6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
11 - 15 June 2018, Glasgow, UK

VALIDATION OF DIFFERENTIATED PROGRAMS AND THEIR APPLICATION TO DATA ASSIMILATION

Yasuyoshi Horibata

Hosei University
Kajino-cho, Koganei, Tokyo, Japan
horibata@hosei.ac.jp

Key words: Automatic Differentiation, Adjoint Program, Optimization, Dot Product Test, Downburst, Data Assimilation

Abstract. Meteorological data assimilation is formulated as a large-scale nonlinear optimization problem. At each search, the optimization algorithm requires the gradient of a cost function. The gradient is computed efficiently by the adjoint program. A source computer program that simulates the atmospheric flow is differentiated in reverse mode by the automatic differentiation tool TAPENADE, and the adjoint program is generated. In this paper, we improve the differentiability of the source program in order to generate the more exact tangent program. The generated tangent and adjoint programs are validated by dot product test. Preliminary numerical experiments are presented for data assimilation.

1 INTRODUCTION

Meteorological data assimilation is formulated as a large-scale nonlinear optimization problem [1, 2]. At each search, the optimization algorithm requires the gradient of a cost function. We compute the gradient by using the adjoint program generated by the automatic differentiation tool TAPENADE; it differentiates in reverse mode a source computer program that simulates the atmospheric flow. In [2], we presented numerical experiments for a three-dimensional downburst. However, the optimization result was not satisfactory. This is because the generated tangent and adjoint programs were not perfect.

In this paper, we identify the problems, and modify the source simulation program to solve the problems. We validate the tangent and adjoint programs which the AD tool generates. Then, using the adjoint program, the optimization is done for the meteorological data assimilation. Preliminary numerical experiments are presented.

2 SOURCE PROGRAM

The source program simulates the atmospheric flow [3]. The governing equations contains prognostic equations for the air velocity, pressure, potential temperature, water

vapor, rainwater, cloud water, and subgrid-scale kinetic energy. The finite difference scheme integrate these equations.

In this paper, a downburst is simulated using the source program. A downburst is a strong downdraft which induces an outburst of damaging winds on or near the ground.

The simulation domain is 20 km in both horizontal directions and 10 km in the vertical. The origin of the coordinate system is located at the center of the base of the simulation domain. The initial undisturbed state is isentropic ($\theta_0 = 300K$) and has zero relative humidity. The initial wind speed is assumed to be zero.

A downburst is initiated by introducing a distribution of rainwater.

Simulation is run to 6 minutes using the grid intervals $\Delta x = \Delta y = 1250m$, $\Delta z = 625m$ and the time step $\Delta t = 5s$.

Figure 1 shows the evolution of the flow field, whereas Figure 2 shows the evolution of the flow and rainwater fields in a vertical plane. Figures 3 ,4, and 5 show the water vapor, potential temperature, and pressure deviation fields in the vertical plane, respectively. The latent heat is required from the ambient air while the rainwater is evaporating. As a result, the air becomes colder and heavier, and yields a downdraft, which achieves its maximum velocity of 21 m/s 4 minutes after the initiation. Then, the downdraft hits the ground and diverges horizontally. The horizontal velocity becomes as high as 17 m/s near the ground 6 minutes after the initiation.

3 VALIDATION

3.1 Differentiability

The source program has the instruction

```
a = b ** 0.525
```

For the instruction, in the tangent mode, the AD tool generates the differentiated instruction:

```
ad = 0.525 bd/b ** 0.475
```

This returns NaN if x is zero.

In [1, 2], in order to circumvent this singularity, we used Lagrange interpolation; using four points

$$x_1 = 0.0, \quad x_2 = 0.5 \times 10^{-2}, \quad x_3 = 1 \times 10^{-2}, \quad x_4 = 1.5 \times 10^{-2}$$

in the interval $0 \leq x \leq 1.5 \times 10^{-2}$, we approximated the function $x^{0.525}$ by constructing a polynomial of degree 3 that matches the function at the four points. The maximum value of x is around 1.5×10^{-2}.

On the other hand, in the present paper, we use Hermite interpolation only in the neighborhood of $x = 0$. We use onlyt two data points

$$\begin{aligned} x_1 &= 0.0 \\ x_2 &= 10^{-4} \end{aligned}$$

The value of the function $x^{0.525}$ is used for the value of the function y_i at each data point:

$$\begin{aligned} y_1 &= 0.0 \\ y_2 &= x_2^{0.525} \end{aligned}$$

The derivative y_1' at x_1 is set to zero, whereas the derivative of the function $x^{0.525}$ is used for the derivative y_2' at x_2:

$$\begin{aligned} y_1' &= 0.0 \\ y_2' &= 0.525 x_2^{-0.475} \end{aligned}$$

We construct a polynomial of degree 3 that matches all the data. Figure 6 shows the curves of the function $x^{0.525}$ and the interpolation. The Hermite interpolation hardly changes the simulation results obtained by the original source program which uses the function $x^{0.525}$.

3.2 Differentiated programs

The source program is differentiated by Tapenade. We validate the generated differentiated programs [4].

Given a vector argument $X \in R^m$, a source computer program computes some vector function $Y = F(X) \in R^l$. The AD tool generates a new source program that, given the argument X, computes some derivatives of F; the output of the tangent program is $\dot{Y} = F'(X) \times \dot{X}$, whereas the output of the adjoint program is $\bar{X} = F'^*(X) \times \bar{Y}$. Here, $F'(X)$ is the Jacobian of $F(X)$.

Introducing a function g of a scalar variable h: $g(h) = F(X + h \times \dot{X})$, we obtain

$$\lim_{\epsilon \to 0} \frac{F(X + \epsilon \times \dot{X}) - F(X)}{\epsilon} = g'(0) = F'(X) \times \dot{X} = \dot{Y} \tag{1}$$

Thus, we can approximate $\dot{Y}$ by running F on X and on $X + \epsilon \times \dot{X}$.

Taking the output $\dot{Y}$ of the tangent program as the input $\bar{Y}$ of the adjoint program, we obtain

$$(\bar{X} \cdot \dot{X}) = (F'^*(X) \times \dot{Y} \cdot \dot{X}) = \dot{Y}^* \times F'(X) \times \dot{X} = \dot{Y}^* \times \dot{Y} = (\dot{Y} \cdot \dot{Y}) \tag{2}$$

In our source program, X is the initial conditions of the air velocity, pressure, potential temperature, water vapor, rainwater, cloud water, eddy mixing coefficient, whereas Y is the corresponding fields after 6 min; $m = l = 37632$.

For the simulation result shown in 2, we compute three norms. Table 1 compares the three norms. The tangent norm $(\dot{Y} \cdot \dot{Y})$ and the adjoint norm $(\bar{X} \cdot \dot{X})$ match completely up to the last digit, whereas the norm obtained with Divided Differences $(F(X + \epsilon \times \dot{X}) - F(X))/\epsilon$ matches up to about half the machine accuracy.

Table 1: Comparison of three norms ($\epsilon = 10^{-10}$)

Divided Differences $(F(X + \epsilon \times \dot{X}) - F(X))/\epsilon$ norm	0.153194571523625D+21
Tangent norm $(\dot{Y} \cdot \dot{Y})$	0.153194435697253D+21
Adjoint norm $(\bar{X} \cdot \dot{X})$	0.153194435697253D+21

4 DATA ASSIMILATION

4.1 Cost function

The vector function $F(X)$ is a composition of vector functions:

$$F = f_N \circ f_{N-1} \circ \cdots \circ f_2 \circ f_1 \tag{3}$$

where each f_n is the elementary function, and

$$X_n = f_n(X_{n-1}) \quad (n = 1, 2, \cdots, N) \tag{4}$$

Here, X_n is the fields at time $t_n = n\Delta t$, and

$$Y = F(X) \tag{5}$$

with

$$X = X_0 \tag{6}$$
$$Y = X_N \tag{7}$$

The chain rule gives the Jacobian $F'(X)$:

$$F'(X) = f'_N(X_{N-1}) \times f'_{N-1}(X_{N-2}) \times \cdots \times f'_2(X_1) \times f'_1(X_0) \tag{8}$$

The cost function is taken as

$$J = \sum_{n=0}^{N} H_n(X_n) \tag{9}$$

where $H_n(X_n)$ is a scalar measuring the distance between X_n and its observations available at time $t_n = n\Delta t$. The available observations are assumed to be distributed over a limited time interval $[t_0, t_N]$. For given initial conditions X_0 and for the corresponding outputs X_n of the source program, the cost function is evaluated. Thus, the cost function is a function of the initial conditions X_0, and its gradient with respect to X_0 is [5]

$$\nabla_{X_0} J = \sum_{n=1}^{N} f'^*_1(X_0) \times \cdots \times f'^*_{n-1}(X_{n-2}) \times f'^*_n(X_{n-1}) \times \nabla_{X_n} H_n(X_n) + \nabla_{X_0} H_0(X_0) \tag{10}$$

In assimilation of meteorological observations, the cost function is minimized. The optimization problem has thousands or millions of variables. Modified L-BFGS-B algorithm [6, 7] is employed in order to solve this problem. At each search, the gradient of the cost function is computed from (10) using the adjoint program.

4.2 Preliminary numerical experiments

Numerical experiments are presented for a three-dimensional downburst.
We take $H_n(X_n)$ as

$$H_n(X_n) = c_u \sum_{i,j,k} (u^n_{ijk} - u^{ob\,n}_{ijk})^2 + c_{q_r} \sum_{i,j,k} (q^n_{r\,ijk} - q^{ob\,n}_{r\,ijk})^2 \tag{11}$$

where u^n_{ijk} and $q^n_{r\,ijk}$ are the x component of the air velocity and the rainwater, respectively, at the grid point (i, j, k) and time t_n, and $u^{ob\,n}_{ijk}$ and $q^{ob\,n}_{r\,ijk}$ are their corresponding observations, respectively; the coefficients c_u and c_{q_r} are weights..

Out of the simulation result shown in 2 , the x component of the air velocity and the rainwater are assumed to be observations from a Doppler radar.

In order to estimate the initial conditions used for the simulation, the modified L-BFGS-B algorithm solves the large-scale optimization problem. Starting from the initial guess, the algorithm repeats search until it reaches the optimal point. Figure 7 shows the rainwater field of the starting point used for the optimization. All the other fields of the starting point are the same as ones used in the simulation.

Figure 8 shows the convergence history of the optimization. The cost function decreases by a factor of 10^{-6} after 300 iterations. On the other hand, in [2] , it decreased only by a factor of 10^{-2}.

Figures 9, 10, 11, 12, and 13 compare the rainwater, water vapor, potential temperature, pressure, and flow fields recovered by the optimization and the corresponding ones of the initial conditions used in the simulation in the $x-z$ cross section at $y = 0$, respectively.

5 CONCLUSIONS

- The source program is modified, and is differentiated in tangent and reverse modes by the AD tool TAPENADE. Three norms computed by the programs are roughly the same; especially, the tangent norm and the adjoint norm are identical. The modification hardly affects the simulation results.

- In the data assimilation, the gradient of the cost function with respect to the initial conditions is calculated by using the generated adjoint program. The cost function decreases by a factor of 10^{-6}.

- The recovered fields agree reasonably with the initial condition used for producing pseudo observations.

REFERENCES

[1] Y. Horibata, Adjoint program generated by automatic differentiation of a meteorological simulation program, and its application to gradient computation, *Proceedings of the 6th International Conference on Computational Methods*, 14th - 17th July 2015, Auckland, G.R. Liu and Raj Das, Paper ID 861, ScienTech Publisher, New Zealand, 2015.

[2] Y. Horibata, Meteorological data assimilation using an adjoint program generated by automatic differentiation, Proceedings of the 7th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016), June 5-10, 2016, Crete Island, Greece, pp. 8275-8282, 2016.

[3] Y. Horibata, H. Oikawa, Numerical Simulation of Downbursts Using a Terrain-following Coordinate Transformation, *Proc. of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Germany, pp. 1002-1009, 1994.

[4] L. Hascoet, V. Pascual, TAPENADE 2.1 user's guide, INRIA, France, 2004.

[5] Kalnay, E. *Atmospheric modeling, data assimilation, and predictability.* Cambridge University Press, 2003.

[6] C. Zhu, R. H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-BFGS-B, FORTRAN subroutines for large scale bound constrained optimization, *ACM Transactions on Mathematical Software* (1997) **23**:550-560.

[7] J.L. Morales, J. Nocedal, L-BFGS-B: Remark on Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization, *ACM Transactions on Mathematical Software*, **23**, Article 7, 2011.

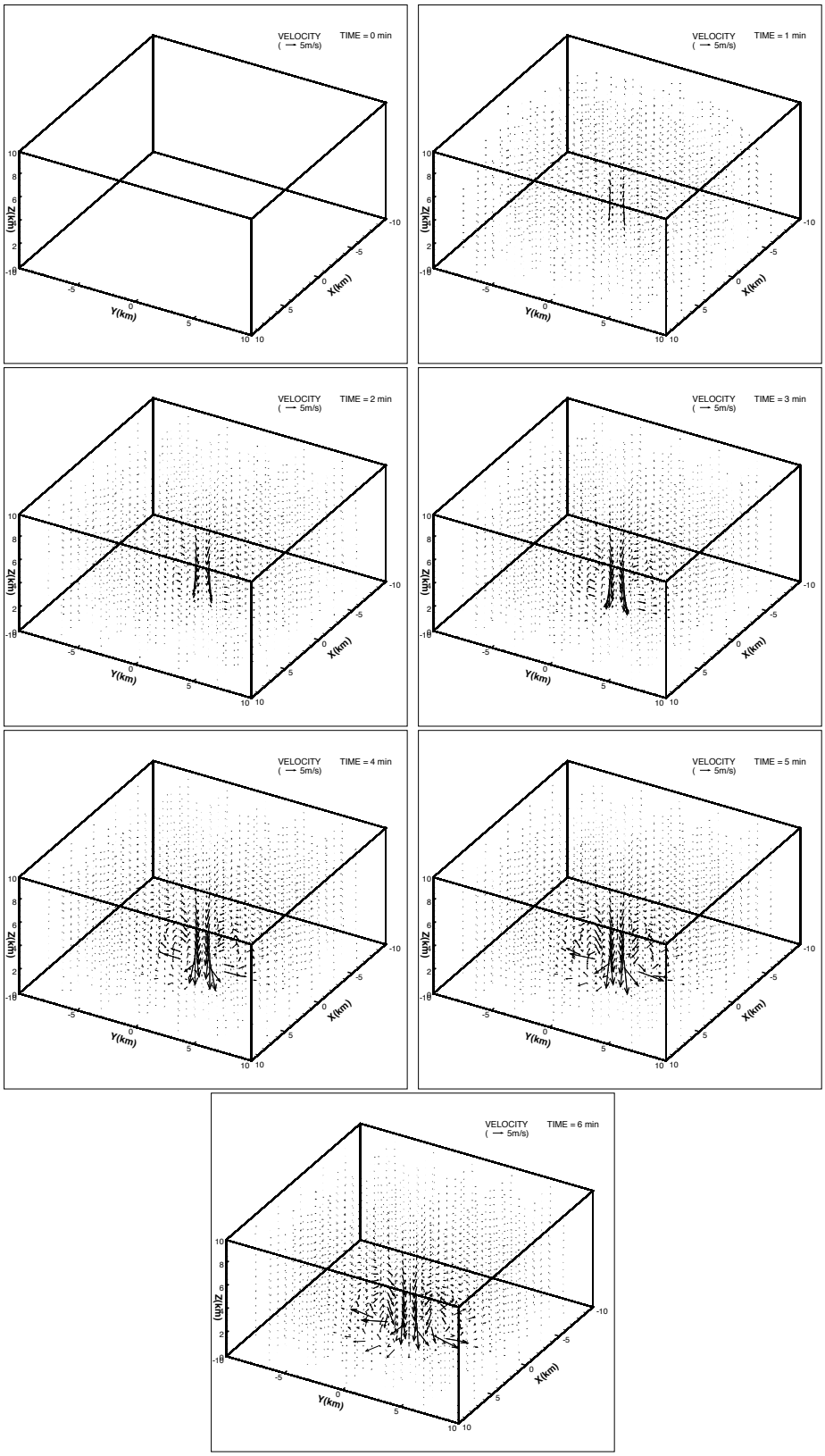

Figure 1: Evolution of the flow field after the downburst initiation.

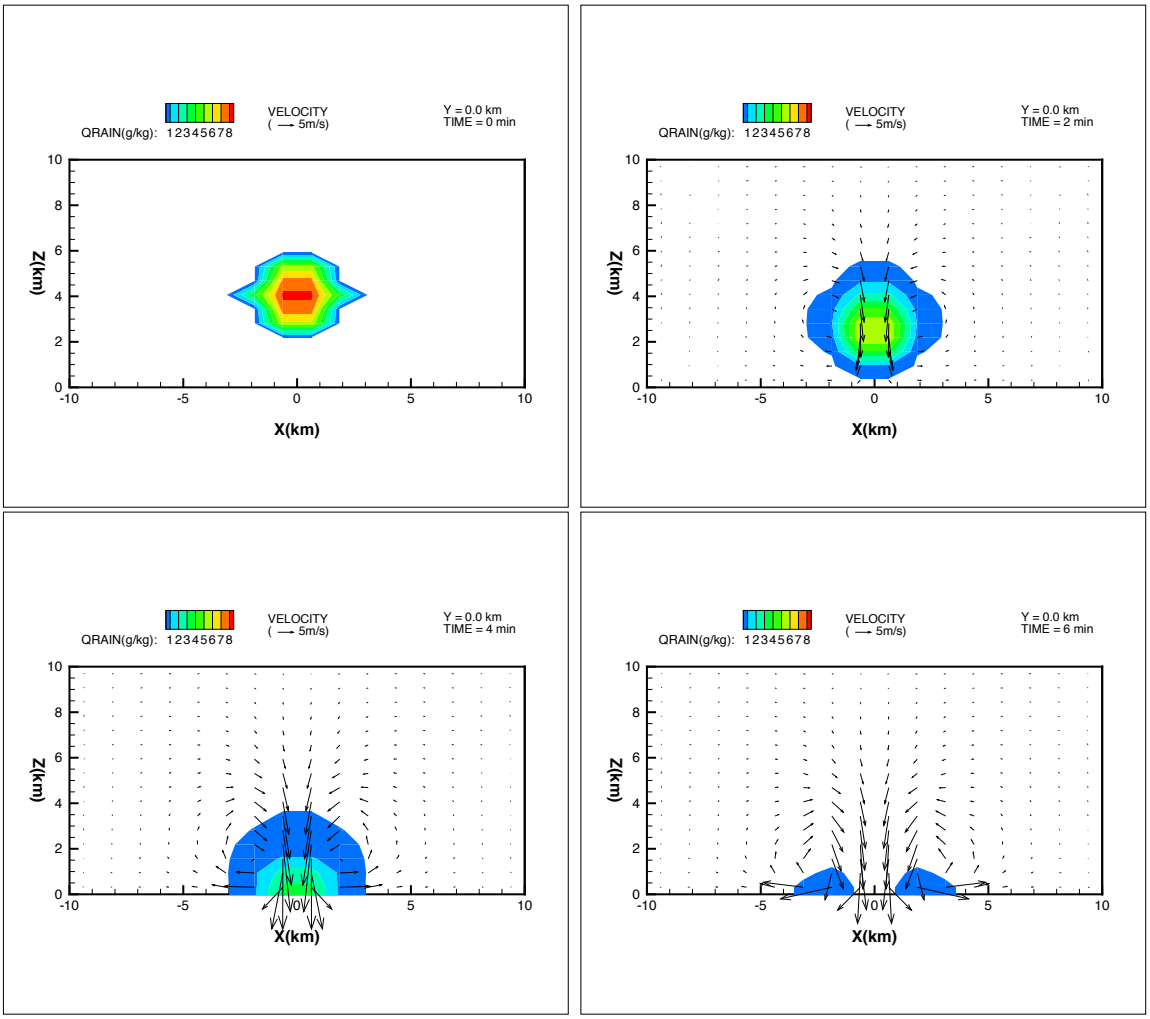

Figure 2: Evolution of the flow and rainwater fields after the downburst initiation in the $x - z$ cross section at $y = 0$.

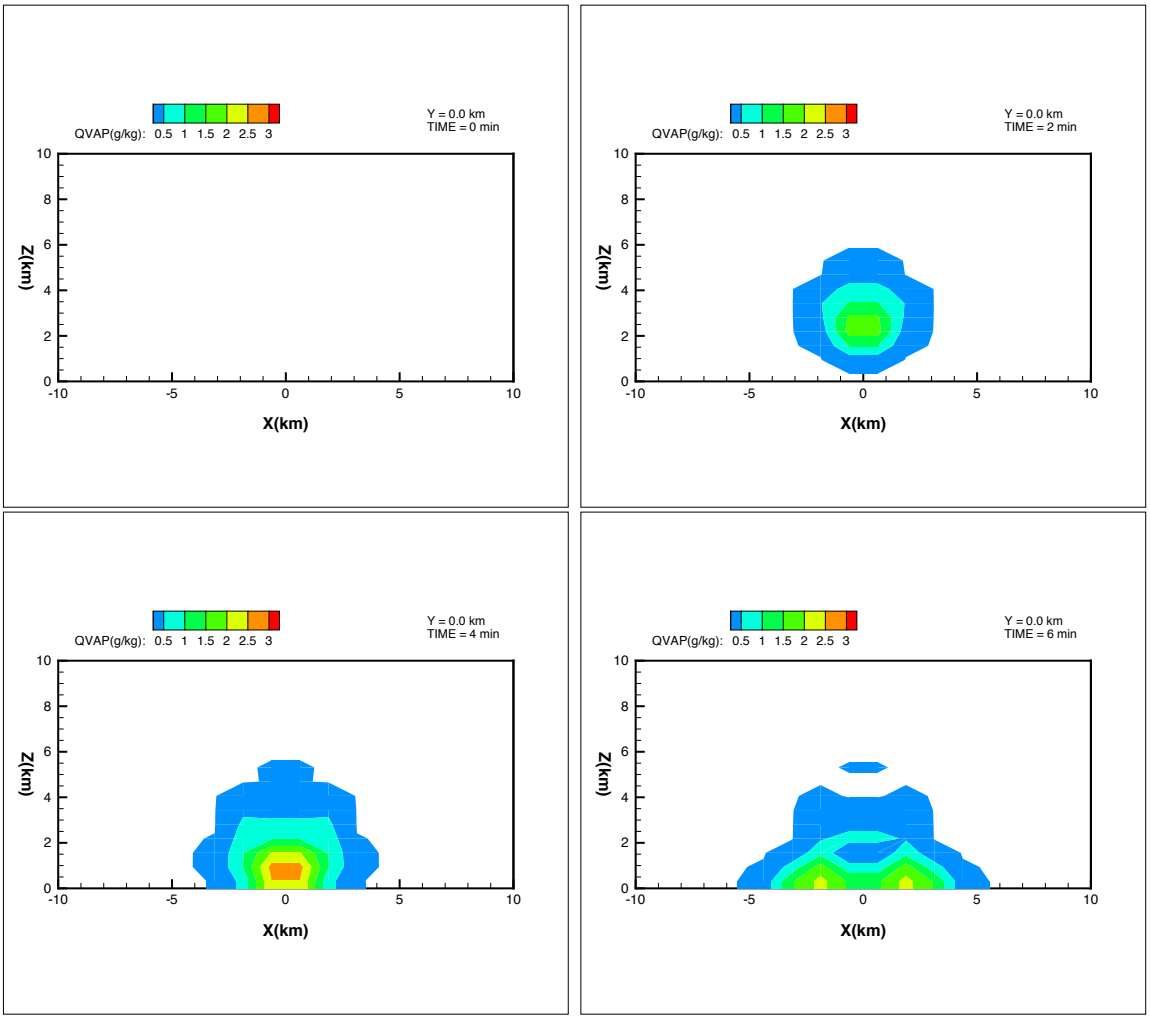

Figure 3: Evolution of the water vapor field after the downburst initiation in the $x - z$ cross section at $y = 0$.

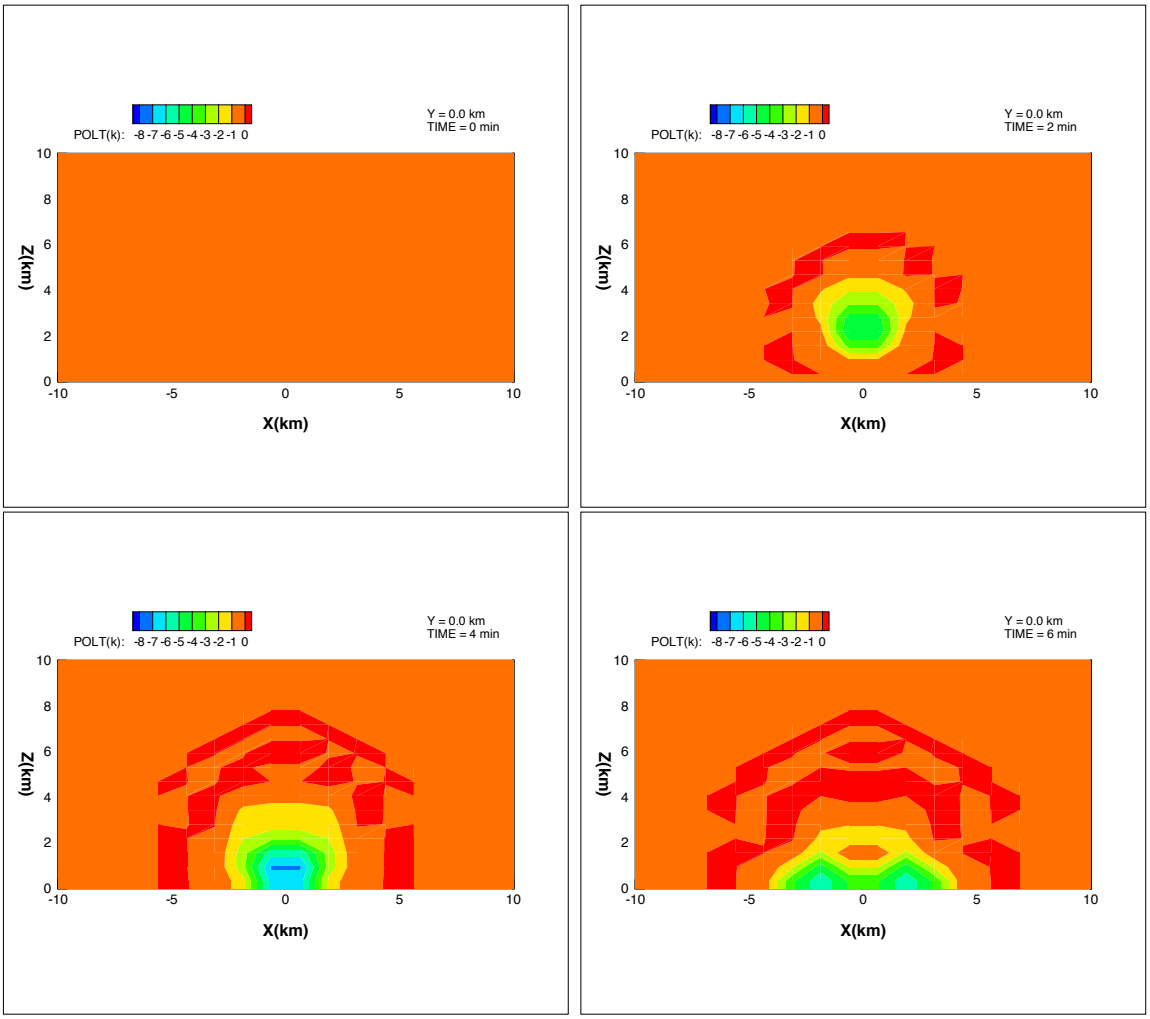

Figure 4: Evolution of the potential temperature field after the downburst initiation in the $x - z$ cross section at $y = 0$.

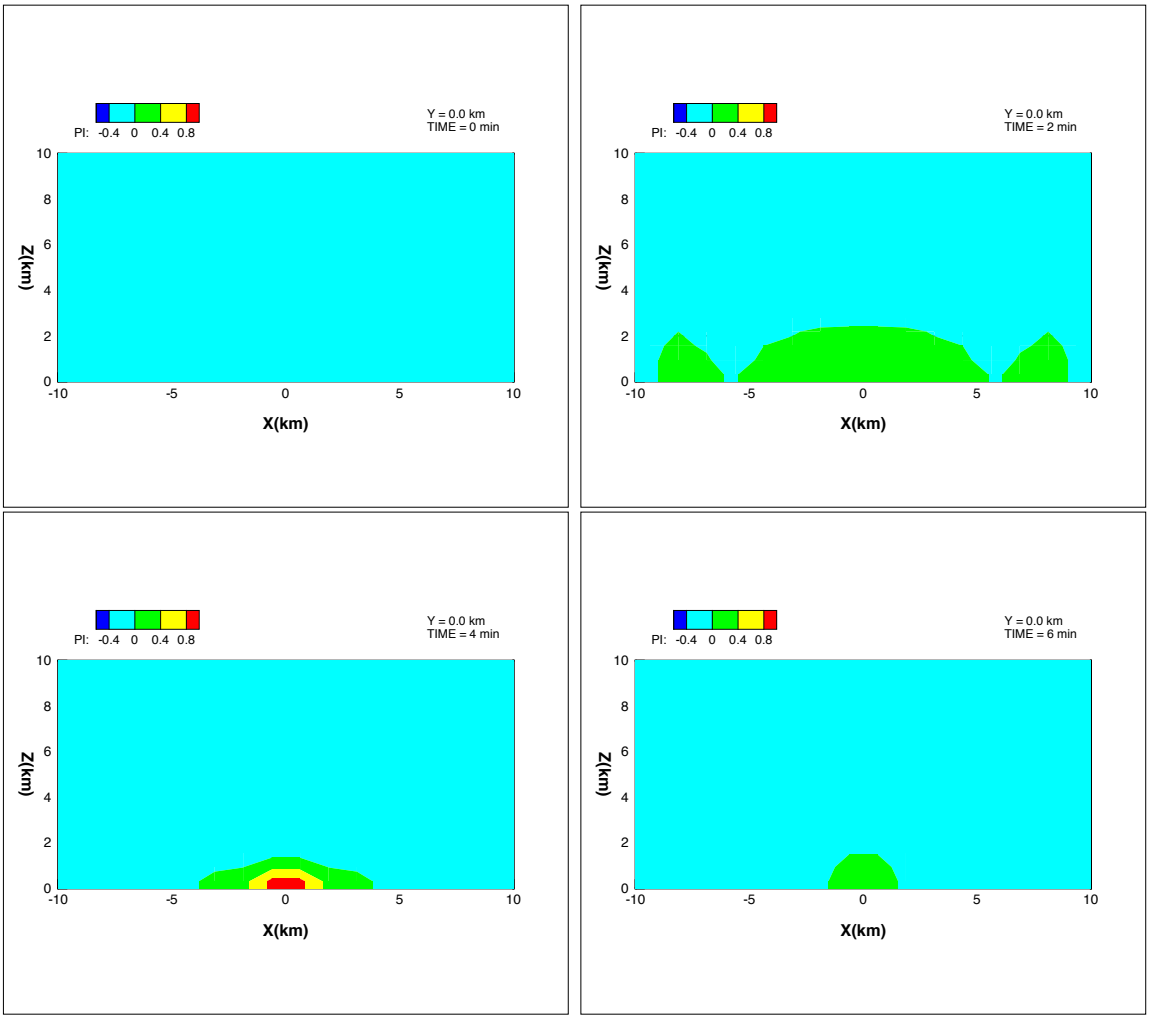

Figure 5: Evolution of the pressure deviation field after the downburst initiation in the $x - z$ cross section at $y = 0$.

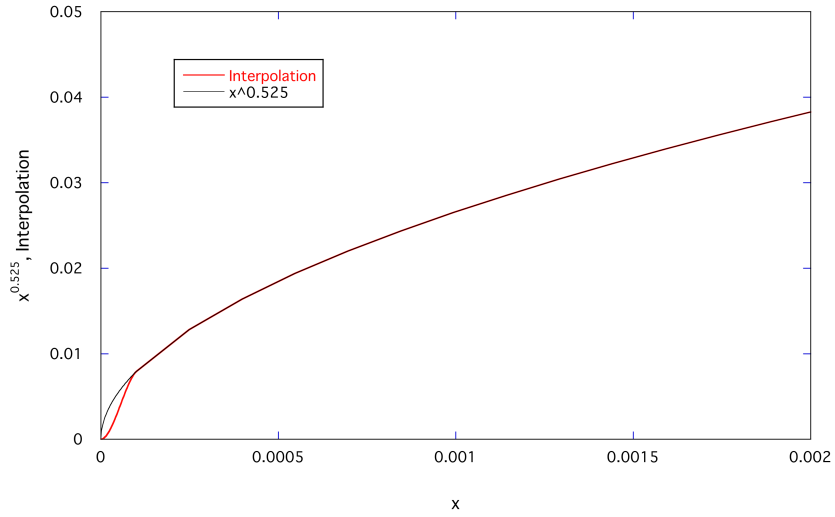

Figure 6: The function $x^{0.525}$ and Hermite interpolation

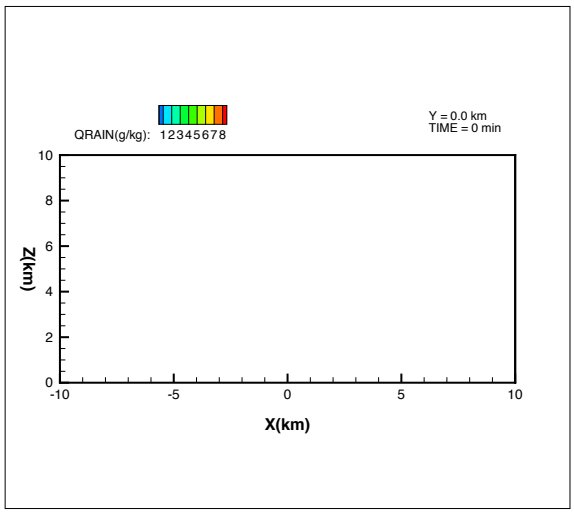

Figure 7: The rainwater field in the $x - z$ cross section at $y = 0$ of the starting point used for the optimization.

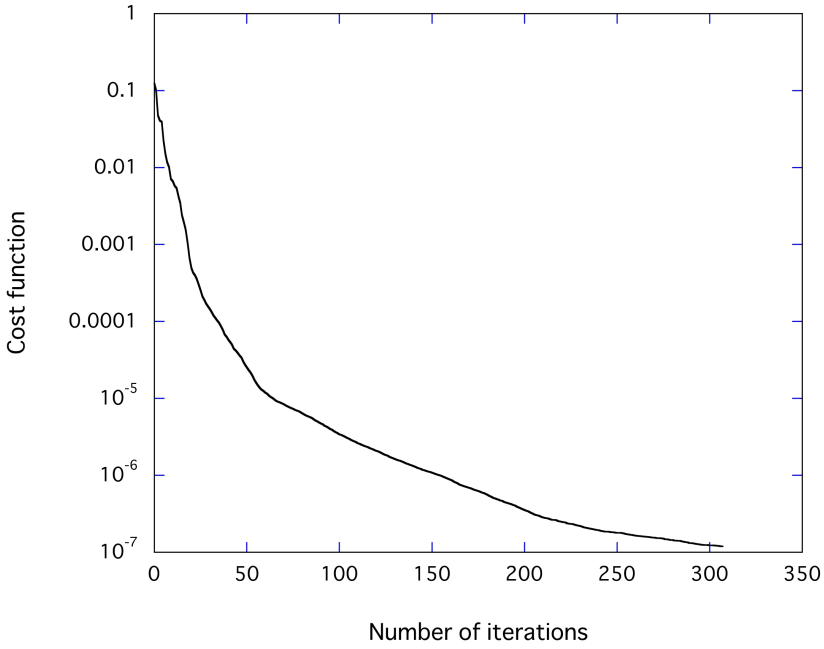

Figure 8: Convergence history of the optimization

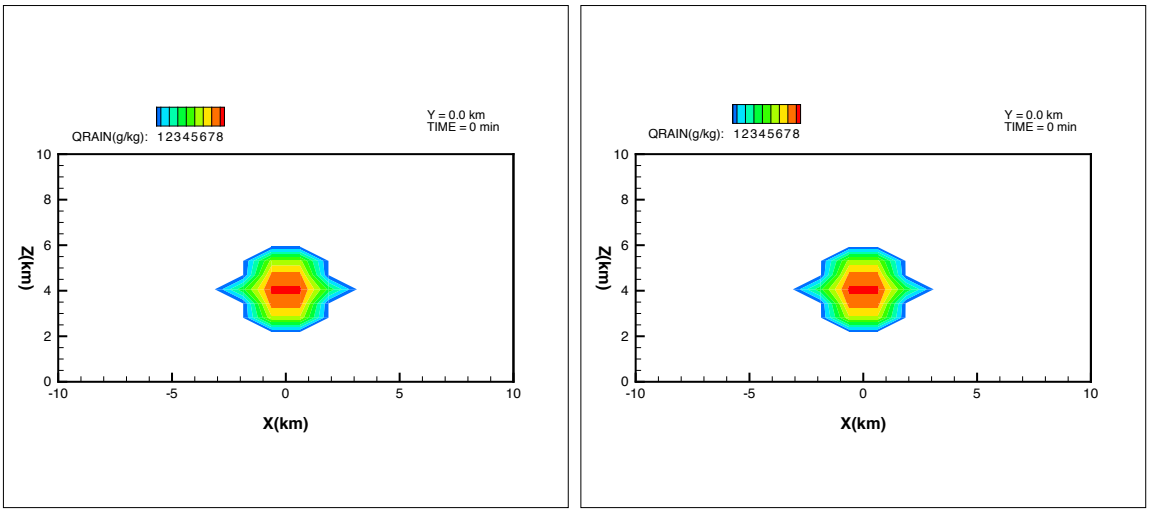

Figure 9: The rainwater field recovered by the optimization (left) and the corresponding one of the initial conditions used in the simulation (right) in the $x - z$ cross section at $y = 0$.

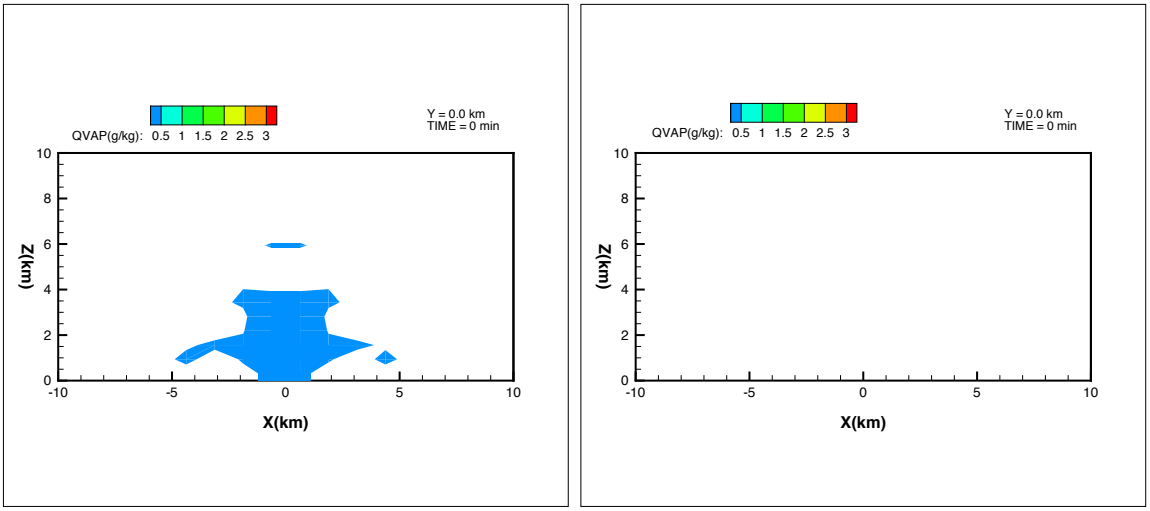

Figure 10: The water vapor field recovered by the optimization (left) and the corresponding one of the initial conditions used in the simulation (right) in the $x - z$ cross section at $y = 0$.

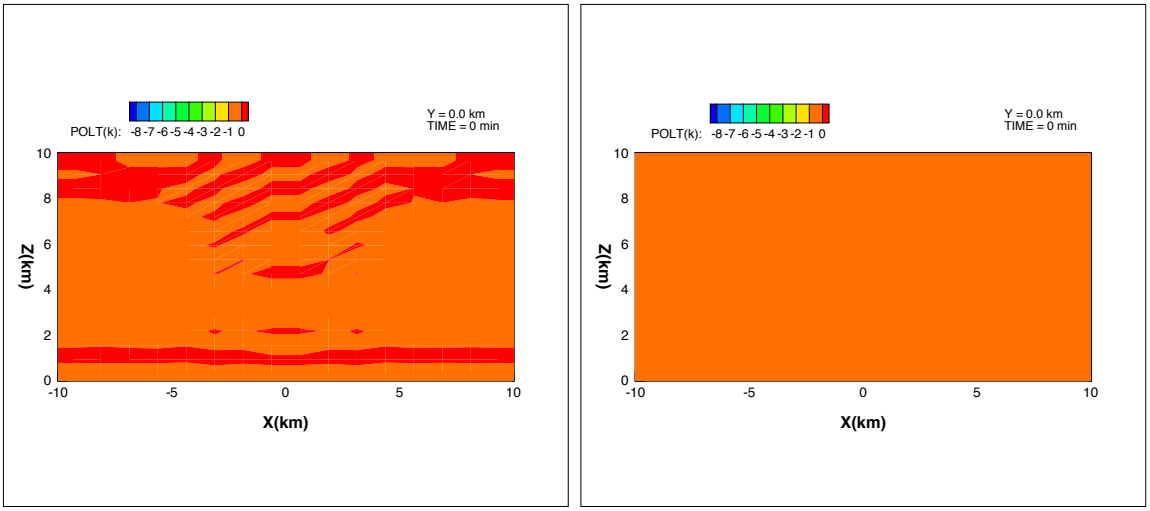

Figure 11: The potential temperature field recovered by the optimization (left) and the corresponding one of the initial conditions used in the simulation (right) in the $x - z$ cross section at $y = 0$.

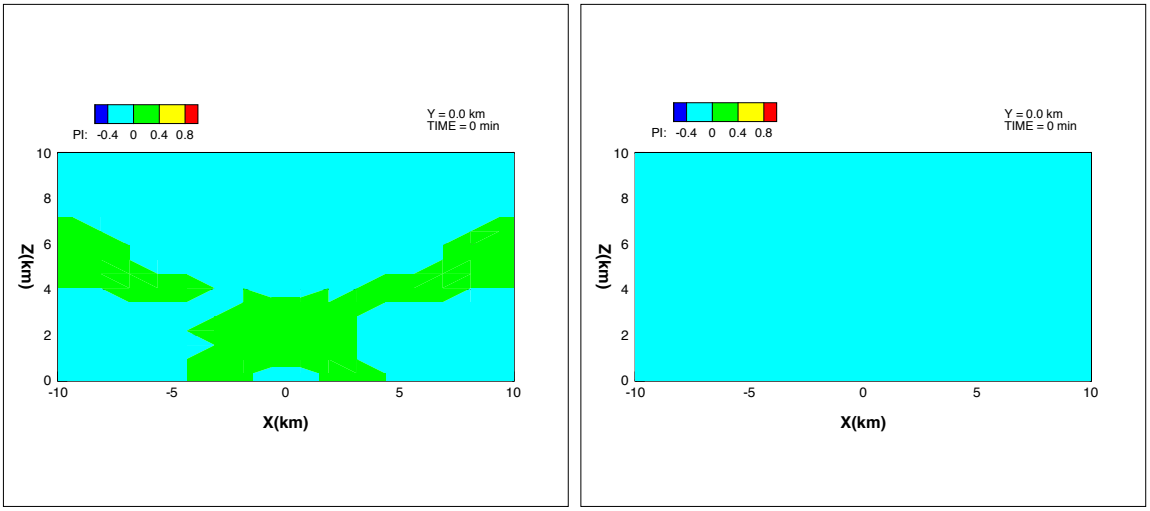

Figure 12: The pressure deviation field recovered by the optimization (left) and the corresponding one of the initial conditions used in the simulation (right) in the $x - z$ cross section at $y = 0$.

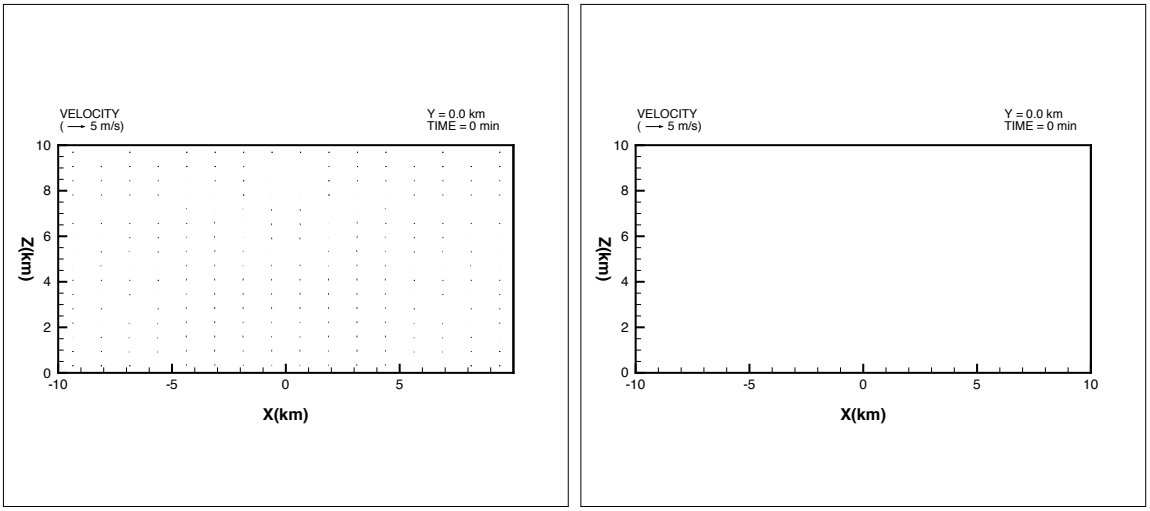

Figure 13: The flow field recovered by the optimization (left) and the corresponding one of the initial conditions used in the simulation (right) in the $x - z$ cross section at $y = 0$.