Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Parameter identification for a reduced transport model in fusion plasma

L. Lamérand | **D. Auroux** | **F. Rapetti**

[1]Université Côte d'Azur, Inria, CNRS, LJAD, Nice, France

Correspondence
L. Lamérand

Present address
louis.lamerand@univ-cotedazur.fr

Abstract

Two-dimensional transport codes for the simulation of tokamak plasmas are reduced versions of full 3D fluid models where plasma turbulence has been smoothed out by averaging. One of the main issues nowadays in such reduced models is the accurate modelling of transverse transport fluxes resulting from the averaging of stresses due to fluctuations. Transverse fluxes are assumed driven by local gradients, and characterized by ad-hoc diffusion coefficients (turbulent eddy viscosity), adjusted by hand in order to match numerical solutions with experimental measurements. However these coefficients vary substantially with the tokamak, the type of experiment and even the location inside the device, reducing drastically the predictive capabilities of these codes for a new configuration. To mitigate this issue, we recently proposed an innovative path for fusion plasma simulations by adding two supplementary transport equations to the mean-flow system for turbulence characteristic variables (here the turbulent kinetic energy k and its dissipation rate ϵ) to estimate the turbulent eddy viscosity. The remaining free parameters are mainly driven by the underlying transport physics and hence vary much less between machines and between locations in the plasma. In this paper, as a proof of concept, we explore on the basis of digital twin experiments, the efficiency of data assimilation to fix these free parameters involved in the transverse turbulent transport models in the set of 2D averaged equations.

KEYWORDS
Data Assimilation, Fusion Plasmas, Parameter Identification, Model Reduction, Numerical Simulations

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, see 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, according to ITER official website [25]. However, many physical and technological issues remain, requiring intensive numerical simulations to complement the sparse experimental measurements and incomplete theoretical models [30].

Despite the exponential growth of computer speed along with significant improvements in computer technology, the numerous physics and engineering issues to address as well as the very large number of degrees of freedom to handle require the development of a chain of models. This latter ranges from low to high fidelity models, from simplified models for optimization and uncertainty propagation to state-of-the-art first principle models of plasma turbulence transport in relevant plasma conditions. In this context, as recently mentioned in a review article [39], the transport codes ([14] [9] [42] [37]) remain the current workhorse of the physicists when studies are closely linked to operations with the aim of studying and designing optimal scenarios for reactors. They rely on a very similar approach to the Reynolds Averaged Navier-Stokes (RANS) codes commonly used for engineering applications in computational fluid dynamics (CFD) [33]: in these fluid reduced models, turbulence is smoothed out by averaging the transport fluxes (transverse to the magnetic field lines, Figure 1) resulting from the averaging of stresses due to fluctuations which are assumed to be driven by local gradients. The effect of the turbulence on the averaged

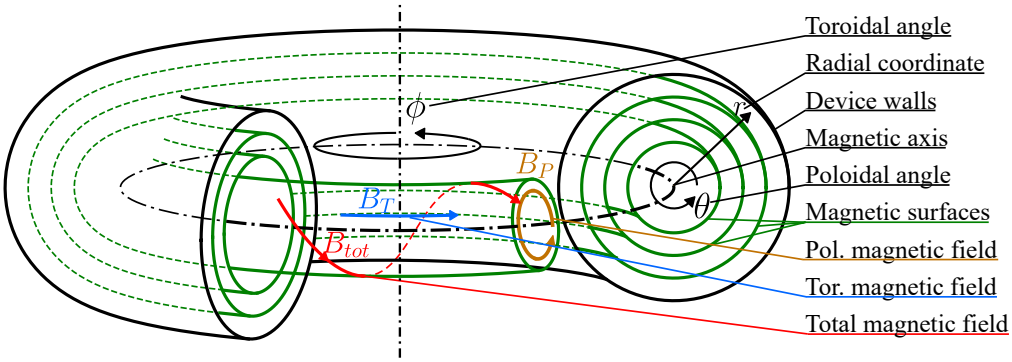


FIGURE 1 Sketch of a simple magnetic geometry of a Tokamak in toroidal and poloidal coordinates. Particles are (imperfectly) bound to helicoidal trajectories which follow the total magnetic field lines.

fields is then characterized by ad-hoc diffusion coefficients (turbulent eddy viscosity) whose values are generally tuned by hand to match experimental data (see for example [42] and [9]). The issue here is that since these ad-hoc coefficients, which must be determined at each point of the mesh, are flow-dependent, they 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 [1]. They must then be considered as free parameters with an extremely large number of degrees of freedom, which drastically reduces the predictive capabilities of these codes.

In the literature, few works have explored the way to solve this issue. The option of a direct coupling between a mean-field code and a local (flux-tube) plasma turbulence code was proposed by Hasenbeck *et al.* [23] for example. In such a scheme, the mean-field code still follows a gradient-diffusion hypothesis to describe transverse transport, but the transport coefficients are dynamically and self-consistently obtained from the output of turbulence simulations run in a gradient-driven manner (with local plasma parameters and gradients used as input) at well-chosen sampling positions in time and space. This approach has been proven successful at reproducing plasma profiles in simple test cases. However, the application to complex cases remains to be undertaken. Additionally, a number of fundamental questions remain in this context, e.g. how to efficiently determine the sampling points and decisive parameters from the mean-field code for the local turbulence simulations, how their choice affects the stability, convergence and performance of such a coupled code system, and if the underlying hypothesis of a scale separation between macroscale (mean-field) and mesoscale (local turbulence) dynamics always holds in the regimes and regions of interest to ensure the feasibility of the approach. Also the physical limitations of this concept as compared to global turbulence simulations and the computational costs still have to be assessed in depth. Similar problems arise when local turbulence codes are used to build a database of transport coefficients depending on the mean-field code's input parameters [32]. To solve the computational cost issue, another approach consists in completing the model with equations describing the time and/or space evolution of the transport coefficients, thus making the call to a turbulence code unnecessary. This approach has already been used successfully in the frame of 1D models for L-H transition studies [31, 41] and can potentially be applied to 2D or 3D edge mean-field models. 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 another one, or from one location to another in the same plasma. Also, their dependencies with plasma characteristics can in general be derived or constrained by theoretical considerations.

Following this premise, an advanced modelling has been proposed in [6], improving significantly the predictive capability of the model, inspired by a method used in the neutral fluid community since the 70s [28]. In this enriched model, the transverse turbulent diffusion coefficients are computed from the turbulence kinetic energy κ and its dissipation rate ε.

Moreover, κ and ε are then determined self-consistently by 2 additional transport equations designed from the knowledge of the physical mechanisms at play at the plasma edge in tokamaks. The new equations depend on a new set of parameters, which are closed by previously established results of perturbation analysis [38], and known scaling laws [19]. Still, some free coefficient remain to be tuned, and the scaling laws would benefit from a precise confrontation with the data; but, by characterizing fundamental processes of turbulence, the new model is expected to need far less tuning depending on the scenario of plasma discharge, or even the type of Tokamak, in contrast to the initial diffusion coefficients. Results in Baschetti *et al.*

[5, 6] showed the remarkable potential of this model to capture key aspects of the physics of turbulent transport throughout the plasma.

Hence, the introduction of the new variables essentially shift the problem of identification from the turbulent diffusion coefficient to the free parameters of the new equations for κ and ϵ. This is essentially at this point that comes the main contribution of this article. Indeed, the objective is to show the ability of a calibration routine, based on data assimilation, to identify those parameters. The ultimate objective will be to use data bases provided by experimental measurements or higher fidelity numerical models to improve the reliability and predictability of the new model of transport code. While data assimilation is traditionally used in oceanography or meteorology [7, 26] to estimate a global 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 are available [2]. In the present work, we choose to exploit the Variational Data Assimilation (VDA) framework [13] which involves the minimisation of a cost function defined as the quadratic distance between the data of reference and the values computed by the model [13, 2, 10, 24]. The gradient of the cost function can then be obtained by automatic differentiation (see Portal for automatic differentiation http://www.autodiff.org or [21]), allowing the use of efficient minimisation algorithms (e.g. conjugate gradient, quasi-Newton, ...), see *e.g.*, [29]. However, as a complex nonlinear optimisation problem, the convergence of the calibration procedure at a reasonable rate is not necessarily guaranteed. Fortunately, the formulation as an optimisation problem gives access to different strategies, like rescaling the optimisation variables or using a penalisation function, which can greatly improve the performances of the algorithm. An extensive part of this article will illustrate how to efficiently use those strategies and adapt them precisely to a model for plasma turbulence.

First, in section 2, the simplified model on which the calibration procedure will be tested is thoroughly introduced with an analysis of its expected behaviour and the introduction of the reference scenario for the calibration tests. Then, in section 3 the calibration algorithm is detailed, from its mathematical principle to the precise definition of its components, as well as the different regularisation strategies used to improve its efficiency and robustness. In chapter 4, the different regularisation strategies are introduced step by step to evaluate their impact and explain how they were adapted to the considered model. Finally, in chapter 5 the robustness of the calibration with the most efficient hyperparameters is tested on multiple different cases, where the information contained in the data is artificially reduced by adding random noise (or rendering it sparser in time).

2 | MODEL SIMPLIFICATION, NORMALISATION AND ANALYSIS

2.1 | 1D model in the field-aligned radial direction

We consider a simplified $\kappa - \varepsilon$ model for transport. We refer the reader to [39] and [6] for more details about the simplification of the full transport model equations to 1D.

Inside the tokamak, magnetic field lines are spiralling around nested surface of approximately toroidal shape, called magnetic surfaces (see figure 2). We consider a generalised radius r following a direction always perpendicular to the magnetic surfaces. We define then the average operator $\langle\cdot\rangle$ of a quantity a on a magnetic surface S of constant generalised radius r surrounding the volume V as:

$$\langle a\rangle(r) = \frac{\partial}{\partial V}\int_{V(r)} a\, dV = \frac{1}{V'(r)}\int_{S(r)} \frac{a\, dS}{|\vec{\nabla} r|} \text{ with } V'(r) = \left(\frac{\partial V}{\partial r}\right)(r) = \int_{S(r)} \frac{dS}{|\vec{\nabla} r|}. \tag{1}$$

The full system of averaged variables include equations for the common density n (assuming quasi-neutrality), almost identical equations for ions and electrons temperature T_α ($\alpha = i$ or e for ion or electron) and the equations for κ and ε.

$$\frac{\partial n}{\partial t} - \frac{1}{V'}\nabla_r V' D \nabla_r n = S_n - h(r - r_{SOL})\frac{n}{\tau_\parallel}, \tag{2a}$$

$$\partial_t n T_\alpha - \frac{1}{V'}\nabla_r (V'(D_n T_\alpha \nabla_r n + r\chi_\alpha n \nabla_r T_\alpha)) = S_{E_\alpha} - h(r - r_{SOL})\frac{n T_\alpha}{\tau_\parallel}, \tag{2b}$$

$$\partial_t \kappa - \frac{1}{V'}\nabla_r \left(V' D_\kappa \nabla_r \kappa\right) = \gamma_\kappa \kappa - \frac{1}{D_\omega}\kappa^2 - \varepsilon, \tag{2c}$$

$$\partial_t \varepsilon - \frac{1}{V'}\nabla_r \left(V' D_\varepsilon \nabla_r \varepsilon\right) = \gamma_\varepsilon \varepsilon - V\frac{\varepsilon^2}{\kappa^{3/2}}, \tag{2d}$$

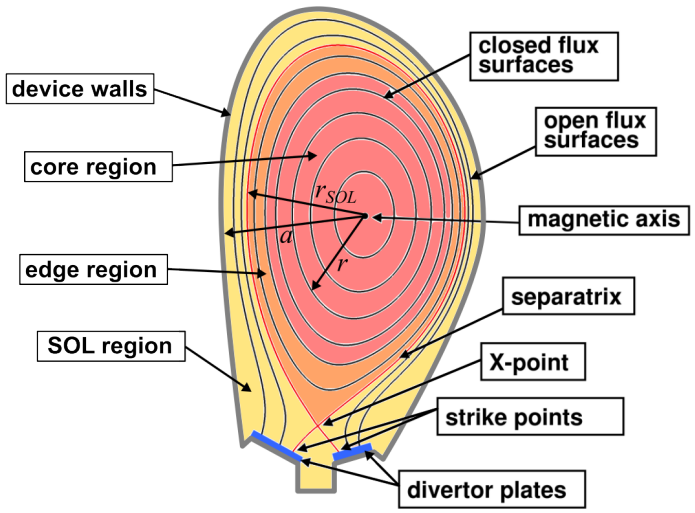


FIGURE 2 Presentation of the domain and variables on a poloidal cross-section of a Tokamak in the divertor configuration.

with h the Heaviside function. lister rapidement les différents paramètres: V', D, Dn, Dkappa, Deps, Domega, rSOL, taull, chi The sources for particle S_n and energy S_{E^α} can be set to represent the refueling of particles, the heating and/or the fusion reaction. Remarkably, there is no linear loss term due to the interaction with wall for κ and ε because the parallel transport can have a stabilizing impact on the turbulence at any radius, and not necessarily more in the SOL. The growth rates γ_κ and γ_ε should then be considered as effective turbulence growth rates, including the linear loss mechanisms. We will detail the different parameters of the $\kappa-\varepsilon$ in the normalisation phase.

With this simplified physics, we consider that the model can be extended to the core plasma, so that the variables are defined on the whole domain from the magnetic axis ($r = 0$) to the beginning of the first parts of the regular wall with radius higher than the limiter ($r = r_{SOL}$). Naturally we set a homogeneous Dirichlet condition at the wall boundary since all variables should have negligible values at this point (otherwise the device would simply be melting). On the magnetic axis, a homogeneous Neumann condition prevents from dealing with the singularity at the point where the radial coordinate vanishes.

2.2 | Decoupled and normalised $\kappa-\varepsilon$ model

The 1D model (see Equations (2)) has revealed to be an interesting tool to investigate plasma turbulence [6]. As a first proof of concept, and to simplify a thorough study of our calibration procedure, we focus in the present work on the $\kappa-\varepsilon$ system, decoupled from the density and temperature equations. Those plasma flow variables will be considered as constant in time in the following. Furthermore, we suppose that the magnetic surfaces are approximately toroidal to simplify the perpendicular diffusion terms in Equations (2). Including the boundary conditions, our system reads as:

$$\partial_\tau \kappa - \frac{1}{r}\nabla_r\left(rC_D^\kappa \frac{\kappa^2}{\varepsilon}\nabla_r\kappa\right) = \gamma_\kappa\kappa - \frac{1}{D_\omega}\kappa^2 - \varepsilon \tag{3a}$$

$$\partial_\tau \varepsilon - \frac{1}{r}\nabla_r\left(rC_D^\varepsilon \frac{\kappa^2}{\varepsilon}\nabla_r\varepsilon\right) = \gamma_\varepsilon\varepsilon - \frac{V}{\kappa^{3/2}}\varepsilon^2 \tag{3b}$$

$$\begin{aligned}&\kappa(\tau=0,r)=\kappa_0,\ \nabla_r\kappa(\tau,r=0)=0,\ \kappa(\tau,r=a)=0,\\ &\varepsilon(\tau=0,r)=\varepsilon_0,\ \nabla_r\varepsilon(\tau,r=0)=0,\ \varepsilon(\tau,r=a)=0.\end{aligned} \tag{3c}$$

The equations include transport terms both in the perpendicular (i.e. radial) and parallel directions with respect to the magnetic field lines. The adaption of the $\kappa-\varepsilon$ model is rather heuristic: the general structure of the $\kappa-\varepsilon$ for neutral fluid is conserved, and the different terms are adapted to account for plasma physics or enforce expected behaviour as detailed in [6] or [27]. The left hand side of equations shows a material derivative simplified by the averaging over the magnetic surfaces and the turbulent treatment of the perpendicular transport. The perpendicular diffusive transport due to turbulence are governed by diffusivities

$C_D^\kappa \nu_e$ and $C_D^\varepsilon \nu_e$ where $\nu_e = \kappa^2/\varepsilon$ is the eddy viscosity. On the right hand side, the local drives of the turbulence are expanded up to order 2: the γ coefficients are effective linear growth rates, D_ω is a saturation rate of κ which prevents the divergence of the system and V weights the damping term of ε, ensuring that ε eventually decreases as κ tends to 0.

Reference closure laws using the variables of the MHD equations have been obtained for the different parameters of the $\kappa - \varepsilon$ system. However, there are obtained by approximations and dimensional analysis so that most of them depend on low dimensional free coefficients and could be improved with by comparing to a solid experimental estimation. Hence, after the normalisation of the model, we will not use the known closures in our calibration tests.

The normalisation is based on the fixed point of the local model, so that if all the normalised parameters are set to one, the right hand side of Equations (3) is equal to 0. Approximate formula for the fixed points of κ_*, ε_* and ν_{e*} are ([27]):

$$\kappa_* \approx \frac{\gamma_\kappa^2 V^2}{\gamma_\epsilon^2} \Rightarrow \kappa_{*0}, \;\; \varepsilon_* \approx \frac{\gamma_\kappa^3 V^2}{\gamma_\epsilon^2}, \;\; \nu_{e*} \approx \frac{\gamma_\kappa V^2}{\gamma_\epsilon^2}. \tag{4}$$

Hence, we express the normalisation scales of κ, ε and ν_e using the scales for both growth rates, γ_0, and the scale for V, V_0:

$$\kappa_0 = V_0^2, \;\; \epsilon_0 = \gamma_0 V_0^2, \;\; \nu_{e0} = \frac{V_0^2}{\gamma_0}. \tag{5}$$

Furthermore, as illustrated by the formula of ν_{e0} in Equations (4), a decrease of both growth rates (γ_κ and γ_ε) would lead to an increase of the fixed point of the eddy viscosity which seems counterintuitive (the decrease of the growth rate of the turbulence and its damping rate should not increase the effect of the turbulence). To avoid this issue and to simplify the study and identification of the parameters of the model, the choice was made [3] to simply equate the normalised value of V and γ_κ: $V/V_0 = \gamma_\kappa/\gamma_0$.

We now set $Z = \kappa/\kappa_0$ and $Y = \varepsilon/\varepsilon_0$ and normalise the radial position by the system size, typically the plasma minor radius a, hence $\rho = r/a$, thus accounting for radial boundary conditions. Time is normalised by $1/\gamma_0$, and the normalised growth rates are $\gamma_Z = \gamma_\kappa/\gamma_0$ and $\gamma_Y = \gamma_\varepsilon/\gamma_0$, leading to $V/V_0 = \gamma_Z$ in order to ensure the expected behaviour for the fixed points. With these definitions and the chosen scales (5), the system (3) now reads:

$$\partial_t Z - \frac{1}{\rho}\nabla_\rho\left(\rho D_{gBZ}\frac{Z^2}{Y}\nabla_\rho Z\right) = \gamma_Z Z - KZ^2 - Y, \tag{6a}$$

$$\partial_t Y - \frac{1}{\rho}\nabla_\rho\left(\rho D_{gBY}\frac{Z^2}{Y}\nabla_\rho Y\right) = \gamma_Y Y - \gamma_Z \frac{Y^2}{Z^{3/2}}, \tag{6b}$$

$$\begin{aligned} &Z(t=0,\rho) = Z_0, \;\; \nabla_\rho Z(t,\rho=0) = 0, \;\; Z(t,\rho=1) = 0, \\ &Y(t=0,\rho) = Y_0, \;\; \nabla_\rho Y(t,\rho=0) = 0, \;\; Y(t,\rho=1) = 0. \end{aligned} \tag{6c}$$

Here, $K = D_0/D_\omega$. The notation refers to the concept of Kubo number, akin to the intensity of the turbulences [4, 34]. In this case the model is supposed to be in a low turbulence regime, so that $K << 1$ [3]. Finally, the normalised diffusion weights are $D_{gBZ} = C_D^\kappa/(\gamma_0 a^2)$ and $D_{gBY} = C_D^\varepsilon/(\gamma_0 a^2)$. Since one expects V_0 to scale like the normalised Larmor radius ρ_*, one then finds $D_{gBZ} \propto D_{gBY} \propto \rho_*^2$, which corresponds to the so-called gyro-Bohm (gB) scaling. Since we generally use a constant ratio Δ_Y between D_{gBZ} and D_{gBY}, we often give only the value of a general parameter D_{gB}, assuming $D_{gBZ} = D_{gB}$ and $D_{gBY} = \Delta_Y D_{gB}$.

3 | CALIBRATION ALGORITHM

3.1 | Optimisation problem and overview of the method

3.1.1 | Definition of the problem

As explained earlier, we want to develop a calibration routine to identify the parameters of the system of Equations (6) (D_gBZ, γ_Z, K, ...) so that the trajectory generated by solving the model corresponds to "experimental" data. In this case, our model is too simplified to be meaningfully compared to data from actual experiments. Thus, we will remain within the frame of twin experiments, where data are generated from our model for a given set of target parameters, and we then try to reproduce the data and recover the target parameters starting from first guesses, a different set of parameters. A notable advantage of testing the calibration procedure with the twin experiment is that we can directly evaluate the precision of the retrieved parameters

by comparison with the target parameters, and check if the algorithm does not stay stuck in an unexpected local minimum. Furthermore, it is possible to simulate the differences between experimental data and the generated trajectory with the addition of noise to the generated data, in order to test the calibration method on a more realistic case (see section 5).

Concretely, for the generation of the data, we run a solver of the system (6) and regularly save the value of Z and Y after a given number of iterations. The target data are then 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^{data}. Naturally, Δ_t^{data} is a multiple of the time step of the direct system solver Δ_t.

Then we can define the cost function that the calibration algorithm will minimise as a simple Euclidean norm of the difference between the generated trajectory and the data. Assuming that, for a given set of parameters p, $Z(p)$ and $Y(p)$ are the unique solutions of the direct model (6), and 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:

$$J(p) = \sum_{i\in[\![0,N_T]\!]} \frac{1}{2}\left(\|(Z(p))(t_i) - Z_i^{obj}\|_2^2 + \|(Y(p))(t_i) - Y_i^{obj}\|_2^2\right). \tag{7}$$

The different parameters of the set p are all considered constant in time but dependent of the radius, except for the diffusion weights D_{gBZ} and D_{gBY}. Considering that the purpose of the model is to evaluate the eddy viscosity, correcting it with an ad-hoc parameter at every space point would clearly reduce its interest. Overall, the set p includes 7 parameters to identify. There are 5 of them in the equations of the direct model (6) :

- D_{gBZ} and $D_{gBY} \in \mathbb{R}^+$ the coefficients of the (nonlinear) diffusion terms, 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 are expected to stay positive.
- $K \in L^\infty([0,1]) \to \mathbb{R}^+$ the Kubo parameter (also referred to as the Strouhal number) weighting the nonlinear saturation term in the equation (6) for Z. The value of K is usually fairly negligible between 10^{-1} and 10^{-4}, and supposed to remain positive in order to act as a saturation term.

We add the two the initial states $Z_0 = Z(t_0)$ and $Y_0 = Y(t_0)$. In our case, the initial states are directly given by the first vectors in the data sets, Z_0^{obj} and Y_0^{obj}, but 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.

Lastly, as we will see in Section 4.3, considering a relative error 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 :

$$J_l(p) = \sum_{i\in[\![0,N_T]\!]} \frac{1}{2}\left(\|\log((Z(p))(t_i)) - \log(Z_i^{obj})\|_2^2 + \|\log((Y(p))(t_i)) - \log(Y_i^{obj})\|_2^2\right). \tag{8}$$

3.1.2 | Sketch of the algorithm

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

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^{data}. Then we have a loop between a nonlinear minimisation routine (the Fortran routine `m1qn3` [17] that implements a BFGS quasi-Newton algorithm) 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 reduced with regards to its initial value, namely, $\|\nabla j\|/\|\nabla j_0\| < \epsilon_g$, where $\epsilon_g > 0$ is the stopping condition threshold.

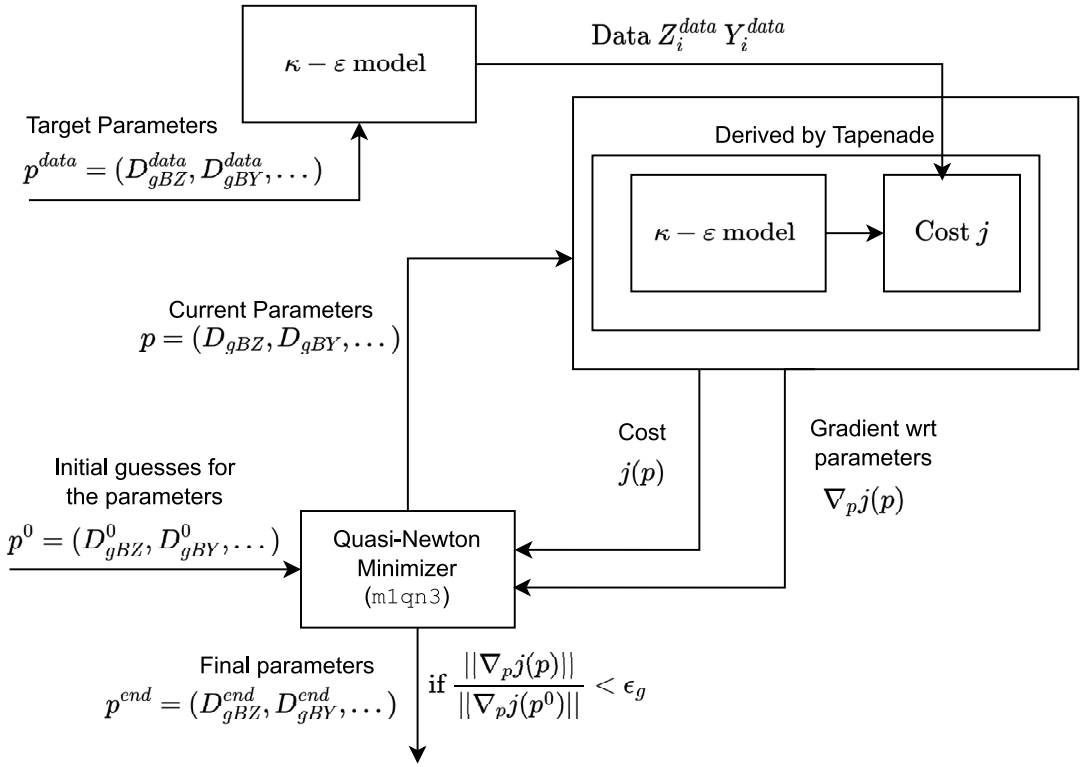


FIGURE 3 Block scheme of the minimisation algorithm.

3.2 | Discretisation and Solver

3.2.1 | Time discretisation

The chosen scheme used in the parameter fitting procedure to solve Equation (6b) is simple and fast but does not allow the convergence of the algorithm for any set of parameters. It makes use of an implicit-explicit scheme for the resolution of the nonlinear diffusion, and a part of the local drive:

$$Z_{n+1} - Z_n = \Delta_t \left(D_{gBZ} \mathcal{D} \left(\frac{Z_n^2}{Y_n} \right) Z_{n+1} + \gamma_Z Z_n - K Z_n^2 - Y_n \right), \tag{9a}$$

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

The interest of this formulation is that the scheme is linear in Z_{n+1}, with $\mathcal{D}\left(Z_n^2/Y_n\right)$ a linear (and even tridiagonal) operator depending on terms of the step n, and Y_{n+1} at the end of the right-hand side of (9b) is multiplied by a term depending only on the step n as well. Hence, the computation of the next step is dominated by the resolution of tridiagonal linear system, but the stability is largely improved from fully explicit cases because there is no CFL type condition on the value of the time step Δ_t (see *e.g.* [35] or [36]).

Still, as previously studied in [27], it can be complex to obtain a scheme for the local drive that would converge for any set of parameters. Notably, since Z_n must stay positive at any step, without considering the impact of the diffusion, the following minimal condition should always be fulfilled:

$$Y_n < \frac{Z_n}{\Delta_t}. \tag{10}$$

Since this condition depend on the values of Z_n and Y_n at every time step, it is hard to evaluate `a priori` the exact sets of parameters which will lead to the divergence of the scheme. In [27], general values of parameters that could lead to instabilities have been found to be high ratio of γ_Y/γ_Z, Z_0/Y_0 or a low value for K, but without much more precise identifications. This is

an interesting test for the minimisation algorithm, as we want it to be able to cope with regions of forbidden sets of parameters, not necessarily known very precisely.

3.2.2 | Data, cost and discrete gradient

The modification of the direct solver 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 distance of the current value of Z and Y to the matching data sample (Z_i^{obj} or Y_i^{obj}) is added to a cost variable. We assume that it is always possible to choose the time step Δ_t so that there is always an iteration for which the time variable in the solver corresponds exactly to the time of the data sample. For simplicity, we will always choose Δ_t^{data} as a multiple of Δ_t.

From there the discrete gradient is obtained by applying the automatic differentiation software Tapenade [22] in reverse mode to the code computing the cost. The automatic differentiation tools can transform a mathematical code which computes a function into a code which computes its derivative. The program generated in reverse mode will first run the original solver, memorising the values of Z and Y at each iteration, and then run backward a derived and transposed version of the original code, which is an efficient technique to get the gradient of the cost known as the discrete adjoint method [22]. Since we are not limited by the available memory, the generated routine is very fast, around only 3 times slower than the direct solver to compute both the cost and its gradient with regard to the parameters. In cases where the memory is limited, a binomial checkpointing strategy can be employed for an optimal computational time using the available memory [20].

The complete run of the direct solver followed by the derived adjoint code, which computes both the cost and its gradient, is called a "simulation". Since there can be multiple simulations for one iteration of the minimisation routine, and as the internal computations of the minimisation routine are negligible, the main indicator of the performance of the calibration is the number of simulations before the stopping condition is reached.

Radial grid spacing Δ_ρ	Time step Δ_t	Time elapsed between two recordings Δ_t^{data}	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

Table 1 displays the default values of the discretisation parameters. The data are 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 do not 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 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 oscillatory, the objective data and the currently computed trajectory may quickly run out of phase even if the parameters are relatively close to their target values. Hence, it is interesting to reduce the length of the time interval to at most a few oscillations.

3.2.3 | Optimisation routine

As is natural for optimisation problems where the gradient is known, but the Hessian is unavailable or too costly to obtain, the chosen minimisation routine belong to the class of the Quasi-Newton methods. Those methods try to reproduce the very efficient Newton iteration using an approximation of the inverse of the Hessian of the cost.

Among the numerous available Quasi-Newton optimisation algorithms, we choose the routine `m1qn3` of the library MODULOPT [17], which implements a limited memory BFGS algorithm [8, 15, 40, 18] for unconstrained nonlinear optimisation (see [16]). The approximated inverse Hessian is evaluated using the gradient ∇J and optimisation variable p from a given number M of past iterations, which we set to $M = 20$ since higher values generally does not add much more precision [16]. 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.

3.3 | Regularising strategies

Due to the formulation of the calibration as an optimisation problem, multiple methods can be implemented to improve the performances of the calibration procedure, in terms of robustness against potential divergence, precision of the retrieved parameters and convergence rate. We will quickly define those strategies here and illustrate their impact and how to adapt them to our problem in Section 4.

3.3.1 | Scaling

Scaling functions are used to improve the conditioning of the problem and impose boundaries on some parameters. A given parameter x of the set p corresponds with the rescaled parameter $\bar{x} \in \bar{p}$, such that $x = s_x(\bar{x})$, where s_x is the scaling function. The non-scaled parameter x is given to the minimisation routine while the rescaled parameter $\bar{x}$ is used in the computation of the cost and its gradient. In this form it is very easy to apply constraints on the parameters inside the PDE solver: typically choosing an always positive function s_x leads to a positive $\bar{x}$. Furthermore, from the chain rule, the gradient of the cost with regard to a parameter x is naturally multiplied by the derivative of s_x, so that the scaling can be used to increase the components related to this parameter in the descent direction.

For a given rescaled parameter $\bar{x}$, we use for the scaling either one of the three following functions, which include 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.
- a positive quasi-linear (PQL) function. It is defined to have a horizontal asymptote for $x \to -\infty$ and an oblique asymptote for $x \to +\infty$ (inspired by a hint on the implementation of box constraints with an unconstrained solver in [11]):

$$s : \bar{x} \mapsto k\epsilon_{s;x} \left(-\frac{1}{\pi} + \frac{1}{-\operatorname{atan}\left(\frac{k_x \bar{x}}{\epsilon_{s;x}} + \frac{1}{\pi} \right) + \frac{\pi}{2}} \right). \tag{11}$$

It is used to prevent a parameter from reaching 0 at some points, without adding too much non-linearity at the other points. The parameter $\epsilon_{s;x}$ is a good approximation of a threshold for the behaviour of the function, as seen in Figure 4.

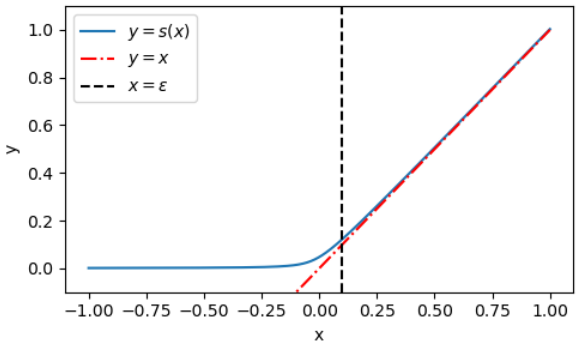


FIGURE 4 Plot of the positive quasi-linear scaling function s for $\epsilon = 10^{-1}$ and $k = 1$.

If not specified, the scaling functions for each parameter are linear with coefficients $k_x = 1$, which is equivalent to no scaling at all.

3.3.2 | 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. For each of them, a scalar weight balances its contribution to the cost. Except where it is mentioned, the penalisation terms and their associated weights are the same for each parameter. Multiple kinds of penalisation functions are used here on different test cases:

- Radial derivative penalisation: for each parameter, the quadratic norm of its derivative with respect to the radius is added to the cost function:

$$w_\rho \sum_{i=1}^{N_\rho - 1} \left(\frac{x_{i+1} - x_i}{\Delta_\rho} \right)^2. \tag{12}$$

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

$$w_\rho \sum_{i=1}^{N_\rho - 1} \left(\frac{x_{i+1} - x_i}{\Delta_\rho (x_{i+1} + x_i)/2} \right)^2. \tag{13}$$

  The weight of the radial derivative penalisation is w_p, equal to 0 by default.
- Negative value penalisation: since negative values in the parameters are either physically incoherent or may lead to the divergence of the numerical $\kappa - \epsilon$ model solver, we will use in some cases a penalisation function which simply adds to the cost the sum of the value to the power 4 of the parameters at the points where they are negative:

$$w_N \sum_{i \in \{j \in [\![1, N_\rho]\!],\ x_j < 0\}} x_i^4. \tag{14}$$

  We use the power 4 to ensure the smoothness of the function (to the order C^3 in this case) and avoid discontinuities in the derivatives. This is less restrictive than the use of a scaling function to force positiveness since the parameters can still be negative -the penalisation even has no effect if they are not. Hence, it is used by default on all the parameters with the weight $w_N = 0.1$.
- Reference value penalisation: A L_2 distance between a parameter and a reference value is added to the cost:

$$w_{ref,x} = \sum_{i=1}^{N_\rho} (x_i - x_i^0)^2. \tag{15}$$

  As can be seen in the formula, the reference value is actually the initial guess for the given parameter, because if we expect a certain value for a parameter, it appears natural to directly start from it. This penalisation may be used only for certain parameters, so the weight $w_{ref,x}$ depends on the parameter x.
- Parameter difference penalisation: Similar to the last one but the L_2 distance is between two parameters:

$$w_{diff,x,y} = \sum_{i=1}^{N_\rho} (x_i - y_i)^2. \tag{16}$$

  This penalisation is introduced because big differences between some parameters may lead to the divergence of the numerical scheme, specifically the ratios Y_0/Z_0 and γ_Y/γ_Z. Hence, limiting their differences may prevent the divergence of the calibration algorithm.

Finally, to maximise the efficiency of the regularising strategies, it can be interesting to launch the algorithm two times, starting the second launch with the parameters obtained after the first, typically to change the weights of the different penalisation terms 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.

	D_{gBZ}	D_{gBY}	γ_Z	γ_Y	K	Z_0	Y_0
Target	$10^{-4} - 5 \times 10^{-3}$	$2^{-6} \times D_{gBZ}$	5a	Fig 5a	0.05	Fig 5b	Fig 5b
Initial	10^{-5}	10^{-5}	1.2	0.7	0.02	$Z_0^{obj} + 0.1$	$Y_0^{obj} + 0.1$

TABLE 2 Target parameters for the reference configuration, compared to the initial guesses used for the first calibration.

By default, all penalisation weights are put to zero, except the negative value penalisation because it has no effect on the retrieved parameters if they are close to their targets. In the next section, we will progressively introduce the different regularising strategies and illustrate experimentally their effects.

4 | OPTIMAL HYPERPARAMETERS FOR THE TWIN EXPERIMENT

4.1 | Reference configuration

To test our calibration procedure, we choose a reference configuration which illustrates the peculiar dynamics arising from the nonlinear diffusion. This configuration is defined with a drop of the growth rates of both Z and Y, which are very low in a relatively large "No Man's Land" region around $\rho = 0.55$ compared to their values on the rest of the radial domain. The "No Man's Land" terminology is sometimes used in plasma physics to designate a region before the separatrix where the turbulence seems to decrease before increasing again in the SOL (Scrape-Off Layer) [12].

In order to focus on the effect of the diffusion, the initial states Z_0 and Y_0 are chosen as the fixed points of the local drive, so that at the initial time $t = 0$, the right-hand side of the normalised system is equal to 0 and only the nonlinear diffusion term instigates a movement in the system. However, the local fixed points Z^* and Y^* are strongly impacted by the value of the growth rates, with $\gamma_Z = \gamma_Y = \gamma$:

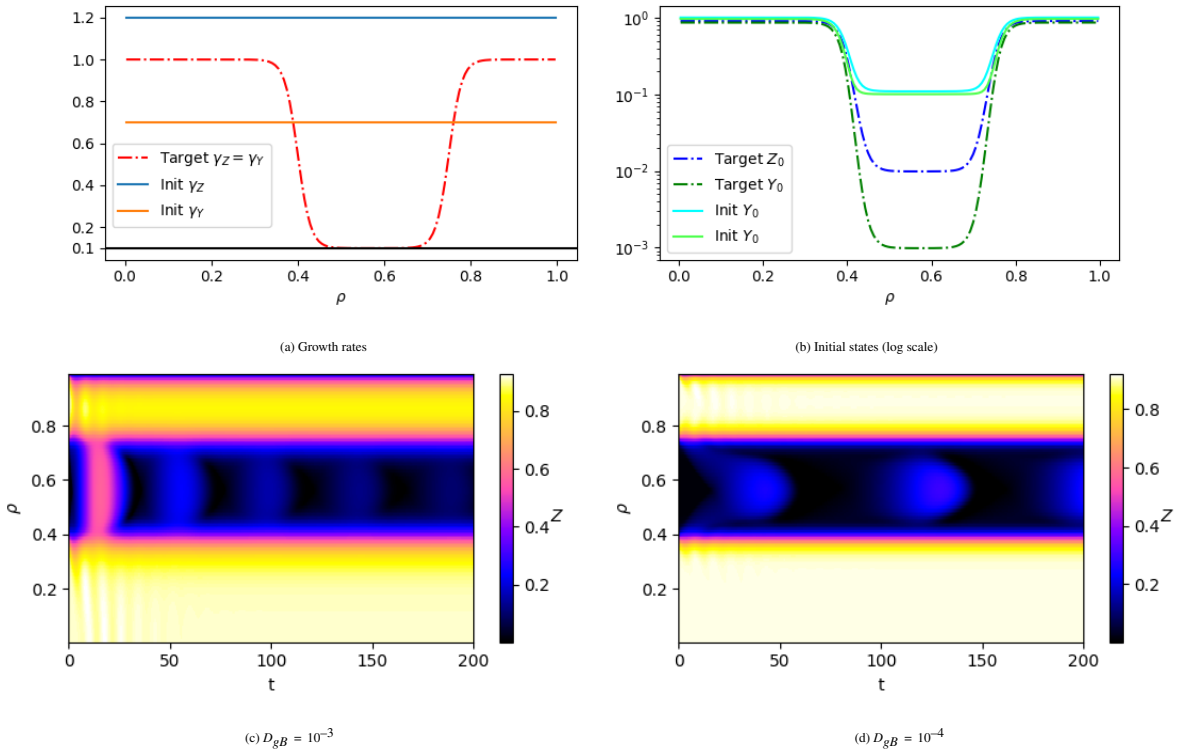
$$Z^* \propto \gamma^2 \text{ and } Y^* \propto \gamma^3. \tag{17}$$

Hence, the initial states display a considerable variation in scale from the boundaries to the No Man's Land region, as illustrated in Figure 5b. This is quite challenging for the algorithm because it has to identify parameters with values that can be both distant and very close to 0 but must always be positive (so that the direct solver does converge). The ability of the calibration routine to identify those parameters will be a convincing illustration of its robustness. Among all the tests, the target parameters will stay the same except for the diffusion weights D_{gBZ} and D_{gBY}, in order to see how the calibration procedure reacts to different diffusion weights. The values of the scalar or constant target parameters are recalled in Table 2 while Figures 5-(a) and (b) display the profiles of the radially dependent parameters.

As can be seen in Table 2, there is always a difference between the target diffusion weights for each variable. Indeed, as was shown in [3], D_{gBY} must be considerably lower than D_{gBZ} for the apparition of the complex oscillatory dynamic observed in Figures 5-(c) and (d).. In particular, we will choose $D_{gBY}^{obj} = \Delta_Y D_{gBZ}^{obj} = 2^{-6} D_{gBZ}^{obj}$, where the ratio $\Delta_Y = 2^{-6}$ is set at a reference value (see [3]).

4.2 | Initial penalisation and nonlinear scaling

Following the methodology of [27], we introduce by default an exponential scaling for D_{gBZ} and D_{gBY}, leaving the linear constant at 1. For the time length interval, we approximately use the period of one typical time oscillation. As can be seen in Figure 5, there seem to be oscillations of very different frequencies inside and outside the No Man's Land region, and the period of oscillation in the No Man's Land is dependent on the diffusion weights. To keep a similar interval length independent of the diffusion, we choose an interval length $L_T = 40$, which is approximately the smallest period observed in the No Man's Land, but still at least a half of the longest period observed. Finally, the negative penalisation is used by default on all the parameters with weight $w_n = 0.1$. It has a low impact on the algorithm since it only has an effect when a parameter has a negative value, but it should limit the apparition of too negative values. Similarly, we introduce a radial derivative penalisation on all radially dependent parameters with weight $w_\rho = 10^{-4}$, to improve the robustness of the algorithm by limiting the apparition of sharp oscillations. This penalisation prevents the parameters to reach their exact targets since it naturally flattens the curves of the parameters with regard to the radius, but it was shown in [27] that it can also improve the convergence of the algorithm, especially with higher diffusion weights.



(a) Growth rates

(b) Initial states (log scale)

(c) $D_{gB} = 10^{-3}$

(d) $D_{gB} = 10^{-4}$

FIGURE 5 Top: profiles of objective and initial guesses of non-constant parameters for calibration toward the reference configuration. Bottom: map of the variable Z solution of the normalised $\kappa - \epsilon$ system, as a function of time and radius for different values of $D_{gB} = D_{gBZ} = D_{gBY}$.

At this stage, the initial guesses for Z_0 and Y_0 have to be quite close to their target values for the algorithm to converge (typically with a similar shape): we use $Z_0^0 = Z_0^{obj} + 0.1$ and $Y_0^0 = Y_0^{obj} + 0.1$. Moreover, it is necessary to use a very precise radial threshold $\epsilon_g = 10^{-10}$ to obtain a relatively precise identification of the parameters.

Even with all the previously introduced regularisations, the calibration procedure is not yet able to converge. This is in part due to the fact that the $\kappa - \varepsilon$ solver cannot converge if Y_0 or Z_0 are negative somewhere. There is no real problem if it happens only once, but since Y_0 and Z_0 must be decreased consequently in scale, the calibration procedure often "overshoots" and reaches a negative value. This leads to multiple wasted simulations which slow down the algorithm, and most of the time to the divergence of the calibration procedure. Hence, it appears necessary to introduce an efficient way to forbid the negative values for Z_0 and Y_0.

The positive quasi linear scaling function decreases the gradient with regard to a parameter where it has a small value while keeping a linear scaling elsewhere. This is an interesting way to keep Y_0 and Z_0 positive because it allows any positive value but makes the lower orders of magnitude increasingly harder to reach. The limit for the linear behaviour is roughly equal to the scaling parameters ϵ_{s,Z_0} and ϵ_{s,Y_0}, which must be chosen carefully. Indeed, a too low ϵ_s limits the stabilising effect, but a too high one could slow down the algorithm.

We have launched multiple calibration procedures to illustrate these effects and find a good value for $\epsilon_{s;Z_0,Y_0} = \epsilon_{s,Z_0} = \epsilon_{s,Y_0}$.

The number of simulations for different values of $\epsilon_{s;Z_0,Y_0}$ are reported in Figure 6. Focusing on the solid blue curve, with our default time interval length L_T and a standard cost, we can see how the algorithm is unable to converge for $\epsilon_{s;Z_0,Y_0} \leq 5 \times 10^{-2}$, illustrating the necessity of the positive scaling. On the other hand, reducing the time interval length to focus on the beginning of the data can help to identify precisely the initial states, so that the scaling can admit a very low $\epsilon_{s;Z_0,Y_0}$. However, as we will quickly see, this shorter time interval reduces drastically the precision on the retrieved parameters, making it less desirable. All the following experiments will use the positive quasi linear scaling with $\epsilon_{s;Z_0,Y_0} = 10^{-1}$, leading to a low number of simulations for all parameters.

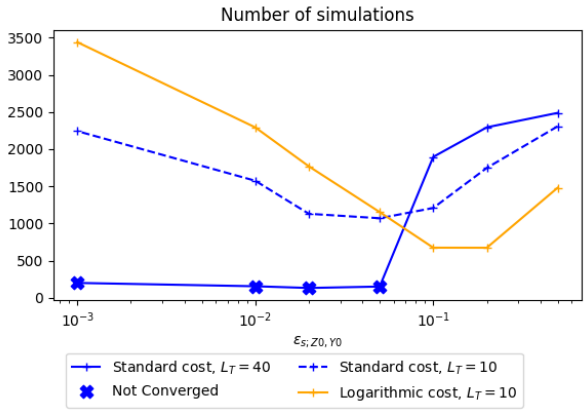


FIGURE 6 Evolution of the number of simulations with regard to the scaling parameter ϵ_s for Z_0 and Y_0 with different time interval lengths L_T and types of cost.

4.3 | Logarithmic cost

With a standard L_2 cost, the biggest errors are probably computed when the trajectories reach their maximal value, so that by comparison, the error at the beginning of the simulation is 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, 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. Figure 7 shows the evolution of multiple metrics with regard to the number of simulation for the two different types of cost and different time interval length.

Focusing first on the cost and the norm of its gradient, it clearly appears that although the value of the cost starts lower with the logarithmic cost, they finish around the same value, as the cost is dominated by the radial derivative penalisation. Unsurprisingly this is linked to a consequently smaller reduction of the gradient norm with the logarithmic cost. To avoid unnecessary iterations, the stopping threshold is raised to $\epsilon_g = 10^-7$ with the logarithmic cost. The cost without penalisation is the most meaningful representation of the accuracy of the model since it is directly representative of the distance between the generated trajectory and the data. Since the formula of the two cost types are different, we cannot directly compare their final value, but we can remark that the logarithmic cost with $L_T = 10$ reduces very efficiently the cost without penalisation. Comparing with the evolution of the relative error on the parameters, it clearly appears that the logarithmic cost with $L_T = 10$ can reach a precision at least as good as the most precise version of the standard cost, but in a fraction of the number of simulation necessary for the convergence of the other calibrations. Hence, the logarithmic cost seems very promising, but mostly with a short time interval: it seems that its efficiency relies on a fast and precise identification of the initial states Z_0 and Y_0. On the other hand, the standard cost with $L_T = 40$ appears as the only calibration where the parameters γ_Z and K are significantly improved while Z_0 is not yet precisely identified. This property might reveal very useful when the identification of the initial states is harder, for instance in the presence of noise.

4.4 | Improving radial derivative penalisation for the standard cost

At this stage we start from first guesses for Z_0 and Y_0 close to their targets. The calibration with the logarithmic cost can converge for the whole range of diffusion weights tested, but not with the standard cost. Clearly the robustness of the latter must be improved if we want to exploit its good properties of convergence without a precise identification of the initial states.

Two methods are implemented to improve the radial derivative penalisation. First, the double launch of the calibration algorithm: the minimisation routine is launched a first time with a high radial penalisation weight $w_\rho^1 = 2.5 \times 10^-5$ in order to prevent the early divergence of the algorithm. Then we launch the minimisation a second time with a lower penalisation weight $w_\rho^2 = 2.5 \times 10^-7$ starting from the parameters obtained after the first launch in order to retrieve more precise parameters. The stopping threshold for the first launch ϵ_g^1 is increased to avoid spending too much time. The second threshold is then adapted so

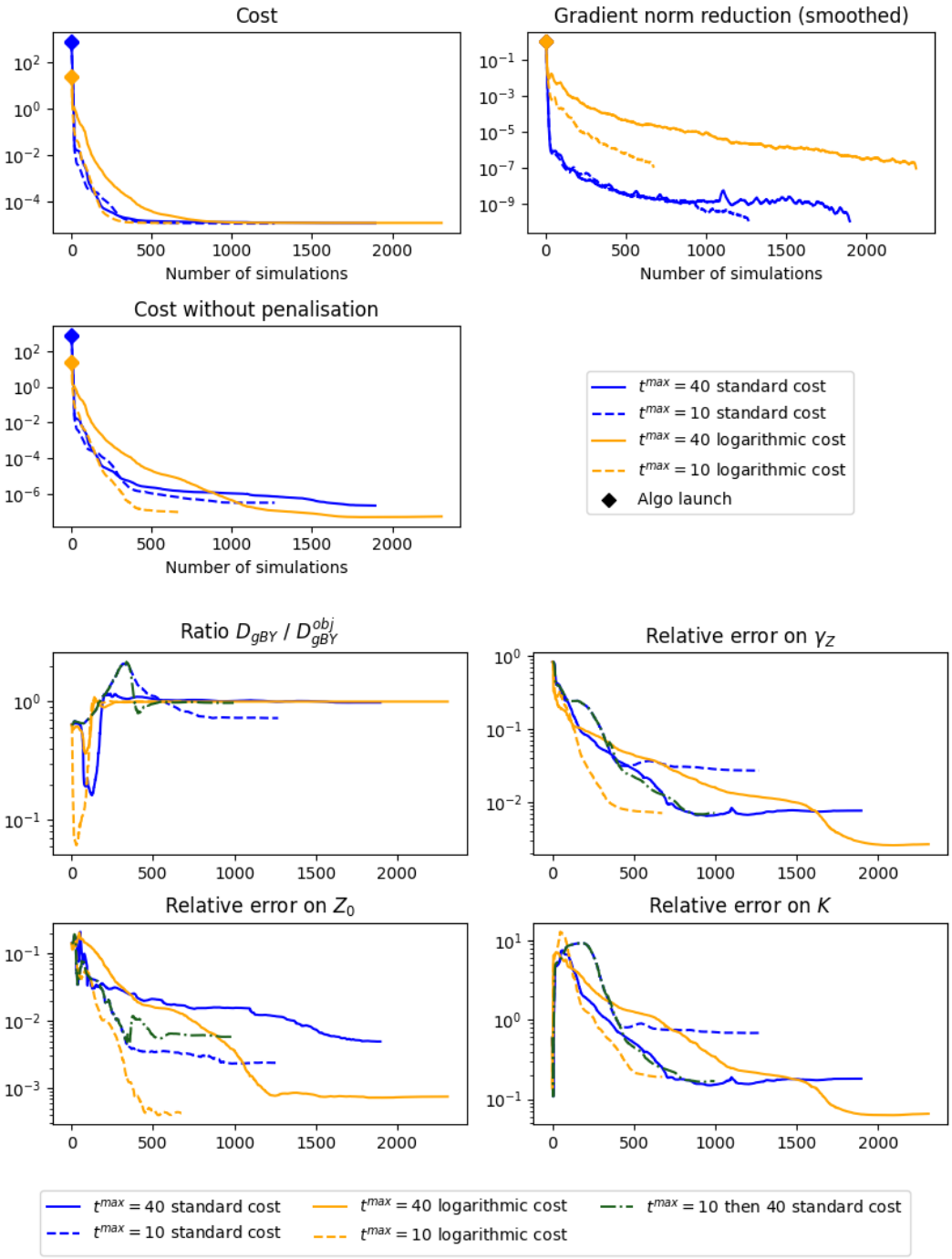


FIGURE 7 Evolution of multiple metrics for calibration with and without logarithmic cost, for time window lengths of 40 and 10 and for different D^{obj}_{gB} values. In each case, $\epsilon_{s;Z_0,Y_0} = 0.1$. The norm of the gradient is smoothed by a moving average of window 20, and "Algo launch" refers to the start of the minimisation routine (`m1qn3`), which can be launched multiple times in a row, starting from the final set of parameters of the previous launch.

that the reduction of the norm of the gradient after both iterations is equal to ϵ_g^2, which is still equal to $10^{-}10$ for standard cost and $10^{-}7$ for the logarithmic cost.

For the second improvement, since the relative evolution of the values of Z and Y is more relevant to the dynamic of the system than their absolute value, penalising the evolution of the derivative of the logarithm of Z_0 and Y_0 might be more helpful.

In the following we will refer to calibration with this modified formula for Z_0 and Y_0 only as "using the logarithmic penalisation" to contrast with the standard penalisation, although the standard formula is always used for the other parameters.

Before presenting the results with the improved penalisation for different weights, let us remark that for the lowest value of $D_{gB}^{obj} = 10^{-4}$, the exact value of the local drive parameters (γ_Z, γ_Y and K) is almost impossible to identify precisely near the core boundary ($\rho = 0$). Indeed, with a low diffusion, a homogeneous Neumann condition and a generally locally flat shape for the parameters, there is almost no movement of the variable Z and Y in this region. For a low diffusion, the region near the core boundary is then quasi-underconstrained: as can be seen in Figure 8 the values of K obtained there are all wrong while there are significant differences of accuracy everywhere else in the domain depending on the choices of cost type and time interval length.

Since the identification of the core boundary has a low impact anyway on the trajectory, we will from now on only compute the error on the parameters for $\rho > 0.3$ to have a more meaningful comparison between the accuracies of the retrieved parameters not dominated by the error on the core boundary.

Figure 9 compares the improvements made to the radial derivative penalisation with a simple increase of the penalisation weight, for different diffusion weights. First, it can be noted that the double launch often requires fewer simulations to converge than the simple launch, particularly with the logarithmic cost. Yet, the calibration algorithm converges every time and the precision of the retrieved parameter is typically almost as good as the simple launch with a low penalisation weight, with the double launch and standard cost yielding the most precise parameters. There is a notable exception for the double launch with the logarithmic penalisation at $D_{gB}^{obj} = 10^{-4}$, where the retrieved value for D_{gBY} is several orders of magnitude lower that its target value. However, since the other parameters are still decently identified, we consider the information that the calibrated D_{gBY} is largely lower than D_{gBZ} as a qualitative information sufficient to estimate that the reference configuration is generally retrieved. At this stage the double launch seems clearly beneficial, but choosing between the standard and the logarithmic penalisation appears as a trade-off between convergence rate and precision. The definitive edge will be given to the logarithmic penalisation due to the increase of robustness that we will illustrate in the next section.

4.5 | Convergence from flat initial guesses with both costs

With the improvement of the radial penalisation, the calibration with the standard cost is now considerably more robust. Indeed, simply by increasing the first penalisation weight, it is now possible to have convergent calibration starting from first guesses for the initial conditions far from their targets, $Z_0^0 = Y_0^0 = 1$.

Figure 10 displays the evolution of the cost and the cost without penalisation for the 250 first iterations of calibrations starting with $Z_0^0 = Y_0^0 = 1$. This time all the simulations are displayed, not only the ones that respected the Wolfe conditions and led to an iteration. By convention the cost is equal to -1 when the PDE solver does not converge. As can be seen, most calibrations are interrupted early, after numerous non convergence of the PDE solver. The only two converging calibrations are the curves that continue after the simulation 250, and both of them are using a logarithmic penalisation with a high first penalisation weight of at least $w_\rho^1 = 5 \times 10^{-5}$. Unfortunately, the improvements of the radial penalisation are not yet enough to converge with the logarithmic cost from flat first guesses for Z_0 and Y_0.

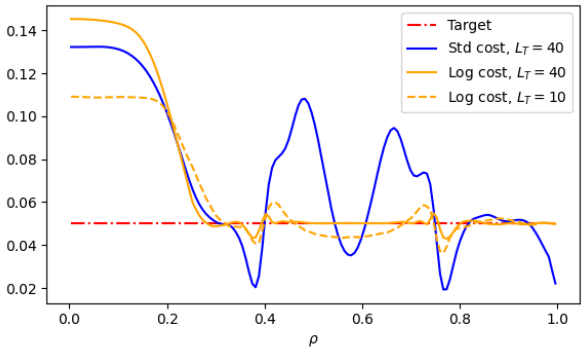
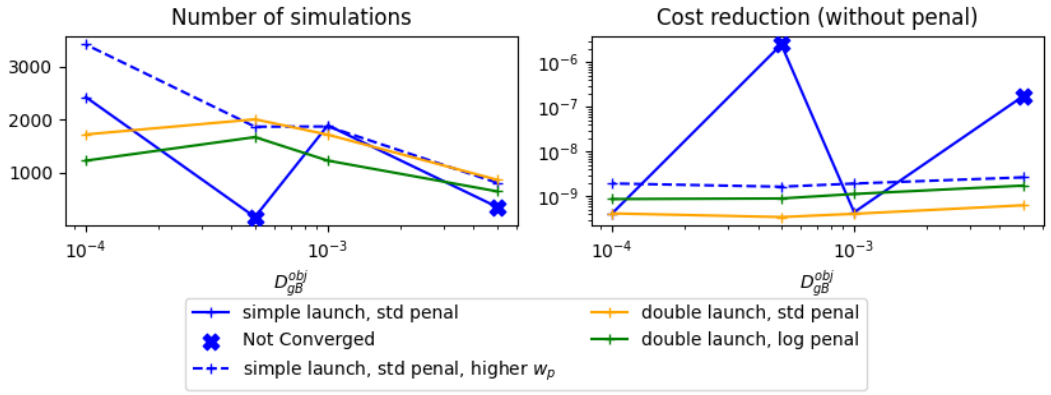
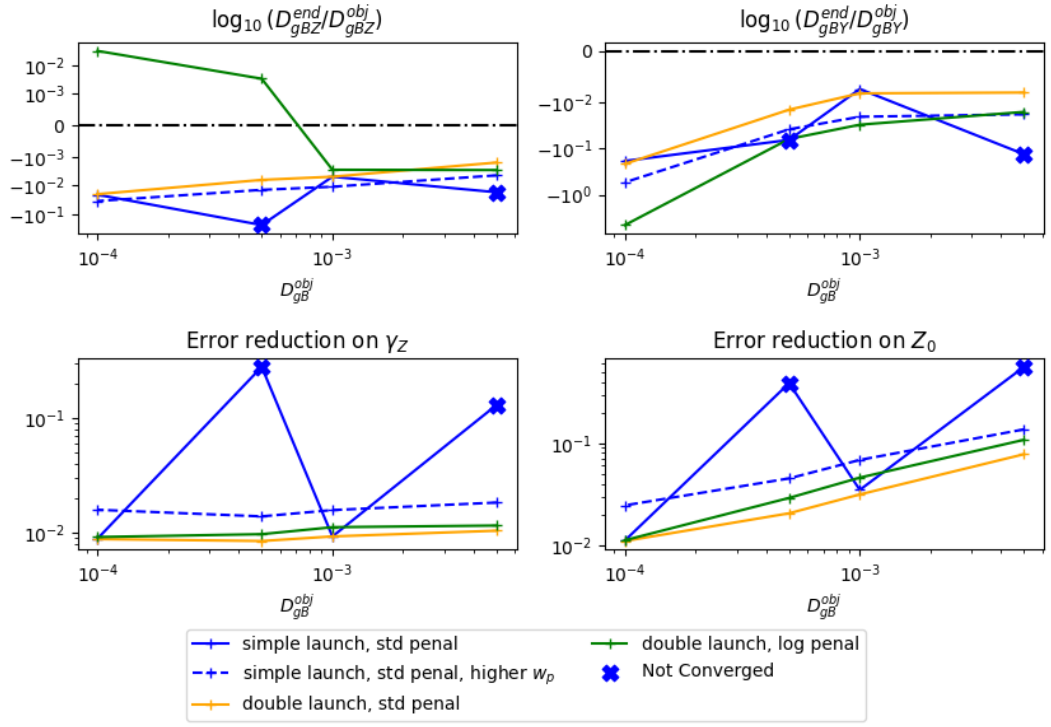


FIGURE 8 Final retrieved shapes of the parameter K for the configuration B with $D_{gB}^{obj} = 10^{-4}$, with different costs and values of L_T.



(a) Number of simulations and reduction of the cost without penalisation.



(b) Error reduction on some parameters. The relative error on γ_Z is computed for $\rho > 0.3$ because the profiles below 0.3 cannot be identified (see Figure 8).

FIGURE 9 Comparison of calibration with and without the use of a penalisation on the radial derivative of the logarithm for Z_0 and Y_0. The logarithmic penalisation is faster for an equivalent precision of the retrieved parameters and this effect cannot be reproduced by simply increasing the penalisation weights.

Since a known source of instability of the direct solver is a too large ratio Y_0/Z_0, the introduction of a penalisation of the distance between Z_0 and Y_0 may allow the calibration algorithm to avoid an error during the first iterations. This works well with the previously introduced double launch of the calibration algorithm, with a high weight $w^1_{diff,Z_0,Y_0} = 10^{-1}$ for this new penalisation in the first launch and no penalisation in the second one ($w^2_{diff,Z_0,Y_0} = 0$). We choose the simple L^2 distance to add as little non-linearity as possible.

The internal parameters for the calibration algorithm for each configuration can be seen in Table 3. The two calibrations using the states difference penalisation are identical, except for a slight modification of the value of the radial derivative penalisation weight for the first launch (w_p^1), which is increased from 2.5×10^{-5} to 5.0×10^{-5}. This is indeed similar to what was used to make the standard cost converge with $Z_0^0 = Y_0^0 = 1$ for all values of D_{gB}^{obj} (Figure 10).

	Δt	w_ρ	ϵ_g	w_{diff,Z_0,Y_0}
standard cost reference	40	2.5×10^{-5} & 2.5×10^{-7}	10^{-8} & 10^{-10}	0
std cost $Z_0^0 = Y_0^0 = 1$	40	5.0×10^{-5} & 2.5×10^{-7}	10^{-8} & 10^{-10}	0
logarithmic cost reference	10	2.5×10^{-5} & 2.5×10^{-7}	10^{-5} & 10^{-7}	0
log cost $Z_0^0 = Y_0^0 = 1$ v1	10	2.5×10^{-5} & 2.5×10^{-7}	10^{-5} & 10^{-7}	10^{-1} & 0
log cost $Z_0^0 = Y_0^0 = 1$ v2	10	5.0×10^{-5} & 2.5×10^{-7}	10^{-5} & 10^{-7}	10^{-1} & 0

TABLE 3 Internal parameters of the calibration procedure for different configurations with the reference first guesses for the initial states or the flat ones. For the cases with multiple values, the calibration algorithm is launched twice.

Figure 11 shows the number of simulations necessary to reach the stopping threshold in those different cases. In the case of the logarithmic cost, the less precise initial condition generally adds a non-negligible number of iterations. Still, the difference remains fairly manageable and the simple convergence in almost every case (with the increased first radial penalisation weight) is already a good result.

As indicated by the almost identical final cost values, the parameters retrieved using the logarithmic cost for a given value of D_{gb}^{obj} are all approximately the same when the algorithm converges. The penalisation terms used on the final launch of the algorithm are the same each time, so this seems to indicate that there is no other local minimum. Seeing the positive impact of this penalisation to prevent divergence with the logarithmic cost, it could be interesting to introduce it also with the standard cost to improve its robustness before the introduction of noise in the next section.

As it can be seen on Figure 12, the introduction of the difference of initial states penalisation clearly does not impact negatively the efficiency of the calibration algorithm with the standard cost. Indeed, even when the penalisation increases the number of simulations before convergence, it is compensated by a better reduction of the cost. Expecting that this penalisation will increase the robustness of the calibration algorithm, it will be used in the next chapter with both versions of the cost.

5 | ROBUSTNESS TEST OF AN EFFICIENT SET OF HYPERPARAMETERS

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 model trajectory. Indeed, the model is by design a simplification of the full turbulent system and there will always be noise and biases due to the measurement method. Thus, it seems important to test our calibration algorithm with data that are not full and unaltered records of Z and Y generated with the same model as the one whose parameters are fitted. For this purpose, we will first add some generated noise to the data, and then reduce the amount of information available in time.

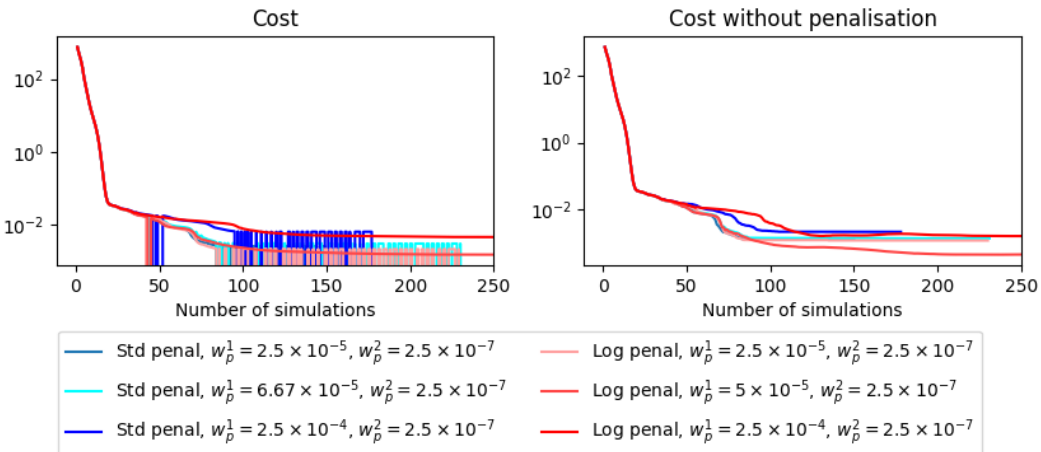


FIGURE 10 Evolution of the cost, cost without penalisation, gradient norm, and error on the parameters for calibration tests starting with Z_0 and Y_0 close to their target values or starting from $Z_0 = Y_0 = 1$. Here $D_{gB}^{obj} = 10^{-3}$, and in each case, we use the standard cost, $L_T = 40$ and the double launch.

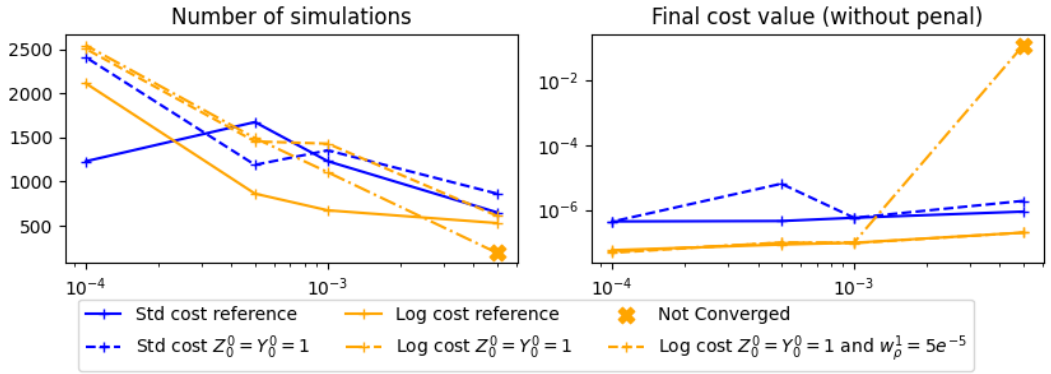


FIGURE 11 Comparison of the number of simulations necessary for convergence with the reference first guesses for the initial states $\{Z_0^0, Y_0^0\} = \{Z_0^{obj} + 0.1, Y_0^{obj} + 0.1\}$ and the flat ones $Z_0^0 = Y_0^0 = 1$. See Table 3 for the main differences of hyperparameters between the curves.

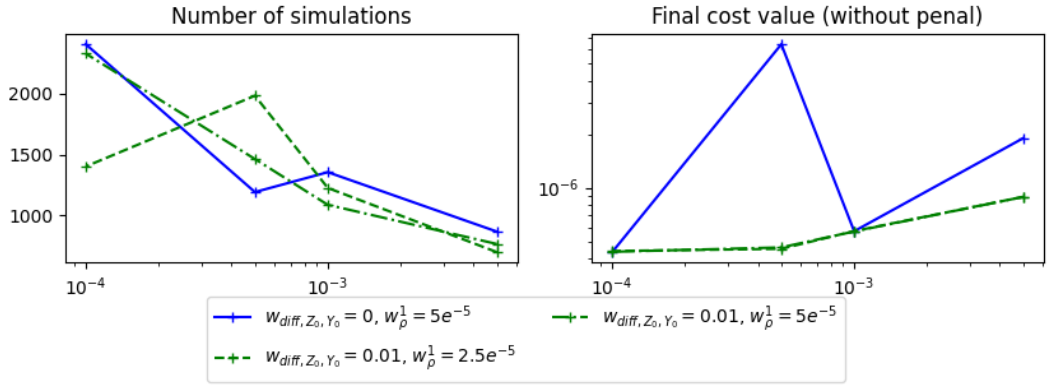


FIGURE 12 Comparison of the number of simulations necessary for convergence with or without the initial states difference penalisation (of weight w_{diff,Z_0,Y_0}) for the standard cost. Two slightly different weights for the radial derivative in the first of the two launches (w_ρ^1) are also compared. All calibration tests start with flat first guesses for the initial states: $Z_0^0 = Y_0^0 = 1$.

5.1 | Additive and multiplicative noise

In a first series of tests, we simply add Gaussian white noise to our data: a random number from the centred normal distribution of a given standard deviation is added to every radial component of every sample of data. The standard deviations σ_Z and σ_Y are adapted to each vector of data using a normalised vectorial norm:

$$\sigma_Z = \epsilon_N \sqrt{\frac{1}{N_T N_\rho} \sum_{i=0}^{N_T} \sum_{j=0}^{N_\rho} \left(Z_{i,j}^{obj}\right)^2}, \quad \sigma_Y = \epsilon_N \sqrt{\frac{1}{N_T N_\rho} \sum_{i=0}^{N_T} \sum_{j=0}^{N_\rho} \left(Y_{i,j}^{obj}\right)^2}, \tag{18}$$

for some $\epsilon_N > 0$ that we will call the relative noise amplitude. Hence, ϵ_N represents the ratio of the amplitude of the noise to the amplitude of the data: it makes sense to compare two calibration experiments toward noisy data using the same ϵ_N even if the norms of the original data matrices are different. We will also limit ourselves to noisy data with $\epsilon_N \leq 1$, as the noise is generally expected to be of lower amplitude than the data.

Since adding white noise may lead to the apparition of negative values, which are physically incoherent, we take the maximum between each scalar element of the noisy data and a chosen minimum value Z_{min} and Y_{min} (generally equal). We choose here a relatively generous value of $Z_{min} = Y_{min} = 10^{-4}$, which would require a good approximation of the expected ranges of Z

and Y since it is not very far from the minimum value of the non-altered Y^{obj} (around 10^{-3}). This was done to help the poorly performing logarithmic cost, and has almost no effect on the standard cost.

With a noise of constant standard deviation, the scale of the data can be considerably altered by the noise at points where they are initially low. This might be hard for the logarithmic cost to manage. A noise which would adapt its standard deviation depending on the value of the variable would probably be more favorable to the logarithmic cost. This is also not uncommon for the accuracies of an experimental measurement to be linked to the value of the measured quantity. For both those considerations we also introduced a multiplicative noise. With $\tilde{X}$ the noisy version of the variable ($X = Z$ or Y), and U the realisation of a vector of random variables following a centred normal distribution of standard deviation σ_X, $\tilde{X} = X \times (1 + U)$. This time the standard deviation σ_X is directly taken equal to the noise amplitude ϵ_N, since the multiplication by the data already scales the amplitude of the noise.

For the first series of experiments where we want to test multiple noise amplitudes and noise types with different cost types (standard or logarithmic), we limit ourselves to one value for the diffusion coefficients: $D_{gB}^{obj} = 10^{-3}$.

5.2 | Results

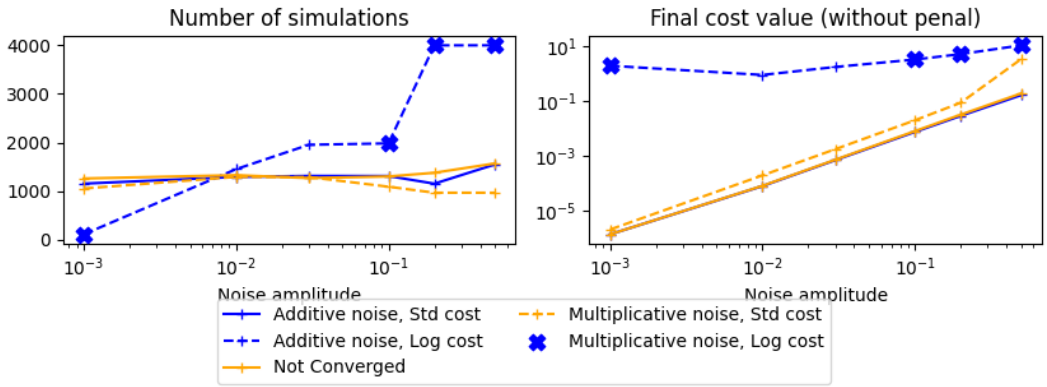


FIGURE 13 Number of simulations before convergence with additive or multiplicative noise of different amplitudes. $D_{gB}^{obj} = 10^{-3}$ for all calibrations.

Figures 13 and 14 show the results of the calibration algorithm for a growing relative noise amplitude ϵ_N. Most remarkably, the logarithmic cost version appears very unstable with the additive noise: in this case, the calibration algorithm diverges for every noise amplitude greater than or equal to $\epsilon_N = 10^{-1}$. Moreover, the logarithmic cost with added noise retrieves consequently less precise parameters and tends to quickly take more simulations to converge than the standard cost as the noise amplitude is increased. Clearly the standard cost is more adapted to the additive noise.

For the multiplicative noise, the results are less clear: both costs converge for every noise amplitude tested and for the error on the parameters the logarithmic cost is very precise with low noise, but the standard cost is better with larger cost, especially for the parameters of the local drive, like γ_Z and K in Figure 14. Since the number of simulations are remarkably independent of the noise amplitude and similar between cost and noise types, except for the logarithmic cost with additive noise, both types of costs can be relevant for the multiplicative noise.

To confirm the results, we launch calibration tests on the full range of values of D_{gB}^{obj} with a relatively high noise amplitude, and compare with calibration toward unaltered data.

Figure 15 displays the results of those experiments, in the form of the number of simulations, final cost reduction and final error on all the different parameters. Both with and without noise, the number of simulations generally quickly increases when D_{gB}^{obj} decreases. This is probably because in our reference configuration, a lower diffusion leads to less movement in the system, limiting the dependency of the trajectory to the parameters so that the latter are harder to identify. For some specific cases the number of simulation tendency is not respected, but this is likely due to the calibration reaching the stopping threshold a bit too early, potentially because of a localised drop of the norm of the gradient. Concerning the error curves, they are quite

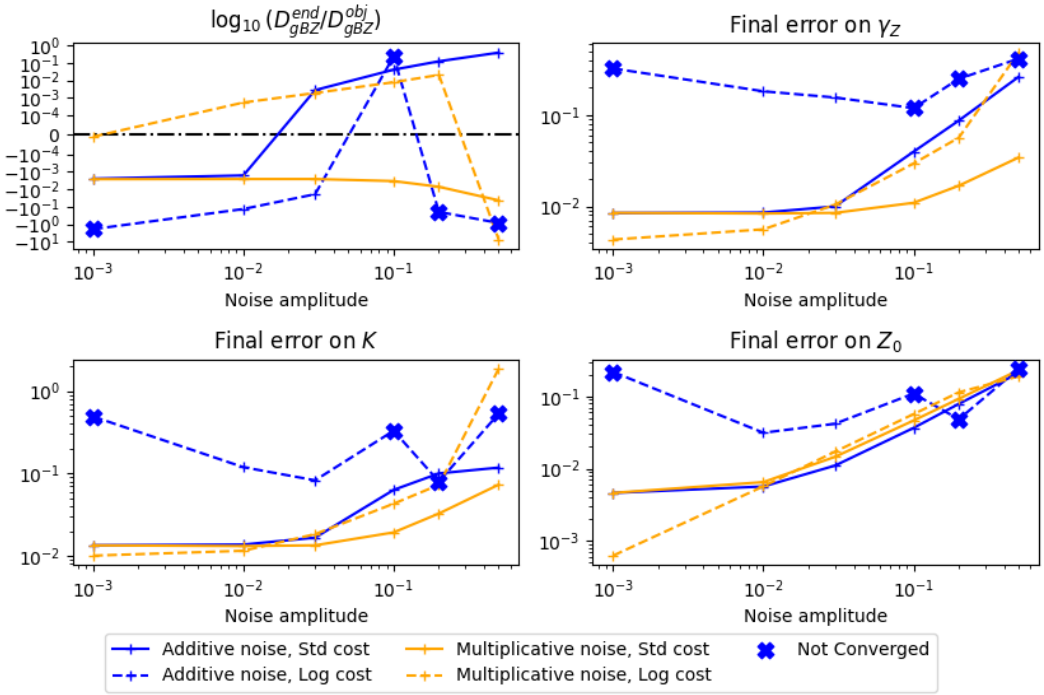


FIGURE 14 Final error (above $\rho^{err}_{min} = 0.3$) on some parameters with additive or multiplicative noise of different amplitudes. $D^{obj}_{gB} = 10^{-3}$ for all calibrations.

flat except for the diffusion weights that are generally better estimated when their targets are higher, since they have more impact on the trajectory. Interestingly, except for the initial states Z_0 and Y_0, the retrieved values with the standard cost are very close regardless of the type of noise or even the presence of noise, while the noise has a huge impact on the precision of the parameters obtained with the logarithmic cost, especially the parameters of the local drive (γ_Z, γ_Y and K). On the other hand the precision of the initial state seems very linked to the type and amplitude of the noise.

In general this confirms the results of Figures 13 and 14, the standard cost appears as the safest choice to ensure robustness and decent error reduction on the parameters for any type and amplitude of noise. In the case of a low noise dependent on the amplitude of the variables, the logarithmic noise could help to obtain more precise parameters.

5.3 | Data sparsity

As was illustrated in [27], a drawback of this calibration procedure is its inability to compensate for radial sparsity: if measurements of the variables are only available in a reduced radial region, the algorithm does not seem to be able to identify the parameters outside the region where there are data. It appears that the nonlinear diffusion does not propagate enough information for the calibration to infer the value of the variables elsewhere.

However, we can check the properties of the algorithm with time sparsity. For that we will modify the time step Δt^{obj} between two values t^{obj}_i and t^{obj}_{i+1} of the time variable corresponding to recorded vectors of data (Z^{obj}_i, Y^{obj}_i) and $(Z^{obj}_{i+1}, Y^{obj}_{i+1})$, representing Z and Y on the whole radial space.

Figures 16 and 17 show the number of simulations as well as the final errors on some retrieved parameters, with or without additive noise and different values of D^{obj}_{gB}. The tests are made with a logarithmic cost without the noise and a standard cost with the noise. Despite some divergence due to the lack of robustness of the logarithmic cost starting from flat initial condition and the difficulties to converge before the limit of 4000 simulations when $D^{obj}_{gB} = 10^{-4}$, the error on the parameters and the cost reduction are almost independent of the data time step when $\Delta t^{data} < 1$. A notable exception seems to be the case without noise and with $D^{obj}_{gB} = 10^{-4}$, where the algorithm seems to stop on an unexpected local minimum for $\Delta t^{data} \geq 0.5$.

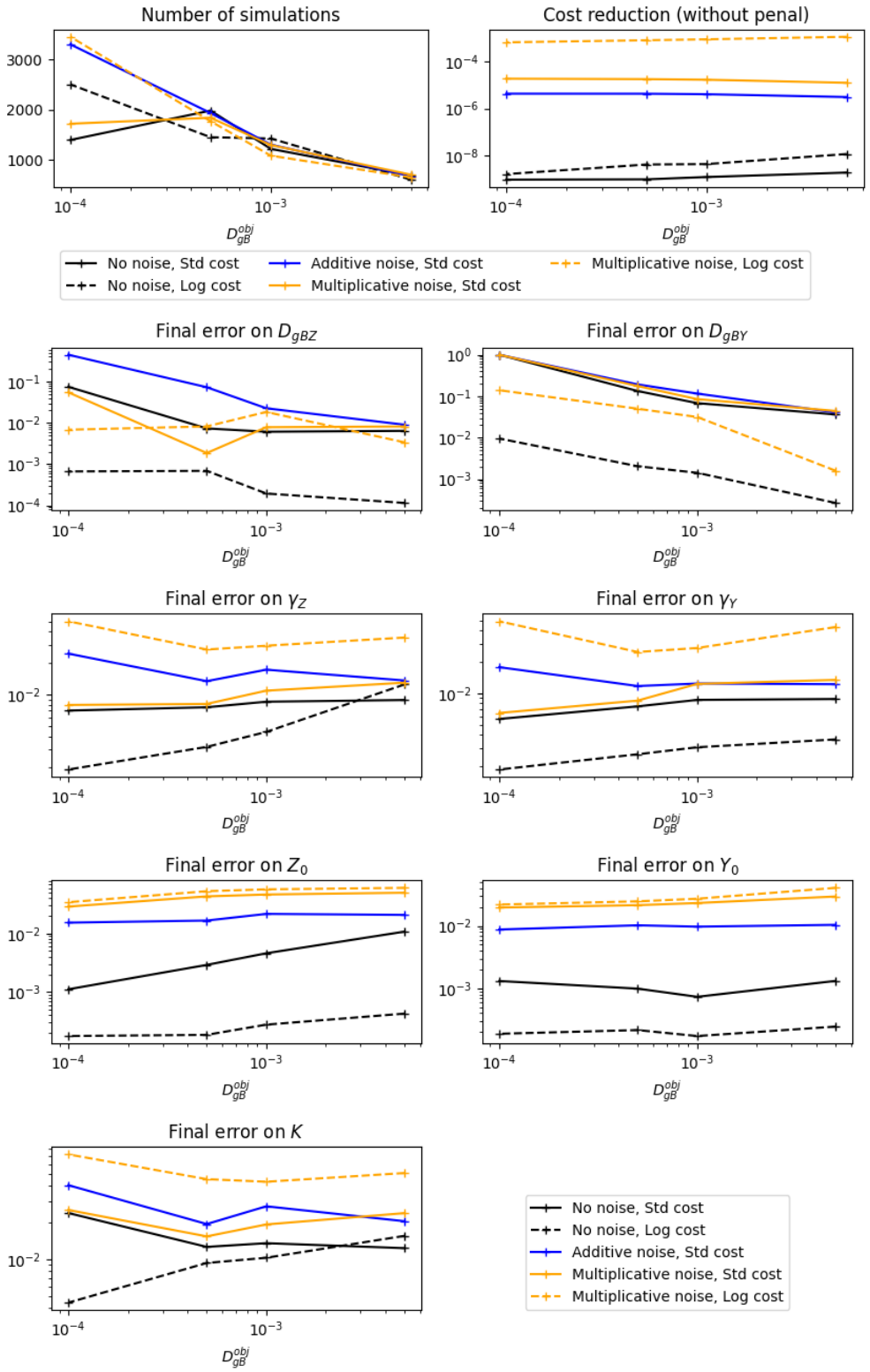


FIGURE 15 Calibration results for different types of noise added to the data, types of cost, and values of D_{gB}^{obj}. When noise is added to the target data, its relative amplitude is $\epsilon_N = 0.05$.

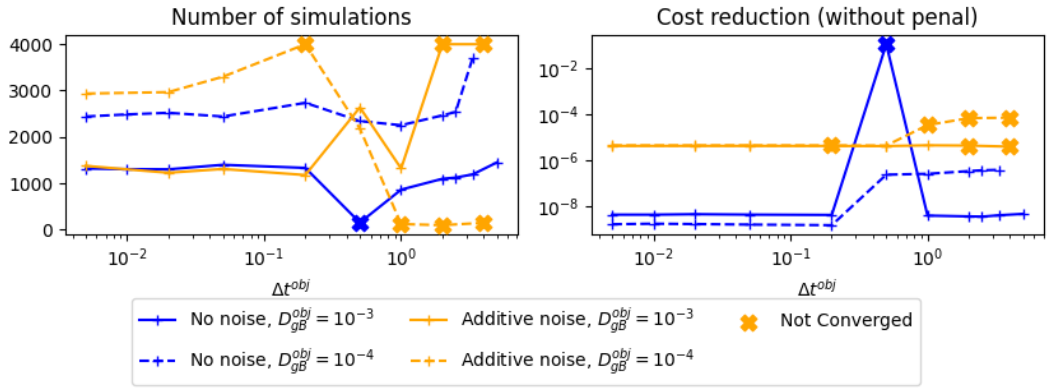


FIGURE 16 Number of iterations before convergence for different values of Δ_t^{data}, using standard cost and flat initial guesses $Z_0^0 = Y_0^0 = 1$. The amplitude of the added noise is $\epsilon_N = 0.05$.

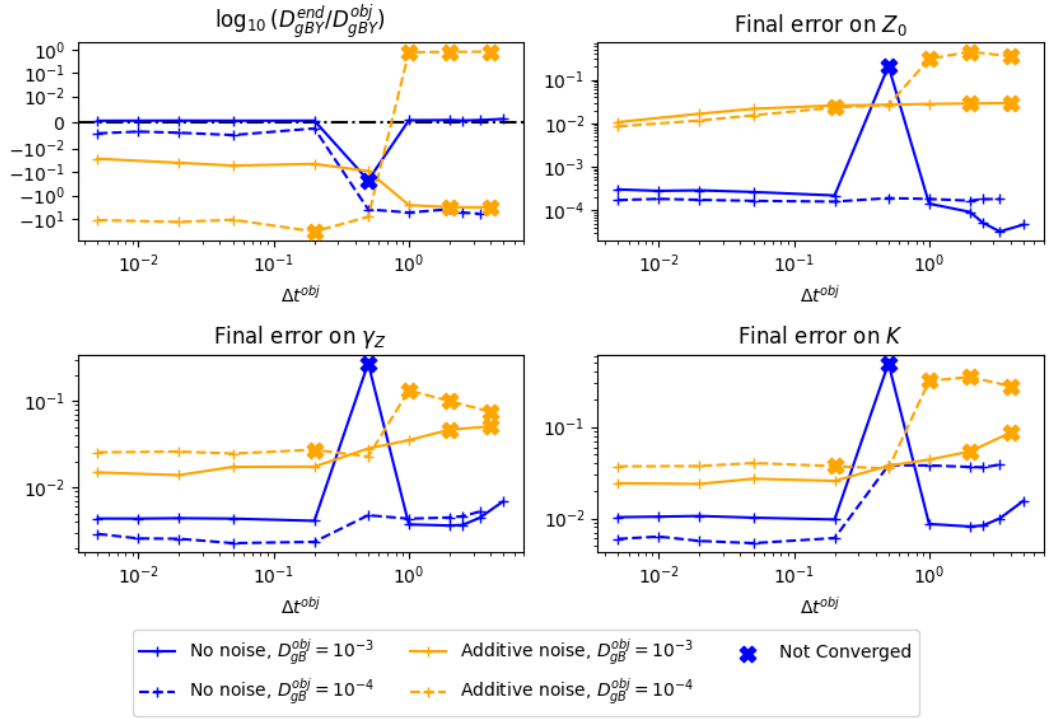


FIGURE 17 Final error (above $\rho = 0.3$) on some retrieved parameters with noisy data, for different values of Δ_t^{data}, using standard cost and flat initial guesses $Z_0^0 = Y_0^0 = 1$. The amplitude of the added noise is $\epsilon_N = 0.05$.

For the values of Δ_t^{obj} above 1 the algorithm struggles with the noisy data: it quickly diverges for $D_{gB}^{obj} = 10^{-4}$, and takes too long to converge for $D_{gB} = 10^{-3}$. Hence, in a setting with a considerable discrepancy between the data and the model, represented here by the intense noise, a too large time step between samples can clearly threaten the convergence of the algorithm. This effect is, however, dependent on the trajectory as it did not seem to happen with the simpler configuration used in [27].

Even with the presence of noise, the algorithm appears relatively robust with regard to the time sparsity, with a mostly stable number of simulations before convergence, and the error on the retrieved parameters at most slowly increasing with the data time step. However, in some cases, too large data time steps may lead to the divergence of the algorithm. Indeed, it is always

preferable to use as many samples as possible in the chosen data interval, but these tests show that the algorithm should be able to treat data without requiring a very precise measurement device.

CONCLUSION

In the context of improving the predictive capabilities of a transport code for edge plasma, an efficient and robust method appears necessary to fit the model parameters against reference data. A variational data assimilation strategy has been thoroughly tested on a simplified version of the transport model. In the framework of twin experiments, this article shows how the tuning of internal parameters 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 other transitory models. Some notable examples of successful regularisation strategies include:

- 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.
- The introduction of a penalisation on the radial derivative of the parameters improves the robustness of the algorithm.
- The rescaling of a parameter to enforce bounds can noticeably increase the robustness of the calibration procedure.
- Changing the formulas of the cost and the radial derivative penalisation for accounting for the higher dependency of the local system to relative errors rather than absolute ones can lead to a great improvement of the performances of the algorithm.
- Applying the minimisation routine twice while changing the penalisation weights between the two runs can significantly improve the robustness of the algorithm without even increasing the overall computational cost of the algorithm.
- A bilinear penalisation based on a critical relation for the stability of the underlying PDE solver can greatly increase the robustness of the calibration procedure.

Furthermore, the robustness analysis has shown both some strengths and weaknesses of the algorithm. On the one hand, 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 (see [6]), the relationship between different radii will be richer than a diffusion term, so the calibration procedure should tackle better spatial sparsity. On the other hand, the presented results show the robustness of the algorithm to time sparsity and noise, despite the weaknesses of the logarithmic version of the cost with noise independent of the local value of the data. Although there are some limits depending on the data to reproduce, the standard version of the cost seems remarkably robust, being able to converge with both intense noise and sparse data in time.

Naturally the next step which is already in progress is to test this efficient calibration procedure on a more complete version of the model. First in 1D with all the MHD equations coupled to the $\kappa - \epsilon$ system, where a comparison with real data will already be relevant. And then on the complete 2D transport version of the SolEdge3X code mettre une citation [**?**] in order to obtain a reliable and predictive plasma transport model for the design of plasma facing component and the research of optimal discharge scenarii.

REFERENCES

1. L. Aho-Mantila et al., *Outer divertor of asdex upgrade in low-density l-mode discharges in forward and reversed magnetic field: I. comparison between measured plasma conditions and solps5.0 code calculations*, Nuclear Fusion **52** (2012), no. 10, 103006.
2. M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation. Methods, Algorithms, and Applications*, *Fundamentals of Algorithms*, vol. 11, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2016, .
3. 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** (2022), no. 10, 102508.
4. R. Balescu, I. Petrisor, and M. Negrea, *Anisotropic electrostatic turbulence and zonal flow generation*, Plasma Physics and Controlled Fusion **47** (2005), 2145.
5. S. Baschetti, H. Bufferand, G. Ciraolo, N. Fedorczak, P. Ghendrih, E. Serre, and P. Tamain, *Optimization of turbulence reduced model free parameters based on l-mode experiments and 2d transport simulations*, Contr. Plasma Phys. **58** (2018), no. 6-8, 511–517.
6. 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** (2021), no. 10, 106020.
7. A. F. Bennett, *Inverse Modeling of the Ocean and Atmosphere*, Cambridge University Press, Cambridge, 2002.
8. C. Broyden, *The convergence of a class of double-rank minimization algorithms 1. general considerations*, IMA Journal of Applied Mathematics **6** (1970), no. 1, 76–90. URL https://doi.org/10.1093/imamat/6.1.76.
9. H. Bufferand et al., *Numerical modeling for divertor design of the west device with a focus on plasma wall interactions*, Nuclear Fusion **55** (2015), no. 5, 053025.

10. P. Chandramouli, E. Mémin, and D. Heitz, *4d large scale variational data assimilation of a turbulent flow with a dynamics error model*, J. Comput. Phys. **412** (2020), no. 109446.
11. F. Devernay, *C/c++ minpack*, http://devernay.free.fr/hacks/cminpack/ (2007).
12. G. Dif-Pradalier et al., *Transport barrier onset and edge turbulence shortfall in fusion plasmas*, Communications Physics **5** (2022), no. 1. URL http://dx.doi.org/10.1038/s42005-022-01004-z.
13. F.-X. L. Dimet and O. Talagrand, *Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects*, Tellus **38A** (1986), 97–110.
14. Y. Feng et al., *Recent improvements in the emc3-eirene code*, Contrib. to Plasma Phys. **54** (2014), no. 4-6, 426–431.
15. R. Fletcher, *A new approach to variable metric algorithms*, The Computer Journal **13** (1970), no. 3, 317–322. URL https://doi.org/10.1093/comjnl/13.3.317.
16. J.-C. Gilbert and C. Lemaréchal, *Some numerical experiments with variable-storage quasi-newton algorithms*, Mathematical Programming **45** (1989), 407–435.
17. J.-C. Gilbert and C. Lemaréchal, *The module m1qn3*, https://who.rocq.inria.fr/Jean-Charles.Gilbert/modulopt/optimization-routines/m1qn3/m1qn3.pdf (2009). Accessed 2021-12-01.
18. D. Goldfarb, *A family of variable metric updates derived by variational means*, Mathematics of Computation **24** (1970), 23–26.
19. R. Goldston, *Heuristic drift-based model of the power scrape-off width in low-gas-puff h-mode tokamaks*, Nuclear Fusion **52** (2011), no. 1, 013009. URL https://dx.doi.org/10.1088/0029-5515/52/1/013009.
20. A. Griewank, *Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation*, Optimization Methods and Software **1** (1992), no. 1, 35–54.
21. A. Griewank, *A mathematical view of automatic differentiation*, A. Iserles (ed.), *Acta Numerica 2003*, Cambridge University Press, Cambridge, London, New York, 2003. 321–398, .
22. L. Hascoët and V. Pascual, *The tapenade automatic differentiation tool: principles, model, and specification*, Research Report RR-7957, INRIA, 2012. URL https://hal.inria.fr/hal-00695839.
23. F. Hasenbeck, D. Reiser, P. Ghendrih, Y. Marandet, P. Tamain, A. Moller, and D. Reiter, *Multiscale modeling approach for radial particle transport in large-scale simulations of the tokamak plasma edge*, Procedia Computer Science **52** (2015), no. 10, 1128–1137.
24. M. J. Hossen, I. M. Navon, and F. Fang, *A penalized four-dimensional variational data assimilation method for reducing forecast error related to adaptive observations*, Num. Meth. Fluids **70** (2012), no. 10, 1207–1220.
25. *Iter official website*, https://www.iter.org/ (2023).
26. E. Kalnay, *Atmospheric modeling, data assimilation and predictability*, Cambridge University Press, Cambridge, 2003.
27. L. Lamerand, D. Auroux, P. Ghendrih, F. Rapetti, and E. Serre, *Inverse problem for determining free parameters of a reduced turbulent transport model for tokamak plasma*, Advances in Computational Mathematics **50** (2024), no. 39.
28. B. Launder and D. Spalding, *The numerical computation of turbulent flows*, Comput. Methods Appl. Mech. Eng. **3** (1974).
29. J.-L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer Berlin, Heidelberg, 1971.
30. A. Loarte et al., *Power and particle control*, Nuclear Fusion **47** (2007), no. 6, S203.
31. K. Miki, P. H. Diamond, L. Schmitz, D. C. McDonald, T. Estrada, O. D. Gurcan, and G. R. Tynan, *Spatio-temporal evolution of the $H \rightarrow L$ back transition*, Phys. of Plasma **20** (2013), 062304.
32. Y. Nishimura, K. Borrass, D. Coster, and B. Scott, *Effects of resistive drift wave turbulence on tokamak edge transport*, Contr. Plasma Phys. **44** (2004), no. 1-3, 194–199.
33. S. Pope, *Turbulent flows*, Cambridge University Press, Cambridge, 2000.
34. G. Qin and A. Shalchi, *The role of the Kubo number in two-component turbulence*, Physics of Plasmas **20** (2013), no. 9, 092302. URL https://doi.org/10.1063/1.4821026.
35. A. Quarteroni and A. Valli, *Numerical Approximation of Partial Different Equations*, vol. 23, Springer, 1994.
36. P.-A. Raviart and J.-M. Thomas, *Introduction to the numerical analysis of the PDEs*, Masson, 1983.
37. T. Rognlien, J. Milovich, M. Rensink, and G. Porter, *A fully implicit, time dependent 2-D fluid code for modeling tokamak edge plasmas*, J. Nucl. Mat. **196-198** (1992), 347–351. Plasma-Surface Interactions in Controlled Fusion Devices.
38. Y. Sarazin, *Étude de la turbulence de bord dans les plasmas de tokamaks*, Ph.D. thesis, Joseph Fourier University (Grenoble, France ; 1971-2015), 1997. URL http://www.theses.fr/1997GRE10265.
39. F. Schwander, E. Serre, H. Bufferand, G. Ciraolo, and P. Ghendrih, *Global fluid simulations of edge plasma turbulence in tokamaks: a review*, Computers & Fluids **270** (2024), 106141. URL https://www.sciencedirect.com/science/article/pii/S0045793023003663.
40. D. F. Shanno, *Conditioning of quasi-newton methods for function minimization*, Mathematics of Computation **24** (1970), 647–656.
41. G. R. Tynan et al., *Recent progress towards a physics-based understanding of the h-mode transition*, Plasma Phys. and Control. Fusion **58** (2016), no. 4.
42. S. Wiesen et al., *The new SOLPS-ITER code package*, J. Nucl. Mater. **463** (2015), no. 5, 480–484.