Progress toward a Python interface for DEGAS2

George J. Wilkie
Princeton Plasma Physics Laboratory

with thanks to: Daren Stotler, Xin Zhang

12 January 2023

PPPL
PRINCETON
PLASMA PHYSICS
LABORATORY

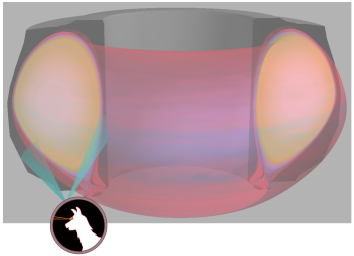
BOUT++ workshop

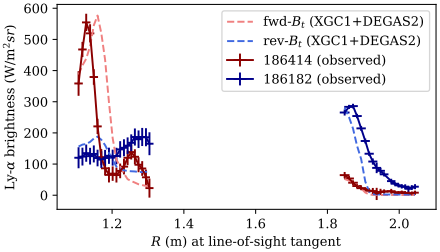
Outline

- Overview of DEGAS2
- The path to modernization
- Installation and workflow demo

Plans for DEGAS2

- Primary computational research focus is **coupling to kinetic plasma models**.
  - Recent success with XGC reproducing line radiation diagnostics on DIII-D:





- Renew coupling to fluid simulations for longer timescales and more accessible computational resources.
- Workflows exist for coupling to UEDGE, and TRANSP's new 2-point SOL model *(Zhang, PhD thesis, 2022)*, and other tools.
- Create a generalized workflow that interfaces with DEGAS2 in python.

Monte Carlo is a useful technique to estimate arbitrary moments in control volumes, especially when colliding against known background distributions

The Boltzmann equation for a neutral species s:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla f_s = S + \sum_{s'} C\left[f_s, f_{s'}\right]$$

Spanier & Gelbard 1969

Monte Carlo is a useful technique to estimate arbitrary moments in control volumes, especially when colliding against known background distributions

The Boltzmann equation for a neutral species s:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla f_s = S + \sum_{s'} C\left[f_s, f_{s'}\right]$$

With an integrating factor, the steady-state linear[†] Boltzmann equation reveals itself as a **Fredholm integral equation of the 2nd kind**:

$$f\left(\mathbf{r}, \mathbf{v}\right) = Q\left(\mathbf{r}, \mathbf{v}\right) + \int \mathrm{d}^3\mathbf{r}' \int \mathrm{d}^3\mathbf{v}' \, \mathcal{V}\left[\mathbf{v}', \mathbf{v}; \mathbf{r}\right] \mathcal{R}\left(\mathbf{r}', \mathbf{r}; \mathbf{v}\right) f\left(\mathbf{r}', \mathbf{v}'\right)$$

[†] DEGAS2 generalizes this for: *Spanier & Gelbard 1969*

- Time dependence: appropriate adjustments to volumetric source.
- Nonlinear collisions: iterating upon simplified BGK model.

Monte Carlo is a useful technique to estimate arbitrary moments in control volumes, especially when colliding against known background distributions

The Boltzmann equation for a neutral species s:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla f_s = S + \sum_{s'} C\left[f_s, f_{s'}\right]$$

With an integrating factor, the steady-state linear† Boltzmann equation reveals itself as a **Fredholm integral equation of the 2nd kind**:

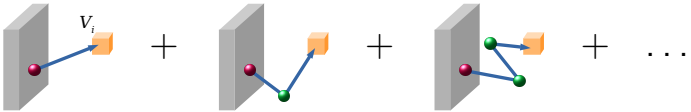
$$f\left(\mathbf{r}, \mathbf{v}\right) = Q\left(\mathbf{r}, \mathbf{v}\right) + \int \mathrm{d}^3\mathbf{r}' \int \mathrm{d}^3\mathbf{v}' \, \mathcal{V}\left[\mathbf{v}', \mathbf{v}; \mathbf{r}\right] \mathcal{R}\left(\mathbf{r}', \mathbf{r}; \mathbf{v}\right) f\left(\mathbf{r}', \mathbf{v}'\right)$$

† DEGAS2 generalizes this for: *Spanier & Gelbard 1969*

- Time dependence: appropriate adjustments to volumetric source.
- Nonlinear collisions: iterating upon simplified BGK model.

Solved with a **Neumann series** expansion with successive applications of the operators $\mathcal{R}$ (propagation in space), and $\mathcal{V}$ (propagation in velocity) acting upon Q (the source).

$$\int_{V_i} \mathrm{d}^3\mathbf{r}' \int \mathrm{d}^3\mathbf{v}' f\left(\mathbf{r}', \mathbf{v}'\right) =$$



$+ \quad + \quad + \quad \ldots$

DEGAS2 is a mature and robust Monte Carlo neutral transport solver

- Written in 1994 by Daren Stotler and Charles Karney.
- Shares many similarities with European counterpart EIRENE.
- Borrows Monte Carlo estimation techniques from neutron transport algorithms for fission.
- Includes reaction rates and cross sections for many atomic physics processes, often employing a collisional radiative model.
- Written in FWEB framework.
- Available on github:
  https://github.com/PrincetonUniversity/degas2
  (email for access)

FWEB

- WEB was concieved as a solution to the problem of code documentation. *(Knuth, Literate Programming, 1992)*
  - Same source file generates pdf documentation and the computer code
  - Notable examples: TeX, Metafont
- FWEB is the Fortran version.
  - Generous (ab)use of macros enable object-oriented design, as the problem generally demands.
- Generated Fortran is **deliberately obtuse**.

FWEB

A WEB system of structured documentation for multiple languages

2.1 The FWEB processors: FWEAVE and FTANGLE

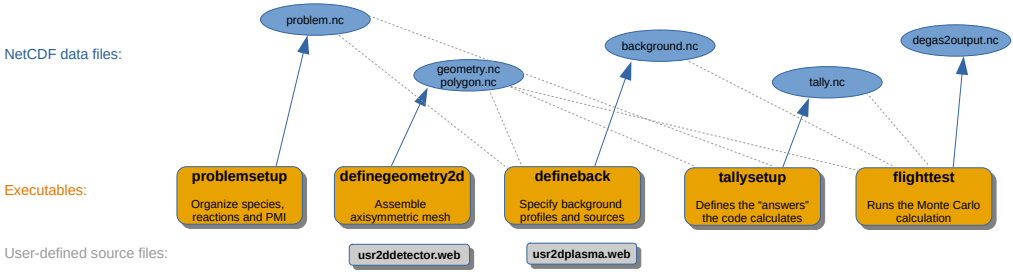
Following Knuth's original design, FWEB consists of two processors, FTANGLE and FWEAVE. Both operate on a single source file, say '`test.web`'. FTANGLE produces compilable code, say '`test.c`', whereas FWEAVE produces a TEX file, '`test.tex`', that can (in principle) be processed with either TEX or LATEX. (If a file '`test.tex`' already exists, FWEAVE will ask for confirmation before overwriting it if it does not think that the file was created by a previous run of FWEAVE.)

The output file produced by FTANGLE is not intended for human eyes (or for editors!); it is for compiling only. All changes to the code should be made to the `web` file, since changes made directly to the output file would be overwritten the next time the `web` source is tangled. In an attempt to discourage messing with FTANGLE's output file, all unnecessary spaces are deliberately removed.

A common way of integrating FWEB into ones program development is to do all com-

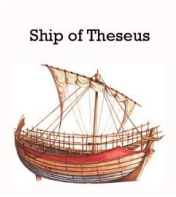
DEGAS2 structure

DEGAS2 is a collection of discrete executables that communicate sequentially via NetCDF files.



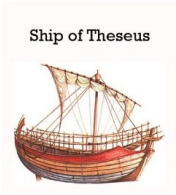
Python workflows are available for specific use-cases and post-processing.

The structure of DEGAS2 uniquely lends itself well to refactoring via encapsulation



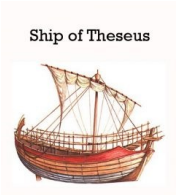
- A productive paradigm for refactoring legacy code: build wrappers in new syntax, write tests, then replace modular capability.
  - No multiple versions, original capability maintained throughout whole process.
- To the extent Python scripts create and manipulate the NetCDF datafiles equally generally to the original code, they *replace* it.

The structure of DEGAS2 uniquely lends itself well to refactoring via encapsulation



- A productive paradigm for refactoring legacy code: build wrappers in new syntax, write tests, then replace modular capability.
  - No multiple versions, original capability maintained throughout whole process.
- To the extent Python scripts create and manipulate the NetCDF datafiles equally generally to the original code, they *replace* it.
- Status:
  - Builds with CMake.
  - Workflows collected for wrapping input-generation and post-processing in Python.
  - Benchmarks distributed as examples being converted to automatic integration tests.
  - Rewrite of geometry specification ongoing.

The structure of DEGAS2 uniquely lends itself well to refactoring via encapsulation



- A productive paradigm for refactoring legacy code: build wrappers in new syntax, write tests, then replace modular capability.
  - No multiple versions, original capability maintained throughout whole process.
- To the extent Python scripts create and manipulate the NetCDF datafiles equally generally to the original code, they *replace* it.
- Status:
  - Builds with CMake.
  - Workflows collected for wrapping input-generation and post-processing in Python.
  - Benchmarks distributed as examples being converted to automatic integration tests.
  - Rewrite of geometry specification ongoing.
- Plan:
  1. Automate example benchmarks as integration tests and implement continuous integration.
  2. Integrate with IMAS databases.
  3. Replace setup routines with equally-general modern code: choosing Python.
  4. Main Monte Carlo calculation TBD: probably C++/Kokkos.

Geometric considerations

- DEGAS2 calculates *volume averages* of selected moments of the neutral distribution.
  - Most directly analogous to first-order finite volume methods
- The fundamental geometric constructs are *quadratic surfaces* described by 10 coefficients:

$$\sum_{i=0}^{2} \sum_{j=0}^{2-i} \sum_{k=0}^{2-i-j} a_{ijk} x^i y^j z^k = 0$$

  Examples: cones, disks, spheres, paraboloids, hyperboloids.
  - Each of these surfaces are uniquely identified and define the boundaries of all control volumes.
- 2D axisymmetric cases are primarily for the *convience of the user* than any limitation of the algorithm.

Demonstrations

- Installation demo
  github.com/gjwilkie/degas2.git
  - Compiles with CMake
  - Distributes FWEB and triangle in repo and SILO as submodule

- Python workflow demonstration from TRANSP 2-point SOL module.

Questions for BOUT++ community

- Is there existing infrastructure for the PFC shape and properties?
- Where, exactly, is "the wall"? Is it the last node on the mesh, a ghost cell beyond it, the face of the cell, or something else?
- Is there a natural *finite volume* representation of the mesh?
- How can we define control volumes for the 3D mesh of BSTING?

Backup

Coupling to non-Maxwellian plasma introduces important and difficult challenges

DEGAS2 needs to know the collision frequency (reaction rates) for each interaction:

$$\nu(\mathbf{v}) = \int |\mathbf{v} - \mathbf{w}| \sigma \left(|\mathbf{v} - \mathbf{w}| \right) f_i(\mathbf{w}) \, \mathrm{d}^3\mathbf{w}.$$

($\mathbf{v}$ = neutral velocity. $\mathbf{w}$ = ion velocity)
These collision frequencies are used to determine how likely a collision will occur for a given trajectory sample. Once an interaction is "chosen", the cross section is used to resolve the collision.

Coupling to non-Maxwellian plasma introduces important and difficult challenges

DEGAS2 needs to know the collision frequency (reaction rates) for each interaction:

$$\nu(\mathbf{v}) = \int |\mathbf{v} - \mathbf{w}| \sigma\left(|\mathbf{v} - \mathbf{w}|\right) f_i(\mathbf{w}) \, \mathrm{d}^3\mathbf{w}.$$

($\mathbf{v}$ = neutral velocity. $\mathbf{w}$ = ion velocity)
These collision frequencies are used to determine how likely a collision will occur for a given trajectory sample. Once an interaction is "chosen", the cross section is used to resolve the collision.

- For Maxwellian ions, the collision frequencies are tabulated as a function of ion temperature and relative speed.
- For non-Maxwellian field particles, this is not so straightforward ...

Maxwell molecules make kinetic ion-neutral coupling feasible, but greater accuracy is desired

Consider treating collision partners as "effective Maxwellians" for the purpose of resolving collisions between ions and neutrals. The error in estimating, for example, the energy exchange, is:

$$\mathrm{Err}\left[v^2\right] = \int\int u\sigma_{cx}(u) f_n(\mathbf{v}) \left[f_i(\mathbf{w}) - f_i^{(M)}(\mathbf{w})\right] \left(w^2 - v^2\right) \, \mathrm{d}^3\mathbf{w} \, \mathrm{d}^3\mathbf{v},$$

and does not vanish even if $f_i^{(M)}$ has identical temperature to f_i.

Using this Maxwellian approximation with the physical cross section results in a lack of energy conservation.

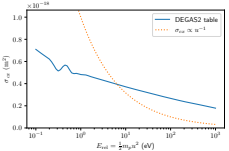
Wilkie, et al. IAEA FEC (2020/2021)

Maxwell molecules make kinetic ion-neutral coupling feasible, but greater accuracy is desired

Consider treating collision partners as "effective Maxwellians" for the purpose of resolving collisions between ions and neutrals. The error in estimating, for example, the energy exchange, is:

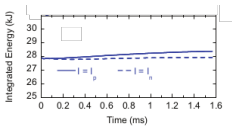
$$\mathrm{Err}\left[v^2\right] = \int\int u\sigma_{cx}(u) f_n(\mathbf{v}) \left[f_i(\mathbf{w}) - f_i^{(M)}(\mathbf{w})\right] \left(w^2 - v^2\right) \,\mathrm{d}^3\mathbf{w}\,\mathrm{d}^3\mathbf{v},$$

and does not vanish even if $f_i^{(M)}$ has identical temperature to f_i.



Using this Maxwellian approximation with the physical cross section results in a lack of energy conservation.

- The reason is because $\mathbf{v}$ and $\mathbf{w}$ are coupled through $\mathbf{u} = \mathbf{v} - \mathbf{w}$.
- Using $\sigma \propto u^{-1}$ ("Maxwell molecules") removes this coupling.
- Incidentally, this is also the form of the cross section that would make a BGK operator rigorous for charge exchange.
  (see: GBS, GKEYLL, XGC's internal neutral module, etc.)
- A solution is forthcoming.



Wilkie, et al. IAEA FEC (2020/2021)