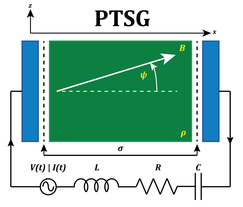
Self-adapting EEDF evaluation frequency in KGMf*

Janez Krek, John Verboncoeur

Michigan State University ({krek,johnv}@msu.edu)

*Work supported by DOE Plasma Science Center grant DE- SC0001939



Motivation

The Kinetic Global Model framework (KGMf) uses 0D simulation model to explore complicated chemistry in multi-species systems. With short simulation time compared to higher dimensional models it present natural choice for parameter scanning in systems with many species and many reactions while keeping model's limitation in mind. By adding self-consistent evaluation of electron energy distribution function (EEDF) one can improve accuracy of simulation but also make total computational time longer, therefore negating the advantage of the KGMf regarding fast simulation times.

Predefined and evaluated EEDF

The predefined EEDF in isotropic velocity space using shape parameter x[1] is defined as:

$$f(\epsilon) = \frac{x}{\left(\frac{3}{2}T_{eff}\right)^{3/2}} \frac{\left[\Gamma\left(\frac{5}{2x}\right)\right]^{3/2}}{\left[\Gamma\left(\frac{3}{2x}\right)\right]^{5/2}} \epsilon^{1/2} \; exp\left\{-\left[\frac{\Gamma\left(\frac{5}{2x}\right)}{\Gamma\left(\frac{3}{2x}\right)} \frac{\epsilon}{\left(\frac{3}{2}T_{eff}\right)^{3/2}}\right]^{x}\right\} \quad (1)$$

where Γ is the Gamma function, ϵ is an electron energy (in eV) and x is a shape parameter ($x = 1.0$: Maxwellian distribution, $x = 2.0$: Druyvesteyn distribution).

The KGMf coupled with Boltzmann equation solver BOLOS[2] enables self-consistent evaluation of the EEDF in given simulation steps. A single computation of the EEDF with the two-term approximation takes on the order of tens of milliseconds[3], which increases the required computational time in KGMf by two orders of magnitude[4] compared to cases with a fixed EEDF, if computation is performed in each step. Implicit integrator in KGMf ("ode" integrator from SciPy library) internally makes additional calls calls to EEDF evaluation when iterating towards solution in current time step. This adds to total number of calls to EEDF evaluation method.

Reaction rate coefficients K

In case of known cross section $\sigma(\varepsilon)$, EEDF $f(\varepsilon, y)$ and impact velocity $v(\varepsilon)$, the reaction rate coefficient $K(t, y)$ is computed as (see [5], page 22):

$$K_i(t, y) = \int_0^\infty v_\alpha(\varepsilon)\sigma_i(\varepsilon)f_\alpha(\varepsilon, y)d\varepsilon \quad (2)$$

where y is any state value in a system (T_e, n_α, n_α/n_{total}) and ε is relative impact kinetic energy. Reaction coefficients depend on state values and not only time and computed even in every call to EEDF evaluation method - even during "sub step" calls from integrator.

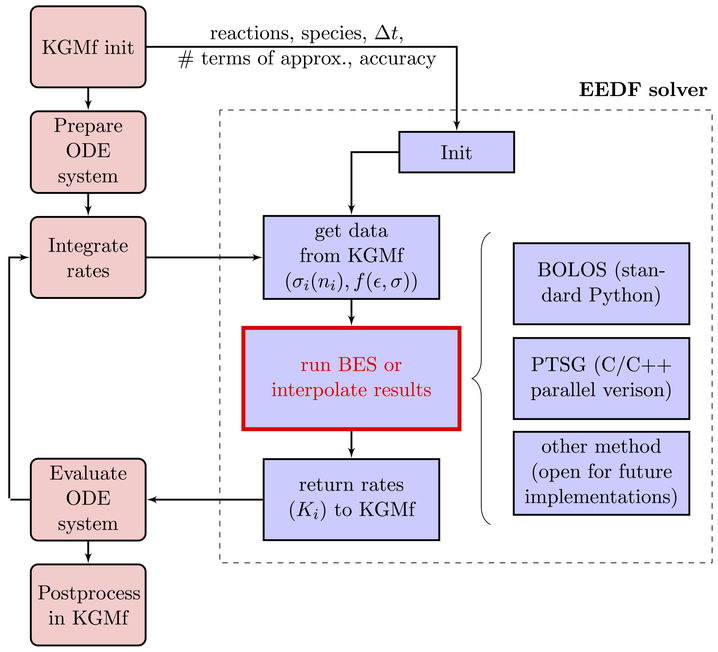


Figure 1: The flowchart of coupling of the "EEDFsolver" and the KGMf.

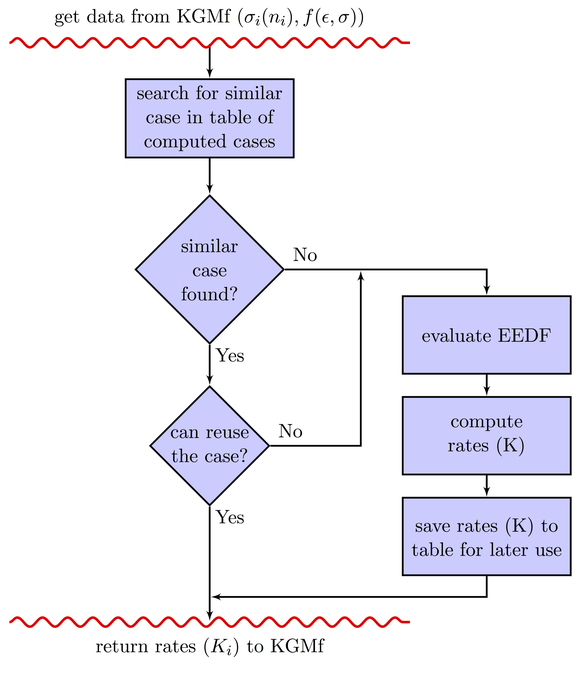


Figure 2: The flowchart of the steps in determining of EEDF evaluation frequency (detailed view of red box from Fig. 1).

Self-adapting EEDF evaluation frequency

The KGMf offers manual definition of the EEDF evaluation frequency. Evaluation frequency can be defined as a number of computational steps, or a relative change in T_e or a relative change in reduced electric field (E/N) between individual EEDF evaluations.
Self-adapting EEDF evaluation frequency is currently based only on change of the reduced electric field.

Results

Figure 3 shows computation time reduction in case of reducing number of EEDF evaluations by defining number of steps between EEDF evaluations.

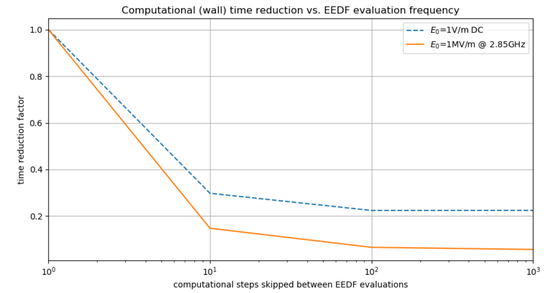


Figure 3: The impact of the EEDF evaluation frequency on total computational (wall) time.

Future work

- use of machine learning to extend the number of parameters that are used to predict changes in rate coefficients (not only reduced electric field) between EEDF evaluations
- reuse values of already computed EEDFs
- optimizing Boltzmann equation solver to utilize multi-core/multi-thread
- consider other methods (e.g. Monte Carlo) for computing EEDF

References

[1] S.K.Nam, J.P.Verboncoeur, App. Phy. Letters, 92, 231502 (2008)
[2] A. Luque, https://pypi.python.org/pypi/bolos (2004)
[3] G.J.M.Hagelaar, L.C.Pitchford, Plasma Sources Sci. Technol. 14, 722-733 (2005)
[4] J.Krek, G.Parsey, J.P.Verboncoeur, ICOPS 2016
[5] G.Parsery, PhD thesis, MSU, (2017)