Variational assimilation of Geophysical images leveraging deep learning tools

Arthur Filoche, Dominique Béréziat, Julien Brajard, Anastase Alexandre Charantonis

▶ **To cite this version:**

Arthur Filoche, Dominique Béréziat, Julien Brajard, Anastase Alexandre Charantonis. Variational assimilation of Geophysical images leveraging deep learning tools. ORASIS 2021, Centre National de la Recherche Scientifique [CNRS], Sep 2021, Saint Ferréol, France. hal-03339674

HAL Id: hal-03339674
https://hal.science/hal-03339674v1

Submitted on 9 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Variational assimilation of geophysical images leveraging deep learning tools

A. Filoche[1] D. Béréziat[1]
A. Charantonis[2] J. Brajard[3]

[1] Sorbonne Université, CNRS, LIP6, 75005 Paris, France
[2] Laboratoire d'Océanographie et du Climat (LOCEAN), 75005 Paris, France
[3] Nansen Environmental and Remote Sensing Center (NERSC), 5009 Bergen, Norway

4 Place Jussieu, 75005 Paris, France
arthur.filoche@lip6.fr

Résumé

De nombreuses applications en géoscience nécessitent d'estimer l'état d'un système physique. L'assimilation de données fournit un cadre rigoureux pour le faire lorsque des connaissances de la dynamique qui régit le système et des observations de ce système sont disponibles. Quand ce type de problème inverse est résolu sous une forme variationnelle, le processus d'optimisation implique la méthode de l'état adjoint pour un calcul efficace du gradient. D'un point de vue informatique, cette méthode est équivalente à l'algorithme de rétropropagation et est mise en œuvre en utilisant la différenciation automatique. Le récent développement des outils d'apprentissage profond permet une mise en œuvre flexible de ces méthodes et ouvre la porte à une modélisation hybride axée sur les données et la connaissance. Ceci est illustré sur deux problèmes d'assimilation d'images géophysiques.

Abstract

Many applications in Earth Sciences require the estimation of a physical system state. Data Assimilation provides a strong framework to do so when knowledge about a governing dynamics and observations of such system are available. If this kind of inverse problem is solved in a variational form, the optimization process involves the adjoint state method for efficient gradient computing. From a computational perspective, this method is equivalent to the backpropagation algorithm and is implemented using automatic differentiation. The ongoing development of deep learning tools allows flexible implementation of such methods and opens the door to hybrid data-knowledge driven modeling. This is illustrated on two geophysical image assimilation problems.

Keywords

automatic differentiation, variational data assimilation, deep learning, geoscience informatics

1 Introduction

Geo-scientific observations are imperfect by nature either they are incomplete, noisy or indirect. Estimating the state of physical systems using such data has led to a variety of inverse problems. When involving knowledge about the system evolution in the form of a dynamical model, these problems are solved with data assimilation [1], a set of techniques producing state-of-the-art results in various numerical weather forecasting application. When it comes to operational forecasting, meteorological centers tend to use ensemble methods, such as the ensemble Kalman filter [2], and variational methods such as 4D-Var [3].

Regarding variational methods, the optimal combination of model and observations is done through the minimization of a cost function where the control parameters are roughly state variables to be estimated. As the state is usually high-dimensional and the dynamics integration computationally expensive, approximating the gradient of the cost function using finite-differences is not viable. Hopefully, we can rely on the adjoint state method [3] to obtain explicitly this gradient in only one forward of the dynamics and one backward of the adjoint dynamics. Computationally, this implies having at disposal a discrete adjoint of the numerical model which is non-trivial but can be done with software based on automatic differentiation like Tapenade [4].

Parallels between data assimilation and deep neural networks architectures have already been drawn [5] and we like to emphasis that adjoint backpropagation through time steps or through hidden layers behave in a similar manner. Even though the analogy is debatable, the possibility to solve various kinds of variational problems using flexible tools from the deep learning community, also based on automatic differentiation, is real. When designing a neural network architecture, only choices regarding the forward model are made. During the optimization, the adjoint network derived from the computational graph is

used to calculate gradients. This part is fully managed by the software. We propose to use the same paradigm to solve variational data assimilation problems, thus avoiding explicit adjoint modeling that is also known to be an operational issue [6].

In this work, we assimilate two set of images, one from rain radar, and one giving sea surface temperatures, using an advection model and the 4D-Var algorithm coded using Autograd [7] from Pytorch. The code is available on Github[1]. We compare the results with a reliable code developed in [8, 9]. We observed that while this flexibility comes at a computational cost, results are relevant and open up perspectives on hybrid modeling connecting traditional data assimilation methods and machine learning ones.

2 Data Assimilation Framework

2.1 Mathematical setting

A state vector $\mathbf{X}$ evolves over a discrete time $t \in [0:T]$ according to a partially-known dynamics $\mathbb{M}$, see Eq. (1). Partial and noisy observations $\mathbf{Y}$ are available through a linear observation operator $\mathbb{H}$, Eq. (2). A background $\mathbf{X}_b$ gives prior information about the initial state, Eq. (3).

$$\text{Evolution:} \quad \mathbf{X}_{t+1} = \mathbb{M}_t(\mathbf{X}_t) + \varepsilon_{m_t} \tag{1}$$

$$\text{Observation:} \quad \mathbf{Y}_t = \mathbb{H}_t \mathbf{X}_t + \varepsilon_{R_t} \tag{2}$$

$$\text{Background:} \quad \mathbf{X}_0 = \mathbf{X}_b + \varepsilon_b \tag{3}$$

Uncertainties about dynamics, observations and background are modeled by additive noises denoted ε_b, ε_R and ε_m, respectively. These noises are quantified by their assumed known covariance matrices $\mathbf{B}$, $\mathbf{R}$ and $\mathbf{Q}$, respectively. For any given matrix $\mathbf{A}$, we note $\|x-y\|_A^2 = \langle (x-y) | \mathbf{A}^{-1}(x-y) \rangle$ the associated Mahalanobis distance.

This framework enables the use of irregularly-sampled in time observations. As models $\mathbb{M}_t$ often stand for numerical schemes approximating the integration of a PDE-system, they may only differ by their total time steps. For the sake of simplicity, we denote multiple integration between two times $\mathbb{M}_{t_1 \to t_2}$. Obviously, this operator depends of dynamical models $\mathbb{M}_t$ between those two times but it is important to note that they also depends on ε_{m_t} which play a role in the evolution (see Eq.1).

Ultimately, Data assimilation aims at optimally combine these various sources of information in order to produce an estimate of the system state $\mathbf{X}$.

2.2 4D-Var

Cost function. In a variational formalism this estimation is done over a temporal window $[0:T]$ via the minimization of a cost function denoted J. Different costs composing J are associated to different error sources and weighted accordingly with error covariance matrices.

$$J(\varepsilon_b, \varepsilon_m) = \frac{1}{2}\|\varepsilon_b\|_{\mathbf{B}}^2 + \frac{1}{2}\sum_{t=0}^{T} \|\varepsilon_{R_t}\|_{\mathbf{R}_t}^2 + \frac{1}{2}\sum_{t=1}^{T-1} \|\varepsilon_{m_t}\|_{\mathbf{Q}_t}^2$$

What motivates this cost function is that under independent Gaussian errors and linear model hypothesis, minimizing it leads to the maximum *a posteriori* estimation of the state vector.

Gradient. Usually, PDE-constrained optimization in inverse problems is solved with adjoint state methods [10]. In our case of variational data assimilation, calculus of variations provides an analytical expression of ∇J:

$$\nabla_{\varepsilon_b} J = \mathbf{B}^{-1}\varepsilon_b - \sum_{t=0}^{T} \left[\mathbb{H}_t \frac{\partial \mathbb{M}_{0\to t}}{\partial \mathbf{X}}\right]^\top \mathbf{R}_t{}^{-1}\varepsilon_{R_t}$$

$$\nabla_{\varepsilon_{m_t}} J = \mathbf{Q}^{-1}\varepsilon_{m_t} - \sum_{t'=t+1}^{T} \left[\mathbb{H}_{t'} \frac{\partial \mathbb{M}_{t\to t'}}{\partial \mathbf{X}}\right]^\top \mathbf{R}_{t'}^{-1}\varepsilon_{R_{t'}}$$

A detailed proof can be found in [11]. Computing such gradients thus requires having numerical representations of adjoint linear tangent models $\left[\frac{\partial \mathbb{M}_{0\to t}}{\partial \mathbf{X}}\right]^\top$. By using deep learning tools based on automatic differentiation and focused on backpropagation, we can obtain those gradients with much less effort. Adjoint modeling being embedded by the software deriving the computational graph, the only requirement is to use differentiable operations.

Computational graph. In Figure 1, we present a schematic view of the forward integration leading to the calculation of J. Designing such forward mapping is very similar to designing a neural network architecture and the associated cost function.

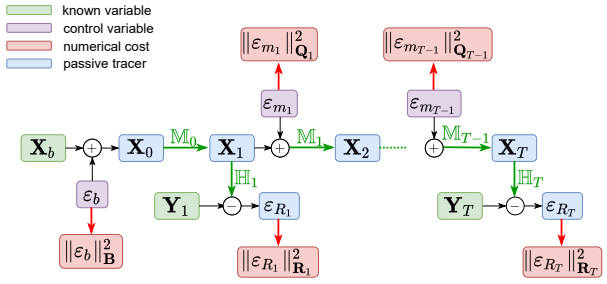


Figure 1: Computational graph of J calculation

Algorithm. The algorithm optimizing the presented cost function is named 4D-Var. It simply consists of alternating forward and backward integrations to update control parameters by gradient descent (see Algorithm 1). When the dynamical model is considered imperfect, we use the weak-constraint version of 4D-Var [12]. In the other case, we use the strong-constraint version which is a particular case supposing errors model, ε_m, to be null.

[1] https://github.com/ArFiloche/Py4DVar.git

After convergence, the whole system state is estimated and the end of the temporal window can be used as initial conditions to produce a forecast.

Algorithm 1 – Weak-constraint 4D-Var

Initialize control variable ε_b, ε_m

forward: integrate $\mathbf{X}$ and compute J
backward: automatic differentiation returns ∇J

while stop criterion **do**
  update control variables:
  $\varepsilon_b, \varepsilon_m$ = optimizer$(\varepsilon_b, \varepsilon_m, J, \nabla J)$

  forward: integrate $\mathbf{X}$ and compute J
  backward: automatic differentiation returns ∇J

return $\mathbf{X}$

3 Motion Estimation

To test our approach, we use two different geophysical systems evolving over a discrete space-time domain $\Omega \times [0:T]$ where Ω is a bounded domain of $\mathbb{Z}^2$. At each time $\mathbf{X}_t = \begin{pmatrix} \mathbf{w}_t & I_t \end{pmatrix}^\top$ is composed of a physical variable tracer I_t and the associated motion field $\mathbf{w}_t$. Observations on I are available at regular date in time but the $\mathbf{w}$ component is never observed. This means that at observational date t, the observation operator is a linear projection such that $\mathbb{H}_t \mathbf{X}_t = I_t$. Having no prior information on velocity, we simply use the first observation as background state so that $\mathbf{X}_b = \begin{pmatrix} 0 & I_0 \end{pmatrix}^\top$. Regarding covariance matrices $\mathbf{B}$, $\mathbf{R}$ and $\mathbf{Q}$, they are set in accordance with experiments conducted in [8, 9]. The main goal of our assimilation process will be to estimate $\mathbf{w}_t$ hence the motion estimation naming.

3.1 Geophysical images

Rain radar images. In Figure 2, we display four consecutive radar rain maps obtained from MeteoNet network [13]. These images have been acquired over the region of Brest during an extremely rainy episode occurred on January 3, 2018. The time step is 5 minutes, and the spatial resolution is approximately 1 square kilometer. We propose to estimate the velocity map from 4 consecutive acquisitions. As we do not have ground-truth, the fourth acquisition will be extrapolated in time, using the estimated velocity map, to provide forecasts at various short time horizon. This is called *Rain Nowcasting*. Forecast images will be compared to Radar rainmap acquisitions to provide relevant performance statistics.

Sea surface temperature images. Figure 3 shows four sea surface temperature maps (SST) over a small area of the North Atlantic Ocean. Data were obtained from

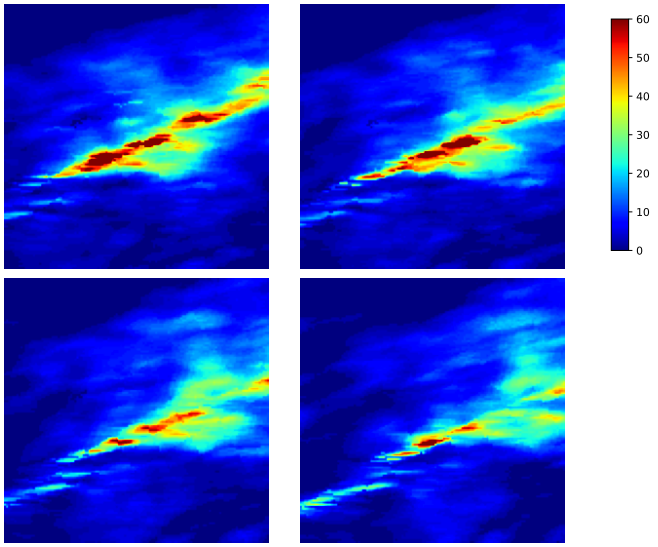


Figure 2: Four successive acquisitions of rainfall maps. Intensity unit is millimeter per hour

the Marine Copernicus Service[2] and are the reanalyze of satellite observations through a physical model of the Ocean. Time step is of 1 day, and spatial resolution of 10 square kilometers. Similarly to rain maps, we propose

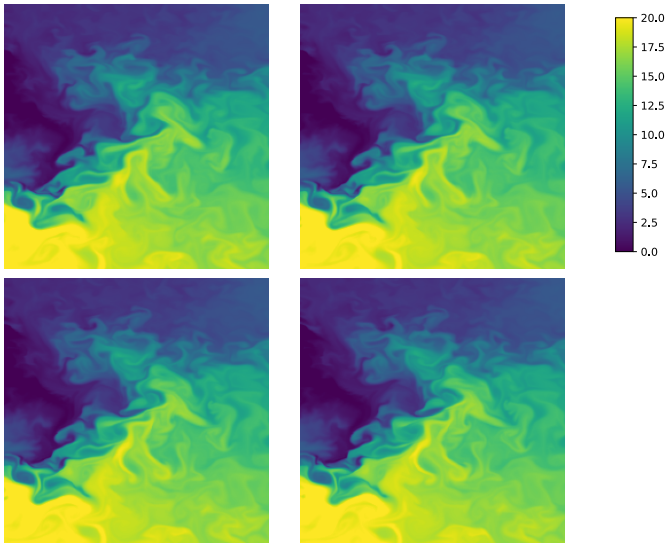


Figure 3: Four successive SST acquisitions of North Atlantic. Intensity unit is Celsius degree

to estimate velocity maps from 4 consecutive SST maps. Sea surface circulation data are available in Copernicus and used as reference.

3.2 Dynamical model

The physical processes we consider can partially be described by the linear and non-linear advection equations

[2] https://resources.marine.copernicus.eu/?option=com_csw&task=results

in Eq. (4). The motion field $\mathbf{w}$ transports the passive tracer I and also itself

$$\begin{cases} \dfrac{\partial I}{\partial t} + \mathbf{w}.\nabla I = 0 \\ \dfrac{\partial \mathbf{w}}{\partial t} + \mathbf{w}.\nabla \mathbf{w} = 0 \\ \nabla I = 0, \quad \partial\Omega \\ \mathbf{w} = 0, \quad \partial\Omega \end{cases} \tag{4}$$

The use of this model is physically justified in both our geophysics experiments. For rain nowcasting on a short time window, it describes well the transport of storm cells by wind, cells formation being neglected. The perfect model hypothesis is then acceptable and the strong-constraint 4D-Var is sufficient to estimate cells velocities (motion fields). For the estimation of sea surface circulation (motion fields) from SST images, Eq (4) constitutes an approximation of Navier-Stokes equations used in the reanalysis therefore we use the weak-constraint 4D-Var.

Discretization of Eq. (4) in time and in space provides the operator $\mathbb{M}$. We use a semi-Lagrangian scheme [14] to approximate linear and non linear advection (respectively $\mathbf{w}.\nabla I$ and $\mathbf{w}.\nabla \mathbf{w}$ terms). The semi-Lagrangian scheme is an implicit scheme, and not subject to CFL condition. That implies the time step can be chosen independently to the space step and velocity magnitude compared to explicit schemes. The implementation of this scheme in Pytorch can be found in our github.

3.3 Regularization

As the motion estimation inverse problem is ill-posed, we employ Tikhonov regularization. We enforce the motion field to be smooth by penalizing $\|\nabla \mathbf{w}_0\|_2^2$ and $\|\nabla.\mathbf{w}_0\|_2^2$. At first glance those terms does not seem to appear in the cost function J but as depicted in [15], they can be implicitly included in the background error cost $\frac{1}{2}\|\varepsilon_b\|_{\mathbf{B}}^2$. This is done by using a Toeplitz matrix for $\mathbf{B}^{-1}$ where descending diagonals are filled with regularization parameters. The same remark can be done about the model errors ε_m that also should be constrained to be smooth. This is embedded in matrix $\mathbf{Q}^{-1}$.

3.4 Forecast

At the end of the assimilation process, an estimation of the full system state trajectory is available. This means that at each time t we have an estimation $\widehat{\mathbf{X}}_t = \begin{pmatrix} \widehat{\mathbf{w}}_t & \widehat{I}_t \end{pmatrix}^\top$. To produce a forecast at given time horizon, $T + k$, we simply integrate the evolution model using the end of the assimilation window as initial condition: $\widehat{\mathbf{X}}_{T+k} = \mathbb{M}_{T\to T+k}(\widehat{\mathbf{X}}_T)$. In the forecast set up, model errors ε_m can not be estimated and therefore are not considered.

4 Results

In this section we present experimental results obtained on both dataset. We compare our newly developed Pytorch code with a C code, already used in several experiments [8, 9] where the adjoint dynamics is obtained with Tapenade.

4.1 Nowcasting on rain radar images

After strong 4D-Var assimilation of 4 rain radar images (Figure 2) on a window representing a 15-minute period, we integrate the estimated state further in time to produce forecast at 10, 20, and 30 minutes horizons. As depicted in Figure 4 both versions of the algorithm behave in the same manner. Also, we note a smoothing effect which is an issue of the semi-Lagrangian scheme and the counterpart of an implicit scheme.

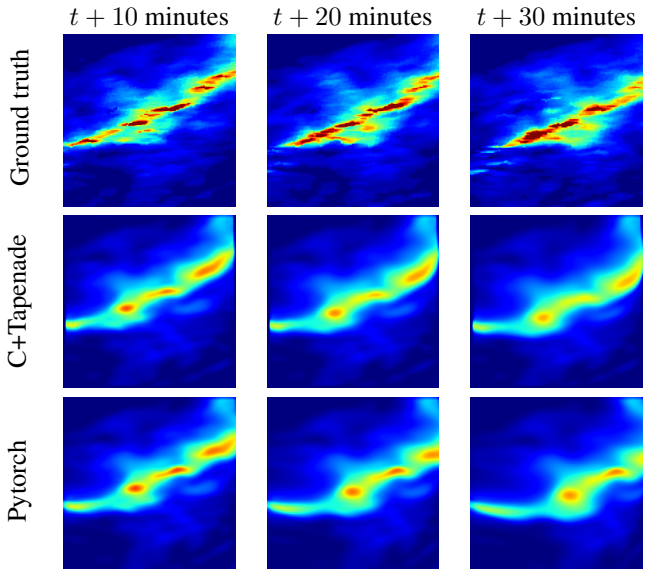


Figure 4: Example of rainmap forecasts at three horizon times produced by the C code and the Python code

When repeating the assimilation algorithm over sliding windows, we can evaluate our forecast over longer time period. In Figure 5 we consider a full day containing almost 800 acquisitions. After each assimilation, the forecasts are evaluated against the available ground truth using RMSE on the tracer I_t. Dealing with rain nowcasting, other metrics can be more informative than RMSE but in our case it is enough to verify that both codes produce comparable performances.

4.2 Sea surface circulation from SST images

The exact same process is applied for SST forecasting. The 3-day windows are assimilated with the weak-constraint algorithm and forecasts at three horizon times as shown in Figure 6. SST circulation being a slow evolving dynamics, it is hard to visualize differences. However, we can similarly calculate the RMSE between the tracer I_t and the ground truth as plotted in Figure 7.

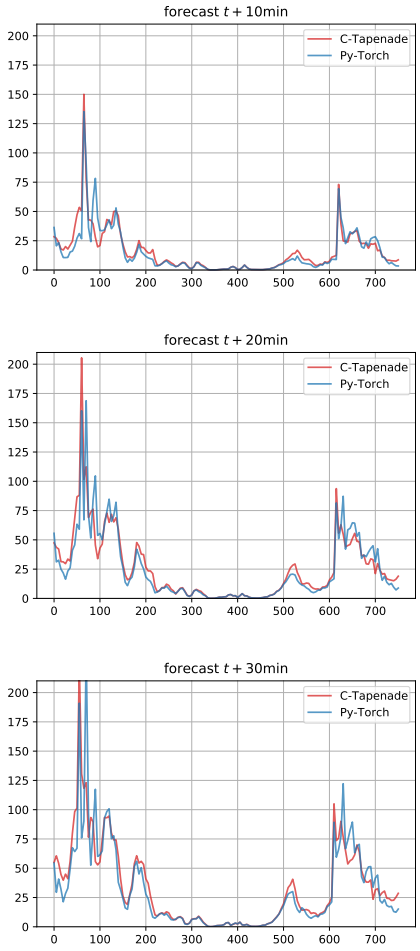


Figure 5: Rainfall forecasts at 10, 20 and 30 minutes: performances of C and Pytorch codes (RMSE)

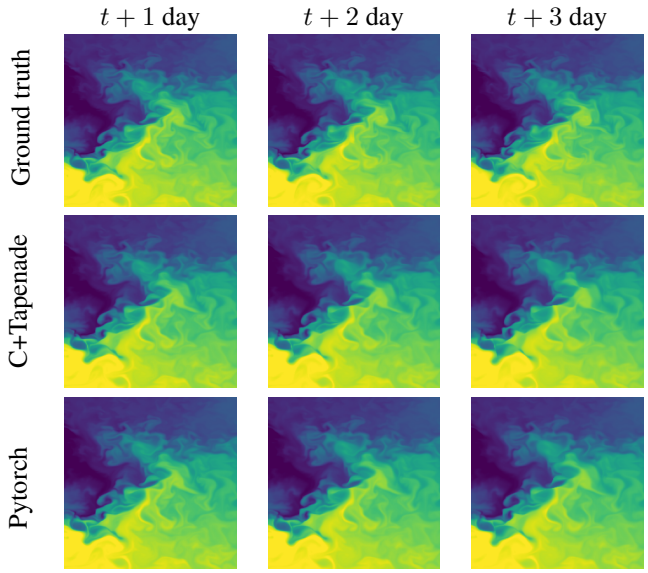


Figure 6: Example of SST forecasts at three horizon times produced by the C code and the Python code

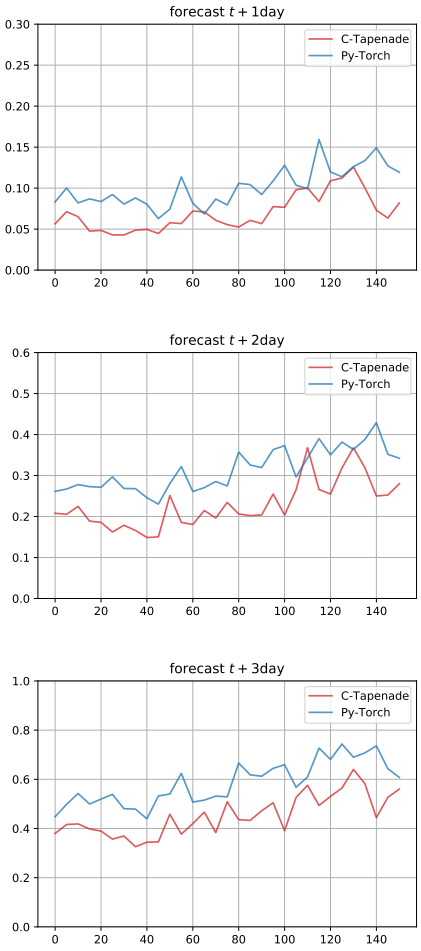


Figure 7: SST forecast: performances of C and Pytorch codes for three horizon times

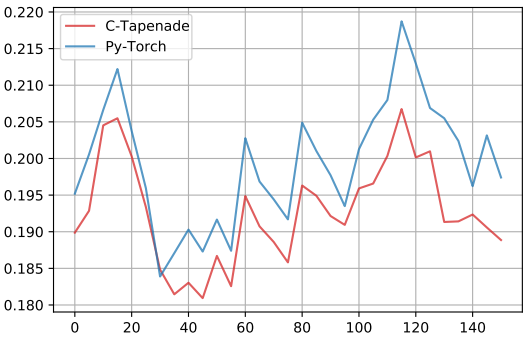
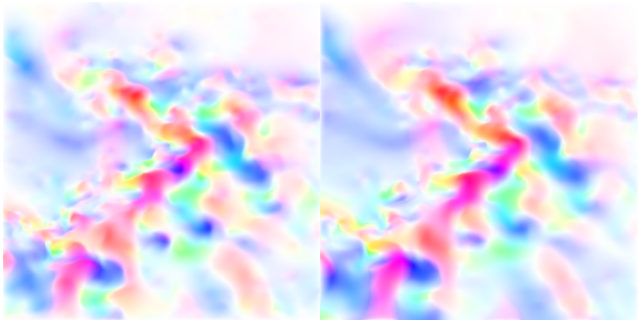
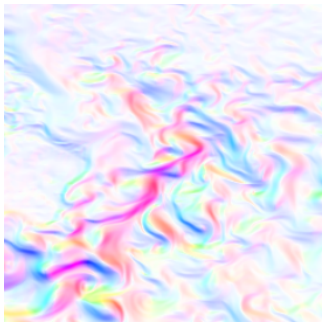


Figure 8: Performance of C and Pytorch codes, metric is the EPE

In the particular case of SST data coming from an other assimilation process, we have access to ground truth motion fields $\mathbf{w}$. In Figure 9, we can see assimilated motion fields again the ground truth, colors standing for motion vectors orientation [16]. We also can quantify these



(a) C+Tapenade (b) Pytorch



(c) Ground-truth

Figure 9: Example of velocity maps obtained with C and Pytorch codes, compared to the ground-truth

differences over time by calculating the end-point-error (EPE) between assimilated fields and the ground truth as presented in Figure 8. Again, we can conclude that both codes are roughly equivalent.

4.3 Discussion

The new code aims at replicating the reference as close as possible. However results obtained are not exactly the same and we believe it can be explained by several reasons. First the gradients coming from regularization terms are calculated manually in the C version but automatically in the other one. Second, even though both codes use an optimizer based on the same second order quasi-Newton method, these optimizers came from different development. C code uses the original implementation of L-BFGS [17] written in Fortran by the authors. Pytorch use its own implementation. Third, the order of successive operations is not strictly respected which may lead to numerical rounding differences.

Lastly, we compare in Table 1 computation time needed in each cases using a standard CPU architecure. Unsurprisingly, the C code version is much more faster, the Pytorch simplicity could not be free. Also, we note that the gap between versions is more important in the strong case and this is because the C code has been optimized using high performance computing technique based on vectorization and parallel programming.

Code paradigm	Rain (strong)	SST (weak)
C-Tapenade	2.0 ± 0.3 s	25.8 ± 1.1 s
Pytorch	90.0 ± 43.6 s	71.3 ± 1.1 s

Table 1: Computation time

5 Conclusion

In this work, we showed that it is possible to use deep learning tools outside the scope of neural networks. More precisely, we employ them to solve geophysical inverse problems requiring automatic differentiation for the adjoint state method. Usually, solving such variational problems numerically is non-trivial but these new tools make it much more accessible. We compared the results we obtain with a trustworthy code and show that there is no significant difference in terms of performance. However, we found that this simplicity and flexibility come at a computational cost. Regarding Earth sciences and modeling, we truly believe in this type of tools to unify data-driven and knowledge driven methods.

References

[1] O. Talagrand. Assimilation of observations, an introduction. *Journal of Meteorological Society of Japan*, 75(1B):191–209, 1997.

[2] G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[3] F-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus*, 38A(10):97, 1986.

[4] L. Hascoët and V. Pascual. The Tapenade Automatic Differentiation tool: Principles, Model, and Specification. *ACM Transactions On Mathematical Software*, 39(3), 2013.

[5] Abarbanel H., P. Rozdeba, and S. Shirman. Machine learning: Deepest learning as statistical data assimilation problems. *Neural Computation*, 30(8):2025–2055, 2018.

[6] M. Bonavita, Y. Trémolet, E. Holm, T.K. Lang, M. Chrust, M. Janiskova, P. Lopez, P. Laloyaux, P. de Rosnay, Hisher M., M. Hamrud, and English Stephen. A strategy for data assimilation. Technical Report 800, ECMWF, April 2017.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[8] D. Béréziat and I. Herlin. Motion and Acceleration from Image Assimilation with evolution models. *Digital Signal Processing*, 83:45–58, 2018.

[9] A. Zebiri, D. Béréziat, E. Huot, and I. Herlin. Rain nowcasting from multiscale radar images. In *International Conference on Computer Vision Theory and Applications*, 2019. `https://hal.sorbonne-universite.fr/hal-02048500v1`.

[10] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.

[11] M. Asch, M. Bocquet, and M. Nodet. *Data assimilation: methods, algorithms, and applications*. Fundamentals of Algorithms. SIAM, 2016.

[12] Y. Trémolet. Accounting for an imperfect model in 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 132:2483–2504, 2006.

[13] Gwennaëlle Larvor, Léa Berthomier, Vincent Chabot, Brice Le Pape, Bruno Pradel, and Lior Perez. Meteonet, an open reference weather dataset by meteo-france, 2020. `https://meteonet.umr-cnrm.fr/`.

[14] A. Staniforth and J. Côté. Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review. *Monthly Weather Review*, 119(9):2206–2223, 09 1990.

[15] Y. Lepoittevin and I. Herlin. Regularization terms for motion estimation. Links with spatial correlations. In *VISAPP - International Conference on Computer Vision Theory and Applications*, volume 3, pages 458–466, 2016.

[16] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal on Computer Vision*, 92(1):1–31, March 2011.

[17] J. L. Morales and J. Nocedal. Remark on algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transaction on Mathematical Software*, 38(1):7:1–7:4, December 2011.