Real-time implementation of the high-fidelity NBI code RABBIT into the discharge control system of ASDEX Upgrade

OPEN ACCESS
IOP Publishing | International Atomic Energy Agency Nuclear Fusion
Nucl. Fusion **63** (2023) 066013 (15pp) https://doi.org/10.1088/1741-4326/acc31c

Real-time implementation of the high-fidelity NBI code RABBIT into the discharge control system of ASDEX Upgrade

M. Weiland[1,*], R. Bilato[1], B. Sieglin[1], F. Felici[2,3], L. Giannone[1], O. Kudlacek[1], M. Rampp[4], M. Scheffer[2], W. Treutterer[1], T. Zehetbauer[1], the ASDEX Upgrade Team[a] and the EUROfusion MST1 Team[b]

[1] Max-Planck-Institut für Plasmaphysik, Garching, Germany
[2] Technische Universiteit Eindhoven, Netherlands
[3] École polytechnique fédérale de Lausanne, Switzerland
[4] Max Planck Computing and Data Facility (MPCDF), Garching, Germany

E-mail: markus.weiland@ipp.mpg.de

Received 26 December 2022, revised 24 February 2023
Accepted for publication 10 March 2023
Published 19 April 2023

Abstract

For the first time, a real-time capable NBI code, which has a comparable fidelity to the much more computationally expensive Monte Carlo codes such as NUBEAM, has been coupled to the discharge control system of a tokamak. This implementation has been done at ASDEX Upgrade and is presented in this paper. Modifications to the numerical scheme of RABBIT for the time-dependent solution of the Fokker–Planck equation have been carried out to make it compatible with the non-equidistant time-steps, as they occur in real-time simulations. We demonstrate that this allows RABBIT to run in real-time both in a steady-state and time-dependent fashion and show and discuss an actual real-time simulation. Its accuracy is identified by comparing to offline RABBIT and TRANSP-NUBEAM runs (where more diagnostics are available for preciser inputs).

Keywords: tokamak, plasma control, Fokker-Planck, numerical noise, non-constant time-steps, fast ions, energetic particles

(Some figures may appear in colour only in the online journal)

[a] See Stroth *et al* 2022 (https://doi.org/10.1088/1741-4326/ac207f) for the ASDEX Upgrade Team.
[b] See Labit *et al* 2019 (https://doi.org/10.1088/1741-4326/ab2211) for the EUROfusion MST1 Team.
[*] Author to whom any correspondence should be addressed.

1. Introduction

The success of future fusion devices will rely heavily on sophisticated real-time control of the plasma behavior. It is therefore of utmost importance to develop and test control schemes already in present-day machines. At the same time, this will allow more enhanced experimental designs already in the present machines, when quantities of importance to a particular experiment—e.g. the q-profile for MHD or advanced scenario studies—can be controlled directly in real-time.

At ASDEX Upgrade, the discharge control system (DCS) is used for plasma control. It is a flexible system, which allows additional codes to run as satellites [1]. These codes are crucial for estimating the machine behavior, based on measurements: the JANET code [2, 3] is used for equilibrium reconstructions, the 1D transport solver RAPTOR [4] is used to estimate the plasma state. The electron cyclotron resonance heating code TORBEAM [5] has been added to calculate the heating deposition and current drive, which has allowed to demonstrate control of neo-classical tearing modes [6].

To further improve the plasma state reconstructions, a code to simulate neutral beam injection (NBI) is needed: heating profiles, current drive, particle sources etc are essential to make the RAPTOR transport calculations more accurate. Fast-ion pressure and current drive are important to further improve the accuracy of the JANET equilibrium reconstructions. The RABBIT code [7, 8] has recently been developed for exactly these purposes, and its implementation into the DCS is the subject of this paper.

2. The RABBIT code and its interface

The RABBIT code consists mainly of three parts: first, the beam deposition is calculated on the beam center-line with a semi-analytic correction for finite beam width. Second, effects of the first fast-ion orbit are taken into account by averaging the deposition over the first orbit. Hereby the orbits are calculated fully realistically with a 4th-order Runge–Kutta method. To still achieve real-time capability, the number of orbits is strongly reduced to typically 20 orbits per beam and energy component. An optimized inter- and extrapolation technique is used subsequently to obtain all needed orbit information. Third, the orbit-averaged (and hence flux-surface averaged) 1D deposition profile serves as input into a semi-analytic, time-dependent solution of the Fokker–Planck equation. With these assumptions, still good agreement can be obtained with the NUBEAM Monte Carlo code. This has been demonstrated for large numbers of ASDEX Upgrade, JET and DIII-D discharges [7–9].

RABBIT is written in Fortran2008. To prepare the code for usage in real-time, the code is compiled into a library (a shared object), which provides three main subroutines as interface: `rabbit_lib_init`, a function which is supposed to be called before the discharge and initializes RABBIT (i.e. allocates memory), `rabbit_lib_step`, which is to be called consecutively during the discharge and `rabbit_lib_deinit`, which is to be called at the end of the discharge and frees all the memory.

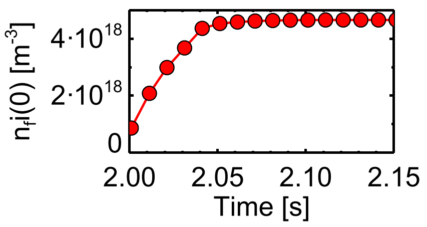


Figure 1. The core fast-ion density as calculated by RABBIT in several calls to the `rabbit_lib_step` function with identical inputs. The result evolves over time because the state of the fast-ion distribution is stored internally in the RABBIT library, and it takes one slowing down time for the fast-ion density to reach a steady state.

The arguments of `rabbit_lib_init` are parameters, which are known before the discharge. This includes the plasma and beam species, NBI geometries, grid sizes and a string containing the filename of a namelist where more specialized code settings are defined. The arguments of the `rabbit_lib_step` function represent then inputs and outputs. The inputs classify into three groups: core profiles, equilibrium and NBI time-traces, for the latter, we allow for time-dependent injection power, energy and energy fraction mixture. The three remaining inputs are: 1) The fast-ion density profile, which is both an input and also an output. This resembles a weak non-linearity in the code, as the fast-ions are considered for diluting the plasma in the collision operator (by displacing thermal ions). For most applications it is sufficient to use the fast-ion density of the previous time-step as input for the subsequent one. 2+3): The desired time step and the output timing. The output timing is thought as a dimensionless number from 0 to 1, and it controls the time when outputs are desired with respect to the finite width of the time bin. In real-time this will usually be 1, as one wants to estimate the state of the fast ions at the end of the time-step. In typical offline simulations, one would rather use 0.5 (centered time bins).

The current state of the fast-ions is kept internally in RABBIT, such that subsequent calls to the step function will yield different results even for identical inputs, as is illustrated in figure 1: If the time-step is smaller than the fast-ion slowing down time, the fast-ion density first builds up over several calls of the step function, until it reaches a steady state.

These library functions are made interoperable with C by using the Fortran-intrinsic iso_c_binding module. It was not possible to expose the internal RABBIT data structures and classes to C, since C cannot cope with the allocatable arrays that they contain. Instead, we restrict ourselves to elementary data types for the interface, i.e. strings, floating point or integer numbers, both as scalars and arrays of 1–2 dimensions. This has the down-side of leading to long argument lists but is conceptually simple. For example, the poloidal flux matrix is received on the Fortran side as a 2D array Psi(m, n), which is given from the C side as a pointer double* Psi. m and n are passed as additional arguments, to ensure that the dimensions on the C side are matched correctly.

Around these basic C functions, a C++ wrapper has been written to re-group the inputs again into classes and provide an abstraction layer. To make the handling of matrices more convenient, a class FortranMatrix has been written to handle the conversion between Fortran column-major and C row-major memory storage.

3. Coupling with the DCS

The DCS is structured into a core part, and several satellites, which can be added or removed in a flexible way. Such satellites can be e.g. diagnostics, actuators, or, in our case, evaluation codes. Each satellite consists of one or more software units, which are called Application Processes (APs) and are written in C++ [1].

For the RABBIT coupling, two strategies were initially discussed: simulating each NBI beam independently within an AP or simulating all beams together in one AP. The first approach would have allowed a simpler setup of the AP, since it needs to deal only with one beam. However, the satellite as a whole would have become more complicated: the eight beams of AUG would require 8 instances of the RABBIT-AP, and an additional AP for adding the results. In addition, the fast-ion calculations cannot be fully separated between each individual beam: e.g., the neutron rate calculations are not fully independent of each other, because for beam–beam neutron reactions, the distribution function needs to be added (rather than adding simply the neutron rates). As a consequence, we opted for the second approach, using only one AP to simulate all eight beams together and at once. This allows to treat beam–beam interactions internally in the Fortran code, thus avoiding unnecessary external data transfers. Still, all RABBIT outputs are given on a per beam basis, which is useful for control applications, where it is crucial to know the impact of an individual beam.

The RABBIT-AP runs on a dedicated computer with two 12-core CPUs (Intel Xeon Gold 6146 CPU @ 3.20 GHz, 192 GB RAM). The communication with DCS is done via network, and one CPU is entirely reserved for this purpose, ensuring a deterministic real-time behavior on the second CPU, where the code is running. The RABBIT code is parallelized with OpenMP, using one thread per neutral beam (i.e. eight threads in total for ASDEX Upgrade).

A main advantage of the satellite concept is that RABBIT can run asynchronously with the DCS main cycle, which is typically 1.5 ms and therefore shorter than the typical RABBIT calculation time. RABBIT simply runs as fast as it can (timings are reported in section 7), then publishes its results to the DCS and asks for new inputs. Even in the unlikely event of an unforeseen termination of the RABBIT code (e.g. a crash or unhandled exception), the main DCS can continue to run and is not affected other than no longer receiving new results from the RABBIT satellite.

The in- and output signals are configured independently of the AP source code in xml files. An overview is given in figure 2 and table 1. The real-time equilibrium is calculated by the JANET code. A few conversions are needed to obtain inputs in a format suitable for RABBIT:

JANET compresses the flux matrix via a discrete cosine transformation. This needs to be inverted again for RABBIT to get an R-z flux matrix. In addition, the conversion from COCOS $= 13$ [10] (JANET) to COCOS $= 5$ (RABBIT) is done (by dividing with -2π). For the 1D profiles, the following set of integrals is necessary:

$$\text{Poloidal flux } \Psi = \frac{-1}{2\pi}\Psi^{(J)},\ \frac{\mathrm{d}\Psi}{\mathrm{d}V} = \frac{-1}{2\pi}\frac{\mathrm{d}\Psi}{\mathrm{d}V}^{(J)} \tag{1}$$

$$\text{`Helper' quantity } H \equiv 2\pi c_\star \frac{\mathrm{d}\Psi}{\mathrm{d}V} \tag{2}$$

$$\text{Inverse of safety factor } \iota = \frac{2\pi H}{F^{(J)}\,\langle 1/R^2\rangle^{(J)}} \tag{3}$$

$$\text{Toroidal flux } \Phi = -2\pi \int_0^{\Psi} \iota^{-1}\mathrm{d}\Psi' \tag{4}$$

$$\text{Enclosed volume } V = \int_0^{\Psi}\left(\frac{\mathrm{d}\Psi}{\mathrm{d}V}\right)^{-1}\mathrm{d}\Psi' \tag{5}$$

Enclosed area of the poloidal cross-section

$$A = c_\star \int_0^{\Psi}\left(\frac{H}{\langle 1/R\rangle^{(J)}}\right)^{-1}\mathrm{d}\Psi'. \tag{6}$$

Hereby, the quantities from JANET are marked with $^{(J)}$. $c_\star = (2\pi)^{-e_{B_p}}\sigma_{\rho\theta\varphi}\sigma_{B_p} = -1$, with the quantities in the intermediate step as in [10]. Note that all three integrals take the form $\mathrm{d}Z = X^{-1}\mathrm{d}\Psi$, where in diverted plasmas $X = 0$ at the separatrix and hence X^{-1} diverges. To compute this integral on a fixed grid Ψ_i, we integrate $X\mathrm{d}Z = \mathrm{d}\Psi$ instead, yielding:

$$\int_{Z_i}^{Z_{i+1}} X\mathrm{d}Z' = \int_{\Psi_i}^{\Psi_{i+1}} \mathrm{d}\Psi \tag{7}$$

which is approximated by the trapezoid rule as:

$$\frac{1}{2}(X_i + X_{i+1})(Z_{i+1} - Z_i) = \Psi_{i+1} - \Psi_i \tag{8}$$

yielding the following recursive relation

$$Z_{i+1} = Z_i + 2\frac{\Psi_{i+1} - \Psi_i}{X_i + X_{i+1}} \tag{9}$$

which can be solved iteratively, then. With this numerical scheme we evaluate the integrals (3)–(6) and thus get all equilibrium inputs for RABBIT. RABBIT then uses the square-root of normalized toroidal flux as radial coordinate ($\rho_{\mathrm{tor}} = \sqrt{\frac{\Phi - \Phi_{\mathrm{axis}}}{\Phi_{\mathrm{lcfs}} - \Phi_{\mathrm{axis}}}}$, where lcfs stands for last closed flux surface).

4. Real-time input signals and comparison to offline signals

In real-time, there are fewer diagnostics available than offline (due to stronger requirements in terms of data acquisition). In addition, there is less CPU time available for data evaluation. In this section, we briefly compare real-time and offline input signals for RABBIT and discuss impacts on the accuracy of the real-time implementation of RABBIT.

Figure 3 shows a comparison between the real-time equilibrium code JANET and the offline IDE [11] calculation. The

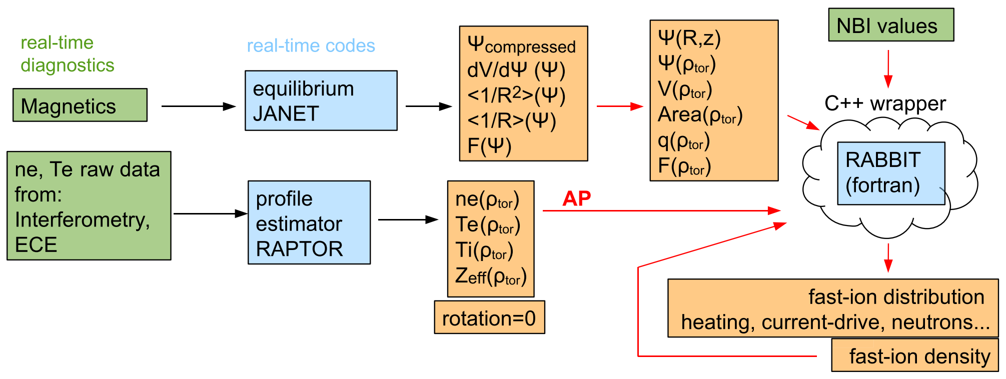

real-time diagnostics
real-time codes
Magnetics
equilibrium JANET
Ψcompressed dV/dΨ (Ψ) <1/R2>(Ψ) <1/R>(Ψ) F(Ψ)
Ψ(R,z) Ψ(ρtor) V(ρtor) Area(ρtor) q(ρtor) F(ρtor)
NBI values
C++ wrapper
RABBIT (fortran)
ne, Te raw data from: Interferometry, ECE
profile estimator RAPTOR
ne(ρtor) Te(ρtor) Ti(ρtor) Zeff(ρtor)
AP
rotation=0
fast-ion distribution heating, current-drive, neutrons...
fast-ion density

Figure 2. Overview of RABBIT in- and outputs within the DCS of ASDEX Upgrade. Red arrows are handled by the Application Process (AP).

Table 1. Overview of RABBIT outputs. nsource is the number of NBI sources, and nrhotor_out the number of radial bins of the output-grid, which can be configured freely and is typically set to 20.

Variable	Dimensions	Explanation	Unit
powe	nrhotor_out, nsource	Power density profile to e^{-}	W m^{3}
powi	nrhotor_out, nsource	Power density profile to ions	W m^{3}
press	nrhotor_out, nsource	Fast-ion pressure	Pa
bdep	nrhotor_out, nsource	Fast-ion deposition (orbit-averaged)	1 m^{-3} s^{-1}
bdens	nrhotor_out, nsource	Fast particle density	1 m^{3}
jfi	nrhotor_out, nsource	Fast-ion current density	A m^{2}
jnbcd	nrhotor_out, nsource	Driven current density (fast ion incl. electron shielding)	A m^{2}
torqe	nrhotor_out, nsource	Torque density to e^{-}	Nm m^{3}
torqi	nrhotor_out, nsource	Torque density to ions	Nm m^{3}
torqth	nrhotor_out, nsource	Thermalization torque density	Nm m^{3}
torqjxb	nrhotor_out, nsource	$\vec{j} \times \vec{B}$ torque density	Nm m^{3}
nrate	nrhotor_out	Neutron rate	1 m^{-3} s^{-1}
rhotor_out	nrhotor_out	Normalized rho toroidal grid on which outputs are given	[-]
nrhotor_out	1	number of points on equidistant rho toroidal output grid	[-]
powe_tot	nsource	Total deposited neutral beam power to electrons	W
powi_tot	nsource	Total deposited neutral beam power to ions	W
Pshine	nsource	Shine-thru power	W
Prot	nsource	Power to rotation	W
Ploss	nsource	Beam power losses in SOL	W
Pcx	nsource	Internal charge exchange loss	W
Inbcd	nsource	Total driven neutral beam current	A
ierr	nsource	Error flag (per beam)	
		optional outputs:	
wfipar	nrhotor_out, nsource	Fast-ion energy density (parallel component, plasma frame)	J m^{3}
wfiparlab	nrhotor_out, nsource	Fast-ion energy density (parallel component, lab frame)	J m^{3}
wfiperp	nrhotor_out, nsource	Fast-ion energy density (perpendicular component)	J m^{3}

agreement of the toroidal flux matrix is quite good, especially the seperatrix is very similar. The inner contours of the square-root of normalized poloidal flux, ρ_{pol}, are however, quite shifted between the codes. This is likely due to difference (and uncertainties) of the safety factor profile (q) and it illustrates that ρ_{tor} is in the core plasma a more reliable radial coordinate than ρ_{pol}, especially when different equilibrium codes are involved.

RABBIT uses ρ_{tor} as radial coordinate, and is in addition, (as any NBI fast-ion simulation) not super-sensitive to fine details of the equilibrium such as the exact position of the magnetic axis or the q-profile. Thus, the real-time equilibrium should not introduce any major lack of accuracy into the RABBIT calculations.

The kinetic profiles n_{e} and T_{e} are provided by the RAPTOR code, constrained by its internal transport model and by measurements with real-time interferometry and ECE, respectively. These two are the 'main players' amongst the kinetic profiles, since they enter heavily in the involved physics processes such as beam attenuation (n_{e}) and slowing down (n_{e} and T_{e}). Hence, for RABBIT it is very good that real-time measurements are available for those. On the other side, there are additional diagnostics available offline (e.g. Thomson scattering, see table 2) which introduces a potential source of inaccuracy.

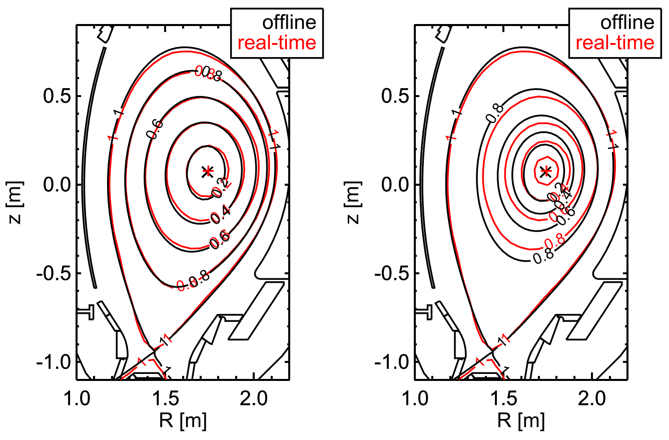


Figure 3. Comparison of the real-time equilibrium (JANET) and an offline equilibrium (here: calculated by IDE [11]) for ASDEX Upgrade discharge 36523 at 2.60 s. Left and right: contours of ρ_{tor} and ρ_{pol}, i.e. the square-root of the normalized toroidal and poloidal flux, respectively.

Table 2. Overview of available input signals for the kinetic profiles to RABBIT.

Profile	Real-time diagnostics	Additional offline diagnostics
ne	Interferometer (5 chords)	Thomson scattering
Te	ECE	Thomson scattering
Ti	None (Ti=Te)	CXRS
rotation	None (=0)	CXRS
Zeff	None (fixed value)	fixed value (CXRS)

Currently, no real-time charge exchange measurements are available on ASDEX Upgrade, such that we set $T_{\mathrm{i}} = T_{\mathrm{e}}$ and the toroidal plasma rotation $\omega_{\mathrm{tor}} = 0$. The approximation '$T_{\mathrm{i}} = T_{\mathrm{e}}$' should have no significant impact on RABBIT accuracy for most applications. T_{i} enters most strongly into the thermo-nuclear part of the neutron calculation. Since thermo-nuclear neutrons usually make up only a few percents of the total ASDEX Upgrade neutron rate during NBI, this is also acceptable and would (ironically) lead only to larger inaccuracies when NBI is switched off (thus thermo-nuclear reactions being the only source of neutrons).

The plasma rotation is a loss channel, since the injection energy is reduced in the plasma frame of reference (as demonstrated in a TRANSP-NUBEAM calculation in figure 4). The 'lost' NBI power is used up to sustain the plasma rotation. For typical ASDEX Upgrade rotations, losses can be up to 10% for heating, 20% for neutrons and 30% for current drive. Therefore, it would be beneficial to have measurements or at least an educated guess for ω_{tor}. Initial tests indicate that a suitable guess could be made by calculating the NBI torque within RABBIT and making an assumption on the momentum confinement time. A detailed study of this is currently ongoing.

Finally, a profile of the effective charge number Z_{eff} is needed as input for RABBIT. Since ASDEX Upgrade is a tungsten machine, the plasmas are usually very clean and thus Z_{eff} is low. Technically the Z_{eff} profile is provided by RAPTOR, which delivers (due to lack of diagnostics) only a fixed

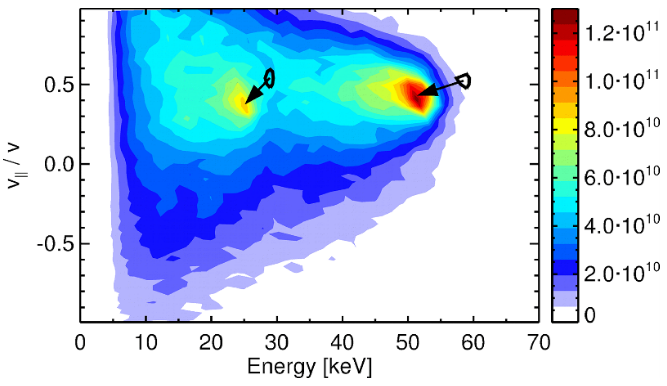


Figure 4. Fast ion distribution function at the magnetic axis as calculated by NUBEAM, transferred to the plasma frame of reference (i.e. rotating with same velocity as the plasma). Colored contours correspond to a toroidal rotation of 263 km s^{-1}, which is equal to a rotation frequency of 24.8 kHz. Black contours show the full and half energy peaks of the distribution function for a 0 rotation calculation. A clear shift of the peaks towards lower energies and pitches $v_{||}/v$ is visible (as depicted by the arrows).

value of $Z_{\mathrm{eff}} = 1.5$. We believe that this is sufficient in terms of accuracy (especially when there is no dedicated impurity seeding), and it is also a common assumption even for offline analysis.

5. Treatment of time-dependent simulations in real-time

The numerical scheme which RABBIT uses to allow time-dependent simulations of the distribution function is explained in detail in [7, 8]. Although not explicitly stated therein, it was initially only designed for equidistant time-steps Δt. In real-time, RABBIT runs asynchronously with respect to the main DCS cycle, which means that it calculates as fast as possible. The RABBIT execution time will typically fluctuate and hence the time-step Δt will not be constant either, so the numerical scheme had to be modified to be compatible with this.

5.1. Modification of the Fokker-Planck time-dependence scheme

The numerical scheme is based on pulses of fast-ions, as illustrated in figure 5. For a pulse of fast-ions injected into the system with a velocity v_{start} and a rate S (in units [1 m^{-3} s^{-1}]), we calculate how far it will slow down during the time-step Δt via:

$$v_{\mathrm{final}}^3 = (v_{\mathrm{start}}^3 + v_{\mathrm{c}}^3) \cdot \exp\frac{-3\Delta t}{\tau_{\mathrm{s}}} - v_{\mathrm{c}}^3. \tag{10}$$

Here, v_{c} is the critical velocity (collisions with electrons dominate above v_{c}, and collisions with thermal ions below) and τ_{s} the Spitzer slowing-down time. In figure 5, the red pulse is injected in the first time-step.

In the next time-step, we continue to follow the pulse, keeping S constant. Now the 'new' v_{start} is equal to v_{final} from the previous step. If the NBI is still switched on, another new pulse

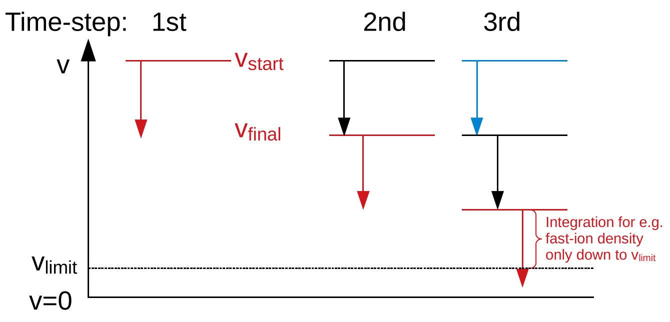


Figure 5. The time-dependent numerical scheme of RABBIT with equidistant time-steps. The whole velocity space is covered smoothly.

is injected at nominal injection velocity (black pulse) and a potentially different birth rate S. This scheme is continued. When v_{final} crosses zero, the pulse will be discarded in the next time-step.

The main outputs of RABBIT are integrals of the distribution function (which is proportional to S), and these are evaluated directly for each pulse:

$$\int_{\max(v_{\text{final}},v_{\text{limit}})}^{\max(v_{\text{start}},v_{\text{limit}})} f\ldots\mathrm{d}v \tag{11}$$

and then summed over all pulses. Here, $v_{\text{limit}}=\sqrt{2CT_{\text{i}}/m}$ is the threshold velocity at which fast-ions are considered ‘thermalized’. It can be chosen by the user by setting C (typically to 0 or to 1.5 to mimic NUBEAM behavior). It should be noted that for heating power the residual values between v_{limit} and $v=0$ are still calculated separately by RABBIT but added to the ion heating while for the torque, a separate ‘thermalization torque’ output exists (NUBEAM has both separate thermalization torque and power outputs).

Many of these integrals are even analytically solvable, e.g. the heating power density:

$$p=\frac{Sm}{2}\left(\max(v_{\text{start}},v_{\text{limit}})^2-\max(v_{\text{final}},v_{\text{limit}})^2\right) \tag{12}$$

or the fast-ion density:

$$n=\frac{S\tau_{\text{s}}}{3}\ln\frac{\max(v_{\text{start}},v_{\text{limit}})^3+v_{\text{c}}^3}{\max(v_{\text{final}},v_{\text{limit}})^3+v_{\text{c}}^3}\overset{\text{if: } v_{\text{final}}\geqslant v_{\text{limit}}}{=\!=} \tag{13}$$

$$\overset{\text{if: } v_{\text{final}}\geqslant v_{\text{limit}}}{=\!=} S\Delta t. \tag{14}$$

The last equality holds only if $v_{\text{final}}\geqslant v_{\text{limit}}$. Therefore the former expression (equation (13)) is implemented in the code.

Now, if Δt is kept constant throughout the simulation, as it is typically done in offline simulations, the numerical scheme works well. This can be understood e.g. by looking at figure 5 and noting that the entire velocity space is smoothly covered, or by considering $n=S\Delta t$ (from equation (14)), which guarantees that the number density of fast-ions is conserved while following a given fast-ion pulse from injection to thermalization.

With a varying Δt, this old numerical scheme will not work. To demonstrate this, we have performed a RABBIT simulation with all inputs completely constant, and switched off the fast-ion dilution and v dependence in the calculation of the Fokker–Planck coefficients, as this introduces weak non-linear effects, which are not part of the problem but would make understanding it more difficult. We have now executed this simulation twice, once with constant Δt, and once with a varying Δt (see figure 6(*a*)) as it happened during a real-time discharge (here, a non-optimized debug build was used which made the code significantly slower than usual). Figure 6(*b*) shows the two results. While the constant-Δt curve (black) goes quickly and smoothly into a steady-state (after full slowing down of the fast ions), the red curve shows wild noise and a large spike associated with the strong increase of Δt at 1.0 s. It should be noted that this strong increase of Δt was caused in the original discharge by switching on the beams, before RABBIT had essentially nothing to calculate and therefore returned much quicker. In a real situation, such a strong variation of Δt would hence not occur during an NBI-on phase, but it serves as a good demonstration case.

This behavior can be understood by the thought experiment illustrated in figure 7, where we make the case even simpler and let Δt be constant in all time-steps except for the 2nd, where we double it. Intuitively, it can be seen that something goes wrong because we get ‘overlaps’ of fast ion pulses in the 2nd time-step, and holes in later time steps. Since $n=S\Delta t$, the conservation of fast-ion density is violated. For pulses not hitting v_{limit}, for which $n=S\Delta t$ is valid, this can be corrected easily, by renormalizing $S=S_0\Delta t_0/\Delta t$ for each fast-ion pulse, where index 0 refers to the values at birth of the fast-ion pulse.

Consequently, this ansatz mitigates the problem (see green curve in figures 6(*b*) and (*c*)) but does not remove it completely. This is due to the pulses which cross the thermalization threshold v_{limit}. For example, in our thought experiment, figure 7, a false numerical noise would still occur in the 5th time step, because the ‘black’ pulse would have been discarded already in the 4th time step.

Therefore, we propose another solution. Instead of renormalizing the source, we change the velocity interval. If we rewrite the velocity propagator (comp. equation 10) as a function:

$$\tilde{v}^3(v_0,t)=\left(v_0^3+v_{\text{c}}^3\right)\cdot\exp\frac{-3t}{\tau_{\text{s}}}-v_{\text{c}}^3. \tag{15}$$

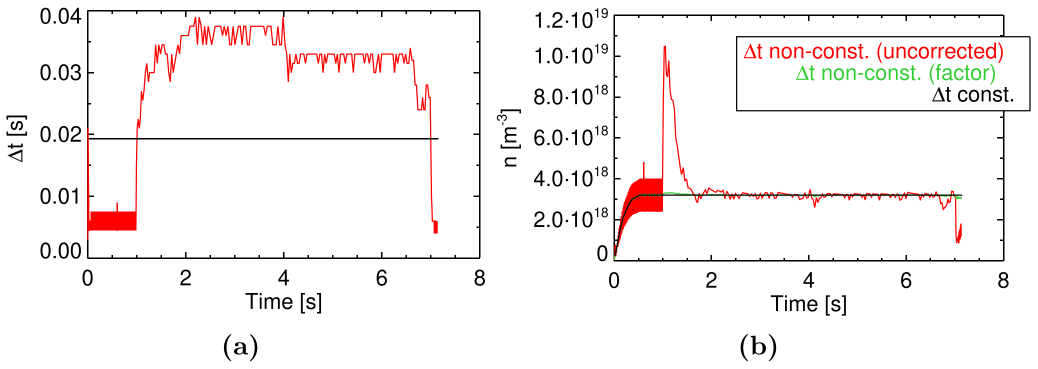
we calculate v_{final} as before:

$$v_{\text{final}}\mapsto\tilde{v}(v_{\text{start-old}},\Delta t) \tag{16}$$

but in addition, we (afterwards) also modify v_{start} (using $v_{\text{start-old}}$ as notation for the previous, un-modified value):

$$v_{\text{start}}\mapsto\tilde{v}(v_{\text{final}},-\Delta t_0). \tag{17}$$

For constant time-steps ($\Delta t=\Delta t_0$) the scheme is unchanged with respect to the original and already working version. For $\Delta t\neq\Delta t_0$, the v intervals of the new scheme are compared to



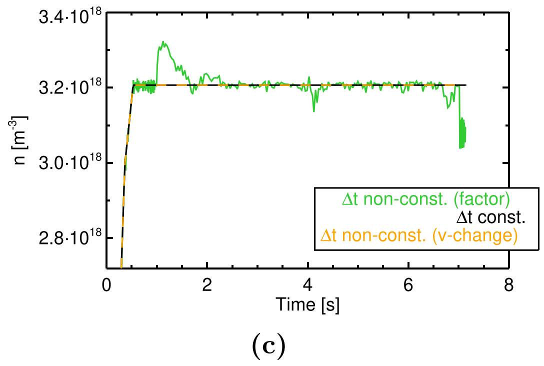


Figure 6. b) RABBIT calculations for the fast-ion density at $\rho_{\mathrm{tor}} = 0.625$ with constant time-step (black curve)) and varying time-step (red curve) as it may happen during real-time calculations. (*a*) shows the values of the time-steps. (*c*) shows a zoomed-in version of (*b*) with the red curve removed and in added orange-dashed lines the calculation with the updated time-dependence scheme (v-change correction), which is now compatible with non-equidistant time-steps and yields the same solution as for $\Delta t = const$.

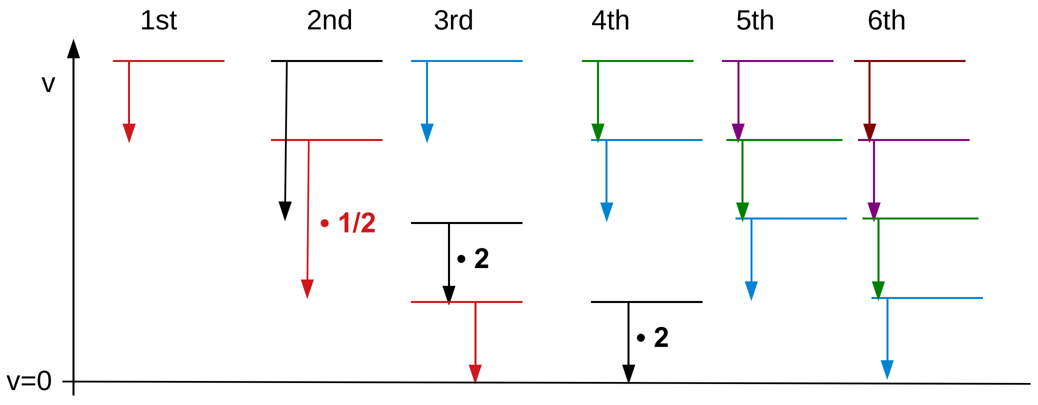


Figure 7. Thought experiment: The old time-dependent scheme of RABBIT with non-equidistant time-steps (2nd time-step twice as long). Overlaps and holes occur, leading to numerical noise. For simplicity, $v_{\mathrm{limit}} = 0$ has been assumed. The proposed multiplication factors would remove the numerical noise for the fast-ion density in time-steps 2–4 but not in time-step 5.

the previous (dashed) ones in figure 8 for our thought experiment. It can be seen that the overlaps and holes for intervals above $v = 0$ are eliminated. Once a pulse reaches $v = 0$, it is important to let alive the pulse even if $v_{\mathrm{final}} \leqslant 0$, because the above mechanism could bring v_{start} again above 0. Such a case occurs in figure 8 in the black fast-ion pulse from the 4th to the 5th step. Only once $v_{\mathrm{start}} \leqslant 0$ (after applying above correction) the pulse can be removed safely.

A further advantage of this scheme is that we can fix the numerical issues for all output quantities, because we solve the holes and gaps (and therefore discontinuities) of the distribution function. In contrast, the renormalization factor would have fixed only the number density conservation (for pulses above v_{limit}), but e.g. energy conservation would still have been violated.

Some output quantities depend on the pitch angle of the distribution function, e.g. the neutral beam current drive. In order to solve the numerical issue also for those quantities, not only the v-interval needs to be treated with a correction, but also the source term for each fast-ion pulse, which is represented as a Legendre series: $S(\xi) = \sum_{l=0}^{\infty} (l + \frac{1}{2}) S_l P_l(\xi)$. The reason for this is the pitch angle scattering: the fast-ion pulses (i.e. $S(\xi)$)

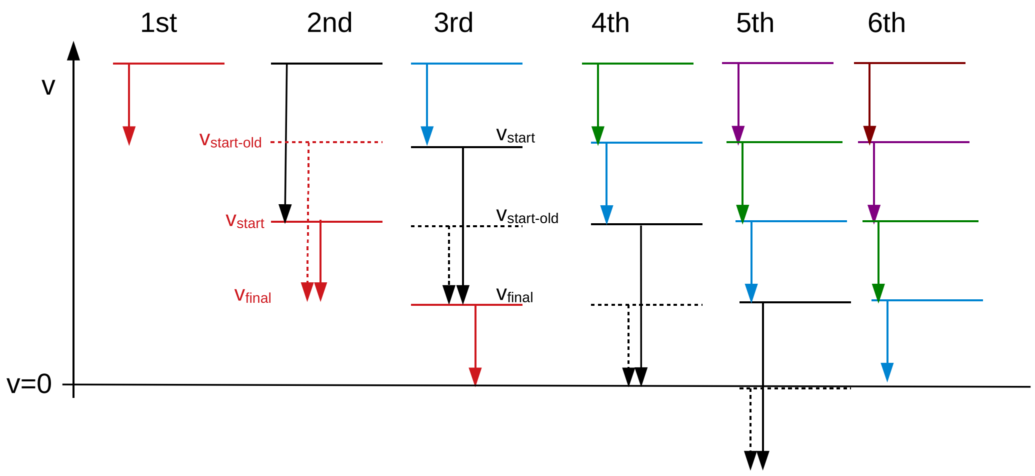


Figure 8. Updated time-dependent scheme of RABBIT (applied to the same thought experiment as in figure 7). The velocity intervals are modified from the dashed lines to the solid lines, eliminating all overlaps and holes.

become broader and more isotropic towards lower velocities: In between two velocities v_{start} and v_{final}, the change of $S(\xi)$ is given by $S_l \mapsto S_l \cdot \Xi_l(v_{\text{start}}, v_{\text{final}})$, with

$$\Xi_l(v_{\text{start}}, v_{\text{final}}) = \left(\frac{v_{\text{start}}^3 + v_c^3}{v_{\text{final}}^3 + v_c^3}\frac{v_{\text{final}}^3}{v_{\text{start}}^3}\right)^{\beta\frac{l(l+1)}{3}}. \quad (18)$$

In the old scheme, $S(\xi)$ was advanced at the end of each time-step:

$$S_l \mapsto S_l \cdot \Xi_l(v_{\text{start}}, v_{\text{final}}). \quad (19)$$

We keep this, using the modified v_{start} and v_{final}. In addition, we need at the beginning of a time-step, in conjunction with the v_{start}-modification, the following correction for S_l:

$$S_l \mapsto S_l \cdot \Xi_l(v_{\text{start-old}}, v_{\text{start}}). \quad (20)$$

It can well be that $v_{\text{start-old}} < v_{\text{start}}$ (comp. figure 8) such that we essentially go backwards in time and 'revert' a part of the pitch angle scattering during the time-step, for our integrals over the modified v-intervals. Still, it is guaranteed that the net-change of $S(\xi)$ during an entire time step is according to the 'physical' interval $[v_{\text{start-old}}, v_{\text{final}}]$, since in total:

$$\begin{aligned} S_l &\mapsto S_l \cdot \Xi_l(v_{\text{start-old}}, v_{\text{start}}) \cdot \Xi_l(v_{\text{start}}, v_{\text{final}}) \\ &= S_l \cdot \Xi_l(v_{\text{start-old}}, v_{\text{final}}). \end{aligned} \quad (21)$$

As shown in figure 9, this new scheme makes RABBIT compatible with non-equidistant time-steps also for outputs depending on the pitch, such as driven current and neutron emission.

5.2. Robustification and test under real plasma conditions

The improved time-dependence scheme, as described in the previous section, works well in the artificial test case with constant inputs, but for routine usage, additional numerical robustification is required. Especially when T_e is low (e.g. at the edge of the plasma), it may happen that τ_s becomes much smaller than the calculation time-step, such that the exponential function in the velocity propagator $\tilde{v}^3(v_0, t)$ (equation (15)), $\exp\frac{-3t}{\tau_s}$, becomes numerically unstable. Occasionally, this lead to extremely high velocities, tens of orders of magnitudes above the speed of light, or even $+$Inf.

To strengthen the code against this, we check in every fast-ion pulse if:

$$v_{\text{start-old}} = \tilde{v}\left(\tilde{v}(v_{\text{start-old}}, \Delta t), -\Delta t\right)(\text{within a fixed accuracy}) \quad (22)$$

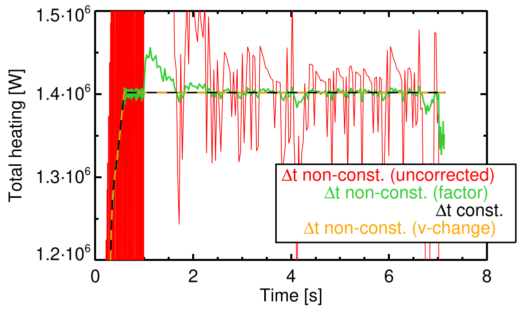
$$v_{\text{start}} = \tilde{v}(v_{\text{final}}, -\Delta t)\text{is finite} \quad (23)$$

$$v_{\text{start}} > v_{\text{final}}. \quad (24)$$

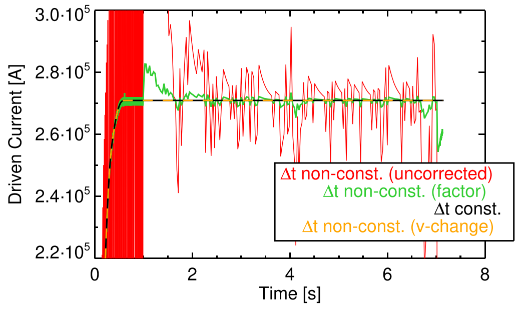
If any of those conditions is not met, we fall back to the 'factor' correction for this fast-ion pulse. In addition, we enforce that v_{start} is not higher than the birth velocity of the given fast-ion pulse (since this would be nonphysical).

We have tested this robustification in ASDEX Upgrade discharge 31216 0.2 s–1 s. The first test is a simulation with equidistant time-steps, where we compare two runs with activated and de-activated 'v-change' correction. Without the robustification, in the 'v-change' case the code would have stopped due to occurring NaNs. With it, the result agrees very well (see figure 10) for all radial bins and time points.

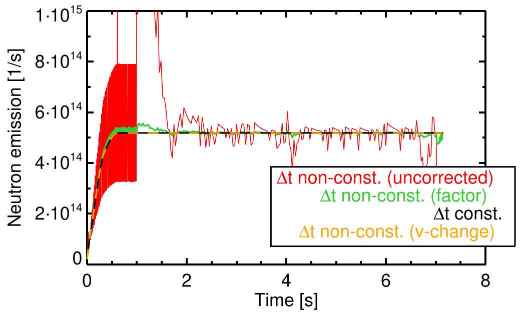
As second test, we have carried out a simulation where the inputs are sampled on a non-equidistant time-base. The number of time-steps is equal to the equidistant case, but we added a random (uniformly distributed) noise onto the timebase. Around 0.54 s we modified a few time-points to be very close together, followed by a very large time-step—in order to mimic the effect of an unforeseen lag in the calculation. Figure 11 shows the results and it can be seen that the non-equidistant simulation matches very well with the time-equidistant one. It should be noted that making this figure (especially the source Q3 only plots) also triggered



(a)



(b)



(c)

Figure 9. RABBIT calculations for volume integrals of (*a*) heating power (*b*) driven current and (*c*) neutron emission. The newly developed numerical scheme is shown in yellow and yields now with non-equidistant time-steps an identical result as with $\Delta t = const$. The inputs are constant over time for this test-case.

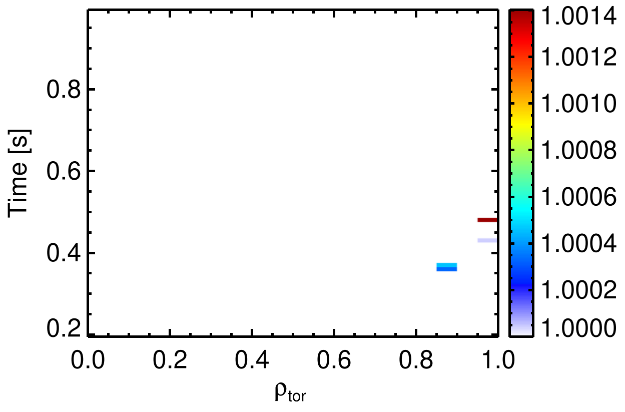


Figure 10. Ratio between two RABBIT neutron rate calculations on an equidistant time-grid: (without 'v-change' correction) / (with 'v-change' correction.) They agree perfectly for almost all radial bins and time-points (white color = perfect match), and have reasonable numeric errors in few bins at the plasma edge. The agreement is as expected, since for equidistant time-steps, the 'v-change' correction is not needed.

an optimization of the speed diffusion correction, which is explained in appendix A.

We conclude that with above robustification, the algorithm works well also under realistic conditions. If there are still missing corner cases which need to be dealt with is a subject of future studies. We hope that this might clarify automatically when RABBIT is run regularity in the time-dependent mode in real time.

6. Steady-state vs time-dependent operation mode

Besides the time-dependent simulation, RABBIT allows also for calculating directly the steady-state solution. Technically, this is done by setting internally the time-step to a sufficiently large value (i.e. larger than the slowing-down time). In the DCS implementation, one can flexibly switch between those two modes with a flag in the configuration file and this section is devoted to a comparison of both modes.

A small but important detail is how we use the NBI input power timetraces in the AP. The NBI system publishes its current power in each cycle to the DCS (typically 1.5 ms), whereas RABBIT runs much slower. It was therefore thought that it would be good to average the NBI power over all cycles since the last RABBIT call. Technically, the DCS framework allows for this, as data from older cycles can be retrieved. In doing so, it would have been guaranteed that the time-integral of applied NBI power matches exactly the time-integral of input NBI power to RABBIT.

The beams on ASDEX Upgrade can either be switched on or off, there are no intermediate power levels possible. To emulate those, pulse width modulation (PWM) with a period of 16 cycles is used, which is similar to the RABBIT calculation period. Therefore, averaging would occur over fractions of the PWM period, which would introduce a lot of noise in the input power to RABBIT (depending on e.g. if the duty cycle falls by chance into the averaging window or lies outside of it). Hence, the adopted strategy is to average over a fixed time window equal to the PWM period, i.e. the last 16 cycles (17

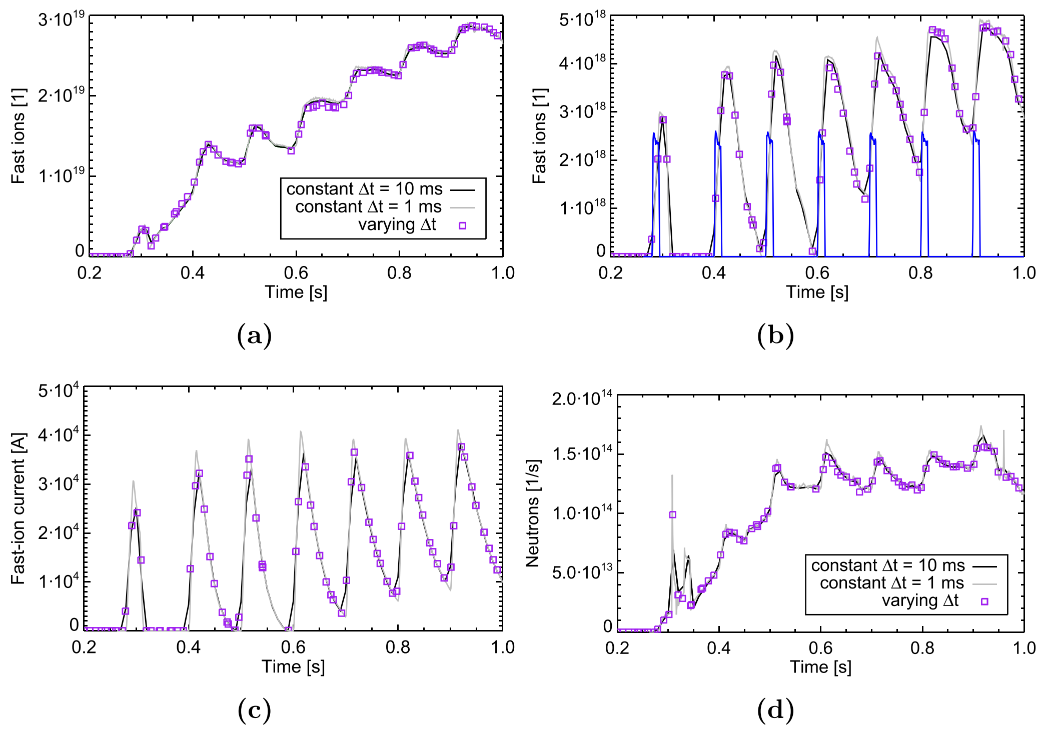


Figure 11. Time traces of *(a)* volume integrated fast-ion density, *(b)* volume integrated fast-ion density (only NBI source Q3, which injects only short blips as depicted by the blue time trace showing the Q3 power) *(c)*, total fast-ion current (Q3 only) and *(d)* volume-integrated neutron emission (all NBI sources, including beam-beam reactions).

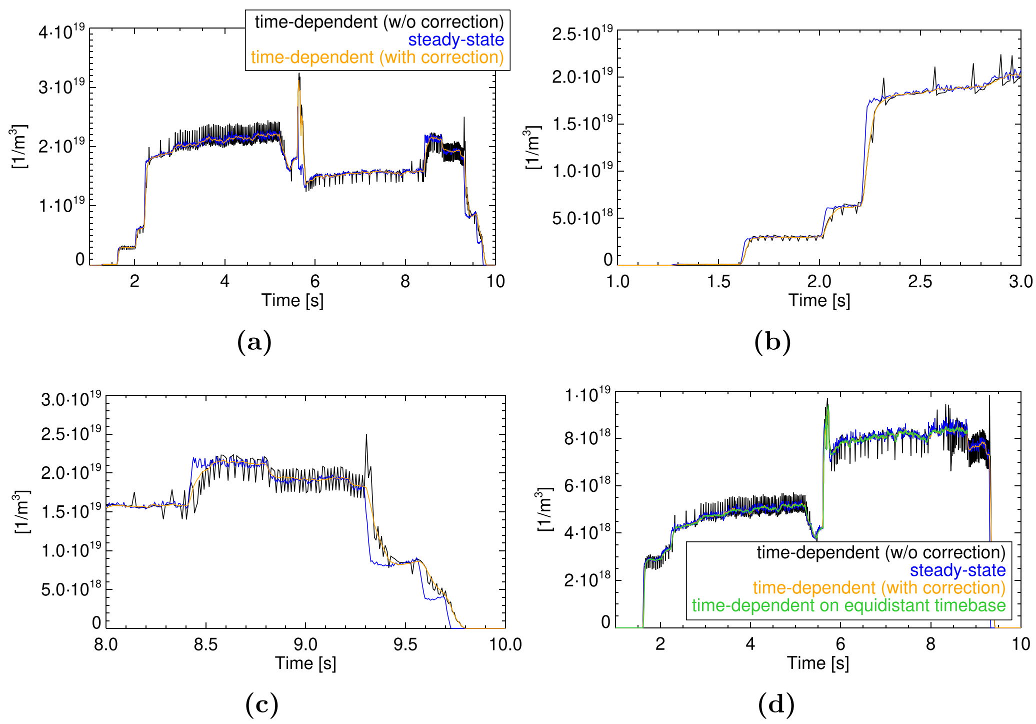


Figure 12. Time traces of the fast-ion density at $\rho_{\mathrm{tor}} = 0.075$ for re-runs of discharge 41254. *(a)* Overview, *(b)* and *(c)* Zooms. *(d)* shows the results for NBI source Q3 only, and with comparison to an offline run with equidistant time-base.

cycles prior to shot 38744 due to a bug). In this way, we get a smooth NBI power input signal into RABBIT and smooth outputs as a consequence. There is no longer a guarantee that the time-integral of applied NBI power matches exactly the time-integral of input NBI power to RABBIT, but this violation should be small and negligible.

To compare the steady-state and time-dependent calculation mode, we have used the ASDEX Upgrade discharge

41254 and used the recently developed DCS test-framework, which allows us to re-run shots based on the xml-protocols from the actual shot. This means that we can re-run RABBIT with the same inputs as it would have received in real-time.

Figure 12 shows three such re-runs: one with the steady-state mode (blue curve) and two with the un-corrected and corrected time-dependent mode (black and orange curves, respectively.) One sees clearly that the un-corrected curve has a strong noise, which is cured in the 'v-change' correction curve. In fact, it is even smoother than the steady-state solution, which can be understood by intrinsic smoothing over one slowing down time, such that noise in inputs such as T_e or n_e does not propagate 1:1 into the final solution.

One can also see that through most of the discharge, the steady-state solution does agree very well with the time-dependent one. Only in transient phases, there is a mismatch for a short period of time, most notably when beams are switched on or off and the time-dependent solution has a delayed response to that. The spike at 5.7 s is caused by a momentary strong (and likely wrong) drop of the input T_e. There is also one beam switching off here, causing a mismatch between steady-state and time-dependent solution. In figure 12(*d*) we have checked that a stand-alone calculation with the RABBIT executable, with identical inputs but on a fixed, equidistant time-base gives the identical solution as the 'v-change' corrected solution on a non-equidistant time-base.

We can conclude that the corrections for non-equidistant time-steps work and RABBIT is ready for use in real-time both in the steady-state and time-dependent operation mode.

7. Application in real-time and comparison with TRANSP-NUBEAM

In this section we finally want to show an example of RABBIT running in real time, and demonstrate that the results are useful by comparing it to an offline simulation with TRANSP-NUBEAM. In offline simulations, the input data is more realistic due to the availability of more diagnostics. Therefore, we also did an 'offline' RABBIT run to disentangle potential inaccuracies due to the short-comings of the real-time inputs and the approximations done in RABBIT.

We show discharge 38310 where RABBIT ran in the steady-state configuration. The NBI input power per source is shown in figure 13. Time-traces of fast ions, fast-ion energy and driven current are shown in figure 14.

The agreement between the time-traces is quite good, the offline RABBIT and TRANSP runs are within ≈10% (as reported also in [7]). Overall, the RABBIT-offline lines are a bit higher than TRANSP which can be explained by missing loss effects in RABBIT: while it does take prompt orbit losses into account, collisionally delayed losses and charge-exchanges losses are only taken into account by TRANSP. Those latter losses are significant in this case (see figure 15), although of course the charge-exchange losses have a high uncertainty because the neutral density is uncertain and partially determined by free parameters in

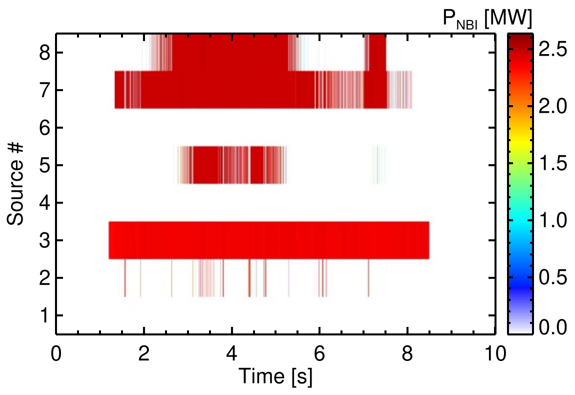


Figure 13. NBI heating power per source in ASDEX Upgrade discharge 38310.

TRANSP. The shine-through agrees very well, indicating that atomic cross-sections in both codes agree and the simplified beam geometry of RABBIT does not have a detrimental effect.

The real-time RABBIT timetraces deviate further, which can be explained due to slightly different and less accurate inputs that are available in real-time. We would judge that these deviations are still tolerable and would not hinder any application of the real-time code. The situation might also improve in the future with more and more diagnostics becoming real-time capable, and more sophisticated data evaluations.

The corresponding radial profiles of fast-ion density, pressure and current-drive density are shown in figure 16 for $t = 4$ s and they agree quite well in shape and in magnitude, even slightly better than the timetraces. This is linked to the real-time equilibrium having a ≈ 8% bigger volume (mostly around the X-point).

Figure 17 shows the time-step of the RABBIT calculations. Since RABBIT runs asynchronously with the main DCS, i.e. it publishes its result as fast as it can, this is equal to the calculation time of RABBIT. In phases without NBI power, this is 1.5 ms, which is equal to the main DCS cycle and means that there is no significant overhead of the AP connecting RABBIT and the DCS. In phases without Q7, RABBIT runs with ≈ 15 ms, and about 25 ms in phases with Q7.

The effect that RABBIT runs slower when sources Q6 or Q7 are activated has proven to be systematic and was also measurable in an offline profiling of the RABBIT-standalone executable. It can be understood as follows: since RABBIT calculates all beams in parallel, the time-step is determined by the slowest beam. Q6 and Q7 are the tangential, current-drive sources of ASDEX Upgrade, and have a much longer path through the plasma as the other six sources (see figure 18). RABBIT calculates the beam attenuation on a fixed step-width of 1 cm, which means that the attenuation for Q6 and Q7 is calculated approximatively on twice as many grid points. This seems to be the bottleneck here and cause the significant increase of calculation time. A possible work-around would be to increase the step-width for Q6 & Q7, but this cannot really be justified because plasma parameters change

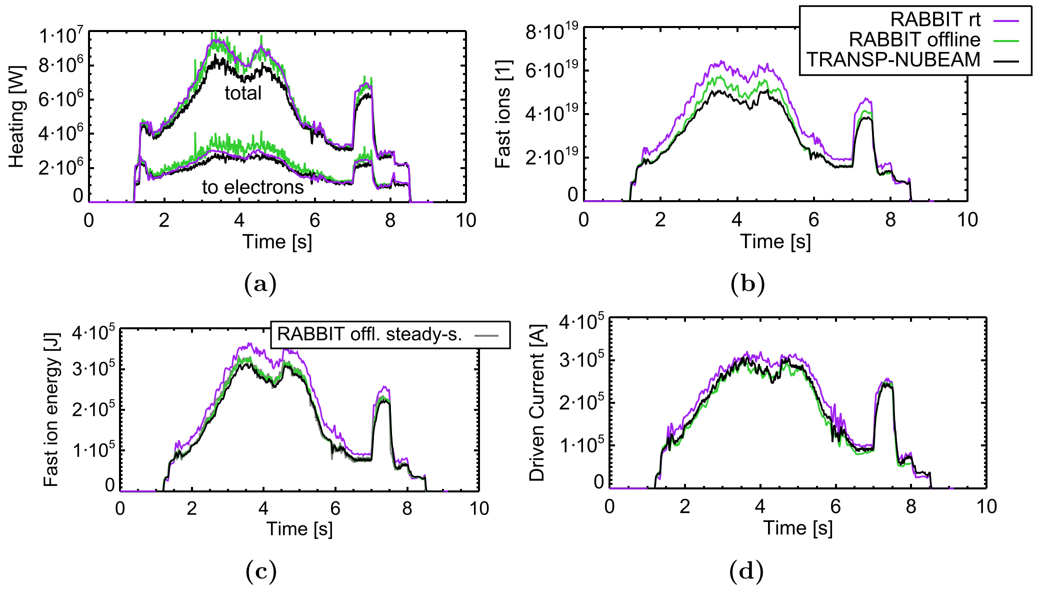


Figure 14. Time traces of RABBIT results in 38310. The actual real-time simulation is compared with an offline RABBIT and TRANSP calculation (which have identical but more realistic inputs). (*c*) shows additionally in grey the offline RABBIT run in steady-state mode. It is slightly smoothed (to remove noise from not averaging properly over the NBI duty cycles in this case) but shows that like in figure 12, steady-state and time-dependent solution do agree (apart from transient phases).

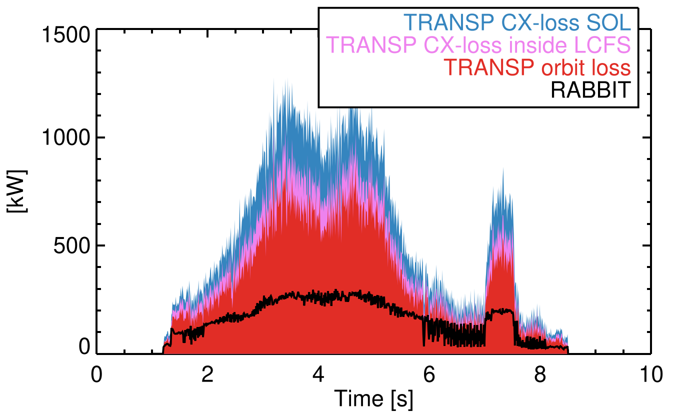


Figure 15. Comparison of the fast-ion orbit related loss terms in offline simulations of 38310. The TRANSP outputs are, respectively: charge exchange (CX) losses in the scrape-off layer (SOL) (the neutral particle density there is a free parameter), CX losses inside the last closed flux surface (the neutral density is calculated by the FRANTIC 1D model, with two sources: neutrals coming from NBI and neutrals coming from the plasma edge (set by a free parameter), lost orbits (promptly and collisionally delayed). The RABBIT curve are losses during orbit averaging, which are prompt losses plus orbits that are confined but intersect only partially with closed flux surfaces. The latter are considered 'partially lost' proportional to the time they spend in the SOL, to mimic SOL CX losses.

on a comparable length scale for these beams as for the other beams. Nevertheless there might be room for further optimization, if it is desirable in the future.

We have also estimated the calculation times for 41254, i.e. the case we studied in the previous section 6. This was done with DCS replay shots, because with this mechanism one gets timings as they would occur during an actual ASDEX Upgrade shot (while the DCS test framework does not guarantee this, but one can even slow down the DCS main cycle on purpose for testing). The results are shown in figure 19 and we can see that the steady-state solution is faster than the time-dependent solution, which can be understood because here, only one fast-ion pulse needs to be tracked for every beam and energy component. One can also see that the corrections for non-constant timesteps make the code only slower by an insignificant amount. Once again, we see that the code is slower in phases with tangential beams, here it is Q6. One can also see that the code execution time does not go down to 1.5 ms in phases where there is no beam on. This is because the real-time NBI power signal for Q2 and Q4 does contain some noise in this discharge, such that its power is above the threshold (currently hardcoded to 0.9 W) such that RABBIT starts calculating.

In summary, all these timings are equal to or below the 25 ms promised in [7], and clearly fast enough for any real-time application, as they are shorter than the slowing-down time and comparable to the period length of the NBI PWM (16 cycles = 24 ms).

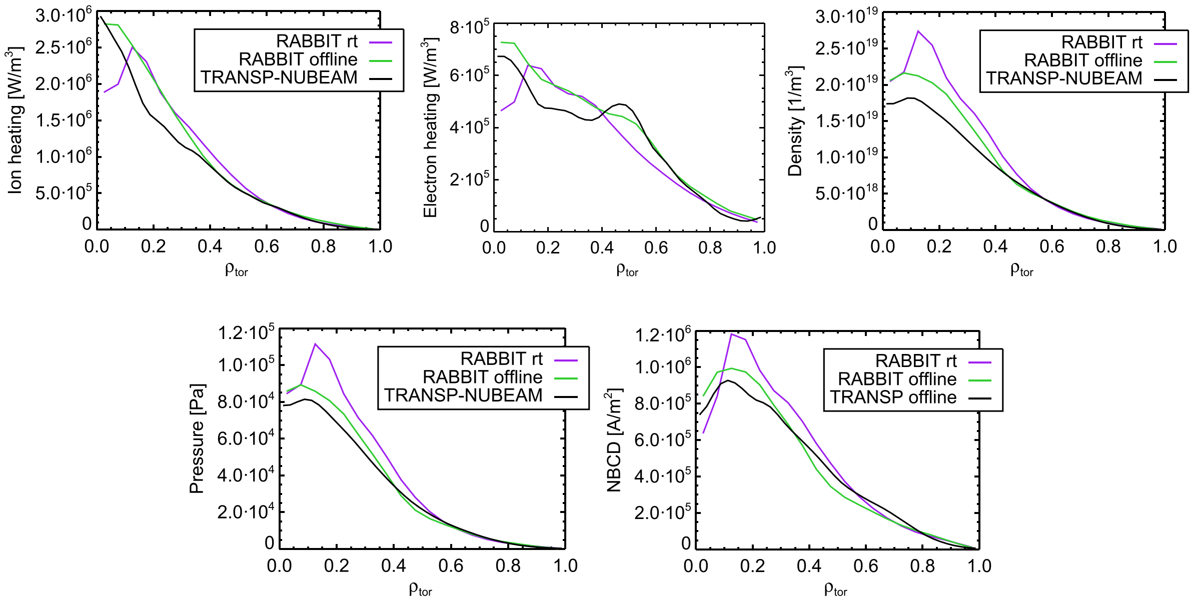


Figure 16. Profiles of RABBIT results in 38310 at 4.00 s. The actual real-time simulation is compared with an offline RABBIT and TRANSP calculation (which have identical but more realistic inputs).

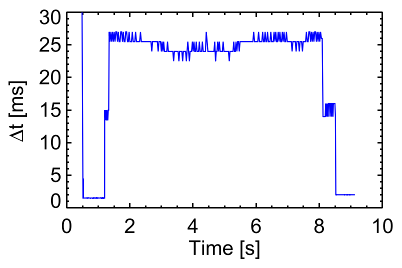


Figure 17. Time-step of real-time RABBIT calculations in ASDEX Upgrade discharge 38310.

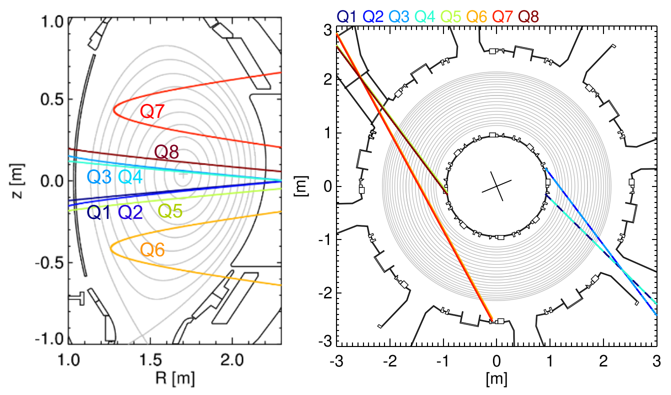
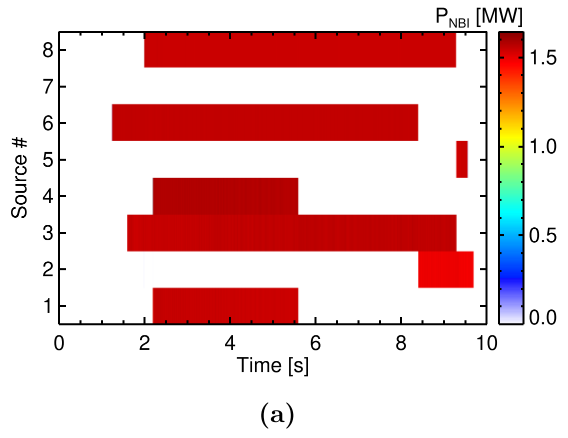


Figure 18. Overview of the eight NBI injectors at ASDEX Upgrade (the inclination of Q6 and Q7 is steerable). Left: poloidal cross-section, right: top-down view.



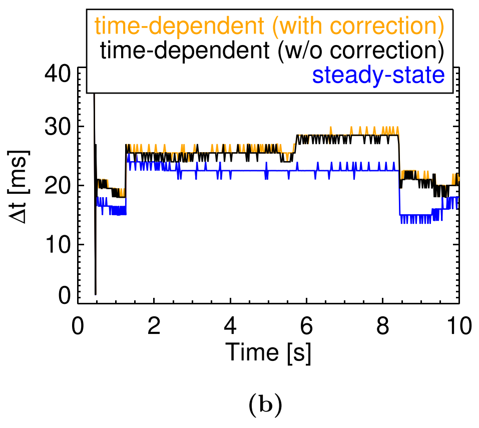


Figure 19. *(a)* NBI heating power in AUG discharge 41254. *(b)* Time-step of RABBIT calculations done in DCS replays of that discharge.

8. Summary, conclusion and outlook

The high fidelity NBI code RABBIT has been coupled to the real-time DCS of ASDEX Upgrade as a satellite. It runs asynchronously with the DCS main cycle ($\approx$1.5 ms), meaning that it can publish its results as fast as possible, typically within 15–25 ms. The first discharge with outputs from real-time RABBIT was 36668 during the 2019 ASDEX Upgrade campaign. RABBIT is written in Fortran2008 and provides a C interface, which is then used by the AP (newly written in C++) which handles the transmission of data between RABBIT and the DCS.

The Fokker–Planck numeric scheme to handle the time-dependence has been revised in order to cope with the non-equidistant time-steps as they happen in real-time. We have shown that the updated scheme handles these irregular time-steps just as well as equidistant ones. Agreement with steady-state simulations is observed in non-transient phases, as expected.

We have also analyzed one discharge (38310) in further detail, comparing it to offline simulations with TRANSP and RABBIT. Here, more diagnostics are available, such that input signals are more reliable. If we use TRANSP as benchmark, the agreement with offline-RABBIT is good (within 10%), while the real-time RABBIT is slightly worse due to less accurate inputs. The deviations are up to 25% for some quantities, which seems still acceptable and very useful for real-time applications. The accuracy might improve even more with further diagnostics becoming available in the future.

The planned applications of real-time RABBIT consist of two categories. In the first, the results of RABBIT will serve as inputs for other real-time codes, such as the transport code RAPTOR or the equilibrium solver JANET. With the help of the additional inputs from RABBIT, the results of these codes should become more accurate, which is essential to reach ambitious goals such as current profile control.

As second, RABBIT allows to control properties of the NBI system directly. Experiments have already been carried out to demonstrate real-time control of e.g. current drive, ion heating and stored fast-ion energy, and they will be reported in future publications.

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.

Appendix A. Improved treatment of speed-diffusion

As mentioned in [7, 8], the speed-diffusion in RABBIT was treated by appending the an exponential tail to the highest fast-ion pulse of each beam and energy component. Integrated over all pitches ξ the tail is written as:

$$f_+(v) = \frac{1}{2\pi}\frac{\tau_s}{v_0^3 + v_c^3}\cdot S_0 \cdot \exp\frac{-(v^2 - v_0^2)}{v_{\text{eff}}^2} \qquad (25)$$

where S_0 is the source strength of the given fast-ion pulse (i.e. how many fast ions have been injected per unit volume and time). All other parameters and the full 2D form $f_+(v,\xi)$ are given in [8]. A correct treatment of time-dependence is neglected here, instead this steady-state tail is used instantaneously.

It was noticed during the preparation of this manuscript that this scheme can lead to spurious noise in the signals, especially when simulating short blips of a beam, like in figures 11(*b*) and (*c*). This is sketched in figure 20: the first plot shows a single fast-ion pulse with appended high-energy tail due to speed diffusion. If during the next time-step, the NBI is switched off, this will (via averaging of the NBI power over the whole time-step) result in a lower averaged NBI power and hence a lower fast-ion pulse. The height of this pulse is solely dependent on the time-sampling and hence is rather arbitrary. In the old scheme, the high-energy tail was only appended to this last pulse, hence the strength of the speed-diffusion was arbitrary. In the sketched example, the fast-ion density in

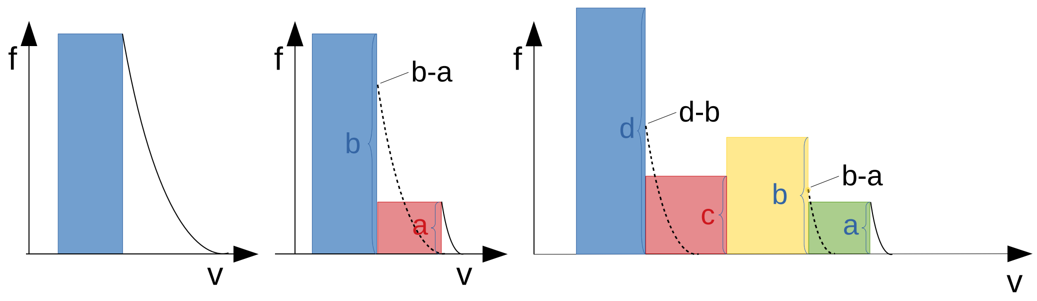


Figure 20. Sketch of improved speed-diffusion correction for three different examples. Previously, only the high-energy tails drawn with full lines were considered, now the dashed ones are added, too.

the high energy tail would drop strongly between time-step 1 and time-step 2.

To cure this, we go now iteratively through the list of fast-ion pulses. The latest pulse (which will have the highest energy) will always get a fast-ion tail. The next pulse will now also get a fast-ion tail, but only if its source term S_0 (i.e. its height) is higher than the last fast-ion pulse which got a high-energy tail. As source term, this fast-ion tail will only get the difference between its source term and the last fast-ion pulse which got a high-energy tail. This is illustrated for two examples in figure 20, where the added high-energy tails in the new version of RABBIT are drawn with dashed lines.

In doing so, we ensure that the correction for speed-diffusion will always be done in a strength corresponding to the highest fast-ion pulse in the list, and not only to the latest one, which is arbitrary. The physical model does by no means get more accurate from this correction (it is anyway wrong to use the steady-state high energy tail in such a transient situation). However, the result will get more self-consistent and independent from the time-sampling. In addition, spurious noise due to different time-sampling is eliminated.

ORCID iDs

M. Weiland https://orcid.org/0000-0002-5819-9724
L. Giannone https://orcid.org/0000-0001-5611-200X
W. Treutterer https://orcid.org/0000-0003-0268-1578

References

[1] Sieglin B., Treutterer W. and Giannone L. 2019 DCS satellite: enhanced plant system integration on ASDEX Upgrade *Fusion Eng. Des.* **146** 1737–40

[2] Giannone L. *et al* 2013 A data acquisition system for real-time magnetic equilibrium reconstruction on ASDEX Upgrade and its application to NTM stabilization experiments *Fusion Eng. Des.* **88** 3299–311

[3] Giannone L. *et al* 2015 Improvements for real-time magnetic equilibrium reconstruction on ASDEX Upgrade *Fusion Eng. Des.* **100** 519–24

[4] Felici F., Sauter O., Coda S., Duval B., Goodman T., Moret J. and Paley J. (The TCV Team) 2011 Real-time physics-model-based simulation of the current density profile in tokamak plasmas *Nucl. Fusion* **51** 083052

[5] Poli E. *et al* 2018 TORBEAM 2.0, a paraxial beam tracing code for electron-cyclotron beams in fusion plasmas for extended physics applications *Comput. Phys. Commun.* **225** 36–46

[6] Reich M., Bilato R., Mszanowski U., Poli E., Rapson C., Stober J., Volpe F. and Zille R. 2015 Real-time beam tracing for control of the deposition location of electron cyclotron waves *Fusion Eng. Des.* **100** 73–80

[7] Weiland M., Bilato R., Dux R., Geiger B., Lebschy A., Felici F., Fischer R., Rittich D. and van Zeeland M. (The ASDEX Upgrade and Eurofusion MST1 Teams) 2018 RABBIT: real-time simulation of the NBI fast-ion distribution *Nucl. Fusion* **58** 082032

[8] Weiland M., Bilato R., Collins C., Heidbrink W., Liu D. and Zeeland M. A. (The ASDEX Upgrade, DIII-D, Eurofusion MST1 Teams and JET Contributors) 2019 Simulation of neutron emission in neutral beam injection heated plasmas with the real-time code RABBIT *Nucl. Fusion* **59** 086002

[9] Tardini G., Weiland M., Angioni C., Cavedon M., Ryter F., and Schneider P. (T. A. U. Team) 2021 New assessment of the fast ion energy in ASDEX upgrade H-mode discharges *Nucl. Fusion* **61** 036030

[10] Sauter O. and Medvedev S. 2013 Tokamak coordinate conventions: COCOS *Comput. Phys. Commun.* **184** 293–302

[11] Fischer R. *et al* (ASDEX Upgrade Team) 2016 Coupling of the flux diffusion equation with the equilibrium reconstruction at ASDEX Upgrade *Fusion Sci. Technol.* **69** 526–36