Noname manuscript No.
(will be inserted by the editor)

Inverse problem to determine key turbulent transport parameters in fusion plasmas

Louis Lamérand · Didier Auroux · Francesca Rapetti · Eric Serre

Received: date / Accepted: date

Abstract Two-dimensional codes rely on models in which plasma turbulence has been smoothed out by averaging, and are thus considered as reduced models with respect to 3D ones. One of the main issue nowadays in such reduced models is the accurate modelling of transverse transport fluxes resulting from the averaging of stresses due to fluctuations. Such models assume transverse fluxes are driven by local gradients, and characterized by ad-hoc diffusion coefficients (turbulent eddy viscosity) whose values are adjusted by hand in order to match numerical solutions with experimental measurements. These coefficients differ from one machine to another, from one pulse to another in the same device and even from one location to another in a given discharge. They must then be considered as free parameters with an extremely large number of degrees of freedom, which reduces drastically the predictive capabilities of these codes. To solve the computational cost issue, we complete the electromagnetic equations with equations describing the time and/or space evolution of the transport coefficients, thus making the call to a turbulence code unnecessary. The consequence is the substitution, as free parameters, of perpendicular diffusion coefficients with parameters defining key properties of the underlying transport mechanisms (e.g., turbulence growth and damping rates). Although this might not lead to a reduction of the number of free parameters, the new parameters are expected to be more driven by the underlying transport physics and hence will vary much less from one machine to the other or from one location to another in the same plasma. In this paper, as a proof of concept, we explore, on the basis of digital twin experiments, the efficiency of the assimilation of data to fix free parameters of the transverse turbulent transport models in the set of averaged equations in 2D.

L. Lamérand, D. Auroux and F. Rapetti
Université Côte d'Azur, Inria, CNRS, LJAD, Nice, France
E-mail: louis.lamerand@univ-cotedazur.fr (corresponding author)
didier.auroux@univ-cotedazur.fr, francesca.rapetti@univ-cotedazur.fr

E. Serre
Université Aix-Marseille II, CNRS, M2P2, Marseille, France
E-mail: eric.serre@univ-amu.fr

Keywords Parameter identification · data assimilation · plasma physics · direct and adjoint problems · minimisation

Mathematics Subject Classification (2010) 65N30

1 Introduction

Magnetic fusion is a promising way to produce carbon free energy in large quantities. It is based on the fusion of two light isotopes of hydrogen into a heavier one in a hot plasma confined by a magnetic field in a toroidal machine called tokamak, Figure 1. The International Tokamak Experimental Reactor (ITER) is to date the most ambitious of these devices under construction by its size and its ultimate goal of achieving a ratio of energy produced to energy consumed equal to 10, [9]. However, many physical and technological issues remain that require intensive numerical simulations to complement the sparse experimental measurements and incomplete theoretical models.

Thus, in view of ITER operation and the design of optimized plasma scenarii, reduced models implemented in fluid transport codes (as in [13] or [4]) are well adapted to provide relevant information on appropriate return times, similarly to the Reynolds averaged Navier–Stokes (RANS) codes commonly used for engineering applications with neutral fluids [12]. Despite the increase in computing power and the performance of numerical methods, they are currently the only models capable of performing simulations in the ranges of parameters and geometries relevant to tokamaks.

In these fluid reduced models, turbulence has been smoothed out by averaging. The transport fluxes (transverse to the magnetic field lines, Figure 1) resulting from the averaging of stresses due to fluctuations and assumed to be driven by local gradients are characterized by ad-hoc diffusion coefficients (turbulent eddy viscosity) whose values must be determined. In a recent study, Baschetti et al. [2] has proposed an advanced modelling that significantly improves the predictive capability of the model. In this model, the transverse diffusion coefficients are determined from the turbulence kinetic energy $\kappa \equiv \frac{1}{2}\langle v'^2_\perp \rangle$ and its dissipation rate

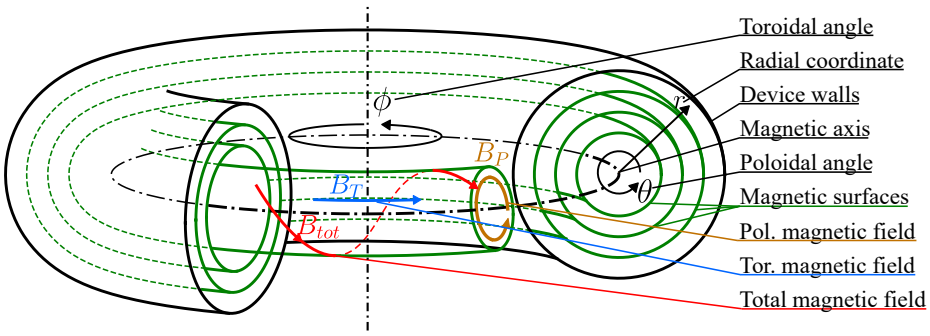


Fig. 1: Sketch of a simple magnetic geometry in a Tokamak with coordinates of the torus. Particles are (imperfectly) bound to helicoidal trajectories which follow the total magnetic field lines.

ε such that $D_\perp \propto \kappa^2/\epsilon$. κ and ε are determined self-consistently by 2 additional transport equations designed on the knowledge of the physical mechanisms at play at the plasma edge in tokamaks. Inherent to the model reduction, free parameters remain in these equations and must be calibrated.

For that purpose, and to improve the reliability and predictability of such reduced models, we explore in the present work data assimilation with the ultimate objective to use data bases provided by experimental measurements or higher fidelity numerical models. While data assimilation is traditionally used in oceanography or meteorology [3,10] to estimate a global current state with sparse measurements of different accuracy, the mathematical methods that are used have the potential to increase the accuracy of any numerical model with loosely defined parameters, assuming data from the modelled system is available. In the present work, we chose to exploit the Variational Data Assimilation (VDA) framework [5] which involves the minimization of a cost function defined as the distance between the data of reference and the values computed by the model [5]. The gradient of the cost function can then be obtained by automatic differentiation, allowing the use of efficient minimisation algorithms (e.g. conjugate gradient, quasi-Newton, ...) [11]. However, as a complex nonlinear optimization problem, the convergence of the calibration procedure at a reasonable rate is not necessarily guaranteed. Fortunately, the formulation as an optimization problem gives access to different strategies typical of this field of mathematics, like the re-scaling of the optimization variables or the use of penalization function, which can greatly improve the performances of the algorithm. Following [1], a recently published proof of concept of the application of the VDA strategy to improve the precision of a $\kappa - \epsilon$ type model for turbulence, this article aims at detailing the choice of the methodologies used to increase the efficiency of the calibration procedure as well as at testing its robustness. The paper is organized as follows. In section 2 we briefly define the $\kappa - \epsilon$ model averaged to 1D in the radial direction and decoupled to the plasma transport equations . In section 3 we give details on the calibration problem as well as on the algorithm to solve it, including the definition of the cost, the computation of its gradient and the different strategies to improve the whole procedure. In section 4 we introduce the configuration of reference for the calibration tests as well as an experimental illustration of different methods to improve the algorithm. Finally, in section 5, the robustness of the calibration algorithm is tested by introducing noise and sparsity in the data.

Following [1], a recently published proof of concept of the application of the VDA strategy to improve the precision of a $\kappa - \epsilon$ type model for turbulence, this article aims at detailing the choice of the methodologies used to increase the efficiency of the calibration procedure as well as at testing its robustness. In section 2 we briefly define the $\kappa - \epsilon$ model averaged to 1-D in the radial direction and decoupled to the plasma transport equations, [2]. Section 3 details the calibration problem as well as the algorithm to solve it, including the definition of the cost, the computation of its gradient and the different strategies to improve the whole procedure. Section 4 introduces the configuration of reference for the calibration tests as well as an experimental illustration of different methods to improve the algorithm. Finally, in section 5, the robustness of the calibration algorithm is tested by introducing noise and sparsity in the data.

2 Reduced model

We consider here the κ-ε model reduced to 1D in the radial direction following the procedure defined in [2] (Section 3). The model is here decoupled from fluid plasma equations and thus plasma flow variables are considered constant in time. As defined in the Introduction, $\kappa \equiv \frac{1}{2}\langle v'^2_\perp \rangle$ and ε relates a damping process acting on κ to determine the turbulent diffusion with $D_\perp = C\kappa^2/\epsilon$, where C is constant.

2.1 $\kappa - \varepsilon$ model

The whole original model equations are detailed in [2]. The evolution of κ and ε is governed by local dynamics and transverse and parallel transport to the magnetic field lines, Figure 1. Both fields are understood to be proportional to energies and thus defined to be positive. The radial coordinate r varies between 0 and a, the little radius of the torus, corresponding to the plasma center and the tokamak wall, respectively.

We normalize κ and ε by typical scales, so that $Z = \kappa/\kappa_0$ and $Y = \varepsilon/\varepsilon_0$. We also normalize the radial position by the plasma minor radius a, so that $\rho = r/a$, $\rho \in [0, 1]$.

The normalized system now reads:

$$\partial_t Z = \frac{1}{\rho}\nabla_\rho\Big(\rho D_{gBZ}\frac{Z^2}{Y}\nabla_\rho Z\Big) + \gamma_Z Z - KZ^2 - Y, \tag{1a}$$

$$\partial_t Y = \frac{1}{\rho}\nabla_\rho\Big(\rho D_{gBY}\frac{Z^2}{Y}\nabla_\rho Y\Big) + \gamma_Y Y - \gamma_Z\frac{Y^2}{Z^{3/2}}, \tag{1b}$$

with initial and boundary conditions such that:

$$\begin{aligned} &Z(t=0,\rho) = Z_0,\ \nabla_\rho Z(t,\rho=0) = 0,\ Z(t,\rho=1) = 0,\\ &Y(t=0,\rho) = Y_0,\ \nabla_\rho Y(t,\rho=0) = 0,\ Y(t,\rho=1) = 0. \end{aligned} \tag{1c}$$

The general structure of the equations above allow to split the model in two parts: a local prey-predatory model, which can either tend to a limit cycle or converge similarly to an underdamped oscillator, and nonlinear diffusion terms which couple the variables at different radii, modelling the way the turbulence diffuses itself. The parameter $K \ll 1$ in (1a) is akin to a Kubo number as it quantifies the condition of low turbulence regime [2]. γ_Z and γ_Y denote normalized growth rates. The diffusion weights D_{gBZ} and D_{gBY} will be identified independently but their target values will be equal in the configurations considered, so we usually refer to their common value as D_{gB}.

2.2 Spatial discretisation

The dimensionless radial coordinate ρ is discretized in $N_\rho + 1$ values, $(\rho_j)_{j\in[\,0,N_\rho\,]}$ such that $\rho_0 = 0$ and $\rho_{j+1} = \rho_j + \Delta_r$, with $\Delta_r = 1/N_\rho$ so that $\rho_j \in [\,0, 1\,]$. The toroidal geometry is approximated by a cylinder of length $2\pi A$, with $A = R/a$ the aspect ratio of the torus, and the toroidal angle ϕ is replaced by the coordinate

x_ϕ (slab geometry). Our cylindrical space Ω is divided into cells $(\omega_j)_{j\in[\,1,N_\rho\,]}$ corresponding to hollow cylinders with outer and inner radii respectively equal to ρ_j and ρ_{j-1}. We can easily compute their volumes V_j, inner surfaces S_j^- and outer surfaces S_j^+ :

$$S_j^+ = 2\pi\rho_j \times 2\pi A\,, \tag{2a}$$
$$S_j^- = 2\pi\rho_{j-1} \times 2\pi A\,, \tag{2b}$$
$$V_j = 2\pi(\rho_j^2 - \rho_{j-1}^2) \times 2\pi A\,, \tag{2c}$$

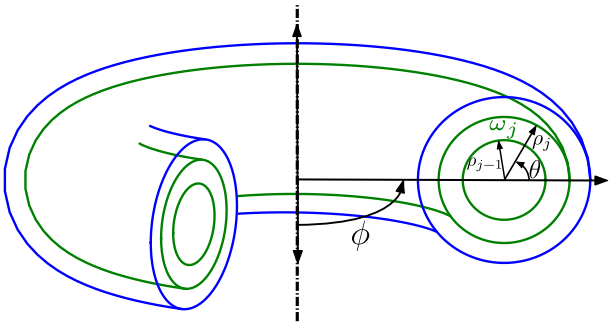


Fig. 2: Representation of the cells in the original toroidal geometry. For simplicity, at the discrete level, it is approximated by a cylindrical geometry (slab), with cells corresponding to embedded hollow cylinders with outer and inner radii respectively equal to ρ_j and ρ_{j-1}.

where X_j is the value of any variable or parameter X at the middle radius $(\rho_j + \rho_{j-1})/2$ of the cell ω_j. Let's now integrate system (1) on the volume of each cell. From (1a), at any $j \in [\,1, N_t\,]$, we get :

$$\int_{\omega_j} \partial_t Z = \int_0^{2\pi A}\int_0^{2\pi}\int_{\rho_{j-1}}^{\rho_j} \frac{1}{\rho}\nabla_\rho\Big(\rho D_{gBZ} D_{Z,Y} \nabla_\rho Z\Big) d\rho\,\rho\,d\theta\,dx_\phi + \int_{\omega_j} (\gamma_Z Z - KZ^2 - Y)\,, \tag{3a}$$

with $D_{Z,Y} = Z^2/Y$. For the local terms, we simply approximate the variables and parameters by their values at the middle point multiplied by the volume of the cell. The diffusion term is integrated with regard to the radius :

$$V_j \partial_t Z_j = D_{gBZ}\int_0^{2\pi A}\int_0^{2\pi}\Big[\rho D_{Z,Y}\nabla_\rho Z\Big]_{\rho_{j-1}}^{\rho_j} d\theta\,dx_\phi + \gamma_{Z,j}\left(Z_j - K_j Z_j^2 - Y_j\right) V_j. \tag{3b}$$

Then since the variables and parameters are already averaged in the toroidal and poloidal directions, we get :

$$\partial_t Z_j = \frac{D_{gBZ}}{V_j}\Big(D_{Z,Y}(\rho_j)(\nabla_\rho Z)(\rho_j)S_j^+ - D_{Z,Y}(\rho_{j-1})(\nabla_\rho Z)(\rho_{j-1})S_j^-\Big) + \gamma_{Z,j}Z_j - K_j Z_j^2 - Y_j. \quad (3c)$$

From there, we simply approximate $D_{Z,Y}(\rho_j)$ by $D_j^+ := (D_{Z,Y;j+1} + D_{Z,Y;j})/2$ and $(\nabla_\rho Z)(\rho_j)$ by $D_j^- := (Z_{j+1} - Z_j)/\Delta_\rho$. We finally get :

$$\partial_t Z_j = \frac{D_{gBZ}}{V_j}\Big(D_j^+ S_j^+ \frac{Z_{j+1} - Z_j}{\Delta_\rho} - D_j^- S_j^- \frac{Z_j - Z_{j-1}}{\Delta_\rho}\Big) + \gamma_{Z,j}Z_j - K_j Z_j^2 - Y_j. \quad (4)$$

Taking into account boundary conditions, we get $(\nabla_\rho Z)(\rho_0) = 0$, and $(\nabla_\rho Z)(\rho_{N_\rho})$ is approximated by $-Z_{N_\rho}$. Furthermore, we consider that $D_{Z,Y} = Z^2/Y$ also has a limit of zero at the SOL boundary, so that $D^+_{N_\rho} = D_{Z,Y;N_\rho}/2$:

$$\partial_t Z_1 = \frac{D_{gBZ}}{V_1} D_1^+ S_1^+ \frac{Z_2 - Z_1}{\Delta_\rho} + \gamma_{Z,j}Z_1 - K_1 Z_1^2 - Y_1, \quad (5a)$$

$$\partial_t Z_{N_\rho} = \frac{D_{gBZ}}{V_{N_\rho}}\Big(- D^+_{N_\rho} S^+_{N_\rho} \frac{Z_{N_\rho}}{\Delta_\rho} - D^-_{N_\rho} S^-_{N_\rho} \frac{Z_{N_\rho} - Z_{N_\rho - 1}}{\Delta_\rho}\Big) + \gamma_{Z,N_\rho} Z_{N_\rho} - K_{N_\rho} Z^2_{N_\rho} - Y_{N_\rho}. \quad (5b)$$

With equations (4), (5a) and (5b) we can define a linear operator $\mathcal{D}(D) : \mathbb{R}^{N_\rho} \to \mathbb{R}^{N_\rho}$ which depends on the viscosity vector D, and is applied to $X = Z$ or Y to obtain the corresponding diffusion term :

$$\forall j \in [\,1, N_\rho - 1\,], \quad D_j^+ = D^-_{j+1} = (D_{j+1} + D_j)/2\,, \quad (6a)$$

$\forall j \in [\,2, N_\rho - 1\,]$,

$$(\mathcal{D}(D)X)_j = \frac{1}{V_j}\Big(D_j^+ S_j^+ \frac{X_{j+1} - X_j}{\Delta_\rho} - D_j^- S_j^- \frac{X_j - X_{j-1}}{\Delta_\rho}\Big), \quad (6b)$$

$$(\mathcal{D}(D)X)_1 = \frac{1}{V_1} D_1^+ S_1^+ \frac{X_2 - X_1}{\Delta_\rho}, \quad (6c)$$

$$(\mathcal{D}(D)X)_{N_\rho} = \frac{1}{V_{N_\rho}}\Big(- \frac{D_{N_\rho}}{2} S^+_{N_\rho} \frac{Y_{N_\rho}}{\Delta_\rho} - D^-_{N_\rho} S^-_{N_\rho} \frac{X_{N_\rho} - X_{N_\rho - 1}}{\Delta_\rho}\Big)\,. \quad (6d)$$

Hence after a similar reasoning for Y, we simply write the discrete system as :

$$\partial_t Z = D_{gBZ}\mathcal{D}\left(\frac{Z^2}{Y}\right) Z + \gamma_Z Z - K Z^2 - Y, \quad (7a)$$

$$\partial_t Y = D_{gBY}\mathcal{D}\left(\frac{Z^2}{Y}\right) Y + \gamma_Y Y - \gamma_Z \frac{Y^2}{Z^{3/2}}\,. \quad (7b)$$

2.3 Time discretization

A partially implicit time discretization is used in the parameter fitting procedure to compute the trajectory of the normalised κ and ε. The derivative is approximated using both the values of the variables at steps n and $n+1$, but in a way that gives a linear system (from now on the indices are linked to the time step) :

$$Z_{n+1} - Z_n = \Delta_t \left(D_{gBZ} \mathcal{D}\left(\frac{Z_n^2}{Y_n}\right) Z_{n+1} + \gamma_Z Z_n - K Z_n^2 - Y_n \right), \tag{8a}$$

$$Y_{n+1} - Y_n = \Delta_t \left(D_{gBZ} \mathcal{D}\left(\frac{Z_n^2}{Y_n}\right) Y_{n+1} + \left(\gamma_Y - \gamma_Z \frac{Y_n}{Z_n^{3/2}}\right) Y_{n+1} \right). \tag{8b}$$

Indeed $\mathcal{D}\left(Z_n^2/Y_n\right)$ is a linear operator depending on terms of the step n, and Y_{n+1} at the end of the right hand side of (8b) is multiplied by a term depending only on the step n as well. This simple strategy only adds the computational cost necessary for the resolution of a tridiagonal linear system, but largely improve the stability of the scheme. On the one hand, the implicit resolution of the anisotropic Laplacian avoids to have CFL type conditions on the value of the time step Δ_t. On the other hand, by inverting the local part of (8b), we get:

$$Y_{n+1} - \Delta_t \left(\gamma_Y - \gamma_Z \frac{Y_n}{Z_n^{3/2}}\right) Y_{n+1} = Y_n. \tag{9a}$$

And so,

$$Y_{n+1} = \frac{Y_n}{1 - \Delta_t \left(\gamma_Y - \gamma_Z \frac{Y_n}{Z_n^{3/2}}\right)}. \tag{9b}$$

If Y_{n+1} is given by (9b), it will remain positive as long as $\Delta_t < 1/\gamma_Y$, which is manageable because it depends directly on the parameters.

Obviously the scheme is still not stable for any set of parameters. This is actually a good test because solvers are rarely expected to give exploitable results in every regime, so the minimisation algorithm must be able to cope with regions -not precisely identified *a priori*- of forbidden sets of parameters.

The modification of the solver algorithm to compute the cost is straightforward : when the model reaches a time t_i corresponding to recorded data vectors Z_i^{obj} and Y_i^{obj}, the squared difference between the logarithm of the current value of Z and Y and of the data Z_i^{obj} and Y_i^{obj}, the cost function is added. The solver of the model is derived by the automatic differentiation tool `Tapenade` [8], mostly as a proof of concept. For this simple solver the derivative could have been written directly but an automatic differentiation tool will be quite useful when extending to more complete models with substantially more complex solvers.

Table 1 displays the default values of the discretization parameters. The data is generated with the same time step as the one used by the model solver in the parameter fitting algorithm ($\Delta_t = 5 \times 10^{-3}$) to ensure that we effectively consider twin experiments. However we don't need data at every time step, and so we can have multiple solver iterations between two records. By default the solution is recorded every 10 iterations, giving a data time step ($\Delta_t^{obj} = 5 \times 10^{-2}$). The influence of data time step will be briefly studied in Section 5.

Furthermore, an important factor for the efficiency of the algorithm is the length of the time interval in the cost function. Since the model is often locally

Radial grid spacing Δ_ρ	Time step Δ_t	Time elapsed between two recordings Δ_t^{obj}	Data time interval length L_T
1/150	5×10^{-3}	5×10^{-2}	40

Table 1: Default numerical values for the discretisation data in the parameter fitting procedure

oscillatory, the objective data and the currently computed trajectory may quickly run out of phase even if the parameters are relatively close from their target values. Hence it is interesting to reduce the length of the time interval to at most a few oscillations, and we use by default a length of 40, approximately equal to four oscillation periods for the considered configuration (see Figure 5).

3 Model calibration

The fitting procedure is only tested on twin experiments, fairly virtual cases where the target data are generated by the model itself, for some parameters that we try to find back. Although avoiding the problem of uncertainty and unavailability of the observations, it is already a sufficient test to give insights on the good behaviour of the algorithm and its ability to detect more or less complex sets of parameter values.

3.1 Definition of the problem

The target data are constituted of two sets of vectors (Z_i^{obj}) and (Y_i^{obj}) (the exponent *obj* stands for objective) containing approximations of Z and Y at different radial positions for a given set of times $(t_i)_{i\in[\,0,N_T\,]}$ separated by the data time step Δ_t^{obj}. The data time step Δ_t^{obj} is a multiple of the time step (here 10 times bigger, as stated in Table 1) used in the numerical solver to generate the data themselves. Since the data are generated here, they are already complete, in the right format, and we do not have to take into account possible differences of confidence on the different observations. With real data, these problems will be mostly treated in the definition of the cost function, including operators to get the generated trajectory in the format of the data and adding a covariance matrix in the definition of the cost to account for observation errors. Here we can simply use an Euclidean scalar product of the difference between the generated trajectory and the data. For a given set of parameters p, knowing the target set of vectors (Z_i^{obj}) and (Y_i^{obj}) corresponding to times $(t_i)_{i\in[\,0,N_T\,]}$ the cost functional simply reads:

$$j(p) = \sum_{i\in[\,0,N_T\,]} \frac{1}{2}\left(\|(Z(p))(t_i) - Z_i^{obj}\|_2^2 + \|(Y(p))(t_i) - Y_i^{obj}\|_2^2\right). \tag{10}$$

Furthermore, as we will see in Section 4.4, considering an error relative to the value of the parameters can be useful to maximise the use of the information in the data. Indeed the variable can have important scale variations and an error at

a small scale can have a huge impact after the exponential growth. Hence, we will alternatively use a cost based on the difference between the logarithms :

$$j_l(p) = \sum_{i\in[\,0,N_T\,]} \frac{1}{2}\Big(\|\log((Z(p))(t_i)) - \log(Z_i^{obj})\|_2^2 + \|\log((Y(p))(t_i)) - \log(Y_i^{obj})\|_2^2\Big). \quad (11)$$

Concerning the parameters to identify, they are 5 in the model given in (1), namely D_{gBZ}, D_{gBY}, γ_Z, γ_Y and Z. We add to these parameters the initial states $Z(t_0)$ and $Y(t_0)$, simply written Z_0 and Y_0. In our case, the initial states are directly given by the first vectors in the data sets, Z_0^{obj} and Y_0^{obj}. However, even if the target initial conditions are given in the data, it is not obvious that the algorithm will easily converge towards them : at least in an intermediate state of the optimisation algorithm, it may be more advantageous to have the initial state different from the one given by the data if it allows for a reduction of the distance to the data as a whole. For real physical problems the data will most likely not be available at any point in space, and not directly for the turbulent variables κ and ε, so it is important to show that the initial condition can be identified.

All parameters are considered constant in time -as said above- and dependent on the radius, except D_{gBZ} and D_{gBY} which are simple scalars. Considering that the purpose of the model is to evaluate the viscosity coefficient, correcting it with an ad-hoc parameter at every space point would clearly reduce its interest. Hence we authorize ourselves the simplification of considering D_{gBZ} and D_{gBY} as scalars.

Summing up, we have 7 different parameters (including the initial conditions), among which 5 vary with the radius, and all of them are used as unknowns of the inverse problem :

- D_{gBZ} and $D_{gBY} \in \mathbb{R}^+$ the coefficients of the (nonlinear) diffusion term, necessarily positive for a physically coherent diffusion effect.
- γ_Z and $\gamma_Y \in L^\infty([0,1] \to \mathbb{R}^+)$ the normalised effective growth rates. They may vary consequently in scale and could theoretically change of sign, but this case is not considered here, even if the PDE solver can marginally converge for negative values.
- $K \in L^\infty([0,1] \to \mathbb{R}^+)$ the Kubo parameter (also referred to the Strouhal number) weighting the nonlinear saturation term in the equation (1) for Z. The value of K is usually fairly negligible between 10^{-1} and 10^{-4}, but it is supposed to remain positive to act as a saturation term.
- Z_0 and $Y_0 \in L^\infty([0,1] \to \mathbb{R}^+)$ the initial states of the model. They are supposed positive due to the physical definition of Z and Y -and the mathematical operations performed on them make this condition necessary.

Despite the boundaries defined here, most of the parameters are allowed to take negative values at intermediate states of the minimisation algorithm, because it improves the convergence of the chosen minimizer. However their final retrieved values should indeed be positive.

3.2 Minimizer

The complete principle of the algorithm is illustrated in Figure 3.

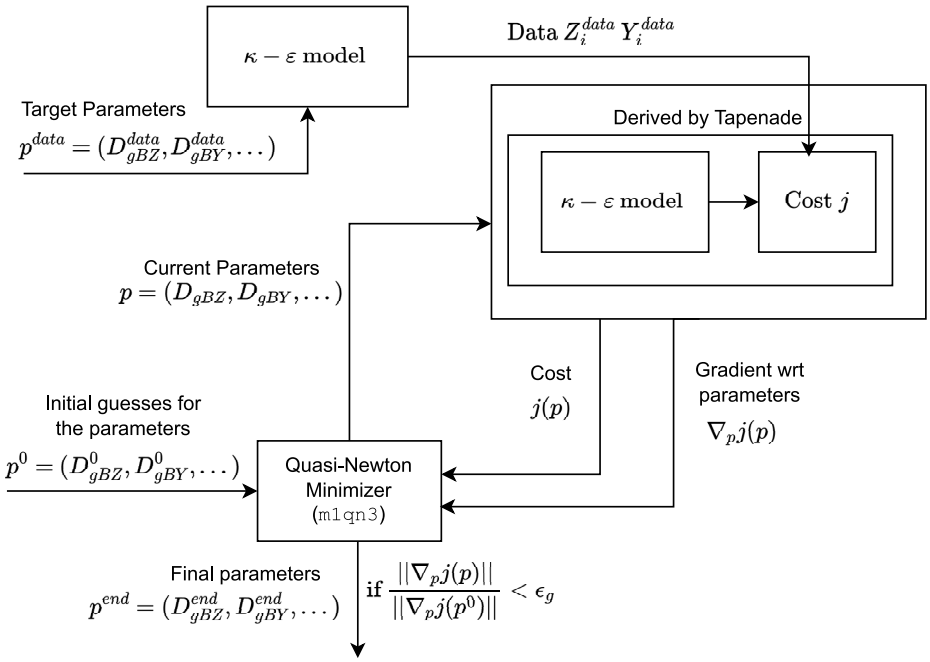


Fig. 3: Block scheme of the minimisation algorithm.

First the $\kappa - \epsilon$ model is solved for a given set of data parameters and the values of Z and Y are recorded for different values of the time variable separated by Δ_t^{obj}. Then we have a loop between a nonlinear minimisation routine, the Fortran routine `m1qn3` [7], and the derived version of the whole procedure returning the cost, *i.e.*, the 1D $\kappa - \epsilon$ model solver slightly modified to compute the difference with the target data. Since the gradient is necessarily 0 at the minimum, the loop stops when the 2-norm of the gradient of the cost function has been enough reduced with regards to its initial value, namely, $\|\nabla j\|/\|\nabla j_0\| < \epsilon_g = 10^{-7}$.

The `m1qn3` routine implements a limited memory BFGS algorithm for unconstrained nonlinear optimisation (see [6]). It is a very efficient numerical minimiser, which converges quickly in terms of iterations, with a negligible cost in terms of computational time and memory space, compared to the evaluation of the cost and its gradient. It relies on the evaluation of an approximated value for the Hessian using the gradient and estimates from a given number M of past iterations. A larger M would generally improve the precision of the Hessian approximation -hence reducing the number of iterations before convergence- but also increasing its computational and memory cost of each iteration. For all calibration tests, we have set $M = 20$ since it has been reported that a higher value does not generally reduce significantly the number of iterations before convergence. Among the two possible modes of initialisation of the approximated Hessian, we choose the Diagonal Initial Scaling (DIS) which should further reduce the number of iterations. Each iteration of this minimiser may actually require multiple "simulations", *i.e.* evaluations of the cost function and of its gradient, because a line search along the computed descent direction is performed until some Wolfe conditions [14] are

satisfied. Most of the time the initial step-size of 1 is sufficient, and in an unconstrained optimisation context, the number of simulations is rarely more than 30% larger than the number of iterations [6].

During this line search, if we reach a set of parameters for which the cost can't be computed (typically because the direct solver does not converge), a special value is returned to `m1qn3` and the step-size is divided by ten. Although this is useful to avoid too big steps typically in the first iterations, it is not a very efficient strategy when the set of parameter is close to a region where the direct solver does not converge, as it simply slows the algorithm without really pushing it in a safer direction. This is where the scheme we have chosen becomes quite useful because its trajectory smoothly expands before diverging by reaching negative values. It will then allow to compute a considerable error near the region of non convergence and a gradient which will naturally push the parameters in safer territories, while not slowing down the algorithm.

Nevertheless it is not always sufficient, as the algorithm is sometimes unable to find a suitable step-size near the boundaries of the region of convergence of the numerical scheme. We want to avoid the use of a constrained minimizer because we expect to find a minimum for which the constraints are not active, and the constrained solver are generally largely heavier computationally. Instead we will introduce multiple regularising strategies to avoid the divergence of the algorithm as well as to improve its general convergence.

3.3 Scaling

Scaling functions are used to improve the conditioning of the problem and impose boundaries on some parameters. Each of the 7 parameters can have a different scaling function s, defined element-wisely for the non constant parameters. A given parameter x of the set p, corresponds with the rescaled parameter $\bar{x} \in \bar{p}$, such that $x = s_x(\bar{x})$. In this form, the scaled cost function is $\bar{j}(\bar{p}) = j((s_x(\bar{x}))_{\bar{x}\in\bar{p}})$ and its gradient with regards to $\bar{x}$ reads:

$$\nabla_{\bar{x}}\bar{j} = \frac{\partial s_x}{\partial \bar{x}} \nabla_x j(p)\,. \tag{12}$$

Hence the gradient with regards to the rescaled parameter is multiplied by the derivative of the scaling function s_x. Moreover, the range of values assumed by the rescaled parameter is different from the non scaled parameter, so the gradient may have a greater or lesser impact on the movement of the parameter. For a simple gradient descent, the value of the parameter x at the iteration $n+1$ is obtained from its value x_n at iteration n as (with α_n the step for this iteration) :

$$x_{n+1} = s_x\big(s_x^{-1}(x_n) - \alpha_n \frac{\partial s_x}{\partial \bar{x}} \nabla_x j(p)\big)\,. \tag{13}$$

Hence for a linear scaling, $s : x \mapsto (k\,x)$,

$$\begin{aligned} x_{n+1} &= k\big(k^{-1}x_n - \alpha_n k \nabla_x j(p)\big) \\ &= x_n - k^2 \alpha_n \nabla_x j(p)\,. \end{aligned} \tag{14}$$

Thus in this case the variation of the parameter x between iteration is enhanced by k^2. A sophisticated solver like `m1qn3` usually compensates itself for the bad

conditioning, but a rough *a priori* scaling for the parameters can significantly improve the convergence.

Moreover the scaling can be used to impose limits on a parameter : we just have to use a scaling function whose image is bounded, typically with horizontal asymptote. However, this adds nonlinearity and leads to a vanishing gradient as soon as the parameter get closer to the limit, what could finally slow down the convergence. Hence this strategy is better used sparsely. For a given rescaled parameter $\bar{x}$, we use for the scaling either one of the two following functions, which includes each time a linear coefficient k_x :

- the linear function $\bar{x} \mapsto k_x \bar{x}$
- an exponential function, $\bar{x} \mapsto e^{k_x \bar{x}}$. In this case the gradient is multiplied by $k_x e^{k_x \bar{x}} = k_x x$, so that the gradient is increased as the value of the parameter increases.

By default the scaling functions for each parameter are linear with coefficients $k_x = 1$, which is equivalent to no scaling at all.

3.4 Penalisation

Penalisation terms are added to the cost function to help keeping expected shapes and values for the parameters. They directly depend on the parameters and not on the result of the physical model. In this article we only use a penalisation on the radial derivative : for each parameter which depends on the radius, the quadratic norm of its derivative with respect to the radius multiplied by a scalar weight w_ρ is added to the cost function :

$$w_\rho \sum_{i=1}^{N_\rho - 1} \left(\frac{x_{i+1} - x_i}{\Delta_\rho} \right)^2 . \tag{15}$$

This penalisation is expected to inhibit the apparition of oscillations on the parameters, and improve the convergence rate at later stage of the calibration. Alternatively, for the initial states Z_0 and Y_0, we may also use a logarithmic version of the radial penalisation function :

$$w_\rho \sum_{i=1}^{N_\rho - 1} \left(\frac{x_{i+1} - x_i}{\Delta_\rho (x_{i+1} + x_i)/2} \right)^2 . \tag{16}$$

The weight of the radial derivative penalisation, w_ρ, is chosen identical for each parameter and equal to 0 by default.

Finally to maximise the efficiency of the regularising strategies it can be interesting to launch the algorithm twice, starting the second launch with the parameters obtained after the first, typically to change the weights of the different penalisation between the two launches. In this case we label the successive weights with exponents 1 or 2 whether they are used on the first or the second launch. By default all penalisation weights are set to zero. The next section will progressively introduce the different regularising strategies and illustrates experimentally their effects.

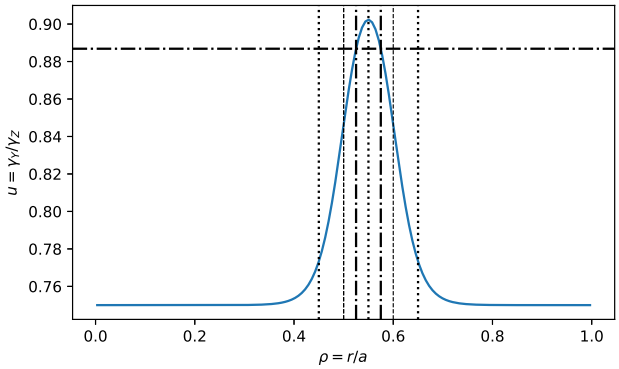


Fig. 4: Radial profile of $u = \gamma_Y/\gamma_Z$. The vertical dashed and dotted lines indicate respectively the middles and boundaries of the tanh shaped transition regions. The dash-dot lines are linked to the expected local behaviour : the horizontal one indicates the critical value $u_{crit} \approx 0.89$ of u between the stable ($u \geq u_{crit}$) and the unstable ($u < u_{crit}$) fixed point, while the vertical ones delimitate the stable region around $\rho = 0.55$.

4 Tuning of the algorithm parameters and numerical results

4.1 Reference configuration for the calibration algorithm

We introduce here the configuration on which the calibration algorithm will be tested for the reduced model (1). It consists of the choice of some specific radial shapes or constant values for each parameter varying with the radius. We will then test different values for the diffusion weights. This configuration will be used to generate the data, so that we can expect here the calibration algorithm to retrieve the corresponding shapes and values of the parameters.

In order to illustrate how the stability of the local system is modified by the diffusion, we use a constant $\gamma = \gamma_Z = 1$ and $K = 5 \times 10^{-2}$ and a fixed profile of $u = \frac{\gamma_Z}{\gamma_Y}$ enforcing limit cycle for the local solutions except in the vicinity of $\rho = 0.55$ where the local analysis predicts convergence. The variation of u is governed by standard tanh-shaped step functions $S(\rho, \rho_b, \delta\rho_b)$ and window function $\Pi(\rho, \rho_{b1}, \delta\rho_{b1}, \rho_{b2}, \delta\rho_{b2})$ defined, respectively, as:

$$S(\rho, \rho_b, \delta\rho_b) = 0.5\left(1 + \tanh\left(\frac{\rho - \rho_b}{\delta\rho_b}\right)\right), \tag{17a}$$

$$\Pi(\rho, \rho_{b1}, \delta\rho_{b1}, \rho_{b2}, \delta\rho_{b2}) = 0.5\left(S(\rho, \rho_{b1}, \delta\rho_{b1}) - S(\rho, \rho_{b2}, \delta\rho_{b2})\right). \tag{17b}$$

For the present simulation we have thus set:

$$u = 0.75 + (u_t - 0.75)\Pi(\rho, \rho_{b1}, \delta\rho_b, \rho_{b2}, \delta\rho_b). \tag{18a}$$

The width of the two transition regions is chosen identical for both step functions, namely:

$$u_t = 1.15 \;\; ; \;\; \rho_{b1} = 0.5 \;\; ; \;\; \rho_{b2} = 0.6 \;\; ; \;\; \delta\rho_b = 0.05. \tag{18b}$$

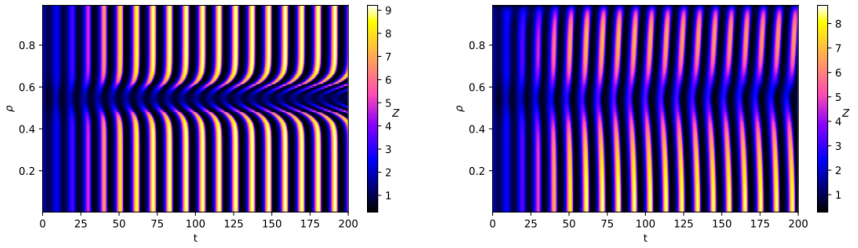


Fig. 5: Contour plot of Z versus time t and radial position ρ. Left : $D_{gB} = 10^{-8}$, Right : $D_{gB} = 10^{-4}$.

D^0_{gBZ}	D^0_{gBY}	γ^0_Z	γ^0_Y	K^0
10^{-5}	10^{-5}	1.2	0.7	0.02

Table 2: Initial guesses used in each test of the calibration procedure.

Since the distance $\rho_{b2} - \rho_{b1} = 2\delta\rho_b$ is small, the step function does reach its target value and u_t is adjusted to ensure that $u > 0.89$ in the window $0.528 \leq \rho \leq 0.572$, reaching $u \approx 0.9$ at $\rho = 0.55$ (see Figure 4). Finally we choose flat initial conditions $Z_0 = Y_0 = 2$.

To limit the number of free parameters, the diffusion weights are kept equal $D_{gBZ} = D_{gBY} = D_{gB}$. When they are increased, two effects become more and more evident (see Figure 5 with $D_{gB} = 10^{-4}$). First the absorbing condition at $\rho = 1$ obtained by enforcing $Z = 0$ and $Y = 0$ is now coupled with the other points in the radial profile governing a gradient in the values of Z and Y. Secondly, after a transient period, the local oscillations have the same period at each radius, but with a certain delay constant in time. Even the region characterized by stable fixed points exhibits relaxation oscillations due to the radial coupling induced by the diffusion.

In the next subsections, we will show how the parameter fitting procedure handles the different behaviours linked to the different values of D_{gB}. Regularising strategies will be introduced one after the other to improve either the robustness or the overall computational cost of the algorithm.

4.2 Choice of the scaling and of the time interval

We use, for all the tests of the calibration procedure, the same first guesses for each different parameter except for the initial states because they vary considerably depending on the configuration. The first guesses generate a trajectory with oscillations of large amplitude (above 50) so that the data is generally almost negligible in the first steps of the iteration, during which the amplitude of the oscillation is quickly reduced. Consequently the initial norm of the gradient does not change too much with the objective parameters, and the stopping condition based on the ratio of reduction of this norm is almost equivalent to an absolute condition

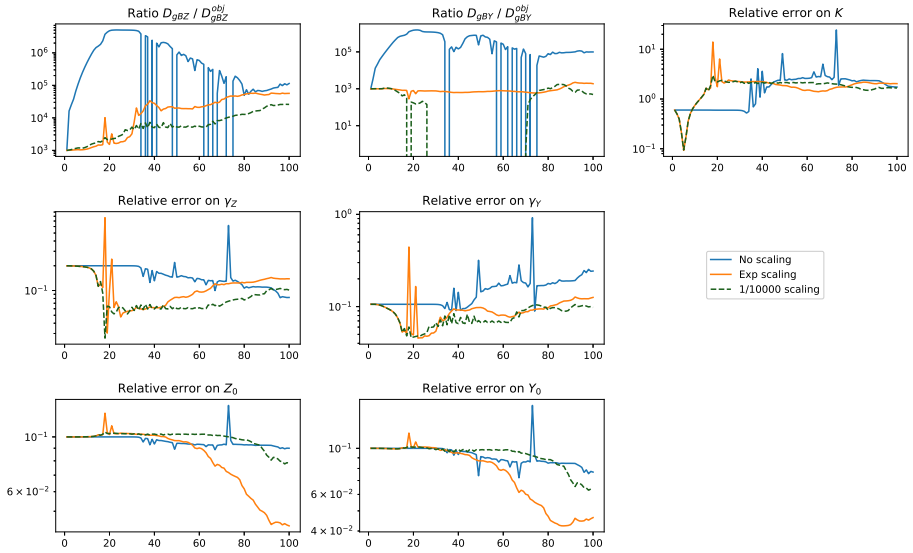


Fig. 6: Evolution of the errors on the parameters for the 100 first simulations of calibration runs with different types of scaling for D_{gBZ} and D_{gBY}. The error on the parameters are shown even for simulations which are not kept as valid iterations, usually when the initial step size lead to extreme values.

on the norm. It also has to be kept quite low to have a precise identification of the parameters, here by default at $\epsilon_g = 10^{-9}$.

4.2.1 Scaling

Running the basic version of the algorithm with $D^{obj}_{gB} = 10^{-8}$ and time interval $[0, 40]$ the algorithm does converge to the stopping threshold of 10^{-9}. However the evolution of the parameters clearly shows room to improve, as it can be seen in Figure 6. In the non scaled version, D_{gBZ} and D_{gBY} quickly increase in the beginning (whereas, they should be decreasing), while most of the parameters stay constant until around the simulation 40. This is typically a sign that the descent direction has a much higher component for D_{gBZ} and D_{gBY}, making the problem poorly conditioned. Indeed if we plot a radial mean of the gradient with regard to each parameter (see Figure 7) the gradient with regards to D_{gBZ} and D_{gBY} is clearly orders of magnitude higher than the one with regards to the other parameters. And since the target and initial values for D_{gBZ} and D_{gBY} are also quite low, their relative values move very quickly. Additionally a difference in the gradient value can be remarked for both Z_0 and Y_0.

We then introduce some scaling to improve the conditioning of the problem. The simplest idea would be to use a linear scaling function with a very low coefficient somewhere between 10^{-3} and 10^{-6}. However considering that we want an evolution of D_{gBZ} and D_{gBY} spanning multiple orders of magnitude, it is tempting to use instead its logarithm inside the minimisation routine. This leads to an

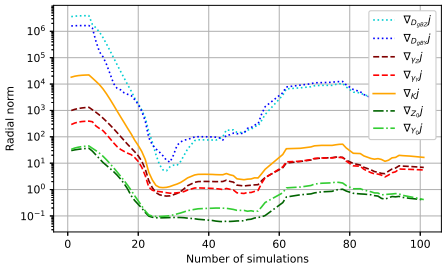


Fig. 7: Evolution of the norm of the gradient of the cost with regards to each parameter for the 100 first simulations of calibration runs with no scaling on D_{gBZ} and D_{gBY}. In contrast with Figure 6, we only plot values at the end of iterations, not the intermediate simulations. An L^2 norm is used for the parameters dependent on the radius.

exponential scaling function $s : \bar{x} \mapsto k \exp(\bar{x})$ for $\bar{x}$ the rescaled parameter and k the linear scaling coefficient. It has two interesting properties. First it insures that the parameter is always positive. Secondly it leads to a gradient decreasing with the value of the parameter :

$$\nabla_{\bar{x}} j = \nabla_{\bar{x}} s(\bar{x}) \nabla_x j = k \exp(\bar{x}) \nabla_x j \,,$$

that yields

$$\nabla_{\bar{x}} j = x \nabla_x j \,. \tag{19}$$

However this compensates an opposite evolution of the gradient with regards to the value D_{gB}. If we approximate (with, for simplicity, an explicit scheme of time step Δ_t) the value of the gradient with regard to D_{gBZ} on one iteration we have $\nabla_{D_{gBZ}} j = [\nabla_{D_{gBZ}} Z] \, (Z - Z^{obj})$ with

$$\nabla_{D_{gBZ}} Z = \nabla_{D_{gBZ}} \left(Z_0 + \Delta_t \left(D_{gBZ} \frac{1}{\rho} \nabla_\rho (\rho \frac{Z_0^2}{Y_0} \nabla_\rho Z_0) + \gamma_z Z_0 - K Z_0^2 - Y_0 \right) \right)$$

that yields

$$\nabla_{D_{gBZ}} j = \left(\Delta_t \frac{1}{\rho} \nabla_\rho (\rho \frac{Z_0^2}{Y_0} \nabla_\rho Z_0) \right) \, (Z - Z^{obj}) \,. \tag{20}$$

We have that the gradient with regard to D_{gBZ} and D_{gBY} is higher if the radial gradient of either Z and Y is higher. And indeed this is what occurs if there is less diffusion, namely, for lower values of D_{gBZ} and D_{gBY}.

Hence we introduce and compare both the linear scaling with the best coefficient experimentally found and the exponential scaling with a linear coefficient of 1.0 (see Figure 6). In both cases the other parameters start evolving much faster as expected. It mostly prevent D_{gBZ} and D_{gBY} to reach unnecessary high values which may then slow down the convergence, or even possibly prevent it. If we now consider the plot of the cost and gradient norm, Figure 8, the times where the value is below the limit of the graph correspond to set of parameters which make the solver diverge. We can see that this happens multiple times without the exponential scaling and this is often caused by either D_{gBZ} or D_{gBY} being

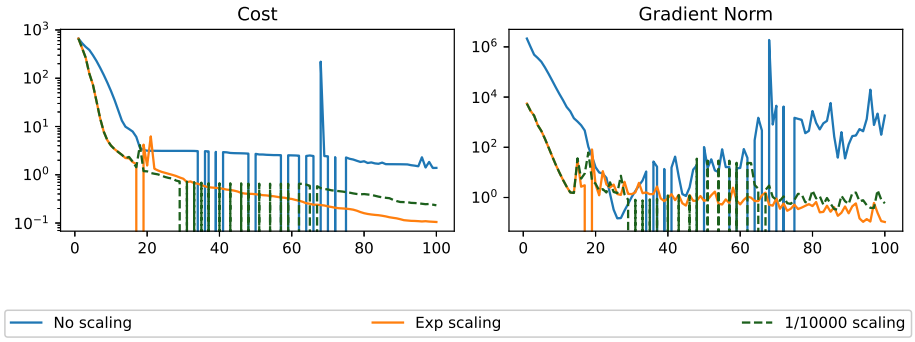


Fig. 8: Evolution of the cost and the norm of its gradient for the 100 first simulations of calibration runs with different types of scaling for D_{gBZ} and D_{gBY}. The value of the cost and its gradient are set to zero when the direct model does not converge.

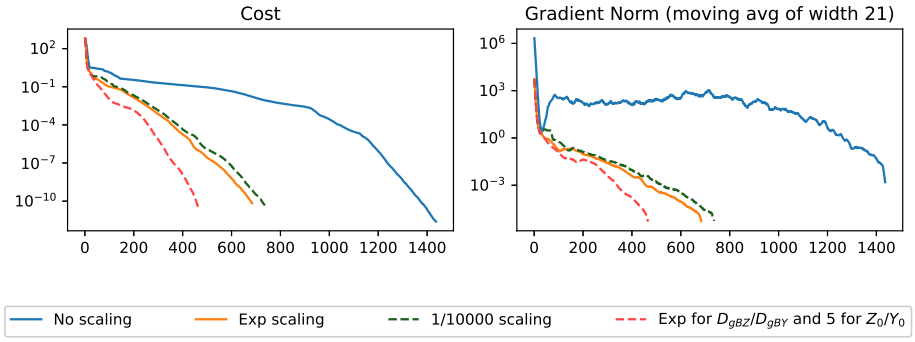


Fig. 9: Evolution of the cost and the norm of its gradient at the end of all the iterations of calibration runs with different types of scaling for D_{gBZ} and D_{gBY}, with regards to the number of simulations.

lower than 0. Hence, since the exponential scaling ensure the positivity of D_{gBZ} and D_{gBY} and does not require to find an efficient scaling coefficient we keep it for the following, although it should be noted that after the initial phase both the exponential scaling and the 1/10000 linear scaling lead to very similar results (see Figure 9). Finally the gradient norm value with regards to Y_0 and Z_0 seen in Figure 7 appears noticeably lower than the other ones. A simple linear scaling of coefficient 10 is introduced to compensate and it effectively speed up again the convergence (see Figure 9).

4.2.2 Time interval

We now want to focus on the importance of the choice of the time interval. It may indeed be critical for the overall computation cost of the whole algorithm, since it may influence the convergence rate and is directly linked to the compu-

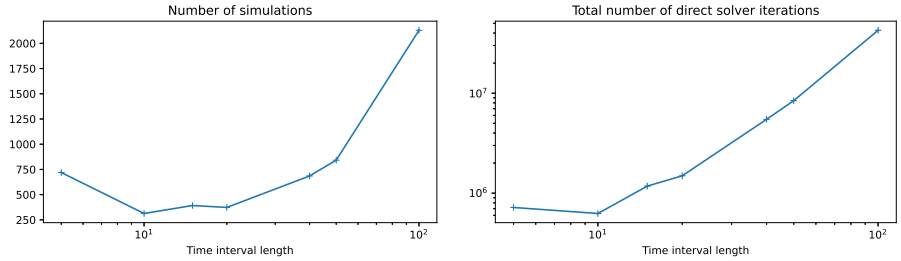


Fig. 10: Number of simulations (left) and total number of iterations of the direct $\kappa - \epsilon$ solver (right) for different time intervals.

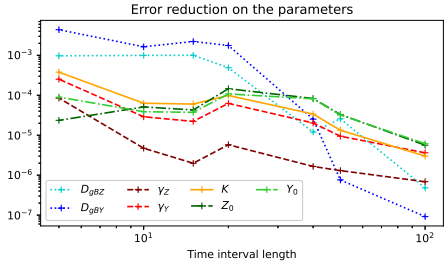


Fig. 11: Ratio of initial and final errors (L^2 distances with targets) on the parameters for different time interval lengths.

tation cost of each iteration, the time step being fixed. For the case we consider the mean period of oscillation of Z and Y at each radial point is around 10, and the first tests were performed with a time interval of length $L_T = 40$ so with approximately 4 oscillations. As it can be seen in the left part of Figure 10, the number of simulations is minimised for a time interval of length $L_T = 10$, corresponding with one mean oscillation. This is a considerable improvement in total computational cost because it more than halves the number of simulations necessary for the convergence and divides by 4 the number of iterations of the direct solver at each iteration, so approximately reducing the overall cost by a factor of 10, as it can be remarked on the right part of Figure 10. However this reduction of the computational cost is compensated by a reduction of the precision on the recovered parameters. Figure 11 shows the improvement (ratio of initial and final L^2 distances with targets) on the parameters for the different time interval tested. Indeed the final precision on the parameters is considerably higher for longer time intervals. This is partially due to a stopping threshold too high for the lower time intervals. Moreover the improvement on the initial condition Y_0 and Z_0 seem to evolve less than for the other parameters while the difference on the precision of D_{gBZ} and D_{gBY} is huge. Obviously the shorter time interval would allow to identify quickly and precisely the initial condition, what would allow a faster convergence of the algorithm. However using the rest of the data would be beneficial to increase the precision on the other parameters, and would explicitly insure that the simulation matches most of the data. Indeed it is then possible to do a first

L_T	number of simulations	err D_{gBZ}	err D_{gBY}
40	685	1.18×10^{-2}	2.5×10^{-2}
10 then 40	554 (313 then 241)	1.52×10^{-3}	6.79×10^{-4}

L_T	err γ_Z	err γ_Y	err K	err Z_0	err Y_0
40	2.73×10^{-6}	5.85×10^{-7}	2.04×10^{-5}	8.27×10^{-6}	8.15×10^{-6}
10 then 40	3.34×10^{-7}	4.5×10^{-8}	3.73×10^{-6}	1.12×10^{-6}	1.01×10^{-6}

Table 3: Comparison of the results of one run of the algorithm over a long time interval $[0, L_T]$ against a run over a short time one followed by a run over a long one. The error on the parameters is a L^2 distance between the retrieved parameters and their target counterparts.

minimisation with a short time interval length, and then improve it with a longer one using the parameters retrieved by the first run $p^{end,1}$ as first guesses for the second : $p^{end,1} = p^{0,2}$. In this case we remove the scaling on D_{gBZ} and D_{gBY} for the second run of the algorithm, because theirs initially low values lead to very low gradient components. Since the gradient norm significantly increases when passing from the first to the second minimisation, the second stopping condition is relative to the new gradient : $\|\nabla j(p^{0,2})\|/\|\nabla j(p^{end,2})\| < \epsilon_g^2 = 10^{-9}$. As we can see in Table 3, we obtain less simulations in total as well as an even more precise estimation.

For the following we will then focus on an interval length corresponding roughly with the mean period of one oscillation, assuming in most cases that the precision obtained is sufficient or could be improved efficiently with a second run of the algorithm.

4.3 Radial derivative penalisation

Two issue lead to the consideration of a penalisation on the radial derivative of the parameters. First the apparition during the minimisation of big oscillations and spikes, especially close to the wall (right) boundary, which may reach the region of instability of the direct solver with a risk of the divergence of the calibration algorithm. Indeed the value of the Dirichlet boundary condition is spread by diffusion so that it is the dominant term in the vicinity of the boundary. But until the diffusion weights D_{gBZ} and D_{gBY} reach their target values the algorithm try to compensate the error with the other parameters, leading to very sharp shapes. We expect the radial derivative penalisation to limit the apparition of those sharp shapes. Secondly for higher value of D_{gBZ} and D_{gBY} the algorithm does not seem to be able to reach consistently the expected target parameters. In this case the very high diffusion will generally flatten the variables Z and Y so that the shape of the parameters is significantly harder to recover from them. Thus for the tested highest value of D_{gB} the algorithm seems to stay stuck on quasi local minima from where it can't get out in a reasonable number of iterations. The radial derivative penalisation should make the cost function convex, in order to help the convergence towards the global minimum.

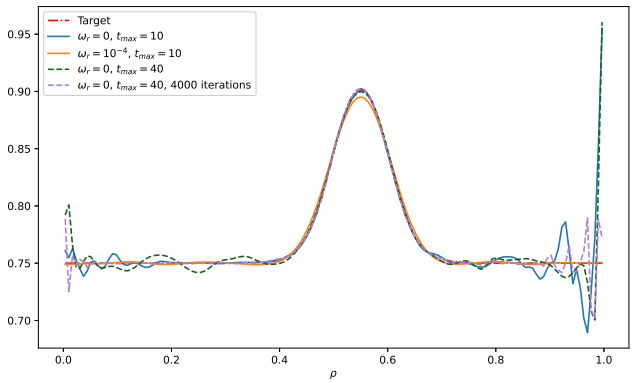


Fig. 12: Final recovered shape of γ_Y for different time intervals, with or without penalisation of the radial gradient. The violet dashed curve is ran until reaching the limit of simulations instead of a threshold on the gradient norm.

We will focus first on the latter issue, and try to improve on the former in section 4.5. If we consider the shape recovered for γ_Z with a target $D_{gBZ} = D_{gBY} = 10^{-2}$ (see Figure 12) we see that sharp oscillations are visible next to both boundaries. Furthermore, the shape obtained is different for the two tested time intervals and even if we let the algorithm run until 4000 simulations the oscillations are still very noticeable. On the other hand, the orange curve shows the result with a small radial derivative penalisation weight $w_\rho = 2.5 \times 10^{-7}$. This specific value is set to correspond with the one used in [1], where the formula for the penalisation is just a difference, not divided by the radial step. Indeed the boundaries are almost flat but there is a noticeable error on the bell shape of γ_Y (and generally where the radial derivative of γ_Y is the highest). Hence, although this method introduces an artificial error, it is quite predictable in the sense that we have a good idea of where the recovered shape may be less precise. Moreover the convergence rate is largely increased with only 715 simulations to reach the stopping condition against 1145 without penalisation.

However, since many parameters are flat, this situation could be a little too favourable to the radial derivative penalisation. To have a more fair comparison without changing the case of study, we will try to start the time interval from later than the value 0. Since the system is autonomous this is equivalent to consider as initial states Z_0 and Y_0 the shapes recorded in the data at a given time. We choose to start at the time 40 where the limit cycle begin to be well established and hence the shapes of Z and Y are far from being flat, as it can be seen in Figure 13. The first guesses for the initial conditions Y_0 and Z_0 are also adapted to not be too far from the target values.

Figures 14 and 15 compare the number of simulations and the improvement of the recovered parameters with or without penalisation. Indeed there is a clear improvement on convergence rate for high values of D_{gB}^{obj} with penalisation while it is more or less equivalent with or without penalisation for lower values of D_{gB}^{obj}. If we compare the improvement on the parameters, they are more precisely identified without penalisation except for the highest target D_{gB}. The penalisation is

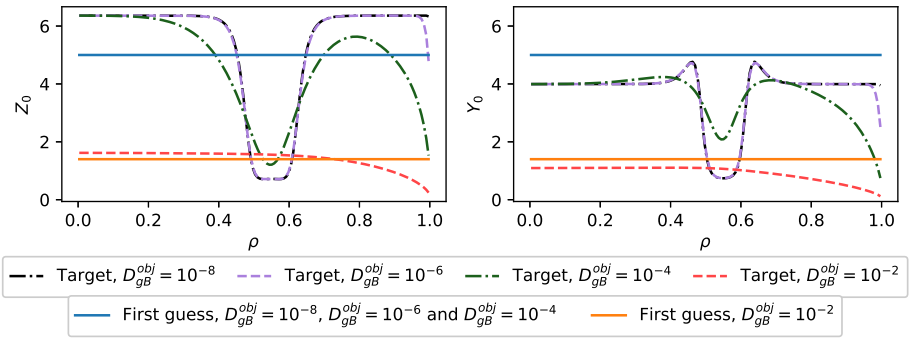


Fig. 13: Values of the target and first guesses for the initial states Z_0 and Y_0 for the different D_{gB}. The targets are obtained by running the simulation until $t = 40$, starting with $Z(0) = Y(0) = 2.0$ for the given D_{gB}^{obj}.

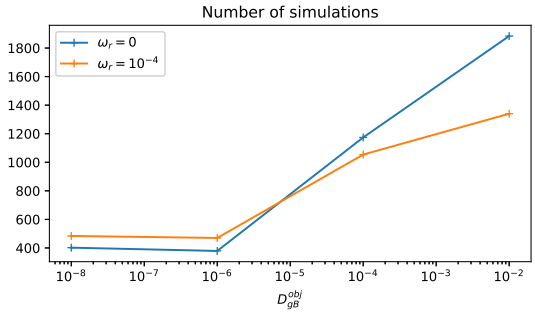


Fig. 14: Number of simulations before convergence with or without radial derivative penalisation, for different values of D_{gB}^{obj}

interesting though because it leads to approximately the same improvement for each target value of D_{gB}^{obj}, making it more reliable. It is still possible to launch a second calibration without penalisation (or with a reduced weight) starting from the recovered parameters to improve them even more. However if it is known that the diffusion weights are very low, it is clearly advantageous to use directly a calibration without radial derivative penalisation, which gives a much higher precision in the same number of simulations.

4.4 Logarithmic cost and penalisation

To improve again the efficiency of the algorithm we try to replace the formula of the cost by a norm of the difference the logarithm of the variables. The reasoning is that with a simple L^2 cost, the biggest differences between the target and current trajectories is probably be computed when the trajectories reach their maximal value, and that by comparison, the error at the beginning of the simulation would

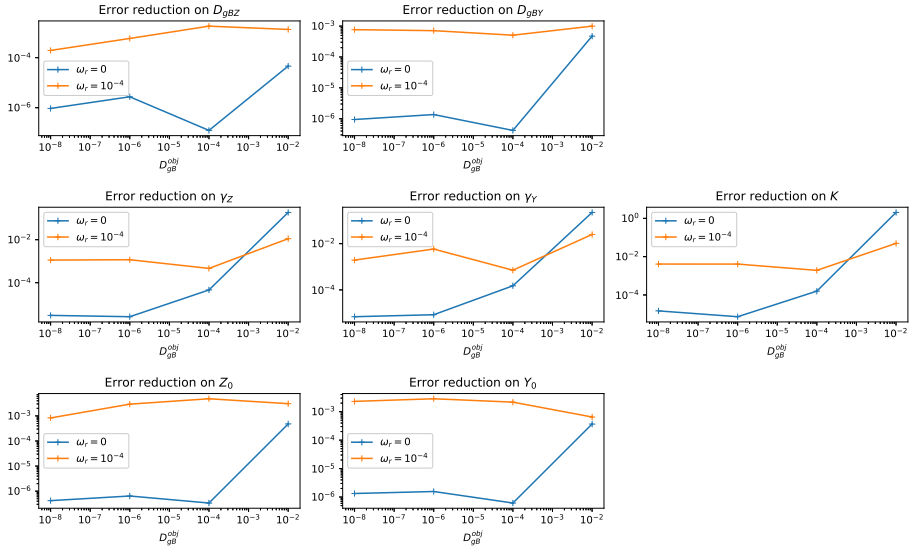


Fig. 15: Error reduction on the final recovered parameters with or without radial derivative penalisation, for different values of D_{gB}^{obj}

appear negligible. However due to the exponential growth of the local model, a consequent relative error made at the beginning of the growth will cause a big error in absolute value at the apex of the trajectory. Hence to maximise the information of the whole trajectory, it seems interesting to consider a relative error instead of an absolute one, and the former is given by a difference between the logarithm of the currently generated trajectory and its target.

If we replace only the formula of the cost the results appear slightly disappointing on our configuration of reference. The two blue curves of Figures 16 and 17 compares the calibrations with a standard and a logarithmic cost both with a radial derivative penalisation of weight 2.5×10^{-5}. As it can be seen, while the logarithmic cost systematically reduces the number of simulations necessary for convergence, it also deteriorates the final precision of the retrieved parameters.

However we can also point out that if the variation of the relative values of the variables Z and Y are more relevant than their absolute variation, the penalisation of the radial derivative might be more adapted if it was applied to the derivative of the logarithm, specifically for Z_0 and Y_0. Indeed changing the formula of the penalisation for the initial states -while still keeping the same cost- appear to improve considerably the precision of the retrieved parameters, for each kind of costs (ref. Figures 16 and 17). Only for the highest value of D_{gB}^{obj} the basic penalisation leads to slightly better results for γ_Z, γ_Y and K, but this is easily compensated by the largely higher number of simulations necessary for convergence. Since Z_0 and Y_0 are significantly superior to 1, it seems likely that the effect of the logarithmic penalisation is mostly equivalent to a reduced weight w_ρ for Z_0 and Y_0 in this case. It could essentially be a strategy to limit the need to modify the weight of the penalisation depending on the values of the initial states, which would make

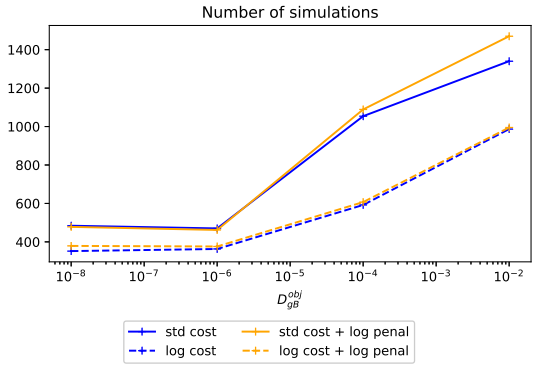


Fig. 16: Comparison of the number of simulations for calibrations with and without logarithmic cost and/or logarithmic radial derivative penalisation, for different values of D_{gB}^{obj}.

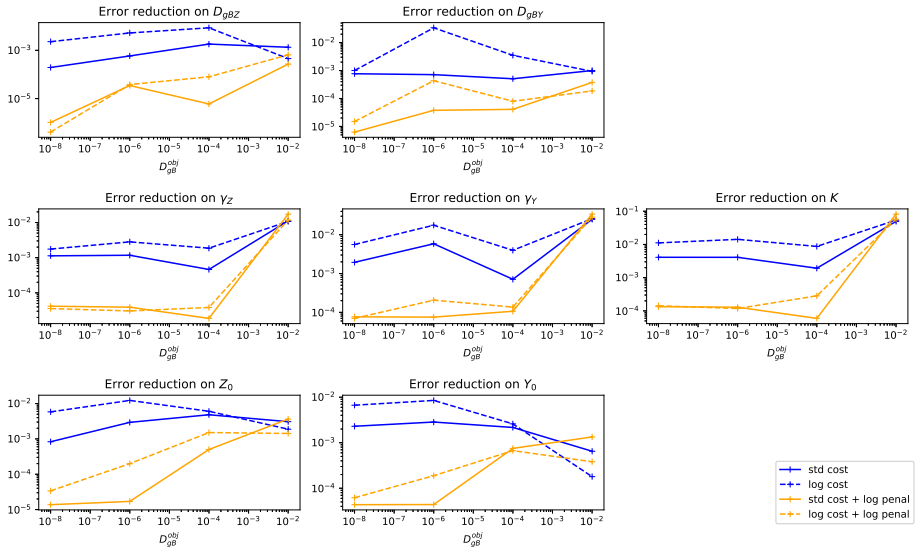
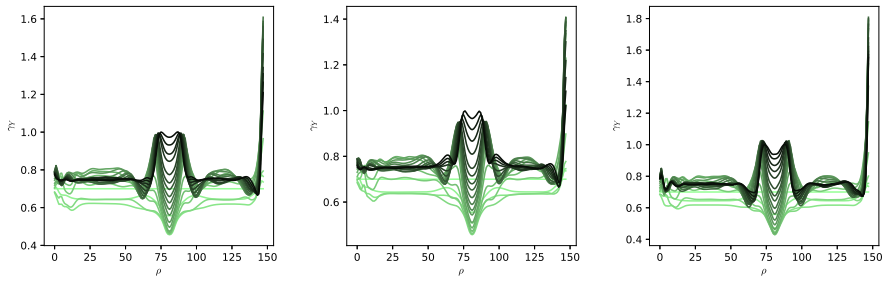


Fig. 17: Comparison of the reduction of the error on the parameters, for calibrations with and without logarithmic cost and/or logarithmic radial derivative penalisation, for different values of D_{gB}^{obj}.

the algorithm more robust. Considering other configurations could help confirm this interesting premise. Regardless of these consideration, with this new penalisation, the logarithmic cost appear the most efficient, being able to reduce the number of simulation while keeping a good precision.



(a) $w_\rho^1 = 0$ (b) $w_\rho^1 = 2.5 \times 10^{-5}$ (c) $w_\rho^1 = 2.5 \times 10^{-4}$

Fig. 18: Comparison of the evolution of the shape of γ_Z during the first 100 simulations (one in five, the later ones are darker) with or without the double launch of the minimisation routine. The calibrations are made with the standard cost and $L_T = 10$. For the double launch, $\epsilon_g^1 = 10^{-7}$, and it is not reached for the plotted simulations.

4.5 Reduction of the penalisation weight over multiple launches

Finally to improve again the robustness of the algorithm we try to launch the algorithm twice with a reduction of the penalisation weight between the two, while keeping the same time interval length. This is mostly because the radial penalisation weight chosen is still not enough to limit the apparition of sharp shapes on the wall boundary of the domain ($\rho = 1$) during the first hundred iterations, as can be seen in the first figure of 18. This effect seems more intense with lower diffusion coefficients so we focus specifically on $D_{gB^{obj}} = 10^{-8}$.

To prevent this specific issue, launching twice the minimisation algorithm with different radial penalisation weights does not appear very efficient. Still in Figure 18, we see a slight reduction of the maximum value at the edge boundary as we increase the weight of the first launch of the algorithm w_ρ^1, but this is at the cost of a considerably increased computational cost. Figure 19 shows that even with a moderate first weight $w_\rho^1 = 2.5 \times 10^{-5}$, the number of simulations is considerably increased even if we try to optimize the stopping threshold of the first launch ϵ_g^1. Furthermore considering higher weights generally lead to a convergence of the first launch on a set of parameters very far from their targets values so that the stability of the whole is once again at risk when the minimiser is launched for the second time.

Hence, since the double launch does not seem very promising with our configuration of reference, we don't keep it for the robustness tests of Section 5.

5 Robustness of the algorithm

As the algorithm seems rather efficient at identifying the parameters that were used to generate the data, we have to consider the fact that real data will not exactly correspond to a trajectory that the model can generate. Indeed the model is by design a simplification of the full turbulent system and there will always

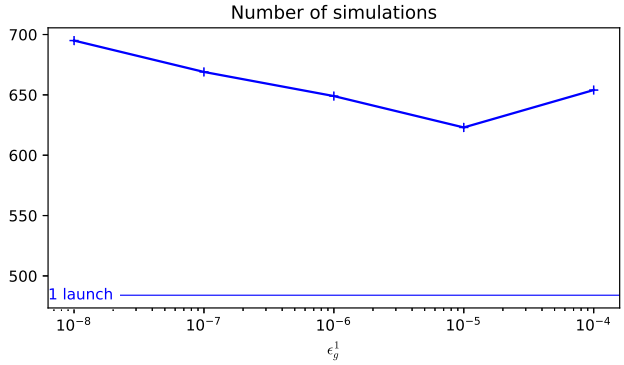


Fig. 19: Comparison of the total number of simulations with two launches of the algorithm for different values of the first stopping threshold ϵ_g^1. The logarithmic versions of both the cost and the radial derivative penalisation for Z_0 and Y_0 are used. The horizontal line show the number of simulations of the procedure with only one launch of the algorithm for the corresponding case.

D_{gB}^{obj}	D_{gBZ}^0	D_{gBY}^0	γ_Z^0	γ_Y^0	K^0	Z_0^0	Y_0^0
10^{-8} to 10^{-4}	10^{-5}	10^{-5}	1.2	0.7	0.02	5	5
10^{-2}	10^{-5}	10^{-5}	1.2	0.7	0.02	1.4	1.4

Table 4: Initial guesses for each test of the calibration procedure in Section 5

be noise and biases due to the measurement method. Thus it seems important to test our minimisation algorithm with data that is not the full and unaltered record of Z and Y generated with the same model as the one whose parameters are fitted. For this purpose, we will add some generated noise to the data, and try to reduce the amount of information available both in time and in space. For all calibrations, the time window starts at 40. The first guesses are the same as in the last section, and reminded in Table 4. The stopping threshold, the radial penalisation weight and the length of the time windows, which are slightly adapted depending on the the value of D_{gB}^{obj} and the chosen cost type, are defined in Table 5. The stopping threshold ϵ_g is increased for the standard cost for most values of D_{gB}^{obj} because the final iterations add almost no precision to the parameters, while the descent direction loose too much precision with the noise, leading to possible divergence. For $D_{gB}^{obj} = 10^{-2}$ however, a consequent reduction of the gradient norm is important to retrieve the parameters with a decent precision. They remain the same for all the calibration tests. Finally, the scaling function types and coefficients for each parameter are reported in Table 6.

D_{gB}^{obj}	Log cost	L_T	ϵ_g	w_ρ	Log penal.
10^{-8} to 10^{-2}	Yes	10	10^{-7}	2.5×10^{-7}	Yes
10^{-8} to 10^{-4}	No	10	10^{-7}	2.5×10^{-7}	Yes
10^{-2}	No	10	10^{-9}	2.5×10^{-7}	Yes

Table 5: Time interval, penalisation type and weights used in the tests of the calibration procedure in Section 5. Log penal indicate whether the radial derivative penalisation for Z_0 and Y_0 is computed on their logarithm or not.

	D_{gBZ}^0	D_{gBY}^0	γ_Z^0	γ_Y^0	K^0	Z_0^0	Y_0^0
Scaling type	Exp	Exp	Lin	Lin	Lin	Lin	Lin
Linear coef k	5	5	1	1	1	5	5

Table 6: Scaling types and coefficients used in each test of the calibration procedure in Section 5 for the different parameters. For the scaling types, Exp stands for exponential, and Lin for linear.

5.1 Time sparsity

We first want to test the influence of the time step Δt^{obj} between two values t_i^{obj} and t_{i+1}^{obj} of the time variable corresponding to recorded vectors of data (Z_i^{obj}, Y_i^{obj}) and (Z_{i+1}^{obj}, Y_{i+1}^{obj}), representing Z and Y on the whole radial space. Figures 20 and 21 show the number of simulations as well as the final errors on the retrieved parameters for the two extremal values of D_{gB}^{obj}. The test are made with a logarithmic cost but similar results are expected with a standard cost. The main takeaway seem to be that the algorithm is remarkably insensitive to the time sparsity as curves for both the error reduction and the number of simulations necessary for convergence are quite flat. The reduction of the number of data samples even allow a better identification of the initial states since it gives more importance to the initial data sample, which is equal to the initial steps. However, in real settings, using of all data data sample available still seems better to reduce the effect of model error or noises, as we will see in Section 5.2.

5.2 Resistance to noises

So far we have used data generated by the model itself, so that we know they can be precisely reproduced by a given set of parameters. But indeed this would not be the case for real data, because the governing equations are always a simplification of the real system, and any measurement device has a limit on the accuracy it can achieve, most of the time largely above the machine precision. A simple way to reproduce those kinds of discrepancies between the model and the data is to add some generated noise to the data we consider.

First we add to every radial component of every sample of data a random real in the interval $]-\epsilon_N/2; \epsilon_N/2[$ for a varying amplitude ϵ_N. Since this may lead to

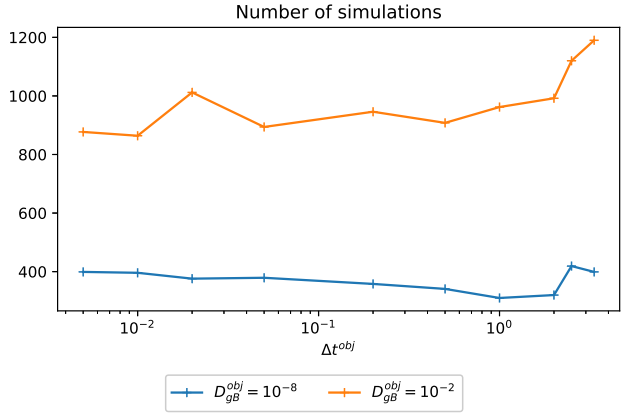


Fig. 20: Impact of the data time step on the number of simulations for different values of D_{gB}^{obj}. We use the logarithmic version of the cost.

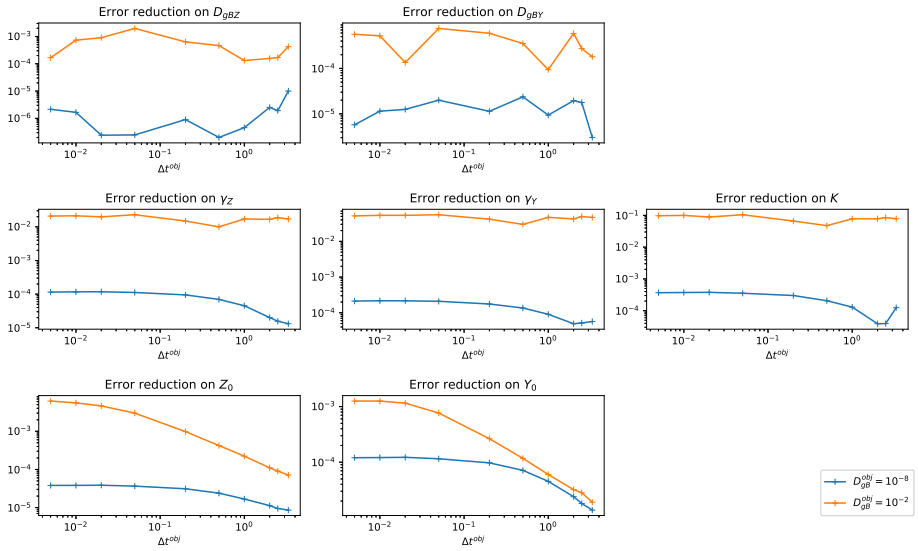


Fig. 21: Impact of the data time step on the number of simulations for different values of D_{gB}^{obj}. We use the logarithmic version of the cost.

the apparition of negative values, we then take the maximum between the each scalar element of the noisy data and $Z_{min} = Y_{min} = 10^{-6}$.

Since we want to show the effect of the choice of cost type (standard or logarithmic), we limit ourselves to one value for the diffusion coefficients : $D_{gB}^{obj} = 10^{-8}$. Figures 22 and 23 show the results of the calibration algorithm for a growing uniform noise amplitude. The logarithmic cost version appears considerably less stable, losing precision on the final parameters and starting to diverge faster as

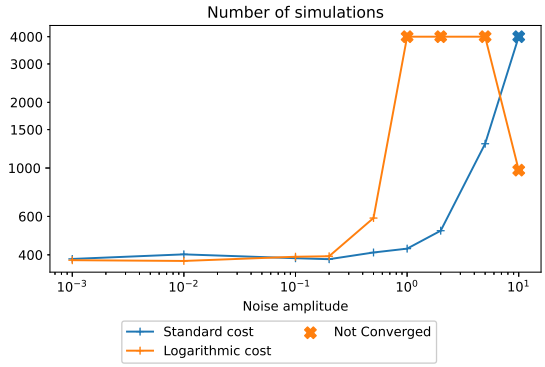


Fig. 22: Number of simulations before convergence for different noises amplitudes. $D_{gB}^{obj} = 10^{-8}$. The vertical axis is logarithmic.

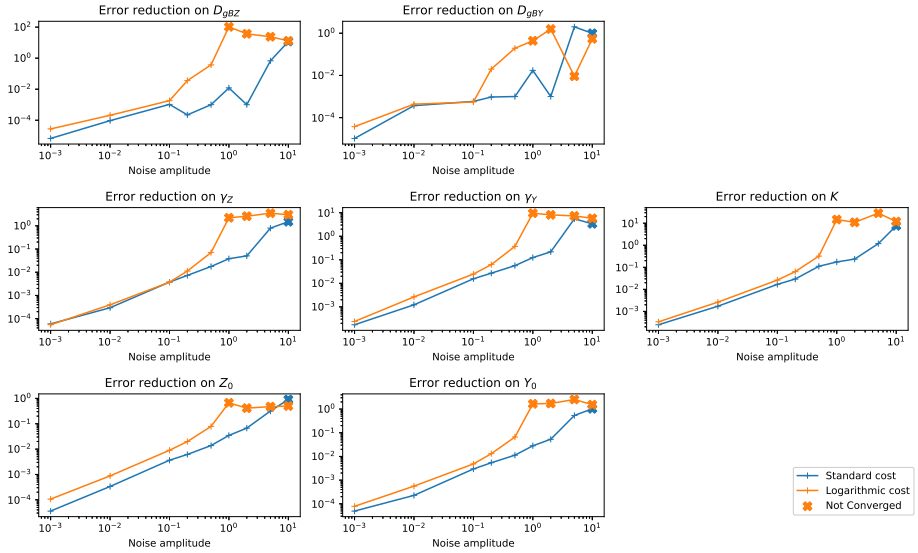


Fig. 23: Final error on the parameter for different noises amplitudes. $D_{gB}^{obj} = 10^{-8}$.

soon as the noise amplitude increases. Indeed since the amplitude of the noise does not depend on the amplitude of the variables, the addition of the noise leads to a significant modification of the relative value of the variables when and where they are originally low. The worst case is indeed when the value of the variable is inferior to the amplitude of the noise, since a negative value could be reached. The enforcement of the minima Z_{min} and Y_{min} cannot avoid a sudden fall of the relative value of Z or Y that the algorithm will try to reproduce, leading to inaccuracies in the retrieved parameter or even divergences.

To have a more fair comparison with the logarithmic cost, we create a relative noise simply by multiplying the uniform noise by the value of the variables before

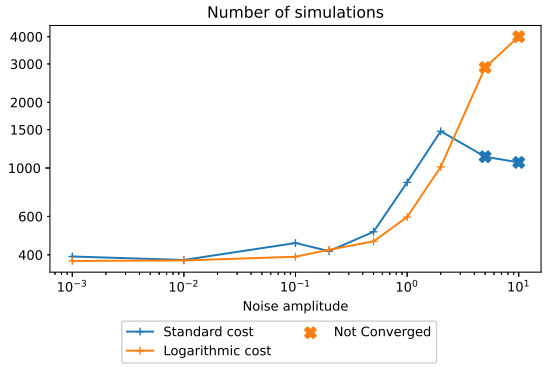


Fig. 24: Number of simulations before convergence for different noises amplitudes with a relative noise. $D_{gB}^{obj} = 10^{-8}$. The vertical axis is logarithmic.

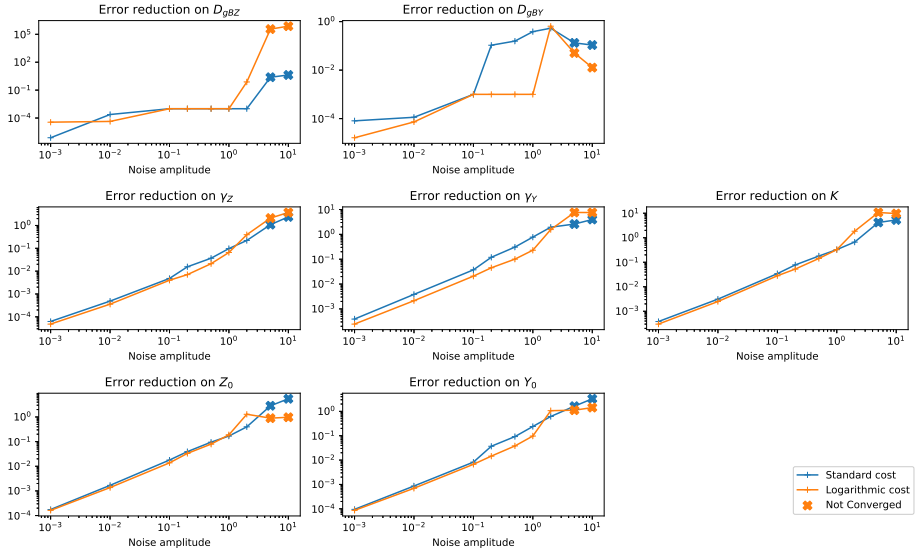


Fig. 25: Final error on the parameter for different noises amplitudes with a relative noise. $D_{gB}^{obj} = 10^{-8}$.

adding it. With $\tilde{X}$ the noisy version of the variable $X = Z$ or Y and U the realisation of a vector of random variables following the uniform law on [0,1] :

$$\tilde{X} = X + X\epsilon_N U \tag{21}$$

The results for this new case are given in Figures 24 and 25.

Although this time both cost function allow the convergence in almost every case, the standard cost still seems slightly more robust. Indeed it converges for one more case and seem to have a lower error on the retrieved parameters for the highest noises amplitude. The logarithmic cost seems however to be advantageous

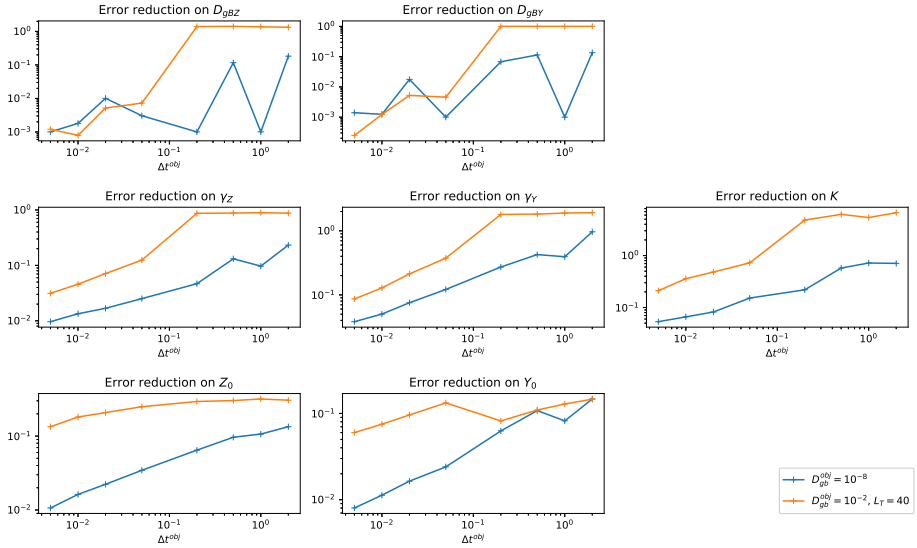


Fig. 26: Final error on the retrieved parameters with uniform noise for different values of Δ_t^{obj}, for different configurations with the standard cost. The amplitude of the added noise is $\epsilon_N = 1.0$.

in term of computational cost, requiring less simulations for convergence in almost every cases.

All in all, the algorithm appears quite robust with the standard cost, since in both configuration we can converge with noises of order of magnitude similar to the data. Moreover the final error on the parameters stays relatively stable until a noise amplitude of $\epsilon_N = 10^{-1}$, where it begins to sensibly increase. This is already a considerable amplitude so it is satisfying that we can still have a precise identification at this point.

The presence of noise may also make the assimilation more dependent on the data time step Δ_t^{obj}. Indeed, with more data sample the error introduced by the noise are more likely to compensate each other and we can expect a better precision on the retrieved parameters. Figure 26 shows how the error on the parameters evolve while changing Δ_t^{obj} with added uniform noise (not relative). We use rather large uniform noise amplitude, $\epsilon_N = 0.1$ for $D_{gB}^{obj} = 10^{-2}$, which is almost a twentieth of the maximum value in the data, and even $\epsilon_N = 1$ for $D_{gB}^{obj} = 10^{-8}$, what put the noise in the same order of magnitude as the recorded variables. Accordingly, we can notice a progressive increase of the error on the parameters γ_Z, γ_Y and K with Δ_t^{obj}, so using as much data as possible -inside an appropriately chosen time interval- is probably always the best option. Concerning the number of simulations necessary for convergence, it seems weakly impacted by Δ_t^{obj}. It grows slowly until $\Delta_t^{obj} = 0.5$ for $D_{gB}^{obj} = 10^{-8}$ and is almost constant for $D_{gB}^{obj} = 10^{-8}$ (see Figure 27). At this point, the robustness of the minimisation algorithm with the standard cost appears quite satisfying, as it can converge with both added noise and time sparsity, and so without requiring much more simulations to converge.

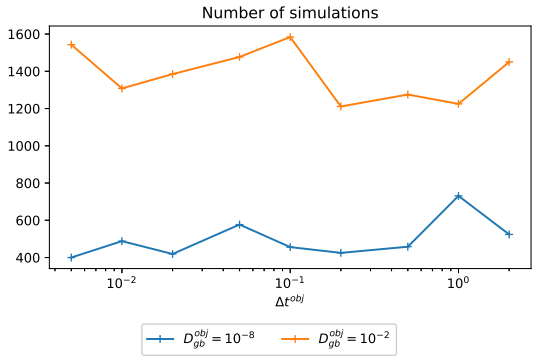


Fig. 27: Number of iterations before convergence for different values of Δ_t^{obj} with added noise, using the standard formula for the cost. The amplitude of the added noise is $\epsilon_N = 1.0$.

5.3 Radial sparsity

In this last section we briefly study the impact of not having data available at any radial point. We simply choose a radius ρ_{max} and do not use the difference of the current trajectory and the data corresponding to any radius above in the formula of the cost. Figure 28 shows the shape of the retrieved parameters for different values of ρ_{max}. As can be seen, after ρ_{max} the shape of the retrieved parameters almost immediately departs from their target except in the case $\rho_{max} = 0.8$. Even in this last case since the shapes after $\rho = 0.8$ are simply flat, the fact that the parameter are fully recovered could just be the effect of the radial derivative penalisation. Moreover the convergence is very slow, with 1604 simulations for $\rho = 0.8$, 3186 for $\rho = 0.5$ and even ending at the limit of 4000 simulations without reaching the stopping condition for $\rho = 0.3$. Thus it appears impossible to reliably estimate the shape of a parameter in radial regions where we don't have data available : the calibration algorithm is not really able to deal with spatial sparsity in this case.

6 Conclusion

In this paper we have presented a variational data assimilation approach to identify the parameters of a 1D time dependent $\kappa-\epsilon$ model for tokamak plasma turbulence. Using a twin numerical experiment, the paper shows how the tuning of parameters intern to the optimisation routine as well as different regularisation strategies can improve the efficiency of the calibration algorithm. The methods used are very general and the different considerations taken to tune the algorithm could be transposed for the calibration of many transitory model for turbulence or not. The main results of the paper can be summarized as:

– A scaling adapted to each parameter as well as a good choice of the time interval when dealing with oscillatory variables can considerably impact the convergence rate of the algorithm.

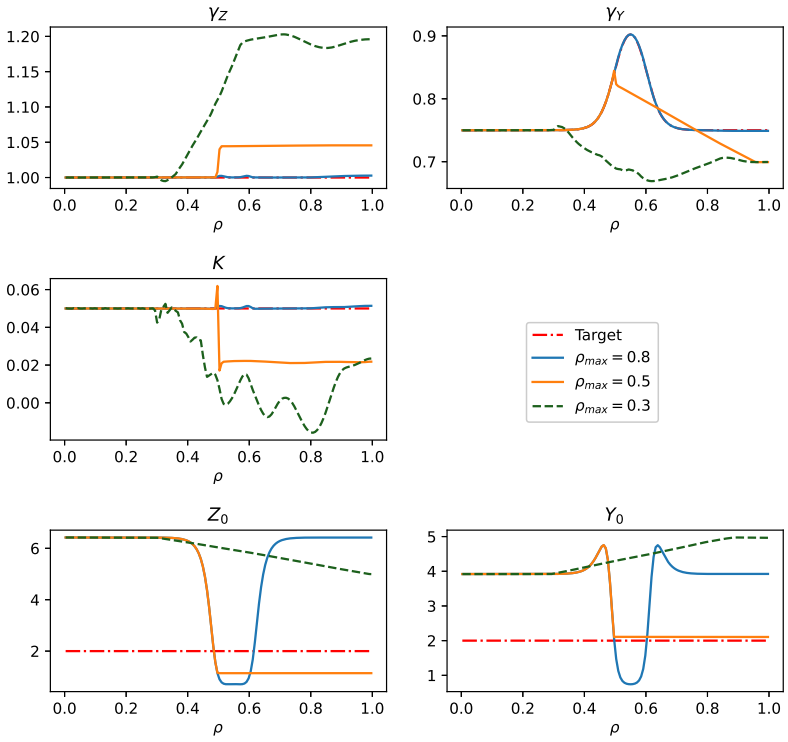


Fig. 28: Shapes of the retrieved parameters when data is not available above a certain radius ρ_{max}. For $\rho = 0.3$ the algorithm did not reach the stopping condition before the limit of 4000 simulations but we still display the last set of parameters. $D_{gB}^{obj} = 10^{-8}$.

- The introduction of a penalisation on the radial derivative of the parameters improves the robustness of the algorithm, and notably reduced the imprecision due to a Dirichlet boundary condition on cases with higher diffusion.
- The introduction of different formulas for the cost and the radial derivative penalisation for accounting for the higher dependency of the local system to relative differences in the value of the variables rather than absolute ones mainly lead to a reduction of the number of simulations necessary for the convergence of the algorithm.
- Applying the minimisation routine twice while reducing the weight of the radial penalty between the two runs to avoid the appearance of erroneous values near the wall boundary had too small an effect compared to the increase in the number of simulations.
- The main default of the calibration procedure seems to be its inability to identify the value of the parameters in radial region where data are not available. With a more complete model where the $\kappa - \epsilon$ equations are coupled to the plasma equations system [2], the relationship between different radii will be

richer than a diffusion term, so the calibration procedure should tackle better spatial sparsity.
– Finally, the present results show the robustness of the algorithm to time sparsity and noise. Although the logarithmic version of the cost has shown some weakness with noise independent of the local value of the data, the standard version of the cost seems remarkably robust, being able to converge with both intense noise and data very sparse in time.

Acknowledgments

Finantial disclosure: This work has been supported by the French National Research Agency Grant SISTEM (Grant No. ANR-19-CE46-0005-03). It has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No. 101052200-EUROfusion). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Competing Interests: The authors have no relevant financial or non-financial interests to disclose.

Data availability: Data are available on request from authors.

References

1. D. Auroux, P. Ghendrih, L. Lamerand, F. Rapetti, and E. Serre. Asymptotic behavior, non-local dynamics, and data assimilation tailoring of the reduced $\kappa - \varepsilon$ model to address turbulent transport of fusion plasmas. *Physics of Plasmas*, 29(10):102508, 2022.
2. S. Baschetti, H. Bufferand, G. Ciraolo, P. Ghendrih, E. Serre, P. Tamain, and the WEST team. Self-consistent cross-field transport model for core and edge plasma transport. *Nuclear Fusion*, 61(10):106020, 2021.
3. A. F. Bennett. *Inverse Modeling of the Ocean and Atmosphere*. Cambridge University Press, Cambridge, 2002.
4. H. Bufferand, G. Ciraolo, Y. Marandet, J. Bucalossi, Ph. Ghendrih, J. Gunn, N. Mellet, P. Tamain, R. Leybros, N. Fedorczak, F. Schwander, and E. Serre. Numerical modeling for divertor design of the west device with a focus on plasma wall interactions. *Nuclear Fusion*, 55(5):053025, 2015.
5. F.-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus*, 38A:97–110, 03 1986.
6. J. Ch. Gilbert and C. Lemaréchal. Some numerical experiments with variable-storage quasi-newton algorithms. *Mathematical Programming*, 45:407–435, 1989.
7. J. Ch. Gilbert and C. Lemaréchal. The module m1qn3. `https://who.rocq.inria.fr/Jean-Charles.Gilbert/modulopt/optimization-routines/m1qn3/m1qn3.pdf`, 2009.
8. L. Hascoët and V. Pascual. The Tapenade Automatic Differentiation tool: principles, model, and specification. Research Report RR-7957, INRIA, 2012.
9. ITER-Organization. Iter official website. `https://www.iter.org/`, 2023.
10. E. Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, Cambridge, 2003.
11. J.-L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations*. Springer Berlin, Heidelberg, 1971.
12. S.B. Pope. *Turbulent flows*. Cambridge University Press, Cambridge, 2000.

13. S. Wiesen, D. Reiter, V. Kotov, M. Baelmans, W. Dekeyser, A. S. Kukushkin, S. W. Lisgo, R. A. Pitts, V. Rozhansky, G. Saibene, I. Veselova, and S. Voskoboynikov. The new SOLPS-ITER code package. *J. Nucl. Mater.*, 463(5):480–484, 2015.
14. P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.