A user's guide to `CODE`

Adam Stahl

Last updated: April 4, 2017

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden, 2017

Introduction

Welcome to A user's guide to `CODE`! This document will tell you all you need to know to start using and understanding the numerical tool `CODE` and its output. The text contains descriptions of features and parameters, as well as concrete code examples and a few exercises and benchmarks to get you started.

`CODE` is a tool for solving the electron kinetic equation in 2D momentum space, i.e. in a homogeneous plasma slab. In particular, the implementation is aimed at the study of runaway electrons in fusion plasmas and includes most of the effects that are relevant to their momentum-space dynamics (including a relativistic treatment). `CODE` is written in MATLAB and uses a finite-difference grid to represent the particle momentum and a Legendre-mode decomposition in pitch angle. The equations and computational scheme are described in Ref. [1], which introduces the original version of the tool. The implementation is very efficient for small to medium-sized problems and `CODE` runs on a standard laptop in anything from milliseconds to minutes, depending on problem size. For really large problems, more computational power may be needed.

Since the original implementation, a number of features have been added. Many are described in Ref. [2], however several other papers also introduce new functionality, as will be discussed in Chapter 1.

This document will start by briefly describing the main features of `CODE`, followed by a discussion of the important numerical parameters in Chapter 2. Chapter 3 discusses the result of a `CODE` calculation, and how to process and visualize it. A description of how to specify the inputs and settings, as well as concrete examples and various use cases are given in Chapter 4, and Chapter 5 provides a few exercises and benchmarks. Finally, the two papers [1, 2] describing the main features are included for convenience.

Contents

1 Versions and features

`CODE` is available in two distinct versions:

- `CODE_steadyState.m` — finds a steady-state solution for the distribution function
- `CODE_timeDependent.m` — solves the kinetic equation as an initial-value problem, i.e. evolves the distribution in time

The steady-state version is very computationally efficient, but has a limited scope as some processes such as avalanche generation or time-dependent parameters cannot be included. For this reason, the steady-state version has seen less development compared to the time-dependent version, and only includes some of the features described below. This text will in general focus on the features of the time-dependent version, however the steady-state version will also be discussed where appropriate. We will use the symbols SS and TD to denote features that are available in the steady-state and time-dependent versions, respectively.

There also exists an adaptation of `CODE` called `CODION`, which is described in Ref. [3]. It is based on the same numerical method, but treats the ion instead of electron kinetic equation, assuming thermal electrons. `CODION` will not be further discussed in this text.

1.1 Basic features

`CODE` calculates the evolution of the electron distribution function f in 2D momentum space under the influence of a parallel electric field and collisions with electrons and ions. No spatial dependence is included, i.e. the plasma is assumed to be spatially uniform (and magnetized), and the electric field normalization is such that a positive value leads to acceleration of the electrons in the positive parallel direction. The collision operator is linearized around a Maxwellian, which is assumed to be nonrelativistic, although the tail of f is allowed to reach arbitrary energies since a relativistic formulation is used (see also Sec. 1.2).

The system is solved efficiently by constructing a single matrix and factorizing it. This is then either immediately used to find the steady-state solution (SS), or the factors used repeatedly to advance the system in time (TD). Unless time-dependent parameters are used (see Sec. 1.4), there is no need to rebuild and refactorize the matrix, which makes time advance very efficient.

1.2 Fokker-Planck collision operators

The ions are assumed to be a stationary Maxwellian on the time scales considered and the small mass of the electrons in relation to the ions is neglected. The electron-ion collision operator is therefore simple, depending only on the effective ion charge Z_{eff}. A less approximate operator, accounting for the partial penetration of the electron cloud of the ion by energetic electrons, has also been implemented and will be discussed in Sec. 1.7.

The electron-electron collision operator is the main source of complexity in the problem. Various variants are included in the time-dependent version:

- a relativistically valid test-particle operator for collisions with a nonrelativistic electron bulk [1], providing only particle conservation (SS, TD)
- the full linearized nonrelativistic operator, i.e. nonrelativistic test-particle and field-particle terms providing energy, momentum and particle conservation [2]. This operator requires the use of a heat sink, which will be discussed in Sec. 1.4 (TD)
- a mixed operator using the nonrelativistic field-particle operator together with the relativistic test-particle operator. This operator, which is a simple attempt to extend the validity of the energy and momentum conserving operator above, is not rigorously correct and should be used with care (TD)

1.3 Avalanche operators

The Fokker-Planck formalism describes small-angle collisions in which the particle trajectory is only weakly affected in each event. These are normally dominant in plasmas. For the particular problem of runaway-electron dynamics, however, large-angle collisions (also called *close*, or *knock-on* collisions) are also important since they can lead to an avalanche of runaway generation. Several operators describing this mechanism are included in `CODE` (TD). They act as source terms in the kinetic equation, creating particles in localized areas, and can all be derived from the Boltzmann collision operator in the appropriate limit.

1. The commonly used Rosenbluth-Putvinski avalanche operator [4] is available [1]. The source magnitude is simply proportional to the density of runaways. It has several disadvantages, as discussed in Ref. [2].
2. A version of the above operator that uses the density of 'fast particles' instead of the density of runaways, with various options for how to define a 'fast particle'. Has the advantage that it is usable also when $E < E_\mathrm{c}$ and no runaway region exists (it is also in general less sensitive to changes in the electric field).
3. A more general operator by Chiu et al. [5], taking the energy distribution of the incoming runaways into account (see Ref. [2]).
4. A generalization of the above operator which also accounts for the angular dependence of the incoming distribution.
5. An operator that conserves energy, momentum and particle number by introducing a sink in addition to the source term above. This yet-to-be-published operator (which is still a work in progress) is the only one that takes changes to the incoming runaway into account and consistently removes particles from the bulk to compensate for the generated secondary particles.

Operators 3–5 are more computationally demanding than operators 1–2 (the cost of which is essentially negligible compared to the total computation time). The difference is small for small to medium-sized problems, but increases significantly with problem size.

1.4 Time-dependent plasma parameters

The original version of `CODE` evolved the system assuming the plasma parameters stayed constant throughout the simulation. This does however exclude the possibility to closely study experimental scenarios, or describe for instance hot-tail runaway generation. The possibility to use time-dependent parameters has therefore been introduced, and the implementation is described in Ref. [2] (TD).

The plasma parameters T, n, Z_{eff} and E may be time dependent, which makes it possible to model the evolution in many experimental scenarios. At present, it is not possible to let the magnetic field B, which affects the synchrotron radiation reaction force (see Sec. 1.5), be time-dependent. When time-dependent plasma parameters are used, it is necessary to rebuild and refactorize the matrix describing the system in `CODE` each time some parameter changes. This makes time-dependent runs significantly more computationally demanding than runs with constant parameters.

In order to perform changes to the temperature and density, the collision operator is modified such that the distribution collides with a bulk of the desired T and n, which will quickly make the distribution adapt to the desired shape. Sources/sinks of heat and particles are however also needed, especially if an energy and momentum-conserving collision operator is used. Both of these add or remove heat/particles close to the bulk using weighted Maxwellians. If desired, the particle source can also be used to enforce numerical density conservation.

If the energy and momentum-conserving collision operator is used, the heat source is important also in the case of constant plasma parameters. It then acts to remove any heat supplied by the electric field (through Ohmic heating). This is to make sure the bulk distribution stays close to the desired Maxwellian, which is necessary for the linearized treatment to be stable.

1.5 Synchrotron radiation reaction

The emission of synchrotron radiation by the runaways is associated with a reaction force. This synchrotron radiation reaction is an important (momentum) loss mechanism for relativistic runaways and leads to more narrow runaway distributions perpendicular to the electric/magnetic field (and to a lesser degree a reduction in energy). The relevant equations are derived and discussed in Ref. [6], but were first implemented in `CODE` in connection with Ref. [7] (SS, TD). If the parameters are suitable, the synchrotron radiation reaction force may eventually lead to the formation of a bump on the runaway tail [6, 8].

1.6 Bremsstrahlung radiation reaction

The emission of bremsstrahlung radiation in inelastic collisions between energetic electrons and ions is also associated with a reaction force. The effect of this on the runaway distribution was explored in detail in Ref. [9]. Several models describing this process (of various degrees of sophistication) are included in `CODE`:

1. A model using a simple continuous slowing-down force [10] (SS, TD)
2. A model derived from the Boltzmann collision operator, taking into account the stochastic nature of the bremsstrahlung emission (TD)
3. A simplified version of the above operator, assuming no angular deflection of the electron in the collision (TD)
4. A simplified version of operator 3, also neglecting the contribution from small changes to the electron energy (TD)

Models 2–4 are all derived and discussed in Ref. [9]. Model 2 is by far the most computationally demanding, but models 3 and 4 are also significantly more costly than model 1 (or no bremsstrahlung model at all).

1.7 Partial screening of the nuclear charge

The partial penetration of energetic electrons into the electron cloud surrounding partially ionized atoms can greatly affect the pitch-angle scattering and slowing down of the electron population. These effects are discussed in Refs. [11, 12]. A number of models for partial screening of the nucleus are implemented in `CODE`, ranging from complete screening (the standard case) to full penetration (TD).

1. A Fokker-Planck operator using a simple Tomas-Fermi model to describe the radial variation of the bound-electron density (works for all charge states of all ion species)
2. Similar to 1, but using a Density Functional Theory (DFT) calculation to obtain a better estimate for the free parameter in the Tomas-Fermi model (only works for some charge states of some species)
3. A Fokker-Planck operator with a full DFT model (only works for Ar^{+})
4. Versions of the above three models using a Boltzmann rather than Fokker-Planck operator
5. Full penetration (the incoming electron sees the full nuclear charge). Correct in the ultra-high-energy limit

The modifications necessary for the screening model also includes an energy-dependent Coulomb logarithm ($\ln \Lambda$).

1.8 Nonuniform and expanding computational grids

The points on the finite-difference grid used to discretize the momentum coordinate in `CODE` can be chosen non-uniformly. This is to reduce the computational requirements by having a large grid spacing at high energies (where the scale of variations in f is large), while simultaneously ensuring a high resolution in the bulk region (where f varies on small scales). In principle any mapping $h(s)$, where s is a uniform vector of points on $[0, 1]$, can be used as long as $\mathrm{d}h/\mathrm{d}s$ and $\mathrm{d}^2h/\mathrm{d}s^2$ are known analytically, although only a few select mappings are implemented. The default choice is $h(s) = s^2 + as$, with $a = 0.03$ (SS, TD).

One grid mapping that can be very effective in reducing the requirement on the number of grid points uses a smooth tanh step in spacing, between a dense bulk and a sparse tail (TD). The position and width of the step can be chosen in accordance with the problem at hand. This mapping can be particularly useful when the bulk temperature is low (10's of eV), but the runaways are expected to reach tens of MeV energies.

When using certain grid mappings (currently `yGridMode=0` or `4`), it is possible to dynamically add more points to the grid when the tail of the distribution "reaches" the end of the grid (TD). This allows for the use of a small grid in the beginning (when the tail is small), and can save a lot of computation time in a long simulation. No interpolation of the distribution is required for grid extensions and no errors due to this process are thus introduced. There is also the option to have the grid extend up to the point when a certain particle energy is reached, but then disable any further grid extensions. In connection with grid extensions, the resolution in pitch (the number of Legendre modes) can also be increased, so that the same resolution in $p_\perp$ is retained. `CODE` is also able to keep track of the number of particles that leave the computational domain and include their contribution to the runaway fraction and current (they are all assumed to have $v = c$) (TD).

1.9 Varying time step

By default, `CODE` advances the system in time using a constant time step (TD). Depending on the specific scenario, this might not be very efficient. If for instance an initial transient is expected, followed by an evolution on a much longer time scale, it might be more appropriate to increase the time step with time. The option to use a logarithmically increasing time step is therefore included.

Another scenario where a varying time step is useful is if the temperature changes significantly during the simulation. `CODE` includes a mode in which the time step is kept constant in *instantaneous* thermal collision times, i.e. the time step is adapted based on the evolution of the temperature of the bulk population.

In general, changing the time step means that the matrix in `CODE` has to be rebuilt and refactorized. This is associated with a cost in runtime, however if the plasma parameters are changing, this needs to be done anyway, and if the time scale of the evolution of f increases a lot, more time will likely be saved by increasing the time step (thus reducing the number of time steps needed) than is lost due to matrix rebuilding.

1.10 Restarts / continued calculations

By default, a `CODE` simulation starts from a Maxwellian distribution with the specified T and n. There is however the option of providing the output of a previous simulation as the starting point for a new run (TD). This makes it possible to for instance switch on or off effects such as the avalanche operator in the middle of a run, or try part of a simulation with several different numerical resolutions. Sometimes it can be beneficial to perform a computation in stages (and save the intermediate result), rather than as one very long run.

When starting from a provided initial distribution, only the distribution itself is taken from the previous run; all the settings are those specified for the new run. The computational grids are not required to match, as `CODE` is able to interpolate the old distribution onto the new grid (the error associated with this procedure is usually negligible).

2 Numerical parameters and resolution

The main parameters defining the numerical resolution of a `CODE` run are: N_ξ, N_y and y_max, and for the time-dependent version also $\mathrm{d}t$ and t_max. The choice of these, together with other relevant parameters, will be discussed in this section.

2.1 Momentum grid

The parameter N_y specifies the number of points in the finite-difference momentum grid, and y_max is the maximum momentum described by the grid. Together with the grid mapping (see Sec. 1.8), these two parameters determine the resolution in momentum. The unit of the momentum grid is $y = \gamma v/v_\mathrm{th}$, with $v_\mathrm{th} = \sqrt{2T/m_\mathrm{e}}$ the electron thermal speed (SS, TD).

To allow for the use of time-dependent plasma parameters (TD), in general the grid normalization is not calculated from the current temperature $T(t)$. Instead, a *reference temperature* T_ref is specified, which is used in the definition of y and in the normalization of f (which is discussed in Chapter 3). Similarly, a reference density n_ref is also needed. In the special case of constant plasma parameters, $T_\mathrm{ref} = T$ and $n_\mathrm{ref} = n$ are used by default (although different values can be used if desired).

Using a T_ref that differs substantially from the temperature of the distribution means that either N_y (for $T < T_\mathrm{ref}$) or y_max (for $T > T_\mathrm{ref}$) needs to be large to properly resolve the distribution. If the temperature changes a lot during a simulation, it is therefore recommended that the calculation is split up into several stages, each with their own value of T_ref, and letting `CODE` interpolate the distribution between the different grids (as discussed in Sec. 1.10).

In general, the desired value of y_max is determined by the problem at hand. The number of grid points N_y must then be set to achieve the desired accuracy. As a rule of thumb, $\mathrm{N}_\mathrm{y} > y_\mathrm{max}$ is usually needed for $y_\mathrm{max} \lesssim 300$–$500$, whereas $\mathrm{N}_\mathrm{y} < y_\mathrm{max}$ often is sufficient for $y_\mathrm{max} \gtrsim 800$–$1000$. These recommendations can only provide an initial guess, however, and the resolution and convergence must be investigated for the specific problem considered.

2.1.1 Grid mapping with a tanh step

This grid mapping is more involved than the other available ones and also has several dedicated parameters (TD). These determine the spacing in the bulk (`gridParameter`), the width of the tanh step (`gridStepWidth`), and its location in y (`gridStepPosition`). These make it possible to for instance place the step such that the lower boundary of the runaway region is well resolved (which allows accurate study of the avalanche mechanism that primarily creates particles close the critical momentum for runaway y_c), while still keeping N_y relatively small. With proper choices of these settings, it should be possible to obtain sufficient (if not great) resolution of both a cold (10 eV) bulk and tens-of-MeV runaways using no more than $\mathrm{N}_\mathrm{y} = 1100$–$1200$. Significant trial and error may be needed to find the appropriate parameter values, however.

2.2 Pitch-angle coordinate

In CODE, the pitch-angle coordinate $\xi = \cos\theta$ (with θ the pitch-angle) is discretized through a decomposition into Legendre modes (SS, TD):

$$f(y,\xi) = \sum_{L=0}^{N_\xi - 1} f_L(y) P_L(\xi), \tag{2.1}$$

where f_L is the Lth Legendre mode of f and P_L is the Lth Legendre polynomial. The parameter N_ξ determines the number of modes to use; the more modes the better the resolution. In general, for features of the distribution to be resolved, the Legendre mode with highest mode number must oscillate faster in ξ than the distribution. This means that in order to resolve a feature with a certain extent in $y_\perp$, a higher N_ξ is needed at high energies, where the feature corresponds to a smaller angular extent $\Delta\xi$ than it does at low energies. Since the width of a runaway tail in $y_\perp$ often is quite insensitive to the particle energy, the necessary angular resolution (and thus N_ξ) increasing with $y_{\max}$. The number of Legendre modes required can vary from 10 or less (CODE requires $\mathrm{N}_\xi \geq 3$ to execute properly) to several hundred, depending on the distribution and accuracy requirements.

2.3 Time step

In the normalized units used in CODE (TD), the timescale of variations in the distribution f is primarily determined by the strength of the electric field. For strong fields, a shorter timestep $\mathrm{d}t$ is needed to resolve the distribution evolution, whereas for weak fields or when studying the slow collisional relaxation of a tail of highly energetic particles, the timestep can be very long. Suitable values of $\mathrm{d}t$ can range from fractions of a collision time (if the normalized electric field $\hat{E}$ is of order 0.1 or more) to thousands of collision times or even more. Scenarios involving changes in temperature or density generally require shorter time steps to avoid large abrupt changes to the distribution.

2.4 Other parameters and settings

The boundary condition at $y_{\max}$ may sometimes affect the distribution close to the end of the grid, especially if $y_{\max}$ is not sufficiently large. Various choices for the boundary condition are available (the parameter `yMaxBoundaryCondition`). In certain cases, small oscillations in f can be observed close to the grid boundary as a consequence of the distribution reaching the end of the grid – these can be reduced by introducing some artificial dissipation close to $y_{\max}$ using the parameters `artificialDissipationStrength` and `artificialDissipationWidth` (SS, TD).

Several schemes for implicit time advance are available: Backward Euler, the backward differentiation formula, and the trapezoid rule (TD). In general, these give similar results but may pose slightly different requirements on the time step $\mathrm{d}t$. Only the first two schemes can be used with time-dependent parameters and avalanche sources.

In addition to these, several other parameters are available that influence some specific CODE functionality. If necessary, these will be introduced in the appropriate section. A list of all available settings, together with a description, can be found in the initial comments in the files `CODESettings.m` (TD) and `CODESettings_steadyState.m` (SS).

3 Output and visualization

The results of a `CODE` calculation are returned as a MATLAB struct with a number of fields containing information about the simulation and the resulting distribution. In this chapter, we describe the contents of this struct and introduce the normalizations used. We also discuss the various tools already available for visualizing the results, as well as some utilities for writing a dedicated tool of your own. The inputs and how to perform a `CODE` run are discussed in the next section.

3.1 Normalizations

The inputs to `CODE` are specified in SI units for convenience. The `CODE` equations are however derived in Gaussian units, and the following normalizations should be viewed with this in mind. We here repeat the normalizations, which (except for $\hat{B}$) can be found in the attached papers. A tilde signifies a *reference quantity*, i.e. a quantity which depends on (or is) either T_{ref} or n_{ref}. Note that in steady-state `CODE` or when $T = T_{\mathrm{ref}}$ and $n = n_{\mathrm{ref}}$ in time-dependent `CODE`, these reduce to the expected values. For instance, the reference electron thermal speed is in general given by $\tilde{v_{\mathrm{e}}} = \sqrt{2\tilde{T}/m_{\mathrm{e}}}$, but reduces to the *instantaneous* (or actual) thermal speed $v_{\mathrm{e}} = \sqrt{2T/m_{\mathrm{e}}}$ in the cases mentioned above. The momentum grid uses the unit $y = \gamma v/\tilde{v_{\mathrm{e}}}$. The quantity $\delta = \tilde{v_{\mathrm{e}}}/c$ is sometimes useful to convert between thermal and relativistic units. The distribution is normalized according to

$$F = \frac{(\sqrt{\pi} m \tilde{v_{\mathrm{e}}})^3}{\tilde{n}} f, \tag{3.1}$$

so that it initially takes the value unity at $y = 0$ if $T = T_{\mathrm{ref}}$ and $n = n_{\mathrm{ref}}$. The electric field is

$$\hat{E} = \frac{e}{m\tilde{v_{\mathrm{e}}}\tilde{\nu_{\mathrm{ee}}}} E = \frac{3\sqrt{\pi}}{2} \frac{E}{\tilde{E_{\mathrm{D}}}}, \tag{3.2}$$

with

$$\tilde{\nu_{\mathrm{ee}}} = \frac{16\sqrt{\pi}}{3} \frac{e^4 \tilde{n} \tilde{\ln\Lambda}}{m^2 \tilde{v_{\mathrm{e}}}} \tag{3.3}$$

the reference collision frequency. The time in units of reference collision times is $\hat{t} = \tilde{\nu_{\mathrm{ee}}} t$. The magnetic field is normalized as

$$\hat{B} = \sqrt{\frac{2e^4}{3m_{\mathrm{e}}^3 c^5 \tilde{\nu_{\mathrm{ee}}}}} B, \tag{3.4}$$

which comes from the ratio of the collision time to the radiation time scale.

3.2 Steady-state `CODE` output struct

The output struct of steady-state `CODE` has the following fields (apart from some duplicates of input information):

y — The momentum grid

f — The electron distribution function (normalized as mentioned above) in the format $F = [F_0(\boldsymbol{y}), F_1(\boldsymbol{y}), \ldots, F_{\text{Nxi-1}}(\boldsymbol{y})]^{\mathrm{T}}$, where $\boldsymbol{y}$ represents all the points on the momentum grid

fracRE — The runaway fraction (n_{r}/n)

growthRate — The runaway growth-rate $(\mathrm{d}n_{\mathrm{r}}/\mathrm{d}t)/(\nu_{\mathrm{ee}}n)$. This is calculated as the (steady-state) flux through a bounding surface in momentum space, located well inside the runaway region

averageREEnergy — The average energy carried by a runaway (MeV)

EHat, BHat, delta — The normalized electric field, magnetic field, and thermal speed

EOverEc — The electric field normalized to the Connor-Hastie critical field [13]

settings — A copy of the CODESettings_steadyState object (see Chapter 4) containing all the input information

3.3 Time-dependent `CODE` output struct

The output struct of time-dependent `CODE` has a number of fields, which can have varying size depending on how many time steps are saved during the run (the property `nStepsToReturn`). Several of them are mainly used when starting `CODE` with a supplied starting distribution, or for other technical reasons. They are all described in the file `CODE_timeDependent.m`. Here we discuss the most important ones from the perspective of the user.

y, Nxi, Ny, yMax, BHat, delta — Similar to the respective input or property in the steady-state case, except that the values are the final ones, after any grid extensions, and that y, $\hat{B}$ and δ are normalized using the *reference* temperature and density. These parameters are scalar (except y which is a vector of grid points)

f — The normalized distribution F, similar to the steady-state case, but with one column for each saved timestep. The normalization is based on the reference temperature, so f may take values larger than unity without being incorrect (for instance if $T_{\text{ref}} > T$)

fracRE, averageREEnergyMeV, EHat, EOverEc — Similar to the steady-state case, but vectors of length `nStepsToReturn`. EHat is normalized to the reference quantities

settings — A copy of the CODESettings object (see Chapter 4) containing all the input information

The following fields are all vectors of length `nStepsToReturn`.

times — The times (in the unit specified by the input parameter `timeUnit`) corresponding to the returned time steps (entries in the output vectors)

growthRate — The runaway growth rate $(\mathrm{d}n_{\mathrm{r}}/\mathrm{d}t)/(\tilde{\nu_{\mathrm{ee}}}n_{\mathrm{r}})$. The units are "per reference collision time". Note that this normalization is different than in the steady-state case. This is calculated as a time difference of the runaway fraction

growthRatePerSecond — Runaway growth rate in units of "per second"

currentDensity, currentDensityRE — Total and runaway current densities ($\mathrm{A/m^2}$)

density — Density moment of the distribution (can be compared to n to investigate conservation properties)

EOverED — The electric field normalized to the Dreicer field [14]

fracRECurrent, fracREEnergy — ratio of runaway current density (or energy) to the total current density (or energy)

totalEnergyMeV, totalEnergyJ — The energy moment of the distribution in MeV or Joule

yWeights — Vector of quadrature weights that can be used to integrate any function on the grid. Useful for post processing

3.4 Included visualization tools

Several tools and aids are provided to simplify the analysis of a `CODE` run. These include conversion scripts and visualizations, some executed during runtime and some accessible for post-processing.

3.4.1 In CODE

In steady-state `CODE`, three different plots are available during runtime:

- A plot showing the parallel cut of the distribution, as well as the first few Legendre modes. It also shows how the runaway growth rate depends on where on the grid it is calculated (allowing to potentially adjust this location to obtain a converged value)
- A plot of the entire solution vector f, containing all the Legendre modes
- A plot of the points in the y grid. This can be used to examine nonuniform grids

Time-dependent `CODE` has a number of plots which are described in the file `CODESettings.m`. These include plots showing

- the parallel cut of the distribution, as well as the first few Legendre modes. Also compares the resistivity with the Spitzer value
- the time dependent parameters (if there are any), including various derived quantities. Also shows in real time when the matrix describing the kinetic equation is recalculated due to changes in the parameters
- the evolution of the parallel cut of the distribution in real time
- the runaway generation rate
- the current and its contributions
- information about the avalanche source
- contour plots of the final distribution

3.4.2 External

Scripts are included for convenient conversion between T and δ; E in V/m, E/E_c, E/E_D and $\hat{E}$; and from y and p to MeV. The script `CODESettingsRelations.m` can be used to calculate various other quantities, such as the collision frequency, $\ln\Lambda$ and $\hat{B}$.

Various scripts for visualizing the distribution are also available. The two files `CODE1DPlot.m` and `CODEContourPlot.m` can be used to plot the parallel cut and contour plots of f, respectively, and take a `CODE` output struct as input. The appearance of the plots can be tweaked using a number of properties described in the files. Two scripts are also provided for creating and saving movies based on the time steps in the output of time-dependent `CODE` runs (`DistMovie1D.m` and `DistMovie2D.m`). These are built around the two plotting scripts and therefore take the same arguments.

3.5 Writing your own plotting script

If the enclosed routines do not provide the functionality you want, some tools are also included to make it easy to write your own plotting script:

`LegendrePolynomials.m` — Provides an improved routine for calculating Legendre polynomials. It only returns the information needed by CODE and is about a factor of 100 faster than MATLAB's standard implementation.

`GetParallelCODEDist.m` — Given a `CODE` distribution f, it returns $f_{||}$, the cut along the parallel axis ($y_\perp = 0$).

`GetCODEDistributionPointPP.m` — Calculates the value of f at any point in 2D momentum space by computing the sum in Eq. (2.1). Can for instance be used to make contour plots of the distribution.

If you are interested in a specific Legendre mode, these can be obtained from the output f. Mode L is accessed by the call

```
fL = f( L*Ny + 1:Ny, idTime );
```

The last argument is the index of the time step of interest – in the case of stead-state `CODE`, it can be omitted.

4 Use cases and examples

To run CODE, no installation or compilation is required – just unzip the files into some directory, open MATLAB, and navigate to that directory. For the following examples, it is assumed that the CODE files and supplied scripts are in the current folder, or otherwise in the MATLAB path.

A simulation with CODE is performed by first specifying parameters and settings in a *CODESettings* or *CODESettings_steadyState* object (for time-dependent and steady-state CODE, respectively) and then passing that as an input to CODE (`CODE_timeDependent.m` or `CODE_steadyState.m`). All settings are properties of the settings object and can be accessed using the . operator. Descriptions of all parameters and their possible values can be found in the beginning of the files `CODESettings.m` and `CODESettings_steadyState.m`.

You are encouraged to try the example code out for yourself while going through the examples below. It may also be instructive to read the information in the source files about the various parameters and properties used.

4.1 Steady-state CODE

Performing a basic CODE run is easy. Let's create a CODESettings_steadyState object, specify the number of grid points we want, and do a calculation:

Listing 4.1: Running steady-state CODE

```
o = CODESettings_steadyState(); %Create a CODESettings_steadyState object
o.Ny = 250;                     %Set the number of grid points
CODE_steadyState(o);            %Call CODE to perform the simulation
```

The above example will return an **error**, since all the necessary parameters have not been specified. When created, the object is initialized with default settings for most properties, but not for the physical and resolution parameters (since they are different for each problem). Two convenient functions are provided to set all the parameters necessary to perform the simulation. Other parameters can also easily be modified before starting the simulation:

Listing 4.2: Setting physical and resolution parameters

```
T = 500;  %eV
n = 5e19; %1/m^3
Z = 1;    %Z_eff
E = 1;    %V/m
B = 2;    %T
o.SetPhysicalParameters(T,n,Z,E,B);

Nxi  = 25;
Ny   = 250;
yMax = 75;
o.SetResolutionParameters(Nxi,Ny,yMax);
```

```
o.FMinForPlot = 1e-13;   %Change the plot range for clarity
s = CODE_steadyState(o); %Call CODE to perform the simulation
```

Now it runs! The output is saved in the struct `s`. A plot showing a parallel cut of the distribution appears, but we are also interested in its 2D shape. Let's do a contour plot of the calculated distribution:

Listing 4.3: Make a contour plot of the output

```
figure(555);
CODEContourPlot(s,'PerpMax',50,'Contours',-13:1);
```

4.1.1 Oscillations at the grid boundary

By inspecting the plots, we notice some oscillations in the tail of the distribution. These are not caused by a lack of resolution, but are instead related to the boundary condition at the upper boundary of the domain. Let's try a different option for the boundary condition which might be more suited to this particular problem:

Listing 4.4: Modifying the yMax boundary condition

```
o.yMaxBoundaryCondition = 2; %Change the boundary condition
s = CODE_steadyState(o);     %Re-run the calculation

figure(556);                 %Make a new plot for comparison
CODEContourPlot(s,'PerpMax',50,'Contours',-13:1);
```

The oscillations have disappeared! Another option is to apply some artificial damping with the original boundary condition to try to get rid of the oscillations.

Listing 4.5: Adding artificial dissipation at the yMax boundary

```
o.yMaxBoundaryCondition = 4; %This is the default boundary condition
o.artificialDissipationStrength = 0.02;
o.artificialDissipationWidth    = 2;
s = CODE_steadyState(o);

figure(557);
CODEContourPlot(s,'PerpMax',50,'Contours',-13:1);
```

The oscillations are now much reduced (at the expense of a slight unphysical dip at the very end of the tail).

4.2 Time-dependent CODE

Although the number of available options is much greater for time-dependent `CODE` than for the steady-state version, the basic work flow is similar. The only extra parameters we need to specify are dt and $t_{\max}$, which are related to the time evolution:

Listing 4.6: Running time-dependent CODE

```
o = CODESettings(); %Create a CODESettings object
```

```
T = 500;  %eV
n = 5e19; %1/m^3
Z = 1;    %Z_eff
E = 1;    %V/m
B = 2;    %T
o.SetPhysicalParameters(T,n,Z,E,B);

Nxi  = 25;
Ny   = 250;
yMax = 50;
dt   = 1;   %Thermal collision times
tMax = 500; %Thermal collision times
o.SetResolutionParameters(Nxi,Ny,yMax,dt,tMax);

o.sourceMode = 0;          %Turn off the avalanche source
s = CODE_timeDependent(o); %Call CODE to perform the simulation
```

CODE automatically shows a "live" plot of the evolution of the distribution in the parallel direction – we can see the runaway tail developing. But we are also interested in the evolution of the runaway fraction during the simulation. We can easily make a plot of it from the data contained in the output struct:

Listing 4.7: Access and plot the output

```
fRE = s.fracRE;  %Access the data on the runaway fraction
ts  = s.times;   %Access the times at which the data is saved

figure(777);
plot(ts,fRE);
xlabel('Time (thermal collision times)');
ylabel('n_r/n_e');
```

All the settings used to produce the output are also available in `s` – a copy of the CODESettings object `o` is included in the field `settings`. The call "`s.settings.sourceMode`" would return the value `0`, since we disabled the avalanche source before starting the simulation.

4.2.1 Continuing a calculation (and a detour on handle objects)

Suppose that we are not entirely happy with the previous run. We would like to see how the distribution develops a little later on. Rather than redoing the entire calculation, we can start from the final result in the previous simulation and continue from there. In this case we also need to add more points to the grid, so that the evolution of the tail can be studied. Other parameters can also be changed; in this case we modify the plot limits, but we could just as easily have changed the collision operator, for instance.

Listing 4.8: Starting from a previous CODE distribution

```
o1      = copy(o);         %Make a copy of the settings object
o1.yMax = 100;             %Extend the grid
o1.Ny   = 350;             %Add some more grid points to compensate
o1.Nxi  = 35;              %Add some Legendre modes as well
o1.yMaxForPlot = o1.yMax; %Change the plot limits to show the tail

s1 = CODE_timeDependent(o1,s); %Run code with an initial distribution
```

Note that we did not change the parameter $t_{\max}$ – the calculation always starts at 0, regardless of the initial distribution.

The reason we need to change the plot limit to be able to see the entire computation grid is that before starting the first run, `yMaxForPlot` is automatically set to the maximum of the initial y grid. This information is saved in the object `o`, which we then make a copy of.

The copy is necessary if we want to retain the information in the old object after making changes. This is not how variables normally behave in MATLAB – if we write

```
a=5;
b=a;
b=3;
```

`a` will have the value 5 and `b` the value 3. But the CODEsettings object is what MATLAB calls a *handle object*, which means that unless we make an explicit copy, both `a` and `b` would refer to the same entity:

```
a   = CODESettings();
a.T = 500;
b   = a;
b.T = 100;
```

would result in the temperature in both `a` and `b` having the value `100`. A change to a handle object affects it instantly everywhere in the MATLAB workspace, meaning that when passing the object to a function which changes some property, it is not necessary to explicitly return the object or save it in a new variable.

Some of the properties of the object can be determined automatically. For instance, it is not necessary (although possible) to specify `yMaxForPlot` before calling `CODE`, and it is initially left empty. After running CODE, however, the property has acquired a value – this is a consequence of changes made to the object inside `CODE`. If the same object is used as input to another `CODE` run, the `yMaxForPlot` property will keep that value, and will not be recalculated automatically.

4.2.2 Automatic grid extension

In the above example, the initial distribution is interpolated onto the larger grid. We can avoid this by instead using automated grid extension (see Sec. 1.8). This way, the grid is only extended once the tail "reaches" the end, and grid points are added to the end – the existing grid points are not affected and there is no need to interpolate the distribution.

Listing 4.9: Using automatic grid extension

```
o1 = copy(o);
o1.useAutomaticGridExtension = 1;
s1 = CODE_timeDependent(o1,s);
```

The grid extension only works for certain grid mappings (currently `yGridMode=0` or `4`). There are several properties of the object that affect the grid extension functionality: `gridCheckSkip`, `gridIncreaseFactor`, `extensionXiSlope` and `usefulThreshold` – the last one in particular might need tweaking, depending on the size of the runaway tail (see `CODESettings.m` for more details). Using the parameter `externalBoundaryMeV`, it is also possible to restrict the grid extension by specifying a maximum allowed particle energy – if this parameter is unmodified, the grid extension is allowed to continue indefinitely.

4.2.3 Time-dependent parameters

Suppose we want to study a case where the temperature changes during the simulation. We must then run CODE with time-dependent parameters. Let us consider a case with a modest heating where the temperature changes from $T_1 = 200\,\text{eV}$ to $T_2 = 300\,\text{eV}$ in 1 ms, and all other parameters remain constant:

Listing 4.10: Running CODE with time-dependent parameters

```
%Specify the temperature change
T  = [200,300]; %eV
tT = [0,1];     %ms
timeUnit = 'ms';

%Specify the remaining physical parameters
n  = 5e19; %1/m^3
Z  = 1;    %Z_eff
E  = 1;    %V/m
B  = 2;    %T
tn = 0; tZ = 0; tE = 0; %The time for the constant parameters also needs
                        %to be specified, but the value is arbitrary

%Set the physical parameters
o = CODESettings();
o.SetPhysicalParameters(T,n,Z,E,B,tT,tn,tZ,tE);
o.timeUnitOfParams = timeUnit; %The times are specified in milliseconds

%Set the numerical parameters
Nxi  = 20;
Ny   = 200;
yMax = 75;
dt   = 0.02; %ms
tMax = 1.2;  %ms. Continue for a while after the temperature change
o.SetResolutionParameters(Nxi,Ny,yMax,dt,tMax,timeUnit);

%Change some settings and run CODE
o.sourceMode  = 0;  %Turn off the avalanche source
o.stepSkip    = 5;  %Update the distribution plot more frequently
o.showVarPlot = 1;  %Show a plot of the varying parameters
s = CODE_timeDependent(o);
```

This example works, but the runaway tail has a strange kink in it. This is because, by default the parameter evolution is treated in a stepwise manner. This can be seen by looking at the top-left panel in the plot of the time-dependent parameters: T_1 is used up until $t = 0.5\,\text{ms}$, at which point the temperature immediately jumps to T_2. The reason why this is the default behavior can be seen in the bottom panel of the figure: the matrix describing the system only needs to be rebuilt and refactorized once, since the temperature only changes in one step. This makes the computation efficient, however in this case, the change from T_1 to T_2 is too large to be performed in one step.

We can improve the performance by introducing intermediate steps in the temperature evolution:

Listing 4.11: Using several smaller parameter steps

```
o.T  = 200:10:300; %eV
o.tT = 0:0.1:1;    %ms
s    = CODE_timeDependent(o);
```

In this case, ten steps in temperature appears to be sufficient for a smooth distribution evolution. To make sure, we can let the temperature change in every time step. This is also useful when it is inconvenient to manually specify intermediate steps. This can be done in `CODE` by changing the way the provided temperature evolution is interpolated to the actual time steps:

Listing 4.12: Changing the parameter interpolation method

```
o.T  = [200,300]; %eV
o.tT = [0,1];     %ms
o.interpolationMethod = 'linear'; %Any string supported by interp1 is OK
s    = CODE_timeDependent(o);
```

The temperature now changes in every timestep, and the matrix needs to be rebuilt each time. The runtime is therefore increased significantly.

Although the temperature of the distribution changes in the above examples, the numerical grid remains the same. This is because the grid is defined using a reference temperature, as discussed in Sec. 2.1. If not specified, this is determined automatically from the supplied temperature evolution. In the above example, the reference temperature was $T_{\text{ref}} = 200$ eV, as can be determined from the input or output (`o.TRef` or `s.TRef`). We can rerun the same calculation on a grid with a different reference temperature and should get the same result (assuming the resolution is sufficient), however the normalization of the distribution will be different since it depends on T_{ref}. This can potentially lead to confusion when continuing a calculation where the reference temperature (or density) has changed: when comparing the distribution in the final step of the initial run with that in the first step of the second run, the values of the normalized distribution F will not agree (even though the distribution is actually the same).

The examples given here only provide a starting point – there are many features of `CODE` left to discover. The next section provides some exercises to get you started!

5 Exercises and benchmarks

5.1 Steady-state CODE

Exercise 1. — Using steady-state `CODE`, produce a distribution with 0.15% runaways and make sure the solution is oscillation-free and converged with respect to the numerical parameters.

Exercise 2. Reproduce Fig. 1 a) and b) in the original `CODE` paper [1] (attached).

Exercise 3. Set up a steady-state `CODE` run with the parameters $T = 4\,\mathrm{keV}$, $n = 10^{19}\,\mathrm{m}^{-3}$, $Z = 2$ and $E/E_\mathrm{c} = 2.5$. Let the grid extend to $y_\mathrm{max} = 250$. What qualitative changes can be seen in the tail of the distribution when the magnetic field strength varies between $B = 0$ and $B = 5\,\mathrm{T}$?

5.2 Time-dependent CODE

Exercise 4. Set up a time-dependent `CODE` run with $T = 300\,\mathrm{eV}$, $n = 5 \cdot 10^{19}\,\mathrm{m}^{-3}$, $Z = 1$ and $E = 1\,\mathrm{V/m}$. Use $y_\mathrm{max} = 150$ and run until the tail has reached the end of the grid and “flattened out”. Make sure the solution is oscillation-free and converged. What t_max do you need to use?

Exercise 5. Reproduce Fig. 7a in the the original `CODE` paper [1].

Exercise 6. Make your own version of Fig. 2 in the original `CODE` paper [1], i.e. confirm that the time-dependent solution agrees with the steady-state solution for long enough times. Use parameters of your choice and include at least 7 different CODE time steps in the plot.

Exercise 7. A tokamak is running a massive gas injection scenario and you want to model it using `CODE`. The setup is as follows: first, a runaway population is generated during the flat-top of the discharge; then the density is quickly increased using a gas puff, after which the evolution of the runaway population is studied. You will create a mock-up of the discharge in several steps

a) Assume the entire shot lasts for $t_\mathrm{end} = 12000$ thermal collision times, and that the gas puff happens between $t_1 = 5000$ and $t_2 = 5500$. To make sure the gas-puff phase is modeled in detail, we want to use a shorter time step during that time. Set up a `CODE` calculation consisting of three stages with the parameters $T = 300\,\mathrm{eV}$, $n_i = 5 \cdot 10^{19}\,\mathrm{m}^{-3}$, $Z = 1$ and $E = 1\,\mathrm{V/m}$. Choose a y_max such that electron energies up to 6 MeV can be studied. Use a long time step for $0 < t < t_1$ and $t_2 < t < t_\mathrm{end}$, and a shorter time step for $t_1 < t < t_2$, and use the final distribution in each stage as the starting distribution in the next stage. Hint: Remember that a `CODE` calculation always starts at $t = 0$

b) Model the gas puff by introducing a time-dependent density in the second stage of the calculation. Let the density increase linearly to the final value $n_f = 10^{20}\,\mathrm{m}^{-3}$ at t_2. Keep the density constant at $n = n_f$ in the third stage of the calculation. For convenience, use $n_\mathrm{ref} = n_i$ throughout the entire calculation. Plot the actual density in `CODE` (the field `density` in the output struct) during the three stages to confirm that the correct density evolution is obtained.

c) Plot the runaway fraction and growth rate. What is the effect of the density increase on these quantities? Plot also the evolution of E/E_c and E/E_D. Is there a connection?

Exercise 8. Revisit Exercise 1. Set up a time-dependent `CODE` run with the same parameters and run it with an avalanche source enabled. How long do you have to wait before the runaway fraction has increased by 50% over the steady-state value (which lacks avalanche generation)? Hint: It might be necessary to use logarithmic time stepping or split the run into several pieces with different dt.

Bibliography

[1] M. Landreman, A. Stahl, and T. Fülöp, Numerical calculation of the runaway electron distribution function and associated synchrotron emission, Computer Physics Communications **185**, 847 (2014), DOI: 10.1016/j.cpc.2013.12.004.

[2] A. Stahl, O. Embréus, G. Papp, M. Landreman, and T. Fülöp, Kinetic modelling of runaway electrons in dynamic scenarios, Nuclear Fusion **56**, 112009 (2016), DOI: 10.1088/0029-5515/56/11/112009.

[3] O. Embréus, S. Newton, A. Stahl, E. Hirvijoki, and T. Fülöp, Numerical calculation of the ion runaway distribution, Physics of Plasmas **22**, 052122 (2015), DOI: 10.1063/1.4921661.

[4] M. Rosenbluth and S. Putvinski, Theory for avalanche of runaway electrons in tokamaks, Nuclear Fusion **37**, 1355 (1997), DOI: 10.1088/0029-5515/37/10/I03.

[5] S. Chiu, M. Rosenbluth, R. Harvey, and V. Chan, Fokker-Planck simulations mylb of knock-on electron runaway avalanche and bursts in tokamaks, Nuclear Fusion **38**, 1711 (1998), DOI: 10.1088/0029-5515/38/11/309.

[6] E. Hirvijoki, I. Pusztai, J. Decker, O. Embréus, A. Stahl, and T. Fülöp, Radiation reaction induced non-monotonic features in runaway electron distributions, Journal of Plasma Physics **81**, 475810502 (2015), DOI: 10.1017/S0022377815000513/.

[7] A. Stahl, E. Hirvijoki, J. Decker, O. Embréus, and T. Fülöp, Effective critical electric field for runaway electron generation, Physical Review Letters **114**, 115002 (2015), DOI: 10.1103/PhysRevLett.114.115002.

[8] J. Decker, E. Hirvijoki, O. Embreus, Y. Peysson, A. Stahl, I. Pusztai, and T. Fülöp, Numerical characterization of bump formation in the runaway electron tail, Plasma Physics and Controlled Fusion **58**, 025016 (2016), DOI: 10.1088/0741-3335/58/2/025016.

[9] O. Embréus, A. Stahl, and T. Fülöp, Effect of bremsstrahlung radiation emission on fast electrons in plasmas, New Journal of Physics **18**, 093023 (2016), DOI: 10.1088/1367-2630/18/9/093023.

[10] M. Bakhtiari, G. J. Kramer, M. Takechi, H. Tamai, Y. Miura, Y. Kusama, and Y. Kamada, Role of bremsstrahlung radiation in limiting the energy of runaway electrons in tokamaks, Physical Review Letters **94**, 215003 (2005), DOI: 10.1103/PhysRevLett.94.215003.

[11] L. Hesslow, *Effect of Screened Nuclei on Fast Electron Beam Dynamics*, Master's thesis, Chalmers University of Technology (2016), http://publications.lib.chalmers.se/records/fulltext/236294/236294.pdf.

[12] L. Hesslow, O. Embréus, A. Stahl, T. C. Dubois, G. Papp, S. Newton, and T. Fülöp, Effect of partly-screened nuclei on fast-electron dynamics, Submitted to Phys. Rev. Lett. (2017).

[13] J. Connor and R. Hastie, Relativistic limitations on runaway electrons, Nuclear Fusion **15**, 415 (1975), DOI: 10.1088/0029-5515/15/3/007.

[14] H. Dreicer, Electron and ion runaway in a fully ionized gas I, Physical Review **115**, 238 (1959), DOI: 10.1103/PhysRev.115.238.

Included papers

Computer Physics Communications 185 (2014) 847–855

Contents lists available at ScienceDirect

Computer Physics Communications

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

Numerical calculation of the runaway electron distribution function and associated synchrotron emission

Matt Landreman [a,b,*], Adam Stahl [c], Tünde Fülöp [c]

[a] University of Maryland, College Park, MD, 20742, USA
[b] Plasma Science and Fusion Center, MIT, Cambridge, MA, 02139, USA
[c] Department of Applied Physics, Nuclear Engineering, Chalmers University of Technology and Euratom-VR Association, Göteborg, Sweden

ARTICLE INFO

Article history:
Received 15 May 2013
Received in revised form
11 September 2013
Accepted 3 December 2013
Available online 9 December 2013

Keywords:
Fokker–Planck
Runaway electrons
Relativistic
Plasma
Kinetic
Synchrotron emission

ABSTRACT

Synchrotron emission from runaway electrons may be used to diagnose plasma conditions during a tokamak disruption, but solving this inverse problem requires rapid simulation of the electron distribution function and associated synchrotron emission as a function of plasma parameters. Here we detail a framework for this forward calculation, beginning with an efficient numerical method for solving the Fokker–Planck equation in the presence of an electric field of arbitrary strength. The approach is continuum (Eulerian), and we employ a relativistic collision operator, valid for arbitrary energies. Both primary and secondary runaway electron generation are included. For cases in which primary generation dominates, a time-independent formulation of the problem is described, requiring only the solution of a single sparse linear system. In the limit of dominant secondary generation, we present the first numerical verification of an analytic model for the distribution function. The numerical electron distribution function in the presence of both primary and secondary generation is then used for calculating the synchrotron emission spectrum of the runaways. It is found that the average synchrotron spectra emitted from realistic distribution functions are not well approximated by the emission of a single electron at the maximum energy.

1. Introduction

Due to the decrease in the Coulomb collision cross section with velocity, charged particles in an electric field can "run away" to high energies. In tokamaks, the resulting energetic particles can damage plasma-facing components and are expected to be a significant danger in the upcoming ITER experiment. Electrons are typically the species for which runaway is most significant [1,2], but runaway ions [3] and positrons [4,5] can also be produced. Relatively large electric fields are required for runaway production, and in tokamaks these can arise during disruptions or in sawtooth events. Understanding of runaway electrons and their generation and mitigation is essential to planning future large experiments such as ITER.

Runaway electrons emit measurable synchrotron radiation, which can potentially be used to diagnose the distribution function, thereby constraining the physical parameters in the plasma. The runaway distribution function and associated synchrotron emission depend on the time histories of the local electric field E, temperature T, average ion charge Z, and density n. To infer these quantities (and the uncertainty in these quantities) inside a disrupting plasma using the synchrotron emission, it is necessary to run many simulations of the runaway process, scanning the various physical parameters. To make such a scan practical, computational efficiency is important.

To this end, in this work we demonstrate a framework for rapid computation of the runaway distribution function and associated synchrotron emission for given plasma parameters. The distribution function is computed using a new numerical tool named CODE (COllisional Distribution of Electrons). Physically, the distribution function is determined by a balance between acceleration in the electric field and collisions with both electrons and ions. The calculation in CODE is fully relativistic, using a collision operator valid for both low and high velocities [6] and it includes both primary and secondary runaway electron generation. If primary runaway electron generation dominates, CODE can be used in both time-dependent and time-independent modes. The latter mode of operation, in which a long-time quasi-equilibrium distribution function is calculated, is extremely fast in that it is necessary only to solve a single sparse linear system. Due to its speed and simplicity, CODE is highly suitable for coupling within larger more

* Correspondence to: Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, MD 20742, USA. Tel.: +1 651 366 9306.
E-mail addresses: mlandre1@alum.swarthmore.edu, mattland@umd.edu (M. Landreman).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.12.004

expensive calculations. Besides the inverse problem of determining plasma parameters from synchrotron emission, other such applications could include the study of instabilities driven by the anisotropy of the electron distribution function, and comprehensive modeling of disruptions.

Other numerical methods for computing the distribution function of runaways have been demonstrated previously, using a range of algorithms. Particle methods follow the trajectories of individual marker electrons. Deterministic particle calculations [7] can give insight into the system behavior but cannot calculate the distribution function, since diffusion is absent. Collisional diffusion may be included by making random adjustments to particles' velocities, an approach which has been used in codes such as ASCOT [8] and ARENA [9]. For a given level of numerical uncertainty (noise or discretization error), we will demonstrate that CODE is more than 6 orders of magnitude faster than a particle code on the same computer. Other continuum codes developed to model energetic electrons include BANDIT [10], CQL3D[11,12] and LUKE [13,14]. These sophisticated codes were originally developed to model RF heating and current drive, and contain many features not required for the calculations we consider. For example, CQL3D contains ∼90,000 lines of code and LUKE contains ∼118,000 lines, whereas CODE contains <1200 lines (including comments). While future more elaborate modeling may require the additional features of a code like CQL3D or LUKE, for the applications we consider, we find it useful to have the nimble and dedicated tool CODE. For calculations of non-Maxwellian distribution functions in the context of RF heating, an adjoint method [15] can be a useful technique for efficient solution of linear inhomogeneous kinetic equations. However, the kinetic equation we will consider is nonlinear (if avalanching is included) and homogeneous, so the adjoint method is not applicable.

In several previous studies, a single particle with a representative momentum and pitch-angle is used as an approximation for the entire runaway distribution [16,17] when computing the synchrotron emission. In this paper, we present a computation of the synchrotron radiation spectrum of a runaway distribution in various cases. By showing the difference between these spectra and those based on single particle emission we demonstrate the importance of taking into account the entire distribution.

The remainder of the paper is organized as follows. In Section 2 we present the kinetic equation and the collision operator used. Section 3 details the discretization scheme and calculation of the primary runaway production rate, with typical results shown in Section 4. The avalanche source term and its implementation are described in Section 5. In this section we also demonstrate agreement with an analytic model for the distribution function [18]. Computation of the synchrotron emission spectrum from the distribution function is detailed in Section 6, and comparisons to single-particle emission are given. We conclude in Section 7.

2. Kinetic equation and normalizations

We begin with the kinetic equation

$$\frac{\partial f}{\partial t} - eE\boldsymbol{b}\cdot\nabla_{\boldsymbol{p}} f = C\{f\} + S. \tag{1}$$

Here, $-e$ is the electron charge, E is the component of the electric field along the magnetic field, $\boldsymbol{b} = \boldsymbol{B}/B$ is a unit vector along the magnetic field, $\nabla_{\boldsymbol{p}}$ is the gradient in the space of relativistic momentum $\boldsymbol{p} = \gamma m\boldsymbol{v}$, $\gamma = 1/\sqrt{1-v^2/c^2}$, $v = |\boldsymbol{v}|$ is the speed, m is the electron rest mass, c is the speed of light, C is the electron collision operator, and S represents any sources. All quantities refer to electrons unless noted otherwise. Eq. (1) is the large-aspect-ratio limit of the bounce- and gyro-averaged Fokker–Planck equation (Eq. (2) in [19]). Particle trapping effects are neglected, which is reasonable since runaway beams are typically localized close to the magnetic axis. We may write $\boldsymbol{b}\cdot\nabla_{\boldsymbol{p}} f$ in (1) in terms of scalar variables using

$$\boldsymbol{b}\cdot\nabla_{\boldsymbol{p}} f = \xi\frac{\partial f}{\partial p} + \frac{1-\xi^2}{p}\frac{\partial f}{\partial \xi} \tag{2}$$

where $p = |\boldsymbol{p}|$, and $\xi = \boldsymbol{p}\cdot\boldsymbol{b}/p$ is the cosine of the pitch angle relative to the magnetic field. The distribution function is defined such that the density n is given by $n = \int d^3p\ f$, so f has dimensions of (length × momentum)$^{-3}$, and we assume the distribution function for small momentum to be approximately the Maxwellian $f_{\mathrm{M}} = n\pi^{-3/2}(mv_{\mathrm{e}})^{-3}\exp(-y^2)$ where $v_{\mathrm{e}} = \sqrt{2T/m}$ is the thermal speed, and $y = p/(mv_{\mathrm{e}}) = \gamma v/v_{\mathrm{e}}$ is the normalized momentum.

We use the collision operator from Appendix B of Ref. [6]. This operator is constructed to match the usual nonrelativistic test-particle operator in the limit of $v \ll c$, and in the relativistic limit it reduces to the operator from Appendix A of Ref. [20]. The collision operator is

$$C\{f\} = \frac{1}{p^2}\frac{\partial}{\partial p}p^2\left[C_A\frac{\partial f}{\partial p} + C_F f\right] + \frac{C_B}{p^2}\frac{\partial}{\partial \xi}(1-\xi^2)\frac{\partial f}{\partial \xi} \tag{3}$$

where

$$C_A = \frac{\Gamma}{v}\Psi(x), \tag{4}$$

$$C_B = \frac{\Gamma}{2v}\left[Z + \phi(x) - \Psi(x) + \frac{\delta^4 x^2}{2}\right], \tag{5}$$

$$C_F = \frac{\Gamma}{T}\Psi(x), \tag{6}$$

$\delta = v_{\mathrm{e}}/c$, $x = v/v_{\mathrm{e}} = y/\sqrt{1+\delta^2 y^2}$, Z is the effective ion charge,

$$\Gamma = 4\pi n e^4 \ln\Lambda = (3\sqrt{\pi}/4)\nu_{\mathrm{ee}} v_{\mathrm{e}}^3 m^2 \tag{7}$$

is identical to the Γ defined in Refs. [6,20,21], $\nu_{\mathrm{ee}} = 4\sqrt{2\pi}e^4 n \ln\Lambda/(3\sqrt{m}T^{3/2})$ is the usual Braginskii electron collision frequency, $\phi(x) = 2\pi^{-1/2}\int_0^x \exp(-s^2)\,ds$ is the error function, and

$$\Psi(x) = \frac{1}{2x^2}\left[\phi(x) - x\frac{d\phi}{dx}\right] \tag{8}$$

is the Chandrasekhar function. In the nonrelativistic limit $\delta \to 0$, then $y \to x$, and (3) reduces to the usual Fokker–Planck test-particle electron collision operator.

The collision operator (3) is approximate in several ways. First, it originates from the Fokker–Planck approximation in which small-angle collisions dominate, which is related to an expansion in $\ln\Lambda \gg 1$. Consequently, the infrequent collisions with large momentum exchange are ignored, so the secondary avalanche process is not included at this stage, but will be addressed later in Section 5. Also, the modifications to the Rosenbluth potentials associated with the high-energy electrons are neglected, i.e. collisions with high-energy field particles are ignored.

The kinetic equation is normalized by multiplying through with $m^3 v_{\mathrm{e}}^3 \pi^{3/2}/(\nu_{\mathrm{ee}} n)$, and defining the normalized distribution function

$$F = (\pi^{3/2} m^3 v_{\mathrm{e}}^3/n) f \tag{9}$$

so that $F \to 1$ at $p \to 0$. We also introduce a normalized electric field

$$\hat{E} = -eE/(mv_{\mathrm{e}}\nu_{\mathrm{ee}}) \tag{10}$$

which, up to a factor of order unity, is E normalized by the Dreicer field. The normalized time is $\hat{t} = \nu_{\mathrm{ee}} t$ and the normalized source is $\hat{S} = S m^3 v_{\mathrm{e}}^3 \pi^{3/2}/(\nu_{\mathrm{ee}} n)$. We thereby obtain the dimensionless equation

$$\frac{\partial F}{\partial \hat{t}} + \hat{E}\xi\frac{\partial F}{\partial y} + \hat{E}\frac{1-\xi^2}{y}\frac{\partial F}{\partial y} - \frac{3\sqrt{\pi}}{4}\frac{1}{y^2}\frac{\partial}{\partial y}y^2\left[\frac{\Psi(x)}{x}\frac{\partial F}{\partial y} + 2\Psi(x)F\right] - \frac{3\sqrt{\pi}}{4}\frac{1}{2xy^2}\left[Z+\phi(x)-\Psi(x)+\frac{\delta^4x^2}{2}\right]\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial F}{\partial\xi} = \hat{S}. \tag{11}$$

Notice that this equation has the form of a linear inhomogeneous 3D partial differential equation:

$$\frac{\partial F}{\partial \hat{t}} + MF = \hat{S} \tag{12}$$

for a linear time-independent differential operator M. If a time-independent equilibrium solution exists, it will be given by $F = M^{-1}\hat{S}$.

Since both the electric field acceleration term and the collision operator in the kinetic equation (1) have the form of a divergence of a flux in velocity space, the total number of particles is constant in time in the absence of a source: $(d/dt)\int d^3p\, f = \int d^3p\, S$. However, runaway electrons are constantly gaining energy, so without a source at small p and a sink at large p, no time-independent distribution function will exist. From another perspective, a nonzero source is necessary to find a nonzero equilibrium solution of (11), because when $\hat{S} = 0$, (11) with $\partial/\partial\hat{t} = 0$ is a homogeneous equation with homogeneous boundary conditions. (The boundary conditions are that F be regular at $y = 0$, $\xi = -1$, and $\xi = 1$, and that $f \to 0$ as $y \to \infty$.) Thus, the solution of the time-independent problem $F = M^{-1}\hat{S}$ for $\hat{S} = 0$ would be $F = 0$.

To find a solution, we must either consider a time-dependent problem or include a nonzero S. In reality, spatial transport can give rise to both sources and sinks, and a sink exists at high energy due to radiation. When included, secondary runaway generation (considered in Section 5) also introduces a source. To avoid the added complexity of these sinks and sources and simultaneously avoid the intricacies of time dependence, when restricting ourselves to primary generation we may formulate a time-independent problem as follows. We take $\hat{S} = \alpha e^{-y^2}$ for some constant α, representing a thermal source of particles. Eq. (11) for $\partial/\partial\hat{t} = 0$ may be divided through by α and solved for the unknown F/α. Then α may be determined by the requirement $F(p = 0) = 1$, and F is then obtained by multiplying the solution F/α by this α.

The constant α represents the rate at which particles must be replenished at low energy to balance their flux in velocity space to high energy. Therefore, α is the rate of runaway production. As we do not introduce a sink at high energies, F will have a divergent integral over velocity space.

CODE can also be run in time-dependent mode. Once the velocity space coordinates and the operator M are discretized, any implicit or explicit scheme for advancing a system of ordinary differential equations (forward or backward Euler, Runge–Kutta, trapezoid rule, etc.) may be applied to the time coordinate. (Results shown in this paper are computed using the trapezoid rule.) Due to the diffusive nature of M, numerical stability favors implicit time-advance schemes.

3. Discretization

We first expand F in Legendre polynomials $P_L(\xi)$:

$$F(y,\xi) = \sum_{L=0}^{\infty} F_L(y)P_L(\xi). \tag{13}$$

Then the operation

$$\frac{2L+1}{2}\int_{-1}^{1} P_L(\xi)(\cdot)d\xi \tag{14}$$

is applied to the kinetic equation. Using the identities in the Appendix, we obtain

$$\frac{\partial F_L}{\partial\hat{t}} + \sum_{\ell=0}^{\infty}\left\{\hat{E}\left[\frac{L+1}{2L+3}\delta_{L+1,\ell} + \frac{L}{2L-1}\delta_{L-1,\ell}\right]\frac{\partial}{\partial y} + \frac{\hat{E}}{y}\left[\frac{(L+1)(L+2)}{2L+3}\delta_{L+1,\ell} - \frac{(L-1)L}{2L-1}\delta_{L-1,\ell}\right] - \frac{3\sqrt{\pi}}{4}\frac{\Psi(x)}{x}\delta_{L,\ell}\frac{\partial^2}{\partial y^2} - \frac{3\sqrt{\pi}}{2}\left[\frac{2\Psi(x)}{y} + \frac{dx}{dy}\frac{d\Psi}{dx}\right]\delta_{L,\ell} - \frac{3\sqrt{\pi}}{4}\left[\frac{1}{x}\frac{dx}{dy}\frac{d\Psi}{dx} + \frac{2\Psi(x)}{xy} - \frac{\Psi(x)}{x^2}\frac{dx}{dy} + 2\Psi(x)\right]\delta_{L,\ell}\frac{\partial}{\partial y} + \frac{3\sqrt{\pi}}{8xy^2}\left[Z+\phi(x)-\Psi(x)+\frac{\delta^4x^2}{2}\right]L(L+1)\delta_{L,\ell}\right\}F_\ell = \hat{S}_L \tag{15}$$

where $dx/dy = (1+\delta^2y^2)^{-3/2}$, $d\Psi/dx = 2\pi^{-1/2}e^{-x^2} - (2/x)\Psi(x)$, and $\hat{S}_L = (2L+1)\,2^{-1}\int_{-1}^{1}\hat{S}\, d\xi$ is the appropriate Legendre mode of $\hat{S}(y,\xi) = \sum_{L=0}^{\infty}\hat{S}_L(y)P_L(\xi)$. Note that the collision operator is diagonal in the L index, and the electric field acceleration term is tridiagonal in L.

It is useful to examine the $L = 0$ case of (15), which corresponds to (half) the integral of the kinetic equation over ξ:

$$\frac{\partial F_0}{\partial\hat{t}} - \frac{1}{y^2}\frac{\partial}{\partial y}\left[-y^2\frac{\hat{E}}{3}F_1 + \frac{3\sqrt{\pi}}{4}y^2\left\{\frac{\Psi(x)}{x}\frac{\partial F_0}{\partial y} + 2\Psi(x)F_0\right\}\right] = \hat{S}_0. \tag{16}$$

Applying $4\pi^{-1/2}\int_{y_b}^{\infty} dy\ \ y^2(\cdot)$ for some boundary value y_b, and assuming the source is negligible in this region, we obtain

$$\frac{1}{\nu_{ee}n}\frac{dn_r}{d\hat{t}} = -\frac{4}{\sqrt{\pi}}\left[-y^2\frac{\hat{E}}{3}F_1 + \frac{3\sqrt{\pi}}{4}y^2\left\{\frac{\Psi(x)}{x}\frac{\partial F_0}{\partial y} + 2\Psi(x)F_0\right\}\right]_{y=y_b} \tag{17}$$

where n_r is the number of runaways, meaning the number of electrons with $y > y_b$, so that $n_r = \int_{y>y_b} d^3p\ f = 2\pi\int_{mv_e y_b}^{\infty} dp\ p^2 \int_{-1}^{1} d\xi\ f$. The runaway rate calculated from (17) should be independent of y_b in steady state (as long as y_b is in a region of $\hat{S}_0 = 0$), which can be seen by applying $\int_{y_{b1}}^{y_{b2}} dy\ y^2(\cdot)$ to (16). We find in practice it is far better to compute the runaway production rate using (17) than from the source magnitude α, since the latter is more sensitive to the various numerical resolution parameters.

To discretize the equation in y, we can apply fourth-order finite differences on a uniform grid. Alternatively, for greater numerical efficiency, a coordinate transformation can be applied so grid points are spaced further apart at high energies. The y coordinate is cut off at some finite maximum value $y_{\max}$. The appropriate boundary conditions at $y = 0$ are $dF_0/dy = 0$ and $F_L = 0$ for $L > 0$. For the boundary at large y, we impose $F_L = 0$ for all L. This boundary condition creates some unphysical grid-scale oscillation at large y, which may be eliminated by adding an artificial diffusion $c_1y^{-2}(\partial/\partial y)y^2\exp(-[y-y_{\max}]/c_2)\partial/\partial y$ localized near $y_{\max}$ to the linear operator. Suitable values for the constants are $c_1 = 0.01$ and $c_2 = 0.1$. This term effectively represents a sink for particles,

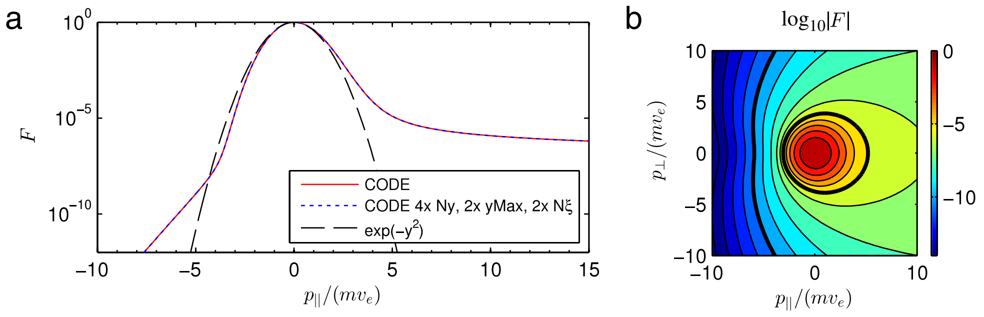

Fig. 1. (Color online) Typical results of CODE, obtained for $\delta = 0.1$, $\hat{E} = 0.1$, and $Z = 1$. (a) Normalized distribution function F for $p_\perp = 0$. Results are plotted for two different sets of numerical parameters ($\{N_y = 300, y_{\max} = 20, N_\xi = 20\}$ and $\{N_y = 1200, y_{\max} = 40, N_\xi = 40\}$). The results overlap completely, demonstrating excellent convergence. A Maxwellian is also plotted for comparison. (b) Contours of F at values 10^z for integer z. Bold contours indicate $F = 10^{-5}$ and 10^{-10}.

which must be included in the time-independent approach due to the particle source at thermal energies. Since this diffusion term is exponentially small away from $y_{\max}$, the distribution function is very insensitive to the details of the ad-hoc term except very near $y_{\max}$. All results shown hereafter are very well converged with respect to doubling the domain size $y_{\max}$, indicating the results are insensitive to the details of the diffusion term.

4. Results for primary runaway electron generation

Fig. 1 shows typical results from a time-independent CODE computation. To verify convergence, we may double N_ξ (the number of Legendre modes), double N_y (the number of grid points in y), and double the maximum y ($y_{\max}$) at fixed y grid resolution (which requires doubling N_y again). As shown by the overlap of the solid red and dashed blue curves in Fig. 1a, excellent convergence is achieved for the parameters used here. Increasing the ad-hoc diffusion magnitude c_1 by a factor of 10 for the parameters of the red curve causes a relative change in the runaway rate (computed using (17) for $y_b = 10$) of $|dn_r/d\hat{t}(c_1 = 0.1) - dn_r/d\hat{t}(c_1 = 0.01)|/[dn_r/d\hat{t}(c_1 = 0.01)] < 10^{-9}$, demonstrating the results are highly insensitive to this diffusion term. As expected, the distribution function is increased in the direction opposite to the electric field ($p_\| > 0$). While the distribution function is reduced in the direction parallel to the electric field ($p_\| < 0$) for $y < 5$, F is actually slightly increased for $y > 5$ due to pitch-angle scattering of the high-energy tail electrons, an effect also seen in Fokker–Planck simulations of RF current drive [22]. The pitch-angle scattering term can be artificially suppressed in CODE, in which case F is reduced in the direction parallel to the electric field for all y.

Fig. 2 compares the distribution functions obtained from the time-independent and time-dependent approaches. At sufficiently long times, the time-dependent version produces results that are indistinguishable from the time-independent version.

For comparison with previously published results, we show in Fig. 3 results by Kulsrud et al. [23], who considered only the non-relativistic case $\delta \to 0$. The agreement with CODE is exceptional. The runaway production rate in CODE is computed using (17) for $y_b = 10$. (Any value of $y_b > 5$ gives indistinguishable results.) Ref. [23] uses a different normalized electric field E_K which is related to $\hat{E}$ by $E_K = 2(3\sqrt{\pi})^{-1}\hat{E}$, and in Ref. [23] the runaway rate is also normalized by a different collision frequency $\nu_K = 3\sqrt{\pi/2}\,\nu_{ee}$. It should also be noted that the Kulsrud computations are time-dependent, with a simulation run until the flux in velocity space reaches an approximate steady state. Each CODE point shown in Fig. 3 took approximately 0.08 s on a single Dell Precision laptop with Intel Core i7-2860 2.50 GHz CPU and 16 GB memory, running in MATLAB. Faster results could surely be obtained using a lower-level language.

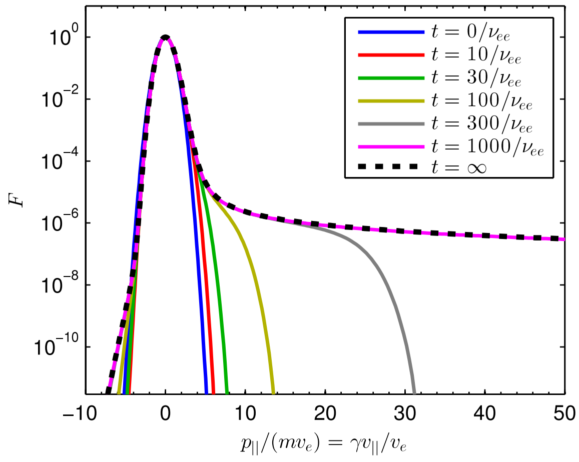

Fig. 2. (Color online) The distribution function from time-dependent CODE at various times. At $t = 1000/\nu_{ee}$, the distribution function is indistinguishable from the solution obtained using the time-independent scheme ($t = \infty$) over the momentum range shown.

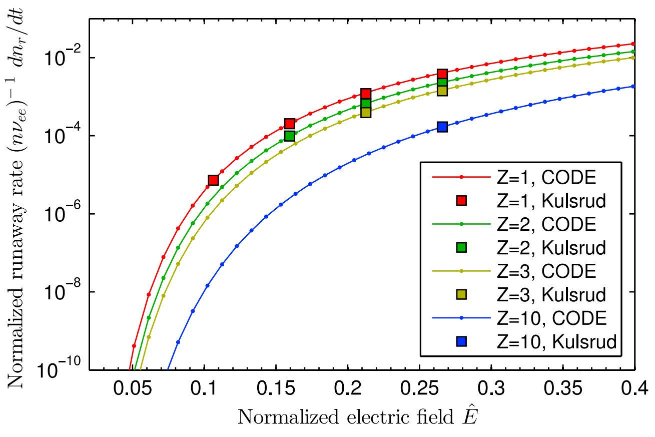

Fig. 3. (Color online) Benchmark of CODE in the nonrelativistic limit $\delta \to 0$ against data in Table 1 of Ref. [23].

To emphasize the speed of CODE, we have directly compared it to the ARENA code [9] for computing the runaway rate using the parameters considered in [23]. ARENA is a Monte Carlo code written in Fortran 90 and designed specifically to compute the runaway distribution function and runaway rate. Detailed description of the current version of ARENA is given in Refs. [24,25]. Both codes were run on a single thread on the same computer with an Intel Xeon 2.0 GHz processor. ARENA required 49,550 s to reproduce the left square point in Fig. 3, and 5942 s to reproduce the top-right square point. 50,000 particles were required for reasonable convergence. For comparison, at a similar level of convergence, time-independent CODE required 0.00106 s and 0.000696 s for the two respective points, and time-dependent CODE required 0.0307 s and 0.00082 s respectively. Thus, for these parameters, both time-independent and time-dependent CODE require less than 1.5×10^{-7} as many cpu-hours as ARENA for the same hardware.

5. Secondary runaway electron generation

In the previous sections we used the Fokker–Planck collision operator, which includes "distant" (large impact parameter) collisions but not "close" (small impact parameter) collisions in which a large fraction of energy and momentum are transferred between the colliding particles. Close collisions are infrequent compared to distant collisions, and are therefore neglected in the Fokker–Planck operator. However, close collisions may still have a significant effect on runaway generation, since the density of runaways is typically much smaller than the density of thermal electrons which may be accelerated in a close collision. The production of runaways through close collisions is known as secondary production, or as avalanche production since it may occur with exponential growth. To simulate secondary generation of energetic electrons, we use a source term derived in [19], starting from the Møller scattering cross-section in the $w \gg 1$ limit, with $w = p/(mc) = \delta y$ a normalized momentum. In this limit, the trajectories of the primary electrons are not much deflected by the collisions. The source then takes the form

$$S = \frac{n_r}{4\pi\tau \ln \Lambda}\delta(\xi - \xi_2)\frac{1}{w^2}\frac{\partial}{\partial w}\left(\frac{1}{1-\sqrt{1+w^2}}\right), \tag{18}$$

where $1/\tau = 4\pi n e^4 \ln \Lambda/(m^2c^3)$ is the collision frequency for relativistic electrons, n_r is the density of the fast electrons and $\xi_2 = w/(1+\sqrt{1+w^2})$ is the cosine of the pitch angle at which the runaway is born. (Our Eq. (18) differs by a factor m^3c^3 compared to the source in Ref. [19] since we normalize our distribution function as $n = \int d^3p\, f$ instead of $n = \int d^3w\, f$. There is also a factor of 2π difference due to the different normalization of the distribution function.)

Due to the approximations used to derive S, care must be taken in several regards. First, to define n_r in (18), it is not clear where to draw the dividing line in velocity space between runaways and non-runaways. One possible strategy for defining n_r is to compute the separatrix in velocity space between trajectories that will have bounded and unbounded energy in the absence of diffusion, and to define the runaway density as the integral of f over the latter region [26]. This approach may somewhat overestimate the true avalanche rate, since it neglects the fact that some time must elapse between an electron entering the runaway region and the electron gaining sufficient energy to cause secondary generation. As most runaways have $\xi \approx 1$, we may approximate the separatrix by setting $dw/dt = 0$ where $dw/dt = eE/(mc) - (1 + 1/w^2)/\tau$ defines the trajectory of a particle with $\xi = 1$, neglecting diffusion in momentum and pitch angle. The runaway region is therefore $w > w_c$ where $w_c = [(E/E_c) - 1]^{-1/2}$ and $E_c = mc/(e\tau)$ is the critical field, and so we take $n_r = 2\pi m^3c^3 \int_{-1}^{1} d\xi \int_{w_c}^{\infty} dw\, w^2 f$. (We cannot define n_r by the time integral of (17), since (17) is no longer valid when S is nonzero away from $p \approx 0$.) A second deficiency of (18) is that S is singular at $w \to 0$, so the source must be cut off below some threshold momentum. Following Ref. [12], we choose the cutoff to be w_c. Neither of the cutoffs discussed here would be necessary if a less approximate source term than (18) were used, but derivation of such an operator is beyond the scope of this paper.

Normalizing and applying (14) as we did previously for the other terms in the kinetic equation, the source included in CODE becomes

$$\hat{S}_L = \frac{n_r}{n}\frac{3\pi\delta^5}{16\ln\Lambda}\frac{2L+1}{2}P_L(\xi_2)\frac{1}{(1-\sqrt{1+w^2})^2\sqrt{1+w^2}y}. \tag{19}$$

When secondary generation is included, CODE must be run in time-dependent mode.

To benchmark the numerical solution of the kinetic equation including the above source term by CODE, we use the approximate analytical expression for the avalanche distribution function derived in Section II of Ref. [18]:

$$f_{aa}(w_\parallel, w_\perp) = \frac{k}{w_\parallel}\exp\left(\tilde{\gamma}t - \frac{\tilde{\gamma}\tau}{E/E_c - 1}w_\parallel - \left[\frac{E/E_c - 1}{Z+1}\right]\frac{w_\perp^2}{2w_\parallel}\right) \tag{20}$$

where k is a constant. The quantity $\tilde{\gamma}$ is the growth rate $\tilde{\gamma} = (1/f)\partial f/\partial t$, which must be independent of both time and velocity for (20) to be valid. Eq. (20) is also valid only where $p_\parallel \gg p_\perp$ and in regions of momentum space where S is negligible. (This restriction is not a major one since $S = 0$ everywhere except on the $\xi = \xi_2$ curve.) If most of the runaway distribution function is accurately described by (20), then we may approximate $n_r \approx \int d^3p\, f_{aa} = 2\pi m^3c^3 \int_{-\infty}^{\infty} dw_\parallel \int_0^\infty dw_\perp\, w_\perp f_{aa}$, giving $\tilde{\gamma} = (1/n_r)dn_r/dt$ and

$$k = n_r e^{-\tilde{\gamma}t}\frac{\tau}{2\pi m^3c^3(1+Z)} \tag{21}$$

where $n_r e^{-\tilde{\gamma}t}$ is constant. (Eq. (21) may be inaccurate in some situations even if (20) is accurate in part of velocity-space, because (21) requires (20) to apply in *all* of velocity-space.) Figs. 4 and 5 show comparisons between distributions from CODE and (20)–(21) for two different sets of parameters. More precisely, the quantity plotted in Fig. 4–5 is $\log_{10}(m^3c^3f/n_r)$. To generate the figures, CODE is run for a sufficiently long time that $(1/f)\partial f/\partial t$ becomes approximately constant. The resulting numerical value of $(1/n_r)dn_r/dt$ is then used as $\tilde{\gamma}$ when evaluating (20)–(21). For a cleaner comparison between CODE and analytic theory in these figures, we minimize primary generation in CODE in these runs by initializing f to 0 instead of to a Maxwellian. For both sets of physical parameters, the agreement between CODE and (20) is excellent in the region where agreement is expected: where $p_\parallel \gg p_\perp$ and away from the curve $\xi = \xi_2$.

6. Synchrotron emission

Using the distribution functions calculated with CODE, we now proceed to compute the spectrum of emitted synchrotron radiation. Due to the energy dependence of the emitted synchrotron power, the emission from runaways completely dominates that of the thermal particles. The emission also depends strongly on the pitch-angle of the particle. In a cylindrical plasma geometry, the emitted synchrotron power per wavelength at wavelength λ from a single highly energetic particle is given by [27]

$$\mathcal{P}(\gamma, \gamma_\parallel, \lambda) = \frac{4\pi}{\sqrt{3}}\frac{ce^2}{\lambda^3\gamma^2}\int_{\lambda_c/\lambda}^{\infty} K_{5/3}(l)\, dl, \tag{22}$$

where the two-dimensional momentum of the particle is determined by γ and $\gamma_\parallel = 1/\sqrt{1 - v_\parallel^2/c^2}$, $K_\nu(x)$ is a modified Bessel function of the second kind, and

$$\lambda_c = \frac{4\pi}{3}\frac{mc^2\gamma_\parallel}{eB\gamma^2}, \tag{23}$$

where B is the magnetic field strength.

Using CODE we will demonstrate that the synchrotron radiation spectrum from the entire runaway distribution is substantially

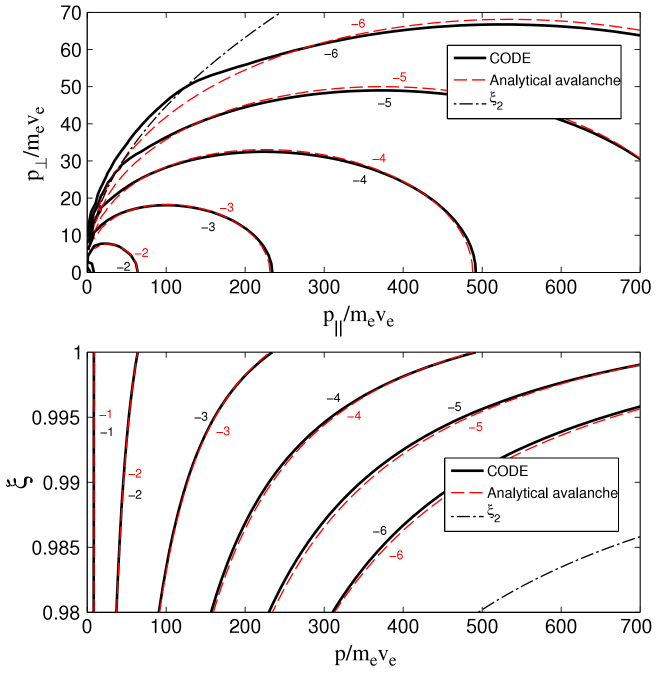

Fig. 4. (Color online) Contour plots of the long-time distribution function from CODE (shown in two different coordinate systems), obtained for $E/E_c = 40$ ($\hat{E} = 0.532$), $Z = 3$, $\delta = 0.1$ and $t = 5000/\nu_{ee}$. Results are plotted for the numerical parameters $N_y = 1500$, $y_{\max} = 1500$ and $N_\xi = 100$, with time step $dt = 10/\nu_{ee}$. The analytical distribution in (20)–(21) for the same physical parameters is also plotted for comparison, together with part of the curve where avalanche runaways are created ($\xi = \xi_2$).

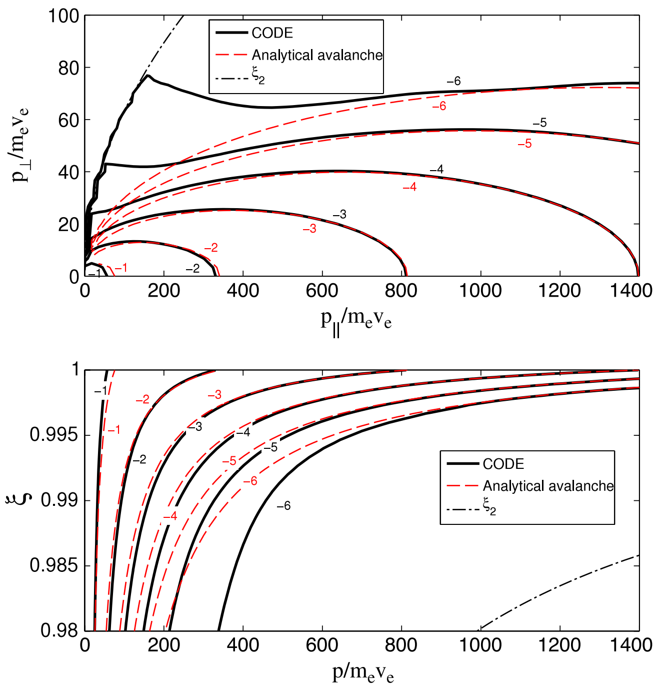

Fig. 5. (Color online) Contour plots of the long-time distribution function from CODE (shown in two different coordinate systems), obtained for $E/E_c = 100$ ($\hat{E} = 0.332$), $Z = 1$, $\delta = 0.05$ and $t = 6000/\nu_{ee}$. Results are plotted for the numerical parameters $N_y = 1500$, $y_{\max} = 3000$ and $N_\xi = 180$, with time step $dt = 25/\nu_{ee}$. The analytical distribution in (20)–(21) is also plotted for comparison, together with part of the curve where avalanche runaways are created ($\xi = \xi_2$).

different from the spectrum obtained from a single particle approximation. By transforming to the more suitable coordinates w and ξ, related to γ and $\gamma_\parallel$ through $\gamma^2 = 1 + w^2$ and $\gamma_\parallel^2 = (1 - w^2\xi^2/(1+w^2))^{-1}$, and integrating (22) over the runaway region R in momentum space, we obtain the total synchrotron emission from the runaway distribution. Normalizing to n_r, we find that the average emitted power per runaway particle at a wavelength λ is given by

$$P(\lambda) = \frac{2\pi}{n_r} \int_R f(w,\xi)\, \mathcal{P}(w,\xi,\lambda)\, w^2 \mathrm{d}w\, \mathrm{d}\xi. \tag{24}$$

Up to a factor ecA, where A is the area of the runaway beam, normalization by n_r is equivalent to normalization by the runaway current, since the emitting particles all move with velocity $\approx c$.

The per-particle synchrotron spectra generated by the CODE distributions in Figs. 4 and 5 were calculated using this formula, and are shown in Fig. 6, together with the spectra radiated by electron distributions for other electric field strengths. For the physical parameters used, we note that the peak emission occurs between 7 and 25 μm. The synchrotron spectra show a decrease in per-particle emission with increasing electric field strength. Even though a stronger electric field leads to more particles with high energy (and thus high average emission), it also leads to a more narrow distribution in pitch-angle. This reduction in the number of particles with large pitch-angle leads to a decrease in average emission. Both figures confirm that the average emission is reduced for higher electric fields, implying that the latter mechanism has the largest impact on the spectrum.

In calculating the spectra, the runaway region of momentum space, R, was defined such that the maximum particle momentum was $w_{\max} = 50$ (which translates to $y_{\max} = 500$ and $y_{\max} = 1000$ respectively for the cases shown in Figs. 4 and 5), corresponding to a maximum particle energy of $\simeq$25 MeV. Physically the cutoff at large energy can be motivated by the finite life-time of the accelerating electric field and the influence of loss mechanisms such as radiation. Since the radiated synchrotron power increases with both particle energy and pitch, this truncation of the distribution is necessary to avoid infinite emission, although the precise value for the cutoff depends on the tokamak and on discharge-specific limitations to the maximum runaway energy. For the low-energy boundary of R, $w_{\min} = w_c = [(E/E_c) - 1]^{-1/2}$ was used, and all particles with $\xi \in [0, 1]$ were included. Although no explicit cutoff was imposed in ξ, the distribution decreases rapidly as this parameter decreases from 1 (as can be seen in Figs. 4 and 5) and there are essentially no particles below some effective cutoff value.

Fig. 6 also shows the synchrotron spectrum from single particles with momentum corresponding to the maximum momentum of the distributions ($w = 50$), and several values of pitch-angle $y_\perp/y_\parallel$. This single-particle "approximation" is equivalent to using a 2D δ-function model of the distribution, as was done in Refs. [28,16] (and with some modification in [17]). The figure shows that this approximation significantly overestimates the synchrotron emission per particle. Note that in the figure, the values for the emitted power per particle were divided by a large number to fit in the same scale. The overestimation is not surprising, since the δ-function approximation effectively assumes that all particles emit as much synchrotron radiation as one of the most strongly emitting particles in the actual distribution. The figure also shows that the δ-function approximation leads to a different spectrum shape, with the wavelength of peak emission usually shifted towards shorter wavelengths. In order to obtain an accurate runaway synchrotron spectrum, it is thus crucial to use the full runaway distribution in the calculation.

In the cases shown in Fig. 6, the runaway electron distribution is dominated by secondary generation. For comparison, in Fig. 7–8 we show a case where the distribution is dominated by primary generation. Fig. 7a shows contours of a distribution from primaries only, together with a distribution obtained with the avalanche

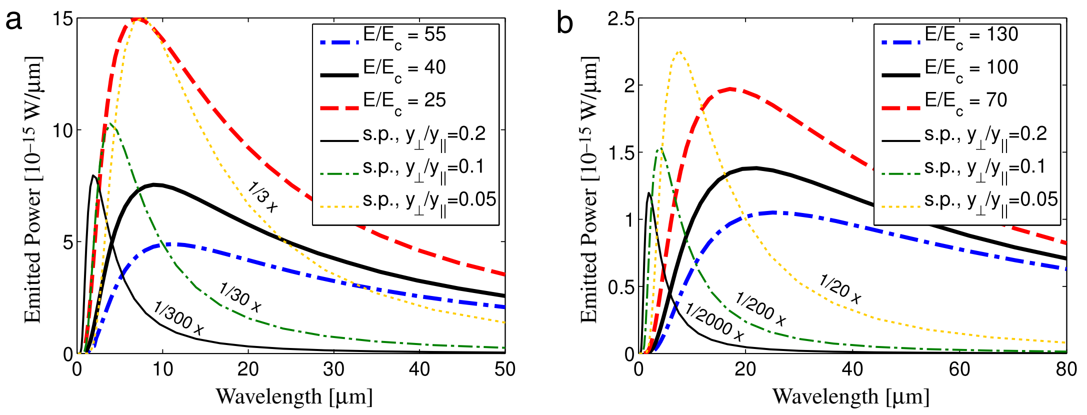

Fig. 6. (Color online) Synchrotron spectra (average emission per particle) for the runaway distributions in (a) Fig. 4 and (b) Fig. 5. Emission spectra from the CODE distributions in Figs. 4 and 5 are shown in solid black, together with spectra from distributions with varying electric field strength but otherwise identical physical parameters. A magnetic field of $B = 3$ T was used. The synchrotron spectra from single particles with $w = 50$ and various pitch angles are also shown. (These single-particle spectra are the same in figures a and b, as the particle parameters are independent of simulation settings.)

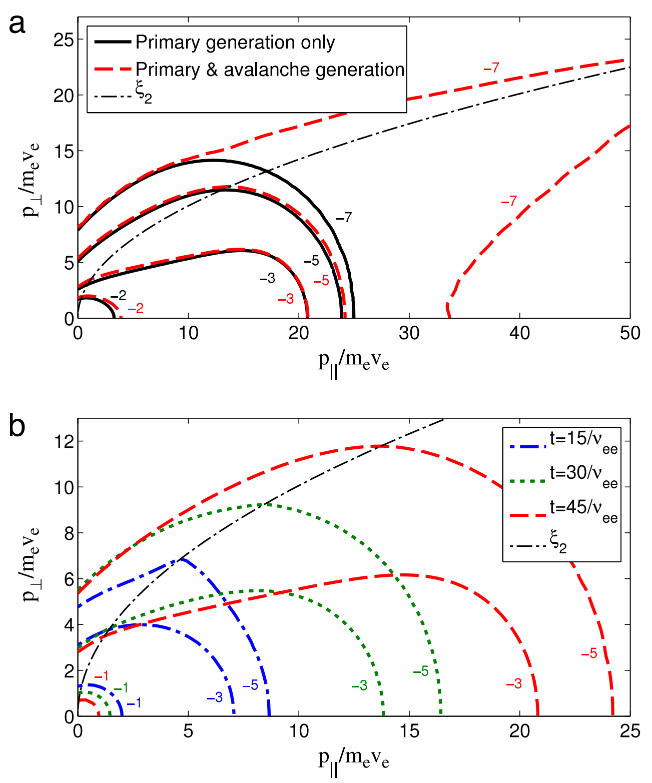

Fig. 7. (Color online) (a) Contour plots of the distribution function from primaries only (black solid line), together with a distribution obtained with the avalanche source enabled (dashed red line), for $E/E_c = 10$ ($\hat{E} = 0.523$), $Z = 1$ and $\delta = 0.2$ at $t = 45/\nu_{ee}$. The quantity plotted is $\log_{10}(F)$. Results are plotted for the numerical parameters $N_y = 20$, $y_{\max} = 100$ and $N_\xi = 130$, with time step $dt = 0.02/\nu_{ee}$. (b) Contour plots of the distribution function at different times with the avalanche source enabled, using the above parameters.

source enabled, and confirms that the distribution is dominated by primaries, except for a small number of secondary runaways generated along the curve $\xi = \xi_2$. Fig. 7(b) shows contour plots with the avalanche source enabled, for three different times. The physical parameters used in Fig. 7 are temperature $T = 10$ keV, density $n = 5 \times 10^{19}$ m^{-3}, effective charge $Z = 1$, and electric field $E = 0.45$ V/m. The collision time in this case is 0.39 ms, so the times shown in the figure correspond to 5.9, 11.8 and 17.6 ms, which correlates well with the time-scale of the electric field spike for a typical disruption in DIII-D (see e.g. Fig. 2 in [29]). Fig. 8 compares

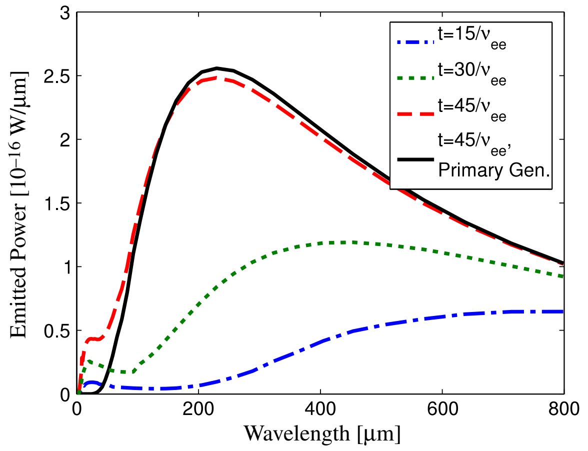

Fig. 8. (Color online) Synchrotron spectra (average emission per particle) for the runaway distributions shown in Fig. 7 for magnetic field $B = 3$ T.

the synchrotron spectra from the distributions shown in Fig. 7. The main difference compared to the case dominated by secondary generation (Fig. 6) is the generally longer wavelengths in the spectrum. The reason is the low runaway electron energy ($w \lesssim 10$) in the runaway electron distribution in this case. The small peak at short wavelengths in the spectra including the avalanche source stems from the secondary runaways generated at $\xi = \xi_2$ (visible in Fig. 7(a)).

In principle, we may also use the synchrotron spectra from distributions calculated through CODE to estimate the maximum energy of the runaways in existing tokamaks. However, due to the region of sensitivity of the available detectors, there is only a limited wavelength range in which calculated spectra can be fitted to experimental data in order to determine the maximum runaway energy. The available range often corresponds to the short wavelength slope of the spectrum, where the emitted power shows an approximately linear dependence on wavelength. Indeed, the short-wavelength spectrum slope has been used to estimate the maximum runaway energy in experiments [16]. If the runaway distribution function is approximated by a δ-function at the maximum available energy and pitch angle, there is a monotonic relationship between the short-wavelength spectrum slope and the maximum particle energy (at fixed pitch angle). Such a relationship holds because increasing the particle energy leads to more emission at shorter wavelengths, resulting in a shift of the wavelength of peak emission towards shorter wavelengths, and a corresponding change in the spectrum slope.

Using an integrated synchrotron spectrum from a CODE distribution is much more accurate than the single particle approximation, but it also introduces additional parameters ($\hat{E}$, δ, Z). If the physical parameters are well known, a unique relation still holds between the spectrum slope and the maximum particle energy. During disruptions however, many parameters (like the temperature and the effective charge) are hard to measure with accuracy. As the shape of the underlying distribution depends on the values of the parameters, the synchrotron spectrum will do so as well. This complexity is apparent in Fig. 6, where the single particle approximation produces identical results in the two cases, whereas the spectra from the complete distributions are widely different. The dependence on distribution shape makes it possible in principle for two sets of parameters to produce the same spectrum slope for different maximum energies. Given this insight, using the complete runaway distribution when modeling experimentally obtained spectra is necessary for an accurate analysis and reliable fit of the maximum particle energy. In this context, CODE is a very useful tool with the possibility to contribute to the understanding of runaways and their properties.

7. Conclusions

In this work, we have computed the synchrotron emission spectra from distribution functions of runaway electrons. The distribution functions are computed efficiently using the CODE code. Both primary (Dreicer) and secondary (avalanche) generation are included. A Legendre spectral discretization is applied to the pitch-angle coordinate, with high-order finite differences applied to the speed coordinate. A nonuniform speed grid allows high resolution of thermal particles at the same time as a high maximum energy without a prohibitively large number of grid points. If secondary generation is unimportant, the long-time distribution function may be calculated by solving a single sparse linear system. The speed of the code makes it feasible to couple to other codes for integrated modeling of complex processes such as tokamak disruptions. CODE has been benchmarked against previous analytic and numerical results in appropriate limits, showing excellent agreement. In the limit of strong avalanching, CODE demonstrates agreement with the analytic distribution function (20) from Ref. [18].

The synchrotron radiation spectra are computed by convolving the distribution function with the single-particle emission. We find that the radiation spectrum from a single electron at the maximum energy can differ substantially from the overall spectrum generated by a distribution of electrons. Therefore, experimental estimates of maximum runaway energy based on the single-particle synchrotron spectrum are likely to be inaccurate. A detailed study of the distribution-integrated synchrotron spectrum and its dependences on physical parameters can be found in [30].

In providing the electron distribution functions (and thus knowledge of a variety of quantities through its moments), the applicability of CODE is wide, and the potential in coupling CODE to other software, e.g. for modeling of runaway dynamics in disruptions, is promising. For a proper description of the runaways generated in disruptions it is important to take into account the evolution of the radial profiles of the electric field and fast electron current self-consistently. This can be done by codes such as GO, initially described in Ref. [31] and developed further in Refs. [32,33]. GO solves the equation describing the resistive diffusion of the electric field in a cylindrical approximation coupled to the runaway generation rates. In the present version of GO, the runaway rate is computed by approximate analytical formulas for the primary and secondary generation. Using CODE, the analytical formulas can be replaced by a numerical solution for the runaway rate which would have several advantages. One advantage would be that Dreicer, hot-tail and secondary runaways could all be calculated with the same tool, avoiding the possibilities for double-counting and difficulties with interpretations of the results. Also, in the present version of GO, it is assumed that all the runaway electrons travel at the speed of light, an approximation that can be easily relaxed using CODE, which calculates the electron distribution in both energy and pitch-angle. Most importantly, the validity region of the results would be expanded, as the analytical formulas are derived using various assumptions which are often violated in realistic situations. The output would be a self-consistent time and space evolution of electric field and runaway current, together with the electron distribution function. This information can then be used for calculating quantities that depend on the distribution function, such as the synchrotron emission or the kinetic instabilities driven by the velocity anisotropy of the runaways.

Acknowledgments

This work was supported by US Department of Energy grants DE-FG02-91ER-54109 and DE-FG02-93ER-54197, by the Fusion Energy Postdoctoral Research Program administered by the Oak Ridge Institute for Science and Education, and by the European Communities under Association Contract between EURATOM and *Vetenskapsrådet*. The views and opinions expressed herein do not necessarily reflect those of the European Commission. The authors are grateful to J. Rydén, G. Csépány, G. Papp, P. Helander, E. Nilsson, J. Decker, and Y. Peysson for fruitful discussions.

Appendix. Integrals of Legendre polynomials

Here we list several identities for Legendre polynomials which are required for the spectral pitch-angle discretization. To evaluate the ξ integral of the $\xi \partial F/\partial y$ term in (11), we use the recursion relation

$$\xi\, P_L(\xi) = \frac{L+1}{2L+1} P_{L+1}(\xi) + \frac{L}{2L+1} P_{L-1}(\xi) \tag{A.1}$$

where P_{L-1} is replaced by 0 when $L = 0$. Applied to the relevant integral in (11), and noting the orthogonality relation $(2L+1)2^{-1}\int_{-1}^{1} P_L(\xi)P_\ell(\xi)d\xi = \delta_{L,\ell}$, we find

$$\frac{2L+1}{2}\int_{-1}^{1} d\xi\ \xi\ P_L(\xi)P_\ell(\xi) = \frac{L+1}{2L+3}\delta_{\ell,L+1} + \frac{L}{2L-1}\delta_{\ell,L-1}. \tag{A.2}$$

Similarly, to evaluate the ξ integral of the $\partial F/\partial \xi$ term in (11), we use the recursion relation

$$(1-\xi^2)(dP_L/d\xi) = LP_{L-1}(\xi) - L\xi P_L(\xi) \tag{A.3}$$

to obtain

$$\frac{2L+1}{2}\int_{-1}^{1} d\xi\ P_L(\xi)(1-\xi^2)\frac{dP_\ell}{d\xi} = \frac{(L+1)(L+2)}{2L+3}\delta_{\ell,L+1} - \frac{(L-1)L}{2L-1}\delta_{\ell,L-1}. \tag{A.4}$$

Finally, the pitch-angle scattering collision term gives the integral

$$\frac{2L+1}{2}\int_{-1}^{1} d\xi\ P_L(\xi)\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial}{\partial\xi}P_\ell(\xi) = -(L+1)L\delta_{L,\ell}. \tag{A.5}$$

References

[1] H. Dreicer, Phys. Rev. 117 (1960) 329.
[2] J.W. Connor, R.J. Hastie, Nucl. Fusion 15 (1975) 415.
[3] H.P. Furth, P.H. Rutherford, Phys. Rev. Lett. 28 (1972) 545.

[4] P. Helander, D. Ward, Phys. Rev. Lett. 90 (2003) 135004.
[5] T. Fülöp, G. Papp, Phys. Rev. Lett. 51 (2012) 043004.
[6] G. Papp, M. Drevlak, T. Fülöp, P. Helander, Nucl. Fusion 51 (2011) 043004.
[7] J.R. Martín-Solís, J.D. Alvarez, R. Sánchez, B. Esposito, Phys. Plasmas 5 (1998) 2370.
[8] J.A. Heikkinen, S.K. Sipilä, T.J.H. Pättikangas, Comput. Phys. Comm. 76 (1993) 215.
[9] L.-G. Eriksson, P. Helander, Comput. Phys. Comm. 154 (2003) 175.
[10] M.R. O'Brien, M. Cox, D.F.H. Start, Nucl. Fusion 26 (1986) 1625.
[11] S.C. Chiu, M.N. Rosenbluth, R.W. Harvey, V.S. Chan, Nucl. Fusion 38 (1998) 1711.
[12] R.W. Harvey, V.S. Chan, S.C. Chiu, T.E. Evans, M.N. Rosenbluth, Phys. Plasmas 7 (2000) 4590.
[13] Y. Peysson, J. Decker, R.W. Harvey, AIP Conf. Proc. 694 (2003) 495.
[14] Y. Peysson, J. Decker, AIP Conf. Proc. 1069 (2008) 176.
[15] C. Karney, N. Fisch, Phys. Fluids 29 (1986) 180.
[16] R. Jaspers, N.J.L. Cardozo, A.J.H. Donné, H.L.M. Widdershoven, K.H. Finken, Rev. Sci. Instrum. 72 (2001) 466.
[17] J.H. Yu, E.M. Hollmann, N. Commaux, N.W. Eidietis, D.A. Humphreys, A.N. James, T.C. Jernigan, R.A. Moyer, Phys. Plasmas 20 (2013) 042113.
[18] T. Fülöp, G. Pokol, P. Helander, M. Lisak, Phys. Plasmas 13 (2006) 062506.
[19] M.N. Rosenbluth, S.V. Putvinski, Nucl. Fusion 37 (1997) 1355.
[20] P. Sandquist, S.E. Sharapov, P. Helander, M. Lisak, Phys. Plasmas 13 (2006) 072108.
[21] C. Karney, N. Fisch, Phys. Fluids 28 (1985) 116.
[22] C. Karney, N. Fisch, Phys. Fluids 22 (1979) 1817.
[23] R.M. Kulsrud, Y.-C. Sun, N.K. Winsor, H.A. Fallon, Phys. Rev. Lett. 31 (1973) 690.
[24] J. Rydén, Monte Carlo Simulation of Runaway Electrons, Master's Thesis, Chalmers University of Technology (2012), http://publications.lib.chalmers.se/records/fulltext/179211/179211.pdf.
[25] G. Csépány, Kinetic Simulation of Runaway Electrons in Tokamaks, Master's Thesis, Budapest University of Technology and Economics (2012), http://deep.reak.bme.hu/~cheoppy/diploma-thesis-2012.pdf.
[26] H. Smith, P. Helander, L.-G. Eriksson, T. Fülöp, Phys. Plasmas 12 (2005) 122505.
[27] G. Bekefi, Radiation Processes in Plasmas, Wiley, 1966.
[28] K.H. Finken, J.G. Watkins, D. Rusbuldt, W.J. Corbett, K.H. Dippel, D.M. Goebel, R.A. Moyer, Nucl. Fusion 30 (1990) 859.
[29] E.M. Hollmann, M.E. Austin, J.A. Boedo, N.H. Brooks, N. Commaux, N.W. Eidietis, D.A. Humphreys, V.A. Izzo, A.N. James, T.C. Jernigan, A. Loarte, J. Martin-Solis, R.A. Moyer, J.M. Munoz-Burgos, D.L. Rudakov, E.J. Strait, C. Tsui, M.A.V. Zeeland, J.C. Wesley, J.H. Yu, Nucl. Fusion 53 (2013) 083004.
[30] Stahl A., Landreman M., G. Papp, E. Hollmann, T. Fülüp, Phys. Plasmas 20 (2013) 093302.
[31] H. Smith, P. Helander, L.-G. Eriksson, D. Anderson, M. Lisak, F. Andersson, Phys. Plasmas 13 (2006) 102502.
[32] K. Gál, T. Fehér, H. Smith, T. Fülöp, P. Helander, Plasma Phys. Controll. Fusion 50 (2008) 055006.
[33] T. Fehér, H.M. Smith, T. Fülöp, K. Gál, Plasma Phys. Controll. Fusion 53 (2011) 035014.

Errata

- In Eq. (11), the partial derivative with respect to y in the third term should be replaced by a partial derivative with respect to ξ, so that the first row reads

$$\frac{\partial F}{\partial \hat{t}} + \hat{E}\,\xi\frac{\partial F}{\partial y} + \hat{E}\frac{1-\xi^2}{y}\frac{\partial F}{\partial \xi}$$

- On the left-hand side of Eq. (17), the collision frequency should be removed, so that the expression starts with

$$\frac{1}{n}\frac{\mathrm{d}n_r}{\mathrm{d}\hat{t}} = \dots$$

- In Fig. 7, a grid resolution of $N_\mathrm{y} = 200$ was used, not $N_\mathrm{y} = 20$ as indicated in the caption

IOP Publishing | International Atomic Energy Agency Nuclear Fusion

Nucl. Fusion **56** (2016) 112009 (10pp) doi:10.1088/0029-5515/56/11/112009

Kinetic modelling of runaway electrons in dynamic scenarios

A. Stahl[1], **O. Embréus**[1], **G. Papp**[2], **M. Landreman**[3] **and T. Fülöp**[1]

[1] Department of Physics, Chalmers University of Technology, Göteborg, Sweden
[2] Max Planck Institute for Plasma Physics, Garching, Germany
[3] University of Maryland, College Park, MD, USA

E-mail: stahla@chalmers.se

Received 5 January 2016, revised 4 March 2016
Accepted for publication 29 March 2016
Published 22 July 2016

Abstract

Improved understanding of runaway-electron formation and decay processes are of prime interest for the safe operation of large tokamaks, and the dynamics of the runaway electrons during dynamical scenarios such as disruptions are of particular concern. In this paper, we present kinetic modelling of scenarios with time-dependent plasma parameters; in particular, we investigate hot-tail runaway generation during a rapid drop in plasma temperature. With the goal of studying runaway-electron generation with a self-consistent electric-field evolution, we also discuss the implementation of a collision operator that conserves momentum and energy and demonstrate its properties. An operator for avalanche runaway-electron generation, which takes the energy dependence of the scattering cross section and the runaway distribution into account, is investigated. We show that the simplified avalanche model of Rosenbluth and Putvinskii (1997 *Nucl. Fusion* **37** 1355) can give inaccurate results for the avalanche growth rate (either lower or higher) for many parameters, especially when the average runaway energy is modest, such as during the initial phase of the avalanche multiplication. The developments presented pave the way for improved modelling of runaway-electron dynamics during disruptions or other dynamic events.

Keywords: runaway electrons, Fokker–Planck equation, avalanche generation, hot-tail generation, linearized collision operator

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

1. Introduction

Runaway electrons, a phenomenon made possible by the decrease of the collisional friction with particle energy [1, 2], are common in plasmas in the presence of strong external electric fields or changing currents. The tightly focused beam of highly relativistic particles can be a serious threat to the first wall of a fusion reactor, due to the possibility of localized melting or halo-current generation [3]. In the quest for avoidance or mitigation of the harmful effects of runaway-electron losses, a greater understanding of the runaway-electron phenomenon is required [4]. Improved knowledge of runaway-electron formation mechanisms, dynamics and characteristics will benefit the fusion community and contribute to a stable and reliable operation of reactor-scale tokamaks.

Kinetic simulation is the most accurate and useful method for investigating runaway-electron dynamics, and we recently developed a new tool called CODE (collisional distribution of electrons [5]) for fast and detailed study of these processes. CODE solves the spatially homogeneous kinetic equation for electrons in 2D momentum space, including electric-field acceleration, collisions, avalanche runaway generation and synchrotron-radiation-reaction losses [5–7]. In CODE, momentum space is discretized using finite differences in momentum and a Legendre-mode decomposition in pitch-angle cosine. Often, the time evolution of the distribution is the desired output, but a (quasi-)steady-state solution can also be efficiently obtained through the inversion of a single sparse system (in the absence of an avalanche source). CODE has been used to study the spectrum of the synchrotron radiation emitted by runaways [5], the corresponding influence of the emission on the distribution function [6–8], and the factors influencing the critical electric field for runaway-electron generation [6, 9].

In this paper we describe improvements to CODE which enable us to investigate the effect of hot-tail runaway generation on the distribution (section 2). This process can be the dominant mechanism in rapidly cooling plasmas. We also discuss the implementation of a full linearized collision operator, and demonstrate its conservation properties (section 3). The use of this operator is necessary in cases where the correct plasma conductivity is required, and our implementation indeed reproduces the Spitzer conductivity [10] for weak electric fields. In addition, an improved model for the large-angle (knock-on) Coulomb collisions leading to avalanche multiplication of the runaway population [11], is described in section 4. This model takes the energy dependence of the runaway distribution into account, and uses the complete energy-dependent Møller scattering cross section [12]. We find that its use can in some cases lead to significant modifications to the avalanche growth rate, compared to the more simplified model of Rosenbluth and Putvinskii [13].

The improvements described in this work enable the detailed study of runaway processes in dynamic situations such as disruptions, and the conservative collision operator makes self-consistent calculations of the runaway population and current evolution in such scenarios feasible [14].

2. Time-dependent plasma parameters

To be able to investigate the behavior of the electron population in dynamic scenarios such as disruptions or sawtooth crashes, it is necessary to follow the distribution function as the plasma parameters change. To this end, CODE has been modified to handle time-dependent background-plasma parameters. Since the kinetic equation is treated in linearized form, the actual temperature and density of the distribution are determined by the background Maxwellian used in the formulation of the collision operator. This allows for a scheme where the kinetic equation is normalized to a *reference* temperature $\tilde{T}$ and number density $\tilde{n}$, so that the discretized equation can be expressed on a fixed *reference grid* in momentum space (throughout this paper, we will use a tilde to denote a reference quantity). By changing the properties of the Maxwellian equilibrium around which the collision operator is linearized, the evolution of the plasma parameters can be modelled on the reference grid without the need for repeated interpolation of the distribution function to new grids.

Analogously to [5], the kinetic equation in 2D momentum space for the electron distribution function f experiencing an electric field E (parallel to the magnetic field) and collisions, can be expressed as

$$\frac{\partial F}{\partial \hat{t}} + \hat{E}\left(\xi\frac{\partial F}{\partial y} + \frac{1-\xi^2}{y}\frac{\partial F}{\partial \xi}\right) = \hat{C}\{F\} + \hat{S}\{F\}. \quad (1)$$

Here we have introduced a convenient normalized momentum $y = \gamma v/\tilde{v}_e$, where $\tilde{v}_e = \sqrt{2\tilde{T}/m}$ is the reference electron thermal speed, and the cosine of the pitch angle $\xi = y_{\parallel}/y$. Using $\kappa = m^3\tilde{v}_e{}^3\pi^{3/2}/\tilde{n}$, we have also defined the distribution function $F = F(y,\xi) = \kappa f$ (normalized so that $F(y=0) = 1$ for a Maxwellian with $T = \tilde{T}$ and $n = \tilde{n}$), time $\hat{t} = \tilde{\nu}_{ee}t$, and electric field $\hat{E} = -eE/m\tilde{v}_e\tilde{\nu}_{ee}$, as well as the normalized operators $\hat{C} = C\,\kappa/\tilde{\nu}_{ee}$ and $\hat{S} = S\kappa/\tilde{\nu}_{ee}$, with $\tilde{\nu}_{ee} = 16\sqrt{\pi}\,e^4\tilde{n}\ln\tilde{\Lambda}/3m^2\tilde{v}_e{}^3$ the reference electron thermal collision frequency, $-e$, m and v the charge, rest mass and speed of the electron, and γ the relativistic mass factor. Note that $|\hat{E}| = (3\sqrt{\pi}/2)E/E_D$, with E_D the Dreicer field [1]. C is the Fokker–Planck collision operator and S an operator describing close (large-angle) Coulomb collisions. These operators will be discussed more thoroughly in sections 3 and 4, respectively; for now we just state the formulation of the collision operator employed in [5] using the normalizations above:

$$\hat{C}^{\rm tp} = c_C\bar{v}_e{}^3y^{-2}\left(\frac{\partial}{\partial y}\left[y^2\Psi\left(\frac{1}{x}\frac{\partial}{\partial y} + \frac{2}{\bar{v}_e{}^2}\right)F\right]\right.$$
$$\left. + \frac{c_\xi}{2x}\frac{\partial}{\partial \xi}(1-\xi^2)\frac{\partial F}{\partial \xi}\right). \quad (2)$$

The superscript tp denotes that this is the test-particle part of the linearized collision operator C^l discussed in section 3. Here (and throughout the rest of this paper), a bar denotes a quantity normalized to its reference value (i.e. $\bar{v}_e = v_e/\tilde{v}_e$), $x = y/\gamma = v/\tilde{v}_e$ is the normalized speed, $c_C = 3\sqrt{\pi}\,\bar{\nu}_{ee}/4$, $c_\xi = Z_{\rm eff} + \Phi - \Psi + \bar{v}_e{}^2\delta^4x^2/2$, $Z_{\rm eff}$ is the effective ion charge, $\Phi = \Phi(x/\bar{v}_e)$ and $\Psi = \Psi(x/\bar{v}_e) = \bar{v}_e{}^2[\Phi - \bar{v}_e{}^{-1}x{\rm d}\Phi/{\rm d}(x/\bar{v}_e)]/2x^2$ are the error and Chandrasekhar functions, respectively, and $\delta = \tilde{v}_e/c$ (with c the speed of light) is assumed to be a small parameter (i.e. the thermal population is assumed to be non-relativistic).

Changes to the plasma temperature manifest as shifts in the relative magnitude of the various terms in equation (2) (through δ and the quantities with a bar), as well as a change in the overall magnitude of the operator, whereas changes in density only have the latter effect. In both cases, the distribution is effectively colliding with (and relaxing towards) a Maxwellian different from the one native to the reference momentum grid. Heat or particles are introduced to (or removed from) the bulk of the distribution when using this scheme, as all changes to plasma parameters are described by changes to the Maxwellian. This provides a powerful way of simulating rapid cooling, for instance associated with a tokamak disruption.

2.1. Hot-tail runaway-electron generation

If the time scale of the initial thermal quench in a disruption event is short enough—comparable to the collision time—the tail of the initial Maxwellian electron distribution will not have time to equilibrate as the plasma cools. The particles in this supra-thermal tail may constitute a powerful source of runaway electrons, should a sufficiently strong electric field develop before they have time to reconnect with the bulk electrons. This process is known as *hot-tail generation*, and can be the dominant source of runaways

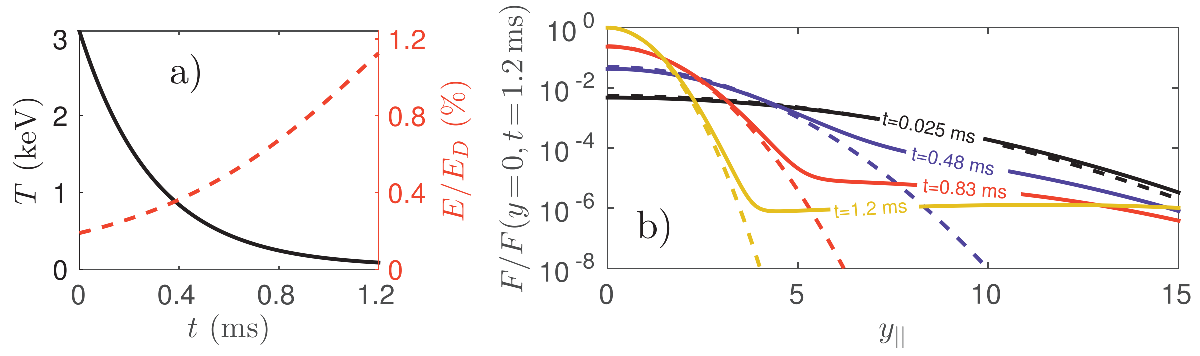

Figure 1. (*a*) Temperature and electric-field evolution in equations (3) and (4). (*b*) Parallel ($\xi = 1$) electron distributions (solid) and corresponding Maxwellians (dashed) at several times during the temperature drop in (*a*). A momentum grid with a fixed reference temperature $\tilde{T} = 100$ eV was used and the distributions are normalized to $F(y = 0)$ in the final time step to facilitate a comparison.

under certain conditions [15, 16]. It has previously been investigated analytically or using Monte-Carlo simulations [16, 17] or purpose-built finite-difference tools [17, 18]. Using CODE to model a temperature drop enables the efficient study of a wider range of scenarios, and allows full use of other capabilities of CODE, such as avalanche generation or synchrotron radiation reaction. Here, we restrict ourselves to a proof-of-principle demonstration, and leave a more extensive investigation to future work.

To facilitate a comparison to the theoretical work by Smith and Verwichte [18], we will model a rapid exponential temperature drop, described by

$$T(t) = T_{\rm f} + (T_0 - T_{\rm f})\mathrm{e}^{-t/t_\star}, \tag{3}$$

with $T_0 = 3.1$ keV the initial temperature, $T_{\rm f} = 31$ eV the final temperature, and $t_\star = 0.3$ ms the cooling time scale. We also include a time-dependent electric field described by

$$\frac{E(t)}{E_{\rm D}} = \left(\frac{E}{E_{\rm D}}\right)_0 \sqrt{\frac{T_0}{T(t)}}, \tag{4}$$

with $(E/E_{\rm D})_0 = 1/530$ the initial normalized electric field. The temperature and electric-field evolutions are shown in figure 1(*a*) and are the same as those used in figure 5 of [18], as are all other parameters in this section.

Figure 1(*b*), in which the additional parameters $n = 2.8 \cdot 10^{19}$ m^{-3}, and $Z_{\rm eff} = 1$ were used, illustrates the distribution-function evolution during the temperature drop. The figure shows that as the temperature decreases, most of the electrons quickly adapt. At any given time t, the bulk of the distribution remains close to a Maxwellian corresponding to the current temperature $T(t)$. The initially slightly more energetic electrons, although part of the original bulk population, thermalize less efficiently. On the short cooling time-scale, they remain as a distinct tail, and as the thermal speed decreases they become progressively less collisional. This process is evident in the first three time steps shown ($t = 0.025$–0.83 ms). In the final time step, the electric field has become strong enough to start to affect the distribution, and a substantial part of the high-energy tail is now in the runaway region. This can be seen from the qualitative change in the tail of the distribution, which now shows a positive slope associated with a strong flow of particles to higher momenta.

For the temperature evolution in equation (3), analytical results for the hot-tail runaway generation were obtained in [18]. Assuming the background density to be constant, the runaway fraction at time t can be written as

$$\frac{n_{\rm r,dir}}{n} = \frac{4}{\sqrt{\pi}} \int_{u_{\rm c}}^{\infty} \left[1 - \frac{(u_{\rm c}^3 - 3\tau)^{2/3}}{(u^3 - 3\tau)^{2/3}}\right] \mathrm{e}^{-u^2} u^2 \mathrm{d}u, \tag{5}$$

where $\tau(t) = (3\sqrt{\pi}/4)\nu_{\rm ee}(t - t_\star) = (3\sqrt{\pi}/4)(\hat{t} - \hat{t}_\star)$ is a normalized time, $u(t) = x^{[0]} + 3\tau(t)$, $x^{[0]}$ is the speed normalized to the initial thermal speed, and $u_{\rm c}$ is related to the critical speed for runaway generation: $u_{\rm c}(t) = x_{\rm c}^{[0]} + 3\tau(t)$. Equation (5), which corresponds to equation (18) in [18], is only valid when a significant temperature drop has already taken place (as manifested by the appearance of the cooling time scale $t_\star$ as a 'delay' in the expression for τ, see [18]). Equation (5) is derived in the absence of an electric field; only an exponential drop in the bulk temperature is assumed. The electric field shown in figure 1(*a*) is only used to define a runaway region, so that the runaway fraction can be calculated. In other words, it is assumed that the electric field does not have time to influence the distribution significantly during the temperature drop.

The runaway fraction calculated using equation (5) includes only the electrons in the actual runaway region, i.e. particles whose trajectories (neglecting collisional momentum-space diffusion) are not confined to a region close to the origin. In this case, the lower boundary of the runaway region is given in terms of the limiting (non-relativistic) momentum y for a given ξ: $y_{{\rm c}\xi} = (\delta^2[(\xi + 1)E/2E_{\rm c} - 1])^{-1/2}$ [17], where $E_{\rm c} = 4\pi e^3 n \ln \Lambda / mc^2$ is the critical electric field for runaway generation [19]. The temperature drop does however lead to an isotropic high-energy tail (in the absence of an electric field). By defining the runaway region as $y > y_{\rm c} = (\delta^2[E/E_{\rm c} - 1])^{-1/2}$, thereby including all particles with $v > v_{\rm c}$, equation (5) can be simplified to

$$\frac{n_{\rm r}}{n} = \frac{2}{\sqrt{\pi}} u_{\rm c} \mathrm{e}^{-u_{\rm c}^2} + \mathrm{erfc}(u_{\rm c}), \tag{6}$$

where erfc(x) is the complementary error function. By default, CODE uses such an isotropic runaway region, which is a good

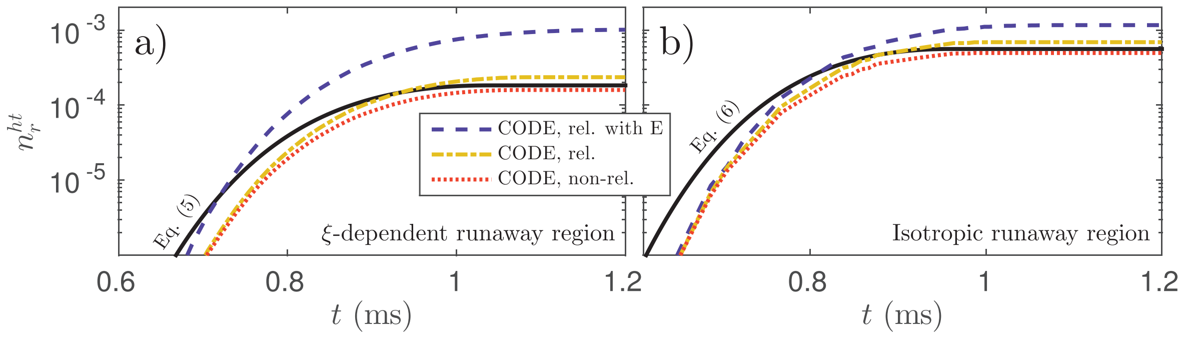

Figure 2. Hot-tail runaway density obtained using CODE—with (blue, dashed) and without (yellow, dash–dotted; red, dotted) an electric field included during the temperature drop—and the analytical estimates equations (5) and (6) (black, solid), for the temperature and E-field evolution in figure 1(*a*). An (*a*) ξ-dependent and (*b*) isotropic lower boundary of the runaway region was used. The collision operator in equation (2) was used for the blue and yellow lines, whereas its non-relativistic limit was used for the red and black lines.

approximation in the case of only Dreicer and avalanche generation (especially once the runaway tail has become substantial); however, in the early stages of hot-tail-dominated scenarios, the isotropic runaway region significantly overestimates the actual runaway fraction, and the lower boundary $y_{c\xi}$ must be used.

Figure 2 compares the runaway density evolution computed with CODE, using both ξ-dependent and isotropic runaway regions, to equations (5) and (6), respectively. The parameters of the hot-tail scenario shown in figure 1 were used, and no avalanche source was included in the calculation. The collision operator used in [18] is the non-relativistic limit of equation (2), with $c_\xi = 0$ (since the distribution is isotropic in the absence of an electric field). CODE results using both this operator (red, dotted) and the full equation (2) (yellow, dash–dotted) are plotted in figure 2, with the latter producing $\sim 50\%$ more runaways in total. This difference can likely be explained by the relatively high initial temperature (3 keV) in the scenario considered, in which case the non-relativistic operator is not strictly valid for the highest-energy particles. Good agreement between CODE results and equations (5) and (6) (black, solid) is seen for the saturated values in the figure. A CODE calculation where the electric-field evolution is properly included in the kinetic equation (corresponding to the distribution evolution in figure 1(*b*)) is also included (blue, dashed), showing increased runaway production. With the isotropic runaway region (figure 2(*b*)), the increase is smaller than a factor of 2, and neglecting the influence of the electric field can thus be considered reasonable for the parameters used, at least for the purpose of gaining qualitative understanding. With the ξ-dependent runaway region (figure 2(*a*)), the change in runaway generation is more pronounced, and the inclusion of the electric field leads to an increase by almost an order of magnitude. Note that the final runaway density with the electric field included is very similar in figures 2(*a*) and (*b*), indicating that the details of the lower boundary of the runaway region become unimportant once the tail is sufficiently large. Throughout the remainder of this paper we will make use of the isotropic runaway region.

We conclude that, in order to obtain quantitatively accurate results, the electric field should be properly included, and a relativistic collision operator should be used. This is especially true when modelling ITER scenarios, where the initial temperature can be significantly higher than the 3 keV used here.

3. Conservative linearized Fokker–Planck collision operator

Treating the runaway electrons as a small perturbation to a Maxwellian distribution function, the Fokker–Planck operator for electron–electron collisions [20, 21] can be linearized and written as $C\{f\} \simeq C^l\{f\} = C^{\rm tp} + C^{\rm fp}$. The so-called *test-particle term*, $C^{\rm tp} = C^{\rm nl}\{f_1, f_{\rm M}\}$, describes the perturbation colliding with the bulk of the plasma, whereas the *field-particle term*, $C^{\rm fp} = C^{\rm nl}\{f_{\rm M}, f_1\}$, describes the reaction of the bulk to the perturbation. Here $C^{\rm nl}$ is the non-linear Fokker–Planck–Landau operator, $f_{\rm M}$ denotes a Maxwellian, and $f_1 = f - f_{\rm M}$ the perturbation to it ($f_1 \ll f_{\rm M}$). Collisions described by $C\{f_1, f_1\}$ are neglected since they are second order in f_1. The full linearized operator C^l conserves particles, momentum and energy. Since it is proportional to a factor $\exp(-y^2)$, the field-particle term mainly affects the bulk of the plasma, and is therefore commonly neglected when studying runaway-electron kinetics. The test-particle term in equation (2) only ensures the conservation of particles, however, not momentum or energy.

Under certain circumstances, it is necessary to use a fully conservative treatment also for the runaway problem, in particular when considering processes where the conductivity of the plasma is important. In the study of runaway dynamics during a tokamak disruption using a self-consistent treatment of the electrical field, accurate plasma-current evolution is essential, and the full linearized collision operator must be used. A non-linear collision operator valid for arbitrary particle (and bulk) energy has been formulated [22, 23]. The collision operator originally implemented in CODE is the result of an asymptotic matching between the highly relativistic limit of the test-particle term of the linearized version of that operator, with the usual non-relativistic test-particle operator [24], and is given in equation (2). The relativistic field-particle term is significantly more complicated, however, and its use would be computationally more expensive. Here we instead

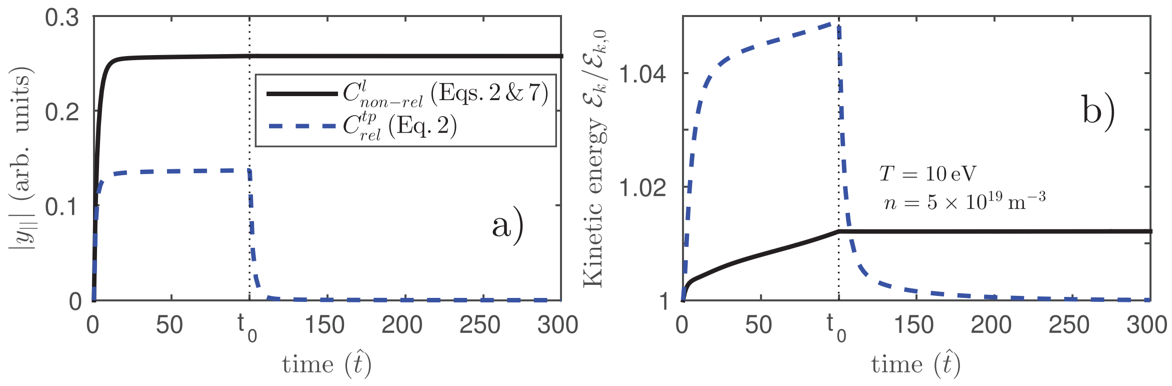

Figure 3. (*a*) Parallel momentum and (*b*) energy moments of the distribution function in CODE, using different collision operators. Initially, $E = 50\,\mathrm{V\,m^{-1}}$ and $Z_{\rm eff} = 1$ were used, but for $t > t_0$, the electric field was turned off and the ion charge set to $Z_{\rm eff} = 0$. Using two Legendre modes for the field-particle term was sufficient to achieve good conservation of energy and parallel momentum.

implemented the non-relativistic field-particle term, as formulated in [25, 26]. As will be shown, this operator (together with the non-relativistic limit of equation (2)) accurately reproduces the Spitzer conductivity for sufficiently weak electric fields and temperatures where the bulk is non-relativistic. Using the normalization in section 2, the field-particle term is

$$\hat{C}^{\rm fp} = \frac{c_C}{\pi^{3/2}} {\rm e}^{-\tilde{v}_{\rm e}^{-2}x^2} \left[\frac{2x^2}{\tilde{v}_{\rm e}{}^4} \frac{\partial^2 G}{\partial x^2} - \frac{2}{\tilde{v}_{\rm e}{}^2} H + 4\pi F \right], \tag{7}$$

where G and H are the Rosenbluth potentials, obtained from the distribution using

$$\tilde{v}_{\rm e}{}^2 \nabla^2_{\mathbf{v}} H = -4\pi F, \qquad \tilde{v}_{\rm e}{}^2 \nabla^2_{\mathbf{v}} G = 2H. \tag{8}$$

The system of equations composed of equations (7) and (8), together with the non-relativistic limits of equations (1) and (2) ($y \to x$ and $\delta \to 0$), is discretized (see [5]) and solved using an efficient method described in [27]. The equations are combined into one linear system of the form

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & 0 \\ 0 & M_{31} & M_{33} \end{pmatrix} \begin{pmatrix} F \\ G \\ H \end{pmatrix} = \begin{pmatrix} S_i \\ 0 \\ 0 \end{pmatrix}, \tag{9}$$

where the first row describes the kinetic equation (1) (with S_i representing any sinks or sources), and the second and third rows correspond to equation (8). This approach makes it possible to consistently solve for both the Rosenbluth potentials and the distribution with a single matrix operation. Since there is no explicit need for the Rosenbluth potentials, however, G and H can be eliminated by solving the block system analytically:

$$(M_{11} - [M_{12} - M_{13}M_{33}^{-1}M_{32}]\, M_{22}^{-1}M_{21})F \equiv MF = S_i. \tag{10}$$

If only the test-particle operator (equation (2)) is used, M reduces to M_{11}. Since the Rosenbluth potentials are defined through integrals of the distribution, the field-particle term introduces a full block for each Legendre mode into the normally sparse matrix describing the system. However, the integral dependence on F also implies that significantly fewer modes are required to accurately describe the potentials (compared to F), and the additional computational cost is modest (the operator $\nabla^2_{\mathbf{v}}$ is proportional to l^2, with l the Legendre mode index, and G and H therefore decay rapidly with increasing l).

The conservation properties of the full non-relativistic collision operator (equations (2) and (7)), as well as the relativistic test-particle operator in equation (2), are shown in figure 3. As an electric field is applied to supply some momentum and energy to the distribution, the parallel momentum (figure 3(*a*)) quickly reaches a steady-state value corresponding to the plasma conductivity, which differs by about a factor of two for the two operators (see below). The electric field is turned off at $t = t_0 = 100$ collision times (and $Z_{\rm eff} = 0$ is imposed to isolate the behavior of the electron–electron collision operator), at which point the parallel momentum for the operator in equation (2) (blue, dashed) is lost on a short time scale as the distribution relaxes back towards a Maxwellian. In contrast, the full linearized operator (black, solid) conserves parallel momentum in a pure electron plasma, as expected.

The electric field continuously does work on the distribution, a large part of which heats the bulk electron population, but the linearization of the collision operator breaks down if the distribution deviates too far from the equilibrium solution. As long as a non-vanishing electric field is used together with an energy conserving collision operator, an adaptive sink term removing excess heat from the bulk of the distribution must be included in equation (1) to guarantee a stable solution. Physically this accounts for loss processes that are not included in the model, such as line radiation, bremsstrahlung and radial heat transport. The magnitude of the black line in figure 3(*b*) therefore reflects the energy content of the runaway population—not the total energy supplied by the electric field—since a constant bulk energy is enforced. The energy sink is not included for $t > t_0$ (since $E = 0$), however, and the energy conservation observed is due to the properties of the collision operator itself. Again, the use of the collision operator in equation (2) is associated with a quick loss of kinetic energy as soon as the electric field is removed.

The electrical conductivity of a fully ionized plasma subject to an electric field well below the Dreicer value—the *Spitzer conductivity*—can be expressed as

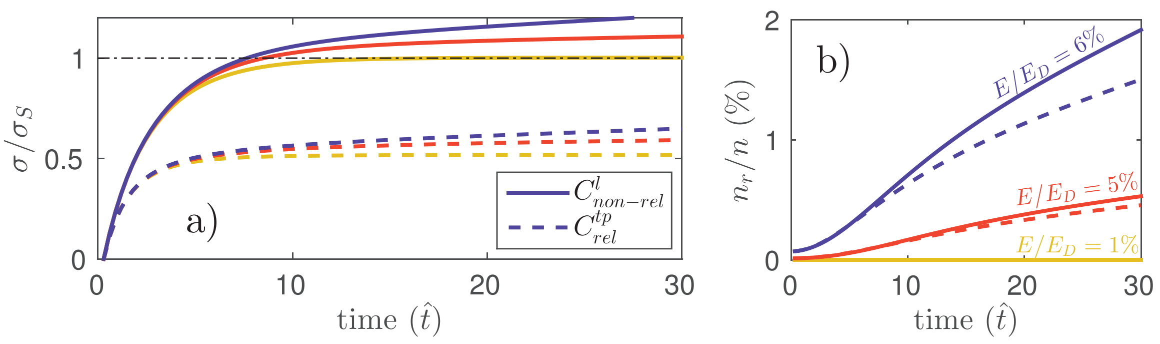

Figure 4. (*a*) Conductivity (normalized to the Spitzer value) and (*b*) normalized runaway density, as functions of time for different collision operators (non-relativistic full linearized: solid; relativistic test-particle: dashed) and E-field strengths ($E/E_D = 1\%$: yellow; $E/E_D = 5\%$: red; $E/E_D = 6\%$: blue), considering only Dreicer runaway generation. The parameters $T = 1$ keV, $n = 5 \cdot 10^{19}$ m^{-3} and $Z_{\text{eff}} = 1$ were used.

$$\sigma_S = L(Z_{\text{eff}}) \frac{ne^2}{Z_{\text{eff}}\, m\nu_{\text{ee}}}, \tag{11}$$

where $L(Z_{\text{eff}})$ is a transport coefficient which takes the value $L \simeq 2$ in a pure hydrogen plasma [10]. Figure 4 demonstrates that the conductivity calculated with CODE reproduces the Spitzer value for moderate electric-field strengths, if the conservative collision operator is used, and the initial Maxwellian adapts to the applied electric field on a time scale of roughly 10 collision times. For field strengths significantly larger than E_c, the conductivity starts to deviate from σ_S, as a runaway tail begins to form (figure 4(*b*)); in this regime, the calculation in [10] is no longer valid. Using the collision operator in equation (2) consistently leads to a conductivity which is lower by about a factor of 2, as expected (see for instance [28]). The runaway growth is also affected, with the conserving operator leading to a larger runaway growth rate.

4. Improved operator for knock-on collision

The Fokker–Planck collision operators discussed in section 3 accurately describe grazing collisions—small-angle deflections which make up the absolute majority of particle interactions in the plasmas under consideration. Large-angle collisions are usually neglected as their cross section is significantly smaller, but in the presence of runaway electrons they can play an important role in the momentum space dynamics, as an existing runaway can transfer enough momentum to a thermal electron in one collision to render it a runaway, while still remaining in the runaway region itself. Such *knock-on* collisions can therefore lead to an exponential growth of the runaway density—an *avalanche* [13, 29].

In the absence of a complete solution to the Boltzmann equation, we model avalanche runaway generation using an additional source term in the kinetic equation (1), evaluated for $y > y_c$. A commonly used operator was derived by Rosenbluth and Putvinski [13] and takes the form

$$\hat{S}_{\text{RP}} = \frac{n_r}{n}\bar{n}^2 \left[\frac{3\pi\delta^3}{16\ln\tilde{\Lambda}} \delta_D(\xi - \xi_2) \frac{1}{y^2}\frac{\partial}{\partial y}\left(\frac{1}{1-\sqrt{1+\delta^2 y^2}} \right) \right], \tag{12}$$

where n_r is the number density of runaway electrons, $\bar{n}$ is the density normalized to its reference value, and δ_D is the Dirac δ-function. In the derivation, the momentum of the incoming particle is assumed to be very large (simplifying the scattering cross section) and its pitch-angle vanishing ($\xi = 1$). It is also assumed that the incoming particle is unaffected by the interaction. These conditions imply that the generated *secondary* particles are all created on the curve $\xi = \xi_2 = \delta y/(1 + \sqrt{1+\delta^2 y^2})$ (which is a parabola in $[y_\|, y_\perp]$-space), and that *all* runaways (from the point of view of the avalanche source) are assumed to have momentum $p = \gamma v/c = \delta y \gg 1$ (since $\hat{S}_{\text{RP}} \propto n_r$). They can therefore contribute equally strongly to the avalanche process. This has the peculiar and non-physical consequence that particles can be created with an energy higher than that of any of the existing runaways. The δ-function in ξ is numerically ill-behaved, as it produces significant oscillations (Gibbs phenomenon) when discretized using the Legendre-mode decomposition employed in CODE (see figure 5(*a*)).

An operator that relaxes the assumption of very large runaway momentum has been presented by Chiu *et al* [11]. It has the form

$$\hat{S}_{\text{Ch}}(y,\xi) = \bar{n}\frac{2\pi e^4}{m^2c^3}\frac{\bar{n}\delta^3}{\tilde{\nu}_{\text{ee}}}\frac{x}{y^2\xi}(y_{\text{in}})^4 F^\star(y_{\text{in}})\,\Sigma(\gamma,\gamma_{\text{in}}), \tag{13}$$

where

$$\Sigma(\gamma,\gamma_{\text{in}}) = \frac{{\gamma_{\text{in}}}^2}{({\gamma_{\text{in}}}^2-1)(\gamma-1)^2(\gamma_{\text{in}}-\gamma)^2}\Bigg[(\gamma_{\text{in}}-1)^2 - \frac{(\gamma-1)(\gamma_{\text{in}}-\gamma)}{{\gamma_{\text{in}}}^2}\left(2{\gamma_{\text{in}}}^2 + 2\gamma_{\text{in}} - 1 - (\gamma-1)(\gamma_{\text{in}}-\gamma)\right)\Bigg] \tag{14}$$

is the Møller scattering cross section [12] and $F^\star$ is the pitch-angle-averaged distribution of incoming runaways with properties y_{in} and γ_{in}. All incoming particles are thus still assumed to have zero pitch angle ($\xi = 1$), but their energy distribution is properly taken into account. In CODE, $F^\star$ is computed from the 0th Legendre mode of F; $F^\star = 2F_0$.

From the conservation of 4-momentum in a collision, the momentum-space coordinates are related through

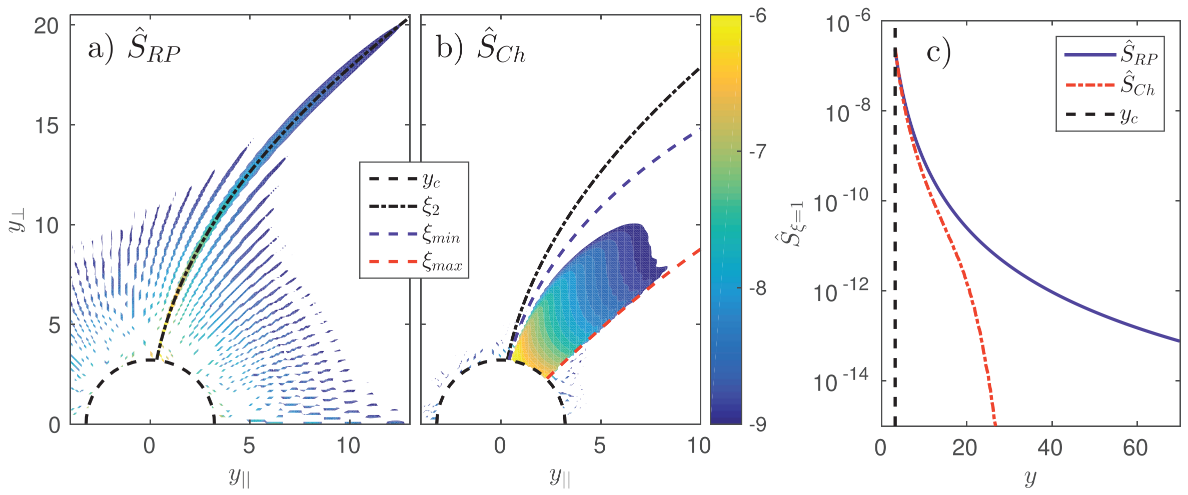

Figure 5. Contour plots of the magnitude of the source in (*a*) equation (12) and (*b*) equation (13) in ($y_{\|}$,$y_\perp$) momentum space, given the same electron distribution. The plotted quantity is $\log_{10}\hat{S}$ and y_c defines the lower bound of the runaway region. The angle-averaged source magnitudes are shown in (*c*). The parameters $T = 1$ keV, $n = 5\cdot 10^{19}\ \mathrm{m}^{-3}$, $Z_{\mathrm{eff}} = 1$ and $E = 1\ \mathrm{V\ m}^{-1}$, with $\max(y) = 70$, were used to obtain the distribution, and the simulation was run for 300 collision times with primary generation only.

$$\xi = \sqrt{\frac{(\gamma-1)(\gamma_{\mathrm{in}}+1)}{(\gamma+1)(\gamma_{\mathrm{in}}-1)}}, \tag{15}$$

which restricts the region where the source is non-vanishing (this relation is analogous to the parabola ξ_2 in the case of the operator in equation (12)). Since the electrons participating in a collision are indistinguishable, it is sufficient to consider only the cases where the energy of the created secondary runaway is less than half of the primary energy, $(\gamma-1)\leqslant(\gamma_{\mathrm{in}}-1)/2$, which with the above equation leads to the condition $\xi\leqslant\xi_{\max}=\sqrt{\gamma/(\gamma+1)}$. By the same argument, the maximum attainable runaway energy in the simulation (the maximum of the momentum grid) leads to the condition $\xi\geqslant\xi_{\min}=\sqrt{(\gamma-1)(\gamma_{\max}+1)/(\gamma+1)(\gamma_{\max}-1)}$.

The magnitudes of the two sources (12) and (13) are computed from a given typical runaway distribution function, and shown in figures 5(*a*) and (*b*). Curves corresponding to the parabola ξ_2, as well as the limits $\xi_{\min}$ and $\xi_{\max}$ are also included. Note that the amount of numerical noise is significantly reduced for the source in equation (13). In order to avoid double-counting the small-angle collisions described by the Fokker–Planck–Landau collision operator C, the knock-on source must be cut off at some value of momentum sufficiently far from the thermal bulk. As can be seen from the figure, however, the magnitude of both sources increases with decreasing momenta, and the avalanche growth rate is therefore sensitive to the specific choice of momentum cut-off. Since our particular interest is the generation of runaway electrons, we choose to place the cut-off at $y = y_c$, so that the sources are non-vanishing only in the runaway region [5, 15]. Secondary particles deposited just below the threshold—although not technically runaways—could eventually diffuse into the runaway region, thereby potentially increasing the Dreicer growth rate. In [30], such effects were however shown to be negligible for the operator in equation (12), indicating that the vast majority of particles deposited at $y < y_c$ are slowed down rather than accelerated (as expected). This reduces the sensitivity of the avalanche growth rate to the choice of momentum cut-off (as long as $y_{\mathrm{cut-off}}\leqslant y_c$), and reaffirms our choice $y_{\mathrm{cut-off}} = y_c$.

Figure 5(*c*) shows the source terms integrated over pitch-angle, and as expected, the source in equation (13) extends only up to $y\simeq y_{\max}/2$, whereas the source in equation (12) is non-vanishing also for larger momenta. The amount of secondary runaways generated by the two sources agrees well at low energies, but less so further away from the bulk. In this particular case, the total source magnitude $\int\hat{S}\,y^2\mathrm{d}y\mathrm{d}\xi$ agrees to within 25%, as most of the secondaries are created close to the boundary of the runaway region.

4.1. Avalanche growth rates for the different operators

In general, the avalanche growth rate produced by the two sources can differ substantially. We will illustrate this point by considering the Møller cross section in more detail. We choose to quantify the source magnitude for an arbitrary distribution by computing the cross section, integrated over the energy of the outgoing (secondary) particle and normalized to r_0^2, with r_0 the classical electron radius. In other words, we look at the total normalized cross section for an incoming particle with γ_{in} to participate in a knock-on collision resulting in avalanche [31]:

$$\begin{aligned}K_{\mathrm{Ch}}(\gamma_{\mathrm{in}}) &= \int_{\gamma_c}^{(\gamma_{\mathrm{in}}-1)/2+1}\Sigma(\gamma,\gamma_{\mathrm{in}})\mathrm{d}\gamma\\ &= ({\gamma_{\mathrm{in}}}^2-1)^{-1}\left[\frac{{\gamma_{\mathrm{in}}}^2}{\gamma_c-1}+\frac{{\gamma_{\mathrm{in}}}^2}{\gamma_c-\gamma_{\mathrm{in}}}\right.\\ &\quad\left.+\frac{2\gamma_{\mathrm{in}}-1}{\gamma_{\mathrm{in}}-1}\ln\left(\frac{\gamma_c-1}{\gamma_{\mathrm{in}}-\gamma_c}\right)+\frac{\gamma_{\mathrm{in}}+1}{2}-\gamma_c\right],\end{aligned} \tag{16}$$

where $\gamma_c=\sqrt{(E/E_c)/(E/E_c-1)}$ corresponds to the critical momentum for runaway generation and the upper integration

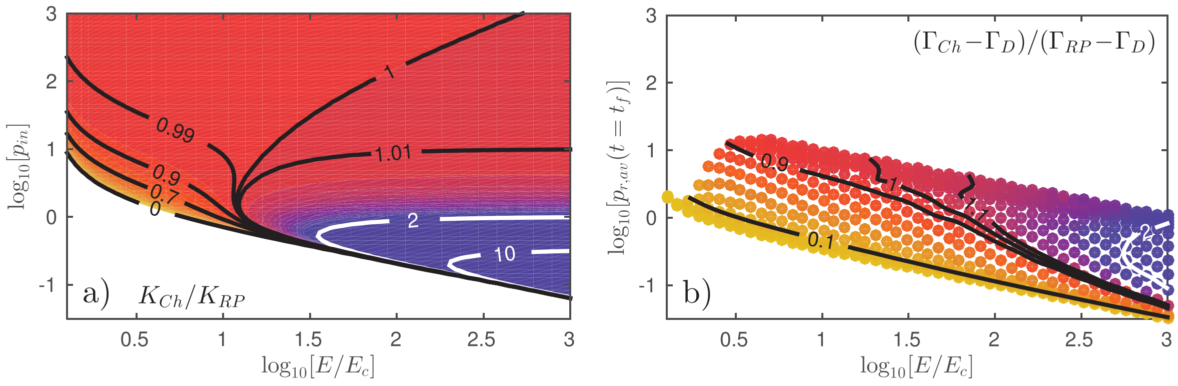

Figure 6. (*a*) Contours (black, white) of the ratio of total cross-sections ($K_{\rm Ch}/K_{\rm RP}$) for an electron with $p_{\rm in}$ to contribute to the avalanche process, as a function of $p_{\rm in}=\gamma_{\rm in}v_{\rm in}/c=\sqrt{\gamma_{\rm in}{}^2-1}$ and $E/E_{\rm c}$. (*b*) Ratio of avalanche growth rates ($[\Gamma_{\rm Ch}-\Gamma_{\rm D}]/[\Gamma_{\rm RP}-\Gamma_{\rm D}]$) in CODE simulations. The parameters $T\in[0.1\,{\rm eV}, 5\,{\rm keV}]$, $E/E_{\rm c}\in[1.1, 1000]$, $n=5\cdot10^{19}\,{\rm m}^{-3}$ and $Z_{\rm eff}=1$ were used.

boundary stems from the condition leading to $\xi_{\rm max}$. This expression is relevant to the source in equation (13), which uses the complete cross section (14), whereas for the more simple source in equation (12), only the leading-order term in $\gamma_{\rm in}$ in the scattering cross section is taken into account. This corresponds to taking the high-energy limit of the above equation, so that

$$K_{\rm RP}=\frac{1}{\gamma_{\rm c}-1} \tag{17}$$

becomes a simple constant.

To systematically explore the relative magnitude of the two sources, the ratio $K_{\rm Ch}/K_{\rm RP}$ is plotted in figure 6(*a*). As expected, the two expressions agree very well at high primary momenta. At somewhat lower momenta, of the order $\gamma\approx p\lesssim5$, two distinct regions are discernible. For $E/E_{\rm c}\lesssim10$ (the orange region), the simplified cross section is larger than the full expression, and the Rosenbluth–Putvinski operator (12) is likely to overestimate the avalanche generation. For $E/E_{\rm c}\gtrsim10$, the opposite is true, and the operator in equation (13) has a significantly larger cross section for $E/E_{\rm c}\gtrsim30$ (the blue region). The more accurate operator (13) should thus be expected to produce more runaways when the runaway population is at predominantly low energies, and $E/E_{\rm c}$ is large. For both of these conditions to be fulfilled simultaneously (and at the same time avoid a slide-away scenario), the temperature must be low so that $E/E_{\rm D}\ll1$ even for large $E/E_{\rm c}$. The effect is also likely to be most apparent at relatively early times, before the runaway tail has extended to multi-MeV energies.

CODE simulations support the above conclusions and show excellent qualitative agreement, as shown in figure 6(*b*). The figure shows the ratio of final avalanche growth rates $(\Gamma_{\rm Ch}-\Gamma_{\rm D})/(\Gamma_{\rm RP}-\Gamma_{\rm D})$, with $\Gamma_i=n_{\rm r}^{-1}({\rm d}n_{\rm r}/{\rm d}\hat{t})$ the growth rate obtained in a CODE run using source i (here the subscript D denotes pure Dreicer generation). Each marker in the figure is thus computed from three separate CODE runs. As a proxy for $p_{\rm in}$, the average runaway momentum $p_{\rm r,av}$ in the final time step $t_{\rm f}$ of the simulation without a source was used, and for a given $E/E_{\rm c}$, different $p_{\rm r,av}$ were obtained from simulations with varying values of T (and corresponding values of $E/E_{\rm D}$). The simulations were run for $t_{\rm max}=5000$ collision times, and $t_{\rm f}$ was set to either $t_{\rm max}$, the first time step for which $n_{\rm r}>5\%$, or the first time step in which the growth rate started to become affected by the proximity of the runaway tail to the end of the simulation grid, whichever occurred first. The parameters of the scan were chosen to focus on the most interesting region of figure 6(*a*)—by performing longer simulations on larger momentum grids, the upper part of the figure could also be studied. Exact agreement between figures 6(*a*) and (*b*) can not be expected, since the source, in addition to the cross section, depends on the details of the runaway distribution. Figure 6(*a*) should thus be viewed as a simplified analytical estimate for figure 6(*b*). The different regions identified in figure 6(*a*) are still apparent in figure 6(*b*), however they are somewhat shifted in parameter space. In particular, the region where the Rosenbluth–Putvinski operator produces a higher growth rate is larger, whereas the opposite region—where the operator in equation (13) dominates—is smaller, or at least shifted to higher values of $E/E_{\rm c}$.

Figure 7 shows all the data points in figure 6(*b*), as a function of temperature. The figure confirms that the region where the more accurate operator produces a significantly higher growth rate is only accessible at temperatures $T<100\,{\rm eV}$ (in the domain of validity of a linearized treatment). As is evident in the figure, however, regions where the Rosenbluth–Putvinski operator significantly overestimates the avalanche growth rate (points below 1 on the vertical axis) are present at all temperatures. The operator in equation (13) is thus of general interest.

Since the electric field spike responsible for the acceleration of runaways during a tokamak disruption is induced by the temperature drop, and therefore occurs slightly later than the drop itself, the temperature is low during the majority of the acceleration process. For significant runaway acceleration, $E/E_{\rm c}\gg1$ is therefore required, and during the initial part of the acceleration process, parameters are likely those corresponding to the blue region of figure 6(*b*), where the improved avalanche source produces a significantly higher growth rate than the Rosenbluth–Putvinski operator. Post-thermal-quench temperatures in ITER are expected be as low as 10 eV and peak electric fields in disruptions can reach 80 V m^{-1} or more [32]. Towards the end of the thermal quench, the

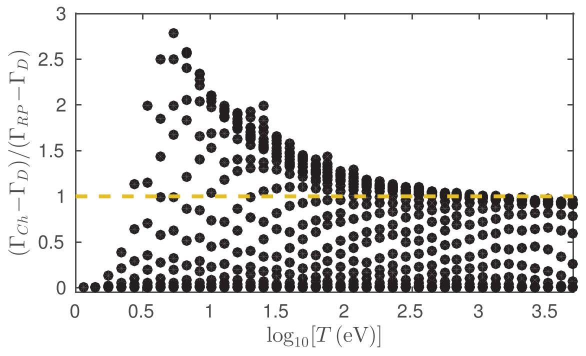

Figure 7. Ratio of avalanche growth rates ($[\Gamma_{\mathrm{Ch}} - \Gamma_{\mathrm{D}}]$ / $[\Gamma_{\mathrm{RP}} - \Gamma_{\mathrm{D}}]$) in CODE simulations, as a function of temperature. The same parameters as in figure 6 were used.

normalized electric field is then $E/E_c \approx 1300$ (with $E = 80\,\mathrm{V\,m^{-1}}$, $T = 50\,\mathrm{eV}$ and $n = 1 \cdot 10^{20}\,\mathrm{m^{-3}}$). A typical ITER disruption would thus (at least initially) be firmly in the blue region of figure 6(*b*), and the avalanche growth should be significantly higher than what the Rosenbluth–Putvinski source predicts. As the temperature is low, the runaways will also spend a comparatively long time at low momenta ($p \ll 1$), where the disagreement between the operators is most pronounced. Note that, according to the figure, an *average* runaway energy of several MeV ($p > 5$–10) is needed for the difference between the growth rates to become small for all E/E_c, at which point the most energetic electrons will have reached energies of several tens of MeV or more. However, since the electric field changes rapidly, the runaways may experience parameters corresponding to both the orange and blue regions in figure 6(*b*) before reaching such energies. Further work is therefore needed to assess the overall impact on the avalanche growth of using the improved operator (13), although it is clear that its use is essential for accurate analysis.

5. Summary

Runaway electrons are intimately linked to dynamic scenarios, as they predominantly occur during disruptions and sawtooth events in tokamaks. An accurate description of their dynamics in such scenarios requires kinetic modelling of rapidly changing plasma conditions, and mechanisms such as hot-tail runaway generation add to the already interesting set of phenomena of importance to the evolution of the runaway population.

In this paper we have described the modelling of several such processes, using the numerical tool CODE to calculate the momentum-space distribution of runaway electrons. In particular, we have investigated rapid-cooling scenarios where hot-tail runaway-electron generation is dominant. Good agreement with previous theoretical work was observed, but CODE simulations also allow for flexible study of a variety of parameter regimes not readily accessible in analytical treatments, and involving other processes such as avalanche generation or synchrotron radiation.

Furthermore, the full linearized non-relativistic Fokker–Planck–Landau collision operator was discussed, and its implementation described. The operator was found to reproduce the expected Spitzer conductivity in the relevant parameter regime and showed excellent conservation properties. The use of such an operator is essential for the correct current evolution in self-consistent modelling, and in particular when studying the interplay between current and electric-field evolution and runaway-electron generation during a disruption.

The process of avalanche multiplication of the runaway population via close Coulomb collisions was also considered, and an improved operator, relaxing some of the approximations of the commonly used Rosenbluth–Putvinski operator, was discussed. It was found that the avalanche growth rate can be significantly affected—increased for low temperatures and high E/E_c and decreased for low E/E_c—by the use of the new operator. The change to the growth rate can be especially large during the early stages of the runaway acceleration process, thus potentially affecting the likelihood of a given runaway seed transforming into a serious runaway beam, and use of the improved operator is of particular relevance in disruption scenarios.

The work presented in this paper paves the way for a better understanding of runaway-electron dynamics in rapidly changing scenarios, for instance during tokamak disruptions. It enables more accurate assessment of the risks posed by runaway electrons in situations of experimental interest, particularly in view of future tokamaks such as ITER.

Acknowledgments

The authors are grateful to I. Pusztai and E. Hirvijoki for fruitful discussions. 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 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. The authors also acknowledge support from Vetenskapsrådet, the Knut and Alice Wallenberg Foundation and the European Research Council (ERC-2014-CoG grant 647121).

References

[1] Dreicer H. 1959 *Phys. Rev.* **115** 238
[2] Dreicer H. 1960 *Phys. Rev.* **117** 329
[3] Hollmann E.M. *et al* 2015 *Phys. Plasmas* **22** 021802
[4] Boozer A.H. 2015 *Phys. Plasmas* **22** 032504
[5] Landreman M., Stahl A. and Fülöp T. 2014 *Comput. Phys. Commun.* **185** 847
[6] Stahl A., Hirvijoki E., Decker J., Embréus O. and Fülöp T. 2015 *Phys. Rev. Lett.* **114** 115002
[7] Hirvijoki E., Pusztai I., Decker J., Embréus O., Stahl A. and Fülöp T. 2015 *J. Plasma Phys.* **81** 475810502
[8] Decker J., Hirvijoki E., Embréus O., Peysson Y., Stahl A., Pusztai I. and Fülöp T. 2015 *Plasma Phys. Control. Fusion* **58** 025016
[9] Paz-Soldan C. *et al* 2014 *Phys. Plasmas* **21** 022514
[10] Spitzer L. and Harm R. 1953 *Phys. Rev.* **89** 977
[11] Chiu S.C., Rosenbluth M.N., Harvey R.W. and Chan V.S. 1998 *Nucl. Fusion* **38** 1711

[12] Møller C. 1932 *Ann. Phys.* **406** 531
[13] Rosenbluth M.N. and Putvinski S.V. 1997 *Nucl. Fusion* **37** 1355
[14] Papp G., Stahl A., Drevlak M., Fülöp T., Lauber Ph.W. and Pokol G.I. 2015 *Eur. Conf. Abstr.* 39E P1.173
[15] Harvey R.W., Chan V.S., Chiu S.C., Evans T.E., Rosenbluth M.N. and Whyte D.G. 2000 *Phys. Plasmas* **7** 4590
[16] Helander P., Smith H., Fülöp T. and Eriksson L.-G. 2004 *Phys. Plasmas* **11** 5704
[17] Smith H., Helander P., Eriksson L.-G. and Fülöp T. 2005 *Phys. Plasmas* **12** 122505
[18] Smith H.M. and Verwichte E. 2008 *Phys. Plasmas* **15** 072502
[19] Connor J.W. and Hastie R.J. 1975 *Nucl. Fusion* **15** 415
[20] Landau L.D. 1936 *Phys. Z. Sowjetunion* **10** 154
[21] Rosenbluth M.N., MacDonald W.M. and Judd D.L. 1957 *Phys. Rev.* **107** 1
[22] Beliaev S.T. and Budker G.I. 1956 *Sov. Phys.—Dokl.* **1** 218
[23] Braams B.J. and Karney C.F.F. 1989 *Phys. Fluids* B **1** 1355
[24] Papp G., Drevlak M., Fülöp T. and Helander P. 2011 *Nucl. Fusion* **51** 043004
[25] Catto P.J. and Tsang K.T. 1977 *Phys. Fluids* **20** 396
[26] Li B. and Ernst D.R. 2011 *Phys. Rev. Lett.* **106** 195002
[27] Landreman M. and Ernst D.R. 2013 *J. Comput. Phys.* **243** 130
[28] Helander P. and Sigmar D. 2002 *Collisional Transport in Magnetized Plasmas* (Cambridge: Cambridge University Press)
[29] Jayakumar R., Fleischmann H.H. and Zweben S. 1993 *Phys. Lett.* A **172** 447
[30] Nilsson E., Decker J., Peysson Y., Granetz R.S., Saint-Laurent F. and Vlainic M. 2015 *Plasma Phys. Control. Fusion* **57** 095006
[31] Aleynikov P. and Breizman B.N. 2015 *Phys. Rev. Lett.* **114** 155001
[32] Papp G., Drevlak M., Fülöp T., Helander P. and Pokol G.I. 2011 *Plasma Phys. Control. Fusion* **53** 095004