Distribution Function Component

Fast Campaign

L. A. Berry, *et al.*

0 Time-Scale Context

The time scale for the evolution of the distribution function for the fast campaign is assumed to be significantly slower than the RF period and the ion-ion and electron-ion collision times; faster than the equilibrium (poloidal flux, toroidal flux, and MHD) timescales; and comparable to transport evolution times. The first separation is possible because dominant fast ion interactions take place on the ion-electron collision time for ion energies much larger than those of the thermal species. The second follows from the slow diffusion of the individual fluxes and of the evolution of the topology.

I Functionality

The minimal functionality for the distribution function component is to:

> Advance in time the distribution function for a low density, energetic-ion plasma component (or components) with constant background plasma and equilibrium. The energetic component is coupled to the background by background distribution functions that have a constant density, temperature, and fluid velocity while the distribution function of the energetic species is advanced in time. In turn, the background interacts with the fast component via modifications to the equilibrium and by source terms derived from the fast ion distribution function. For the fast campaign, the physics in the distribution component must include:
>
> 1. Coulomb collisions with all of the background components.
> 2. Source/sink terms for the fast ion component.
> 3. RF quasilinear diffusion.
> 4. The effects of finite drift orbits (finite banana width) and the resulting radial transport. This capability is required to explain the change in behavior for co- versus counter-parallel wave number minority heating experiments on JET.

The component may (should?, will? These are needed, but could be computed outside of the distribution function component) calculate quantities derived from this distribution function including:

> 1. Source (sink) terms for the fluid equations for thermonuclear reactions with the energetic species.
> 2. Moments of the distribution function which estimate forces and energy flows.
> 3. Moments of the Coulomb collision operator of the energetic species with each background component. The latter may be utilized by, for example, N-Class for analyzing the parallel force balance.

4. Particle loss rates through the outer boundary and viscous stress tensors for the energetic component are also possible processed outputs.

I.A Equations

The response of the energetic species distribution function, $f_0(\vec{x},\vec{v},t)$, on timescales slower than the RF period and the cyclotron period is obtained from an RF-period and gyro-period averaged of Vlasov-Boltzman equation and is referred to as the quasilinear Fokker-Planck equation. It is given by:

$$\frac{df_0}{dt} = Q(\vec{E}_{RF}, f_0) + C(f_0, f_i) + S(\vec{x},\vec{v}) . \tag{1.1}$$

Langevin equivalents to this equation have been formulated to enable Monte Carlo solutions. Note that I have borrowed from the notation used in the RF component description. Here, the total time derivative includes particle drift motion. In this equation, divergence of the RF quasilinear flux is indicated by $Q(\vec{E}_{RF}, f_0)$ and describes wave-induced velocity-space diffusion of f_0; $C(f_0, f_i)$ is Coulomb integral of the energetic component with a background species *i*, and $S(\vec{x},\vec{v})$ represents the source and sink terms for f_0. For an axisymmetric system, f_0 is (save for the sign of the parallel velocity) 4D in configuration space. A particular set of independent variables might be energy, magnetic moment, canononical toroidal angular momentum , and position along a drift orbit. In the absence of sources, the collision operator relaxes f_0 to a local Maxwellian distribution. However, quasilinear diffusion, particle or energy sources, or gradients in macroscopic plasma quantities drive f_0 away from Maxwellian.

The general form of the divergence of the quasilinear flux is

$$\nabla_v \cdot \vec{\vec{Q}} \cdot \nabla_v f_0 = \nabla_v \cdot \left\langle \left(\vec{E}^*_{RF} + \vec{v} \times \vec{B}^*_{RF} \right) f_1 \right\rangle_t \tag{1.2}$$

where the time average is over a time long compared to the gyro- and wave periods but short compared to the time scale for evolution of the distribution function.

When the time scale for completing a drift orbit is short compared to the evolution of the distribution function, an additional average may be performed on (1.1) to annihilate the drift-orbit variable and reduce dimensionality to a 3D function of time. This bounce average is over a particle's drift orbit and may be simplified, in the small banana-width limit, to an appropriate field-line average. The bounce-average approximation breaks down at orbit stagnation points, in particular at the boundary between trapped and passing orbits. This issue is usually addressed by solving the problem in two different domains, i.e., for trapped particles and passing particles, and then imposing suitable boundary conditions.

I.B Possible Codes for initial implementation of the component

At the present time, it is my (LAB) assessment that additional physics and algorithm development is required in order to implement the fast campaign. The codes described

below are (to a lesser or greater degree) possible starting points for this implementation. Appendix B contains a table with summary comparisons and comments.

I.B.i NUBEAM

"The NUBEAM module is a Monte Carlo package for time dependent modeling of fast ion species in an axisymmetric tokamak. Multiple fast ion species can be present, due either to beam injection of energetic neutral particles, or as a result of the product of nuclear fusion reactions. The model self consistently handles classical guiding center drift orbiting, collisional and atomic physics effects during the slowing down of the fast specie population (represented by an ensemble of Monte Carlo model particles), with options for sawtooth-induced or anomalous radial diffusion, in a time evolving plasma with nested magnetic flux surfaces represented by a numerical MHD equilibrium. Time stepping is explicit, with the Monte Carlo deposition of new particles and slowing down of the Monte Carlo ion ensemble against a fixed plasma for a prescribed time interval, after which the Monte Carlo model state is recorded, and control is returned to the plasma model, allowing the plasma model to then evolve for the same time interval, before resuming Monte Carlo deposition and slowing down in the next time interval, and so on until the simulation is complete." [Quoted from the NUBEAM help file.]

Comments:

1...RF quasilinear diffusion is not included at the present time but efforts to include it are underway/planned.

2. NUBEAM has a rich set of procedures for evaluating source terms for coupling to fluid equations.

3. A relatively small number of particles, ~10^3-10^4, is needed for adequate statistics when slowing down of fast ions from neutral beam is the dominant physics and when average quantities are sufficient for coupling to other components.

4. Acceleration techniques are used for improved computational efficiency.

(a) Programming languages, libraries, and other software used:

(b) Parallel programming system used (e.g. MPI, OpenMP, …):

(c) Efficiency of the calculation:

(d) Data storage and transfer requirements:

I.B.ii ORBIT-RF

ORBIT-RF is also a Monte Carlo code that shares many of the model characteristics of DELTA-5D (see discussion below). It, however, has been extensively used for studying the interaction of ICH with energetic particles [CHOI]. The quasilinear operator is constructed internally based on the amplitudes of a relatively small number of modes (usually, 31 or 63 modes). These inputs have, to date, been based on fields from either ray-tracing or from the largest amplitude mode from TORIC4. Effort to include all coherent modes is underway.

Comments

1. One of main features of ORBIT-RF is that finite orbit width effect of non-Maxwellian ions is included in Hamiltonian guiding center drift equations. At the present time, RF quasilinear diffusion is not calculated self-consistently with wave fields (i.e., amplitudes of wave fields are fixed during simulations) but efforts to calculate it self-consistently are planned in a future.

2. Acceleration techniques are used for improved computational efficiency.

(a) Programming languages, libraries, and other software used:

Programming language is F90. Libraries used are EZCDF and NETCDF.

(b) Parallel programming system used (e.g. MPI, OpenMP, …):

Parallel programming system used in ORBIT-RF is MPI

(c) Efficiency of the calculation:

Usually, we are running ORBIT-RF with 10000 test particles using 16 processors on our local LINUX cluster at GA. One run (corresponding to a few slowing down times of energetic particles) usually takes a few hours.

(d) Data storage and transfer requirements:

Several data files are stored at the end of run for visualization or restart such as restarting particle data, particle phase shots, particle trajectories, power history, many diagnostics. IDL routines are used for visualizing ORBIT-RF (post-processed) outputs.

Ref

[CHOI]

II Code specific data

II.B ORBIT-RF

II.B.i Specific inputs: grids, code specific controls, and parameters

Necessary input files

- A file controlling first start and restart
- A file describing input parameters
  - Numerical magnetic equilibrium data file
- Kinetic profile for ne, Te and Ti profile
  - Profile for wave fields and wave numbers

II.B.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

ORBIT-RF solves Hamiltonian orbit drift equations in White-Chance-Boozer coordinate ($\psi,\vartheta,\rho,\zeta$). Equivalent meshes in (ψ,ϑ) directions are used. Mappings can be applied to describe the trajectories of test ions from flux coordinate to real space.

Solution is based on Monte Carlo techniques in 5D. The problem is 3D in space and 2D in velocity (not gyrophase). For axisymmetric systems, the toroidal dimension is required for following particles, but will be binned to produce a 4D distribution function.

II.B.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

Output of mapping code is used to create magnetic numerical equilibrium data file for ORBIT-RF using the cubic spline.

II.B.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

- Numerical magnetic equilibrium data file is coming from output of mapping code
- Kinetic profile for ne, Te and Ti profile is coming from experimental measurements
- Profile for wave fields and wave numbers are coming from outputs of TORIC4.

II.B.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

I.B.iii DELTA5D

DELTA5D is a general purpose Monte Carlo particle code. It follows the guiding center orbits of a collection of particles [REF]. Both Coulomb collisions (including pitch angle and energy scattering) and an RF quasilinear heating operator are included based on a Langevin approximation. The quasilinear operator is taken from reference [ ] and is most appropriate for RF inputs with a few (a few tens?) of resonant modes. It is similar to that used in ORBIT-RF.

The following options are available with respect to the particle populations: thermal diffusive transport coefficients, bootstrap current, neutral beam slowing down and beam loss rates (heat deposition to the thermal ions and electrons to be added), RF tail generation, MHD closure relations, global transport rates through the outer flux surface

(a) Programming languages, libraries, and other software used:

(b) Parallel programming system used (e.g. MPI, OpenMP, …):

(c) Efficiency of the calculation:

(d) Data storage and transfer requirements:

I.B.iv CQL3D

CQL3D is a multi-species, fully relativistic, bounce-averaged, time-dependent Fokker-Planck equation solver [REF] that includes both Coulomb and quasilinear RF diffusion. It is 2D in momentum space with a radial coordinate for noncircular plasmas. The code is run in combination with lower hybrid, fast wave, electron cyclotron and electron Bernstein wave ray tracing or full-wave (AORSA) RF data, the FREYA neutral beam deposition package, and a given toroidal electric field, thereby providing a general model for the distortion of the electron and ion distribution functions resulting from auxiliary heating and current drive injected from the plasma periphery. The distributions are taken to be toroidally symmetric and independent of azimuthal angle about the ambient magnetic field. Radial drifts are neglected. With the bounce-average, account is taken of variations as a function of (non-circular) radial coordinate, poloidal angle, and two momentum-space directions. A kinetic bootstrap current calculation is included. The code may be run with separate 2D momentum-space solves on each flux surface, on in 3D mode including radial transport according to prescribed diffusion and pinch terms. Solution is by splitting for the 3D equations, alternating between direct solve of the implicit 2D-in-V equations and in the radial coordinate. An implicit sparse matrix solve of the full set of 3D equations is now being developed to improve stability and convergence as well as an MPI parallelization.

(a) Programming languages, libraries, and other software used:

(b) Parallel programming system used (e.g. MPI, OpenMP, …):

(c) Efficiency of the calculation:

(d) Data storage and transfer requirements:

I.B.v DKES

The Drift Kinetic Equation Solver (DKES)) code solves a time independent form of the drift kinetic equation (not bounce averaged) by velocity space expansion in orthogonal polynomials [REF]. This code is widely applied for transport in 3D magnetic configurations, but it presently does not include quasilinear RF diffusion or energy scattering. Although this code requires magnetic flux surfaces, they need not be nested and islands can be treated. High resolution spatial calculations can be performed by inverting a dense block-tridiagonal system of equations.

(a) Programming languages, libraries, and other software used:

(b) Parallel programming system used (e.g. MPI, OpenMP,):

(c) Efficiency of the calculation:

(d) Data storage and transfer requirements:

I.C Interface

I.C.i Inputs required for all distribution function solvers

$n_j(\vec{x})$, $Z_j(\vec{x})$ $T_j(\vec{x})$ and $\vec{V}_j(\vec{x})$ of all species j treated as Maxwellian.

Equilibrium data, including electric and magnetic fields sufficient to define particle orbits.

Source/sink terms for the fast ion component.

Quasilinear diffusion operator of required dimensionality or, if calculated internally (from rays), the RF field data required for this calculation.

Wall geometry description.

I.C.ii Outputs required from all distribution function components

The distribution function, $f_j(\vec{x},\vec{v})$. Whether the quantities derived from the distribution function discussed in I are calculated in the distribution function component or in separate routines/components should be discussed.

I.C.iii Interfaces provided

I.C.iv Interfaces used

Plasma state – Require all of the inputs listed in I.C.i .

Possible direct connection to RF solver to exchange $f_j(\mathbf{x},\mathbf{v})$ and $\ddot{\mathbf{Q}}(\mathbf{E},\mathbf{x},\mathbf{v})$ or $\mathbf{E}(\mathbf{x})$ for internal iteration to self-consistency.

I.C.v Expectations about input data. e.g., that equilibrium has been reached, or that numerically conservation has been maintained.

The density, but not necessarily the pressure of the “fast component is assumed small compared to background species.

I.C.vi Shared infrastructure components such as global time-keeper, interpolators from one representation to another, flux surface average routines, storage of the component's state for restart.

Shared components for calculation of derived quantities may be required.

I.C.vii Physics analysis/development needs, possible mathematical or algorithmic problems, opportunities for improvement, additional opportunities for parallelism that are not presently realized.

General: The quasilinear operator is well defined (and contains the delta function in $\omega - k v_{\|} - n\Omega_c$) for bounce averaged formulations of (1.1) under sufficiently collisional conditions to decorrelate successive resonance crossings and thus avoid bounce resonances. An additional complexity is added by full-wave RF codes (TORIC and AORSA) whose solutions are based on basis functions with the potential for 10^2 to 10^5 resonances which may be correlated with each other. To test the validity of the presently-used formulation, the RF SciDAC project is comparing bounce-averaged quasilinear operators obtained through mode summations with that obtained by a direct integration of the Lorentz equation on the RF time scale using both RF and equilibrium E&M fields [REF Harvey et al. Park City]. A possible outcome is that the QL operators obtained by direct integration are more appropriate.

Monte Carlo codes: RF codes require accurate first derivatives of the distribution function—the real part of the RF conductivity is directly proportional to this derivative. Techniques/mappings need to be developed to allow coupling of particle-based distribution functions to RF codes. The Monte Carlo codes require local [in $(\vec{r}, \vec{v})$ space] quasilinear operators. This requirement needs to be reconciled with the bounce average nature of the QL operator. The use of localized delta functions for a few modes is typical, but the validity of extending this formulation to large numbers of modes needs to be investigated. Data communication issues will have to be investigated given potentially large size of a 4D QL operator data file.

CQL3D: Finite-banana width effects (especially radial transport) are treated on an ad hoc basis. First principles justification of these techniques needs to be developed. A parallel version is being developed and is needed.

DKES: DKES will be likely be used for the slow campaign where RF and collisional effects (for electrons) have comparable time scales. For this analysis, the 3D structure of DKES can likely be retained, but modified to "substitute" energy scattering for one of the spatial dimensions.

II Code specific data

II.A NUBEAM

II.A.i Specific inputs: grids, code specific controls, and parameters

Control parameters

Code state data that is required to re restart

II.A.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

Solution is based on Monte Carlo techniques in 5D. The problem is 3D in space and 2D in velocity (not gyrophase). For axisymmetric systems, the toroidal dimension is required for following particles, but will be averaged (binned) to produce a 4D distribution function.

II.A.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

II.A.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

Most of the generic inputs will come from the Plasma State component.

NUBEAM uses a Python script and associated input parameters to setup the specific problem.

II.A.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

II.B ORBIT-RF

II.B.i Specific inputs: grids, code specific controls, and parameters

Control parameters

Code state data that is required to re restart:

II.B.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

Solution is based on Monte Carlo techniques in 5D. The problem is 3D in space and 2D in velocity (not gyrophase). For axisymmetric systems, the toroidal dimension is required for following particles, but will be binned to produce a 4D distribution function.

II.B.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

II.B.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

Most of the generic inputs will come from the Plasma State component.

II.B.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

II.C DELTA-5D

II.C.i Specific inputs: grids, code specific controls, and parameters

Control parameters

Code state data that is required to re restart:

II.C.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

Solution is based on Monte Carlo techniques in 5D. The problem is 3D in space and 2D in velocity (not gyrophase). For axisymmetric systems, the toroidal dimension is required for following particles, but will be binned to produce a 4D distribution function.

II.C.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

II.C.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

Most of the generic inputs will come from the Plasma State component.

II.C.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

II.D CQL3D

II.D.i Specific inputs: grids, code specific controls, and parameters

Control parameters

Code state data that is required to re restart:

II.D.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

Solution is based on difference techniques in 3D. There are two velocity dimensions (magnitude of the velocity and pitch angle at the outside midplane location), and one spacial dimension (flux surface).

II.D.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

II.D.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

Most of the generic inputs will come from the Plasma State component.

II.D.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

II.E DKES

II.E.i Specific inputs: grids, code specific controls, and parameters

Control parameters

Code state data that is required to re restart:

II.E.ii What type of spatial discretization is used for each field? (finite difference, finite element, spectral, finite volume, ...)? What are the dimensionalities (1D, 2D, 3D)? What are the coordinates (Cartesian, flux …)? How is the poloidal angle defined? Are

mappings applied to restructure the domain (e.g., to be able to use a regular mesh or a Collela-Burger AMR mesh)?

Solution is based on orthogonal polynomials 3D [2D in space (two angle variables on each flux surface), 1D in velocity (pitch angle)] for each flux surface. Because of the assumption of thin bananas, the flux-surface variable is decoupled, resulting in a series of 3D rather than a 4D problem.

II.E.iii Other code outputs (that might be useful for diagnostics or visualization for example) that doesn't specifically require inter-component interaction

II.E.iv Where the required inputs (both generic and code specific) are going to come from: Plasma State or other components, external file, … (Any inputs that are strictly local to the component code can and should be input through the code's current mechanisms e.g. file I/O usually).

Most of the generic inputs will come from the Plasma State component.

II.E.v Summary of computational needs this component will have when used for the fast MHD campaign. Including computations, memory, amount of time to carry out its designated task in the framework, and kind of parallelism (threaded or distributed memory or both).

III Appendix – sample input files

III.A NUBEAM

III.B ORBIT-RF

III.C DELTA-5D

III.D CQL3D

III.E DKES

Appendix A Sources component diagram.

Appendix B Code Comparisons

Code	Solution type	Dimensionality	Used with RF	Drift Orbit Physics	Issues All: QL operator formulation
NUBEAM	Monte Carlo	5D (4D)	No	Full	Not coupled with RF, statistics for smooth derivatives,
ORBIT-RF	Monte Carlo	5D (4D)	Yes, one pass, RF to distribution function	Full	Statistics for smooth derivatives,
DELTA-5D	Monte Carlo	5D (4D)	No	Full	Not iteratively coupled with RF, statistics for smooth derivatives, QL operator formulation
CQL3D	finite difference, bounce averaged	3D (2D velocity, 1D space)	Yes, iterated with AORSA2D	Ad hoc	Finite banana width physics, QL operator formulation
DKES	orthogonal polynomials	3D (2D space, 1D velocity, per flux surface)	No	Yes, but ssumes small departures from flux surfaces	Modification to 1D space, 2D velocity, time dependence, finite banana width, QL operator formulation