Computer Physics Communications 264 (2021) 107876

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Performance of the BGSDC integrator for computing fast ion trajectories in nuclear fusion reactors☆,☆☆

Krasymyr Tretiak [a,*], James Buchanan [b], Rob Akers [b], Daniel Ruprecht [c]

[a] School of Mathematics, University of Leeds, United Kingdom
[b] CCFE, Culham Science Centre, Abingdon, United Kingdom
[c] Lehrstuhl Computational Mathematics, Institut für Mathematik, Technische Universität Hamburg, Germany

ARTICLE INFO

Article history:
Received 5 June 2020
Received in revised form 24 November 2020
Accepted 2 February 2021
Available online 2 March 2021

Keywords:
Fast ions
Boris integrator
Particle tracking
Spectral deferred corrections
DIII-D
JET
Neutral beam injection

ABSTRACT

Modelling neutral beam injection (NBI) in fusion reactors requires computing the trajectories of large ensembles of particles. Slowing down times of up to one second combined with nanosecond time steps make these simulations computationally very costly. This paper explores the performance of BGSDC, a new numerical time stepping method, for tracking ions generated by NBI in the DIII-D and JET reactors. BGSDC is a high-order generalization of the Boris method, combining it with spectral deferred corrections and the Generalized Minimal Residual method GMRES. Without collision modelling, where numerical drift can be quantified accurately, we find that BGSDC can deliver higher quality particle distributions than the standard Boris integrator at comparable computational cost or comparable distributions at lower computational cost. With collision models, quantifying accuracy is difficult but we show that BGSDC produces stable distributions at larger time steps than Boris.

1. Introduction

Computer simulations are a critical tool for the design and operation of fusion reactors [1]. Numerical computation of the trajectories of fast ions generated, for example, from neutral beam injection[1] is important to minimize wall loads and energy loss from ions escaping magnetic confinement [2]. At their core, particle trackers integrate the Lorentz equations

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t) \tag{1a}$$

$$\dot{\mathbf{v}}(t) = \alpha\left[\mathbf{E}(\mathbf{x}) + \mathbf{v} \times \mathbf{B}(\mathbf{x})\right] =: \mathbf{f}(\mathbf{x}, \mathbf{v}). \tag{1b}$$

for a large ensemble of particles and use the resulting trajectories to generate statistical quantities like wall load. Note that fast ions in fusion reactors are typically not fast enough to require consideration of relativistic effects and so the non-relativistic Lorentz equations can be used.

Fast ions interact with the plasma and deposit energy, thus heating it. To compute steady-state distributions, trajectories need to be computed until the fast ions loose their energy through collisions and thermalize. This slowing down time over which an ensemble of trajectories needs to be calculated depends on the energy of the ions when injected and the bulk plasma parameters. For DIII-D, the required simulation time is around 0.1 s. For the larger JET, it is around 1 s. Because resolving gyro effects requires time steps of the order of nanoseconds, simulations involve many millions of time step, leading to substantial computational cost and thus long solution times. Simulating a full ensemble of fast ions in JET until thermalization takes several days, despite making use of modern GPU clusters.

Some models, for example NUBEAM [3,4] or OFMC [5], use a guiding centre approximation where gyro effects are neglected or only included once a particle is near the walls. This allows taking larger time step and thus reduces computational cost. However, taking orbit effects into consideration is important to generate realistic wall loads [6]. Other particle tracking codes, for example ASCOT [7] and LOCUST [8], compute the full equations with gyro effects. For numerical methods, the only choices available are the Boris integrator [9], that is, a velocity Verlet or a

☆ This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This work was also supported by the Engineering and Physical Sciences Research Council EPSRC, UK under grant EP/P02372X/1 "A new algorithm to track fast ions in fusion reactors".
☆☆ The review of this paper was arranged by Prof. David W. Walker.
* Corresponding author.
E-mail addresses: k.tretiak@leeds.ac.uk (K. Tretiak), james.buchanan@ukaea.uk (J. Buchanan), Rob.Akers@ukaea.uk (R. Akers), ruprecht@tuhh.de (D. Ruprecht).

[1] Since charged particles would be deflected by the magnetic field before reaching the plasma, a fusion reactor is heated by injecting neutral particles. Once inside the plasma, these neutral particles undergo collisions with the bulk plasma and become ionized.

https://doi.org/10.1016/j.cpc.2021.107876

Leapfrog scheme with a geometrical trick to resolve the implicit dependence on velocity,[2] and the Cash–Karp Runge–Kutta 4(5) method [10]. LOCUST also features a mover based on Strang splitting, which is very similar to Boris but avoids some issues around loss of accuracy in cylindrical coordinates [11]. There seems to be agreement that due to its significant energy drift, RK4(5) is less accurate than Boris and so the latter is typically used. Although Boris is surprisingly efficient [12], long solution times remain an issue: the time for injected fast particles to reduce their energy to that of the thermal plasma via collisions, called the collisional slowing down time, is on the order of seconds for a reactor like ITER. Resolving cyclotron dynamics requires timesteps on the order of nanoseconds so that simulations require computing around 10^9 steps. Even on CCFE's dedicated GPU cluster, such a simulation takes days to complete. While other algorithms have been proposed for solving the Lorentz equations [13–18], they have so far not been adopted for fast ion tracking.

Tretiak and Ruprecht [19] introduce BGSDC, a new high-order algorithm for solving the Lorentz equations based on a combination of spectral deferred corrections, a Generalized Minimal Residual (GMRES) iteration and the Boris integrator. They show improvements in computational performance over Boris for individual particle trajectories in a mirror trap as well as trapped and passing particles in a Solev'ev equilibrium. This paper extends their results by demonstrating performance for realistic test cases, studying practically relevant, aggregate quantities to assess quality of solutions instead of individual trajectories. Instead of analytically given idealized magnetic fields, we use toroidally axisymmetric free-boundary equilibria that are more closely representative of the actual conditions on DIII-D and JET (i.e. with a poloidal flux map that is expressed as a bicubic spline so as to generate a divertor X-point). In the non-collisional case, we track particle ensembles corresponding to real neutral beam injection (NBI) scenarios [2,20] and assess statistical distribution of particle drift in the ensemble in contrast to exploring accuracy of individual trajectories. We show that BGSDC delivers better distributions with smaller mean and standard deviation than Boris and, if means and standard deviations of the order of micrometres are desired, can deliver them with less computational work. Then, we investigate the case where models for collisions of fast ions with the plasma are active. While the stochastic nature of these results makes a quantitative assessment with respect to work versus precision difficult, we demonstrate that BGSDC can deliver similar results as Boris with larger time steps. Delivering additional evidence to show that these gains are enough to also deliver computational gains in the collisional case will require a more developed framework to quantify accuracy for the generated statistical distributions as well as substantially more computational results and is left for future work. Our results demonstrate that there can be a computational benefit from using BGSDC or other particle trackers with order of accuracy higher than two, in particular for high fidelity simulations with tight accuracy requirements.

All results were generated with the BGSDC implementation [21] that is now part of the LOCUST code and the ITER Integrated Modelling & Analysis Suite (IMAS) [22].

2. Methodology

The GMRES-accelerated Boris-SDC algorithm (or BGSDC for short) is described in detail by Tretiak et al. [19] whereas its predecessor, without GMRES-acceleration, was introduced by Winkel et al. [23]. This original Boris-SDC combined the Boris algorithm introduced by Boris in 1970 [9] with spectral deferred corrections (SDC) introduced by Dutt et al. in 2000 [24] to generalize it to higher order. BGSDC incorporates a GMRES-based convergence accelerator for SDC, introduced by Huang et al. in 2006 [25] for first order problems, that leads to faster convergence to the collocation solution and thus improved long-term energy stability. All simulation results reported in this paper were generated using a BGSDC implementation in the GPU-accelerated LOCUST particle tracking code developed at CCFE.

2.1. Collocation methods

In essence, BGSDC is an iterative solver for a collocation method. Over one time step $[t_n, t_{n+1}]$, the Lorentz Eqs. (1) written in integral form become

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_n}^{t} \mathbf{v}(s)\, ds \tag{2a}$$

$$\mathbf{v}(t) = \mathbf{v}_0 + \int_{t_n}^{t} \mathbf{f}(\mathbf{x}(s), \mathbf{v}(s))\, ds \tag{2b}$$

with $\mathbf{x}_0$, $\mathbf{v}_0$ being approximations of position and velocity at time t_n brought forward from the previous step. Note that we consider only the case where the electric and magnetic field vary in space but not in time, but the method can easily be generalized to the non-autonomous case. Typically, the Boris method is based on a Leapfrog discretization of the differential form of the Lorentz Eqs. (1). Here, we use a variant based on the Velocity-Verlet method instead

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \left(\mathbf{v}_n + \frac{\Delta t}{2} \mathbf{f}(\mathbf{x}_n, \mathbf{v}_n) \right) \tag{3a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{\Delta t}{2} \left(\mathbf{f}(\mathbf{x}_n, \mathbf{v}_n) + \mathbf{f}(\mathbf{x}_{n+1}, \mathbf{v}_{n+1}) \right), \tag{3b}$$

see Tretiak et al. for a discussion [19]. A geometric trick was introduced by Boris in 1970 [9] to avoid the seemingly implicit dependence on $\mathbf{v}_{n+1}$. Birdsall and Langdon give a detailed description [26, Section 4-4].

Collocation methods discretize the integral form (2) using numerical quadrature with nodes $t_n \leq \tau_1 < \cdots < \tau_M \leq t_{n+1}$ instead of the differential form. Approximate values $\mathbf{x}_{\text{new}}$ and $\mathbf{v}_{\text{new}}$ at time t_{n+1} are computed via

$$\mathbf{x}_{\text{new}} = \mathbf{x}_0 + \sum_{m=1}^{M} q_m \mathbf{v}_m \tag{4a}$$

$$\mathbf{v}_{\text{new}} = \mathbf{v}_0 + \sum_{m=1}^{M} q_m \mathbf{f}(\mathbf{x}_m, \mathbf{v}_m) \tag{4b}$$

where q_m are quadrature weights while $\mathbf{x}_m$, $\mathbf{v}_m$ are approximations to position and velocity at the quadrature nodes τ_m. These are equivalent to the stages of a collocation method, an implicit Runge–Kutta method with a dense Butcher tableau [27, Theorem 7.7], and can be computed or approximated by solving the stage equations

$$\mathbf{x}_m = \mathbf{x}_0 + \sum_{j=1}^{M} q_{m,j} \mathbf{v}_j \tag{5a}$$

$$\mathbf{v}_m = \mathbf{v}_0 + \sum_{j=1}^{M} q_{m,j} \mathbf{f}(\mathbf{x}_j, \mathbf{v}_j). \tag{5b}$$

Depending on the choice of quadrature nodes, collocation methods can have a range of desirable properties. When applied to Hamiltonian systems in canonical coordinates, they are symplectic for Gauss–Legendre nodes [27, Theorem 16.5] and symmetric

[2] ASCOT seems to use the Verlet variant whereas LOCUST uses the Leapfrog version.

for Gauss–Lobatto nodes [28, Theorem 8.9] as well as A-stable [29, Theorem 12.9]. Symplectic integration in non-canonical coordinates is extremely challenging. By combining the stages $\mathbf{x}_m$, $\mathbf{v}_m$ into one vector $\mathbf{U}$, the stage Eqs. (5) can compactly be written as a nonlinear system

$$\mathbf{U} - \mathbf{Q}\mathbf{F}(\mathbf{U}) = \mathbf{U}_0 \tag{6}$$

with $\mathbf{Q}$ containing quadrature weights,

$$\mathbf{F}(\mathbf{U}) = (\mathbf{v}_1, \ldots, \mathbf{v}_M, \mathbf{f}(\mathbf{x}_1, \mathbf{v}_1), \ldots, \mathbf{f}(\mathbf{x}_M, \mathbf{v}_M)), \tag{7}$$

and $\mathbf{U}_0$ containing repeated entries of $\mathbf{x}_0$ and $\mathbf{v}_0$. See Winkel et al. for details [23].

2.2. Boris-GMRES-SDC (BGSDC)

Spectral deferred corrections use an iteration based on a low order method to solve Eq. (6). For first order problems, this is typically an implicit or implicit–explicit Euler [24,30]. For second order problems, velocity-Verlet integration or, in the special case of the Lorentz equations, the Boris integrator can be used [23]. If the collocation problem (6) is linear and thus, in a slight abuse of notation, reads

$$(\mathbf{I} - \mathbf{Q}\mathbf{F})\,\mathbf{U} = \mathbf{U}_0, \tag{8}$$

one can apply a preconditioned GMRES iteration instead of SDC to solve it [25]. The key point is that GMRES does not require assembly of the system matrix but only a function that applies $\mathbf{I} - \mathbf{Q}\mathbf{F}$ to a given vector $\mathbf{U}$. This simply means computing Eq. (5) for $m = 1, \ldots, M$. However, to improve performance, it is advisable to use a preconditioner. In the GMRES interpretation, the low order base method (Euler in the first order case, Boris in the second order case) can be understood as a preconditioner, modifying the original collocation system (6) to

$$(\mathbf{I} - \mathbf{Q}_\Delta\mathbf{F})^{-1}\,(\mathbf{I} - \mathbf{Q}\mathbf{F})\,\mathbf{U} = (\mathbf{I} - \mathbf{Q}_\Delta\mathbf{F})^{-1}\,\mathbf{U}_0, \tag{9}$$

where $\mathbf{Q}_{\Delta t}$ has a block structure with each block being a lower triangular matrix [19]. To apply GMRES to the preconditioned problem, a second function is required that can solve

$$(\mathbf{I} - \mathbf{Q}_\Delta\mathbf{F})\,\mathbf{U} = \mathbf{b} \tag{10}$$

for a given right-hand side $\mathbf{b}$ [31]. Because of the special structure of $\mathbf{Q}_\Delta$, this can be done in a sweep-like fashion, very similar to the sweeps in the original variant of SDC. When using $M = 3$ nodes, solving Eq. (10) amounts to a block-wise solve of

$$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix} - \begin{bmatrix}0 & 0 & 0\\ \Delta\tau_2\mathbf{I} & 0 & 0\\ \Delta\tau_2\mathbf{I} & \Delta\tau_3\mathbf{I} & 0\end{bmatrix}\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\end{bmatrix} - \frac{1}{2}\begin{bmatrix}0 & 0 & 0\\ \Delta\tau_2^2\mathbf{I} & 0 & 0\\ \Delta\tau_2^2\mathbf{I} & \Delta\tau_3^2\mathbf{I} & 0\end{bmatrix}\begin{bmatrix}\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1)\\ \mathbf{F}(\mathbf{x}_2,\mathbf{v}_2)\\ \mathbf{F}(\mathbf{x}_3,\mathbf{v}_3)\end{bmatrix} = \begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{b}_3\end{bmatrix}$$

and

$$\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\end{bmatrix} - \frac{1}{2}\begin{bmatrix}\Delta\tau_1\mathbf{I} & 0 & 0\\ (\Delta\tau_1+\Delta\tau_2)\mathbf{I} & \Delta\tau_2\mathbf{I} & 0\\ (\Delta\tau_1+\Delta\tau_2)\mathbf{I} & (\Delta\tau_2+\Delta\tau_3)\mathbf{I} & \Delta\tau_3\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1)\\ \mathbf{F}(\mathbf{x}_2,\mathbf{v}_2)\\ \mathbf{F}(\mathbf{x}_3,\mathbf{v}_3)\end{bmatrix} = \begin{bmatrix}\mathbf{b}_4\\ \mathbf{b}_5\\ \mathbf{b}_6\end{bmatrix}$$

via first computing

$$\mathbf{x}_1 = \mathbf{b}_1 \tag{11a}$$

$$\mathbf{v}_1 = \mathbf{b}_4 + \frac{1}{2}\Delta\tau_1\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1), \tag{11b}$$

then

$$\mathbf{x}_2 = \mathbf{b}_2 + \Delta\tau_2\mathbf{v}_1 + \frac{1}{2}\Delta\tau_2^2\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1) \tag{12a}$$

$$\mathbf{v}_2 = \mathbf{b}_5 + \frac{1}{2}(\Delta\tau_1+\Delta\tau_2)\,\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1) + \frac{1}{2}\Delta\tau_2\mathbf{F}(\mathbf{x}_2,\mathbf{v}_2), \tag{12b}$$

and finally

$$\mathbf{x}_3 = \mathbf{b}_3 + \Delta\tau_2\mathbf{v}_1 + \Delta\tau_3\mathbf{v}_2 + \frac{1}{2}\Delta\tau_2^2\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1) + \frac{1}{2}\Delta\tau_3^2\mathbf{F}(\mathbf{x}_2,\mathbf{v}_2) \tag{13a}$$

$$\mathbf{v}_3 = \mathbf{b}_6 + \frac{1}{2}(\Delta\tau_1+\Delta\tau_2)\,\mathbf{F}(\mathbf{x}_1,\mathbf{v}_1) + \frac{1}{2}(\Delta\tau_2+\Delta\tau_3)\,\mathbf{F}(\mathbf{x}_2,\mathbf{v}_2) + \frac{1}{2}\Delta\tau_3\mathbf{F}(\mathbf{x}_3,\mathbf{v}_3) \tag{13b}$$

using Boris' trick to compute the velocities [19]. Note that for a single particle, where $\mathbf{F} = \mathbf{f}$, with given initial values $\mathbf{x}_n$, $\mathbf{v}_n$ at the beginning of the time step, setting $\mathbf{b}_1 := \mathbf{x}_n + \Delta\tau_1\left(\mathbf{v}_n + \frac{\Delta\tau_1}{2}\mathbf{F}(\mathbf{x}_n,\mathbf{v}_n)\right)$ and $\mathbf{b}_4 := \mathbf{v}_n + \frac{\Delta\tau_1}{2}\mathbf{F}(\mathbf{x}_n,\mathbf{v}_n)$ means that (11) becomes identical to (3) with $\Delta t = \Delta\tau_1$. For specific choices of $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_5$ and $\mathbf{b}_6$, the steps (11), (12) and (13) correspond to a total of three Boris steps (3) with step sizes $\Delta\tau_1$, $\Delta\tau_2$ and $\Delta\tau_3$. However, to apply the GMRES procedure, the algorithm is modified to accept any input for $\mathbf{b}$. This procedure is straightforward to generalize for any number of nodes M.

For nonlinear collocation problems, it is possible to apply a Newton iteration and use GMRES-SDC to solve the linear inner problems. For the case where the nonlinearity is due to the magnetic field depending on $\mathbf{x}$, this was found not to be competitive, as the gains in convergence speed could not offset the significant added computational cost [19]. Instead, we linearize the collocation problem by freezing the magnetic field after the first sweep provides values $\mathbf{x}_m^0$ for $m = 1, \ldots, M$ by approximating

$$\mathbf{F}(\mathbf{U}) \approx \mathbf{F}_{\text{lin}}(\mathbf{X}^0)(\mathbf{U}) =: \begin{pmatrix}\mathbf{v}_1\\ \vdots\\ \mathbf{v}_M\\ \mathbf{f}(\mathbf{x}_1^0,\mathbf{v}_1)\\ \vdots\\ \mathbf{f}(\mathbf{x}_M^0,\mathbf{v}_M)\end{pmatrix},$$

that is, the electric and magnetic field are evaluated at the approximate position $\mathbf{x}_m^0$ from the first sweep instead of using the positions $\mathbf{x}_m$ in the argument $\mathbf{U}$. The linearized system is preconditioned using $(\mathbf{I} - \mathbf{Q}_\Delta\mathbf{F}_{\text{lin}})$ and solved with GMRES. The result is then corrected to account for the nonlinearity by a small number of computationally cheap discrete Picard iterations

$$\mathbf{U}^{k+1} = \mathbf{U}_0 + \mathbf{Q}\mathbf{F}(\mathbf{U}^k). \tag{14}$$

If the fields only change weakly over a single time step, the solution of the linearized collocation problem will be very close to the nonlinear solution and the Picard iteration converges quickly in very few iterations. BGSDC(k, l) refers to this combination of k GMRES-SDC iterations for the linearized collocation problem followed by l Picard iterations for the fully nonlinear problem.

Note that ideally we would want to fix an overall tolerance and have the algorithm set k and l to reach this tolerance. However, the combination of GMRES iterations for the linearized collocation problem with nonlinear Picard iteration makes this difficult. Currently we do not have a good heuristic how to set k for a given nonlinear tolerance so that the Picard iteration converges reasonably fast but the GMRES iteration does not significantly oversolve the problem. Finding a way to do that efficiently will require a more comprehensive numerical exploration and is left for future work.

2.3. LOCUST-GPU implementation

LOCUST stands for "Lorentz Orbit Code for Use in Stellarators and Tokamaks" [8,32]. It is a software platform for solving efficiently the Lorentz equations of motion in the presence of a

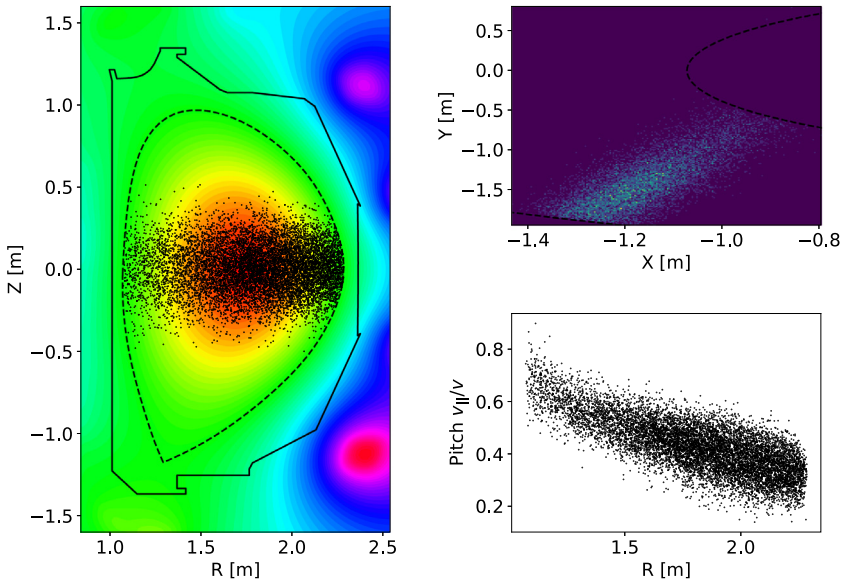

Fig. 1. Configuration of the simulated NBI shot. Poloidal profile (left) and top-down view (upper right) of particles in birth list superimposed with the poloidal flux function. The dashed line is the last closed flux surface (separatrix) while the solid line is the DIII-D reactor wall. Distribution of pitch versus major radius (lower right).

collision operator that models small angle Coulomb scattering. Kernels are instantiated upon Nvidia GPU hardware as PGI CUDA Fortran kernels which allows millions of Monte Carlo markers to be tracked in a typical simulation. LOCUST is being used extensively to design plasma facing components, e.g. for MAST-U [33], and for studying the physics of fast ion distribution and loss due to Neoclassical Tearing Modes and the application of ELM Control Coils for ITER. It is part of the EUROfusion HALO programme, where it is used to study the implications of finite gyro-radius effects, e.g. for Toroidal Alfven Eigenmode (TAE) activity in Tokamak plasma [34].

3. Numerical results

We compare BGSDC against the Leapfrog-based staggered Boris method for tracking fast ions generated by NBI in both the DIII-D and Joint European Torus (JET) tokamak. For both reactors, we study the deterministic case without models for the collision of fast ions with the plasma and the stochastic case with collision models. In the collionless case, we launch a particle ensemble corresponding to a NBI shot and use the mean and standard deviation of the numerical drift distribution for individual runs to compare the quality of both integrators. In the presence of collisions with the plasma, stochasticity makes particle drift measurements of trajectories meaningless and a very large number of markers would be required to get statistically converged results. Therefore, to focus on the impact of the numerical error and minimize the spread of ensembles, we instead study many trajectory realizations for the same particle with identical initial conditions and analyse the effect that time step size has on the resulting distribution function profiles.

Orbit drift. LOCUST measures orbit drift by recording the maximum and minimum major radius at midplane crossing of the gyro-centre for the first 10% of simulation time. At this time, all orbits will have passed through the midplane multiple times. The gyro-centre is computed for a uniform field, which leads to the presence of some residual gyro-oscillation. This is the reason that multiple midplane traversals are necessary to find reliable values for minimum and maximum major radius. The procedure is then repeated for the last 10% of simulation time and drifts are computed by taking the differences of the measured maximum and minimum major radius.

3.1. Results for the DIII-D tokamak: non-collisional case

We compare Boris and BGSDC for full orbit simulations in a 2D equilibrium for the DIII-D NBI shot sketched in Fig. 1. This setup has been used, for example, in fast ion transport studies for applied 3D magnetic perturbations in DIII-D [35]. The fast particle birth list contains 10 000 unique Deuterium ions derived from an NBI source that is injected counter to the plasma current. If more particles are simulated, LOCUST creates copies of those particles to fill in information. To match the GPU architecture that LOCUST runs on, we model $64 \times 64 \times 16 = 65\,636$ particles, since we run 64 threads per block, with 64 blocks per grid on 16 GPUs. Fig. 2 shows the radial (left) and vertical (right) component of the DIII-D magnetic field. LOCUST uses bicubic splines of the poloidal flux to calculate field components from the equilibrium data.

The duration of all runs is 100 ms and, for simplicity, we do not include plasma facing components (PFC) in our simulations and instead stop particles when they hit the wall of the equilibrium computational box. While inclusion of PFC makes the detection of where a particle hits the wall much more difficult, they do not impact performance of the numerical integration scheme.

Numerical drift. Fig. 3 shows the distribution of numerical drift for four integrators available in LOCUST at final time $t_{end} = 100$ ms for a particle ensemble in the non-collisional case. All integrators use a fixed time step of $\Delta t = 1$ ns and the same initial conditions for particles. The High Field Side (HFS)[3] drift distributions are shown in the upper two graphs and σ indicates the standard deviation. The lower graphs show the Low Field Side (LFS) drift distributions. The Strang splitting mover and classical Boris deliver comparable accuracy at both sides of the plasma volume. However, the Strang splitting mover requires

[3] Because the magnetic field in a Tokamak changes radially like $1/R$ with R being the major radius, the outboard side with large major radius is often referred to as Low Field Side (LFS) while the interior side, with small R, is called High Field Side (HFS).

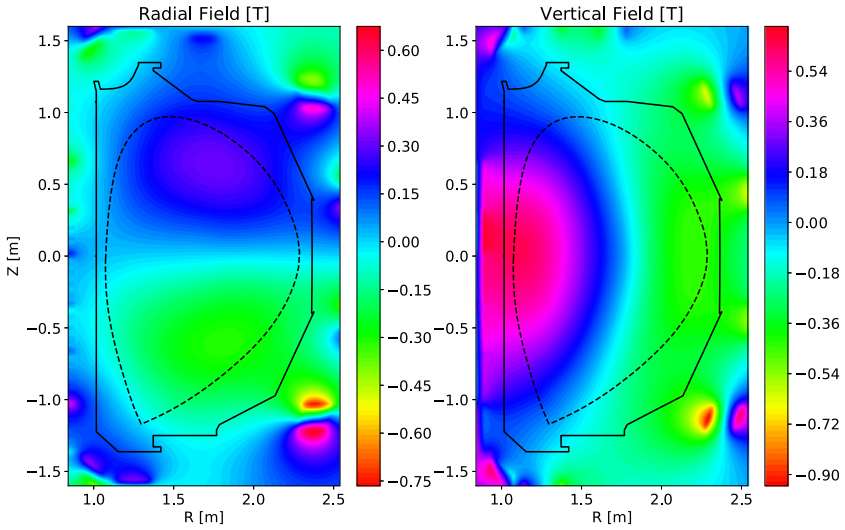

Fig. 2. Radial (left) and vertical (right) component of the DIII-D magnetic field.

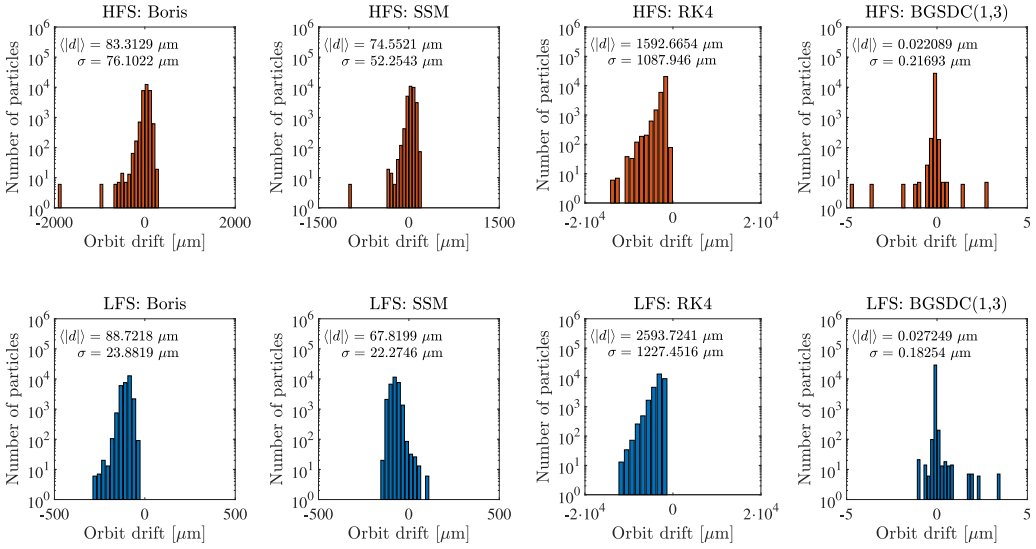

Fig. 3. Accuracy comparison of classical Boris, Strang Splitting Mover, Runge–Kutta Cash & Karp and BGSDC(1,3) methods with fixed time step $\Delta t = 1\,\text{ns}$ and $t_{end} = 100\,\text{ms}$. Please note that the x-axes are scaled differently.

more computational work than Boris and requires about 5% longer runtime. RK4 shows unsatisfactory result with very large σ, most likely due to its inherent energy drift. Since the Boris integrator performs better than SSM or RK4 we use it as a baseline to compare BGSDC against. BGSDC(1,3) is more accurate than Boris and Strang and for both LFS and HFS delivers a σ two orders of magnitude smaller. Of course, it also requires substantially more computational work per time step. Below we will demonstrate that this additional work per time step can be offset by using larger time step sizes and thus computing fewer steps, leading to computational gains when values of σ of around 1 μm or below are required.

Charged particles in a magnetic field can experience mirror trapping when entering regions of higher magnetic field strength as a consequence of conservation of the magnetic moment. These particles do not complete full orbits but instead experience a bounce point where their parallel velocity is zero and become trapped in the low field side. Because these sharp turns and resulting large gradients can affect performance of the numerical integrator [19], we show results separately for passing and trapped particles. Fig. 4 shows the resulting numerical orbital drift of each particle against the major radius R at the end of simulation at $t_{end} = 100\,\text{ms}$. The left figure is for the Boris method with time step 1 ns while the right shows results from BGSDC(2,6) with a time step of 5 ns. The upper graphs show trapped particles, the lower graphs passing particles where particles at the LFS are indicated by black markers and particles at HFS are marked in red. Note that the y-axes in the two lower graphs are scaled differently. For trapped particles, Boris and BGSDC deliver comparable results with particle drifts of up to 5 cm. For passing particles, BGSDC drifts are about an order of magnitude smaller than Boris, despite a five times larger time step.

Fig. 5 shows the frequency distribution of orbital drift in μm for all particles (both trapped and passing) shown in Fig. 4. The upper figures show values at the HFS, the lower figures at the LFS. The σ in each figure's title indicates the standard deviation for the

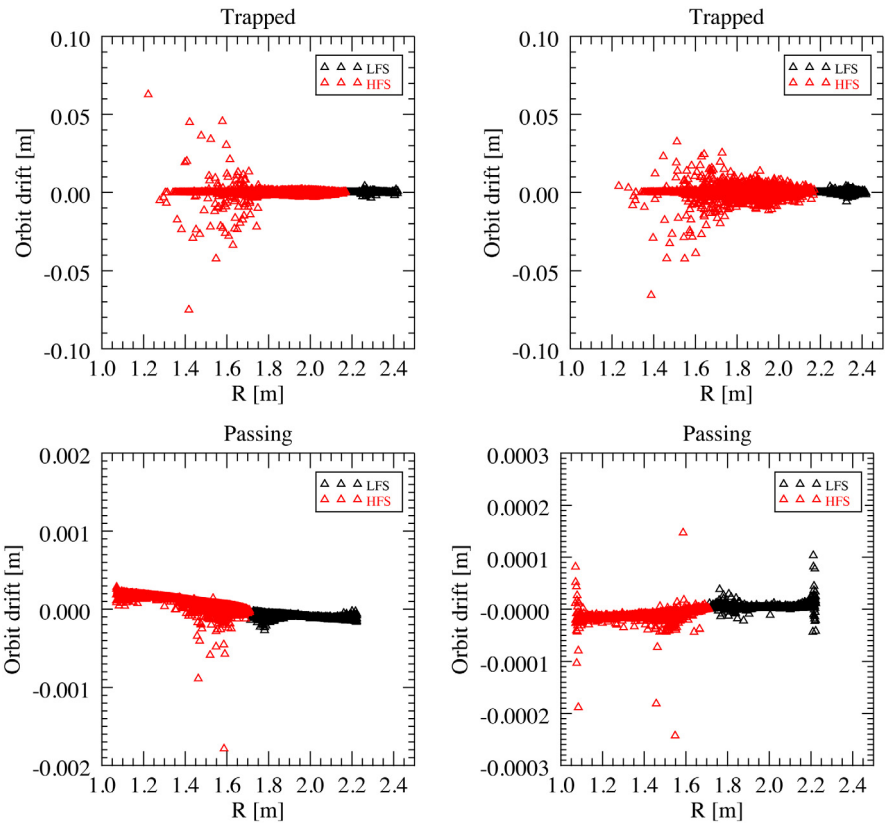

Fig. 4. Particle drifts for DIII-D for Boris method with $\Delta t = 1$ ns (left) and BGSDC(2,6) with $\Delta t = 5$ ns (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

particle cloud. The smaller σ is, the narrower and thus better is the distribution. At HFS, the standard deviation for BGSDC(2,6) is about nine times smaller than for Boris while at LFS it is a factor of around six more accurate.

Magnetic moment. While the Boris integrator conserves magnetic moment by design, for BGSDC this is only guaranteed for the underlying collocation. For small iteration numbers, non-conservation of magnetic moment cannot be ruled out by theory. However, in numerical experiments we confirmed that both Boris and BGSDC(2,6) conserve magnetic moment up to machine precision.

Accuracy versus time step size. Fig. 6 shows how the mean (upper) and standard deviation σ (lower) for the HFS (left) and LFS (right) changes with time step size. The higher order of accuracy of both BGSDC(1,3) and BGSDC(2,6) translates into a faster drop of σ with Δt and thus a steeper slope of the curve. This demonstrates that there is an accuracy gain from using integrators of higher orders, not only for individual trajectories but also for the full ensemble. It also demonstrates that BGSDC can deliver a particle distribution with comparable accuracy with a much larger time step than Boris. For a value of $\sigma = 1\,\mu$m for example, Boris requires a time step of $\Delta t = 0.1$ ns whereas for BGSDC a twenty times larger step size of $\Delta t = 2$ ns suffices.

Work-precision. BGSDC(2,6) with $M = 3$ Gauss–Lobatto nodes means we perform $k = 2$ iterations of GMRES-SDC on the linearized problem and $l = 6$ Picard iterations [19]. In contrast to Boris, which only needs one, BGSDC(2,6) therefore requires 19 right hand side (*RHS*) evaluations per time step. However, as shown in Figs. 4 and 5, it can provide comparable accuracy with a much larger Δt and will thus require fewer time steps. If the number of steps is sufficiently smaller to compensate for the increased work per step, BGSDC will be computationally more efficient.

To pinpoint where BGSDC starts to deliver computational gains, Fig. 7 (left) shows the number of total RHS evaluations required by Boris to reach some standard deviation σ divided by the number of evaluations required by BGSDC(1,3). A value of $S_u > 1$ means that BGSDC(1,3) requires fewer evaluations and thus less computational work whereas $S_u < 1$ means Boris requires fewer evaluations. We assume perfect second order convergence of Boris method to interpolate between data points, which is line with the behaviour we see in Fig. 6. For the HFS, the break-even point for BGSDC is for accuracies slightly below $\sigma = 1\,\mu$m and gains increase sharply from there. A distribution with HFS $\sigma = 0.5\,\mu$m will require only about $1/1.5 \approx 66\%$ as many evaluations as when using Boris. Gains are less pronounced when looking at LFS values, where the break-even point seems to be slightly above $\sigma = 0.1\,\mu$m.

Fig. 7 (right) shows the total runtime for LOCUST required to reach a given σ. To match a HFS of $\sigma = 10^0$, BGSDC(1,3) requires 5980 s whereas Boris needs 13 169 s. This corresponds to a speedup of 2.2, which is somewhat better than what we predict from counting RHS evaluations.

3.2. Results for the DIII-D tokamak: collisional case

LOCUST uses the same Fokker–Planck collision operator to model collisions as ASCOT. A detailed description is provided by Hirvijoki et al. [36]. In the collisional case, we launch one particle 131072 times from the same position. The duration of each run is $t_{end} = 100$ ms, the same as in the collisionless case, and we use the same 2D plasma equilibrium. We compare distribution functions generated by Boris and BGSDC. Distribution functions have four arguments: velocity, coordinates R and Z and pitch

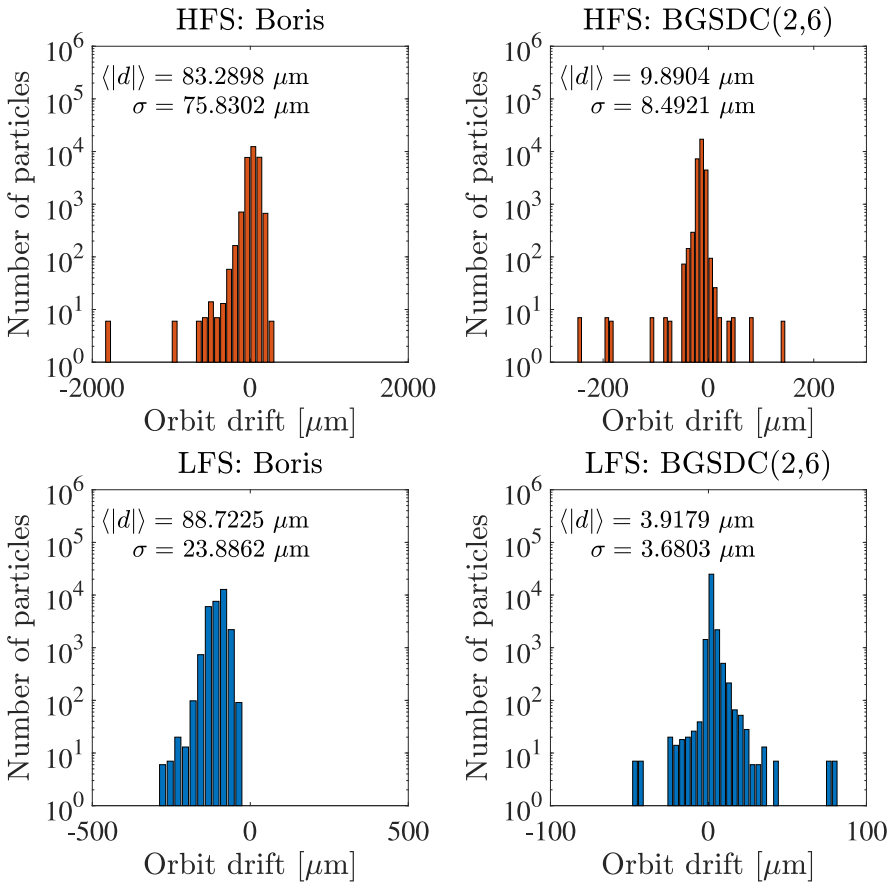

Fig. 5. Drift distribution: Boris method $\Delta t = 1$ ns (left) and BGSDC(2,6) $\Delta t = 5$ ns (right).

angle L. To visualize them in 2D plots, we fix velocity, R and Z and plot against the pitch angle L. Fig. 8 shows 1D cross-sections of the distribution function $F(v, L, R, Z)$ against pitch angle $L = v_\perp/v_\parallel$ in a phase box for a trapped particle on the top with fixed velocity $v = 1.8 \times 10^6$ m/s, major radius $R = 2$ m, axis $Z = 0.5$ m and for a passing particle at the bottom with $v = 1.8 \times 10^6$ m/s, $R = 1.9$ m and $Z = 0.3$ m.

Boris and BGSDC both converge to the same distribution. However, BGSDC produces a stable distribution for larger time steps than Boris. For trapped orbits, the distributions provided by BGSDC with 1 ns and 10 ns time steps are very similar whereas Boris shows visible differences at a time step of 10 ns. For passing orbits, both methods require slightly smaller time steps to produce stable distributions. BGSDC has converged for a step size of 5 ns whereas Boris requires a smaller step size of 2 ns.

3.3. Results for the JET tokamak: non-collisional case

For JET, full orbit simulations were carried out in a 2D equilibrium representing typical JET conditions. We use an artificial source of particles uniformly distributed in pitch angle and inside the plasma volume at a fixed injection energy. All runs last 1 s and use 65536 unique particles. As in the DIII-D runs, we do not include PFCs.

Numerical drift. Fig. 9 shows the numerical orbital drift for all particles at the end of simulation at $t_{end} = 1$ s. Results from the Boris method with time step 0.5 ns are shown on the left and from BGSDC(1,4) with 2 ns on the right. Again, the upper graphs show drift for trapped particles while the lower graphs show drift for passing particles and values for LFS are marked in black while values for HFS are marked red. For trapped particles BGSDC(1,4) shows a maximum deviation of 5×10^{-4} m whereas Boris with a 4 times smaller step shows a 2×10^{-3} m deviation. Please note that the y-axes scale changes. For passing particles both methods deliver comparable accuracy with BGSDC showing somewhat lower peak values.

Fig. 10 shows the mean (upper) and standard deviation σ (lower) of the drift distribution for both integrators depending on the time step. As for DIII-D, the higher order of accuracy of BGSDC leads to a steeper decrease in σ with time step size. For a fixed Δt, BGSDC produces much narrower distributions than Boris for both HFS and LFS.

Work-precision. Fig. 11 (left) shows again the ratio of RHS evaluations for Boris compared to BGSDC required to produce a distribution with a given σ. Results are very similar to those for DIII-D. BGSDC becomes competitive in the range between $\sigma = 0.1\,\mu$m and $\sigma = 1\,\mu$m for the HFS and LFS with better gains for HFS. Fig. 11 (right) shows runtimes needed to reach a given σ. For JET, the break-even point comes earlier than for DIII-D. BGSDC is faster than Boris for distributions with σ around $3\,\mu$m or less for both HFS and LFS.

3.4. Results for the JET tokamak: collisional case

For the collisional case we launch one particle 131072 times from the same position and track it until simulated time 1 s. Fig. 12 shows 2D profiles of the distribution function in a minor cross-section of the JET tokamak. BGSDC with $\Delta t = 2$ and $\Delta t = 5$ ns as well as Boris with $\Delta t = 2$ ns deliver comparable profiles. Profiles computed with Boris with time steps 5 ns and larger show noticeable differences.

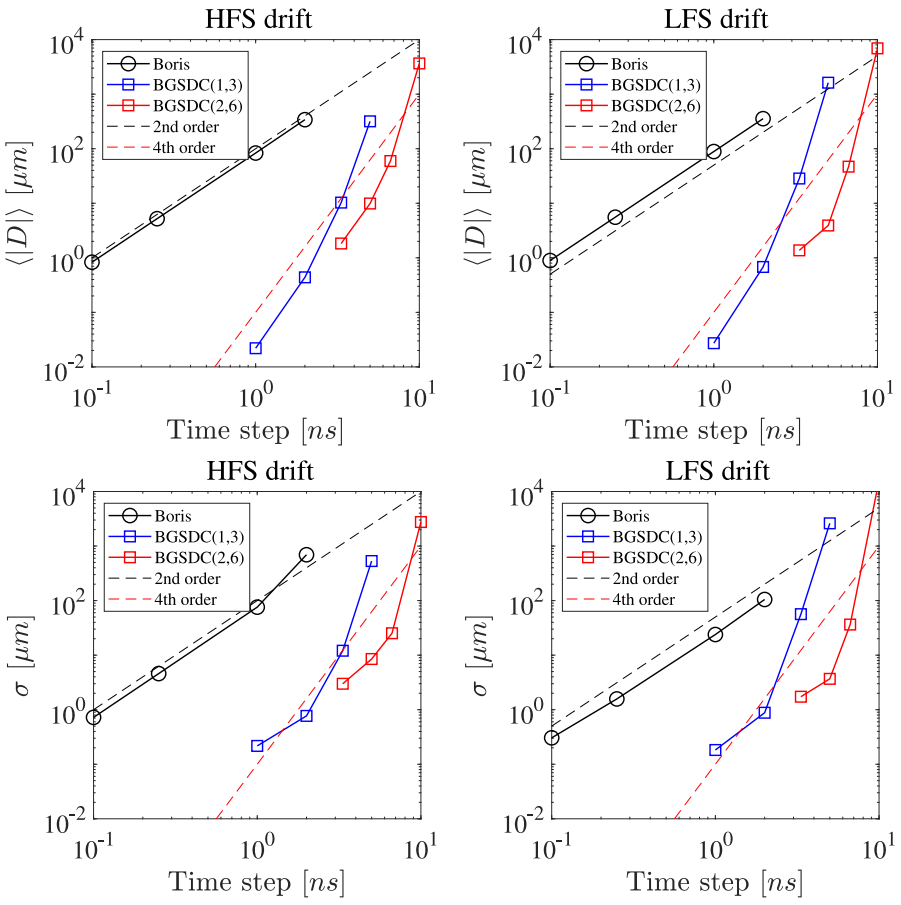

Fig. 6. Mean (upper) and standard deviation σ (lower) of drift distributions for classical Boris and BGSDC methods for different time steps in the non-collisional regime in DIII-D.

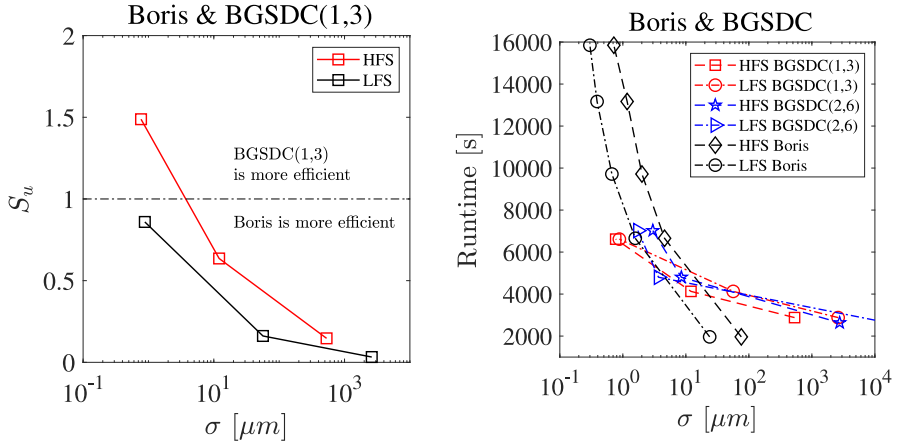

Fig. 7. Left: Ratio S_u of field evaluations required by Boris to reach some standard deviation σ divided by the number of evaluations required by BGSDC(1,3) for the DIII-D reactor. Right: Runtime to reach some standard deviation σ for the Boris method, BGSDC(1,3) and BGSDC(2,6).

Figs. 13 and 14 show 1D profiles of the distribution function in a phase box with fixed velocity and position. In the slice shown in Fig. 13, BGSDC with $\Delta t = 2$ and $\Delta t = 5$ ns and Boris method with $\Delta t = 1$ ns produce similar profiles of $F(L)$. Small deviations can be seen on both ends for the blue line (Boris with $\Delta t = 2$ ns) in the left picture and in the centre of the right picture. Boris with time step ≥ 5 ns shows significant differences in the $F(L)$ profiles and the radial profiles $F(R)$ shown in Fig. 14. Similar results have been observed at different values of v, R, Z and L but are not documented here.

The shown results suggest that, for the collisional case, BGSDC(2,6) can use a time step about 5 to 10 times larger than the Boris integrator and produce results that look very similar. This is not enough to offset the fact that it needs 19 instead of one right hand side evaluations. If the distributions are to be reproduced with better accuracy (probably around 1% or better), we would expect BGSDC to become competitive, as was observed in the collisionless case. However, as we do not currently have a useful quantitative measure for accuracy for the collisional case, this analysis is left for future work.

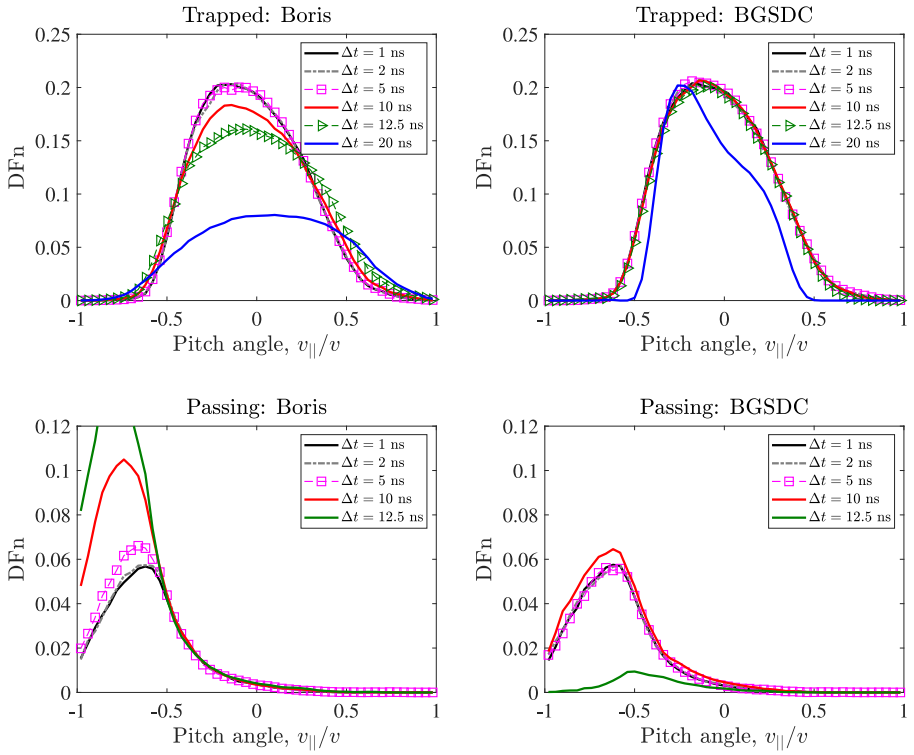

Fig. 8. 1D profiles of the distribution function for DIII-D for Boris method on the left and BGSDC(2,6) on the right.

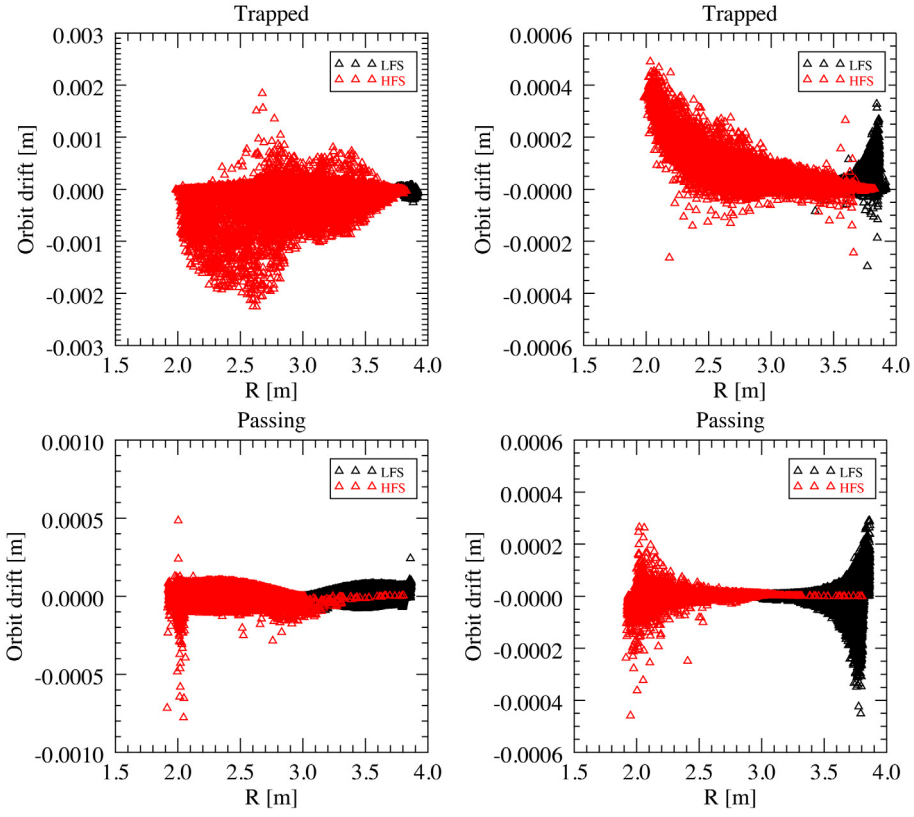

Fig. 9. Particle drift for the JET tokamak after 1 s simulation time. Boris method $\Delta t = 0.5$ ns (left) and BGSDC(1,4) $\Delta t = 2$ ns (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

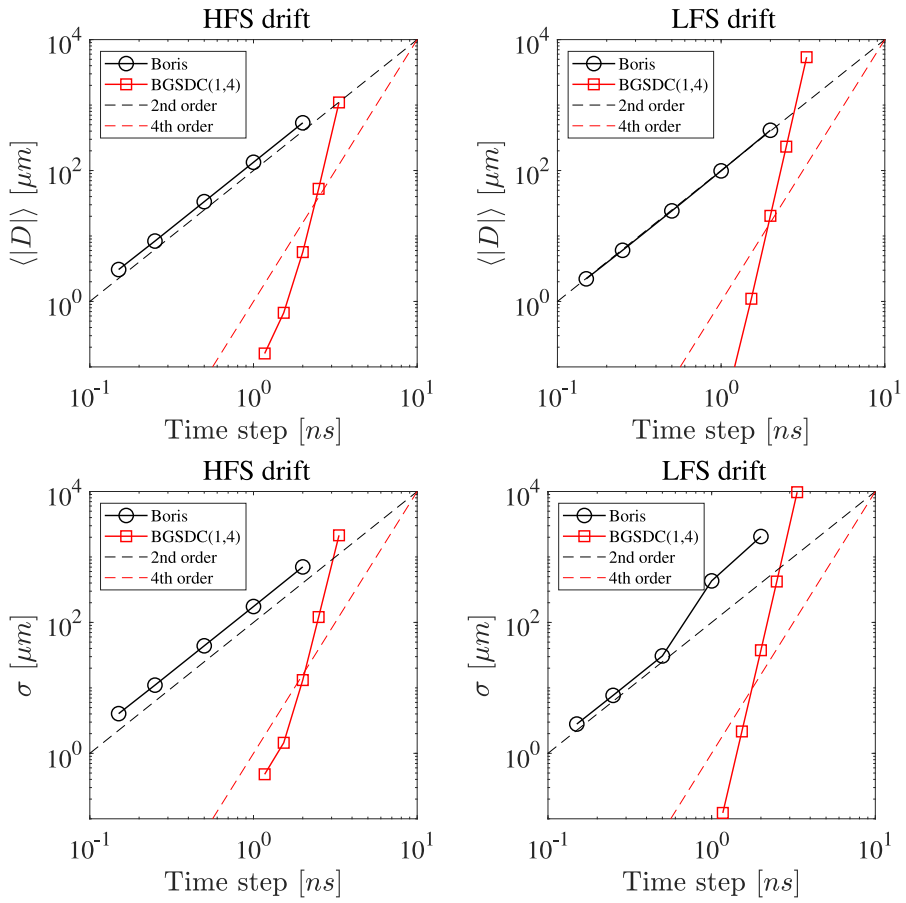

Fig. 10. Mean (upper) and standard deviation σ (lower) of the drift distribution for classical Boris and BGSDC methods for different time steps for the non-collisional regime in JET.

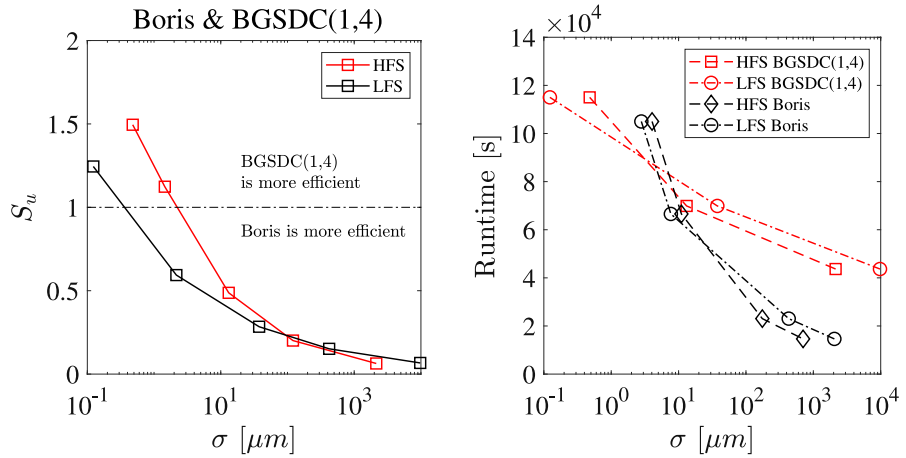

Fig. 11. Left: Ratio S_u of field evaluations required by Boris to reach some standard deviation σ divided by the number of evaluations required by BGSDC(1,4) for the JET reactor. Right: Runtime required to reach some standard deviation σ for Boris (black) and BGSDC(1,4) (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4. Conclusions

We compare the performance of the BGSDC time stepping algorithm against the standard Boris integrator when computing trajectories of fast ions generated by neutral beam injection into a fusion reactor. Numerical examples are shown for the DIII-D and JET reactors and include non-collisional and collisional models with plasma. For the non-collisional case, the model is deterministic and we can use the standard deviation of the numerical drift distribution across all particles as a metric for solution quality. The results in this paper show that for both DIII-D and JET, BGSDC produces a substantially narrower and thus better distribution with smaller mean and standard deviation than Boris at the same time step. BGSDC can provide computational gains compared to classical Boris when trajectory simulations need to be of high precision with means and standard deviations of the order of 1 μm or lower. A recent study of toroidal Alfvén eigenmodes (TAE) with LOCUST required time steps of less than 0.5 ns to

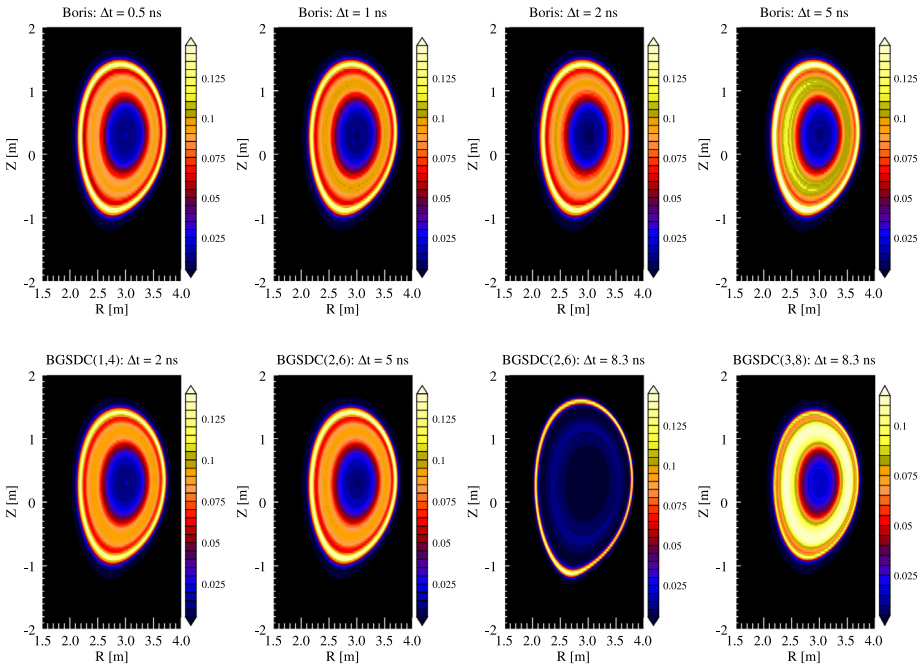

Fig. 12. 2D profiles of the distribution function for fixed velocity and pitch angle.

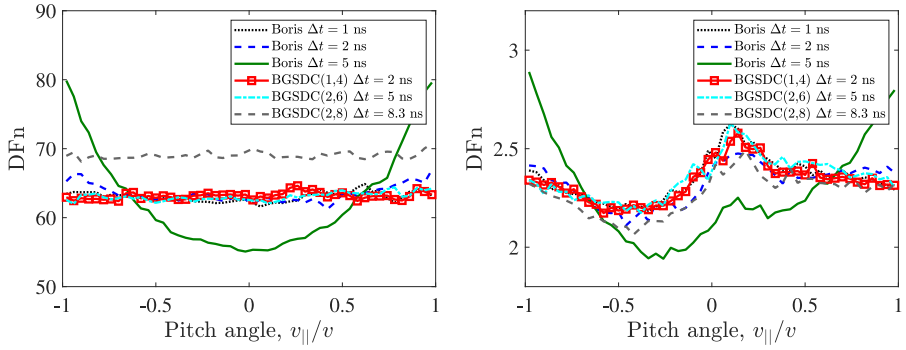

Fig. 13. 1D profiles of the distribution function against pitch angle at fixed $v = 0.1 \cdot 10^6, R = 2.2, Z = -0.1$ on the left and $v = 0.5 \cdot 10^6, R = 2.2, Z = -0.5$ on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

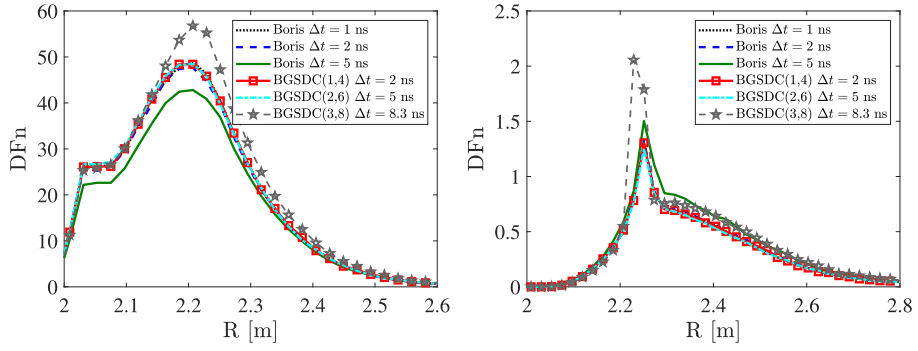

Fig. 14. 1D profiles of the distribution function along major radius with $v = 0.2 \cdot 10^6, L = 0.2, Z = 0.3$ on the left and $v = 0.8 \cdot 10^6, L = -0.9, Z = 0.1$ on the right.

deliver converged results [34]. Although no drift distribution was computed in these simulations, in our setup this would correspond to a standard deviation of around $\sigma \approx 10\,\mu$m. For studies involving higher frequencies and sharper spatial features, even smaller time steps will be required. Therefore, while most simulations do not yet require such highly accurate distribution, this will likely change in the near future.

Detailed quantitative assessment of particle trajectories in the presence of collisions is left for future work, as there are no mathematical tools established in the fusion community to define a precise notion of accuracy in these cases. However, we show that BGSDC and Boris converge to similar distributions and that BGSDC provides stable distributions at larger time steps than Boris.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Artaud, et al., Nucl. Fusion 50 (4) (2010) 043001.
[2] R. Hemsworth, H. Decamps, J. Graceffa, B. Schunke, M. Tanaka, M. Dremel, A. Tanga, H.D. Esch, F. Geli, J. Milnes, T. Inoue, D. Marcuzzi, P. Sonato, P. Zaccaria, Nucl. Fusion 49 (4) (2009) 045006.
[3] R. Goldston, D. McCune, H. Towner, S. Davis, R. Hawryluk, G. Schmidt, J. Comput. Phys. 43 (1) (1981) 61–78.
[4] A. Pankin, D. McCune, R. Andre, G. Bateman, A. Kritz, Comput. Phys. Comm. 159 (3) (2004) 157–184.
[5] K. Shinohara, Y. Suzuki, J. Kim, J.Y. Kim, Y.M. Jeon, A. Bierwage, T. Rhee, Nucl. Fusion 56 (11) (2016) 112018.
[6] A. Snicker, S. Sipilä, T. Kurki-Suonio, Nucl. Fusion 52 (9) (2012) 094011.
[7] E. Hirvijoki, O. Asunta, T. Koskela, T. Kurki-Suonio, J. Miettunen, S. Sipilä, A. Snicker, S. Äkäslompolo, Comput. Phys. Comm. 185 (4) (2014) 1310–1321.
[8] R.J. Akers, E. Verwichte, T.J. Martin, S.D. Pinches, R. Lake, Proceedings of the 39th EPS Conference & 16th Int. Congress on Plasma Physics, 2012.
[9] J.P. Boris, Proceedings of the Fourth Conference on Numerical Simulation of Plasmas, Naval Research Laboratory, Washington, DC, 1970, pp. 3–67.
[10] J.R. Cash, A.H. Karp, ACM Trans. Math. Software 16 (3) (1990) 201–222.
[11] G. Delzanno, E. Camporeale, J. Comput. Phys. 253 (2013) 259–277.
[12] H. Qin, S. Zhang, J. Xiao, J. Liu, Y. Sun, W.M. Tang, Phys. Plasmas 20 (8) (2013).
[13] Y. He, Y. Sun, J. Liu, H. Qin, J. Comput. Phys. 281 (2015) 135–147.
[14] M. Quandt, C. Munz, R. Schneider, The 30th International Electric Propulsion Conference, 2007, pp. 1–9, URL: http://bibliothek.fzk.de/zb/veroeff/69220.pdf.
[15] M. Tao, J. Comput. Phys. 327 (2016) 245–251.
[16] Y. He, Y. Sun, J. Liu, H. Qin, J. Comput. Phys. 305 (2016) 172–184.
[17] E. Hairer, C. Lubich, J. Comput. Math. 3 (2017) 205–218.
[18] T. Umeda, Comput. Phys. Comm. 228 (2018) 1–4.
[19] K. Tretiak, D. Ruprecht, J. Comput. Phys. X 4 (2019) 100036.
[20] O. Asunta, J. Govenius, R. Budny, M. Gorelenkova, G. Tardini, T. Kurki-Suonio, A. Salmi, S. Sipilä, Comput. Phys. Comm. 188 (2015) 33–46.
[21] R. Akers, K. Tretiak, 2018, URL: https://git.iter.org/projects/TRAJ/repos/locust/browse?at=BGSDC_integrator.
[22] F. Imbeaux, S. Pinches, J. Lister, Y. Buravand, T. Casper, B. Duval, B. Guillerminet, M. Hosokawa, W. Houlberg, P. Huynh, S. Kim, G. Manduchi, M. Owsiak, B. Palak, M. Plociennik, G. Rouault, O. Sauter, P. Strand, Nucl. Fusion 55 (12) (2015) 123006.
[23] M. Winkel, R. Speck, D. Ruprecht, J. Comput. Phys. 295 (2015) 456–474.
[24] A. Dutt, L. Greengard, V. Rokhlin, BIT Numer. Math. 40 (2) (2000) 241–266.
[25] J. Huang, J. Jia, M. Minion, J. Comput. Phys. 214 (2) (2006) 633–656.
[26] C.K. Birdsall, A.B. Langdon, Plasma Physics Via Computer Simulation, McGraw-Hill, New York, 1985.
[27] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, second ed., Springer-Verlag Berlin Heidelberg, 1993, http://dx.doi.org/10.1007/978-3-540-78862-1.
[28] E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, Springer Verlag Berlin Heidelberg New York, 2002, http://dx.doi.org/10.1007/3-540-30666-8.
[29] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff Problems, Springer-Verlag Berlin Heidelberg, 1996, http://dx.doi.org/10.1007/978-3-642-05221-7.
[30] M.L. Minion, Commun. Math. Sci. 1 (3) (2003) 471–500.
[31] C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations, Society for Industrial and Applied Mathematics, 1995, http://dx.doi.org/10.1137/1.9781611970944.
[32] R. Akers, et al., 2016, Available at https://nucleus.iaea.org/sites/fusionportal/Shared Documents/FEC 2016/fec2016-preprints/preprint0489.pdf.
[33] J. Milnes, N.B. Aye, F. Dhalla, G. Fishpool, J. Hill, I. Katramados, R. Martin, G. Naylor, T. O'Gorman, R. Scannell, the MAST Upgrade team, 2015, URL: https://arxiv.org/abs/1503.06677.
[34] M. Fitzgerald, J. Buchanan, R. Akers, B. Breizman, S. Sharapov, Comput. Phys. Comm. 252 (2020) 106773.
[35] M. Van Zeeland, N. Ferraro, B. Grierson, W. Heidbrink, G. Kramer, C. Lasnier, D. Pace, S. Allen, X. Chen, T. Evans, M. García-Muñoz, J. Hanson, M. Lanctot, L. Lao, W. Meyer, R. Moyer, R. Nazikian, D. Orlov, C. Paz-Soldan, A. Wingen, Nucl. Fusion 55 (7) (2015) 073028.
[36] E. Hirvijoki, T. Kurki-Suonio, S. Äkäslompolo, J. Varje, T. Koskela, J. Miettunen, J. Plasma Phys. 81 (3) (2015) 435810301.