Algorithmic Differentiation for adjoint sensitivity calculation in plasma edge codes

Stefano Carli, Laurent Hascoët, Wouter Dekeyser, Maarten Blommaert

Algorithmic Differentiation for adjoint sensitivity calculation in plasma edge codes

Stefano Carli[1], Laurent Hascoët[2], Wouter Dekeyser[1], and Maarten Blommaert[3]

[1]KU Leuven, Department of Mechanical Engineering, Celestijnenlaan 300, 3001 Heverlee, Belgium
[2]INRIA Sophia-Antipolis, Route des lucioles 2004, 06902 Valbonne, France
[3]KU Leuven, Department of Mechanical Engineering, Kleinhoefstraat 4, 2440 Geel, Belgium

August 3, 2023

Abstract

In the framework of fusion energy research, divertor design and model calibration based on plasma edge codes currently rely either on manual iterative tuning of parameters, or alternatively on parameter scans. This, combined with the complex and computationally expensive nature of plasma edge codes, makes these procedures extremely cumbersome. Gradient-based optimization methods can significantly reduce this effort, but require an efficient strategy for sensitivity calculation. Algorithmic Differentiation (AD) offers an efficient and accurate solution, allowing semi-automatic sensitivity computation in complex, continuously developed codes. The adjoint AD mode is especially attractive, as its cost is independent of the number of input parameters. In this paper, adjoint AD sensitivity calculation is deployed for the first time in plasma edge codes, applying the TAPENADE AD tool to SOLPS-ITER. Adjoint AD results are verified to be machine precision accurate compared to tangent AD mode, and up to 10^{-9} compared to finite differences. Scalings of AD computational efforts prove the advantages of adjoint compared to tangent AD, while memory requirements rapidly increase for adjoint, showing the need for an improved checkpointing strategy. With the new tool, a wealth of information becomes accessible to the modeler. An adjoint sensitivity analysis for a COMPASS density scan identifies the input parameters with largest impact on the solution, and 2D sensitivity maps show their spatial dependence.

1 Introduction

Nuclear fusion has the potential for a sustainable, large scale electricity generation, thanks to its near-zero CO_2 emissions and a cheap, widely available, and almost unlimited fuel. However, key physical and technological challenges remain to be dealt with to achieve reliable power generation in magnetically confined fusion devices. Among these challenges stands the reliable power and particle exhaust through the divertor component [1]. This component must handle heat loads of 10 MW/m^2 and beyond in normal steady-state operation, together with fluxes of highly energetic particles. This poses severe challenges on divertor lifetime in terms of surface melting, material erosion, and cooling capabilities. Therefore, accurate assessment of design and operational scenarios employing plasma edge simulations is required.

Plasma edge simulations are typically carried out with mean-field codes such as SOLPS-ITER [2,3], SOLEDGE2D [4], UEDGE [5], and EDGE2D [6]. Especially SOLPS-ITER has been extensively used for designing and assessing the ITER divertor [7–9], it is routinely used for analysis of present-day fusion devices [10–12], and is currently supporting the design of next-step fusion reactors like DEMO [13]. SOLPS-ITER is a complex and computationally expensive multi-physics code. The plasma transport

in the magnetic field is described by fluid equations based on the Braginskii equations [14]. Interactions with neutral particles and solid walls, strongly affecting the plasma edge behavior, are accounted for by coupling to a neutral transport model, solving either fluid or kinetic equations [15, 16] and extensively relying on reaction rates and reflection databases.
Divertor design and model calibration is currently mostly based on expensive parameter scans or manual iterative procedures [17]. The high computational cost of plasma edge codes combined with many input parameters or design variables, makes these procedures excessively demanding. To alleviate this problem, an efficient adjoint-based optimization strategy has been recently developed for a simplified plasma edge model [18–20]. In this approach, sensitivities of the cost functional are efficiently computed using the (continuous) adjoint equations, at a cost comparable to only a few plasma edge simulations. Moreover, the availability of adjoint sensitivities paves the way towards sensitivity-accelerated Uncertainty Quantification (UQ) methods [21], which are indispensable to reduce the otherwise unfeasible burden of sampling-based UQ methods for thorough model validation with complex and expensive plasma edge codes.
Implementation of the adjoint equations in state-of-the-art plasma edge codes like SOLPS-ITER, to enable optimization and UQ methodologies, is however not straightforward. In fact, these are complex and well-established codes, continuously developed by scientists spread across the globe. As such, analytical derivation and manual implementation of (continuous) adjoint equations is practically unfeasible and prone to programming errors. In this context, Algorithmic Differentiation (AD) offers a viable alternative, allowing semi-automatic sensitivity calculation in computer codes [22].
We have recently applied AD for the first time in the field of nuclear fusion to SOLPS-ITER [23]. However, so far we exploited only the *tangent* mode of AD. In this paper, we apply *adjoint* AD to SOLPS-ITER, providing adjoint sensitivity analysis for the first time in state-of-the-art plasma edge codes.
The remainder of the paper is organized as follows. In section 2 we summarize the main features of AD, and in section 3 we introduce the modeling approach of SOLPS-ITER and the AD strategies adopted. In section 4 we prove the accuracy of adjoint AD sensitivities. Next, we assess AD efficiency on SOLPS-ITER in section 5, providing scalings of the required computational effort. Finally, in section 6 we carry out an adjoint-sensitivity analysis for a density scan and results are discussed.

2 Algorithmic Differentiation

Algorithmic Differentiation, also known as Automatic Differentiation, is a methodology for computing derivatives of numerical models, within floating-point accuracy. The first tangent AD applications date back to the 1960s [24, 25], and the adjoint mode was developed by several scientists in the following decades [26–28]. Hand-in-hand with scientific computing, AD developments continued and came within reach of several scientific domains, such as aerodynamic design [29], ice sheet modeling [30], and coil design for nuclear fusion stellarators [31]. We give next a general overview of AD and its main features, based on the thorough analysis available in [22].
Consider a function $\mathcal{F} : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^m$, mapping the input vector $X \in \mathcal{D} \subset \mathbb{R}^n$ into the output vector $Y \subset \mathbb{R}^m$. The computational model evaluating $\mathcal{F}$ is often referred to as the *primal*. The basic principle of AD is that the primal of $\mathcal{F}$ is a chain of several elementary operations φ_i, for which the derivative is simple and well-known. These elementary operations are the program instructions implemented in the primal. Evaluation of $\mathcal{F}(X)$ can thus be written as a composition of p elementary operations

$$Y = \mathcal{F}(X) = \varphi_p \circ \varphi_{p-1} \circ \cdots \circ \varphi_1(X) = \varphi_p(V_{p-1}) \times \varphi_{p-1}(V_{p-2}) \times \cdots \times \varphi_1(X), \tag{1}$$

where φ_i are possibly nonlinear operators, and $V_i = \varphi_i(V_{i-1})$ for all $i > 0$ are the successive vectors of intermediate variables as evaluation proceeds, with $V_0 = X$. To evaluate the Jacobian $\mathcal{F}' : \mathcal{D} \subset \mathbb{R}^n \to$

$\mathbb{R}^{m\times n}$ one can straightforwardly apply the chain rule

$$\mathcal{F}'(X) = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} = \big(\varphi_p' \circ \varphi_{p-1} \circ \cdots \circ \varphi_1(X)\big) \times \big(\varphi_{p-1}' \circ \varphi_{p-2} \circ \cdots \circ \varphi_1(X)\big) \times \cdots \times \varphi_1'(X) =$$

$$= \varphi_p'(V_{p-1}) \times \varphi_{p-1}'(V_{p-2}) \times \cdots \times \varphi_1'(X), \quad (2)$$

where φ_k' are the Jacobian matrices of the elementary operations. Note that each $\times$ operation is an expensive matrix-matrix multiplication. AD aims at evaluating a cheaper version of Eq. 2, where matrix-vector operations are employed.
For example, multiplying Eq. 2 to its right by a perturbation direction vector $\dot{X}$, one obtains the *tangent* or *directional derivative* vector

$$\dot{Y} = \mathcal{F}'(X) \times \dot{X} = \dot{\mathcal{F}}(X, \dot{X}) = \varphi_p'(V_{p-1}) \times \varphi_{p-1}'(V_{p-2}) \times \cdots \times \varphi_1'(X) \times \dot{X}. \quad (3)$$

In Eq. 3 going from right to left the $\times$ operations are cheaper matrix-vector multiplications. In particular, choosing $\dot{X}$ as e_i, the i^{th} component of the Cartesian basis of the input space, provides $\dot{Y} = \mathcal{F}'(X) \times e_i = \begin{pmatrix} \frac{\partial y_1}{\partial x_i} & \frac{\partial y_2}{\partial x_i} & \cdots & \frac{\partial y_m}{\partial x_i} \end{pmatrix}^T$. The resulting $\dot{Y}$ stores an entire column of the Jacobian. In other words, $\dot{Y}$ contains the sensitivity of each output with respect to a single input i. The tangent mode of AD effectively implements Eq. 3, and the notation $\dot{\mathcal{F}}$ indicates the tangent function or program. This AD mode is also referred to as *forward* mode, as perturbations are propagated from the input towards the output of $\mathcal{F}$, in the same order as the primal. Clearly, evaluation of the full Jacobian requires repeating Eq. 3 while perturbing separately each element in $\dot{X}$, for a total of n evaluations. This AD mode for obtaining the Jacobian is known as the tangent *vector* mode. This mode entails evaluating the Jacobian in a column-by-column fashion, with a computational cost proportional to the input dimension. The cost of tangent AD is thus about the same as that of finite differences (FD), with the advantageous absence of truncation error. For practical purposes, tangent AD is very efficient when the function $\mathcal{F}$ has many outputs and few inputs ($m \gg n$), in which case the Jacobian is "tall". Indeed, by careful exploitation of vectorial quantities and sparsity, the computational time for evaluating the full Jacobian is bounded by

$$COST\left\{\mathcal{F}'(X)\right\}_{TGT} \leq (1 + 1.5\tilde{n})\ COST\left\{\mathcal{F}(X)\right\}, \quad (4)$$

where $\tilde{n} < n$ is a complexity measure of the domain of $\mathcal{F}$ [22]. The memory required by tangent AD roughly follows the same proportionality.
Conversely, Eq. 2 can be multiplied on the left by any vector $\overline{Y}^T$, dimensioned after the space of outputs, which can be seen as a set of weights attached to each output defining a single scalar composite output. This yields the *gradient* or *adjoint directional derivative* vector

$$\overline{X} = \overline{Y}^T \times \mathcal{F}'(X) = \overline{\mathcal{F}}(X, \overline{Y}) = \overline{Y}^T \times \varphi_p'(V_{p-1}) \times \varphi_{p-1}'(V_{p-2}) \times \cdots \times \varphi_1'(X). \quad (5)$$

In this case, the cheaper matrix-vector multiplications are retrieved in Eq. 5 going from left to right. In particular, choosing $\overline{Y}$ as e_j, the j^{th} component of the Cartesian basis of the output space, provides $\overline{X} = \begin{pmatrix} \frac{\partial y_j}{\partial x_1} & \frac{\partial y_j}{\partial x_2} & \cdots & \frac{\partial y_j}{\partial x_n} \end{pmatrix}$. The result $\overline{X}$ is an entire row of the Jacobian, storing the sensitivity of a single output j with respect to all inputs. Because of the equivalence $\overline{Y}^T \times \mathcal{F}'(X) = \mathcal{F}^*(X) \times \overline{Y}$, in which $\mathcal{F}^*(X)$ is the Hermitian transpose or adjoint of the Jacobian, the AD mode implementing Eq. 5 is referred to as adjoint mode, with $\overline{\mathcal{F}}$ indicating the adjoint program. It is also referred to as *reverse* AD, as perturbations are propagated from the output towards the input of $\mathcal{F}$, in reverse order compared to the primal. Symmetrically to tangent AD, in adjoint AD the Jacobian is evaluated row-by-row, repeating Eq. 5 for each of the m elements in Y, in the adjoint *vector* mode. Therefore, the

computational cost is independent of the input dimensionality, efficiently evaluating a "fat" Jacobian ($n \gg m$). This is the major advantage of adjoint AD and of analytical adjoint equations. In several optimal design and engineering applications such as shape and topology optimization, with few outputs and up to millions of design variables, adjoint (AD) is the only viable option to obtain gradients. In fact, an upper bound estimate of adjoint AD computational time is given by

$$COST\left\{\mathcal{F}'(X)\right\}_{ADJ} \leq (1.5 + 2.5\tilde{m})\ COST\left\{\mathcal{F}(X)\right\}, \tag{6}$$

where $\tilde{m} < m$ is a complexity measure of the range of $\mathcal{F}$ [22]. Regarding adjoint AD memory requirements, these highly depends on the AD implementation, as will be elucidated in the next section.

2.1 AD implementations

The mechanical processing of the primal code and derivation of the tangent/adjoint instructions is automatized by AD tools, for which an extensive list is available on the AD community website for different programming languages [32]. Irrespective of the language, AD tools can be grouped into *source transformation* and *operator overloading* tools. The difference between them lies in the computer science techniques employed to provide derivative instructions.

Source transformation tools like TAPENADE, ADIFOR, and TAF [33–35] parse source files of a program and internally build a representation of it. Next, they transform such internal representation, producing a new set of source files containing the supplementary derivative information, also called the *differentiated code*. In this perspective, they can be seen as an additional preprocessing step before compilation. Source transformation tools are complex and require a costly development, similar to the cost of developing a compiler. In addition, the AD model they implement is more sophisticated for adjoint mode. In tangent mode, derivative instructions require intermediate variables with same ordering as the primal, so that the AD tools write tangent instructions between the original ones. In adjoint mode, intermediate variables are instead required in reverse order. Thus, the computation is split into two phases: the *forward* sweep, which evaluates the primal, and the *backward* sweep, which evaluates the adjoint. Intermediate variables through the backward sweep need to be retrieved if they have been overwritten, either by recomputing or storing them. The first option leads to a higher computational time, while the second option increases memory requirements (thus an estimate on the memory bound for adjoint mode was previously not given). The optimal choice is case-dependent, and a balanced strategy between the two approaches, known as *checkpointing*, provides the best results [36]. From a user point of view, the advantage is that the backward sweep is fully available as a code and can be tailored to specific needs.

The other group of AD tools, operator overloading tools such as CoDiPack and ADOL-C [37, 38], essentially consist of a library that is compiled together with the primal code. This library augments the primal variables' types with new types containing the sensitivities. At the same time, the library overloads the elementary operations to deal with the new variable type and propagate the sensitivities according to the chain rule. In forward mode for example, intermediate variables v_i are augmented to $\tilde{v}_i \equiv (v_i, \dot{v}_i)$, while the product elementary operation "$*$" is overloaded such that $\tilde{v} \tilde{*} \tilde{u} \equiv (v * u, u * \dot{v} + v * \dot{u})$. The advantage of these AD tools is that the core of the primal code remains essentially untouched, with only few simple instructions to be inserted by the user. On the other hand, their model for adjoint AD is radically different: the augmented program fills a so-called tape of all intermediate operations and operands pointers, and at the end reads and interprets this tape in the reverse direction to propagate the adjoint sensitivities. The storage cost of this model is obviously quite high and is typically a factor 10 larger compared to source transformation AD [39]. Moreover, the backward sweep is hardly accessible by the user and cannot be modified.

3 AD for SOLPS-ITER

In this section, we give the specific details of adjoint AD implementation in SOLPS-ITER using the TAPENADE [33] tool, which we employed in a previous work to retrieve tangent derivatives [23]. We chose this source transformation tool to keep memory requirements low and for the flexibility it allows in fine-tuning the differentiated code. The optimal performance of AD-differentiated codes highly depends on user's choices and insights. Knowledge of the underlying strategies for solving the primal code is vital, and allows fine-tuning the differentiated code, instead of simply employing it as a black-box. Therefore, we first start presenting the plasma edge modeling approach of SOLPS-ITER, and then describe the AD strategies adopted.

3.1 Plasma edge modeling approach

In the most common plasma edge modeling application, one models a *tokamak*, a fusion reactor where the superposition of toroidal and poloidal magnetic fields forms a helical field which confines the hot plasma (ions and electrons) in nested torus-shaped flux surfaces, as the one shown in Fig. 1a. However, due to turbulence and collisions, plasma particles still escape this confinement, and need to be safely exhausted. This is achieved through the divertor magnetic geometry, as shown in Fig. 1b. In this configuration, a magnetic null is formed, the so-called X-point, which is intersected twice by the last closed flux surface, or *separatrix*. Within the separatrix, magnetic flux surfaces are closed, confining the *core* plasma. Outside of the separatrix, flux surfaces are open and intersect solid structures called divertor *targets*, which concentrate plasma-surface interactions away from the core. The region of open magnetic flux surfaces is also called the Scrape Off Layer (SOL). The SOL, together with a small core region near the separatrix, and the private flux region (PFR) below the X-point, form the plasma edge. The plasma transport in such magnetic geometry is extremely fast (order of sound speed) along the magnetic field lines, the so-called parallel direction, as charged particles are almost unimpeded along this direction. On the other hand, turbulent and collisional transport across the magnetic field, in the radial direction, is rather slow. SOLPS-ITER models multi-species transport of ions and electrons in the plasma edge based on a set of coupled, nonlinear, partial differential equations (PDEs) based on the Braginskii equations [14], closely resembling the Navier-Stokes ones. For ease of notation, we provide hereafter a general form of the governing equations valid in any geometry, while a detailed description is available in [40, 41]. In particular, SOLPS-ITER assumes toroidal symmetry, such that the following equations are rewritten and discretized in the 2D radial-poloidal plane of Fig. 1b.

$$\frac{\partial n_a}{\partial t} + \nabla \cdot (n_a \boldsymbol{V}_a) = S_{n,a}, \tag{7}$$

$$m_a \frac{\partial n_a \boldsymbol{V}_a}{\partial t} + \nabla \cdot (m_a n_a \boldsymbol{V}_a \boldsymbol{V}_a) = -\nabla p_a - \nabla \cdot \boldsymbol{\pi}_a + Z_a e n_a (\boldsymbol{E} + \boldsymbol{V}_a \times \boldsymbol{B}) - \boldsymbol{R} + \boldsymbol{S}_{m,a}, \tag{8}$$

$$0 = -\nabla p_e - e n_e (\boldsymbol{E} + \boldsymbol{V}_e \times \boldsymbol{B}) + \boldsymbol{R}, \tag{9}$$

$$\frac{3}{2} \frac{\partial n_e T_e}{\partial t} + \nabla \cdot \boldsymbol{q}_e = -e n_e \boldsymbol{E} \cdot \boldsymbol{V}_e - \boldsymbol{R} \cdot \boldsymbol{V}_e + Q_\Delta + S_{h,e}, \tag{10}$$

$$\frac{3}{2} \frac{\partial n_i T_i}{\partial t} + \nabla \cdot \left(\boldsymbol{q}_i + \sum_a \boldsymbol{\pi}_a \cdot \boldsymbol{V}_a \right) = \sum_a (Z_a e n_a \boldsymbol{E} - \boldsymbol{R}) \cdot \boldsymbol{V}_a - Q_\Delta + S_{h,i}, \tag{11}$$

$$\nabla \cdot \boldsymbol{j} = 0. \tag{12}$$

The first equation ensures the continuity for each ionized species a, namely each charged state of an element contained in the plasma. In Eq. 7, n_a is the ion particle density, $\boldsymbol{V}_a$ is the ion velocity vector, and the term $S_{n,a}$ includes sources and sinks of particles due to collisional processes between plasma and neutral particles, as well as plasma-wall interactions.

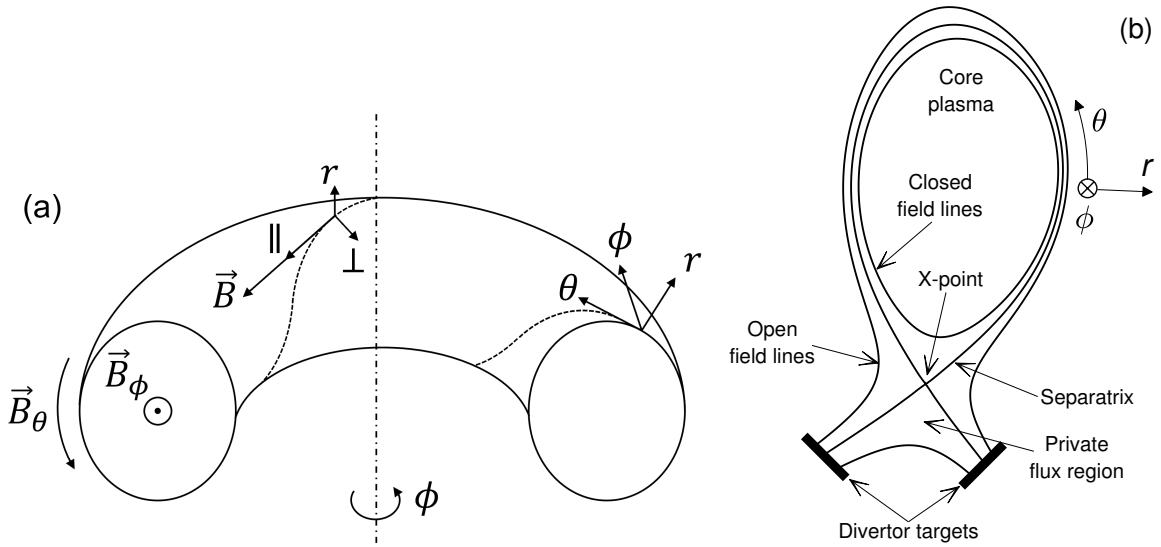

Figure 1: Sketch of the toroidal magnetic field geometry in a tokamak (a), and of the divertor geometry (b). The dashed line represents an helical magnetic field line, which creates a flux surface by revolving around the toroidal ϕ direction. The symbol θ indicates the poloidal, r the radial, $\parallel$ the parallel, and $\perp$ the perpendicular directions.

The momentum conservation for each species is ensured through Eq. 8, where m_a is the ion mass and the partial pressure for ion a is defined as $p_a = n_a T_i$. Note that all ions are assumed to share the same temperature T_i[1]. Moreover, $\boldsymbol{\pi}_a$ is the viscosity tensor. The third term on the r.h.s. of Eq. 8 is peculiar of plasmas and represents the Lorentz force acting on charged particles due to the electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{B}$, where e is the elementary charge and Z_a is the charge state. The term $\boldsymbol{R}$ lumps the momentum transfer between ions and electrons. The last term, $\boldsymbol{S}_{m,a}$, describes again sources and sinks due to plasma-neutral and plasma-wall interactions. The component of the momentum equation parallel to the magnetic field is solved for the parallel ion velocity $u_{\parallel a}$. Due to the highly anisotropic transport, the components of the equation perpendicular to the field require special treatment, and give rise to so-called plasma drifts perpendicular to the field. For details, we refer to [40].
The momentum balance for electrons is shown in Eq. 9, where $p_e = n_e T_e$ is the electron pressure and $\boldsymbol{V}_e$ is the electron velocity. This equation is noticeably simpler compared to the ion momentum equation, as terms proportional to the very small electron mass are neglected. When further elaborating the equation, with the assumption of quasi-neutrality enforcing the local balance $n_e = \sum_a Z_a n_a$, the parallel component of the electron momentum equation is simply evaluated to obtain the parallel current $j_\parallel$, while the perpendicular components give rise to electron drifts.
Next, the two equations 10 and 11 impose conservation of internal energy for electrons and ions, respectively. In there, $n_i = \sum_a n_a$ is the total ion density, $\boldsymbol{q}_{e/i}$ represent the heat fluxes, including both conduction and advection terms. Q_Δ is the ion-electron temperature equilibration term, while $S_{h,e/i}$ lump all energy sources and sinks due to radiative processes, plasma-neutral, and plasma-wall interactions.
Last, Eq. 12 imposes charge conservation, determining the plasma potential ϕ, and in turn the electric field. The current is defined as $\boldsymbol{j} = e\left(\sum_a Z_a n_a \boldsymbol{V}_a - n_e \boldsymbol{V}_e\right)$.
An important remark needs to be made regarding the turbulent transport across the magnetic field: since turbulent fluctuations are not resolved by the mean-field fluid equations, an approximate diffusive closure is assumed, where ad-hoc turbulent transport coefficients are introduced for particles ($D_\perp$),

[1] Plasma temperatures in the fusion community are generally expressed in Joules or electron volts (eV), and it is understood that $T = k_B T$, with k_B Boltzmann's constant.

momentum ($\eta_\perp$), heat ($\chi_{e,\perp}$ and $\chi_{i,\perp}$), and current ($\sigma_\perp$). These coefficients are typically estimated from experimental data and are, currently, one of the main sources of uncertainty in plasma edge codes. Sources and sinks related to plasma-neutral and plasma-wall interactions are modeled using either a fluid or a kinetic description. In the fluid approach, mass and momentum conservation for atoms are imposed through equations similar to Eq. 7 and 8, solved together with the plasma equations. In this approach it is also typically assumed that neutrals have the same temperature as the ions. In the second approach, the kinetic Boltzmann equation for neutrals is solved with a Monte Carlo method [15]. Clearly, the two approaches have pros and cons: while the fluid description is cheaper in terms of computation, its accuracy decreases for low plasma-neutral collisionality regimes, where the fluid approximation is no longer valid; on the other hand, a more accurate kinetic description requires a larger computational effort and introduces statistical noise in the set of coupled plasma-neutrals equations. In either case, plasma-neutral and plasma-wall interactions introduce various uncertain parameters such as reaction rates and recycling coefficients R_c[2].

3.2 A fixed-point iterative solver

The governing equations described in the previous section are discretized using a finite volume approach, and are then marched in time with an implicit Euler scheme. Several tens of thousands of iterations are usually required for convergence, due to limitations on the time step caused by numerical instabilities. In this situation, the standard storing strategy of TAPENADE for adjoint AD needs adaptation. In fact, this would by default store the entire plasma state at each iteration of the time loop, requiring a prohibitive amount of memory. However, in a typical plasma edge simulation one is interested in the final steady-state solution, with the (pseudo-)time stepping retained for stabilization and to resolve nonlinearities. Therefore, the set of discretized equations can be reinterpreted as a fixed-point iterator, for which the efficient two-phase adjoint AD approach of Christianson is available [42][3]. The essence of this strategy is that in a fixed-point iterator only the final converged state matters. As such, all intermediate states are irrelevant and do not need storing in the forward sweep, with the backward sweep iteratively calculating the adjoints based on the converged final state.

To display this approach, we first introduce some notations. First, we succinctly denote the set of governing equations 7 - 12 as $\mathcal{C}(Z, X) = 0$, where $Z = \left(n_a, u_{\parallel a}, T_e, T_i, \phi\right)$ is the vector of (primal) state variables, and X the vector of input variables, such as the turbulent transport coefficients $D_\perp$, $\chi_{e,\perp}$, etc. Second, we define an output quantity of interest (QoI) or cost function $\mathcal{J} = f(Z, X)$, which depends both on the state and input vectors. Note that for general sensitivity analysis $\mathcal{J}$ can be a vector, while in an optimization context this is typically a scalar, with $\mathcal{C}(Z, X) = 0$ acting as the PDE constraint. Third, we define the fixed-point iterator $Z^{k+1} = G(Z^k, X)$, with which the PDEs are

[2] Recycling refers to the process of ions hitting a solid surface, recombining with an electron, and being re-emitted as neutrals. R_c is the fraction of impinging ions recycled as neutrals, with $1 - R_c$ the fraction of absorbed ions. For neutrals, R_c is more precisely the fraction of *reflected* particles.

[3] This approach is sometimes referred to as "reverse accumulation". However, reverse accumulation is more generally considered to be an alias for the adjoint mode of AD.

solved. The general structure of the fixed-point operator G is as follows[4]:

$$G: \begin{cases} \texttt{for} \quad a = 0, N_s - 1 \\ \quad \texttt{solve momentum eq.} \rightarrow u_{\parallel a} \\ \texttt{end} \\ \texttt{for} \quad a = 0, N_s - 1 \\ \quad \texttt{solve continuity eq.} \rightarrow n_a \\ \texttt{end} \\ \texttt{solve potential eq.} \rightarrow \phi \\ \texttt{solve ion energy eq.} \rightarrow T_i \\ \texttt{solve electron energy eq.} \rightarrow T_e \end{cases} \tag{13}$$

where N_s is the total number of plasma species, possibly including fluid neutrals.
The two-phase strategy of Christianson [42] is then implemented as displayed in Table 1, where some variable initializations have been omitted for simplicity. During the forward sweep (top half of the table), no value is stored in calls to G within the fixed-point loop, but an extra iteration shown as $Z^* = \overrightarrow{G}(Z^{k+1}, X)$ is performed on the already converged state, this time storing the converged primal state Z^* and all intermediate variables required for the reverse evaluation of G. At the corresponding location in the backward sweep (bottom half of the table), a new adjoint state $\overline{Z}^k$ is converged by an adapted fixed-point loop, employing the converged primal state Z^* and its carefully preserved adjoint $\overline{Z}^\star$ stemming from the adjoint cost function $\overline{f}$. The iterations are stopped when the increment in adjoint state sensitivities is sufficiently small. After the loop, the last instruction computes the final sensitivity by accumulating the effect of the converged $\overline{Z}^{k+1}$ into the adjoint cost function sensitivity $\overline{X}$. Notice that two different adjoints of $G(Z, X)$ are used: $\overleftarrow{G}_Z$, the adjoint of G with respect to the state Z only, and $\overleftarrow{G}_X$, with respect to the input X only. Both of them use repeatedly the values stored during $\overrightarrow{G}$, and their general structure follows the inverse ordering of Eq. 13, as usual for adjoints:

$$\overleftarrow{G}: \begin{cases} \texttt{solve adj. electron energy eq.} \rightarrow \overline{T}_e \\ \texttt{solve adj. ion energy eq.} \rightarrow \overline{T}_i \\ \texttt{solve adj. potential eq.} \rightarrow \overline{\phi} \\ \texttt{for} \quad a = N_s - 1, 0, -1 \\ \quad \texttt{solve adj. continuity eq.} \rightarrow \overline{n}_a \\ \texttt{end} \\ \texttt{for} \quad a = N_s - 1, 0, -1 \\ \quad \texttt{solve adj. momentum eq.} \rightarrow \overline{u}_{\parallel a} \\ \texttt{end} \end{cases} \tag{14}$$

3.3 Linear solvers

A final point of interest is the derivative of the solution of a linear system of equations, which takes place when solving each of the discretized PDEs in Eq. 13. This can be significantly simplified both in tangent and adjoint mode. In fact, it is more efficient to directly provide the differentiated form of linear solvers in a discrete setting than having the AD tool differentiating the whole solver method [43, 44]. Moreover, the source code of the solvers may not be available at all, for example when using external libraries.

[4] In the actual code a total momentum equation is solved before the continuity ones, and a total energy equation is solved before the ion energy one. These are omitted here for ease of notation.

Table 1: Adjoint of fixed-point iterator by the two-phase method of Christianson

```
initialize Z^0
while ||Z^{k+1} − Z^k|| ≥ ε
    Z^k = Z^{k+1}
    Z^{k+1} = G(Z^k, X)
end
Z* = G→(Z^{k+1}, X)
J = f(Z*, X)
```

```
J̄ = 1
[Z̄*, X̄] = f̄(Z*, X, J̄)
initialize Z̄^0
while ||Z̄^{k+1} − Z̄^k|| ≥ ε
    Z̄^k = Z̄^{k+1}
    Z̄^{k+1} = G←_Z(Z*, X, Z̄^k) + Z̄*
end
X̄ = X̄ + G←_X(Z*, X, Z̄^{k+1})
```

For a general linear system $Ax = b$, the discrete tangent counterpart follows the basic product chain rule and yields

$$A\dot{x} = \dot{b} - \dot{A}x, \tag{15}$$

which is solved for the sensitivity $\dot{x}$ after the primal linear system is solved for x, employing the same solver routine. In adjoint mode, by definition of adjoint variable and operator with its inner product, the following result is obtained:

$$A^T\overline{b} = \overline{x}, \tag{16}$$

$$\overline{A}_{i,j} = -x_j\overline{b}_i. \tag{17}$$

In this case, the linear system in Eq. 16 is first solved for $\overline{b}$, employing again the same primal solver routine and simply transposing the matrix A. Afterwards, the primal linear system is solved and x fed to Eq. 17. We point out that, in the current implementation, both the matrix A and the primal solution x are recomputed in the backward sweep of the fixed-point iterator $\overleftarrow{G}$. In future work, we will assess the advantage of storing A and/or x for each state variable along with the converged primal state Z^*.

4 Adjoint AD verification

Having applied the AD strategies just described, in this section we demonstrate the correctness of adjoint AD sensitivities in SOLPS-ITER obtained through TAPENADE. Compared to our previous work [23], we employ here the most recent SOLPS-ITER version, featuring a fully unstructured solver [45], and a new discretization scheme correctly accounting for grid non-orthogonality, significantly improving the accuracy of fluid neutral models [46].
Furthermore, we adopt the recently developed Advanced Fluid Neutral models (AFNs) [16]. AFNs provide neutral solutions close to kinetic ones, at a much lower computational cost. Moreover, AFNs do not require additional free parameters compared to standard fluid neutral models [47], consistently accounting for plasma-surface interactions and neutral transport.

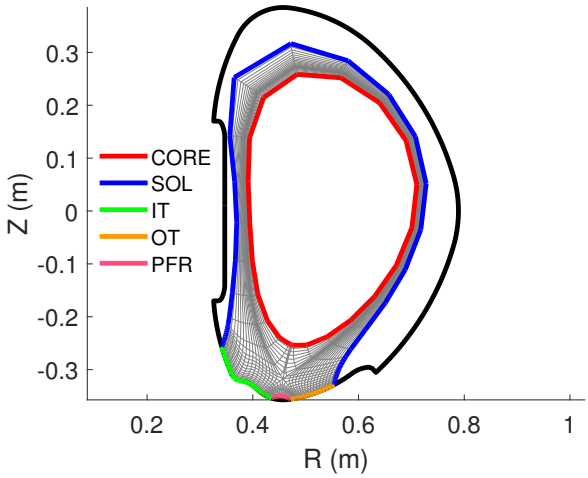

Figure 2: Geometry and computational grid for the COMPASS case (gray), with identification of boundaries: core (red), SOL (blue), IT (green), OT (orange), and PFR (pink). The vessel wall is represented in black.

4.1 Case description

The verification is based on the realistic case of the COMPASS tokamak presented in [48], for which the 2D geometry in the radial-poloidal plane is shown in Figure 2. In this figure we also identify the boundaries of the simulation domain, which will be addressed in the remainder of the work: the core, the SOL, the inner and outer targets (IT and OT) of the divertor, and the PFR. Note that in the following we split the PFR into its inner and outer part, PFR_i and PFR_o respectively. The plasma consists of deuterium neutral atoms D^0 and ions D^+. Full details of the simulation setup are given in A.

The OT peak heat load $q_\perp$ is taken as output QoI $\mathcal{J}$, since it is among the critical quantities affecting the divertor performance, and can be directly linked to engineering limits for surface melting, cooling, and safe operation of this component. The computation of $\mathcal{J}$ is done according to

$$\mathcal{J} = \max_s q_\perp = \max_s \left[\Gamma_e T_e + q_e + q_i + \Gamma_i \left(T_i + E_{ion} + K_i\right) + \Gamma_n (T_n + K_n)\right] / A_\perp, \tag{18}$$

where Γ is the poloidal particle flux, q is the internal energy flux, T is the temperature, and K the particle kinetic energy. The subscripts $e/i/n$ refer to electrons, ions, and neutrals respectively. Finally, $E_{ion} = 13.6$ eV is the D ionization energy, $A_\perp$ the perpendicular area of contact, and s is the curvilinear coordinate along the target. Note that in principle the "max" function is non-differentiable. In practice, TAPENADE computes the sensitivity of the r.h.s. of Eq. 18 without the max statement, at the coordinate where the maximum of the primal is located.

The relative sensitivity S of $\mathcal{J}$ with respect to an input parameter x is then evaluated as

$$S = \frac{\partial \mathcal{J}}{\partial x} \frac{x}{\mathcal{J}}. \tag{19}$$

Such sensitivity can be interpreted as the relative change in $\mathcal{J}$ for a given relative change in the input x, while other inputs are unchanged. A positive sensitivity thus means an increase in $\mathcal{J}$ when x increases, while a negative sensitivity means that increasing x leads to a reduction in $\mathcal{J}$.

In the following we will consider input parameters with the strongest impact on the solution, and for which sensitivity information was so far hardly available. Moreover, these parameters often affect the simulation in a non obvious way, and differently across plasma regimes, as will be shown in section 6. The parameters considered include boundary conditions (BCs), anomalous transport coefficients, and recycling coefficients R_c. The latter influence the role of neutral particles and in turn the plasma

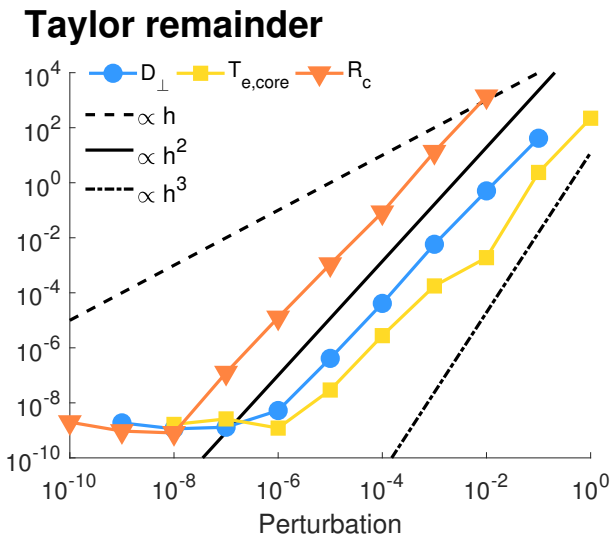

Figure 3: Taylor remainder showing a second order convergence for the adjoint AD sensitivities of $D_\perp$ (blue circles), $T_{e,core}$ (yellow squares), and R_c (orange triangles). The black lines indicate the convergence order h (dashes), h^2 (solid), and h^3 (dash-dots).

regime. Turbulent transport coefficients strongly affect plasma flow across the magnetic field. They are mostly uncertain and needs tuning to match experimental data. Lastly, among BCs the core density and temperature ones have the main role in determining the plasma regime.

4.2 Verification results

At first, we carry out the verification by checking the second-order Taylor remainder for adjoint AD sensitivities, defined as

$$\left| \mathcal{J}(x_i + h) - \mathcal{J}(x_i) - h \frac{\partial \mathcal{J}(x_i)}{\partial x_i} \right|, \tag{20}$$

where $x_i + h$ means a perturbation of element i of the input vector X, with h the perturbation that is repeatedly decreased. We evaluate the Taylor remainder in Eq. 20 for three specific parameters: the particles diffusion coefficient $D_\perp$, the core electron temperature BC $T_{e,core}$, and the OT recycling coefficient R_c for D^0. Figure 3 displays the Taylor remainder when these parameters are perturbed with respect to their reference value, proving that adjoint AD sensitivities are correct, as they follow a second order convergence rate. Since the largest primal residual stagnates just below 10^{-8}, the Taylor remainder cannot be reduced below this level and reach machine precision.

In a second step, we compare sensitivities for all BCs, transport, and recycling coefficients to central FD adopting relative perturbations $h_{FD} \sim 10^{-6}$-10^{-7}, providing the optimal results according to the Taylor test. Tangent AD provides results as accurate as adjoint AD, up to roundoff, and we therefore use it as additional cross-check. The results are listed in Table 2 in terms of relative error of adjoint AD sensitivities with respect to central FD, $\varepsilon_{FD} = (S_{ADJ} - S_{FD})/S_{ADJ}$, and relative error of adjoint AD sensitivities with respect to tangent AD, $\varepsilon_{TGT} = (S_{ADJ} - S_{TGT})/S_{ADJ}$, for parameters with $S \geq 10^{-4}$. A good agreement is found between FD and adjoint AD, with relative differences in the range 10^{-5}-10^{-9}. As expected, adjoint and tangent AD agreement is excellent, with relative differences among the two methods reaching machine precision. For parameters with $S < 10^{-4}$, not listed in Table 2, the agreement between adjoint and tangent AD is still excellent and near machine precision. Instead, FD results degrade, with relative errors up to 10^{-2}, likely caused by roundoff errors.

The results provided in this section, complemented with additional verification tests available in the online repository [49], prove the reliability of adjoint AD sensitivities obtained by applying TAPENADE

to SOLPS-ITER.

5 AD code efficiency

Now that we have proven correctness of adjoint AD derivatives, we study their computational efficiency to probe if further improvements are required. Hence, in this section we compare the computational effort estimates from Eq. 4 and Eq. 6 in section 2 to empirical computational time and memory scalings for the COMPASS case described previously, which is taken as reference. These scalings are obtained for two sets of simulations, one in which the grid is refined, and the second in which the number of species N_s is increased. For the grid, radial and poloidal directions are alternatively refined by a factor of two, starting with the radial. For N_s, the increase is obtained by adding He ($N_s = 5$), He+N ($N_s = 13$), He+Ar ($N_s = 24$), He+N+Kr ($N_s = 50$), and He+Ar+W ($N_s = 99$) to the reference scenario, employing similar BCs, transport, and recycling coefficients as for D. Note that these multi-species cases are not representative of COMPASS operation, but are simply a way of obtaining scalings as a function of N_s.
Three code bases have been used to compute the scalings:

- the primal, *i.e.* the standalone B2.5 solver in the standard SOLPS-ITER code distribution;
- the tangent AD code, obtained by differentiating the primal in vector mode with respect to 2, 4, 8, and 16 parameters;
- the adjoint AD code, obtained by differentiating the primal with respect to the same 2 and 16 parameters as for tangent AD.

The results in the following sections have been obtained on the Genius cluster of the Flemish Supercomputer Centre using computing nodes with 2 Intel® Xeon® Gold 6240 CPUs, 192 GB RAM, and 200 GB SSD local disk.

5.1 Computational time

We estimate first the computational time scaling, by running several times the primal, tangent, and adjoint AD code for a fixed amount of time, and averaging the number of iterations performed. The ratio of primal over AD iterations, at the same grid refinement and number of species, provides the time increase estimate. Note that for adjoint AD we also include the time for executing the first half of Table 1 with a single primal iteration. This is needed for correct initialization of the problem but is negligible compared to the remainder of the adjoint computation. The scalings are shown in Figure 4, where the fixed cost of adjoint AD is between five and nine times the primal, clearly advantageous for many input parameters, albeit slightly larger than the theoretical value given by Eq. 6. Tangent AD is instead more efficient than the theoretical bound $\propto (1+1.5n)$ as expected from Eq. 4, and is preferable up to about ten parameters. For larger grid sizes, adjoint AD efficiency seems to improve, while tangent AD is less affected, as can be deduced from Figure 4a. In contrast, for increased number of species both adjoint and tangent AD efficiencies reduce. This is likely a consequence of the significant increase in memory required when the number of species is increased, as will be shown in the next section.
Furthermore, we note that the differences between results for adjoint AD with 2 and 16 parameters are negligible (∼2% in average). In fact, most of the computational effort is put in the adjoint fixed point iterator loop of Table 1, employing $\overleftarrow{G}_Z$, which is the same for all parameter sets. The effect of different parameters comes into play only in the adjoint cost function $\overline{f}$ and the last operation in Table 1, employing $\overleftarrow{G}_X$, both of which are negligible compared to the fixed point iterator part.
A scaling based on converging both AD modes to the same level accuracy would provide roughly the same results, as convergence rates of tangent and adjoint are similar and only slightly lagging behind the primal [50].

Table 2: Relative error of adjoint AD against central FD ($\varepsilon_{\mathbf{FD}}$) and tangent AD ($\varepsilon_{\mathbf{TGT}}$). $\mathcal{O}(\mathbf{h_{FD}})$ indicates the order of magnitude of the FD relative perturbation.

Parameter	$\mathcal{O}(\mathbf{h_{FD}})$	$\varepsilon_{\mathbf{FD}}$	$\varepsilon_{\mathbf{TGT}}$
Electron energy BC			
Core	10^{-7}	3×10^{-8}	2×10^{-14}
$\mathrm{PFR_i}$	10^{-6}	7×10^{-6}	4×10^{-14}
$\mathrm{PFR_o}$	10^{-6}	3×10^{-6}	1×10^{-13}
SOL	10^{-6}	6×10^{-9}	2×10^{-14}
Ion energy BC			
Core	10^{-7}	7×10^{-8}	2×10^{-14}
IT	10^{-7}	2×10^{-5}	1×10^{-13}
OT	10^{-7}	1×10^{-5}	4×10^{-12}
$\mathrm{PFR_i}$	10^{-6}	5×10^{-6}	3×10^{-14}
SOL	10^{-6}	1×10^{-8}	2×10^{-14}
Continuity BC $\mathrm{D^+}$			
Core	10^{-6}	2×10^{-8}	1×10^{-13}
IT	10^{-6}	4×10^{-7}	3×10^{-13}
OT	10^{-6}	2×10^{-7}	6×10^{-14}
$\mathrm{PFR_i}$	10^{-6}	3×10^{-6}	0
$\mathrm{PFR_o}$	10^{-6}	1×10^{-6}	2×10^{-13}
SOL	10^{-6}	9×10^{-8}	2×10^{-13}
Momentum BC $\mathrm{D^+}$			
IT*	10^{-6}	2×10^{-6}	2×10^{-14}
OT*	10^{-6}	7×10^{-8}	3×10^{-14}

*These parameters are equal to zero, therefore the absolute and not the relative perturbation is shown.

Parameter	$\mathcal{O}(\mathbf{h_{FD}})$	$\varepsilon_{\mathbf{FD}}$	$\varepsilon_{\mathbf{TGT}}$
Potential BC			
IT*	10^{-6}	9×10^{-7}	6×10^{-15}
OT*	10^{-6}	3×10^{-7}	2×10^{-14}
Transport coefficients			
$D_\perp$	10^{-7}	5×10^{-7}	2×10^{-13}
$\chi_{e,\perp}$	10^{-7}	1×10^{-7}	9×10^{-15}
$\chi_{i,\perp}$	10^{-7}	3×10^{-7}	2×10^{-14}
$\eta_\perp$	10^{-6}	4×10^{-6}	7×10^{-13}
$\sigma_\perp$	10^{-7}	3×10^{-6}	3×10^{-14}
Recycling $\mathrm{D^0}$			
IT	10^{-7}	2×10^{-8}	2×10^{-14}
OT	10^{-7}	2×10^{-8}	2×10^{-14}
$\mathrm{PFR_i}$	10^{-7}	2×10^{-7}	2×10^{-14}
$\mathrm{PFR_o}$	10^{-7}	1×10^{-7}	3×10^{-13}
SOL	10^{-7}	5×10^{-8}	2×10^{-14}
Recycling $\mathrm{D^+}$			
IT	10^{-7}	2×10^{-7}	2×10^{-14}
OT	10^{-7}	6×10^{-8}	2×10^{-14}
$\mathrm{PFR_i}$	10^{-7}	5×10^{-6}	3×10^{-14}
$\mathrm{PFR_o}$	10^{-7}	7×10^{-7}	3×10^{-13}
SOL	10^{-7}	2×10^{-7}	3×10^{-14}

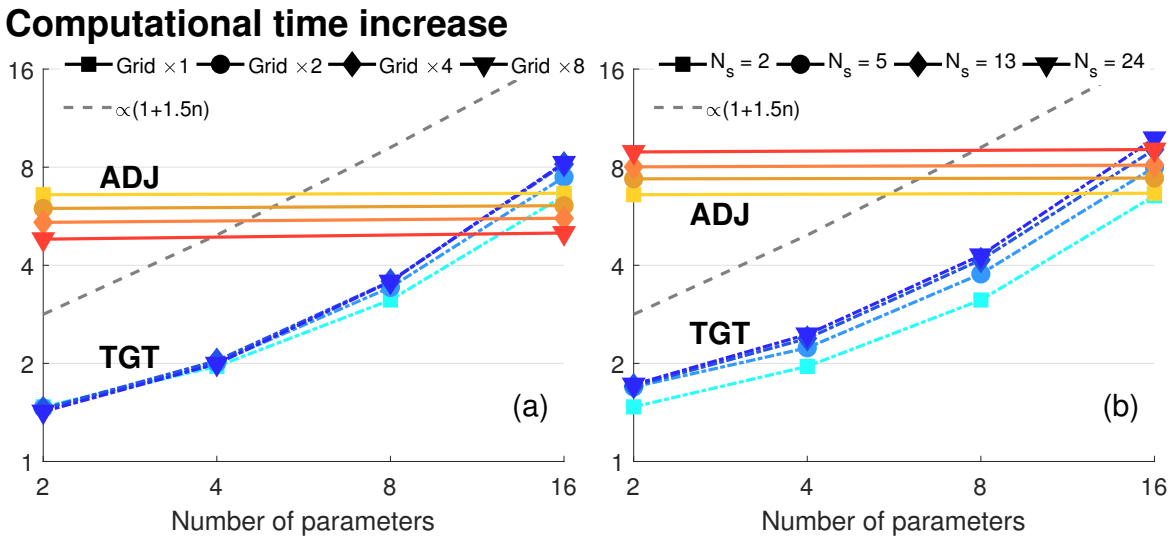

Figure 4: Increase in computational time of the differentiated AD code compared to primal, plotted against the number of input parameters n for: different grid refinements with fixed number of species $N_s = 2$ (a); different N_s with fixed grid refinement $\times 1$ (b). Adjoint AD is identified with solid lines in shades of red while tangent vector AD with dash-dotted lines in shades of blue. The theoretical tangent AD scaling is identified with a gray dashed line.

We also point out that tangent AD in vector mode could in principle be parallelized, splitting the input range in different subsets and computing them in parallel, *e.g.* four tangent simulations with four inputs each, instead of one simulation with 16.

5.2 Memory requirement

We analyze next the memory footprint, employing the Arm® MAP profiler to assess memory requirements of AD code compared to primal. The resulting scalings are shown in Figure 5, where we display only one curve for adjoint AD, since memory requirements for 2 and 16 parameters are essentially the same. This confirms that the main computational bottleneck is indeed the adjoint fixed point iterator. From Figure 5 it is clear that adjoint AD memory requirements grow more rapidly compared to tangent vector AD, either when the grid is refined or species are added. In fact, primal and tangent AD roughly follow a linear scaling, at least for larger case sizes, as internal variables are defined on the grid itself and for each species. In adjoint AD, the standard checkpointing strategy of TAPENADE stores variables at each procedure call within the fixed-point iterator, thus causing a more-than-linear growth. The sharper increase visible for increasing number of species is due to more frequent checkpoints within for-loops on species at the high levels of the call tree. An example are the for-loops solving the momentum and continuity equations in Eq. 14. This superlinear growth in adjoint AD memory requirements could significantly hamper adjoint AD applicability for reactor-scale simulations. First of all, the memory footprint increases sharply with increasing number of species, and reactor-scale scenarios often include many high-Z impurities such as krypton or even tungsten (+36 and +74 plasma species respectively). Moreover, simulations up to the vessel walls, enabled by the new unstructured solver in SOLPS-ITER, require grids with significantly more cells than typical structured meshes. In addition to that, computational meshes finer than typically used in the community should be employed, to reduce discretization error. Lastly, the adjoint of the kinetic neutral code that is not considered in this paper, could contribute significantly to the total memory consumption.
To cap memory requirements, in future work we will tune the procedure-oriented checkpointing strategy to make it less systematic. Experience shows that it is often detrimental to apply checkpointing

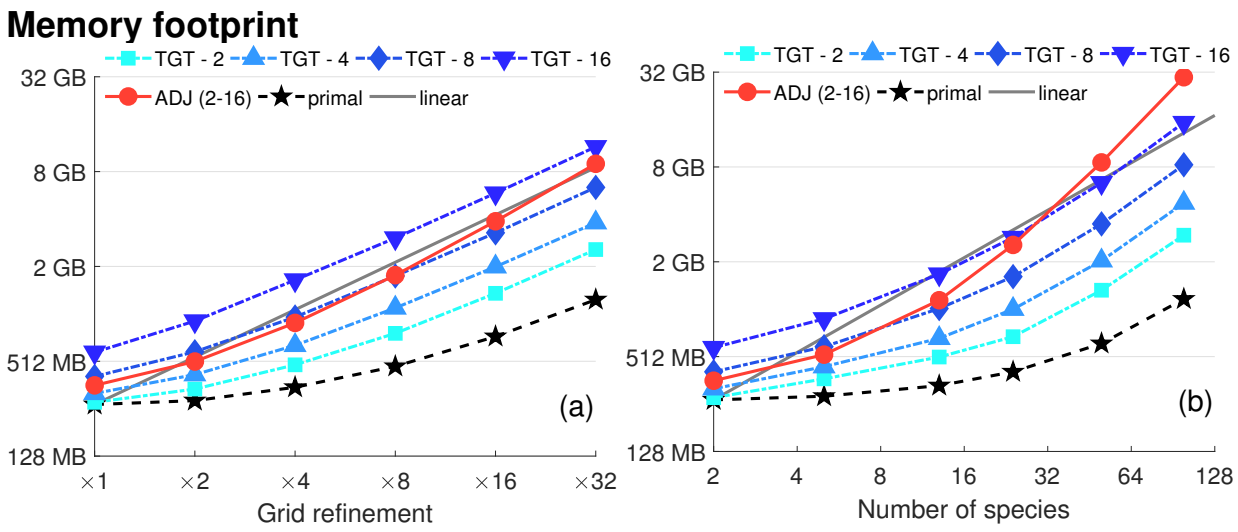

Figure 5: Memory footprint of the differentiated AD code and the primal, as a function of: different grid refinements with fixed number of species $N_s = 2$ (a); different N_s with fixed grid refinement $\times 1$ (b). Adjoint AD for all number of input parameters is identified with a solid red line with circles. Dash-dotted lines indicate tangent vector AD, with different shades of blue and markers for different number of input parameters. The primal is indicated with a black dashed line with stars, while the linear trend line with a solid gray line.

on "small" calls near the bottom leaves of the call tree. Added directives in the source code can trigger an alternative mode known as "split adjoint", where small calls are treated as inlined code, thus providing a better trade-off between storage and duplicate execution. Similarly, some procedure checkpoints can be replaced with recomputation, potentially leading to reductions in computational time, as recomputation may occur from a memory level with faster access, e.g. cache instead of RAM. Another potential improvement is to identify gather-scatter loops in the code, and more generally loops whose iterations are independent and could be executed in any order. Adjoint AD of these loops can take advantage of this, reducing the peak amount of memory used to store intermediate values.
From Figure 5 it can also be noticed that increasing the number of input parameters in tangent AD linearly shifts the curves upwards by roughly the same amount, as could be expected from Eq. 4. Finally, adjoint AD proves more efficient than tangent also in terms of memory for many input parameters and small case size. Indeed, in Figure 5 tangent AD shows lower requirements only up to eight parameters for typical grid sizes and number of species.

6 Application to sensitivity analysis

We now apply adjoint AD sensitivity analysis to the typical way of performing plasma edge simulations, namely a core density scan at fixed input power. This allows analyzing how divertor target quantities, characteristic for the power exhaust problem, are affected by the changing density regime. In particular, we will consider heat load, plasma (electron) temperature, and ion particle flux profiles. While we already identified the first in section 4.1 as the key quantity linked to engineering and material constraints, the other two are linked to surface erosion, which in turn affects thermo mechanical properties and divertor lifetime.
Interpretation of plasma profiles and sensitivities requires the concept of *detachment* [51, 52]. This is a favorable operating condition in which a significant fraction of plasma particle, momentum, and energy flux in the parallel direction is dissipated before reaching the divertor targets. Detachment is often achieved by increasing plasma density, by which a cloud of recycled neutral particles forms in

front of the targets, effectively protecting them from the hot plasma. At first, increasing the density also increases particle and energy fluxes towards the targets (attached regime). For further density increases, these fluxes flatten and subsequently decrease, in what is called the rollover, with plasma temperature dropping to relatively low values in detached regime (<5 eV).
We employ three cases for the density scan, with core density set to $n_{e,core} = [0.8, 1.5, 2.0] \times 10^{19}$ m^{-3}. These three cases are identified as Low, Medium, and High (L, M, H respectively) density. Profiles of OT heat load $q_\perp$ and electron temperature T_e are shown in Figure 6a and 6b respectively, for the three cases. Both profiles decrease with increasing density, with the heat load flattening and the peak moving further away from the separatrix. This indicates attached conditions for the low density case, given the high separatrix temperature, while the medium density case is in partially detached conditions, with $T_e < 5$ eV around the separatrix. Finally, the high density case goes deeper into detachment, with temperatures well below 5 eV on the entire target. Next, we show with the relative adjoint sensitivities how the same input parameter differently affects the outer target peak heat load $\mathcal{J}$ for varying operating conditions.

6.1 Results and interpretation

Figure 7 summarizes the largest sensitivities among BCs, transport, and recycling coefficients. Starting with BCs, $n_{e,core}$ shows a significant influence on the peak heat load. While for the M and H cases the sensitivity is negative, in accordance to the heat load reduction shown in Figure 6a, the L case exhibits a positive sensitivity, meaning that it would increase the peak. This is typical of the rollover caused by detachment. This effect is confirmed by the L* heat load profile in Figure 6a, obtained by slightly increasing the L density to $n_{e,core} = 1.0 \times 10^{19}$ m^{-3}. The input power BC P_{core} trivially increases the target heat load, thus shows a positive sensitivity, albeit more pronounced for electrons than ions. This can be explained by separating the electron and ion contributions to the total target heat load, as displayed in Figure 8a. In fact, for low and medium densities, the electron contribution dominates the heat load, hence the larger sensitivity when heat is introduced through electrons. At these low and intermediate densities, the ion contribution is smaller, hence the lower $P_{i,core}$ sensitivity. For high densities instead, $P_{i,core}$ still has a relatively small effect because target temperature is low and the ion contribution is dominated by the large recombination term $\Gamma_i E_{ion}$ in Eq. 18, see the target particle flux increase in Figure 8b.
The sensitivities in Figure 7b show that turbulent transport coefficients drive heat load reduction, although their effect is about 80% smaller compared to the BCs just analyzed. Larger particle and heat diffusivities increase the cross-field transport and essentially allow spreading the heat load over a wider area before this is convected-conducted along the magnetic field towards the target. The role of particle diffusivity $D_\perp$ increases with density, which is explained by the larger and more peaked ion particle flux Γ_i at the target shown in Figure 8b: increasing the diffusivity would flatten the particle flux and, as a consequence, the heat load. Regarding $\chi_{e,\perp}$, its spreading effect reduces with increasing density, as the electron heat flux contribution reduces and flattens, as visible in Figure 8a. Finally, the ion heat diffusivity $\chi_{i,\perp}$ has a small effect due to the same reasoning provided for $P_{i,core}$: either the ions only partially contribute to the heat load (L-M cases), or their contribution mainly arises from the recombination term (H case).
Lastly, the sensitivity of recycling coefficients for D^0 is shown in Figure 7c (for D^+ the behavior is essentially the same). Clearly, R_c at the OT has an enormous effect on reducing the peak heat load at medium-high densities. Since the neutral density in front of the target increases with $n_{e,core}$, driving detachment, a larger R_c at the OT essentially means more neutrals are locally available to further drive detachment, and the overall density in the domain increases with an effect similar to $n_{e,core}$. This high sensitivity is also related to R_c being close to 1 at the OT. In this case, the target over upstream particle flux scales as $\sim 1/(1 - R_c)$ in a 0D flux tube balance. Thus, small R_c perturbations lead to a large increase in target particle flux. In the same way, an increased recycling at the IT and SOL

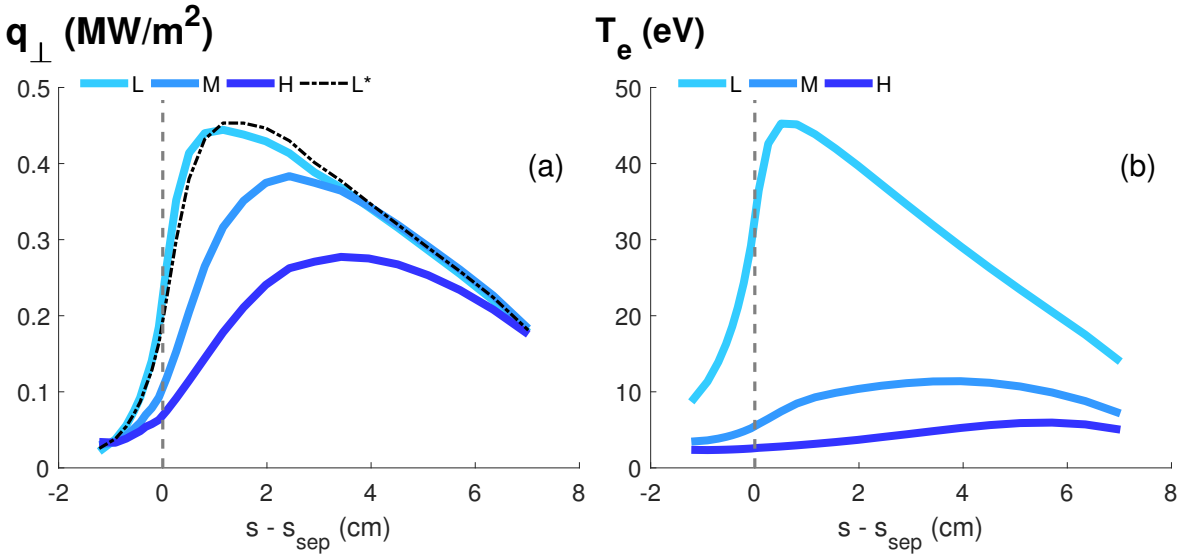

Figure 6: Outer target profiles of the heat load (a) and electron temperature (b). The color shades indicate the L (light blue), M (medium blue), and H (dark blue) case. The dash-dotted black line in (a) refers to the perturbed L* density. The separatrix position is identified with a vertical dashed line.

boundaries effectively produces a density increase in the domain. The less pronounced effect for IT is primarily due to the distance between IT and OT, so that there is little direct impact on OT plasma profiles, while close to the corner between SOL and OT, see Figure 2, neutrals are recycled near to or in front of the OT, thus having a stronger impact.

Large sensitivities put in the spotlight parameters potentially requiring better characterization: it is well known for example that recycling coefficients strongly influence simulation results, while their actual value is often uncertain [53,54]. In the cases at hand, the sensitivity on R_c at the OT turns out to be dominating that of transport coefficients in all regimes. Remarkably, in the high density case even the *inner* target R_c has a larger effect on the *outer* target peak heat load than $D_\perp$. The turbulent transport coefficients are also amongst the main sources of uncertainty in plasma edge simulations given their empirical nature [55], although so far no attempt has been made to estimate this uncertainty. Despite that their sensitivity is smaller than that of other parameters in the analyzed cases, this potentially leads to large uncertainties on simulation outputs.

6.1.1 Sensitivity maps

Adjoint AD enables for the first time assessing the effect on $\mathcal{J}$ of parameters in each specific domain location (grid cell). This leads to sensitivity maps, informing modelers on where their solution is more sensitive to an input perturbation. To display this, the sensitivity of the OT peak heat load on electron heat diffusion $\chi_{e,\perp}$ in each grid cell is shown in Figure 9, for the three density cases. Accessing this kind of information through brute-force FD is practically impossible, as each sensitivity map would require $48 \times 24 = 1152$ perturbed simulations, one for each grid cell, while it requires only one adjoint AD computation. The difference in computational effort proves once more the advantage of adjoint AD.

In Figure 9, the flux tube delimited by the two flux surfaces in black indicates where the OT peak heat load is located, hereafter referred to as the peak flux tube. Note that $\chi_{e,\perp}$, as other transport coefficients, is assumed constant on the entire domain in our test case. However, its local effect is noticeably diverse, and significantly varies with density regimes, as the maps clearly show.

Interpretation is straightforward considering that $\chi_{e,\perp}$ drives heat diffusion in the cross-field direction, which is then transported towards the target in the parallel direction, along flux tubes: a larger

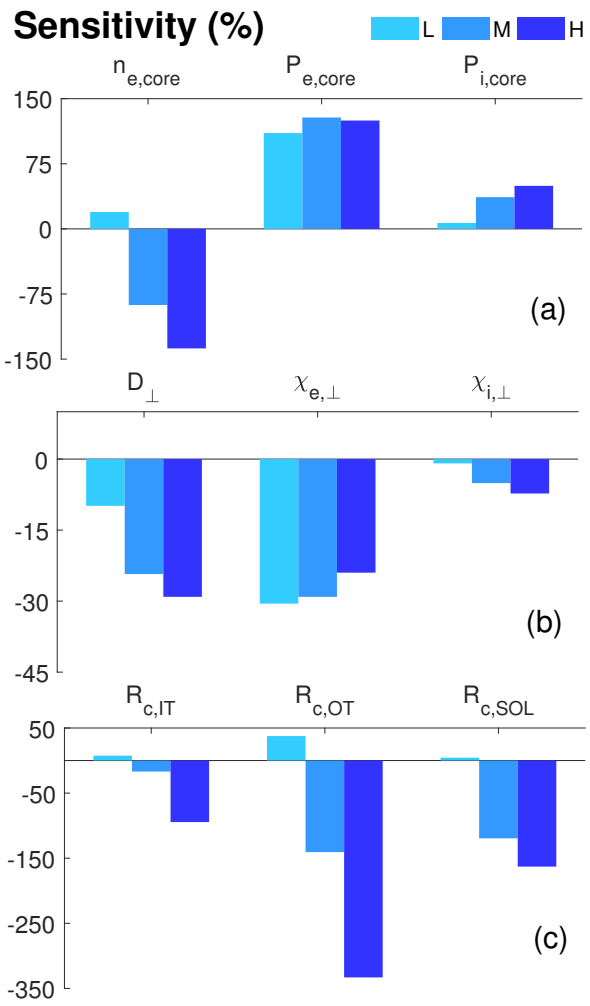

Figure 7: Relative sensitivity of the most significant BCs (a), transport coefficients (b), and recycling coefficients (c). The color shades indicate the L (light blue), M (medium blue), and H (dark blue) case.

coefficient on the left of the peak flux tube increases heat transport towards that flux tube, thus showing a positive sensitivity. Vice-versa, on the right of the peak flux tube an increase in $\chi_{e,\perp}$ drives heat transport away from that flux tube, thus the negative sensitivity. It can also be noticed that the effect of $\chi_{e,\perp}$ is stronger upstream, as closer to the target heat diffusion towards the peak flux tube is simply not fast enough to compensate the more rapid parallel transport.

7 Conclusions

In this work, we provided for the first time adjoint AD sensitivity calculation in plasma edge codes, employing the AD tool TAPENADE on SOLPS-ITER. Adjoint AD provides sensitivities of a single output QoI with respect to all input parameters, at a cost comparable to only a small multiple of primal SOLPS-ITER simulations. Moreover, AD sensitivities are accurate to floating-point precision and do not suffer from truncation error.

We demonstrated accuracy of adjoint AD on a COMPASS case employing the OT peak heat load as QoI with respect to which we calculated the sensitivities of input parameters such as BCs, transport, and recycling coefficients. The Taylor remainder test proved that adjoint AD sensitivities are correct, following the expected second order convergence. Comparison against FD resulted in relative differences in the range 10^{-5}-10^{-9}, while against tangent AD these differences reduced to machine precision.

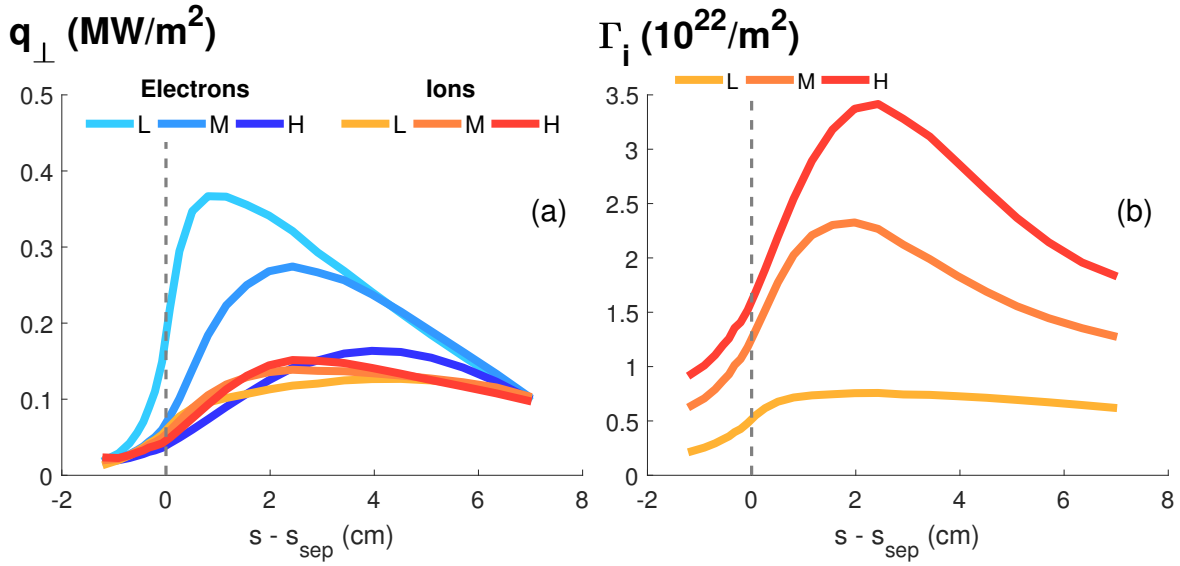

Figure 8: Outer target profiles of: the contributions to the heat load from electrons (blue) and ions (red) (a); and ions particle flux (b). The color shades indicate the L (light), M (medium), and H (dark) case. The separatrix position is identified with a vertical dashed line.

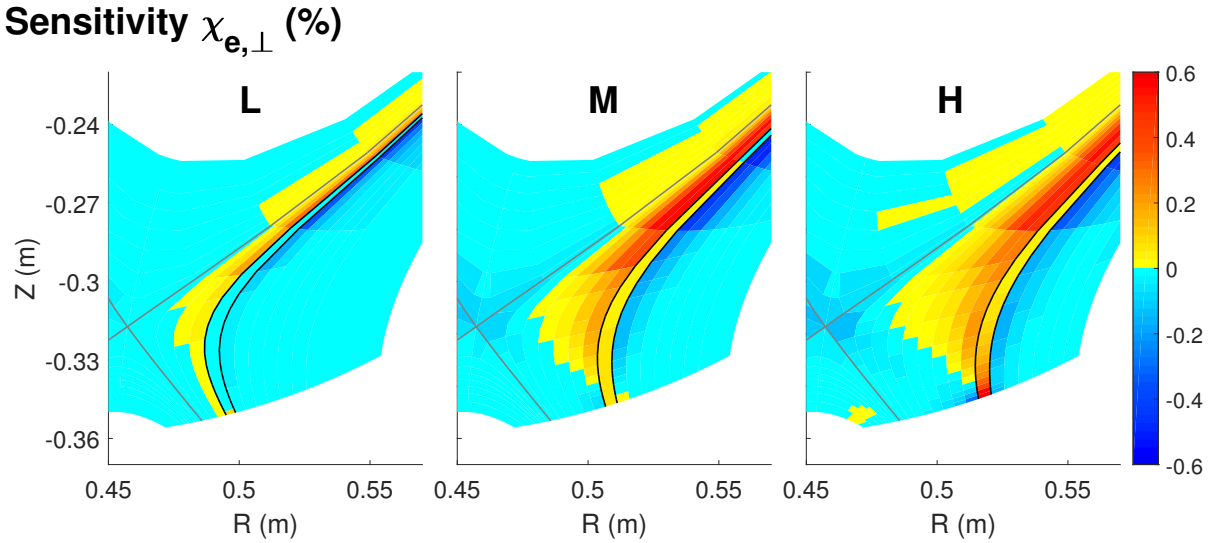

Figure 9: Zoom in the divertor region of the $\chi_{e,\perp}$ sensitivity maps for the L (left), M (middle), and H (right) density cases. Red and blue indicate respectively positive and negative sensitivities, with darker shades for larger absolute values. The separatrix is indicated with a gray line. The two flux surfaces with black lines delimit the flux tube with the OT peak heat load.

To deploy adjoint AD for SOLPS-ITER, we applied the two-phase fixed-point iterator strategy of Christianson for steady-state simulations, and provided a discrete version of the linear solvers. With these efficiency improvements, we assessed empirical scalings of AD computational time and memory requirements for different case sizes while varying grid refinement and the number of plasma species involved. The scalings agree with the expected performance of AD, and show that adjoint AD computational time is only five to nine times larger than that of the primal, irrespective of the number of input parameters. Instead, tangent AD cost linearly grows with the number of inputs, being more efficient than adjoint only up to around ten parameters. Even if memory requirements of adjoint appear similar to tangent AD for the smaller case size in Figure 5, they increase faster with the case size, due

to storage of intermediate values and the checkpointing mechanism. Tangent AD memory is observed to scale linearly, as a function of both case size and number of inputs.
Finally, we applied adjoint AD for sensitivity analysis on a COMPASS density scan, to assess the effect of key input parameters across plasma regimes. Three operating conditions have been achieved varying the core density BC. The OT recycling coefficient has been shown to be the most sensitive parameter for the OT peak heat load in (partially-)detached conditions, followed by the SOL boundary recycling. As expected, the core ions density and electrons energy BC also significantly affect the peak heat load. Despite being among the main sources of uncertainty in plasma edge codes, particles and heat turbulent diffusivities showed instead smaller sensitivities compared to target recycling and core BCs. Finally, we efficiently obtained for the first time sensitivity maps in plasma edge codes, showing that a large amount of new sensitivity information is now available for the community through adjoint AD.
Several plasma edge applications will benefit from adjoint AD sensitivity calculation now available in SOLPS-ITER. First and foremost, gradient-based optimizations with large sets of design variables are now feasible by replacing tangent AD gradient calculation with adjoint AD in the optimization framework recently deployed [56]. This enables divertor design with high-fidelity codes in terms of optimal shape and magnetic field [19,20], and model calibration based on experimental data of several unknown input parameters and their uncertainty in a Bayesian setting [56]. Furthermore, also UQ efforts benefit from an efficient sensitivity analysis, starting from simple first order moment methods to more accurate sensitivity-accelerated Monte Carlo methods [21]. Moreover, sensitivity analysis provides indication on parameters which need careful uncertainty characterization due to their large sensitivity, while others could be neglected given their small impact on the simulation output.
The next steps concerning application of AD to SOLPS-ITER involve two separate tracks, addressing the main limitations of this study: first, optimization of adjoint AD performance has not been fully explored; second, the sensitivity calculation is currently based on a fluid neutral model, while the more accurate kinetic neutral model has not been considered. The first track of developments will therefore focus on efficiency improvements in terms of memory requirement for adjoint AD, since its fast growth for increasing case size may preclude reactor-scale simulations, such as a realistic impurity mix, computational grids up to the vessel walls, or finer grids to reduce discretization error. Reduction of memory usage requires fine-tuning the checkpointing strategy, to reach a reasonable trade-off with computational time, and possibly its reduction. Moreover, both tangent and adjoint modes can benefit from another advantage: the largest cost for solving a linear system is typically the matrix factorization, and since this linear system is also self-adjoint, the same (transposed) matrix is employed for the adjoint solution. Thus, the same factorization can be re-used at almost no cost to speed-up the solution of Eq. 15 and 16 [57].
The second track of developments will target sensitivity calculation when the Monte Carlo kinetic neutral model of SOLPS-ITER is employed. This introduces statistical noise in the simulation chain, hampering gradient's accuracy. The discrete adjoint method has proven robust in case of statistical noise, albeit in a simplified 1D setting, by ensuring exact correlation between the particle trajectories of primal and adjoint simulations [58]. However, correlation in higher dimensional settings is hard to maintain. This correlation is automatically ensured in tangent AD, since tangent sensitivity calculation is performed at the same time as the primal, and thus can employ the same random seed. For adjoint AD instead, brute-force differentiation would require storing all particle trajectories to ensure such correlation, which is infeasible due to memory limitations. The challenge is thus devising an ad-hoc adjoint strategy at the particle level, requiring limited storage while ensuring exact correlation between primal and adjoint trajectories. Alternatively, one could act at the level of the random number generator, as recently proposed in [59]. The development of the adjoint strategy is additionally challenged by the large computational effort typically required by Monte Carlo methods, to keep sensitivity calculation and optimization costs feasible.

Acknowledgments

This work 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 author(s) 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. Stefano Carli is a postdoctoral research fellow of the Research Foundation-Flanders (FWO) under grant number 12E3623N. Parts of the work are supported by the Research Foundation Flanders (FWO) under project grant G078316N and G085922N. The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

Data Availability Statement

The results presented in this work have been obtained with SOLPS-ITER version ea8c1a68067, with B2.5 version 01a98c0af71, and TAPENADE version 3.16-v2-120-g686e29d1a.
The data that support the findings of this study are available in an open source online repository [49]. The scripts to reproduce the figures of this paper using such data are also available online [60].

A COMPASS case setup

For the COMPASS case adopted in this work, we employ a coarse 48×24 grid to speed-up convergence. The following BCs are applied:

- Core: Dirichlet BCs are applied to continuity and momentum equations of D^+ ions, and electron and ion energy equations, imposing plasma density $n_{e,core} = 3.25 \times 10^{19}$ m^{-3}, zero parallel velocity, and electron and ion temperature $T_{e,core} = 33$ eV and $T_{i,core} = 44$ eV, respectively. Zero drift-current BC for the potential equation is employed. For D^0 neutral atoms, zero parallel velocity gradient is imposed, together with zero particle flux.
- IT and OT: standard Bohm-Chodura BC [61].
- SOL, PFR_i, and PFR_o boundaries: an outward particle flow is imposed for D^+ continuity equation, defined as a fraction of the local sound speed flux $\Gamma_{out} = \alpha c_s n_{D^+}$, with $\alpha = -1.5 \times 10^{-3}$. A similar BC is also enforced for energy equations, with flux $\Gamma_{out} = \alpha c_s n_{i/e} T_{i/e}$, with factor $\alpha = -1.5 \times 10^{-2}$ both for ions and electrons. Zero-gradient BC is applied both to D^+ parallel velocity and to the potential. For D^0, zero particle and momentum flux is imposed.

AFNs are employed for D^0, assuming Maxwellian distributions at all boundaries and a Franck-Condon dissociation energy equal to 3 eV. Moreover, the following values of anomalous transport coefficients are employed: particle diffusivity $D_\perp = 3.5$ m^2/s, electrons heat diffusivity $\chi_{e,\perp} = 5$ m^2/s, ions heat diffusivity $\chi_{i,\perp} = 3.5$ m^2/s, viscosity $\eta_\perp = 0.2$ kg/m/s, current conductivity $\sigma_\perp = 2 \times 10^{-4} e n_e$ S/m (with e the elementary charge and n_e electron density). The parallel heat transport coefficients for electrons and ions are flux-limited, while for neutrals an isotropic flux-limit is applied to heat conductivity and viscosity. Finally, for both deuterium species a recycling coefficient $R_c = 0.99$ is employed at the targets, and $R_c = 1$ elsewhere.
Note that for the results shown in section 6 we modify the core Dirichlet temperature BC into fixed power BC both for ions and electrons, with $P_{i,core} = P_{e,core} = 172$ kW.

References

[1] A. Donné, "The European roadmap towards fusion electricity," Philosophical Transactions of the Royal Society A, vol. 377, p. 20170432, 2019.

[2] S. Wiesen, D. Reiter, V. Kotov, M. Baelmans, W. Dekeyser, A. Kukushkin, S. Lisgo, R. Pitts, V. Rozhansky, G. Saibene, I. Veselova, and S. Voskoboynikov, "The new SOLPS-ITER code package," Journal of Nuclear Materials, vol. 463, pp. 480–484, 2015.

[3] X. Bonnin, W. Dekeyser, R. Pitts, D. Coster, S. S. Voskoboyinikov, and S. Wiesen, "Presentation of the new SOLPS-ITER code package for tokamak plasma edge modelling," Plasma and Fusion Research, vol. 11, p. 1403102, 2016.

[4] H. Bufferand, C. Baudoin, J. Bucalossi, G. Ciraolo, J. Denis, N. Fedorczak, D. Galassi, P. Ghendrih, R. Leybros, Y. Marandet, N. Mellet, J. Morales, N. Nace, E. Serre, P. Tamain, and M. Valentinuzzi, "Implementation of drift velocities and currents in SOLEDGE2D–EIRENE," Nuclear Materials and Energy, vol. 12, pp. 852–857, 2017.

[5] T. Rognlien, J. Milovich, M. Rensink, and G. Porter, "A fully implicit, time dependent 2-D fluid code for modeling tokamak edge plasmas," Journal of Nuclear Materials, vol. 196-198, pp. 347–351, 1992. Plasma-Surface Interactions in Controlled Fusion Devices.

[6] G. J. Radford, A. V. Chankin, G. Corrigan, R. Simonini, J. Spence, and A. Taroni, "The particle and heat drift fluxes and their implementation into the EDGE2D transport code," Contributions to Plasma Physics, vol. 36, no. 2-3, pp. 187–191, 1996.

[7] R. Pitts, X. Bonnin, F. Escourbiac, H. Frerichs, J. Gunn, T. Hirai, A. Kukushkin, E. Kaveeva, M. Miller, D. Moulton, V. Rozhansky, I. Senichenkov, E. Sytova, O. Schmitz, P. Stangeby, G. De Temmerman, I. Veselova, and S. Wiesen, "Physics basis for the first ITER tungsten divertor," Nuclear Materials and Energy, vol. 20, p. 100696, 2019.

[8] S. Carli, R. Pitts, X. Bonnin, F. Subba, and R. Zanino, "Effects of strike point displacement on the ITER tungsten divertor heat loads," Nuclear Fusion, vol. 58, p. 126022, 2018.

[9] J.-S. Park, X. Bonnin, and R. Pitts, "Assessment of ITER divertor performance during early operation phases," Nuclear Fusion, vol. 61, p. 016021, 2021.

[10] L. Aho-Mantila, M. Wischmeier, H. Müller, S. Potzel, D. Coster, X. Bonnin, and G. C. and, "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, vol. 52, p. 103006, aug 2012.

[11] F. Reimold, M. Wischmeier, M. Bernert, S. Potzel, D. Coster, X. Bonnin, D. Reiter, G. Meisl, A. Kallenbach, L. Aho-Mantila, and U. Stroth, "Experimental studies and modeling of complete h-mode divertor detachment in ASDEX Upgrade," Journal of Nuclear Materials, vol. 463, pp. 128–134, 2015.

[12] W. Dekeyser, X. Bonnin, S. Lisgo, R. Pitts, D. Brunner, B. LaBombard, and J. L. Terry, "SOLPS-ITER study of neutral leakage and drift effects on the alcator C-Mod divertor plasma," Nuclear Materials and Energy, vol. 12, pp. 899–907, 2017.

[13] F. Subba, D. Coster, M. Moscheni, and M. Siccinio, "SOLPS-ITER modeling of divertor scenarios for EU-DEMO," Nuclear Fusion, vol. 61, p. 106013, sep 2021.

[14] S. Braginskii, "Transport processes in a plasma," Reviews of Plasma Physics, edited by M. A. Leontovich, Consultants Bureau, New York, p.1, 1965.

[15] D. Reiter, M. Baelmans, and P. Börner, "The EIRENE and B2-EIRENE codes," Fusion Science and Technology, vol. 47, no. 2, pp. 172–186, 2005.

[16] N. Horsten, G. Samaey, and M. Baelmans, "Development and assessment of 2D fluid neutral models that include atomic databases and a microscopic reflection model," Nuclear Fusion, vol. 57, no. 11, p. 116043, 2017.

[17] A. Xuereb, M. Groth, K. Krieger, O. Asunta, T. Kurki-Suonio, J. Likonen, and D. Coster, "DIVIMP-B2-EIRENE modelling of 13C migration and deposition in ASDEX Upgrade L-mode plasmas," Journal of Nuclear Materials, vol. 396, no. 2, pp. 228–233, 2010.

[18] M. Baelmans, M. Blommaert, J. D. Schutter, W. Dekeyser, and D. Reiter, "Efficient parameter estimation in 2D transport models based on an adjoint formalism," Plasma Physics and Controlled Fusion, vol. 56, p. 114009, oct 2014.

[19] W. Dekeyser, D. Reiter, and M. Baelmans, "Automated divertor target design by adjoint shape sensitivity analysis and a one-shot method," Journal of Computational Physics, vol. 278, pp. 117–132, 2014.

[20] M. Blommaert, M. Baelmans, W. Dekeyser, N. Gauger, and D. Reiter, "A novel approach to magnetic divertor configuration design," Journal of Nuclear Materials, vol. 463, pp. 1220–1224, 2015.

[21] Q. Wang, Uncertainty quantification for unsteady fluid flow using adjoint-based approaches. PhD Thesis, Stanford University, 2008.

[22] A. Griewank and A. Walther, Evaluating Derivatives. SIAM, 2008.

[23] S. Carli, M. Blommaert, W. Dekeyser, and M. Baelmans, "Sensitivity analysis of plasma edge code parameters through algorithmic differentiation," Nuclear Materials and Energy, vol. 18, pp. 6–11, 2019.

[24] L. Beda, L. Korolev, N. Sukkikh, and T. Frolova, Programs for Automatic Differentiation. Academy of Science, Moscow, 1959.

[25] R. Wengert, "A simple automatic derivative evaluation program," Communications of the ACM, vol. 7, no. 8, pp. 463–464, 1964.

[26] M. Iri, T. Tsuchiya, and M. Hoshi, "Automatic computation of partial derivatives and rounding error estimates with applications to large scale systems of nonlinear equations," Journal of Computational and Applied Mathematics, vol. 24, no. 3, pp. 365–392, 1988.

[27] W. Baur and V. Strassen, "The complexity of partial derivatives," Theoretical Computer Science, vol. 22, no. 2, pp. 317–330, 1983.

[28] A. Griewank, On automatic differentiation. Kluwer, 1989.

[29] F. Alauzet, S. Borel-Sandou, L. Daumas, A. Dervieux, Q. Dinh, S. Kleinveld, A. Loseille, Y. Mesri, and G. Rogé, "Multimodel design strategies applied to sonic boom reduction," European Journal of Computational Mechanics, vol. 17, no. 1-2, pp. 245–269, 2008.

[30] A. Hück, C. Bischof, M. Sagebaum, N. R. Gauger, B. Jurgelucks, E. Larour, and G. Perez, "A usability case study of algorithmic differentiation tools on the ISSM ice sheet model," Optimization Methods and Software, vol. 33, no. 4-6, pp. 844–867, 2018.

[31] N. McGreivy, S. Hudson, and C. Zhu, "Optimized finite-build stellarator coils using automatic differentiation," Nuclear Fusion, vol. 61, no. 2, p. 026020, 2021.

[32] "autodiff.org, community portal for automatic differentiation," 2022. Available from: `http://www.autodiff.org` [last accessed 24 March 2022].

[33] L. Hascoët and V. Pascual, "The tapenade automatic differentiation tool: principles, model, and specification," ACM Transactions on Mathematical Software, vol. 39, no. 3, p. 20, 2013.

[34] C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. D. Hovland, "ADIFOR: Generating derivative codes from Fortran programs," Scientific Programming, vol. 1, no. 1, pp. 11–29, 1992.

[35] R. Giering, T. Kaminski, and T. Slawig, "Generating efficient derivative code with TAF: Adjoint and tangent linear Euler flow around an airfoil," Future Generation Computer Systems, vol. 21, no. 8, pp. 1345–1355, 2005.

[36] B. Dauvergne and L. Hascoët, "The data-flow equations of checkpointing in reverse automatic differentiation," in Computational Science – ICCS 2006 (V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, eds.), (Berlin, Heidelberg), pp. 566–573, Springer Berlin Heidelberg, 2006.

[37] M. Sagebaum, T. Albring, and N. Gauger, "High-Performance Derivative Computations using CoDiPack," ACM Transactions on Mathematical Software, vol. 45, no. 4, 2019.

[38] A. Walther, A. Griewank, and O. Vogel, "ADOL-C: Automatic Differentiation Using Operator Overloading in C++," Proceedings in Applied Mathematics and Mechanics, vol. 2, pp. 41–44, 2003.

[39] B. Maugars, S. Bourasseau, C. Content, B. Michel, B. Berthoul, J. N. Ramirez, P. Raud, and L. Hascoët, "Algorithmic Differentiation for an efficient CFD solver," in ECCOMAS 2022 - 8th European Congress on Computational Methods in Applied Sciences and Engineering, (Oslo, Norway), June 2022.

[40] V. Rozhansky, E. Kaveeva, P. Molchanov, I. Veselova, S. Voskoboynikov, D. Coster, G. Counsell, A. Kirk, S. Lisgo, and and, "New b2solps5.2 transport code for h-mode regimes in tokamaks," Nuclear Fusion, vol. 49, no. 2, p. 025007, 2009.

[41] X. Bonnin, D. Coster, O. Wenisch, K. Kukushkin, A. Kukushkin, M. Stanojevic, and S. Voskoboyinikov, "Solps-iter user manual," tech. rep., ITER Organization, 2022.

[42] B. Christianson, "Reverse accumulation and implicit functions," Optimization Methods and Software, vol. 9, no. 4, pp. 307–322, 1998.

[43] M. Giles, "An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation," Tech. Rep. 08/01, Oxford University Computing Laboratory, Oxford, England, 2008.

[44] "Tapenade users website," 2022. Available from: `https://team.inria.fr/ecuador/en/tapenade` [last accessed 24 March 2022].

[45] W. Dekeyser, P. Boerner, S. Voskoboynikov, V. Rozhanksy, I. Senichenkov, L. Kaveeva, I. Veselova, E. Vekshina, X. Bonnin, R. Pitts, and M. Baelmans, "Plasma edge simulations including realistic wall geometry with SOLPS-ITER," Nuclear Materials and Energy, vol. 27, p. 100999, 2021.

[46] W. Dekeyser, X. Bonnin, S. W. Lisgo, R. A. Pitts, and B. LaBombard, "Implementation of a 9-point stencil in SOLPS-ITER and implications for Alcator C-Mod divertor plasma simulations," Nuclear Materials and Energy, vol. 18, pp. 125–130, 2019.

[47] M. E. Rensink, L. Lodestro, G. D. Porter, T. D. Rognlien, and D. P. Coster, "A comparison of neutral gas models for divertor plasmas," Contributions to Plasma Physics, vol. 38, no. 1-2, pp. 325–330, 1998.

[48] S. Carli, W. Dekeyser, R. Coosemans, R. Dejarnac, M. Komm, M. Dimitrova, J. Adámek, P. Bílková, and P. Böhm, "Interchange-turbulence-based radial transport model for SOLPS-ITER: a COMPASS case study," Contributions to Plasma Physics, vol. 60, no. 5-6, p. e201900155, 2020.

[49] S. Carli, "Replication Data for: Algorithmic Differentiation for adjoint sensitivity calculation in plasma edge codes," 2022.

[50] A. Griewank and D. Kressner, "Time-lag Derivative Convergence for Fixed Point Iterations," Revue ARIMA, vol. Numero Special CARI'04, pp. 87–102, 2005.

[51] A. W. Leonard, "Plasma detachment in divertor tokamaks," Plasma Physics and Controlled Fusion, vol. 60, p. 044001, feb 2018.

[52] P. C. Stangeby, "Basic physical processes and reduced models for plasma detachment," Plasma Physics and Controlled Fusion, vol. 60, p. 044022, mar 2018.

[53] A. Smolders, M. Wensing, S. Carli, H. D. Oliveira, W. Dekeyser, B. P. Duval, O. Février, D. Gahle, L. Martinelli, H. Reimerdes, C. Theiler, K. Verhaegh, and the TCV team, "Comparison of high density and nitrogen seeded detachment using SOLPS-ITER simulations of the tokamak á configuration variable," Plasma Physics and Controlled Fusion, vol. 62, no. 12, p. 125006, 2020.

[54] I. Ivanova-Stanik, M. Poradziński, R. Zagórski, and M. Siccinio, "Analyses of the influence of the recycling coefficient on He confinement in DEMO reactor," Fusion Engineering and Design, vol. 146, pp. 2021–2025, 2019.

[55] B. LaBombard, M. Umansky, R. Boivin, J. Goetz, J. Hughes, B. Lipschultz, D. Mossessian, C. Pitcher, J. Terry, and A. Group, "Cross-field plasma transport and main-chamber recycling in diverted plasmas on Alcator C-Mod," Nuclear Fusion, vol. 40, pp. 2041–2060, dec 2000.

[56] S. Carli, W. Dekeyser, M. Blommaert, R. Coosemans, W. Van Uytven, and M. Baelmans, "Bayesian MAP-estimation of κ turbulence model parameters using Algorithmic Differentiation in SOLPS-ITER," Contributions to Plasma Physics, p. e202100184, 2021.

[57] D. N. Goldberg, S. H. K. Narayanan, L. Hascoet, and J. Utke, "An optimized treatment for algorithmic differentiation of an important glaciological fixed-point problem," Geoscientific Model Development, vol. 9, no. 5, pp. 1891–1904, 2016.

[58] W. Dekeyser, M. Blommaert, K. Ghoos, N. Horsten, P. Boerner, G. Samaey, and M. Baelmans, "Divertor design through adjoint approaches and efficient code simulation strategies," Contributions to Plasma Physics, vol. 58, no. 6-8, pp. 643–651, 2018.

[59] E. Løvbak, F. Blondeel, A. Lee, L. Vanroye, A. V. Barel, and G. Samaey, "Reversible random number generation for adjoint monte carlo simulation of the heat equation," 2023.

[60] S. Carli, "Code for: Algorithmic Differentiation for adjoint sensitivity calculation in plasma edge codes," 2022.

[61] P. C. Stangeby, "The bohm–chodura plasma sheath criterion," Physics of Plasmas, vol. 2, no. 3, pp. 702–706, 1995.