Implementation of Common Non-Maxwellian Background Distributions in GENE

Boris Andrews, Salomon Janhunen

Tokamak Energy Ltd

October 2022

1 Background and Motivation

In its base release, the gyrokinetic code GENE has support for only Maxwellian backgrounds

$$f_{s0}[\mathbf{x};\mathbf{v}] = f_{s0}^{\mathrm{M}}[\psi; v_\parallel, \mu] \tag{1}$$

$$= n_s \sqrt{\frac{m_s}{2\pi k_B T_s}}^3 \exp\left[-\frac{1}{k_B T_s}\left(\frac{m_s}{2} v_\parallel^2 + B\mu\right)\right] \tag{2}$$

in species s, where:

- $v_\parallel$ = the parallel velocity
- $\mu = \left(\frac{m_s v_\perp^2}{2B}\right)$ = the magnetic moment

ψ is constant along flux tubes (using the notation as used in [Abe+13]) and the following are functions of ψ only, such that they are constant on flux tubes.

- $n_s(\psi)$ = the particle density
- $T_s(\psi)$ = the temperature

In reality, the space of background distributions for tokamak plasmas is much more rich than just Maxwellians, and different background distributions can have great effects on the behaviour of the model.

1.1 Shifted Maxwellians and Bi-Maxwellians

Examples include:

- Shifted Maxwellians

$$f_{s0}[\mathbf{x};\mathbf{v}] = f_{s0}^{\mathrm{SM}}[\psi; v_\parallel, \mu] \tag{3}$$

$$= n_s \sqrt{\frac{m_s}{2\pi k_B T_s}}^3 \exp\left[-\frac{1}{k_B T_s}\left(\frac{m_s}{2}(v_\parallel - u_{\parallel s})^2 + B\mu\right)\right] \tag{4}$$

  for some parallel velocity $u_{\parallel s}$, allowing for travelling backgrounds. This is particularly relevant for beam injection simulations.

- Bi-Maxwellians

$$f_{s0}[\mathbf{x};\mathbf{v}] = f_{s0}^{\mathrm{BM}}[\psi; v_\parallel, \mu] \tag{5}$$

$$= n_s \sqrt{\frac{m_s}{2\pi k_B T_s}}^3 \exp\left[-\frac{1}{k_B}\left(\frac{m_s}{2T_{\parallel s}} v_\parallel^2 + \frac{B}{T_{\perp s}}\mu\right)\right] \tag{6}$$

  for some parallel and perpendicular temperature fields $T_{\parallel s}$ and $T_{\perp s}$, allowing for backgrounds with differing variations in velocity in the parallel vs. perpendicular direction.

These two modified backgrounds were implementing in an old version of `GENE` as part of Alessandro Di Siena's 2020 thesis, [Di 20], for $u_{\parallel s}$, $T_{\parallel s}$, $T_{\perp s}$ functions of ψ. Unfortunately, the code in its form from the thesis is not available to the public, and is only partially maintained as part of the `GENE` GPU development branch- many portions of the code presume a Maxwellian background, and do not account for non-Maxwellian backgrounds.

1.2 Damped Distributions

Another common effect to consider is Alfvénic damping, wherein high-energy particles with velocities on the scale of the Alfvén velocity $v_A = \frac{B}{\sqrt{\mu_0 \rho}}$ lose energy due to resonance with the Alfvén waves, causing velocity distributions with sharper cut-offs than the $\mathcal{O}\left[\exp\left(-\mathrm{const.}v^2\right)\right]$ tails in Maxwellians, shifted Maxwellians, bi-Maxwellians etc. A general way to consider such "damped distributions" is presented here.

Suppose, before damping, the background has some classic distribution, e.g. a shifted Maxwellian, with distribution function $f_{s0}^{\mathrm{pre}}\left[\psi; v_\parallel^{\mathrm{pre}}, \mu^{\mathrm{pre}}\right]$ at a given position. Consider this velocity being modified by some "damping function", $(v_\parallel^{\mathrm{pre}}, \mu^{\mathrm{pre}}) \mapsto (v_\parallel, \mu)$ that reduces the velocity of the high-velocity particles, but leaves the low-velocity particles unchanged. Denote this as:

$$v_\parallel(v_\parallel^{\mathrm{pre}}, \mu^{\mathrm{pre}}) \qquad \mu_\parallel(v_\parallel^{\mathrm{pre}}, \mu^{\mathrm{pre}}) \tag{7}$$

Similarly, this has the inverse:

$$v_\parallel^{\mathrm{pre}}(v_\parallel, \mu) \qquad \mu_\parallel^{\mathrm{pre}}(v_\parallel, \mu) \tag{8}$$

A reasonable condition on this damping function is that it leaves the velocity of low-energy particles unchanged, namely:

$$\partial_{v_\parallel^{\mathrm{pre}}} v_\parallel(0,0) = 1 \qquad \partial_{v_\parallel^{\mathrm{pre}}} \mu(0,0) = 0 \tag{9}$$

$$\partial_{\mu^{\mathrm{pre}}} v_\parallel(0,0) = 0 \qquad \partial_{\mu^{\mathrm{pre}}} \mu(0,0) = 1 \tag{10}$$

Similarly, for the inverse:

$$\partial_{v_\parallel} v_\parallel^{\mathrm{pre}}(0,0) = 1 \qquad \partial_{v_\parallel} \mu^{\mathrm{pre}}(0,0) = 0 \tag{11}$$

$$\partial_{\mu} v_\parallel^{\mathrm{pre}}(0,0) = 0 \qquad \partial_{\mu} \mu^{\mathrm{pre}}(0,0) = 1 \tag{12}$$

The distribution function f_{s0} for the damped distribution will then evaluate as

$$f_{s0}[\psi; v_\parallel, \mu] = J(v_\parallel, \mu) f_{s0}^{\mathrm{pre}} \left[\psi; v_\parallel^{\mathrm{pre}}(v_\parallel, \mu), \mu^{\mathrm{pre}}(v_\parallel, \mu)\right] \tag{13}$$

where J is the Jacobian

$$J(v_\parallel, \mu) := \partial_{v_\parallel} v_\parallel^{\mathrm{pre}}(v_\parallel, \mu) \partial_\mu \mu^{\mathrm{pre}}(v_\parallel, \mu) - \partial_\mu v_\parallel^{\mathrm{pre}}(v_\parallel, \mu) \partial_{v_\parallel} \mu^{\mathrm{pre}}(v_\parallel, \mu) \tag{14}$$

As such, to create a damped velocity distribution, only 3 "ingredients" are needed:

- $f_{s0}^{\mathrm{pre}} \left[\psi; v_\parallel^{\mathrm{pre}}, \mu^{\mathrm{pre}}\right]$, the velocity distribution *before damping*.
- $v_\parallel^{\mathrm{pre}}(v_\parallel, \mu)$, the parallel velocity a damped particle had *before damping*.
- $\mu^{\mathrm{pre}}(v_\parallel, \mu)$, the magnetic moment a damped particle had *before damping*.

Example. (Uniform damping) *Consider a damping function wherein a particle's pitch angle remains unaffected by the damping, but the speed* $v = \sqrt{v_\parallel^2 + \frac{2B}{m_s}\mu}$ *is damped uniformly among all pitch angles through some function* $v^{\mathrm{pre}}(v)$*, such that:*

$$v_\parallel^{\mathrm{pre}}(v_\parallel, \mu) = \frac{v^{\mathrm{pre}}(v)}{v} v_\parallel \tag{15}$$

$$\mu^{\mathrm{pre}}(v_\parallel, \mu) = \frac{v^{\mathrm{pre}}(v)}{v} \mu \tag{16}$$

Such a $v^{\mathrm{pre}}(v)$ *would need to:*

- *satisfy the low-velocity condition* $\partial_v v^{\mathrm{pre}}(0) = 1$.
- *go to* ∞ *much faster than* v *for* $v > v_A$ *(or whatever the velocity scale on which the damping takes effects may be).*

Such a function may for example take the form

$$v^{\mathrm{pre}}(v) := \left[1 + \left(\frac{v}{v_A}\right)^r\right] v \tag{17}$$

2 Implementation and Usage

2.1 Background Distributions in GENE

The source code for GENE can be found in src/. This implementation builds on the GPU support branch of the development version of GENE, as it comes the closest to having fully functional support for varied background distributions.

This branch of GENE features options for background distributions (e.g. Maxwellian/shifted Maxwellian/etc.) through classes dist_eq_*_t (in a corresponding dist_eq_*_m module) that extend the base distribution class dist_eq_t (part of the dist_eq_base_m module). (See Figure 1) For each new species that uses such a background distribution, a new instance of this class is created.

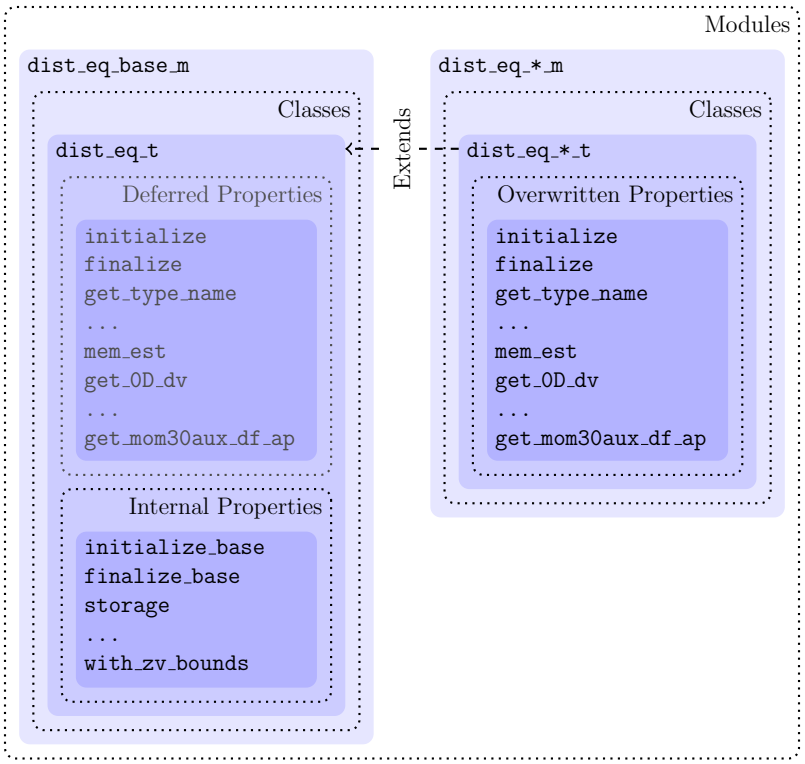

Figure 1: GENE equilibrium distribution class structure *before* addition of damping modules.

Creating a new background distribution involved creating a new such dist_eq_*_t type (in a corresponding dist_eq_*_m module) and defining the properties (all procedures) deferred from dist_eq_t. These include procedures for:

- Evaluating certain mathematical properties of the distribution: get_0D_dv, get_C12, get_mom00_phi, etc.
- Interaction with the rest of GENE: initialize, finalize, get_type_name, etc.

The distribution procedures for evaluating mathematical properties correspond to the functionals on the background distribution f_{s0} as indicated in Figure 2.[1]

Procedure	Functional
`get_*D_dv`	$\partial_{v_\parallel} f_{s0}$
`get_*D_dm`	$\partial_\mu f_{s0}$
`get_*D_dvdm`	$\frac{B_0}{2}\partial_{v_\parallel} f_{s0} - v_\parallel \partial_\mu f_{s0}$
`get_edr_prefactor`	$\partial_\psi f_{s0}$ (without B derivative?)
`get_edr`	$\partial_\psi f_{s0}$ (without B derivative?) + stuff
`get_C12*`	const. $\int (1 - J_0^2)\, v_\parallel dv_\parallel \partial_\mu f_{s0} d\mu$
`get_C22*`	const. $\int \left[\frac{B_0 v_\parallel}{2}\partial_{v_\parallel} f_{s0} - v_\parallel^2 (1 - J_0^2)\partial_\mu f_{s0}\right] dv_\parallel d\mu$
`get_C32*`	const. $\int J_0 I_1 v_\parallel \mu \partial_\mu f_{s0} dv_\parallel d\mu$
`get_mom*'a''b'*_phi`	const. $\int \left(\phi_1 - \langle\overline{\phi}_1\rangle\right) \partial_\mu f_{s0} v_\parallel^a \sqrt{\mu}^b dv_\parallel d\mu$
`get_mom*'a''b'*_ap`	const. $\int \left(A_{1,\parallel} - \langle\overline{A}_{1,\parallel}\rangle\right) \left(\frac{B_0}{2}\partial_{v_\parallel} f_{s0} - v_\parallel \partial_\mu f_{s0}\right) v_\parallel^a \sqrt{\mu}^b dv_\parallel d\mu$
`get_mom*'a''b'*_bpar`	const. $\int \mu \langle\overline{B}_{1,\parallel}\rangle \partial_\mu f_{s0} v_\parallel^a \sqrt{\mu}^b dv_\parallel d\mu$

Figure 2: Table of how the various procedures for a `GENE` distribution class should evaluate.

The terms in the last two sections of Figure 2 can be found in [Di 20], in section 2.12 *"Ampère's Law for $A_{1,\parallel}$"* as $\mathcal{L}$, $\mathcal{H}$, $\mathcal{K}$, and as the components of equation (2.80) in section 2.9 *"Velocity Moments for General Backgrounds"*, respectively. See there also for an explanation of the J_0 and I_1 terms, although these are already available in general in `GENE` as part of the `gyroaverager_m` module.

2.2 Damping Implementation

"Damped distribution functions" are incorporated in this work into `GENE` similarly how one would any other distribution function: with a new `dist_eq_*_damped_t` class (in a `dist_eq_*_damped_m` module) extending `dist_eq_t`. Whereas a traditional background distribution needs only its distribution function, f_{s0} (or more accurately its distribution function's parameters, n_s, $u_{\parallel s}$, T_s, etc.) to be defined, a damped distribution requires (as highlighted in subsection 1.2) 3 "ingredients":

[1] *DISCLAIMER*: For the latter of these terms, do not take my word as gospel. The expressions in the code seem to differ from those in the literature by some *wild* constants in places—in some cases in so far as a *sign switch*—and, even in Di Siena's implementation in the GPU `GENE` branch, he says he hasn't evaluated some of these terms. There is very little reference point or annotation in the pre-existing code, and no clear definitions for how these procedures should evaluate in the literature.
What connections I have drawn between these terms and those in the literature, I have done so by (very laboriously) scanning it through to find the terms that resemble those in the code.

- $f_{s0}^{\text{pre}}\left[\psi; v_\parallel^{\text{pre}}, \mu^{\text{pre}}\right]$ (or more accurately the parameters, n_s, $u_{\parallel s}$, T_s, etc. of) the velocity distribution *before damping*.
- $v_\parallel^{\text{pre}}(v_\parallel, \mu)$, the parallel velocity a damped particle had *before damping*.
- $\mu^{\text{pre}}(v_\parallel, \mu)$, the magnetic moment a damped particle had *before damping*.

Within the implementation, therefore, a class/module for a *damped* distribution looks almost identical to one for a *non-damped* distribution, except for its definition containing a new variable that carries information on the damping function, $(v_\parallel, \mu) \mapsto (v_\parallel^{\text{pre}}, \mu^{\text{pre}})$. This is done through:

1. General implementation of the new class, `damping_t` (in the `damping_base_m` module), the class of damping functions, with space all the relevant information on the damping function.
2. User implementation of a new class, `damping_*_t` (in a `damping_*_m` module), extending `damping_t`, for each specific damping function—say, *uniform* damping, as defined in subsection 1.2—the user would like to consider, defining the damping function $(v_\parallel, \mu) \mapsto (v_\parallel^{\text{pre}}, \mu^{\text{pre}})$ in consideration.
3. User implementation of a new subclass `dist_eq_*_damped_t` (in a `dist_eq_*_damped_m` module), extending `dist_eq_t`, just as one might do already to implement a new distribution, except this time containing (and initializing) a new variable `dist_eq_*_damped%damp` of type `damping_*_t` containing information on the damping function in question.

This new class structure is highlighted in red in Figure 3.

As for what information should be contained in the `damping_t` class, consider what information is needed about the damping function to evaluate all the procedures in Figure 2. Since both `get_*D_dvdm` and the terms in the last two sections can be written[2] in terms of those in the first section, and the only component of `get_edr` that changes between distribution function classes is that which is featured in `get_edr_prefactor`, the only 3 terms that actually need evaluating to implement a new distribution function class are simply the 3 derivatives of f_{s0} in $v_\parallel$, μ, ψ:

Procedure	Functional
`get_*D_dv`	$\partial_{v_\parallel} f_{s0}$
`get_*D_dm`	$\partial_\mu f_{s0}$
`get_edr_prefactor`	$\partial_\psi f_{s0}$ (without B derivative?)

Recall then equation (13) for f_{s0}

$$f_{s0}[\psi; v_\parallel, \mu] = J(v_\parallel, \mu) f_{s0}^{\text{pre}}\left[\psi; v_\parallel^{\text{pre}}(v_\parallel, \mu), \mu^{\text{pre}}(v_\parallel, \mu)\right]$$

[2] if they cannot be evaluated directly

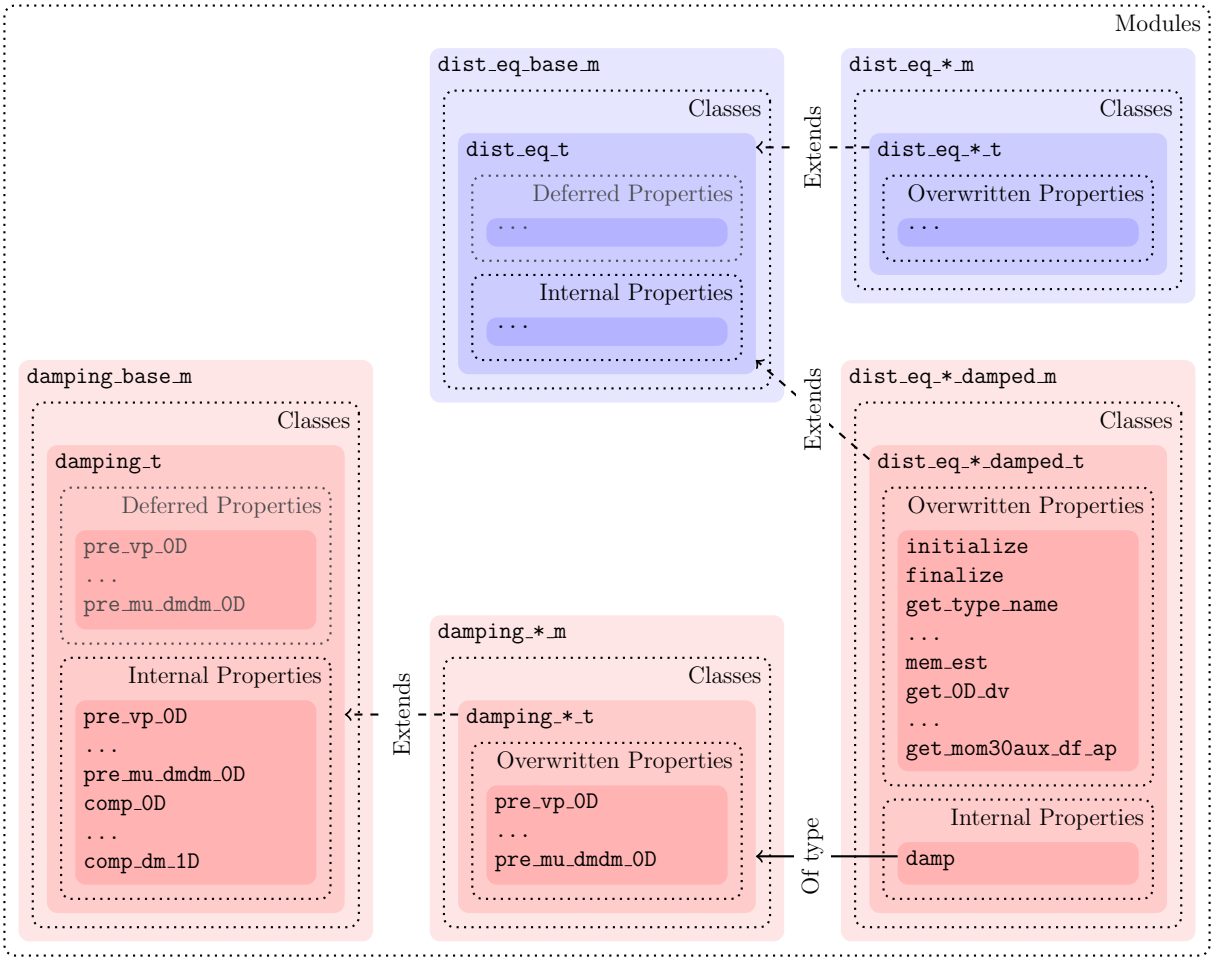

Figure 3: GENE equilibrium distribution class structure *after* addition of damping modules. Base modules marked in blue, additions marked in red.

alongside equation (14) for J

$$J(v_\parallel, \mu) := \partial_{v_\parallel} v_\parallel^{\text{pre}}(v_\parallel, \mu)\partial_\mu \mu^{\text{pre}}(v_\parallel, \mu) - \partial_\mu v_\parallel^{\text{pre}}(v_\parallel, \mu)\partial_{v_\parallel} \mu^{\text{pre}}(v_\parallel, \mu)$$

The 3 relevant derivatives in f_{s0} then evaluate as:

$$\partial_{v_\parallel} f_{s0} = \partial_{v_\parallel} J \cdot f_{s0}^{\text{pre}} + J \cdot \left(\partial_{v_\parallel^{\text{pre}}} f_{s0}^{\text{pre}} \cdot \partial_{v_\parallel} v_\parallel^{\text{pre}} + \partial_{\mu^{\text{pre}}} f_{s0}^{\text{pre}} \cdot \partial_{v_\parallel} \mu^{\text{pre}}\right) \tag{18}$$

$$\partial_\mu f_{s0} = \partial_\mu J \cdot f_{s0}^{\text{pre}} + J \cdot \left(\partial_{v_\parallel^{\text{pre}}} f_{s0}^{\text{pre}} \cdot \partial_\mu v_\parallel^{\text{pre}} + \partial_{\mu^{\text{pre}}} f_{s0}^{\text{pre}} \cdot \partial_\mu \mu^{\text{pre}}\right) \tag{19}$$

$$\partial_\psi f_{s0} = J \cdot \partial_\psi f_{s0}^{\text{pre}} \tag{20}$$

alongside the derivatives in J:

$$\partial_{v_\parallel} J = \left(\partial_{v_\parallel}^2 v_\parallel^{\text{pre}} \cdot \partial_\mu \mu^{\text{pre}} + \partial_{v_\parallel} v_\parallel^{\text{pre}} \cdot \partial_{v_\parallel}\partial_\mu \mu^{\text{pre}}\right) - \left(\partial_{v_\parallel}\partial_\mu v_\parallel^{\text{pre}} \cdot \partial_{v_\parallel} \mu^{\text{pre}} + \partial_\mu v_\parallel^{\text{pre}} \cdot \partial_{v_\parallel}^2 \mu^{\text{pre}}\right) \tag{21}$$

$$\partial_{v_\parallel} J = \left(\partial_{v_\parallel}\partial_\mu v_\parallel^{\text{pre}} \cdot \partial_\mu \mu^{\text{pre}} + \partial_{v_\parallel} v_\parallel^{\text{pre}} \cdot \partial_\mu^2 \mu^{\text{pre}}\right) - \left(\partial_\mu^2 v_\parallel^{\text{pre}} \cdot \partial_{v_\parallel} \mu^{\text{pre}} + \partial_\mu v_\parallel^{\text{pre}} \cdot \partial_{v_\parallel}\partial_\mu \mu^{\text{pre}}\right) \tag{22}$$

A new class of damping function `damping_*_t` should therefore specify, not only $v_\parallel^{\text{pre}}$, μ^{pre}, but their derivatives up to 2nd order. The evaluation of J and its derivatives is identical for all damping functions, and so is implemented in general as an internal procedure contained in `damping_t`.

2.3 Code and Usage

The repository for the implementation can be found here (https://gitlab.mpcdf.mpg.de/g-borisandrews/gene-gpu). Unfortunately, since I don't have access to the `GENE` development repository, this is built off a copy of the latest version (as of time of writing) of the CUDA GPU development branch, `cuda_under_the_hood`, and so it won't be possible for me to maintain this. Anyone with access to the `GENE` development repository however is more than welcome to do so!

To overview the contents of subsection 2.2, on the contents of the implementation, to run a GENE simulation featuring a species with a damped background distribution function, one would need to do the following:

1. Create a new class, `damping_*_t` (in a new `damping_*_m` module) extending `damping_t` for the damping function to be considered.

   Uniform damping (as detailed in section 1.2) is already supported, in `damping_uniform_m`.

2. Create a new class, `dist_eq_*_damped_t` (in a new `dist_eq_*_damped_m` module) extending `dist_eq_t` with a new internal variable `dist_eq_*_damped%damp` of type `damping_*_t`. Overwrite the procedures deferred from `dist_eq_t` as one normally would for a new class of distribution functions. (Make reference to Figure 2 to check how these distribution procedures are intended to evaluate, and don't forget to initialize the damping object `dist_eq_*_damped%damp`!)

   Damped Maxwellians and shifted Maxwellians are already supported, in `dist_eq_maxwellian_damped_m` and `dist_eq_shifted_maxwellian_damped_m` respectively.

3. Add the new class of distribution function as a case in the `initiailize_fm` procedure in the `vel_space` module.

4. Ensure, as ever, all the new files and their dependencies are added to the relevant makefiles.

5. Compile.

6. Cross your fingers!

References

[Abe+13] I. G. Abel et al. “Multiscale Gyrokinetics for Rotating Tokamak Plasmas: Fluctuations, Transport and Energy Flows”. In: *Reports on Progress in Physics* 76.11 (Oct. 2013), p. 116201. DOI: 10.1088/0034-4885/76/11/116201. URL: https://doi.org/10.1088/0034-4885/76/11/116201.

[Di 20] A. Di Siena. “Implementation and Investigation of the Impact of Bifferent Background Distributions in Gyrokinetic Plasma Turbulence Studies”. en. PhD thesis. Universität Ulm, Apr. 2020, p. 247. ISBN: 9781695601505. DOI: 10.18725/OPARU-29528.