LATEST ADVANCEMENTS OF THE TEST-PARTICLE MONTE CARLO CODE MOLFLOW

P. Trifunović*, M. Ady, R. Kersevan
European Organization for Nuclear Research, Geneva, Switzerland

Abstract

MolFlow is a test-particle Monte Carlo code for ultra-high vacuum simulations, primarily intended for use in the field of particle accelerators. This contribution gives an overview of updates made to the code in recent years. The graphical user interface has been improved and a new feature has been added for extracting simulation results along user-defined 3D paths. Additionally, the code can now count the number of particles present on a surface at a specified time. Furthermore, the calculation of particle residence time (sojourn time) has been updated to allow for temperature variations during a particle's residence on a surface. This enables simulations of surface decontamination through heating (bake-out). Finally, MolFlow can now simulate particle collisions with static background gases using the hard-sphere collision model.

INTRODUCTION

MolFlow is a simulation tool for ultra-high vacuum, originally developed in the 1990s. In 2007 it was modernized and rewritten in C++, preserving the original algorithm [1]. Using the test-particle Monte Carlo method, it traces virtual particles in 3D geometries from gas sources to absorption points such as vacuum pumps. The geometry is defined as a set of planar polygons, which represent the boundaries of the vacuum space. Together with their physical properties, such as the outgassing rate, sojourn time, temperature, sticking factor etc., these polygons are referred to as *facets*. As the simulation runs, MolFlow keeps track of the number of virtual particles hitting each facet, which enables calculation of physical properties of the system, such as pressure, impingement rate and density. These properties can be visualized in the form of plots or color-coded cells on facets called textures. With time, the simulation converges to a more precise state and the user can decide when the precision is satisfactory. A more detailed description of this simulation tool and its internal workings can be found in the official documentation [2], in the introduction to MolFlow's algorithm [1], and in previous IPAC contributions [3, 4].

This article presents an overview of the most important updates of MolFlow in recent years, with reference to version 2.11.0 of the MolFlow code [5], the latest release at the time of writing.

* petar.trifunovic@cern.ch

NEW TOOL FOR DATA EXTRACTION ALONG A PATH

Until recently, extracting results and values at different points in the geometry at once was not possible out-of-the-box and required workarounds such as creating transparent sampling facets that stretch along the geometry and record particle hits. In the case of curved geometry shapes, these workarounds were cumbersome and not straightforward for users to configure manually.

A new tool has been added to facilitate result extraction. It enables users to define paths in the geometry and extract results along them. Each path is defined as a set of points with two consecutive points defining a vector as one path segment. Paths are split into a number of equal-sized bins specified by the user. The users can select a set of facets with textures and extract simulation results from them along a path. Each texture cell's result are assigned to the closest bin. The distance from a texture cell to a bin is the length of the orthogonal projection vector from the cell's center to the bin. If a cell cannot be orthogonally projected to the path, it is not included in the extraction. The extraction can be visualized as a plot of either the sum of texture cell results per bin (useful for number of hits) or as an average per bin (useful for pressure, impingement rate, or density).

Figure 1 shows a curved geometry, with a path visualized with red dots and blue arrows. Simulation results can be extracted from the line of textured facets. Figure 2 shows the new tool with a plot of average pressure extracted along the defined path.

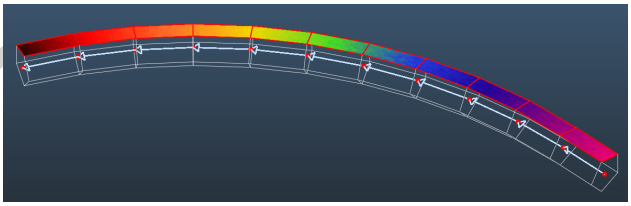

Figure 1: Curved shape for texture extraction.

REFLECTION COUNTER AND PARTICLE BALANCE CALCULATION

In previous versions, facet counters recorded three quantities—desorptions (number of particles created and outgassed from the facet), absorptions (number of particles stuck to (pumped by) the facet), and incident hits (sum of absorbed and bounced particles). *Bouncing*, or *reflection*, denotes particles that hit a facet and reflect from it without being absorbed. In time-dependent mode, such a counter is maintained for each defined time moment.

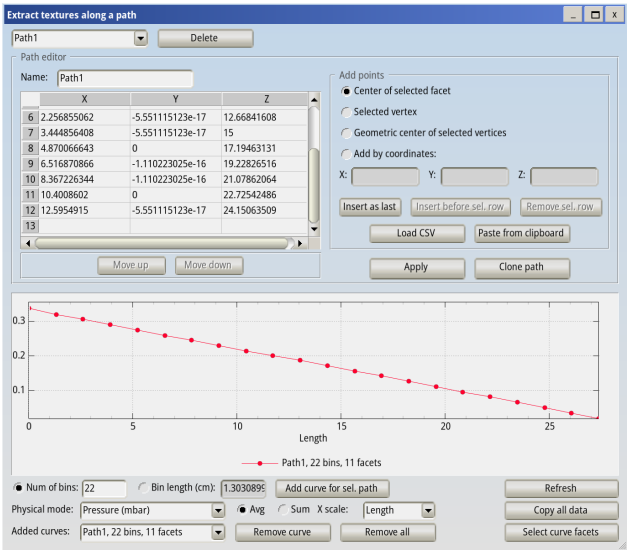

Figure 2: Texture extraction tool window.

For context, it is useful to understand the definition of time moments in MolFlow. Each moment is defined by its center-point and a recording tolerance called a *window*. If a moment is defined with a center-point of 5 s and a window of 2 s, events that occur in the period from 4 s to 6 s belong to this moment.

It is also possible to enable *sojourn time* calculation for simulations. If enabled, particles that hit a facet at a certain moment will spend some time on it, defined by user-specified sojourn time properties, and may leave the facet during the same moment's window, or at a later moment. Previously, there was no counter for recording the number of particles reflected at each moment, but this feature was added recently. Note that it only counts particles that leave a facet as a result of bouncing. Those created by a facet itself, which leave during initial outgassing, are counted as a desorption, not a reflection. As MolFlow is not time-driven, the counters for moments are independent of each other, and are not necessarily updated in chronological order. For example, a particle may arrive at a facet at moment M, spend some time on it, and leave at moment $M + 2$. The reflection counter for $M + 2$ can still be updated, even if the counter for $M + 1$ was not updated previously.

By summing independent moment counters, it is possible to calculate the so-called *cumulative quantities*. For example, cumulative hits for moment M are equal to the sum of all hits from moments <= M. This allows the calculation of particle balance on a facet at any given moment M as the cumulative hits until M reduced by the cumulative absorptions (particles that are pumped-out and are not present on the facet anymore) and cumulative reflections. It is important to note that cumulative calculations are possible only if the defined time moments and their windows are continuous, without gaps, and with no overlapping—a time window of one moment must start where the preceding window ends.

VARIABLE TEMPERATURE DURING SOJOURN TIME

The mean sojourn time of a particle adsorbed by a facet with binding energy E at temperature T is given by Frenkel's equation [6]:

$$\tau = \tau_0 \exp\left(\frac{E}{RT}\right) \tag{1}$$

where R is the ideal gas constant and τ_0 is the particle's vibration period. The sojourn time has an exponential probability distribution [6], so the probability density function of the sojourn time t is:

$$f(t) = \nu \exp(-\nu t) \tag{2}$$

where $\nu = 1/\tau$ is the escape rate of an adsorbed particle.

Integrating Eq. (2) from 0 to some time t' results in the probability $F(t')$ that the sojourn time is less than or equal to t':

$$F(t') = \int_0^{t'} \nu \exp(-\nu t)\, dt = 1 - \exp(-\nu t') \tag{3}$$

To generate random sojourn times for particles, following this exponential distribution, MolFlow first generates a random value r uniformly distributed between 0 and 1. By starting from $F(t') = 1 - r$, a random sojourn time t for a particle can be calculated as:

$$t = -\frac{\ln(r)}{\nu}$$

This approach works if the temperature remains unchanged during particle's sojourn time. However, if the temperature can change with time, ν also becomes time dependent and the same approach cannot be used for calculating the sojourn time.

To solve this problem, MolFlow was recently updated to allow temperature changes during sojourn time. To achieve this, the survival probability function is used. For a time t, this function represents the probability that the particle survives on the facet until t, and is defined as:

$$S(t) = 1 - F(t) \tag{4}$$

At the start of a simulation, the user defines temperature values at discrete time moments. This temperature function is divided into smaller time intervals inside which temperature can be assumed to be constant. The initial survival probability at time 0 s is 1, $S(0) = 1$. Knowing the value of $S(t)$, if temperature is constant between t and a later time t', the probability of surviving from t to t' is

$$S(t'|t) = 1 - F(t'|t) = \exp(-\nu(t' - t)) \tag{5}$$

The probability to survive from the start until t' is:

$$S(t') = S(t) \cdot S(t'|t) = S(t) \cdot \exp(-\nu(t' - t)) \tag{6}$$

Since $S(0) = 1$, it is possible to pre-calculate the values of S at the limits of all the following small time intervals,

assuming constant temperature during these intervals. Every subsequent value of S is less than the previous one, reaching 0 at infinity. If a particle arrives at a moment t_{arr}, the algorithm looks for the closest pre-calculated $S(t)$ such that $t < t_{arr}$ and calculates $S(t_{arr})$ using Eq. (6) and swapping t' for t_{arr}. Next, as the value of S decreases towards 0 with time, a uniform random number, $S(t_{refl})$, is generated between $S(t_{arr})$ and 0. Again, Eq. (6) is used, but this time to calculate t', which is now t_{refl}. This is the point in time at which the particle leaves the facet. All other values in the equation are known—$S(t')$ is $S(t_{refl})$; $S(t)$ is the closest pre-calculated S before $S_(t_{refl})$. Finally, the sojourn time is equal to $S(t_{refl}) - S(t_{arr})$.

Previously, the input of particles into the system could only be specified as an outgassing rate per unit of time, and the sojourn time was not applied to these initially outgassed particles. Another recent addition to MolFlow allows users to set up outgassing as the initial number of particles on a facet, as well as an option to apply sojourn time to these initial particles. Together with the described algorithm for temperature-dependent sojourn time, this enables simulations of surface decontamination through heating (bakeout). Consider a simulation where a set of outgassing facets is populated with some amount of initial particles that are subject to sojourn time, and the system is being heated. The simulation is time dependent and the defined moments cover the time span from 0 to 10 000 s. The temperature is a step function—it remains constant for 1000 s before increasing by 5 K, starting from 300 K, as shown by the blue line in Fig. 3.

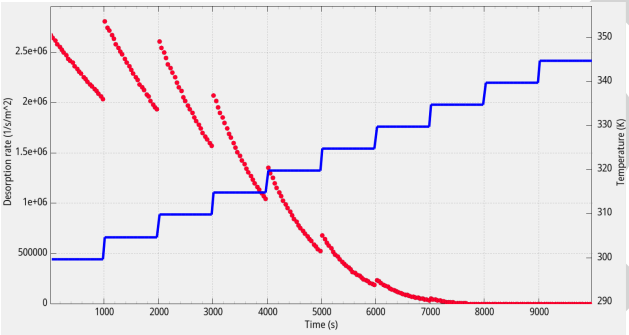

Figure 3: Average desorption rate (red) and temperature (blue) evolution with time. Taken from MolFlow's built-in plotter.

The sojourn time is calculated with a binding energy of 49.69 kJ mol^{-1} and attempt frequency of 124.41 kHz. At $T = 300$ K, this results in a mean sojourn time of 3600 s. At $T = 305$ K, the mean is 2597 s, at $T = 310$ K it is 1893 s and so on. As the temperature rises, the mean sojourn time decreases and the particles leave the facets earlier. Red points in Fig. 3 show the average desorption rate at each observed time moment. Since the temperature changes by "jumping" by 5 K every 1000 s, the decrease of mean sojourn time and the consequent increase in desorption rate is also sudden, as shown in the figure. However, since the facets were populated with a limited number of initial particles, the overall desorption rate reduces with time, as there are less and less particles left in the system.

COLLISIONS WITH BACKGROUND GAS

CERN's vacuum group often performs coating of surfaces by a sputtering process, where a cathode emits particles that end up on the wall of the coated element. These particles don't fly in a straight path, as they occasionally collide with the plasma process gas.

To model this behavior, a *background gas collision* feature was added, where the simulated gas can randomly hit particles of the background gas through the hard-sphere collision model, which is the simplest assumption.

The user defines the *mean free path*, that can be calculated from the radius of the simulated and background gases and the background gas density, and the *mass ratio* of the simulated/background gas. MolFlow then generates random collision events, and executes the three governing equations (energy and momentum conservation, and force direction acting along the collision path) to determine the post-collision velocity and direction. In this approximation, the background gas is considered static.

As seen in Fig. 4, an initially collimated gas jet diverges through subsequent collisions—in practice, particle trajectories are "blurred" around the straight line flight path, but for smaller mass ratios, they can even turn back following a collision.

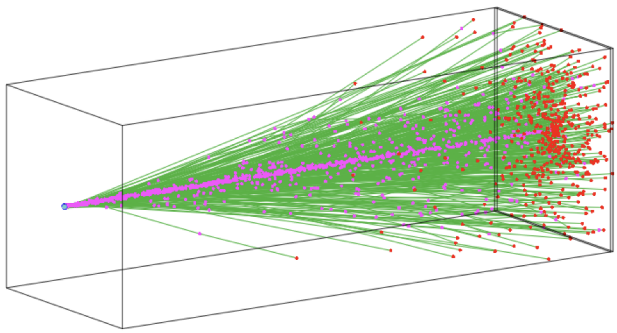

Figure 4: A gas jet randomly hitting static background gas and changing flight path. Collision locations are marked with purple, and wall hits with red.

SUMMARY

This contribution presented several recent updates of MolFlow. Some of these improved the data extraction and visualization capabilities of the software, while others added entirely new features that, if used, affect the simulation algorithm and the results. Together, these changes improve the user experience and extend the overall applicability.

REFERENCES

[1] R. Kersevan and J.-L. Pons, "Introduction to MOLFLOW+: New graphical processing unit-based Monte Carlo code for simulating molecular flows and for calculating angular coefficients in the compute unified device architecture environment",

J. Vac. Sci. Technol., A, vol. 27, no. 4, pp. 1017–1023, Jun. 2009. doi:10.1116/1.3153280

[2] M. Ady and P. Trifunović, Molflow/synrad documentation, 2026, https://molflow.docs.cern.ch/

[3] M. Ady and R. Kersevan, "Introduction to the Latest Version of the Test-particle Monte Carlo Code Molflow+", in *Proc. IPAC'14*, Dresden, Germany, Jun. 2014, pp. 2348–2350. doi:10.18429/JACoW-IPAC2014-WEPME038

[4] R. Kersevan and M. Ady, "Recent Developments of Monte-Carlo Codes Molflow+ and Synrad+", in *Proc. IPAC'19*, Melbourne, Australia, May 2019, pp. 1327–1330. doi:10.18429/JACoW-IPAC2019-TUPMP037

[5] R. Kersevan, M. Ady, P. Trifunovic, and J.-L. Pons, MolFlow: A Monte Carlo simulator for Ultra High Vacuum systems, 2026, https://gitlab.cern.ch/molflow_synrad/molflow.git

[6] J. Frenkel, "Theorie der Adsorption und verwandter Erscheinungen", *Zeitschrift für Physik*, vol. 26, no. 1, pp. 117–138, Dec. 1924. doi:10.1007/BF01327320