})X, (\partial_X \mathbb{M})X>}{<X, (\partial_X \mathbb{M}^{\#})Y>} = \frac{<\dot{Y}, \dot{Y}>}{<\dot{X}, \bar{Y}>} = 0.99999999999998990. \tag{4.21}$$

Adjoint process validation

Finally, we evaluated the validity of the entire 4DVar assimilation code by computing the following ratio for several differential step α

$$R_3 = \frac{J(\mathbf{x}_0 + \alpha d) - J(\mathbf{x}_0)}{\alpha < \lambda(t_0), d>}. \tag{4.22}$$

This test was at first carried out at first in FORTRAN double precision, then in FORTRAN single precision which corresponds to the accuracy of the simulation and assimilation. In opposition to the previous validation test, we don't require the adjoint procedure to be close to the finite difference as the latter is a rough approximation of the cost function gradient. The results are summarized in table 4.3.

α	R_3 (single precision)	R_3 (double precision)
10^{-2}	0.43749039322927519	-27.578171220965167
10^{-3}	0.95755116373350546	-1.8429534822499196
10^{-4}	1.0093158427293105	0.72996059356742782
10^{-5}	1.0149894966177055	0.98725112730444675
10^{-6}	1.0135159015484660	1.0129792001335909
10^{-7}	1.2662584846733522	1.0155449210116843
10^{-8}	2.3891918787645077	1.0156921716925300
10^{-9}	20.272515591281547	1.0148236474889325

TABLE 4.3 – 4DVar validation result for different values of the differentiation step α in FORTRAN single and double precision.

4.3 Validation on a synthetic case with 3D observations

We consider an ideal synthetic experiment to apprehend the mechanism of the reconstruction of a 3D turbulent flow using the 4DVar code developed during this thesis. This experiment doesn't aim at reproducing any specific experimental setting, instead our goal is to assess the capacity of the code to correct a background flow with a set of 3D observations sparse in time and at the DNS spatial resolution. The experimental context and the synthetic reference flow are described in § 4.3.1. As it is unfeasible to perform the assimilation within the same domain as the reference flow, we investigate the consequence of running the assimilation within a smaller domain in § 4.3.2. The description of the synthetic observations and the initial background are given in § 4.3.3 and § 4.3.4, respectively. Finally, the results are presented in § 4.3.5.

4.3.1 Synthetic configuration

In this synthetic context, we consider a cylinder wake flow at $Re = 300$ as a reference flow. The latter is generated by running Incompact3d from an initial null velocity field within the reference domain $\Omega_{\text{ref}} = 20D \times 12D \times 6D$ descritized into $360 \times 217 \times 108$ nodes. The turbulent structures are induced by imposing an inflow velocity $u_{\text{in}}/U = 1 + \epsilon$ at each DNS time step $\Delta t' = \Delta t \cdot U/D = 0.005$. The cylinder is modeled by an Immersed Boundary Method (Parnaudeau et al., 2004, 2008). We impose free-slip conditions on the y-axis borders and periodic conditions along the cylinder axis. The outflow is modeled by a first-order advection model (see § 4.1). We obtain satisfying turbulent structures at $T'_{\text{ref}} = 34000\Delta t' = 170$ (see figure 4.5).

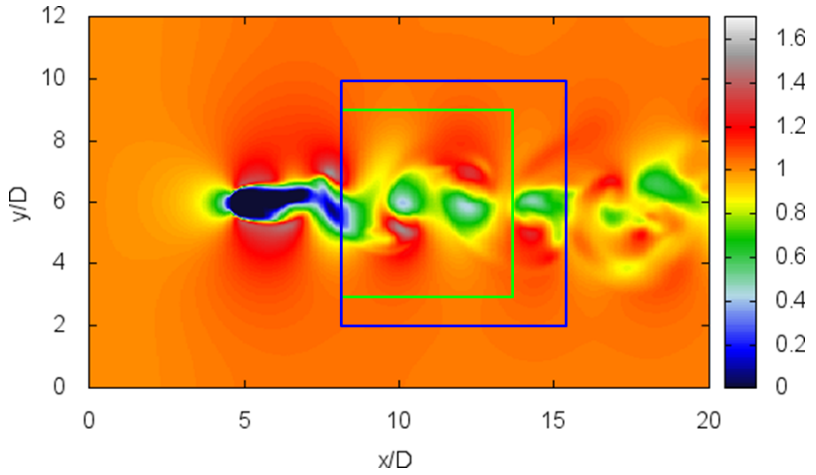

FIGURE 4.5 – Snapshot of the stream-wise velocity field u/U in the reference domain Ω_{ref}. The turbulent structures are obtained from an initial null velocity field and the input of an inflow velocity over $T'_{\text{ref}} = 34000\Delta t'$. The assimilation is run within the sub-domain Ω (in blue) and we perform statistics of the results in the the sub-domain Ω_{obs} (in green).

The temporal setup for the generation of a reference flow followed by the ap-

plication of the 4DVar is illustrated in figure 4.6. The characteristics of each run performed within the present synthetic configuration are summarized in table 4.4.

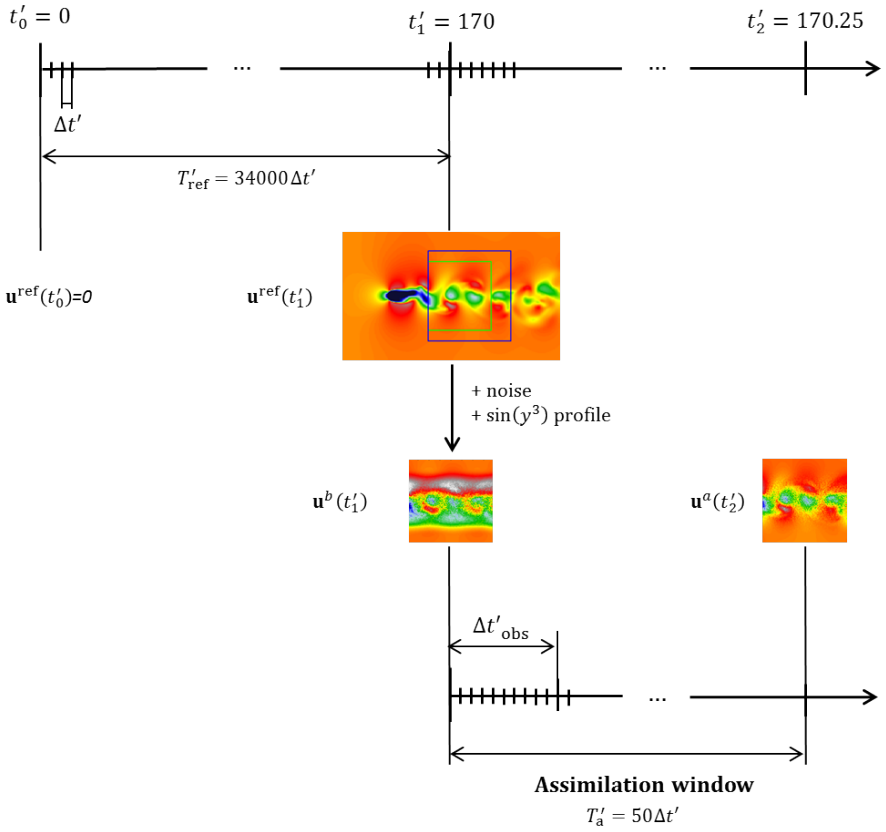

FIGURE 4.6 – Temporal setup for the reconstruction of a complete three dimensional flow with the 4DVar technique. In a first step we build the reference flow of this study by running a DNS over $T'_{\text{ref}} = 34000\Delta t' = 170$ within the reference domain Ω_{ref}. We then extract the three components of the velocity at $t'_1 = 170$ and add a 15% noise and a $\sin(y^3)$ profile. The latter is used to initialize the 4DVar code performed over the assimilation window $T'_a = 50\Delta t' = 0.25$, by injecting three dimensional synthetic observations every $\Delta t'_{\text{obs}} = 10\Delta t'$.

4.3.2 Initialization from a pre-existing turbulent flow

In regards to the computational resources, we clearly can't afford to run the 4DVar assimilation on the reference domain Ω_{ref}. Unfortunately, running the DNS on a smaller domain inevitably leads to incompatible boundary conditions models of the reference domain and the smaller domain. Furthermore, most of the experimental setups can't provide the quantities required for the restart procedure described in § 4.1.6 (pressure field and convection/diffusion terms at previous time steps). We assess in the present section the inherent errors induced by the model and the initialization with the components of the velocity field only.

We ran Incompact3d within the smallest sub-domain possible $\Omega = 8D \times 8D \times 6D$ which contains the turbulent structure, over a period of $500\Delta t' = 2.5$. This DNS

Run	$Lx \times Ly \times Lz$	$nx \times ny \times nz$	Re	Duration	CPU time (h)
Reference	$20D \times 12D \times 6D$	$361 \times 217 \times 108$	300	$30000\Delta t'$	50
4DVar	$8D \times 8D \times 6D$	$145 \times 145 \times 108$	300	$50\Delta t'$	5
4DVar-rampe	$8D \times 8D \times 6D$	$145 \times 145 \times 108$	300	$50\Delta t'$	5

TABLE 4.4 – Characteristics of each run performed to reconstruct the three dimensional flow from 3D synthetic observations. Each data assimilation cases considered a static background error covariance matrix $\mathbf{B}^{-1} = 1$. The 4DVar and 4DVar-rampe runs fix the observation error covariance matrix to $\mathbf{R}^{-1} = 1$ and to the formulation given by (4.23), (4.24) and (4.25), respectively.

is only initialized with the velocity field components directly extracted from the reference flow described in § 4.3.1, the pressure and gradient pressure are set to zero and we use of a first order temporal scheme at the start. We also input at each DNS time step $\Delta t'$ the exact inflow extracted from the reference. Figure 4.7 displays the difference between the reference and the simulated stream-wise velocity at $T'_{\text{ref}}+50\Delta t' = 170.25$ and $T'_{\text{ref}}+500\Delta t' = 172.5$. We see that the errors remain in a range of $(u-u_{\text{ref}})/U = [-0.01; 0.01]$ in time. These errors are clearly amplified after a long period. The errors at the inlet are likely induced by the lack of information of the gradient pressure ∇p^k which intervenes in the pre-correction step (4.4). The free slip and the outflow models also modify the dynamics within the sub-domain.

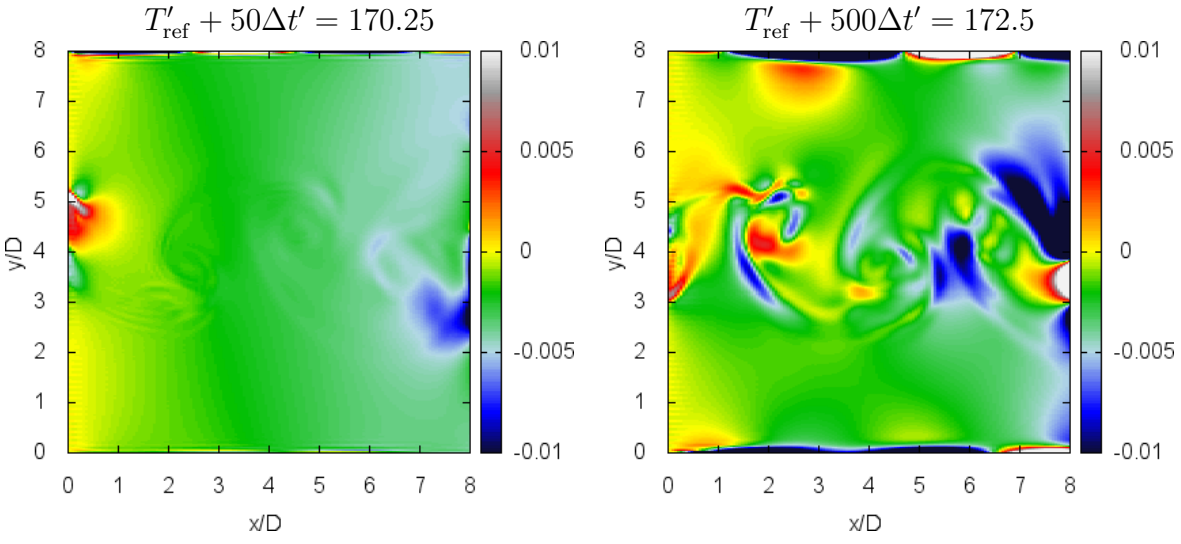

FIGURE 4.7 – Error between the reference and the simulated stream-wise velocity field at $T'_{\text{ref}} + 50\Delta t' = 170.25$ and $T'_{\text{ref}} + 500\Delta t' = 172.5$ at the stream-wise plane $z = 3D$.

In our context, we recall that we perform the 4DVar over a small assimilation window $T'_a = 50\Delta t'$ as illustrated in figure 4.6. Within the assimilation window, the errors are reasonably confined at the boarders and remain in a range of $[-0.01; 0.01]$.

A solution could consist in using sponge layers (Bogey et al., 2000) which will require the construction of the adjoint as suggested by Lemke and Sesterhenne (2013). However, as explained in § 4.3.3, for the sake of simplicity we chose to reduce boundary zone effects by tuning the observation error covariance matrix $\mathbf{R}$. Finally, we perform the statistics assessments within a smaller Ω_{obs}, illustrated in figure 4.5.

4.3.3 Synthetic observations

In this ideal synthetic configuration, we assume we possess a set of 3D synthetic observations every $\Delta t'_{\text{obs}} = 10\Delta t'$, the observation time step. These observations are directly extracted from the reference flow within the sub-domain Ω.

In light of the analysis made in § 4.3.2, the errors induced by the incompatibility of the boundary condition models over the assimilation window T'_a are confined within boundary zones (see figure 4.7). Therefore, we attempt to reduce the effect of these errors by tuning the observation error covariance matrix $\mathbf{R}$ within the sub-domain Ω. We run two assimilation cases :

Case a : we assume that we trust the observations which comes to impose $\mathbf{R}^{-1} = \mathbb{I}$ within the sub-domain Ω. This case is referred to as "4Dvar-all" in the following results.

Case b : we consider the incompatibilities between the model and the observations within the boundary zones. This case is referred to as "4Dvar-rampe" in the following results. We set the observation covariance matrix as a decreasing function depending on the distance to the boundaries such that

$$\mathbf{R}^{-1}(x,y,z) = diag\left(\exp\left(-\frac{(x-6D)^2}{2}\right)\right), x \in [6D; 8D], \tag{4.23}$$

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1} diag\left((x,y,z)\exp\left(-\frac{(y-1D)^2}{2}\right)\right), y \in [0D; 1D], \tag{4.24}$$

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1} diag\left((x,y,z)\exp\left(-\frac{(y-7D)^2}{2}\right)\right), y \in [7D; 8D]. \tag{4.25}$$

The resulting observation covariance matrix $\mathbf{R}^{-1}$ is illustrated in figure 4.8.

4.3.4 Background choice

In this purely synthetic application, we build a background initial condition which is likely to generate a divergent trajectory to the reference flow. Our main concern in this experiment is to assess the 4DVar code ability to retrieve the reference from an un-phased flow and few observations by correcting the initial state. We recall that we don't aim at reproducing a synthetic equivalent of a realistic experiment in this application.

As illustrated in figure 4.6, the background initial condition is deduced from the reference flow at t'_1 by adding a 15% white noise and a $\sin\left(\left(\frac{y}{D}\right)^3\right)$ profile bias along the y-axis. The comparison between the synthetic reference and the background initial stream-wise velocity fields is illustrated in figure 4.9. In this configuration, we fix the background error covariance matrix to $\mathbf{B} = 1$.

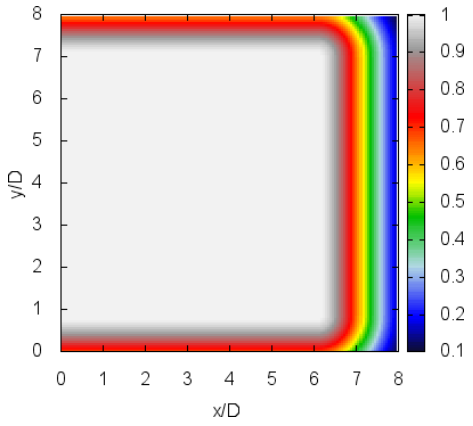

FIGURE 4.8 – Representation of the inverted observation covariance matrix $\mathbf{R}^{-1}$ at the stream-wise plane $z = 3D$. We reduce gradually the confidence within the boundary zones by the application of several 1D Gaussian profiles described by (4.23), (4.24) and (4.25).

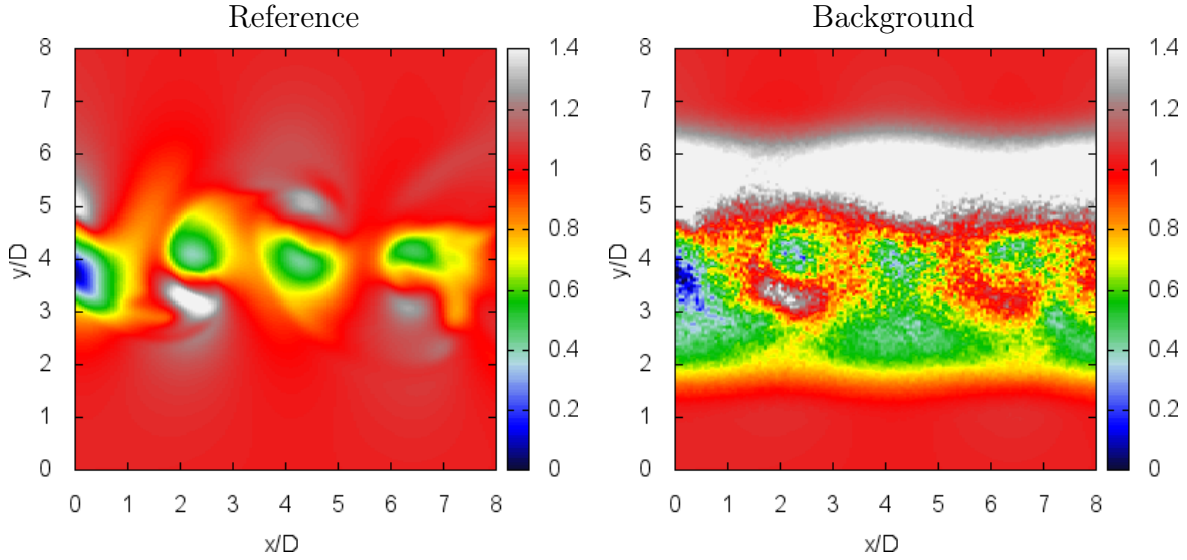

FIGURE 4.9 – Assimilation with 3D synthetic observations : comparison between the stream-wise velocity field of the reference flow (left) with the background flow (right) obtained by adding a 15% white noise and a $\sin(y^3)$ profile bias along the y-axis at $t'_1 = 170$.

4.3.5 Results

We assess the reconstruction of the velocity fields and compare each experimental case described in § 4.4.2. Both cases were carried out on a single assimilation window T'_a with a single outer loop iteration. Since the observation time step is small ($\Delta t'_{\text{obs}} = 10\Delta t'$), there are few changes between two consecutive observations. Besides, the observed inlet corresponds to the reference, thus the inlet correction didn't bring much improvements to the analysis flow. Therefore, we only assess the correction brought by the assimilation to the initial condition at t'_1.

Optimization characteristics

First and foremost, we compare the evolution of the cost function $J(\delta\gamma_0)$ and the norm of its gradient $\|\nabla J(\delta\gamma_0)\|$ with respect to the number of inner loop iterations. Figure 4.29 shows that both cases have a similar convergence rate of the cost function. In the one hand, the case a performs more inner loop iterations and reach a lower cost function evaluation than in case b. In the other hand, the case b found the correct descent gradients most of the time. We assume that the observation covariance matrix guided the case b in the right path.

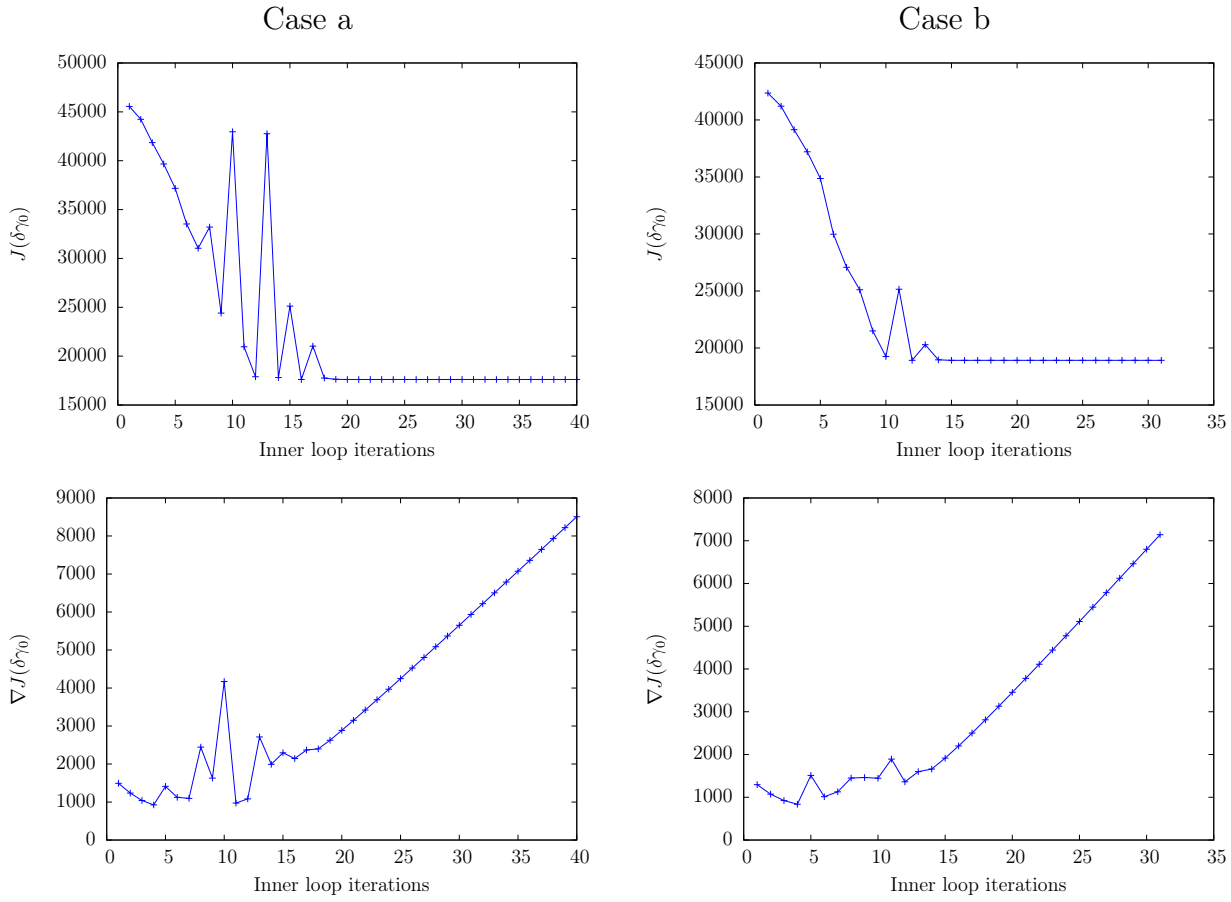

FIGURE 4.10 – Assimilation with 3D synthetic observations : comparison of the evolution of the cost function and cost function gradient for the case a (on the left) and the case b (on the right) with respect to the inner loop optimization iterations.

RMSE comparison

We study the evolution of the Root Mean Square Error (RMSE) defined for a given state $\mathbf{x}$ with respect to a reference state $\mathbf{x}_{\text{ref}}$ such that,

$$RMSE(\mathbf{x}) = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x} - \mathbf{x}_{ref}), \tag{4.26}$$

where N is the size of the state of interest $\mathbf{x}$. We compute and compare the RMSE for each component of the velocity field within the sub-domain Ω. The RMSE provides a global estimation of the correction of the large scale structures.

We compare in figure 4.11 the RMSE evolution of the background, observation and the analysis results in both cases, with respect to the reference flow over the data assimilation window $T'_a = 50\Delta t'$. Each RMSE was computed within the sub-domain Ω_{obs}. We recall that case a and case b correspond to the labels "4DVar-all" and "4DVar-rampe", respectively. In both cases, the data assimilation performed few corrections of the initial state and the solution tends to get closer to the reference in time. Both cases provide very similar results, we note that the case b provides the closest solution to the reference flow for the components v and w.

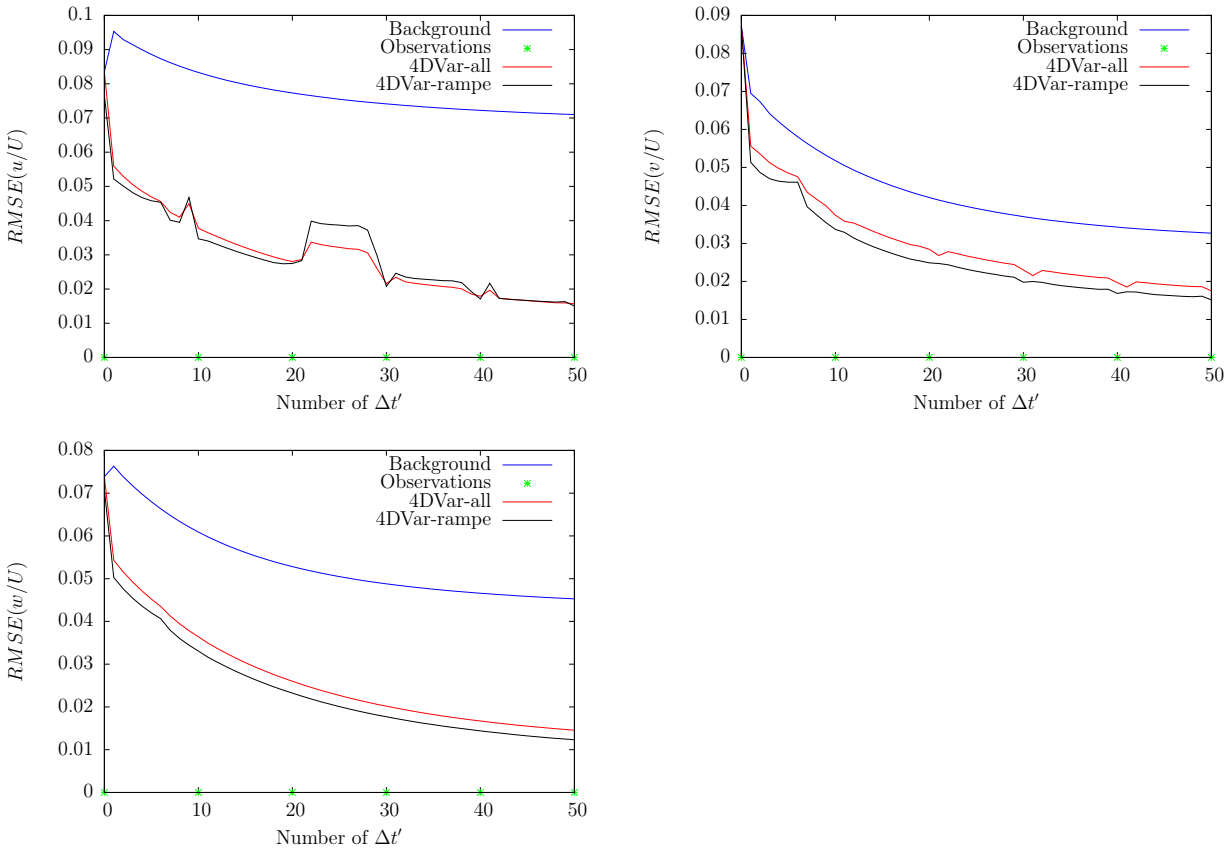

FIGURE 4.11 – Assimilation with 3D synthetic observations : comparison of the RMSE evolution of the background (in blue), the observations (in green), the analysis in case a (in red) and the analysis in case b (in black) with respect to the reference state for each velocity components over the assimilation window of $T'_a = 50\Delta t'$ within the sub-domain Ω_{obs}.

Map comparison

We compare in figure 4.12 the snapshots of each component u, v and w of the velocity field in the stream-wise plane $z = 3D$ at $t'_1 = 170$. The figure shows that

over the short assimilation window $T'_a = 50\Delta t'$, the 4DVar technique was able to correct the bias imposed on the initial background flow. In the downside, both cases haven't lowered the level of noise added in the initial background flow despite the observations and the inputted inlet are noise-free.

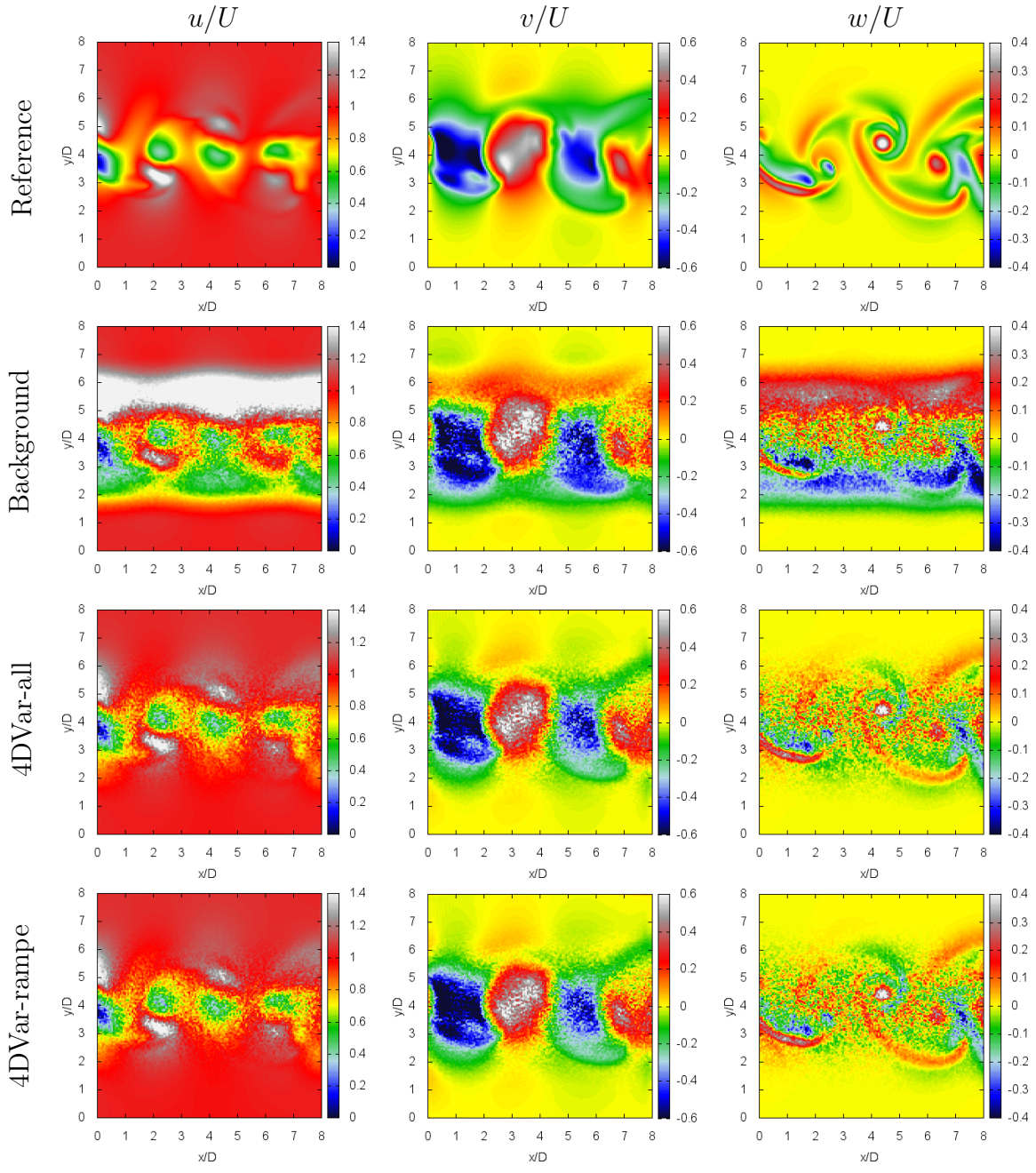

FIGURE 4.12 – Assimilation with 3D synthetic observations : snapshots of the velocity field for each component u, v and w at the beginning of the assimilation window at $t'_1 = 170$ in the stream-wise plane $z = 3D$. The row correspond to the reference, background, 4DVar-all and 4DVar-rampe flow, respectively.

We also compare in figure 4.13 the snapshots of each component u, v and w of the velocity field in the stream-wise plane $z = 3D$ at $t'_2 = 170.25$. We notice that

the DNS code is able to fairly denoise the initial perturbed flow in time. However the imposed bias influences the behavior of the background flow. Both assimilation cases corrected the bias and ensured a better reconstruction in time. The case b provides a slightly better result than the case a in terms of noise reduction within the sub-domain Ω_{obs}. This improvement is especially observed on the w component.

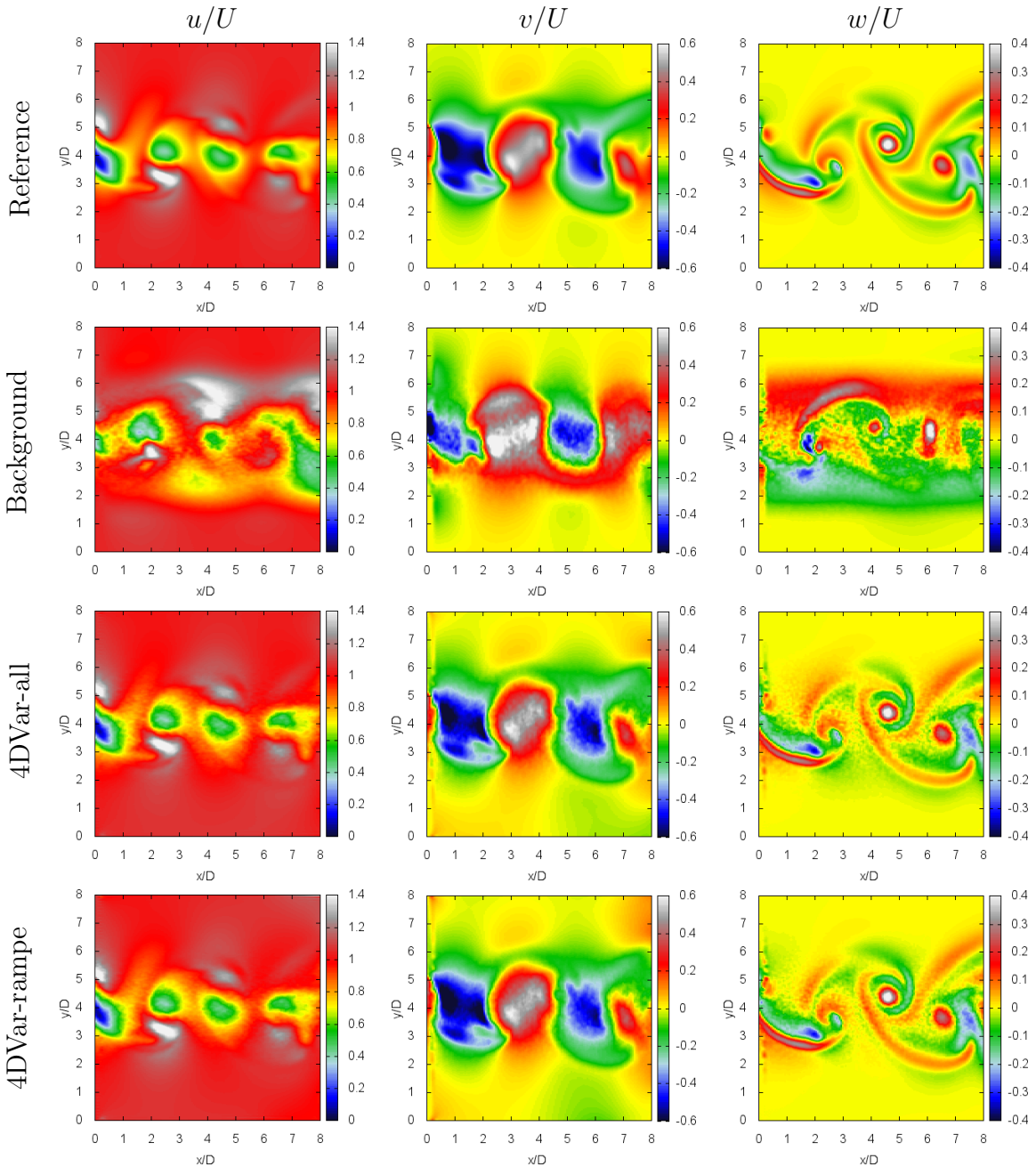

FIGURE 4.13 – Assimilation with 3D synthetic observations : snapshots of the velocity field for each component u, v and w at the end of the assimilation window at $t_2' = 170.25$ in the stream-wise plane $z = 3D$. The row correspond to the reference, background, 4DVar-all and 4DVar-rampe flow, respectively.

Energy spectrum comparison

Finally, we compare in figure 4.14 the energy spectrum in each case at $t'_1 = 170$ and $t'_2 = 170.25$. In both 4Var methods corrected the flow at the largest scale but didn't perform very well at the smallest scale. We notice that case b performed a slightly better correction of the initial smallest scales than case a. At the end of the assimilation window, the analysis results tend to stick to the background flow, nevertheless the case b still shows slight improvement with respect to the case a.

4.3.6 Conclusion and discussion

We assessed throughout this section the ability of the 4DVar code to correct a non-physical noise and bias over a short assimilation window $T'_a = 50\Delta t'$.

First and foremost, we identified the inherent source of model errors that can be encountered during the assimilation. In the one hand, these errors stem from the incomplete initialization of the DNS code. While the errors remain in a small range of $err/U = [-0.01; 0.01]$ over $500\Delta t'$, they might be much more amplified at a higher Reynolds number. A solution would consist in adding an additional control of the initial quantities in the same spirit as the control on the initial velocity components. On the other hand, the incompatibility between the boundary conditions model led to local errors on the y-axis borders and the outlet. We could have used sponge layers and construct their corresponding adjoint in the same vein as Lemke and Sesterhenne (2013). However, the implementation of the parallel 4DVar code for two different domains is a delicate task so we opted for an assimilation within the domain Ω and tuned the covariance matrix $\mathbf{R}$ and $\mathbf{B}$ (see § 4.3.3).

Globally, the 4DVar code was able to reconstruct the initial conditions from a noisy and biased background and a set of ideal three dimensional observations. The tuning of the observation error covariance matrix $\mathbf{R}$ improves slightly the solution at the largest and the smallest scale and eases the optimization process.

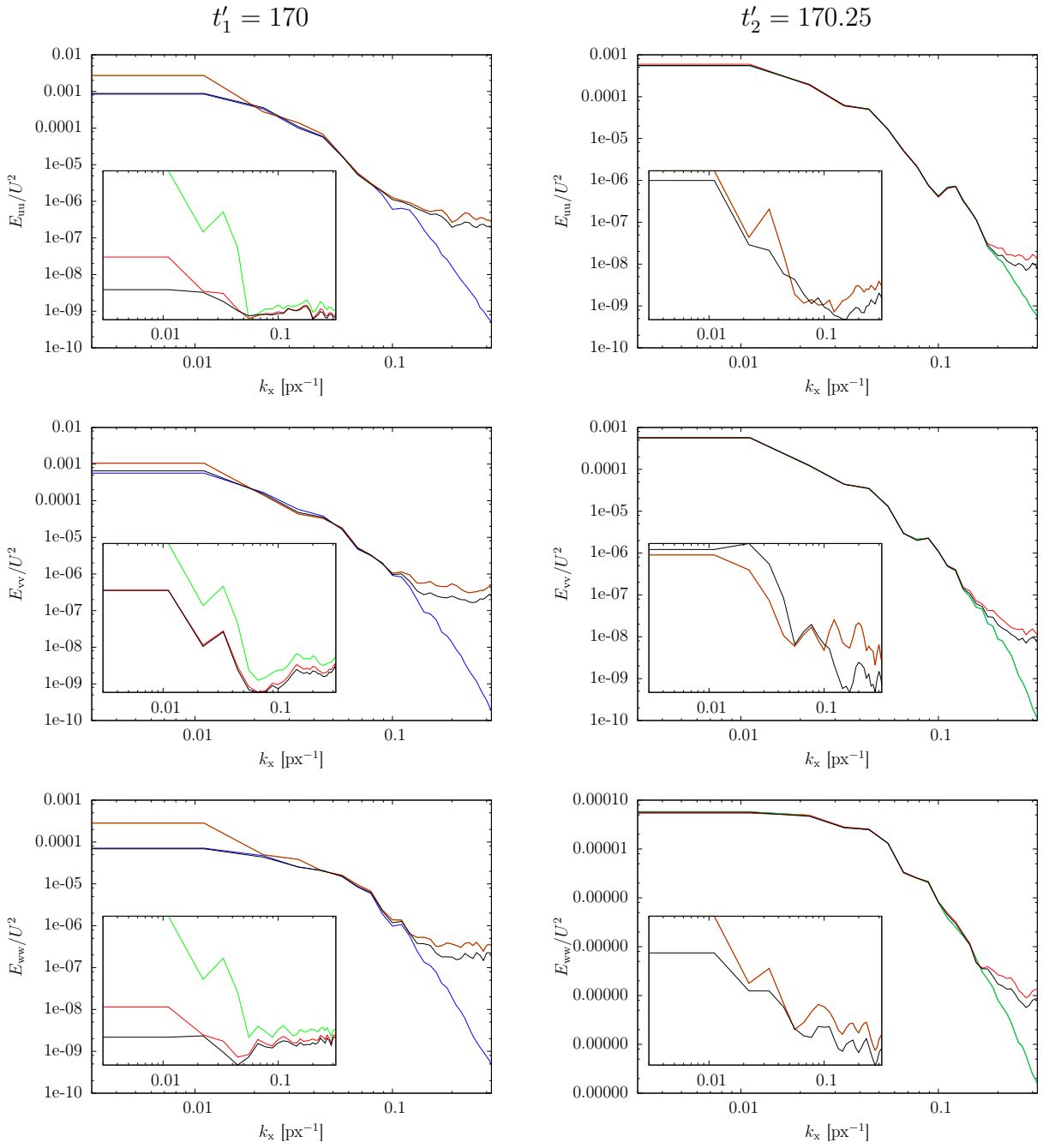

FIGURE 4.14 – Assimilation with 3D synthetic observations : comparison of the energy spectrum evolution of the reference flow (in blue), the background flow (in green), the 4DVar-all flow (in red) and the 4DVar-rampe black (in black) at $t'_1 = 170$ (on the left) and at $t'_2 = 170.25$ (on the right) in a cross-section plane $x = 4D$. The comparison of the spectra of the error for the same data with respect to the reference are shown in inset.

4.4 Towards the reconstruction of a full 3D velocity field with orthogonal-plane stereo PIV observations

We investigated in this section the capacity of the 4DVar technique to reconstruct a fully three dimensional flow with orthogonal-plane stereo PIV observations. In this context, we consider the reconstruction of a cylinder wake at Reynolds 300 given a set of observations within the inlet and the stream-wise observation planes. We layout in § 4.4.1 the experimental context and the measurement techniques employed to provide the experimental observations. We consider at first the reconstruction of the three dimensional velocity field with synthetic observations in § 4.4.2. We then consider in § 4.4.3 the reconstruction of the three dimensional velocity field with real observations extracted from two stereo-PIV systems.

4.4.1 Experimental context

The experimental cylinder wake flow was generated by the wind tunnel located at Irstea regional center in Rennes. The wind tunnel, illustrated in figure 4.15, was designed to simulate experimentally a mixing layer flow and was adapted to simulate experimentally a cylinder wake flow. It is equipped with two independent aeraulic circuits, each of them equipped with a 7.5kW blower, a $42kW$ cold battery and a 31kW heater. The two parallel air streams are blown separately from two air intakes into a conditioning chamber composed of filters and screens, before they are blown into a pressure conditioning chamber. This configuration can generate low turbulence homogeneous flows. The air streams are then blown into a convergent entrance cone with a contraction ratio of 2.5 and are separated at the inlet by an aluminum plate. Both flows end up in a test section of 3m $\times$ 1m $\times$ 1m. The velocity and temperature of each incoming flows can be set continuously and independently in a range of 0.5 to 5m/s and in a range of 5 to $35°C$.

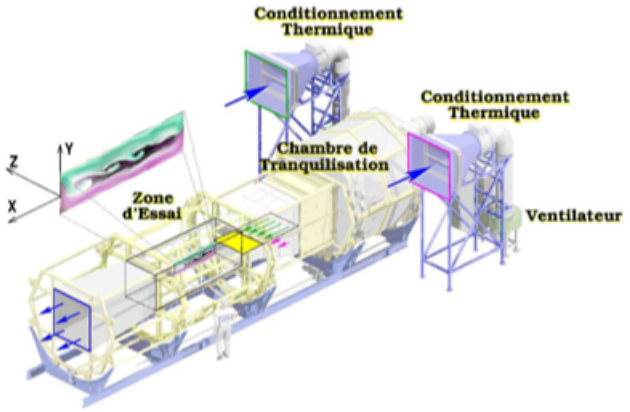

FIGURE 4.15 – General view of the mixing layer wind tunnel

The main experiment consists in two perpendicular 2D3C PIV measurements at the planes $x = 19D$ and $z = 0D$ behind the circular cylinder which are referred to

as SPIVS1 and SPIVS2 respectively. SPIV1 is composed of a Litron 15mJ laser and two cameras which covers a field of view of $14.9D \times 18.5D$. SPIV2 is composed of a Litron 15mJ laser and two cameras which covers a field of view of $15.3D \times 18D$. The experimental measurement setup is illustrated in figure 4.16.

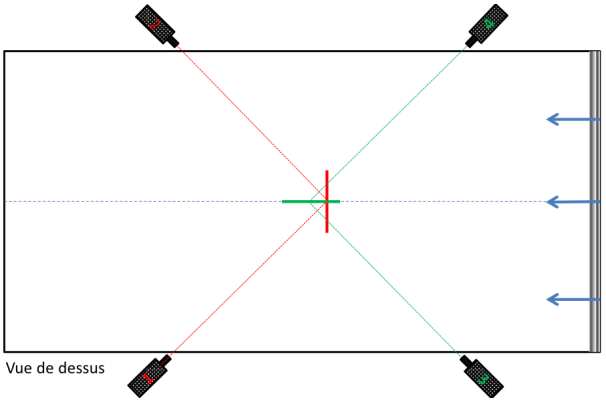

FIGURE 4.16 – Representation of the 2D3C PIV systems : SPIVS1 (in red) measures the three components velocity at the plane $x = 19D$ and PIVS2 (in green) measures the three components velocity at the plane $z = 0D$.

We acquired alternatively 3000 image pairs of the inlet and the stream-wise flows with SPIV1 and SPIV2 respectively, with a time interval of $2000\mu s$ between two successive images (500Hz). The time acquisition of each stereo PIV system is illustrated in figure 4.17.

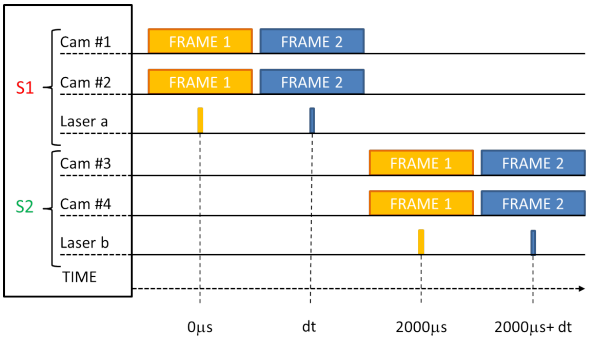

FIGURE 4.17 – Single frame configuration : each stereo PIV system requires a frequency dt to acquire two consecutive frames. In the single frame configuration, $dt = 1000\mu s$.

As the wind tunnel was initially designed for a mixing layer configuration, a cushion at the separation plaque was added to reduce the mixing layer effect. The circular cylinder has a length $L = 1000$mm and a diameter $D = 10$mm. It is equipped with two thin rectangular end plates with the specification recommended by Stansby (1974). The distance between the end plates is 300mm providing an aspect ratio $L/D = 30$. The clearance between the walls and the end plates is about of 350mm which is much larger than the thickness of the boundary layer. The blockage ratio is 1. The free stream velocity is adjusted to $U = 0.48$m/s so that the Reynolds number Re is 300.

4.4.2 Synthetic orthogonal plane stereo PIV observations

General context

We consider the same cylinder wake reference flow defined in § 4.3.1 to generate the synthetic observations. We recall that the reference flow is obtained by running Incompact3d from an initial null velocity field and by inputting an inflow velocity within the reference domain $\Omega_{\text{ref}} = 20D \times 12D \times 6D$. We conserve the same DNS step $\Delta t' = 0.005$ and run the simulation until $T'_{\text{ref}} = 30000\Delta t'$. We mimic the PIV observations by extracting alternatively the inlet plane $x = 0D$ and a stream-wise plane $z = 3D$ from the reference flow every $\Delta t'_{\text{obs}} = 10\Delta t'$. Thus, every observation in a given plane occurs every $\Delta t'_{\text{PIV}} = 20\Delta t'$. The time setup is illustrated in figure 4.18 and the characteristics of each run performed during this experiment are summarized in table 4.5.

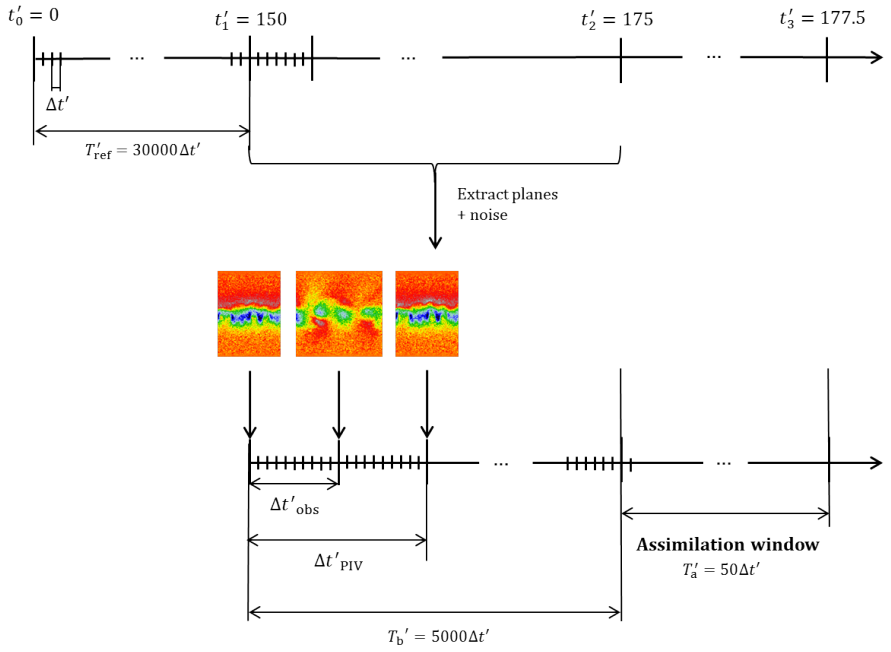

FIGURE 4.18 – Assimilation with synthetic orthogonal-plane observations : temporal setup for the reconstruction of a complete three dimensional flow with the 4DVar technique. In a first step we build the reference flow of this study by running a DNS over $T'_{\text{ref}} = 30000\Delta t'$ within the reference domain Ω_{ref}. In a second step, we generate an initial background from a null velocity by injecting the three components of the velocity field every $\Delta t'_{\text{PIV}} = 20\Delta t'$ at the inlet $x = 0D$ within the sub-domain Ω. After a $T'_b = 5000\Delta t'$ time period, the initial background is used to initialize the 4DVar code performed over the assimilation window $T'_a = 60\Delta t' = 0.3$.

Synthetic observations

In order to apply the 4DVar code to this problem, we have to carry out a pre-procedure which generates three dimensional observations from each occurrence of

Run	$Lx \times Ly \times Lz$	$nx \times ny \times nz$	Re	Duration	CPU time (h)
Reference	$20D \times 12D \times 6D$	$361 \times 217 \times 108$	300	$30000\Delta t'$	50
Empty box	$8D \times 8D \times 6D$	$145 \times 145 \times 108$	300	$5000\Delta t'$	9
4DVar	$8D \times 8D \times 6D$	$145 \times 145 \times 108$	300	$50\Delta t'$	5

TABLE 4.5 – Assimilation with synthetic orthogonal-plane observations : characteristics of each run performed to reconstruct the three dimensional flow from 3D synthetic observations. The empty box case corresponds to the simulation performed to obtain an acceptable initial background from an initial null velocity field and the injection of an inlet every $\Delta t'_{\text{PIV}} = 20\Delta t'$. The data assimilation method 4DVar considered a static background error covariance matrix $\mathbf{B}^{-1} = 1$ and the observation error covariance matrix $\mathbf{R}^{-1}$ built by algorithm 29.

an observed plane. We draw the reader's attention on the necessity to filter the observations at the borders in order to verify the free-slip boundary conditions and ensure that the code won't explode numerically. The entire pre-procedure that transforms the 2D observations into the 3D observations is summarized in algorithm 28. Figure 4.19 shows two snapshots of the reference and the observed flows at each observed plane at t'_2.

Algorithm 28 Synthetic observations pre-processing in the orthogonal-plane PIV observations configuration

1: Add a 10% white noise to each component of the observed velocity field at each occurence $\Delta t'_{\text{PIV}}$ and at each point of the grid
2: At each plane $x = 0D$ and $z = 3D$ perform a temporal interpolation at every $\Delta t'_{\text{obs}}$
3: Set a weight for the inlet plane $w_1(x) = \exp\left(-\frac{x^2}{2\sigma_1^2}\right)$
4: Set a weight for the stream-wise plane $w_2(z) = \exp\left(-\frac{(z-3D)^2}{2\sigma_1^2}\right)$
5: Compute the value of each component of the velocity flow $\mathbf{u}(x, y, z)$ at each non-observed point (x, y, z) from the values of the observed velocities at the inlet $\mathbf{u}_1(y, z)$ and at the stream-wise plane $\mathbf{u}_2(x, y)$

$$\mathbf{u}(x, y, z) = \frac{\mathbf{u}_1(y, z)w_1(x) + \mathbf{u}_2(x, y)w_2(z)}{w_1(x) + w_2(z)}$$

6: Filter the values at the boundary conditions in the y-axis to fit the free-slip boundary conditions

In light of the previous results in the synthetic case (see § 4.3), we tune the observation error covariance matrix with respect to the level of confidence at each point. In the orthogonal-plane observations context, we impose the highest confidence on the observed planes and lower rapidly the confidence in time. The construction of the observation error covariance matrix is summarize in algorithm 29 and is illustrated

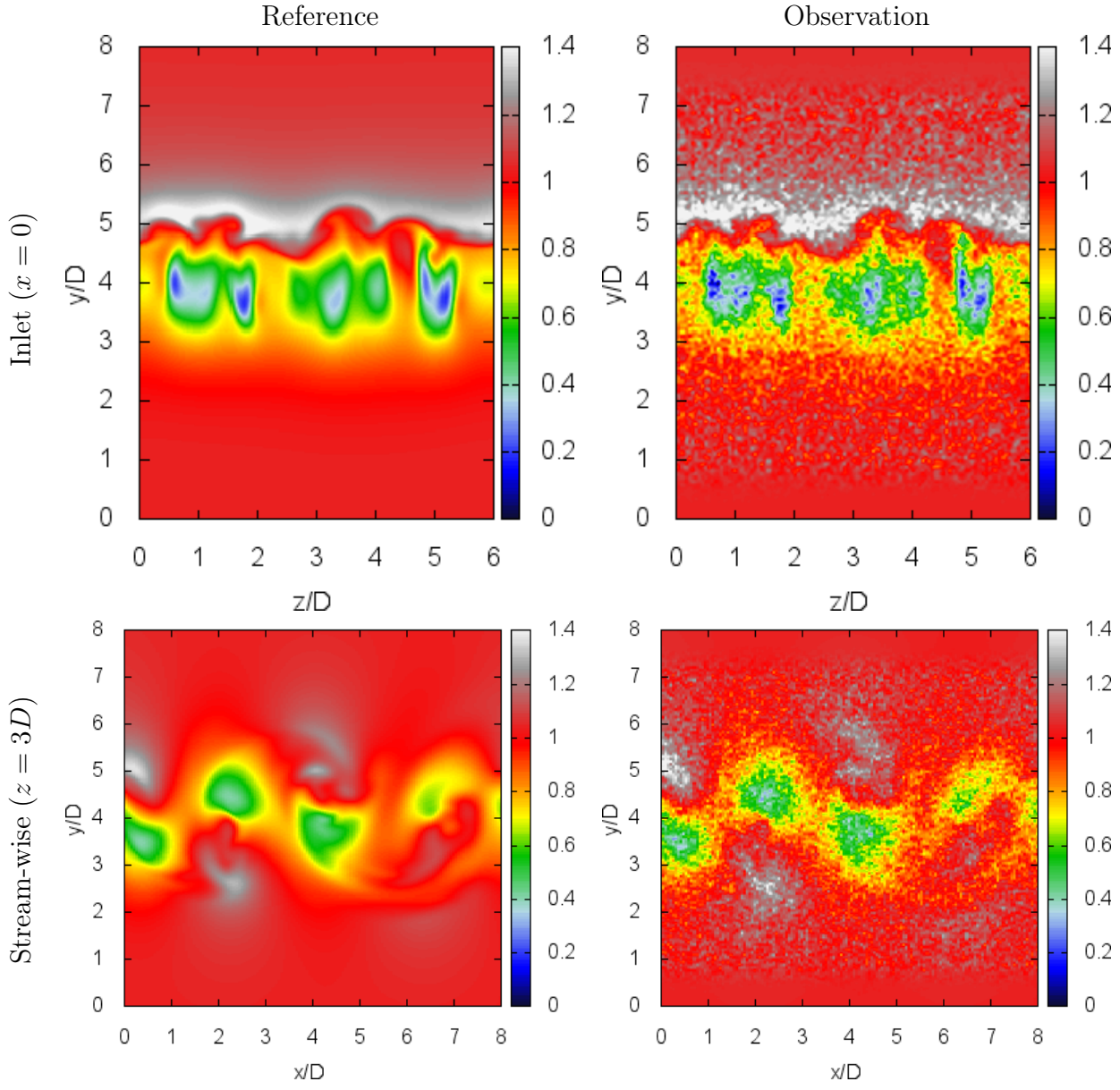

FIGURE 4.19 – Assimilation with synthetic orthogonal-plane observations : comparison of the initial reference and synthetic observation stream-wise velocity u/U at the observed planes $x = 0D$ (top) and $z = 3D$ (bottom) at $t'_2 = 175$.

in each plane $x = 0D$, $y = 3D$ and $y = 4D$ in figure 4.20.

Choice of the intial background

As we don't possess any additional information aside the observed planes, the background initial solution is obtained by performing the DNS from a null velocity and input the reference inflow every $\Delta t'$. The inlet planes at every $\Delta t'$ are obtained by a linear temporal interpolation of the observed inlet every $\Delta t'_{\text{PIV}} = 20\Delta t'$. This procedure is referred to as "Empty box" in table 4.5.

A first simulation was carried out by considering an ideal inflow which is directly

Algorithm 29 Construction of the invert observation error covariance matrix $\mathbf{R}^{-1}$

1: We set the most confidence on the observed orthogonal planes $x = 0D$ and $z = 3D$

$$\mathbf{R}^{-1}(x,y,z) = diag\left(\max\left(\exp\left(-\frac{x^2}{0.2}\right), \exp\left(-\frac{(x-3D)^2}{0.2}\right)\right)\right)$$

2: We reduce the confidence along they-axis to take into account the incompatibility between the background free-slip boundary condition and observations' model

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1}(x,y,z).diag\left(\exp\left(-\frac{(y-1D)^2}{0.2}\right)\right), y < 1D$$

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1}(x,y,z).diag\left(\exp\left(-\frac{(y-7D)^2}{0.2}\right)\right), y > 7D$$

3: We reduce the confidence along the x-axis to take into account the incompatibility between the background and observations model at the outlet

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1}(x,y,z).diag\left(\exp\left(-\frac{(x-6D)^2}{0.2}\right)\right), x > 6D$$

4: We reduce the confidence along the z-axis as the generated observations don't strictly verify the periodicity condition

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1}(x,y,z).diag\left(\exp\left(-\frac{(z-1D)^2}{0.2}\right)\right), y < 1D$$

$$\mathbf{R}^{-1}(x,y,z) = \mathbf{R}^{-1}(x,y,z).diag\left(\exp\left(-\frac{(z-5D)^2}{0.2}\right)\right), y > 5D$$

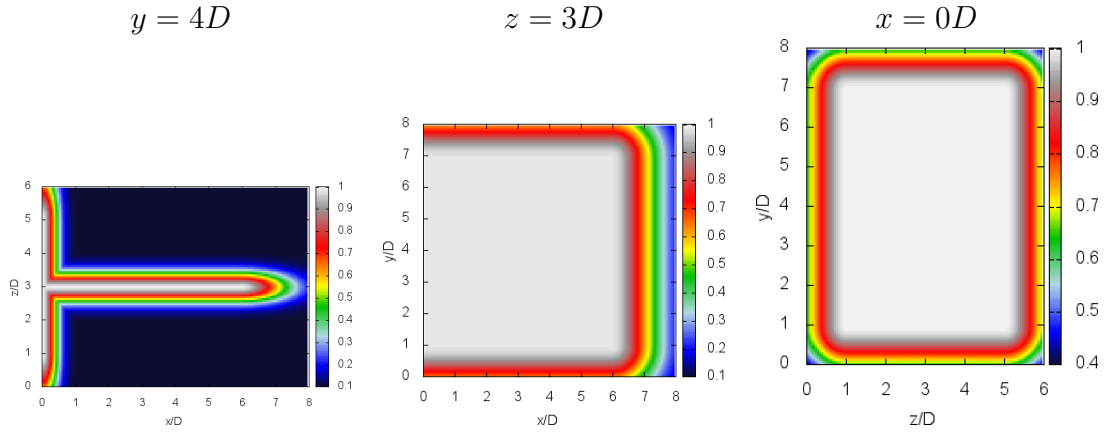

FIGURE 4.20 – Assimilation with synthetic orthogonal-plane observations : representation of the inverted observation covariance matrix $\mathbf{R}^{-1}$ in the plane $y = 4D$, $z = 3D$ and $x = 0D$

extracted from the reference flow at each $\Delta t'_{\text{PIV}}$. After performing a simulation from scratch over $T'_b = 5000\Delta t'$ time steps, the obtained background flow revealed to be very close to the reference flow as illustrated in figure 4.21. In this particular configuration, the DNS is capable to retrieve a good estimation of the reference flow, therefore the assimilation is unnecessary.

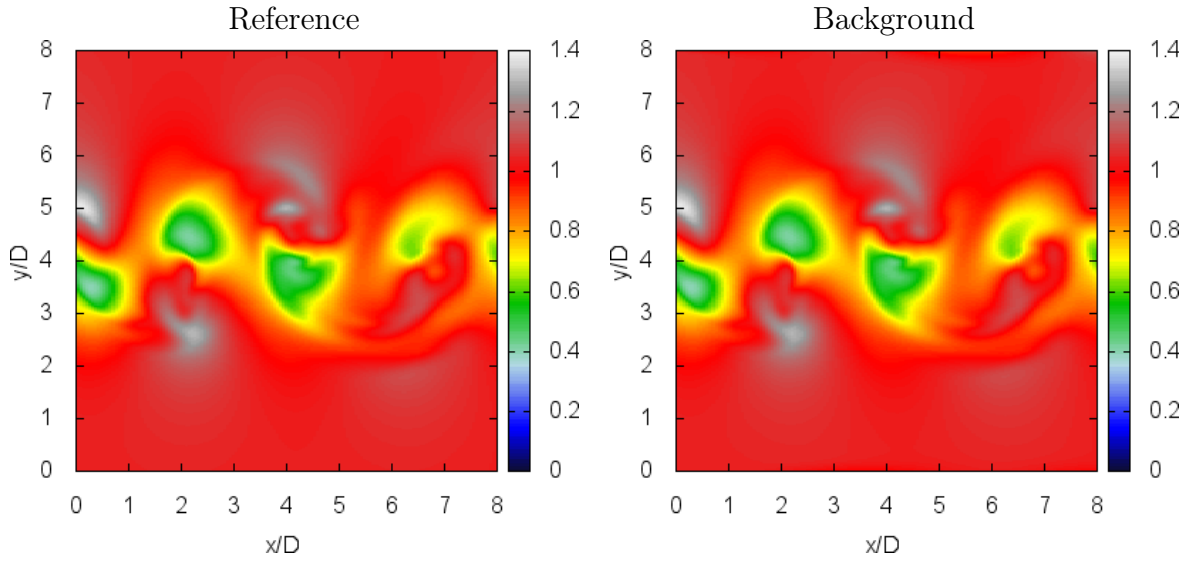

FIGURE 4.21 – Assimilation with synthetic orthogonal-plane observations : comparison between the streamwise velocity of the reference flow (left) with the background flow (right) after $T'_b = 5000\Delta t'$. The background is obtained by injecting the temporal integration of the exact inflow at each $\Delta t'$.

We induced a 10% synthetic noise into the inflow and performed once more the simulation from scratch. The synthetic noise is adjusted according to a Gaussian profile to avoid the apparition of instabilities at the free slip boundaries which can compromise the simulation. After $T'_b = 5000\Delta t'$, the obtained background has a different structure than the reference as illustrated in figure 4.22. We also notice that the noise injected at the inlet is denoised by the DNS code.

Results

We show in this section the results of the assimilation run with the initial background and synthetic observations described in § 4.4.2 and § 4.4.2, respectively. The assimlation was carried out with two outer loop iterations over the data assimilation window $T'_a = 50\Delta t$.

We analyze the evolution of the cost function $J(\delta\gamma_0)$ and the norm of its gradient $\|\nabla J(\delta\gamma_0)\|$ with respect to the number of inner loop iterations. Figure 4.29 shows that during the first outer loop iteration, the value of the cost function decreased rather slowly. The first correction of the second outer loop provided a better correction and led the assimilation to a better solution.

We display the evolution of the velocity norms of each component of the reference, background, observation and analysis at each observed planes $x = 0$ and $z = 3D$.

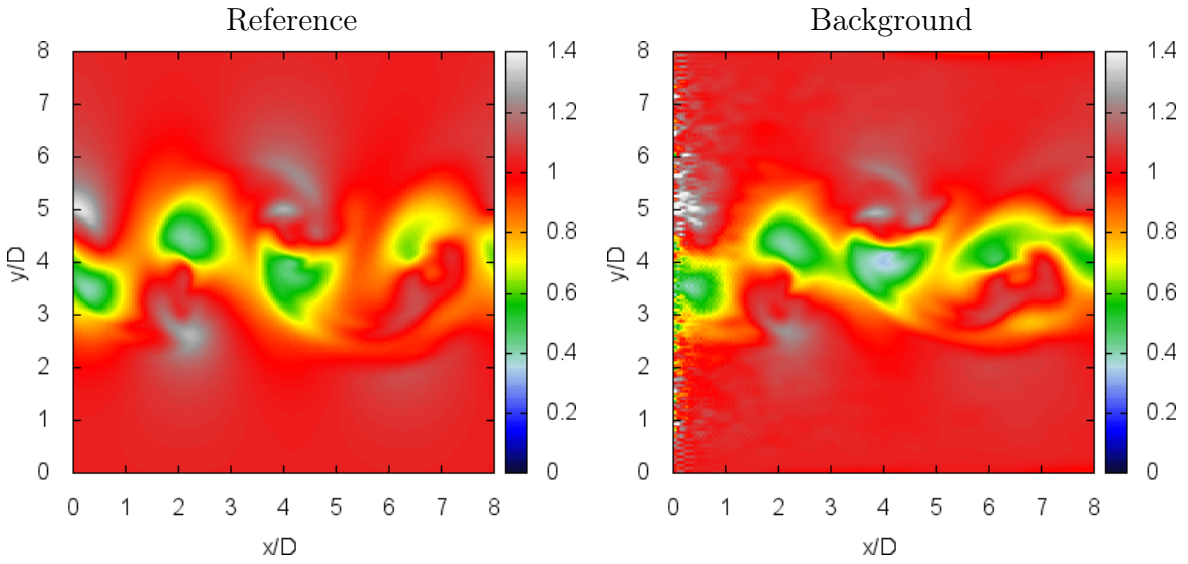

FIGURE 4.22 – Assimilation with synthetic orthogonal-plane observations : comparison between the strea-mwise velocity of the reference flow (left) with the background flow (right) after $T_b' = 5000\Delta t'$. The background is obtained by injecting the temporal integration of the a noisy inflow at each $\Delta t'$.

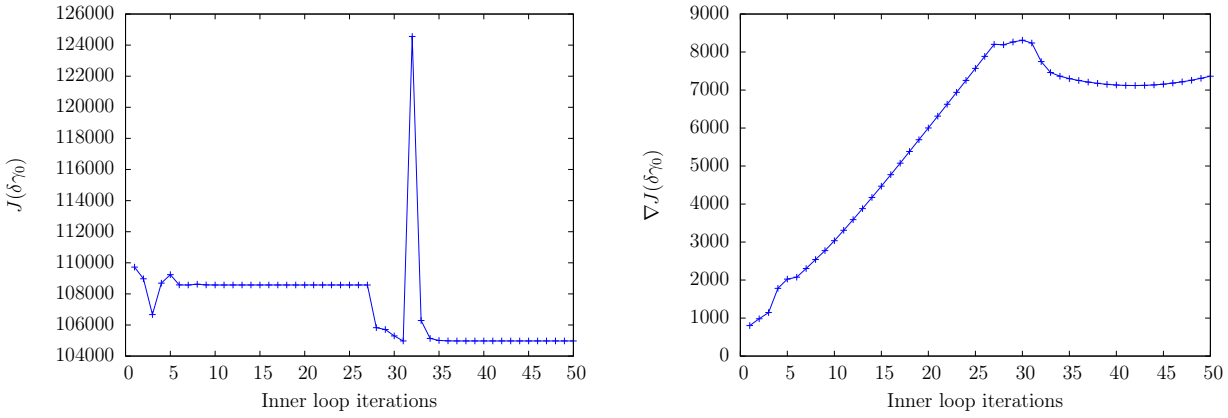

FIGURE 4.23 – Assimilation with synthetic orthogonal-plane observations : evolution of the cost function (left) and cost function gradient (right) with respect to the inner loop iterations for two outer loop iterations.

Figure 4.24 shows that the assimilation achieved notable corrections at the inlet plane every $\Delta t'_{\mathrm{PIV}}$, however, in general the analysis trajectory tends to stick to the background trajectory. The snapshots of each velocity components showed very few visible corrections of the background. Therefore, the assimilation failed to provide a satisfying correction of the initial and inflow conditions with synthetic observations, however the assimilation performed better with real observations as described in § 4.4.3.

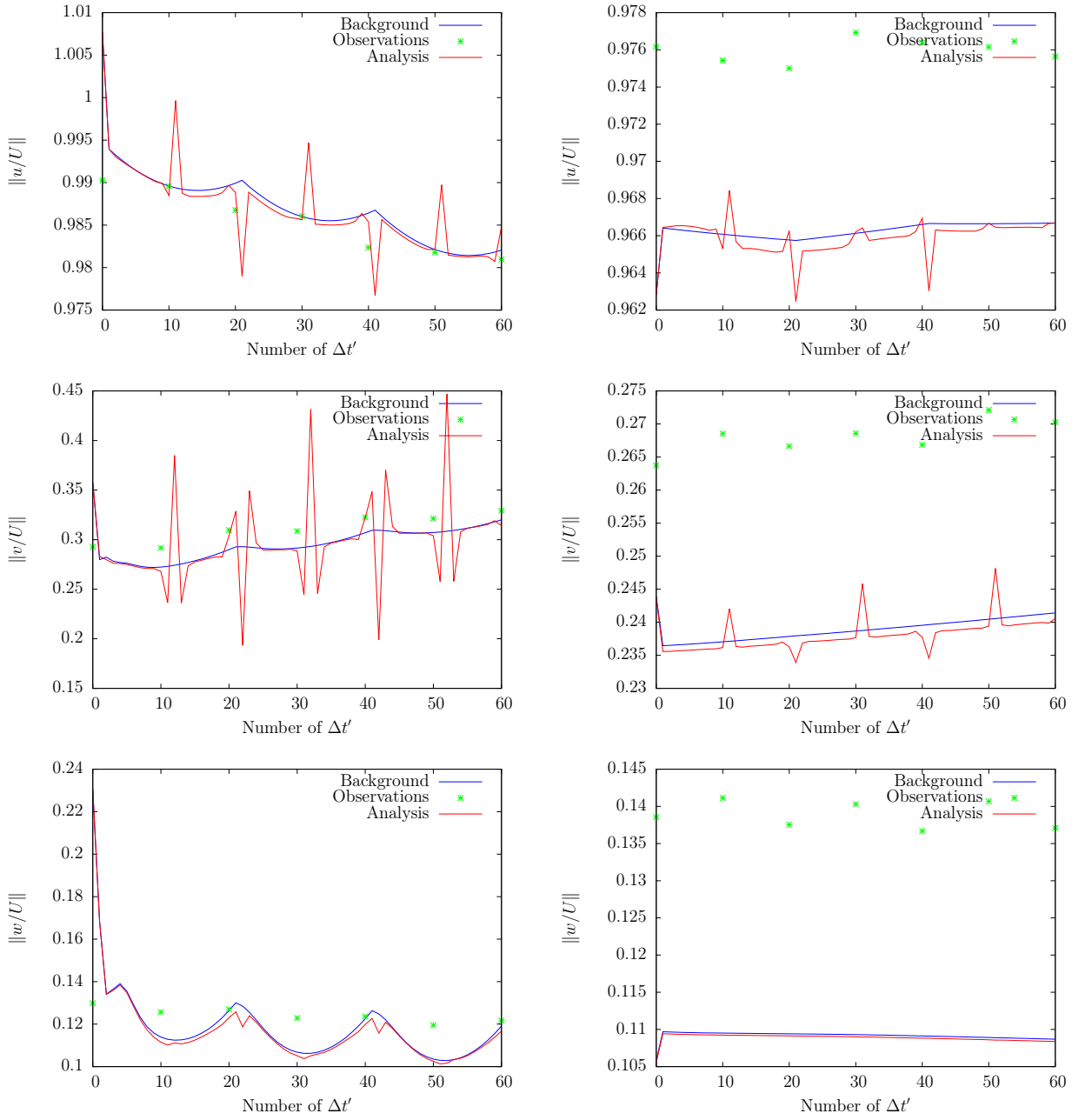

FIGURE 4.24 – Assimilation with synthetic orthogonal-plane observations : comparison of the norm of each component of the velocity field from $t_2 = 175$ to $t_3 = 175.3$.

4.4.3 Real orthogonal-plane stereo PIV observations

General context

The PIV measurements were calibrated to the same resolution than the DNS. In this configuration, the cylinder diameter is discretized on $1D = 17$ nodes instead of 18 nodes. The velocity observations are defined on $20D \times 16D$ at the inlet plane $x = 0$ and on $16D \times 20D$ at a stream-wise plane $z = 3D$. The characteristic velocity is set as $U = 0.48$m/s. The temporal setup of this experiment is described in figure 4.25.

The characteristics of each run performed within this experiment are summarized in table 4.6.

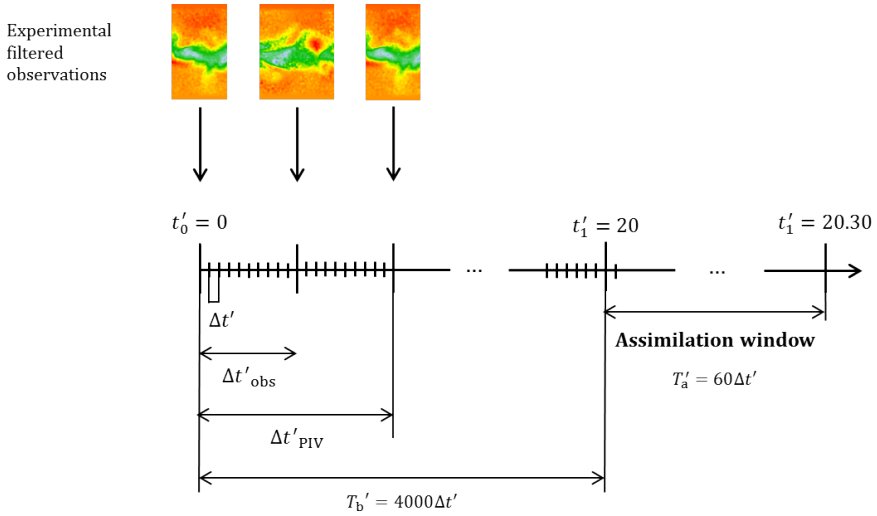

FIGURE 4.25 – Assimilation with real orthogonal-plane observations : temporal setup for the reconstruction of a complete three dimensional flow with the 4DVar technique. In a first step we build the reference flow of this study by running a DNS over $T'_{\text{ref}} = 30000\Delta t' = 150$ within the reference domain Ω_{ref}. In a second step, we generate an initial background from a null velocity and by injecting alternatively the three components of the velocity field every $\Delta t'_{\text{obs}} = 10\Delta t'$ at the plane $x = 0D$ and $z = 3D$. After a $T'_b = 5000\Delta t'$ time period, initial background is used to initialize the 4DVar code performed over the assimilation window $T'_a = 50\Delta t' = 0.25$.

Run	$Lx \times Ly \times Lz$	$nx \times ny \times nz$	Re	Duration	CPU time (h)
Empty box	$8D \times 10D \times 6D$	$137 \times 171 \times 102$	300	$4000\Delta t'$	7
4DVar	$8D \times 10D \times 6D$	$137 \times 171 \times 102$	300	$60\Delta t'$	10

TABLE 4.6 – Assimilation with real orthogonal-plane observations : characteristics of each run performed to reconstruct the three dimensional flow from 3D piv observations. The empty box case corresponds to the simulation performed to obtain an acceptable initial background from an initial null velocity field and the injection of an inlet every $\Delta t'_{\text{PIV}} = 20\Delta t'$. The data assimilation method 4DVar considered a static background error covariance matrix $\mathbf{B}^{-1} = 1$ and the observation error covariance matrix $\mathbf{R}^{-1}$ built by algorithm 29.

Similarly to the synthetic case, we define the smallest sub-domain possible which contains the turbulent structures. In this context, the definition of the sub-domain depends on the observations. The real observations structures aren't as regular as the synthetic structures and are occasionally spread along the y-axis. In order to fit as much as possible the free-slip condition along the y-axis, we have to increase

the sub-domain to $L_y = 10D$. In addition to the errors inevitably induced by the intialization of the DNS and the boundary condition model incompatibilities as described in § 4.3.2, the observed flow doesn't present a strict periodicity length as assumed in the DNS. We estimated instead the statistical periodicity length by comparing the mean and the standard deviation of each component of the velocity field in time at different position at fixed position in the inlet plane. Figure 4.26 shows a comparison between the mean and the standard deviation of each component of the velocity field in time at a punctual position. We fixed the periodic length at $6D$. All in all, we run the assimilation on the sub-domain $\Omega = 8D \times 10D \times 6D$.

In terms of observations, we reconstruct the three dimensional observations from the two dimensional observations by applying the algorithm 28. The first step of the algorithm is replaced by a filter that We also employ the algorithm 29 to construct the $\mathbf{R}^{-1}$ matrix which is illustrated in figure 4.27 at plane $y = 4D$, $z = 3D$ and $x = 0D$. In this configuration, as we don't have a strictly periodic flow it is necessary to lower the confidence along the z-axis.

Choice of the intial background

Similarly to the synthetic case described in § 4.3.4, the background initial solution is obtained by performing the DNS from a null velocity and input the temporal interpolation of the reference inflow every $\Delta t'$. The latter is obtained by a linear temporal interpolation of the observed inlet every $\Delta t'_{\text{PIV}}$. Figure 4.28 compares the snapshots the real observation with the initial background at the stream-wise plane $z = 3D$ obtained after running $4000\Delta t'$ from scratch. We notice that the background flow is less detailed with respect to the observations. We also notice that the flow seems to diverge from the observation at the outflow as the influence of the inlet faded and the outflow boundary condition is incompatible with the observation. Nevertheless, the three dimensional background flow is conform to the dynamics of the system and captures the trend of the observations on the observed plane, which is sufficient to start the 4DVar code.

Results

We analyze the evolution of the cost function $J(\delta\gamma_0)$ with respect to the number of inner loop iterations. Figure 4.29 shows that the assimilation process has difficulties to find the correct descent direction.

As we only observe two orthogonal planes, it is clearly unreasonable to trust the extrapolated data within the 3D volume. Therefore, we limit our statistics comparison to the observed plane $x = 0$ and $z = 3D$. Moreover, the observations are noisy and only represent a trend of the true solution which lies somewhere in between the observations and the dynamical representation of the flow. Therefore, instead of studying the evolution of the RMSE in the observed planes, we analyze norm of each velocity component of the observation, the background and the analysis at the observed planes $x = 0$ and $z = 3D$. Figure 4.30 shows that the analysis achieves notable corrections on the inlet similarly to the the synthetic case § 4.4.2. This phenomenon could be improved by implementing a temporal average on the inflow condition as

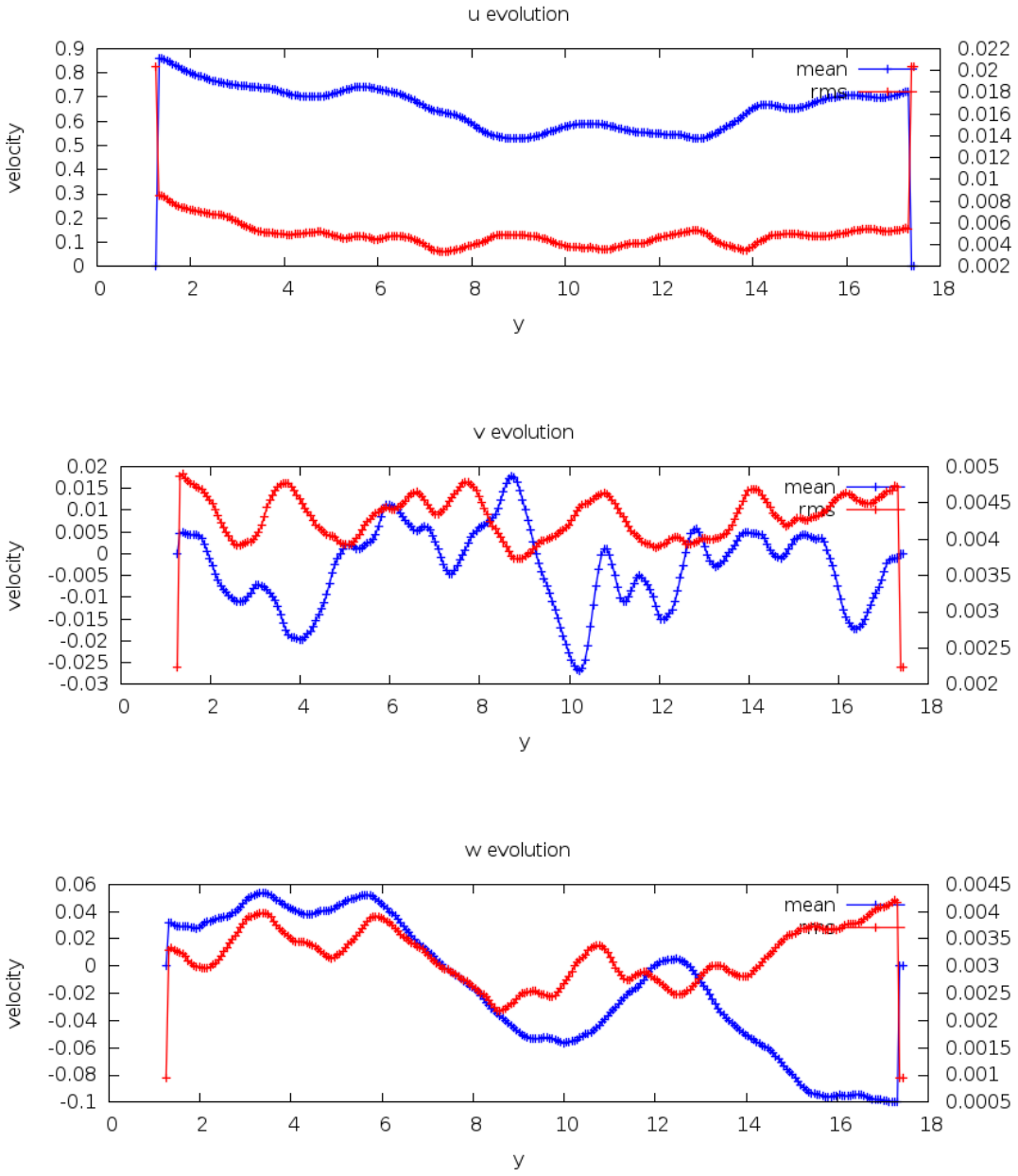

FIGURE 4.26 – Assimilation with real orthogonal-plane observations : comparison of the mean and the standard deviation of each component of the velocity field in time at a punctual position on the inlet plane at $y = 4D$

proposed in Gronskis et al. (2013). At the stream-wise plane, the analysis trajectory generally tends to get closer to the observations, excepted for the v component.

We analyze in figure 4.31 the snapshots of each component u/U, v/U and w/U of the velocity field in the observed stream-wise plane $z = 3D$ at the beginning of the assimilation window t'_0. We compare the snapshots of the observations, the background and the analysis obtained by the 4DVar code. We observe that the analysis retrieved similar turbulent structures to the observations for each component, while it conserved the same velocity levels as the background flow. This correction leads

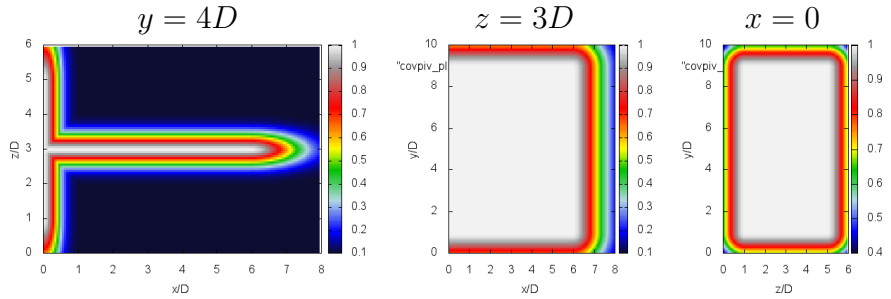

FIGURE 4.27 – Assimilation with real orthogonal-plane observations : representation of the observation covariance matrix $\mathbf{R}^{-1}$ in the plane $y = 4D$, $z = 3D$ and $x = 0D$.

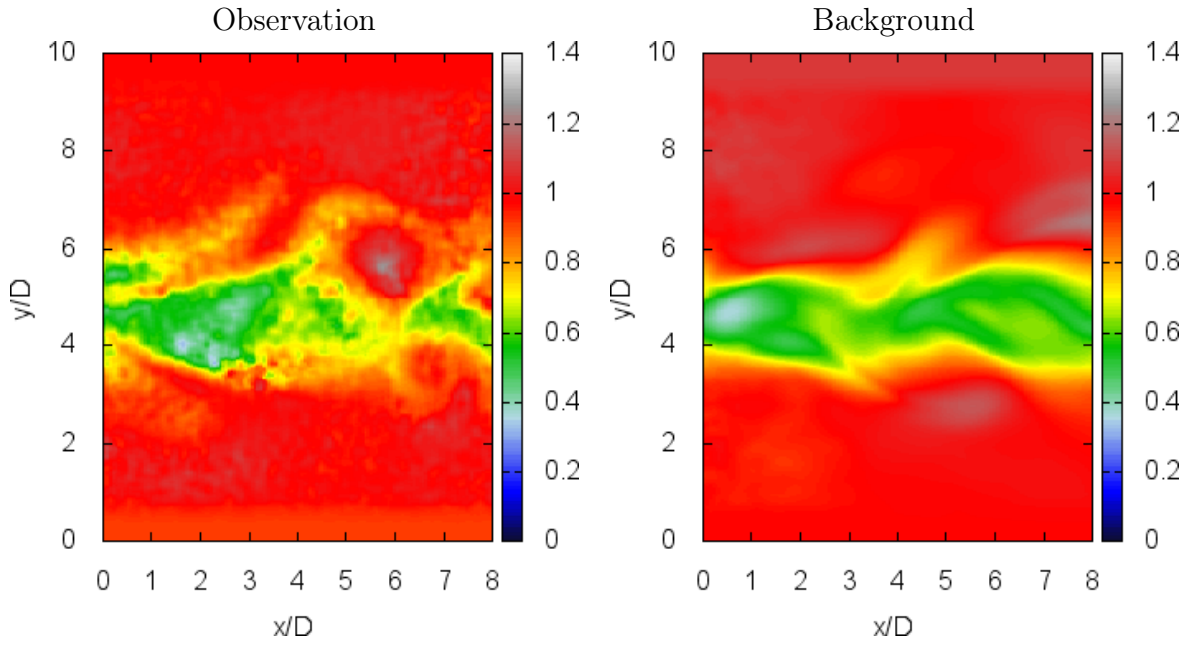

FIGURE 4.28 – Assimilation with real orthogonal-plane observations : comparison of the initial observation (left) and background (right) stream-wise velocity u/U at the observed stream-wise plane $z = 3D$. The background is obtained after running the DNS over a $4000\Delta t'$ period from a null velocity field and the input of the temporal interpolation of the observation inlet at each $\Delta t'$.

to a denoised and more detailed flow in the observed plane $z = 3D$.

In terms of inlet correction, the correction of the inlet at the beginning of the assimilation window $t'_1 = 20$ is too small to be discerned directly as illustrated in figure 4.32.

We can assess the correction performed within the entire volume by analyzing each component in the cross-section plane $y = 4D$. Figure 4.33 shows that the analysis flow leveled the values of each velocity component to follow the trend imposed by the observations. Nevertheless, the analysis map remains close to the background as it doesn't possess any other information beside the observed planes.

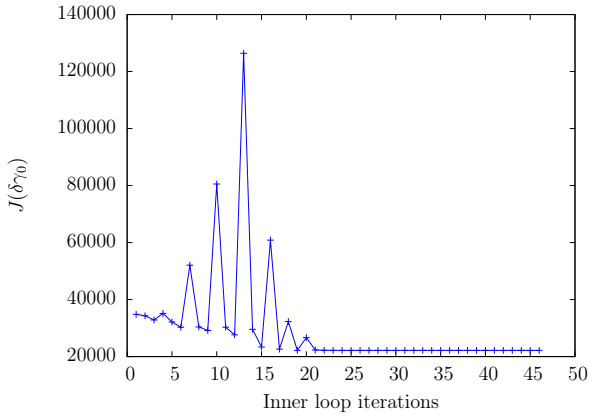

FIGURE 4.29 – Assimilation with real orthogonal-plane observations : cost function evolution with respect to the inner loop iterations.

4.4.4 Conclusion and discussion

We carried out the reconstruction of the three components of the velocity fields from a sequence of two orthogonal-plane synthetic and real observations. In both cases, it is crucial to perform a post processing procedure on the observations described in algorithm 28. The latter generated by Incompact3d from a null velocity field with a input of the temporal interpolation of the inflow at each $\Delta t'$. In the synthetic case, we are able to retrieve a very comparable flow to the reference, however, by adding noise to the inflow we obtain a sufficiently modified flow to justify the use of the variational assimilation technique. In opposition, the initial background obtained by the temporal interpolation of the real inflow observations isn't as detailed as the observations but provides a sufficient estimation of the flow to perform the assimilation.

In the one hand, we performed the assimilation with synthetic noisy observations. In this configuration, the assimilation had difficulties to converge and achieved negligible corrections of the background trajectory. This case could be improved by implementing a temporal average on the inflow condition similarly to Gronskis et al. (2013). As the assimilation performed better results with real orthogonal-plane observations, this case lays out the difficulty of generating synthetic realistic observations.

On the other hand, we achieved the reconstruction of the three dimensional flow from pre-processed observations acquired from a stereo-PIV setup. This experiment represented a particularly tedious task as the observations don't obey to a strict periodicity conditions imposed at the boundaries along the z-axis in addition to the model incompatibilities described in § 4.3.2. The observation error covariance matrix $\mathbf{R}$ played an important role in this reconstruction as it indicates the areas in which the boundary conditions model is incompatible. We also remind that these results were obtained on a short assimilation window $T'_a = 60\Delta t'$ containing 6 observations including 4 observations at the inlet $x = 0$. The increase of this temporal window can clearly ease the reconstruction of the flow.

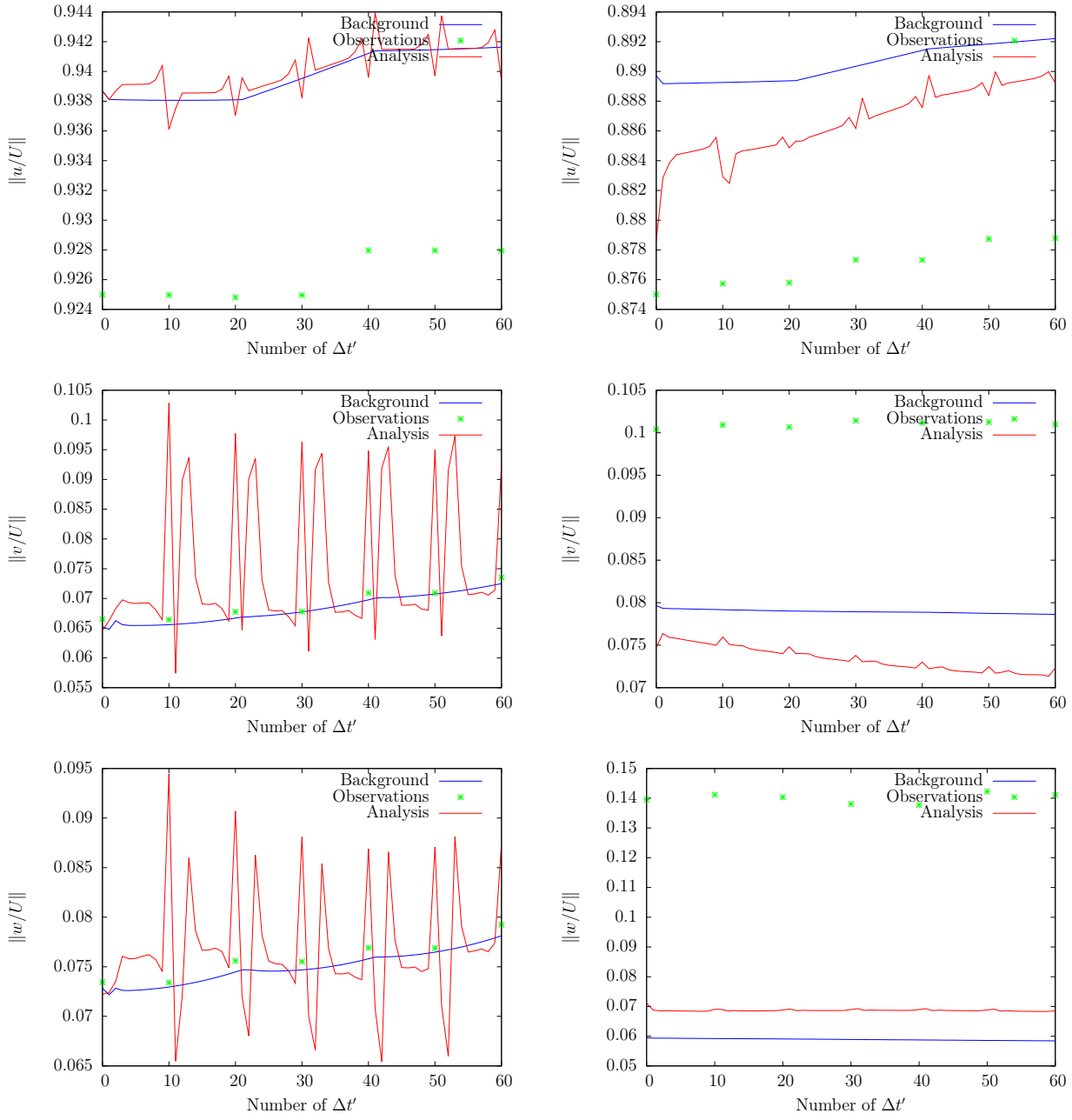

FIGURE 4.30 – Assimilation with real orthogonal-plane observations : comparison of of the norm of each component of the velocity field from $t'_1 = 20$ to $t'_2 = 20.3$.

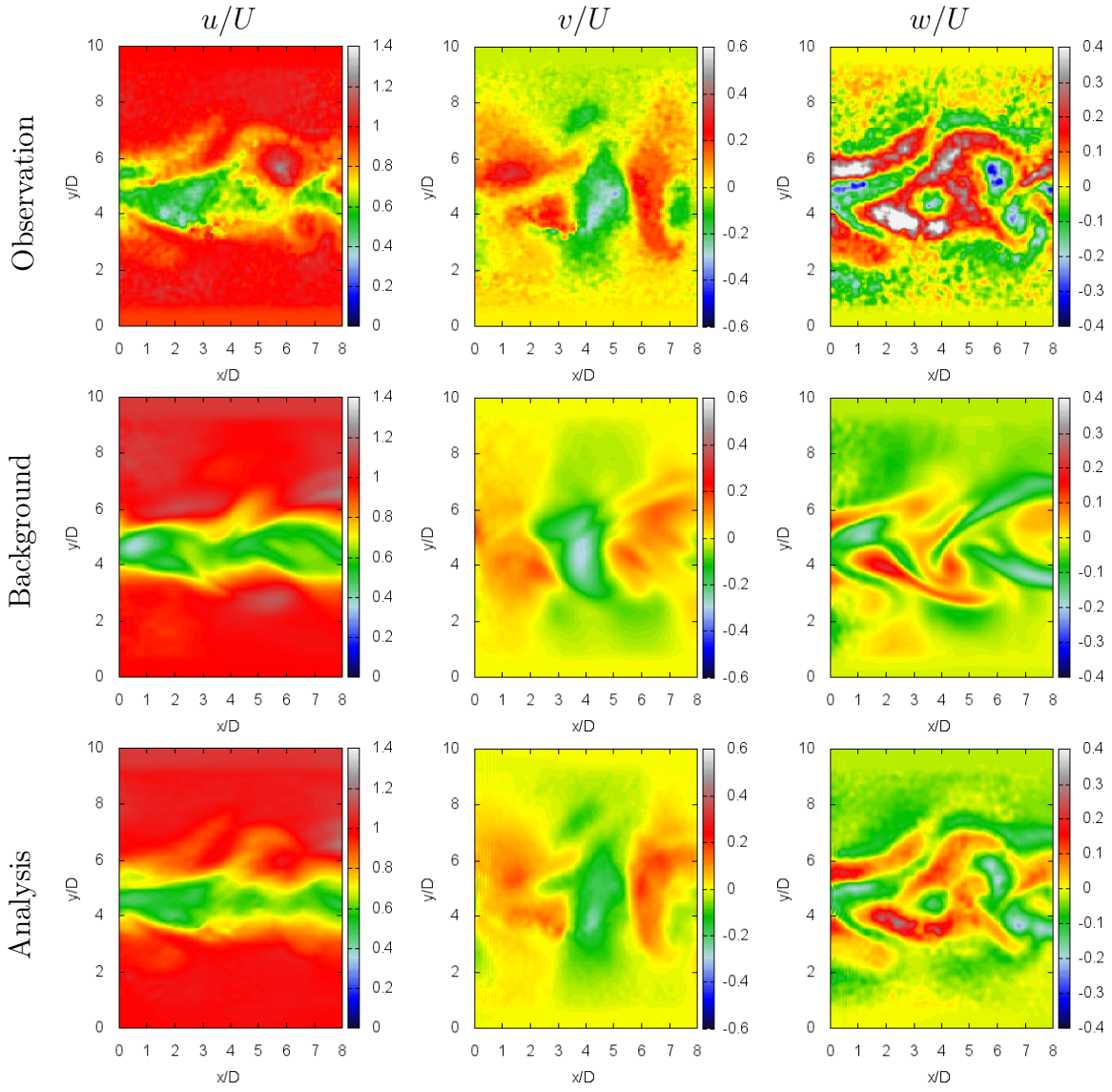

FIGURE 4.31 – Assimilation with real orthogonal-plane observations : snapshots of the velocity field for each component u, v and w at the beginning of the assimilation window at $t'_1 = 20$ in the stream-wise plane $z = 3D$. Each row corresponds to the observation, the background and the analysis, respectively.

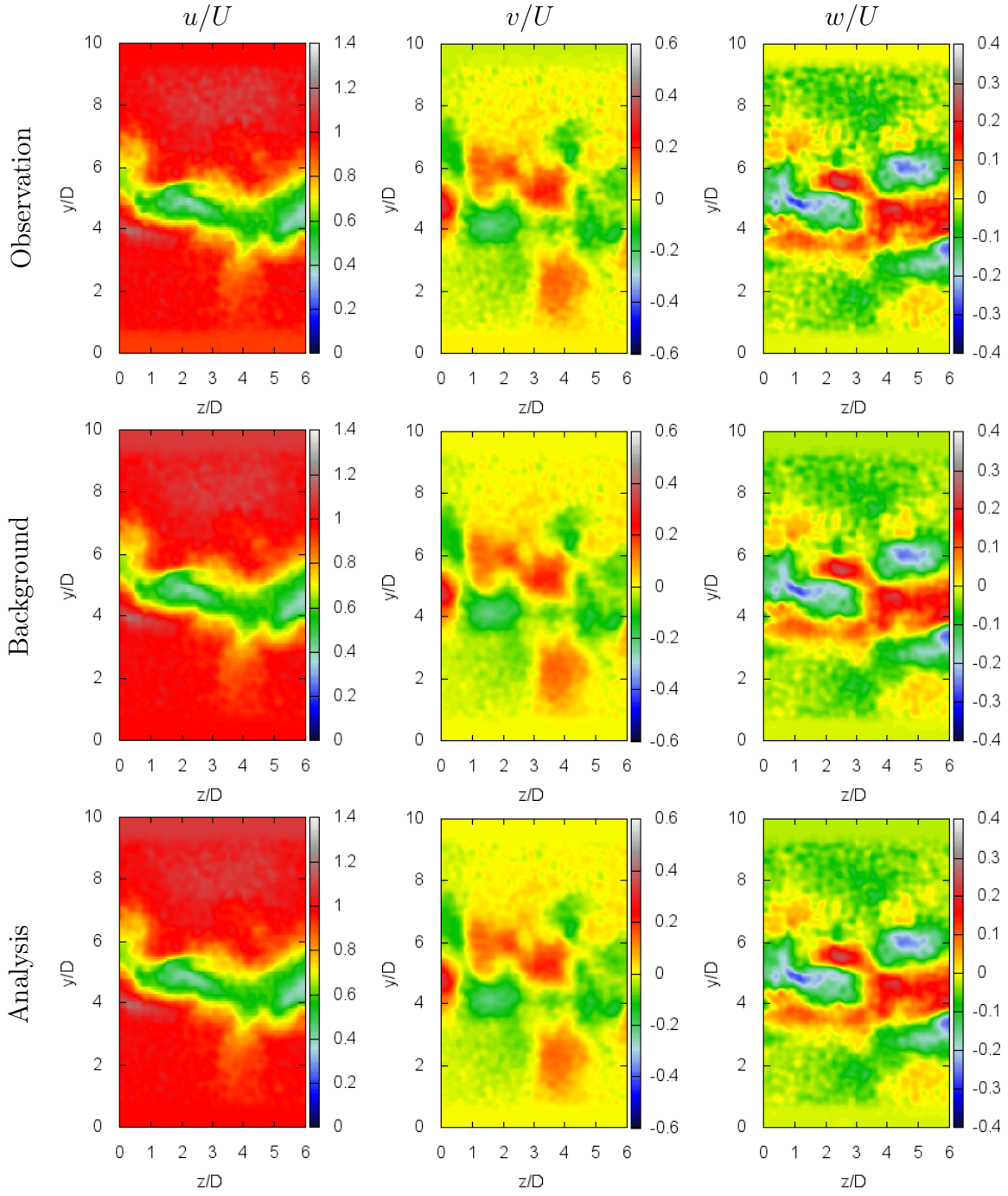

FIGURE 4.32 – Assimilation with real orthogonal-plane observations : snapshots of the velocity field for each component u, v and w at the beginning of the assimilation window at $t'_1 = 20$ in the stream-wise plane $z = 3D$. Each row corresponds to the observation, the background and the analysis, respectively.

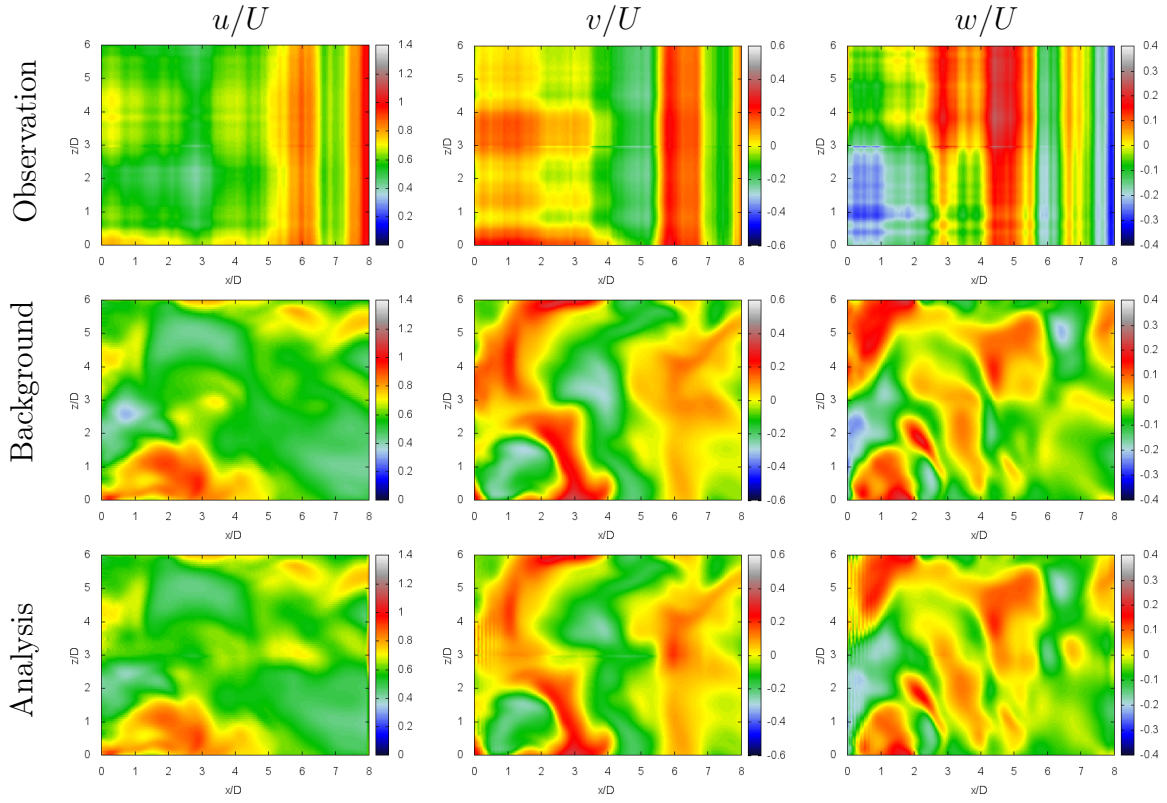

FIGURE 4.33 – Assimilation with real orthogonal-plane observations : snapshots of the velocity field for each component u, v and w at the beginning of the assimilation window $t_1' = 20$ in the stream-wise plane $y = 4D$. Each row corresponds to the observation, the background and the analysis, respectively.

Conclusions and perspective

Conclusion

The fluid mechanics field context possess a wide range of various methods to reconstruct a realistic turbulent flow. During these last decades, the trend is to couple experimental measurements with dynamical models, particularly by using data assimilation techniques. These techniques emerged from the Numerical Weather Prediction field and are well suited for large sized problems.

We covered in chapter 2 the main data assimilation approaches so far. In the framework of this thesis, we applied a variational data assimilation approach, referred to as 4DVar in the following, which consists in solving a optimal control problem under the constraint of the dynamical model. This approach is well-known for its accuracy and relies on the construction and the implementation of a so-called adjoint procedure. The latter is built upon the original dynamics code and provides the gradient descent direction for the optimization. The construction of the adjoint is eased by the use of an automatic differentiation tool but requires a systematic and rigorous validation procedure to ensure that the accuracy of the original dynamics code is conserved.

Reconstruction of a free surface flow

In chapter 3, we implemented and evaluated a classical incremental 4DVar technique (Courtier et al., 1994) to a simplified SWE code. We compared this method with the ensemble-based method 4DEnVar, proposed by Liu et al. (2008), and various strategies based on the latter (Yang et al., 2015), to reconstruct the complete characteristics of a free surface from partial and noisy observations.

In general, both methods retrieved successfully the characteristics of the free surface with synthetic and real-ground observations. The major advantage of the presented 4DEnVar strategies lies in the fact that we don't have to construct the adjoint procedure in opposition to the 4DVar approach. In general, the ensemble-based methods provided good results in terms of RMSE for partial and complete observations without the use of the adjoint procedure. In return, these techniques require a significant number of ensemble members to have comparable results with the 4DVar technique. The parallelization of the generation of the ensemble members also yields a computational cost comparable to the 4DVar. While the parallelization can be easily achieved for simple dynamical models, this task can be tedious for a DNS or even a LES model which require themselves a parallelization by domain decomposition.

Reconstruction of a turbulent three dimensional flow

The reconstruction of the three dimensional three components of a turbulent flow was at the heart of this thesis. In chapter 4, we carried out the reconstruction of the three dimensional three components velocity fields at the downstream of a cylinder wake at Reynolds 300. In this chapter, we combined the parallelized highly-accurate DNS code Incompact3d and a set of orthogonal-plane stereo PIV obserations. This study follows the work of Gronskis et al. (2013) who performed successfully the reconstruction in a two dimensional context.

First and foremost, we emphasize on the difficult task of the adjoint construction. Indeed, we had to reach the double objective of conserving the accuracy and the parallelized structure of the original Incompact3d code, as laid out in § 4.2.

Once we validated the adjoint procedure, we carried out a first reconstruction in an ideal context. In § 4.3, we assessed the ability of the 4DVar code to retrieve a reference flow from an initial noisy and biased flow. We encountered difficulties to find a compromise between the size of the domain (an implicitly the computational cost) and the accuracy of the flow with respect to a reference run in a larger domain. While the DNS code alone is capable to denoise the initial noise, the 4DVar successfully corrected the initial flow bias. The tuning of the observation error covariance matrix also improved the reconstruction and paved the way for future improvements.

Finally, we performed in § 4.4.3 the 4DVar with a set of orthogonal-plane stereo PIV observations. We draw the reader's attention on the difficulty of this challenge. Indeed, despite the quality of the DNS code and the DNS-like resolution of the PIV measurements, the passage from a 2D reconstruction to the 3D reconstruction is a delicate task. We described in § 4.4.3 the construction of three dimensional observations and their associated covariance error matrix.

Perspective

In light of the conclusions of this work, we lay out various subjects for further research.

Preconditioning model

In the first place, the incremental variational method could be enhanced by implementing a preconditioning technique similar to § 3.2.1 to avoid the ill-posed nature of the system of interest. We can also improve the variational method by updating the background error covariance matrix B by injecting the dependence of each controlled variables from the spatial correlation. These lines of research can be easily implemented for simple dynamics and be compared to hybrid data assimilation techniques in the same vein as the work carried out in chapter 3.

Control of the boundary conditions

In chapter 4 we achieved the correction of the flow by controlling the initial condition and the inflow at each time step. In the first step, a short term improvement to the presented results consists in implementing a temporal average on the inflow

condition similarly to Gronskis et al. (2013). As further study in § 4.3.2 revealed that the boundary conditions and the incomplete initialization of the DNS can lead to additional errors. While these errors remained in a reasonable range, at a higher Reynolds they are likely to be amplified. This situation could be avoided by including the control of the initial pressure field and every boundary conditions of the domain into the assimilation process.

Towards three dimensional observation

Despite the possibility to reconstruct a turbulent flow with orthogonal-plane stereo PIV observations, it is preferable to have a three dimensonal representation of the observation. During the last decade, experimental fluid dynamics community head to 3D PIV techniques. While the tomo-PIV measurements are only capable to retrieve a very small domain (40 mm $\times$ 40 mm $\times$ 10mm) (Elsinga et al., 2006), recent studies are able to retrieve cubic-meter sized volume (Kuhn et al., 2011; Scarano et al., 2015). These observations are particularly sparse in space and could be coupled with an LES model in the future.

Annexe A

BLUE

We seek for the gain matrix K involved in the analysis state $\mathbf{x}^a$ given by (2.9) and recalled below

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{K}(\boldsymbol{\mathcal{Y}} - \mathbf{H}\mathbf{x}^b)$$

The background error is defined by $\boldsymbol{\eta} = \mathbf{x}^t - \mathbf{x}^b$ associated with the background error covariance matrix

$$\mathbf{B} = \mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^T), \tag{A.1}$$

and the observation error $\boldsymbol{\varepsilon} = \boldsymbol{\mathcal{Y}} - \mathbf{H}\mathbf{x}^t$ associated with the observation error covariance matrix

$$\mathbf{R} = \mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T). \tag{A.2}$$

The gain matrix minimizes the variance $E(||\boldsymbol{\varepsilon}^a||^2)$ by using the information provided by the covariance matrices (A.1) and (A.2). The cost function of interest reads

$$J_{BLUE} = E(||\mathbf{x}^a - \mathbf{x}^t||^2) = \int (\mathbf{x}^b - \mathbf{x}^t + \mathbf{K}(\boldsymbol{\mathcal{Y}} - \mathbf{H}\mathbf{x}^b))^T (\mathbf{x}^b - \mathbf{x}^t + \mathbf{K}(\boldsymbol{\mathcal{Y}} - \mathbf{H}\mathbf{x}^b)) P(\mathbf{x}|\boldsymbol{\mathcal{Y}}) dx,$$

and we cancel ints gradient with respect to $\mathbf{K}$:

$$\begin{aligned}
&\frac{\partial J_{BLUE}}{\partial \mathbf{K}} = \int (\mathbf{K}(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta}) - \boldsymbol{\eta})(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})^T \mathbf{P}(\mathbf{x}|\boldsymbol{\mathcal{Y}}) dx = 0, \\
&\Leftrightarrow \mathbb{E}\left((\mathbf{K}(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta}) - \boldsymbol{\eta})(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})^T\right) = 0, \\
&\Leftrightarrow \mathbf{K}\mathbb{E}\left((\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})^T\right) - E(\boldsymbol{\eta}(\boldsymbol{varepsilon} + \mathbf{H}\boldsymbol{\eta})^T\big) = 0, \\
&\Leftrightarrow \mathbf{K} = (\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})^T) \left(\mathbb{E}((\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})(\boldsymbol{\varepsilon} + \mathbf{H}\boldsymbol{\eta})^T\right)^{-1}, \\
&\mathbf{K} = (\mathbf{H}\mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta})^T) \left(\mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T) + \mathbf{H}\mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^T)\mathbf{H}^T\right)^{-1}.
\end{aligned}$$

We then deduce the analysis covariance matrix $\mathbf{A}$

$$\begin{aligned}
&\mathbf{A} = E((\mathbf{x}^a - \mathbf{x}^t)(\mathbf{x}^a - \mathbf{x}^t)), \\
&\Leftrightarrow \mathbf{A} = \mathbf{B} - \mathbf{B}\mathbf{H}^T(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{B}
\end{aligned}$$

Annexe B

Time integration discretization schemes

We recall the general discretization of the time advancement of (4.2) which deduces the state u^{k+1} at time t_{k+1} from the previous state u^k at time t_k,

$$\begin{array}{lcl} \frac{u^*-u^k}{\Delta t} & = & a_k F^k + b_k F^{k-1} + c_k F^{k-2} - g_k \nabla \tilde{p}^k, \\ \frac{u^{**}-u^*}{\Delta t} & = & g_k \nabla \tilde{p}^k, \\ \frac{u^{k+1}-u^{**}}{\Delta t} & = & -g_k \nabla \tilde{p}^{k+1}, \end{array} \quad . \tag{B.1}$$

We specify the discretization scheme by fixing the coefficient triplet $\{a_k, b_k, c_k\}_{k=1,\cdots,n_k}$ (where $g_k = a_k + b_k + c_k$ for simplicity) on n_k sub-time steps with $t_1 = t_n$ and $t_{n_k} = t_{n+1}$ ($\Delta t = t_{n+1} - t_n$ being the full time step). The discretization schemes available in Incompact3d are the Runge Kutta methods at order 3 (RK3) and 4 (RK4), and the Adams-Bashford methods at order 2 (AB2) and 3 (AB3). The values attributed to each coefficient for each technique are given by :

AB2 $n_k = 1$
$(a_1,b_1,c_1) = (3/2\ ,-1/2,0)$

AB3 $n_k = 1$
$(a_1,b_1,c_1) = (23/12\ ,-16/12,5/12)$

RK3 (Williamson, 1980) $n_k = 3$
$(a_1,b_1,c_1) = (8/15\ ,0,0)$
$(a_2,b_2,c_2) = (5/12\ ,-17/60,0)$
$(a_3,b_3,c_3) = (3/4\ ,-5/12,0)$

RK4 (Kennedy et al., 2000) $n_k = 5$
$(a_1,b_1,c_1) = (0,0.1496590219993,0)$ and $g_1 = 0.1496590219993$
$(a_2,b_2,c_2) = (-0.4178904745\ ,0.1496590219993,0)$ and $g_2 = 0.220741935365$
$(a_3,b_3,c_3) = (-1.192151694643\ ,0.3792103129999,0)$ and $g_3 = 0.25185480577$
$(a_4,b_4,c_4) = (-1.697784692471\ ,0.8229550293869,0)$ and $g_4 = 0.33602636754$
$(a_5,b_5,c_5) = (-1.514183444257\ ,0.1530572479681,0)$ and $g_5 = 0.041717869325$

Annexe C

Automatic Differentiation

Automatic Differentiation (AD) is a set of techniques that aims at evaluating derivatives of a given function defined by a computer program. Regardless of the complexity of a given computer program, it consists in a sequence of simple instructions consisting in elementary arithmetic operations (additions, multiplications etc...) and/or elementary functions (exponential, cosinus etc...). As we know beforehand the derivative of each instruction, AD consists simply in applying the chain rule of derivative calculus repeatedly. Conceptually, AD is different from symbolic and numerical differentiation and takes advantage of both approaches. Indeed, the application of the chain rule avoids round-off errors encountered by finite differentiation and provides the most accurate derivatives of the original program. At the same time, it avoids the delicate task of performing a discretization by providing directly a discretized program.

We describe in this chapter the main principles of automatic differentiation, with the different computational strategies. We then describe the implementation techniques and present the AD tool used to ease the implementation. Finally, we illustrate the AD technique on a simple example. The reader can find further information in Hascoët (2006, 2012).

C.1 Automatic Differentiation principles

Let's consider a continuous function $f : \mathbb{R}^m \to \mathbb{R}^n$ which can be decomposed by a sequence of k functions such as $f = f_p \circ f_{p-1} \circ \cdots \circ f_1$. The value of f at an initial value $x := x_0 \in \mathbb{R}^m$ is given by $f(x) = f_p(x_{p-1}) = f_{p-1} \circ f_{p-1}(x_{p-2}) = f_{p-1} \circ f_{p-1}(x_{p-2}) \circ \cdots \circ f_1(x_0)$. The function f is implemented as a program `P` composed of a sequence of p instructions such that `P = {`$\mathtt{I}_1$ `; ... ;` $\mathtt{I}_{p-1}$ `;` $\mathtt{I}_p$`}`. Our aim consists in defining the differential tangent and adjoint program of `P` corresponding to the function f with respect to $x \in \mathbb{R}^m$. According to the chain rule, the differential tangent associated to f with respect to x is given by :

$$f'(x) = f'_k(x_{p-1}) \cdot f'_{p-1}(x_{p-2}) \cdots f'_1(x_0) \tag{C.1}$$

The derivative $f'(x)$ is too large and requires expensive computation resources. We can counter this issue by introducing the differential vector $\dot{x}$ such that :

$$\dot{y} = f'(x) \cdot \dot{x} = f'_k(x_{p-1}) \cdot f'_{p-1}(x_{p-2}) \cdots f'_1(x_0) \cdot \dot{x} \tag{C.2}$$

Instead of computing each matrix f'_k of size n for $k \in \{1, \cdots, p\}$ and deduce $f'(x) \cdot \dot{x}$, we compute each vector of size $n \times m$. This is done by computing (C.2) from right to left : we first compute the vector $f'_1(x_0) \cdot \dot{x}$, then $f'_2(x_1) \cdot (f'_1(x_0) \cdot \dot{x})$ and finally deduce $\dot{y} = f'(x) \cdot \dot{x}$.

The reverse mode is deduced from the chain rule by introducing the differential adjoint vector $\bar{y} \in \mathbb{R}^n$ and the adjoint derivatives $f_p^{\#}$ for $p \in \{1, ..., k\}$ such that :

$$\bar{x} = f^*(x) \cdot \bar{y} = f_1^*(x_0) \cdots f_{p-1}^*(x_{p-2}) \cdot f_p^*(x_{p-1}) \cdot \bar{y} \tag{C.3}$$

As in the tangent mode, (C.3) is also computed from right to left which rises an important issue. Indeed, for the first step calculation for instance, we need the value of x_{p-1} which, we recall, is obtained by $x_{p-1} = f_{p-1}(x_{p-2}) \circ \cdots \circ f_1(x_0)$. Thus, at each step of the reverse mode computation, we require all the values of x_p for $p \in \{1, ..., k\}$.

There are two main strategies to tackle the reverse mode issue :

Store all (SA) consists in saving all the values of $\{x_p; p = 1, \cdots, k-1\}$ and invoke each intermediate value when it is required. This technique is illustrated by figure C.1. While this strategy costs twice the computation time of the source program `P`, it clearly requires a tremendous amount of memory which can't be provided when we consider very large size problems. We can deal with this memory problem by a checkpoint strategy which consists in applying the SA strategy on some parts of the program and recompute from the beginning of the subprograms as illustrated by figure C.1.

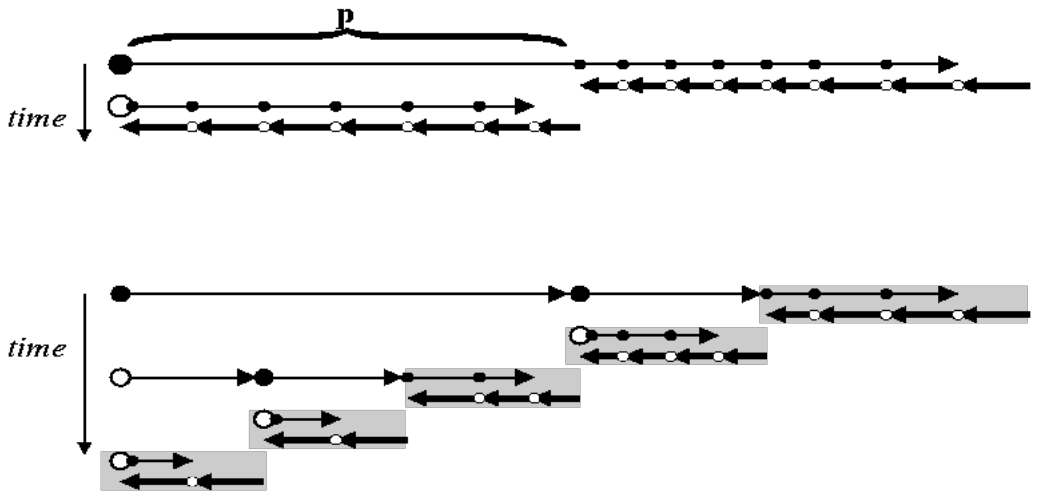

FIGURE C.1 – SA strategy (top) stores all the intermediate values (black points) at the end of each instructions $\{I_k; k = 1, \cdots, p-1\}$. In the reverse mode, each values are invoked and used. Checkpointing SA strategy (bottom) consists in applying the SA strategy on small blocs of the program.

Recompute all (RA) consists in recomputing all the values of $\{x_p; p = 1, \cdots, k-1\}$ when they are needed, as illustrated by figure C.2. In opposition to the SA strategy, this strategy doesn't require any additional storage however the computational cost is clearly colossal. As in the SA strategy, there is a checkpointing RA strategy which also consists in fragmenting the whole program and

saving the complete state of the problem, called snapshot, and recompute from the closest snapshot instead of the initial state. This strategy is illustrated by figure C.2.

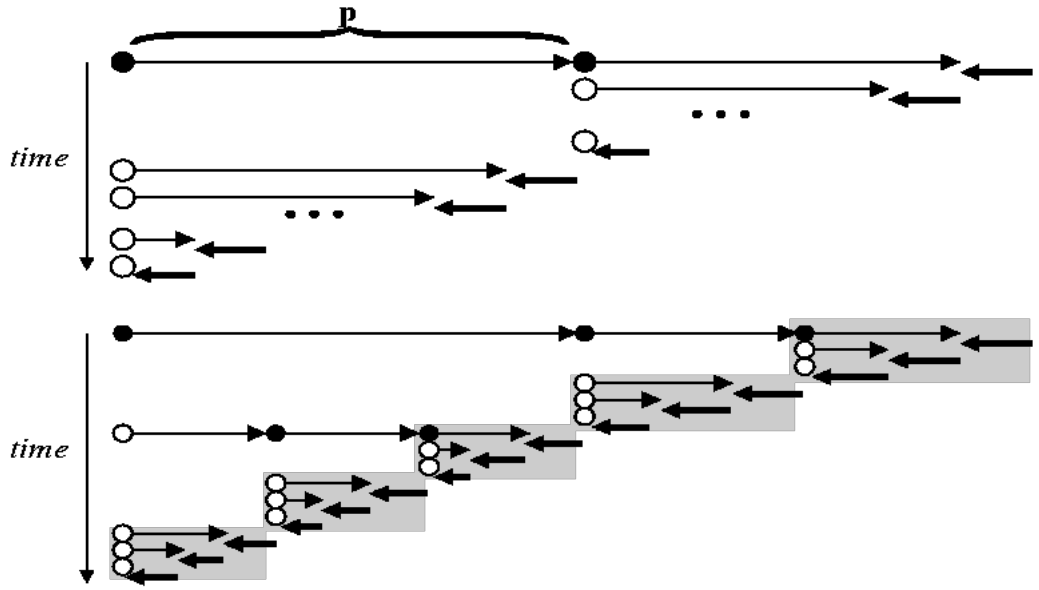

FIGURE C.2 – RA strategy (top) performs all the intermediate instructions from the initial value x (black points) to obtain the required value. Checkpointing RA strategy (bottom) recomputes the intermediate values from each snapshot (black points)

In the framework of this thesis, we are led to opt for the checkpointing SA strategy. This choice stems from the implementation of the AD as discussed in the following section.

C.1.1 Automatic Differentiation implementation

The numerical implementation of the AD can be achieved by applying one of the following strategies :

Source Code Transformation (SCT) consists in replacing the source code by an automatically generated from the source code. This new code includes additional variables, arrays and data structures that hold the derivatives and all the necessary new instructions to compute them. It is also easier to implement the reverse mode with this strategy. On the downside, the transformation of the source code is more complex as the complexity of the source code increases. Therefore, in practice we resort to AD tools that performs automatically and rapidly the transformation. This strategy can also be implemented for all programming languages and the resulting program can be compiled into an efficient code.

Operator Overloading (OO) consists in telling the compiler that each real number is replaced by a pair of real numbers, the second holding the differential. This strategy is obviously possible for a source code written in a language

which supports operator overloading. Each elementary operation on real numbers is overloaded, in other words, it is replaced by a new one. working on pairs of reals, that computes the value and its differential. In opposition to source code transformation, the original program is virtually unchanged since everything is done at compile time. However, the resulting program will run slowly because it constantly builds and destroys pairs of real numbers. Moreover, it is very hard to implement the reverse mode with this strategy.

In the framework of this thesis, we turn to the SCT strategy and perform the code transformation with the help of the AD tool TAPENADE (Hascoët, 2004; Hascoët and Pascual, 2013). TAPENADE, which is successor of Odyssée (Faure and Papegay, 1998), was developed by the Tropics team (Inria Sophia Antipolis) since 1999. This tool is specialized among others in AD reverse and tangent mode and adopts a store-all strategy with the possibility of checkpointing on calls. This tool is implemented at 90% Java and 10% C and supports code sources written in Fortran (Fortran95, Fortran177 and older). TAPENADE was previously employed in several such as optimum design problems citepHascoet2003,Dervieux2006, oceanography (Ferron and Hascoët, 2006; Tber et al., 2007) and was also used for the reconstruction of a 2D turbulent flow by a variational assimilation technique (Gronskis et al., 2013). In the framework of this thesis, we recall that we aim at reconstruction the state of a 2D flow chapter 3 and a 3D turbulent flow chapter 4. On the one hand, the 2D application is based on a homemade MATLAB code, which can be easily translated into a Fortran code ; and in another hand, the 3D application is based on the DNS code Incompact3d which is entirely written in Fortran. Therefore, TAPENADE is a good candidate for both applications and was naturally selected.

In practice, TAPENADE can be used directly as a web server by a distant user, without any need for a local installation at the following address :

`http://tapenade.inria.fr:8080/tapenade/index.jsp`

However, it is usually more convenient to download and install the tool on a local computer from the ftp server

`ftp://ftp-sop.inria.fr/tropics/tapenade`

and call the tool directly as a command with arguments. The reader can refer to TAPENADE's user guide Hascoët and Pascual (2004).

C.1.2 Automatic Differentiation example using TAPENADE

Many examples simple examples can be found in the litterature, in this section we illustrate the construction of the tangent and adjoint of a homemade short code source written in Fortran90. In this example, we consider a subroutine `foo2`, defined by 2, nested in a main subroutine `foo1`, defined by 6, and aim at differentiating the latter with respect to the active variables `m1` $\in \mathbb{R}^n$ and `v4` $\in \mathbb{R}$.
where,

These source codes are automatically transformed by TAPENADE and are verified by the validations tests § 2.4.7. The tangent mode results to 11 and 4, where

Algorithm 30 foo1(m1,v4)

```
for i=1,n,2 do
    v1 = m1(i)**2
    v2 = cos(m1(i+1))
    call foo2(v1,v2,v3)
    v4 = v4+v3
end for
```

Algorithm 31 foo2(v1,v3,v4)

```
v2 = 2*v1 + 5
v4 = v2 + p1*v3/v2
```

Algorithm 32 foo1_d(m1,v4)

```
v4d = 0.0
v3d = 0.0
for i=1,n,2 do
    v1d = 2*m1(i)*m1d(i)
    v1 = m1(i)**2
    -(m1d(i+1)*SIN(m1(i+1)))
    v2 = cos(m1(i+1))
    call foo2_d(v1, v1d, v2, v2d, v3, v3d)
    v4d = v4d + v3d
    v4 = v4+v3
end for
```

Algorithm 33 foo2_d(v1,v3,v4)

```
v2d = 2*v1d
v2 = 2*v1 + 5
v4d = v2d + (p1*v3d*v2-p1*v3*v2d)/v2**2
v4 = v2 + p1*v3/v2
```

the variables ending by d denotes the differential tangent variable which are always initialized at the begining of the code for security :

We recall that the reverse mode use the checkpointing SA strategy. As illustrated previously, this procedures consists in a forward computation of the source code and a backward computation to deduce the differential adjoints. The checkpointing is performed by repeatedly calling PUSH procedures which push the value at a given stage of the source algorithm during the forward procedure and POP procedures which reads the first value of the pile. The result of the reverse mode is given by 26 and 6.

Algorithm 34 foo1_b(m1,v4)

```
! Forward procedure
for i=1,n,2 do
    v1 = m1(i)**2
    v2 = cos(m1(i+1))
    ! Save intermediate values for the adjoint process
    call PUSH(v1)
    call PUSH(v2)
    call PUSH(v3)
    call foo2(v1,v2, v3)
end for
! Backward procedure
m1b = 0.0
v3b = 0.0
for i=n-mod(n-1, 2),1,-2 do
    v3b = v3b + v4b
    ! Invoke intermediate values
    call POP(v1)
    call POP(v2)
    call POP(v3)
    v1b = 0.0
    v2b = 0.0
    call foo2_b(v1, v1b, v2, v2b, v3, v3b)
    m1b(i+1) = m1b(i+1) - SIN(m1(i+1))*v2b
    m1b(i) = m1b(i) + 2*m1(i)*v1b
end for
v4b = 0.0
```

Algorithm 35 foo2_b(v1,v3,v4)

```
v2 = 2*v1 + 5
tempb = p1*v4b/v2
v2b = v4b - v3*tempb/v2
v3b = tempb
v1b = 2*v2b
v4b = 0.0
```

Annexe D

Discrete adjoint of auxiliary procedures used in SWE

We describe in the tangent and adjoint procedures corresponding to each auxiliary routines used within the shallow water code SWE. We recall in algorthm 36 the structure of the original SWE code.

Algorithm 36 SWE(h^0,u^0,v^0)

Initialization: $n = 0$, h^0, u^0, v^0
  while $n < N$ **do**
    call u2q(h^n,u^n,v^n,q_x^n,q_y^n)
    call RK3(h^n,q_x^n,q_y^n,h^{n+1},q_x^{n+1},q_y^{n+1})
    call q2u(h^{n+1},u^{n+1},v^{n+1},q_x^{n+1},q_y^{n+1})
    save (h^{n+1},u^{n+1},v^{n+1})
    $n = n + 1$
  end while
  return h^N, u^N, v^N

We will provide the numerical code or the equivalent matrix formulation for each routine laid out in this chapter. In terms of the notations, we inherit from the formalism imposed by TAPENADE such that :

- the tangent procedures are ended by the subscript _d, the differential tangent variable associated to a given variable $\mathbf{x}$ are denoted $\dot{\mathbf{x}}$,
- the adjoint procedures are ended by the subscript _b, the differential adjoint variable associated to a given variable $\mathbf{x}$ are denoted $\bar{\mathbf{x}}$.

D.1 Adjoint of the boundary conditions procedure

D.1.1 Contribution of the adjoint variable in the east direction

Algorithm 37 BC_east_b($\bar{h}_E$, $\bar{u}_E$, $\bar{v}_E$)

Define value within the domain
for i=1,nx **do**
  for j=1,ny **do**
    $\bar{h}(i,j) = \bar{h}(i,j) + \bar{h}_E(i,j)$
    $\bar{u}(i,j) = \bar{u}(i,j) + \bar{u}_E(i,j)$
    $\bar{v}(i,j) = \bar{v}(i,j) + \bar{v}_E(i,j)$
  end for
end for
Sum values from the ghost cells into the boundaries of the domain
for i=1,nx **do**
  $\bar{h}(i,ny) = h(i,ny) + \bar{h}_E(i,ny+1)$
  $\bar{u}(i,ny) = u(i,ny) + \bar{u}_E(i,ny+1)$
  $\bar{v}(i,ny) = v(i,ny) - \bar{v}_E(i,ny+1)$
end for
for j=1,ny **do**
  $\bar{h}(nx,j) = \bar{h}(nx,j) + \bar{h}_E(nx+1,j)$
  $\bar{u}(nx,j) = \bar{u}(nx,j) - \bar{u}_E(nx+1,j)$
  $\bar{v}(nx,j) = \bar{v}(nx,j) + \bar{v}_E(nx+1,j)$
end for
$\bar{h}(nx,ny) = \bar{h}(nx,ny) + \bar{h}_E(nx+1,ny+1)$
$\bar{u}(nx,ny) = \bar{u}(nx,ny) - \bar{u}_E(nx+1,ny+1)$
$\bar{v}(nx,ny) = \bar{v}(nx,ny) - \bar{v}_E(nx+1,ny+1)$
return $\bar{h}, \bar{u}, \bar{v}$

D.1.2 Contribution of the adjoint variable in the west direction

Algorithm 38 BC_west_b($\bar{h}_W$, $\bar{u}_W$, $\bar{v}_W$)

Define value within the domain
for i=1,nx **do**
  for j=1,ny **do**
    $\bar{h}(i,j) = \bar{h}(i,j) + \bar{h}_W(i+1,j)$
    $\bar{u}(i,j) = \bar{u}(i,j) + \bar{u}_W(i+1,j)$
    $\bar{v}(i,j) = \bar{v}(i,j) + \bar{v}_W(i+1,j)$
  end for
end for
Sum values from the ghost cells into the boundaries of the domain
for i=1,nx **do**
  $\bar{h}(i,ny) = \bar{h}(i,ny) + \bar{h}_W(i+1,ny+1)$
  $\bar{u}(i,ny) = \bar{u}(i,ny) + \bar{u}_W(i+1,ny+1)$
  $\bar{v}(i,ny) = \bar{v}(i,ny) - \bar{v}_W(i+1,ny+1)$
end for
for j=1,ny **do**
  $\bar{h}(1,j) = \bar{h}(1,j) + \bar{h}_W(1,j)$
  $\bar{u}(1,j) = \bar{u}(1,j) - \bar{u}_W(1,j)$
  $\bar{v}(1,j) = \bar{v}(1,j) + \bar{v}_W(1,j)$
end for
$\bar{h}(1,ny) = \bar{h}(1,ny) + \bar{h}_W(1,ny+1)$
$\bar{u}(1,ny) = \bar{u}(1,ny) - \bar{u}_W(1,ny+1)$
$\bar{v}(1,ny) = \bar{v}(1,ny) - \bar{v}_W(1,ny+1)$
return $\bar{h}, \bar{u}, \bar{v}$

D.1.3 Contribution of the adjoint variable in the north direction

Algorithm 39 BC_north_b($\bar{h}_N$, $\bar{u}_N$, $\bar{v}_N$)

Define value within the domain
for i=1,nx **do**
  for j=1,ny **do**
    $\bar{h}(i,j) = \bar{h}(i,j) + \bar{h}_N(i,j)$
    $\bar{u}(i,j) = \bar{u}(i,j) + \bar{u}_N(i,j)$
    $\bar{v}(i,j) = \bar{v}(i,j) + \bar{v}_N(i,j)$
  end for
end for
Sum values from the ghost cells into the boundaries of the domain
for i=1,nx **do**
  $\bar{h}(i,ny) = h(i,ny) + \bar{h}_N(i,ny+1)$
  $\bar{u}(i,ny) = u(i,ny) + \bar{u}_N(i,ny+1)$
  $\bar{v}(i,ny) = v(i,ny) - \bar{v}_N(i,ny+1)$
end for
for j=1,ny **do**
  $\bar{h}(nx,j) = \bar{h}(nx,j) + \bar{h}_N(nx+1,j)$
  $\bar{u}(nx,j) = \bar{u}(nx,j) - \bar{u}_N(nx+1,j)$
  $\bar{v}(nx,j) = \bar{v}(nx,j) + \bar{v}_N(nx+1,j)$
end for
$\bar{h}(nx,ny) = \bar{h}(nx,ny) + \bar{h}_N(nx+1,ny+1)$
$\bar{u}(nx,ny) = \bar{u}(nx,ny) - \bar{u}_N(nx+1,ny+1)$
$\bar{v}(nx,ny) = \bar{v}(nx,ny) - \bar{v}_N(nx+1,ny+1)$
return $\bar{h}, \bar{u}, \bar{v}$

D.1.4 Contribution of the adjoint variable in the south direction

Algorithm 40 BC_south_b($\bar{h}_S$, $\bar{u}_S$, $\bar{v}_S$)

Define value within the domain
for i=1,nx **do**
  for j=1,ny **do**
    $\bar{h}(i,j) = \bar{h}(i,j) + \bar{h}_S(i,j+1)$
    $\bar{u}(i,j) = \bar{u}(i,j) + \bar{u}_S(i,j+1)$
    $\bar{v}(i,j) = \bar{v}(i,j) + \bar{v}_S(i,j+1)$
  end for
end for
Sum values from the ghost cells into the boundaries of the domain
for i=1,nx **do**
  $\bar{h}(i,1) = h(i,1) + \bar{h}_S(i,1)$
  $\bar{u}(i,1) = u(i,1) + \bar{u}_S(i,1)$
  $\bar{v}(i,1) = v(i,1) - \bar{v}_S(i,1)$
end for
for j=1,ny **do**
  $\bar{h}(nx,j) = \bar{h}(nx,j) + \bar{h}_S(nx+1,j+1)$
  $\bar{u}(nx,j) = \bar{u}(nx,j) - \bar{u}_S(nx+1,j+1)$
  $\bar{v}(nx,j) = \bar{v}(nx,j) + \bar{v}_S(nx+1,j+1)$
end for
$\bar{h}(nx,1) = \bar{h}(nx,1) + \bar{h}_S(nx+1,1)$
$\bar{u}(nx,1) = \bar{u}(nx,1) - \bar{u}_S(nx+1,1)$
$\bar{v}(nx,1) = \bar{v}(nx,1) - \bar{v}_S(nx+1,1)$
return $\bar{h}, \bar{u}, \bar{v}$

D.1.5 Discrete adjoint of the BC routine

Given the adjoint procedures presented in this section, the discrete adjoint $(\bar{h}, \bar{u}, \bar{v})$ of the boundary conditions are given by algorithm 41.

Algorithm 41 BC_b($\bar{h}$, $\bar{u}$, $\bar{v}$, $\bar{h}_W$, $\bar{u}_W$, $\bar{v}_W$, $\bar{h}_E$, $\bar{u}_E$, $\bar{v}_E$, $\bar{h}_N$, $\bar{u}_N$, $\bar{v}_N$, $\bar{h}_S$, $\bar{u}_S$, $\bar{v}_S$)

[$\bar{h}$, $\bar{u}$, $\bar{v}$]= BC_east_b($\bar{h}_E$, $\bar{u}_E$, $\bar{v}_E$)
[$\bar{h}$, $\bar{u}$, $\bar{v}$]= BC_west_b($\bar{h}_W$, $\bar{u}_W$, $\bar{v}_W$)
[$\bar{h}$, $\bar{u}$, $\bar{v}$]= BC_north_b($\bar{h}_N$, $\bar{u}_N$, $\bar{v}_N$)
[$\bar{h}$, $\bar{u}$, $\bar{v}$]= BC_south_b($\bar{h}_S$, $\bar{u}_S$, $\bar{v}_S$)

D.2 Adjoint of the routines involved in the Roe solver

The Roe routine which solves the 1D Riemann problem from the left and right state $\mathbf{x}_L = (h_L, u_L, v_L)$ and $\mathbf{x}_R = (h_R, u_R, v_R)$ is described by algorithm 42. We describe in this section the adjoint procedures of the auxiliary routines Priestley, eigenvalues, eigenvectors, fluxH and fluxV called from the Roe routine.

Algorithm 42 Roe(h_L, u_L, v_L, h_R, u_R, v_R, $flag$)

! Preprocessing
$\mathbf{w}_L$ = Priestley (h_L, u_L, v_L)
$\mathbf{w}_R$ = Priestley (h_R, u_R, v_R)
$\Delta\mathbf{w} = \mathbf{w}_L - \mathbf{w}_R$
$\tilde{\mathbf{w}} = \frac{1}{2}(\mathbf{w}_L + \mathbf{w}_R)$
! Characteristics of the Roe matrix $\tilde{A}(X_R, X_L)$
call eigenvalues($\tilde{\mathbf{w}}$,e_1, e_2, e_3)
call eigenvectors($\tilde{\mathbf{w}}$,r_1, r_2, r_3)
call coefficients($\tilde{\mathbf{w}}$,$\Delta\mathbf{w}$,α_1, α_2, α_3)
! Compute the left and right fluxes
if $flag$ **then**
  F_L = fluxH (h_L, u_L, v_L)
  F_R = fluxH (h_R, u_R, v_R)
else
  F_L = fluxV (h_L, u_L, v_L)
  F_R = fluxV (h_R, u_R, v_R)
end if
return $\frac{1}{2}(F_L + F_R) - \frac{1}{2}\sum_{k=1}^{3} \alpha_k |e_k| \mathbf{r}_k$

D.2.1 Adjoint of the Preistley algorithm

In order to implement the Roe solver to the 2D SWE model, Priestley (1987) introduced parameter vectors which are determined by the left and right state vectors $\mathbf{x}_L$ and $\mathbf{x}_R$ (see § 3.3.2). Given an input $\mathbf{x} = (h, u, v)$, we define the Preistley routine by the following function

$$\texttt{Preistley}(\mathbf{x}) = (\sqrt{h}, u\sqrt{h}, v\sqrt{h})$$

The tangent procedure with respect to a linearization state $\mathbf{x} = (h, u, v)$ can be written into the following matrix form,

$$\dot{\mathbf{w}} = \frac{1}{2\sqrt{h}} \begin{pmatrix} h & 0 & 0 \\ u & 2h & 0 \\ v & 0 & 2h \end{pmatrix} \dot{\mathbf{x}},$$

and we deduce straightforwardly the matrix form of the corresponding adjoint procedure,

$$\bar{\mathbf{x}} = \frac{1}{2\sqrt{h}} \begin{pmatrix} h & u & v \\ 0 & 2h & 0 \\ 0 & 0 & 2h \end{pmatrix} \bar{\mathbf{w}}.$$

D.2.2 Adjoint of the eigenvalues of $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$

The eigenvalues e_1, e_2, e_3 of the linearized Jacobian $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$ are computed from the input variable $\tilde{\mathbf{w}} := \frac{1}{2}(\mathbf{w}_L + \mathbf{w}_R)$ such that

$$e_1 = \frac{\tilde{w}_2}{\tilde{w}_1} d + \frac{\tilde{w}_3}{\tilde{w}_1}(1-d) - \tilde{w}_1, \tag{D.1}$$

$$e_2 = \frac{\tilde{w}_2}{\tilde{w}_1} d + \frac{\tilde{w}_3}{\tilde{w}_1}(1-d), \tag{D.2}$$

$$e_3 = \frac{\tilde{w}_2}{\tilde{w}_1} d + \frac{\tilde{w}_3}{\tilde{w}_1}(1-d) + \tilde{w}_1, \tag{D.3}$$

where $d = 1$ for the horizontal fluxes and $d = 0$ for the vertical fluxes. In the following, we provide the tangent and adjoint procedures of the `eigenvalues` routine with respect to a given linearization state $\mathbf{w} := (w_1, w_2, w_3)$.

The tangent procedure can be written into the following matrix form

$$\dot{\mathbf{e}} = \frac{1}{w_1^2} \begin{pmatrix} -w_2 d - w_3(1-d) + w_1 & w_1 & w_1 \\ -w_2 d - w_3(1-d) & w_1 & w_1 \\ -w_2 d - w_3(1-d) + w_1 & w_1 & w_1 \end{pmatrix} \dot{\mathbf{w}},$$

and the corresponding adjoint procedure simply reads,

$$\bar{\mathbf{w}} = \frac{1}{w_1^2} \begin{pmatrix} -w_2 d - w_3(1-d) + w_1 & -w_2 d - w_3(1-d) & -w_2 d - w_3(1-d) + w_1 \\ w_1 & w_1 & w_1 \\ w_1 & w_1 & w_1 \end{pmatrix} \bar{\mathbf{e}}.$$

D.2.3 Adjoint of the eigenvectors of $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$

Similarly to the eigenvalues, the eigenvectors r_1, r_2, r_3 of the linearized Jacobian $\tilde{A}(\mathbf{x}_R, \mathbf{x}_L)$ are computed from the input variable $\tilde{\mathbf{w}} := \frac{1}{2}(\mathbf{w}_L + \mathbf{w}_R)$. We recall below the expression of each eigenvectors,

$$\mathbf{r}_1 = \begin{pmatrix} w_1 \\ w_2 - w_1\sqrt{w_1^2} d \\ w_3 - w_1\sqrt{w_1^2}(1-d) \end{pmatrix}, \quad \mathbf{r}_2 = \begin{pmatrix} 0 \\ 0 \\ w_1 \end{pmatrix}, \quad \mathbf{r}_3 = \begin{pmatrix} w_1 \\ w_2 + w_1\sqrt{w_1^2} d \\ w_3 + w_1\sqrt{w_1^2}(1-d) \end{pmatrix},$$

where $d = 1$ for the horizontal fluxes and $d = 0$ for the vertical fluxes. We provide the tangent and adjoint procedures with respect to a given linearization state $\mathbf{w} := (w_1, w_2, w_3)$.

The tangent and adjoint procedures corresponding to the calculation of the eigenvector $\mathbf{r}_2$ are elementary, therefore we only provide the formulation for the eigenvectors $\mathbf{r}_1$ and $\mathbf{r}_3$ which are summarized by :

$$\mathbf{r}_{1,3} = \begin{pmatrix} w_1 \\ w_2 \pm w_1\sqrt{w_1^2 d} \\ w_3 \pm w_1\sqrt{w_1^2(1-d)} \end{pmatrix}.$$

The tangent procedure can be written into the following matrix form

$$\dot{\mathbf{w}} = \begin{pmatrix} 1 & 0 & 0 \\ \pm\sqrt{w_1^2 d} & 1 & 0 \\ \pm\sqrt{w_1^2(1-d)} & 0 & 1 \end{pmatrix} \dot{\mathbf{r}}_{1,3},$$

and the adjoint procedure reads,

$$\bar{\mathbf{r}}_{1,3} = \begin{pmatrix} 1 & \pm\sqrt{w_1^2 d} & \pm\sqrt{w_1^2(1-d)} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \bar{\mathbf{w}}.$$

D.2.4 Adjoint of the nonlinear flux

We recall that the nonlinear flux of a given state $\mathbf{x} = (h, u, v)^T$ in the horizontal direction is given by $FH := F(\mathbf{x}) = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}$. The tangent of the horizontal flux is given by

$$\dot{FH} = \begin{pmatrix} u & h & 0 \\ u^2 + gh & 2hu & 0 \\ uv & hu & hv \end{pmatrix} \dot{\mathbf{x}},$$

and its adjoint is deduced by

$$\bar{\mathbf{x}} = \begin{pmatrix} u & u^2 + ghh & uv \\ h & 2hu & hu \\ 0 & 0 & hv \end{pmatrix} \overline{FH}.$$

Similarly, we recall that the nonlinear flux in the vertical direction is given by $FH := F(\mathbf{x}) = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}$ and $FV := G(\mathbf{x}) = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}$. The tangent of the horizontal flux is given by

$$\dot{FV} = \begin{pmatrix} u & 0 & h \\ uv & hv & hu \\ v^2 + gh & 0 & 2hv \end{pmatrix} \dot{\mathbf{x}},$$

and its adjoint is deduced by

$$\bar{\mathbf{x}} = \begin{pmatrix} u & uv & v^2 + gh \\ 0 & hv & 0 \\ h & hu & 2hv \end{pmatrix} \overline{FV}.$$

D.3 Adjoint for the conversion of non-conservative variables to conservative variables

The SWE routine requires two auxiliary routines u2q and q2u which transforms the non-conservative $\mathbf{x} = (h, u, v)^T$ into the conservative variable state $\mathbf{x} = (h, h.u, h.v)^T$ and vice versa. These routines are defined by elementary operations, therefore their corresponding tangent and adjoint routines are straightforwardly obtained with TAPENADE.

In the one hand, the u2q routine is defined by the elementary operations

$$q_x = hu, \tag{D.4}$$
$$q_y = hv, \tag{D.5}$$

its corresponding tangent procedure u2q_d is given by

$$\dot{q}_x = h\dot{u} + \dot{h}u, \tag{D.6}$$
$$\dot{q}_y = h\dot{v} + \dot{h}v, \tag{D.7}$$

and its corresponding adjoint procedure u2q_b is deduced by

$$\bar{h} = h + u\bar{q}_x + v\bar{q}_y \tag{D.8}$$
$$\bar{u} = h\bar{u} \tag{D.9}$$
$$\bar{v} = h\bar{v}. \tag{D.10}$$

In the one hand, the q2u routine is defined by the elementary operations

$$u = q_x/h, \tag{D.11}$$
$$v = q_y/h, \tag{D.12}$$

its corresponding tangent procedure q2u_d is given by

$$\dot{u} = \frac{h\dot{q}_x - \dot{h}q_x}{h^2}, \tag{D.13}$$
$$\dot{v} = \frac{h\dot{q}_y - \dot{h}q_y}{h^2}, \tag{D.14}$$

and its corresponding adjoint procedure q2u_b is deduced by

$$\bar{h} = \bar{h} - q_x\frac{\bar{q}_x}{h^2} - q_y\frac{\bar{q}_x}{h^2} \tag{D.15}$$
$$\bar{u} = \frac{\bar{q}_x}{h^2} \tag{D.16}$$
$$\bar{v} = \frac{\bar{q}_y}{h^2}. \tag{D.17}$$

Annexe E

Discrete adjoint of the compact schemes

The first and second order derivatives are computed for each direction, after the swap to the adequate pencil. For the sake of simplicity, we consider within this section that swapped to the X-pencil configuration and we aim at computing the first and second order derivatives of a given vector $\mathbf{f} := (f_i)_{n_x}$. We recall that the 6th order numerical scheme used to approximate the first order derivative $f'(x_i)$ at x_i reads

$$\alpha f'_{i-1} + f'_i + \alpha f'_{i+1} = a\frac{f_{i+1} - fi - 1}{2\Delta x} + b\frac{f_{i+2} - f_{i-2}}{4\Delta x}.$$

and the approximation of the second order derivative $f''(x_i)$ at x_i reads

$$\alpha f''_{i-1} + f''_i + \alpha f''_{i+1} = a\frac{f_{i+1} - 2f - i + f_{i-1}}{2\Delta x^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4\Delta x^2} + c\frac{f_{i+3} - 2f_i + f_{i-3}}{9\Delta x^2},$$

The main asset of this numerical scheme is the fact that it's a linear algebraic combination, therefore the adjont procedure can be easiely deduced. It is possible to write the compact schemes in the following matrix forms :

$$A_x\mathbf{f'} = \frac{1}{\Delta x}B_x\mathbf{f},$$

$$A'_x\mathbf{f''} = \frac{1}{\Delta x}B'_x\mathbf{f},$$

where the matrices A_x, B_x, A'_x and B'_x of size $n_x \times n_x$ depends on the selected boundary conditions. Hence, the adjoint procedure corresponding to the compact schemes are simply deduced by :

$$B_x^T\mathbf{f} = \frac{1}{\Delta x}A_x^T\mathbf{f'},$$

$$(B'_x)^T\mathbf{f} = \frac{1}{\Delta x}(A'_x)^T\mathbf{f''}.$$

Periodic condition conserves the order of the numerical scheme and is easy to implement. The values of f, f', f''at the boundaries are simply deduced by :

$$f_0 = f_{n_x}, f_{-1} = f_{n_x-1}, f'_0 = f'_{n_x}, f''_0 = f''_{n_x}.$$

Therefore the compact scheme can be formulated as :

$$A_x = \begin{pmatrix} 1 & \alpha & & & & & & \alpha \\ \alpha & 1 & \alpha & & & & & \\ & . & . & . & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & \alpha & 1 & \alpha \\ \alpha & & & & & & \alpha & 1 \end{pmatrix} \text{ and } B_x = \frac{1}{\Delta x} \begin{pmatrix} 0 & a & b & & & & -b & -a \\ -a & 0 & a & b & & & & -b \\ -b & -a & 0 & a & b & & & \\ & . & . & . & . & . & & \\ & & . & . & . & . & . & \\ & & & -b & -a & 0 & a & b \\ b & & & & -b & -a & 0 & a \\ a & b & & & & -b & -a & 0 \end{pmatrix}$$

Free slip condition has the same assets than the periodic condition and corresponds whether to the symmetric condition

$$f_0 = f_2, f_{-1} = f_3, f'_0 = f'_2, f''_0 = f''_2,$$

This boundary condition can be considered as a symetric or antisymetric condition according to the velocity component considered. If f is even, the matrix A_x and B_x reads :

$$A_x = \begin{pmatrix} 1 & 0 & & & & & & \alpha \\ \alpha & 1 & \alpha & & & & & \\ & . & . & . & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & \alpha & 1 & \alpha \\ \alpha & & & & & & 0 & 1 \end{pmatrix} \text{ and } B_x = \frac{1}{\Delta x} \begin{pmatrix} 0 & & & & & & & \\ -a & -b & a & b & & & & \\ -b & -a & 0 & a & b & & & \\ & . & . & . & . & . & & \\ & & . & . & . & . & . & \\ & & & -b & -a & 0 & a & b \\ b & & & & -b & -a & b & a \\ a & b & & & & & & 0 \end{pmatrix}$$

or the asymmetric condition

$$f_0 = -f_2, f_{-1} = -f_3, f'_0 = f'_2, f''_0 = -f''_2.$$

If f is odd, the matrix A_x and B_x reads :

$$A_x = \begin{pmatrix} 1 & 2\alpha & & & & & & \alpha \\ \alpha & 1 & \alpha & & & & & \\ & . & . & . & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & \alpha & 1 & \alpha \\ \alpha & & & & & & 2\alpha & 1 \end{pmatrix} \text{ and } B_x = \frac{1}{\Delta x} \begin{pmatrix} 0 & 2a & 2b & & & & & \\ -a & b & a & b & & & & \\ -b & -a & 0 & a & b & & & \\ & . & . & . & . & . & & \\ & & . & . & . & . & . & \\ & & & -b & -a & 0 & a & b \\ b & & & & -b & -a & b & a \\ a & b & & & & & 2b & 2a & 0 \end{pmatrix}$$

Open boundary condition doesn't make any assumptions on the flow outside the computational domain. Single sided formulations are used for the approximation of first derivatives and second derivatives for these types of boundaries using relations of the form :

$$f'_1 + 2f'_2 = \frac{1}{2\Delta x}(-5f_1 + 4f_2 + f_3),$$
$$f''_1 + 11f''_2 = \frac{1}{2\Delta x^2}(13f_1 - 27f_2 + 15f_3 - f_4),$$

that are third order accurate (Lele, 1992). At the adjacent nodes, because a three point formulation must be used Padé schemes are employed with forth-order accurate schemes :

$$\frac{1}{4}f'_1 + f'_2 + \frac{1}{4}f'_3 = \frac{3}{2}\frac{f_3 - f_1}{2\Delta x},$$
$$\frac{1}{10}f''_1 + f''_2 + \frac{1}{10}f''_3 = \frac{6}{5}\frac{f_3 - 2f_2 + f_1}{\Delta x}.$$

$$A_x = \begin{pmatrix} 1 & \alpha & & & & & & \alpha \\ \alpha & 1 & \alpha & & & & & \\ & . & . & . & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & . & . & . & \\ & & & & & \alpha & 1 & \alpha \\ \alpha & & & & & & \alpha & 1 \end{pmatrix} \text{ and } B_x = \frac{1}{\Delta x}\begin{pmatrix} 0 & a & b & & & & -b & -a \\ -a & 0 & a & b & & & & -b \\ -b & -a & 0 & a & b & & & \\ & . & . & . & . & . & & \\ & & . & . & . & . & . & \\ & & & -b & -a & 0 & a & b \\ b & & & & -b & -a & 0 & a \\ a & b & & & & -b & -a & 0 \end{pmatrix}$$

Annexe F

Discrete adjoint of `FFT_postproc`

We describe the construction of the `FFT_postproc` routine which intervenes in the `poisson` routine (see algorithm 25. This routine perfoms a one dimensional post processing applied in every direction. In this section, we consider the construction in an arbitrary direction for an input a vector in of size n. The input variable $sign$ indicates if we consider the forward post processing ($sign = 1$) or the backward post processing ($sign = -1$). The coefficients used in this routine are determined by the routine `abxyz` which doesn't involve any active variables, therefore we don't need to compute its discrete adjoint.

The original `FFT_postproc` routine is described by algorithm 43.

Algorithm 43 `FFT_postproc`(in,out,$sign$)

Initialization: call `abxyz`(ax,bx,ay,by,az,bz)
  for i=2,n **do**
    $tmp1 = real(in(i))$
    $tmp2 = aimag(in(i))$
    $tmp3 = real(in(n - i + 2))$
    $tmp4 = aimag(in(n - i + 2))$
    $xx1 = tmp1.bx/2$; $xx2 = tmp1.ax/2$
    $xx3 = tmp2.bx/2$; $xx4 = tmp2.ax/2$
    $xx5 = tmp3.bx/2$; $xx6 = tmp3.ax/2$
    $xx7 = tmp4.bx/2$; $xx8 = tmp4.ax/2$
    if ($sign$) **then**
      $out(i) = \text{cmplx}(xx1 + xx4 + xx5 - xx8, -xx2 + xx3 + xx6 + xx7)$
    else
      $out(i) = \text{cmplx}(xx1 - xx4 + xx6 + xx7, -(-xx2 - xx3 + xx5 - xx8))$
    end if
  end for

The discrete adjoint corresponding to the `FFT_postproc` routine is obtained straightforwardly by TAPENADE and is described by algorithm 44.

Algorithm 44 FFT_postproc_b(in,$\overline{in}$,out,$\overline{out}$,$sign$)

Initialization: call abxyz(ax,bx,ay,by,az,bz)
  for i=2,n **do**
    $tmp5b = \text{real}(\overline{out})$
    $tmp6b = \text{aimag}(\overline{out})$
    if $(sign)$ **then**
      $xx1b = tmp5b$; $xx4b = tmp5b$; $xx5b = tmp5b$; $xx8b = -tmp5b$
      $xx2b = -tmp6b$; $xx3b = tmp6b$; $xx6b = tmp6b$; $xx7b = tmp6b$
    else
      $xx1b = tmp5b$; $xx4b = tmp5b$; $xx5b = tmp5b$; $xx8b = -tmp5b$
      $xx2b = -tmp6b$; $xx3b = tmp6b$; $xx6b = tmp6b$; $xx7b = tmp6b$
    end if
    $tmp4b = bx.xx7b/2 + ax.xx8b/2$
    $tmp3b = bx.xx5b/2 + ax.xx6b/2$
    $tmp2b = bx.xx3b/2 + ax.xx4b/2$
    $tmp1b = bx.xx1b/2 + ax.xx2b/2$
    $\overline{in}(nx - i + 2) = \overline{in} + \text{cmplx}(tmp3b, tmp4b)$
    $\overline{in}(i) = \overline{in} + \text{cmplx}(tmp1b, tmp2b)$
  end for

Annexe G

PIV system

In the double frame configuration, we can change each system's acquisition time which corresponds to the period between two snapshots which are used to determine the velocity of each particules. Then we set a delay between two acquisition in order to perform another acquisition with system 2.In figure G.1, the acquisition of the system 1 and system 2 is $500\mu s$ and $400\mu s$, respectively. We set a cooldown of $2000\mu s$ between each acqusition. The period between the begin of the system 1 and system 2 velocity acquisition is $1000\mu s$. Thus, in this configuration we can set the observation time step $\Delta t_{obs} = 1000\mu s$.

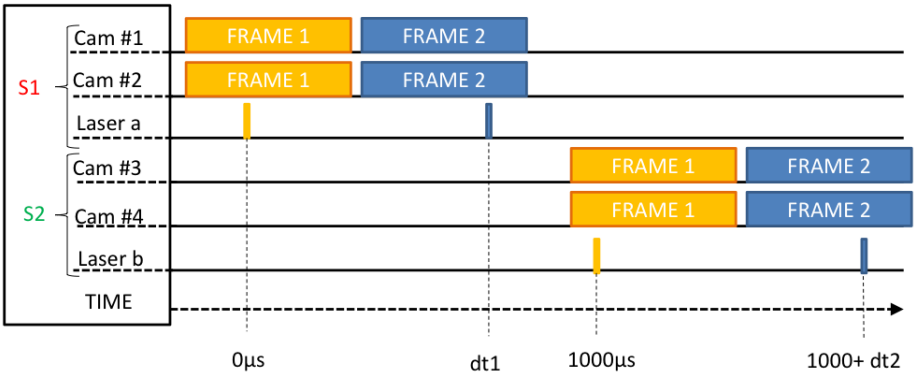

FIGURE G.1 – Double frame configuration : the acquisition of the system 1 and system 2 is dt1 and dt2 respectively.

In the single frame configuration, each system's acquisition time is fixed, we perform a shit between each acquisition to avoid an overlap between each system's acquisition. In figure G.2, the acquisition of both is $2000\mu s$. We set system 2's acquisition $1000\mu s$ after system 1's acquisition. Thus, in this configuration we can set the observation time step $\Delta t_{obs} = 1000\mu s$.

The stereo PIV systems were calibrated beforehand to ensure that the common area of both observed planes are equivalent. The correct spatio-temporal positioning of the common area was verified by comparing measurements of the velocity profile in the common area for each PIV system as illustrated in figure G.3. The comparison was performed by A. Guibert.

As the wind tunnel was initially designed for a mixing layer configuration, we

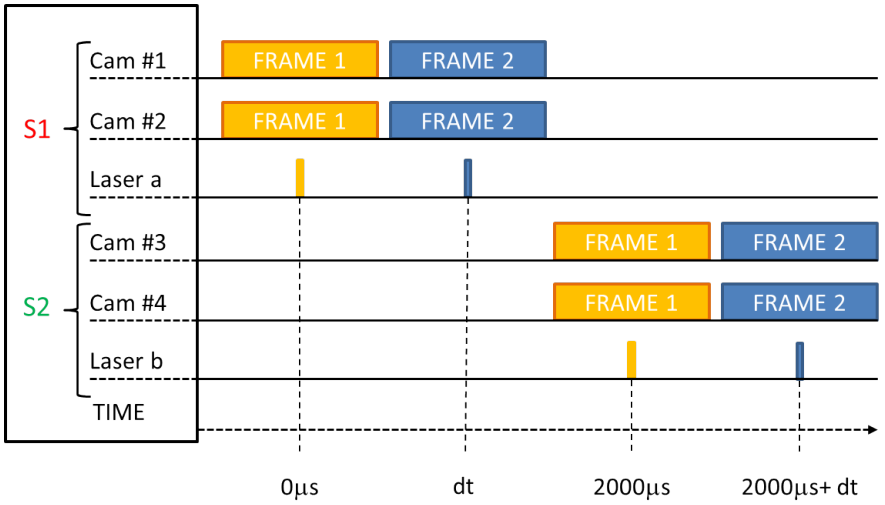

FIGURE G.2 – Single frame configuration

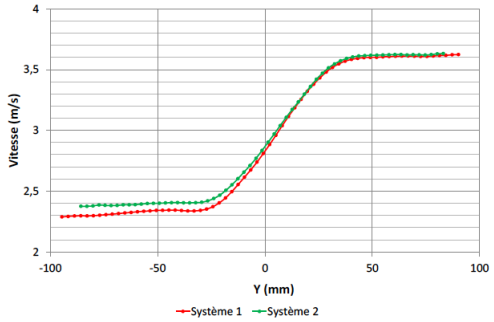

FIGURE G.3 – Comparison of the velocity profile u at the inlet

added a cushion at the separation plaque to reduce its effect as illustrated in figure **??**. After installing the cushion a the entry of the test section, we controlled the initial flow in contact with the cylinder. We input two air streams of $0.5m/s$ at the same temperature and measured the velocity profile in the x-direction and the turbulence rate at 830 mm from the separation plaque. As shown in figure G.4, the measured velocity is more or less/acceptably constant along the height, which shows the cushion reduced well the effects of the separation plaque.

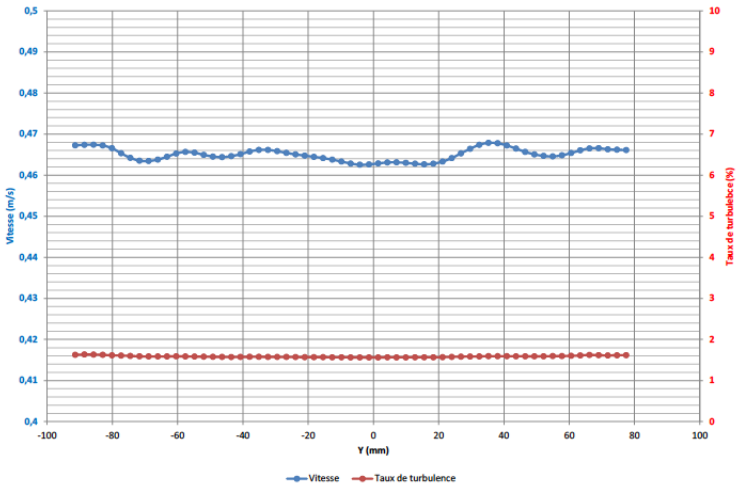

FIGURE G.4 – Dual plane position in the cylinder wake configuration at Reynolds 300

Bibliographie

R J Adrian. Particle-imaging techniques for experimental fluid mechanics. *Annual review of fluid mechanics*, 23(1) :261–304, 1991.

R J Adrian. Limiting resolution of particle image velocimetry for turbulent flow. *Advances in Turbulence Research*, 1, 1995.

R J Adrian. Twenty years of particle image velocimetry. *Experiments in Fluids*, 39 : 159–169, 2005.

M P Arroyo and C A Greated. Stereoscopic particle image velocimetry. *Measurement Science and Technology*, 2(12) :1181, 1991.

S Avril, M Bonnet, A S Bretelle, M Grediac, F Hild, P Ienny, F Latourte, D Lemosse, S Pagano, E Pagnacco, and F Pierron. Further observations on the mean velocity distribution in fully developed pipe flow. *Exp. Mech.*, 48 :381–402, 2008.

P Bergthorsson and B R Döös. Numerical weather map analysis. *Tellus*, 7(3) : 329–340, 1955.

M Bocquet and P Sakov. An iterative ensemble kalman smoother. *Quarterly Journal of the Royal Meteorological Society*, 140(682) :1521–1535, 2014.

C Bogey, C Bailly, and D Juvé. Numerical simulation of sound generated by vortex pairing in a mixing layer. *AIAA journal*, 38(12) :2210–2218, 2000.

J F Brady and A Acrivos. Steady flow in a channel or tube with an accelerating surface velocity. an exact solution to the navier– stokes equations with reverse flow. *Journal of Fluid Mechanics*, 112 :127–150, 1981.

C Brücker. 3d scanning piv applied to an air flow in a motored engine using digital high-speed video. *Measurement Science and Technology*, 8(12) :1480, 1997.

M Buehner. Ensemble-derived stationary and flow-dependent background-error covariances : Evaluation in a quasi-operational nwp setting. *Quarterly Journal of the Royal Meteorological Society*, 131(607) :1013–1043, 2005.

M Buehner, P L Houtekamer, C Charette, H L Mitchell, and B He. Intercomparison of variational data assimilation and the ensemble kalman filter for global deterministic nwp. part i : Description and single-observation experiments. *Monthly Weather Review*, 138(5) :1550–1566, 2010a.

M Buehner, P L Houtekamer, C Charette, H L Mitchell, and B He. Intercomparison of variational data assimilation and the ensemble kalman filter for global deterministic nwp. part ii : One-month experiments with real observations. *Monthly Weather Review*, 138(5) :1567–1586, 2010b.

T Bui-Thanh, M Damodaran, and K E Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal*, 42(8) : 1505–1516, 2004.

G Burgers, P van Leeuwen, and G Evensen. Analysis scheme in the ensemble kalman filter. *Monthly Weaver Review*, 126 :1719–1724, 1998.

C Canuto, M Y Hussaini, A Quarteroni, and T A Zang. Spectral methods in fluid dynamics. Technical report, Springer, 1988.

V S S Chan, W D Koek, D H Barnhart, N Bhattacharya, J J M Braat, and J Westerweel. Application of holography to fluid flow measurements using bacteriorhodopsin (br). *Measurement Science and Technology*, 15(4) :647, 2004.

L Chen, SW Coleman, JC Vassilicos, and Zhiwei Hu. Acceleration in turbulent channel flow. *Journal of Turbulence*, (11) :N41, 2010.

B Cheng. A duality between forward and adjoint mpi communication routines. *Computational Methods in Science and Technology. Polish Academy of Sciences*, pages 23–24, 2006.

A M Clayton, A C Lorenc, and D M Barker. Operational implementation of a hybrid ensemble/4d-var global data assimilation system at the met office. *Quarterly Journal of the Royal Meteorological Society*, 139(675) :1445–1461, 2013.

C H Colburn, J B Cessna, and T R Bewley. State estimation in wall-bounded flow systems. part 3. the ensemble kalman filter. *Journal of Fluid Mechanics*, 682 : 289–303, 2011.

B Combes, A Guibert, E Memin, and D Heitz. Free-surface flows from kinect : Feasibility and limits. In *FVR2011*, pages 4–p, 2011.

B Combés, D Heitz, A Guibert, and E Mémin. A particle filter to reconstruct a free-surface flow from a depth camera. *Fluid Dynamics Research*, 47(5) :51404–51428, 2015.

P Courtier and O Talagrand. Variational assimilation of meteorological observations with the adjoint vorticity equation. ii : Numerical results. *Quarterly Journal of the Royal Meteorological Society*, 113(478), 1987.

P Courtier, J N Thépaut, and A Hollingsworth. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(120) :1367–1388, July 1994.

L J Crane. Flow past a stretching plate. *Zeitschrift fuer Angewandte Mathematik und Physik*, 21 :645–647, 1970.

J P Cressman. An operational objective analysis system. *Monthly Weather Review*, 87(10) :367–374, 1959.

A Cuzol and E Mémin. A stochastic filtering technique for fluid flows velocity fields tracking. *IEEE Trans. on Pattern Anaysis and Machine Intelligence*, 31(3) :329–349, 2009.

A Cuzol, P Hellier, and E M´ emin. A low dimensional fluid motion estimator. *Int. J. Computer Vision*, 75(3) :329–349, 2007.

J D'Adamo, N Papadakis, E Mémin, and G Artana. Variational assimilation of pod low-order dynamical systems. *J. Turb.*, 8(9) :1–22, 2007.

M F De Pando, D Sipp, and P J Schmid. Efficient evaluation of the direct and adjoint linearized dynamics from compressible flow solvers. *Journal of Computational Physics*, 231(23) :7739–7755, 2012.

G Desroziers, J-T Camino, and L Berre. 4denvar : link with 4d state formulation of variational assimilation and different possible implementations. *Quarterly Journal of the Royal Meteorological Society*, 140(684) :2097–2110, 2014.

S Dong, G E Karniadakis, A Ekmekci, and D Rockwell. A combined direct numerical simulation-particle image velocimetry study of the turbulent near wake. *Journal of Fluid Mechanics*, 569 :185–207, 2006.

N Dovetta, D Foures, P Schmid, B McKeon, and D Sipp. Assimilation de données pour la reconstruction d'écoulements et l'identification de turbulence. *21ème Congrès Français de Mécanique, 26 au 30 août 2013, Bordeaux, France (FR)*, 2014.

P Druault, S Lardeau, J-P Bonnet, F Coiffet, J Delville, E Lamballais, J-F Largeau, and L Perret. Generation of three-dimensional turbulent inlet conditions for large-eddy simulation. *AIAA journal*, 42(3) :447–456, 2004.

G E Elsinga, F Scarano, B Wieneke, and B W van Oudheusden. Tomographic particle image velocimetry. *Experiments in Fluids*, 41(6) :933–947, 2006.

G Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. 1994.

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

R Everson and L Sirovich. Karhunen–loeve procedure for gappy data. *JOSA A*, 12 (8) :1657–1664, 1995.

C Faure and P Dutto. Extension of odyssée to the mpi library-reverse mode. 1999.

C Faure and Y Papegay. Odyssée user's guide version 1.7. 1998.

W E Ferguson Jr. A simple derivation of glassman general-n fast fourier transform. comput. and math. with appls., 8 (6) : 4018211 ; 411, 1982. also. *Report AD-A083811, NTIS*, 1979.

B Ferron and L Hascoët. Capacités actuelles de la différentiation automatique : l'adjoint d'opa par tapenade. In *Colloque National sur l'Assimilation de Données, Toulouse, France*, 2006.

E J Fertig, J Harlim, and B R Hunt. A comparative study of 4d-var and a 4d ensemble kalman filter : Perfect model simulations with lorenz-96. *Tellus A*, 59 (1) :96–100, 2007.

D P G Foures, N Dovetta, D Sipp, and P J Schmid. A data-assimilation method for reynolds-averaged navier–stokes-driven mean flow reconstruction. *Journal of Fluid Mechanics*, 759 :404–431, 2014.

N Fujisawa, S Tanahashi, and K Srinivas. Evaluation of pressure field and fluid forces on a circular cylinder with and without rotational oscillation using velocity data from piv measurement. *Measurement Science and Technology*, 16(4), 2005.

L Gandin. Objective analysis of meteorological fields : Gidrometeorologicheskoe izdatel'stvo (gimiz), leningrad. Translated by Israel Program for Scientific Translations, Jerusalem., 1965.

G Gaspari and S E Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554) : 723–757, 1999.

M Germano, U Piomelli, P Moin, and W H Cabot. A dynamic subgridâĂŘscale eddy viscosity model. *Physics of Fluids A : Fluid Dynamics (1989-1993)*, 3(7), 1991.

P Glaister. Approximate riemann solutions of the shallow water equations. *Journal of Hydraulic Research*, 26(3) :293–306, 1988.

S K Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3) :271–306, 1959.

D O Gottlieb and S A Orszag. *Numerical Analysis of Spectral Methods : Theory and Applications*. NSF-CBMS Monograph, 1977.

R Griesse and A Walther. Evaluating gradients in optimal control : continuous adjoints versus automatic differentiation. *Journal of optimization theory and applications*, 122(1) :63–86, 2004.

A Gronskis, D Heitz, and E Mémin. Inflow and initial conditions for direct numerical simulation based on adjoint data assimilation. *Journal of Computational Physics*, 242 :480–497, 2013.

S A Haben, A S Lawless, and N K Nichols. Conditioning and preconditioning of the variational data assimilation problem. *Computers & Fluids*, 46(1) :252–256, 2011.

J Habert, S Ricci, A Piacentini, G Jonville, E Le Pape, O Thual, N Goutal, F Zaoui, and R Ata. Estimation of lateral inflows using data assimilation in the context of real-time flood forecasting for the marne catchment in france. In *Advances in Hydroinformatics*, pages 93–105. Springer, 2014.

W W Hager. Runge-kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2) :247–282, 2000.

T M Hamill and C Snyder. A hybrid ensemble kalman filter-3d variational analysis scheme. *Monthly Weather Review*, 128(8) :2905–2919, 2000.

A Harten, P D Lax, and B V Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1) :35–61, 1983.

L Hascoët. Tapenade : a tool for automatic differentiation of programs. In *Proceedings of 4^{th} European Congresson Computational Methods, ECCOMAS'2004, Jyvaskyla, Finland*, 2004.

L Hascoët. Automatic differentiation by program transformation. *Ecole d'été CEA*, 2006.

L Hascoët. Adjoints by Automatic Differentiation. In E. Blayo, M. Bocquet, and E. Cosme, editors, *Advanced data assimilation for geosciences*. Oxford University Press, 2012. to appear.

L Hascoët and V Pascual. *Tapenade 2.1 user's guide*, 2004.

L Hascoët and V Pascual. The Tapenade Automatic Differentiation tool : Principles, Model, and Specification. *ACM Transactions On Mathematical Software*, 39(3), 2013. URL http://dx.doi.org/10.1145/2450153.2450158.

D Heitz, E Mémin, and C Schnörr. Variational fluid flow measurements from image sequences : synopsis and perspectives. *Exp. Fluids*, 48(3) :369–393, 2010.

K D Hinsch. Holographic particle image velocimetry. *Measurement Science and Technology*, 13(7) :R61, 2002.

R A Humble, G E Elsinga, F Scarano, and B W van Oudheusden. Three-dimensional instationary structure of a shock wave turbulent boundary layer interaction. *J Fluid Mech*, 44 :33–62, 2009.

B R Hunt, E Kalnay, E J Kostelich, E Ott, D J Patil, T Sauer, I Szunyogh, J A Yorke, and A V Zimin. Four-dimensional ensemble kalman filtering. *Tellus A*, 56 (4) :273–277, 2004.

B R Hunt, E J Kostelich, and I Szunyogh. Efficient data assimilation for spatiotemporal chaos : A local ensemble transform kalman filter. *Physica D : Nonlinear Phenomena*, 230(1) :112–126, 2007.

A Jameson. Optimum aerodynamic design using cfd and control theory. *AIAA paper*, 1729 :124–131, 1995.

C J Kähler and J Kompenhans. Fundamentals of multiple plane stereo particle image velocimetry. *Experiments in Fluids*, 29(1) :S070–S077, 2000.

R E Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1) :35–45, 1960.

H Kato and S Obayashi. Approach for uncertainty of turbulence modeling based on data assimilation technique. *Computers & Fluids*, 85 :2–7, 2013.

C A Kennedy, M H Carpenter, and R M Lewis. Low-storage, explicit runge–kutta schemes for the compressible navier–stokes equations. *Applied numerical mathematics*, 35(3) :177–219, 2000.

M Kuhn, K Ehrenfried, J Bosback, and C Wagner. Large-scale tomographic particle image velocimetry using helium-filled soap bubbles. *Experiments in Fluids*, 50 : 929–948, 2011.

S Laizet and E Lamballais. High-order compact schemes for incompressible flows : A simple and efficient method with quasi-spectral accuracy. *Journal of Computational Physics*, 228(16) :5989–6015, 2009.

S Laizet, E Lamballais, and J C Vassilicos. A numerical strategy to combine high-order schemes, complex geometry and parallel computing for high resolution dns of fractal generated turbulence. *Computers and Fluids*, 3(39) :471–484, 2010.

F Le Dimet and O Talagrand. Variational algorithms for analysis and assimilation of meteorological observations : theoretical aspects. *Tellus. Series A, Dynamic meteorology and oceanography*, 38(2), 1986.

S K Lele. Compact finite difference schemes with spectral-like resolution. *Journal of computational physics*, 103 :16–42, 1992.

M Lemke and J Sesterhenne. Adjoint based pressure determination from piv-data - validation with synthetic piv measurements. *10th International Symposium On Particle Image Velocimetry - PIV13*, 2013.

M Lesieur. *Turbulence in fluids : stochastic and numerical modelling (Vol. 1)*. Springer Science & Business Media, 2012.

N Li and S Laizet. 2decompfft a highly scalable 2d decomposition library and fft interface. *In Cray User Group 2010 conference*, pages 1–13, 2010.

J Lions. *Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles*. Dunod, 1968.

J Lions. *Optimal control of systems governed by PDEs*. Springer-Verlag, New York, 1971.

C Liu, Q Xiao, and B Wang. An ensemble-based four-dimensional variational data assimilation scheme. part i : Technical formulation and preliminary test. *Monthly Weather Review*, 136(9) :3363–3373, 2008.

C Liu, Q Xiao, and B Wang. An ensemble-based four-dimensional variational data assimilation scheme. part ii : Observing system simulation experiments with advanced research wrf (arw). *Monthly Weather Review*, 137(5) :1687–1704, 2009.

A C Lorenc. The potential of the ensemble kalman fi filter for nwpâĂŤa comparison with 4d-var. *QJR Meteorol. Soc*, 129 :3183–3203, 2003.

J Lundvall, V Kozlov, and P Weinerfelt. Iterative methods for data assimilation for burger's equation. *J. Inverse Ill-Posed Probl.*, 14(5) :505–535, 2006.

B J McKeon, J D Li, W Jiang, J F Morrison, and A J Smits. Further observations on the mean velocity distribution in fully developed pipe flow. *Journal of Fluid Mechanics*, 501 :135–147, 2004.

C D Meinhart, A K Prasad, and R J Adrian. A parallel digital processor system for particle image velocimetry. *Measurement Science and Technology*, 4(5) :619, 1993.

E Mémin. Fluid flow dynamics under location uncertainty. *Geophysical & Astrophysical Fluid Dynamics*, 2014.

P Mercier and M Deville. A multidimensional compact higher-order scheme for 3-d poisson's equation. *Journal of Computational Physics*, 39(2) :443–455, 1981.

V Mons, J-C Chassaing, T Gomez, and P Sagaut. Is isotropic turbulence decay governed by asymptotic behavior of large scales ? an eddy-damped quasi-normal markovian-based data assimilation study. *Physics of Fluids (1994-present)*, 26 (11) :115105, 2014.

S Nadarajah and A Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA paper*, 667, 2000.

S Nagarajan, S K Lele, and J H Ferziger. A robust high-order compact method for large eddy simulation. *Journal of Computational Physics*, 191(2) :392–419, 2003.

J Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151) :773–782, July 1980.

N Papadakis and E Mémin. Variational assimilation of fluid motion from image sequences. *SIAM J. Imaging Sci.*, 1(4) :343–363, 2009.

P Parnaudeau, E Lamballais, D Heitz, and J H Silvestrini. Combination of the immersed boundary method with compact schemes for dns of flows in complex geometry. In *Direct and Large-Eddy Simulation V*, pages 581–590. Springer, 2004.

P Parnaudeau, J Carlier, D Heitz, and E Lamballais. Experimental and numerical studies of the flow over a circular cylinder at reynolds number 3900. *Physics of Fluids (1994-present)*, 20(8), 2008.

A T Patera. A spectral element method for fluid dynamics : Laminar flow in a channel expansion. *Journal of computational physics*, 54 :468–488, 1984.

A Ponçot, J-P Argaud, B Bouriquet, P Erhard, S Gratton, and O Thual. Variational assimilation for xenon dynamical forecasts in neutronic using advanced background error covariance matrix modelling. *Annals of Nuclear Energy*, 60 : 39–50, 2013.

A Priestley. *Roe Type Schemes for the 2-D Shallow Water Equations*. University of Reading. Department of Mathematics, 1987.

M Raffel, C E Willert, J Kompenhans, et al. *Particle image velocimetry : a practical guide*. Springer, 2013.

P L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2) :357–372, 1981.

P Sagaut. *Large eddy simulation for incompressible flows : an introduction*. Springer Science and Business Media, 2006.

P Sakov, G Evensen, and L Bertino. Asynchronous data assimilation with the enkf. *Tellus A*, 62(1) :24–29, 2010.

P Sakov, D S Oliver, and L Bertino. An iterative enkf for strongly nonlinear systems. *Monthly Weather Review*, 140(6) :1988–2004, 2012.

A Sandu. On the properties of runge-kutta discrete adjoints. *In Computational Science–ICCS 2006. Springer Berlin Heidelberg, 2006*, pages 550–557, 2006.

Y Sasaki. An objective analysis based on the variational method. *Meteor. Soc. Japan*, 36(3), 1958.

F Scarano, S Ghaemi, G C A Caridi, J Bosbach, U Dierksheide, and A Sciacchitano. On the use of helium-filled soap bubbles for large-scale tomographic piv in wind tunnel experiments. *Experiments in Fluids*, pages 56–42, 2015.

M Schanen, U Naumann, L Hascoët, and J Utke. Interpretative adjoints for numerical simulation codes using mpi. *Procedia Computer Science*, 1(1) :1825–1833, 2010.

D Schanz, A Schröder, S Gesemann, D Michaelis, and B Wieneke. 'shake the box' : A highly efficient and accurate tomographic particle tracking velocimetry (tomo-ptv) method using prediction of particle positions. In *PIV13 ; 10th International Symposium on Particle Image Velocimetry, Delft, The Netherlands, July 1-3, 2013*. Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, and Faculty of Aerospace Engineering, 2013.

A Schröder, R Geisler, G E Elsinga, F Scarano, and U Dierksheide. Investigation of a turbulent spot and a tripped turbulent boundary layer flow using time-resolved tomographic piv. *Exp. Fluids*, 44 :305–316, 2008.

L Sirovich. Turbulence and the dynamics of coherent structures. part i : Coherent structures. *Quarterly of applied mathematics*, 45(3) :561–571, 1987.

J Smagorinsky. General circulation experiments with the primitive equations : I. the basic experiment*. *Monthly weather review*, 91(3) :99–164, 1963.

A Solonen, H Haario, J Hakkarainen, H Auvinen, I Amour, and T Kauranne. Variational ensemble kalman filtering using limited memory bfgs. *Electronic Transactions on Numerical Analysis*, 39 :271–285, 2012.

P R Spalart. Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow*, 21(3) :252–263, 2000.

P K Stansby. The effects of end plates on the base pressure coefficient of a circular cylinder. *Aeronautical Journal*, 78 :36, 1974.

T Suzuki. Pod-based reduced-order hybrid simulation using the data-driven transfer function with time-resolved ptv feedback. *Experiments in Fluids*, 55(8) :1–17, 2014.

T Suzuki, H Ji, and F Yamamoto. Unsteady ptv velocity field past an airfoil solved with dns : Part 1. algorithm of hybrid simulation and hybrid velocity field at re $\approx$ 103. *Experiments in fluids*, 47(6) :957–976, 2009a.

T Suzuki, A Sanse, T Mizushima, and F Yamamoto. Unsteady ptv velocity field past an airfoil solved with dns : Part 2. validation and application at reynolds numbers up to re $\leq$ 104. *Experiments in fluids*, 47(6) :977–994, 2009b.

B A Szabo and I Babu**v**ska. *Finite element analysis*. John Wiley & Sons, 1991.

M-H Tber, L Hascoët, A Vidard, and B. Dauvergne. Building the tangent and adjoint codes of the ocean general circulation model OPA with the automatic differentiation tool tapenade. Research Report 6372, INRIA, 2007. URL https://hal.inria.fr/inria-00192415.

J Utke, L Hascoët, P Heimbach, C Hill, P Hovland, and U Naumann. Toward adjoinable mpi. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

J Utke, P Hovland, L Hascoët, V Pascual, P Heimbach, C Hill, U Naumann, and M Schanen. Gradient of mpi-parallel codes. 2012.

B W Van Oudheusden. Principles and application of velocimetry-based planar pressure imaging in compressible flows with shocks. *Experiments in fluids*, 45(4) : 657–674, 2005.

H K Versteeg and W Malalasekera. *An introduction to computational fluid dynamics : the finite volume method*. Pearson Education, 2007.

M Virant and T Dracos. 3d ptv and its application on lagrangian motion. *Measurement science and technology*, 8(12) :1539, 1997.

C B Vreugdenhil. *Numerical methods for shallow-water flow*. Water Science and Technology Library, 1994.

C Y Wang. Exact solutions of the steady-state navier–stokes equations. *Annual Review of Fluid Mechanics*, 23(159-177), 1992.

Z J Wang. Spectral (finite) volume method for conservation lazs on unstructured grids. basic formulation. *Journal of computational physics*, 54 :468–488, 2002.

Z J Wang and Y Liu. Extension of the spectral volume method to high-order boundary representation. *Journal of computational physics*, 211(1) :154–178, 2006.

M Wei, G Jacobs, C Rowley, N C N Barron, P Hogan, P Spence, O M Smedstad, P Martin, P Muscarella, and E Coelho. The impact of initial spread calibration on the relo ensemble and its application to lagrangian dynamics. *Nonlinear Processes in Geophysics*, 20(5) :621–641, 2013.

D C Wilcox. *Turbulence modeling for CFD*. La Canada, CA : DCW industries, 1998.

R B Wilhelmson and J H Ericksen. Direct solutions for poisson's equation in three dimensions. *Journal of Computational Physics*, 25(4) :319–331, 1977.

J H Williamson. Low-storage runge-kutta schemes. *Journal of Computational Physics*, 35(1) :48–56, 1980.

L Wu, V Mallet, M Bocquet, and B Sportisse. A comparison study of data assimilation algorithms for ozone forecasts. *Journal of Geophysical Research*, 113(D20), 2008.

Y Yang. *Study of variational ensemble methods for image assimilation*. PhD thesis, Université Rennes 1, 2014.

Y Yang, C Robinson, D Heitz, and E Mémin. Enhanced ensemble-based 4dvar scheme for data assimilation. *Computers and Fluids*, 115 :201–210, 2015.

M M Zdravkovich. *Flow around Circular Cylinders : Volume 2 : Applications (Vol. 2)*. Oxford university press, 2003.

M Zupanski. Maximum likelihood ensemble filter : Theoretical aspects. *Monthly Weather Review*, 133(6) :1710–1726, 2005.

Abstract : In the one hand, flow dynamics are usually described by the Navier-Stokes equations and the literature provides a wide range of techniques to solve such equations. On the other hand, we can nowadays measure different characteristics of a flow (velocity, pressure, temperature etc...) with non-intrusive Particle Image Velocimetry techniques. Within this thesis, we take interest in the data assimilation techniques, that combine a dynamics model with measurements to determine a better approximation of the system. This thesis focus on the classic variational assimilation technique (4DVar) which ensures a high accuracy of the solution by construction.

We carry out a first application of the 4DVar technique to reconstruct the characteristics (height and velocity field) of a uni directional wave at its free surface. The fluid evolution is simulated by the shallow water equations and solved numerically. We use a simple experimental setup envolving a depth sensor (Kinect sensor) to extract the free surface height. We compared the results of the 4DVar reconstruction with different versions of the hybrid data assimilation technique 4DEnVar.

Finally, we apply the 4DVar technique to reconstruct the downstream of a three dimensional cylinder wake at Reynolds 300. The turbulent flow is simulated by the high-performance multi-threading DNS code Incompact3d. This dynamics model is first combined with synthetic three dimensional observations, then with real orthogonal-plane stereo PIV observations to reconstruct the full three dimensional flow.
Key words : variational assimilation, fluid dynamics, particle image velocimetry

Résumé : D'une part, les équations de Navier-Stokes permettent de décrire les écoulements fluides, la littérature est riche de méthodes numériques permettant la résolution de celle-ci. D'autre part, nous sommes capables de mesurer de manière non-intrusive différentes caractéristique d'un écoulement (champ de vitesse et pression, etc.). Dans le cadre de cette thèse, nous nous intéressons aux techniques d'assimilation de données qui combinent les modèles numériques avec les observations afin de déterminer une meilleure approximation du système. Cette thèse s'articule autour de l'assimilation de donnée variationelle (4DVar) qui est plus précise par construction.

Nous avons mené une première application sur la reconstruction de la hauteur et vitesse de la surface libre d'un fluide contenu dans un récipient rectangulaire à fond plat. L'écoulement est modélisé par les équations de shallow water et résolues numériquement. Les observations de l'évolution de la hauteur de la surface libre ont été prélevées par un capteur de profondeur (Kinect). Nous avons comparé les résultats de la reconstruction par ls 4DVar avec plusieurs version de la méthode d'assimilation hybride 4DEnVar.

Enfin, nous avons appliqué la technique 4DVar à la reconstruction volumique de l'aval d'un sillage de cylindre à Reynolds 300. L'écoulement turbulent a été simulé par un code DNS parallèle Incompact3D. La reconstruction a été éffectué en combinant tout d'abord des observations synthètiques en trois dimension, puis en combinant des observations de plans orthogonales en stéréo PIV.
Mots clefs : assimilation variationelle de données, dynamique des fluides, particle image velocimetry