Radiation Penetration Depth in Particle Beds

Monte Carlo Ray Tracing for beam radiation striking a particle bed

Evan F Johnson

https://github.com/ef-johnson/Radiation-Penetration-Depth-in-Particle-Beds

This code is a modified version of the Monte Carlo ray tracing code “Ray-Tracing-Many-Spheres” (https://github.com/ef-johnson/Ray-Tracing-Many-Spheres). Ray Tracing Many Spheres finds the RDF or View Factors from one sphere to other spheres and walls. In contrast, "Radiation Penetration Depth in Particle Beds" is for the case of beam radiation (parallel rays) striking a particle bed. Since most of the code and parameters are the same, see Ray Tracing Many Spheres for more background detail.

If you use the concepts, equation or code from this project, please cite one of the following articles:

[1] E.F. Johnson, İ. Tarı, D. Baker, Radiative heat transfer in the discrete element method using distance based approximations, Powder Technol. 380 (2021) 164–182. doi:10.1016/j.powtec.2020.11.050.

[2] E. Johnson, İ. Tarı, D. Baker, A Monte Carlo method to solve for radiative effective thermal conductivity for particle beds of various solid fractions and emissivities, J. Quant. Spectrosc. Radiat. Transf. 250 (2020). doi:https://doi.org/10.1016/j.jqsrt.2020.107014.

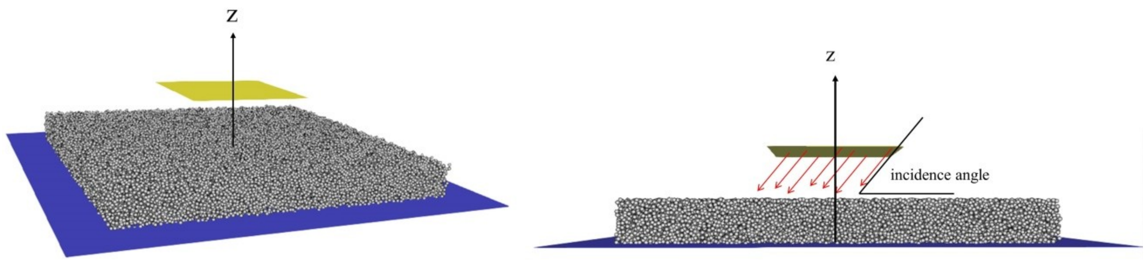
Code and OS:

The code is written in parallel C++, developed Ubuntu 16.04. Ubuntu or a similar Linux based operating system is recommended, since other useful programs exist for working with particles. Most importantly, these include LIGGGHTS for Discrete Element Method (DEM) modeling to generate the particle domains, and ParaView for visualization.

Particle Bed Geometry

The particle bed must be generated in some other program. The geometry convention is that the particle bed sits on the z=0 plane, as shown in the figures below. Photons are emitted from a random location on the plane above the particle bed, which is parallel to the z plane and shown in yellow in the figures. Photons are emitted from this plane at an incidence angle, which is defined as the angle from the z-plane to the photon trajectory: 90^{0} means photons directly strike the particles from above.

Any photon reaching the wall at z=0 is considered an automatic absorption into the wall.



The particle bed is defined with the *position* (or *pos*) file. This is a space-separated text file containing the xyz positions of the particle centers, with one line for each set of particle coordinates. For an ordered packing (such as simple cubic, body centered cubic, etc.) the particle centers could be computed and saved as a text file with a program such as Matlab or Octave. For generating realistic particle beds, LIGGGHTS is recommended for DEM modeling.

The *position* file must be named: *pos_[*number of particles*]_[*solid fraction*]_[*DEM time step number*].txt*, and placed in the *pos* directory. The number of particles is the number of lines in the *pos* file. The solid fraction and DEM time step number are not actually used in the simulation, but these parameters will be passed to the name of the output file. This is done so you can keep track of many different simulations based on their solid fraction and DEM time step number. If these are unknown, they could be assigned as any numbers which help you keep track of the different simulations and their input/output files.

Two *pos* files are included in the *pos* folder. The one named *pos_17500_0.52_1000.txt* has a simple cubic packing structure, and *pos_28367_0.60_9000.txt* has a random packing. Particles have a radius of 0.01 m in both packed beds.

Recommended Prerequisites:

1) LIGGGHTS for doing DEM simulations, to generate particle beds

2) OpenMPI must be installed to run parallel simulations. This is a prerequisite for LIGGGHTS and many other programs.

3) ParaView, recommended for visualizing particles as spheres, with the “Point Gaussian” representation.

To run a simulation (in Ubuntu):

1) Make sure the *pos* file is in the *pos* folder

2) The source code in the main folder is compiled to make the executable with the command: *mpicxx -std=c++11 Rad_Depth_1.1.cpp -o Rad_Depth_1.1*

3) The executable is run using MPI with the command: *mpirun -np 10 Rad_Depth_1.0* (where the number of processors is 10 in this case)

4) Input parameters are requested by the program:

- "Enter number of photons in scientific notation (ex: 1e5):" This refers to the number of photons PER PROCESSOR which will be emitted.
- "Enter nominal volume fraction (must be in format 0.XX):" The volume fraction (solid fraction) of particles, for keeping track of input/output files
- "Enter number of particles:" This must match the actual number of particles, or lines in the *pos* file.
- "Enter absorptivity of the particles (format 0.XX):"
- "Enter LIGGGHTS time step number (format XXXXXX):" Simply for naming the input/output files.
- "Enter radius of the particles (meters):"

- "Enter the xy max to emit photons from. Square is from x=+/- this value and y=+/- this value. (meters):" Photons are emitted from a random location from inside the square of emission, which is parallel to the z-plane and has a maximum x and y value defined here. Recommendation: use a smaller square for emitting photons than the particle bed itself, so you don't lose photons out the sides (especially important for low solid fractions and low particle absorptivities).
- "Enter the z plane to emit photons from (meters):" The z-position of the square of emission. Recommendation: put this plane just above the highest part of the highest particle – you don't want photons to miss the particle bed entirely, if the incidence angle is acute, say 15^{o}.
- "Enter the incidence angle (angle between the xy plane the incident rays) in degrees:"

One final input parameter, which is not included in these initial questions, is the specular/diffuse choice. This must be changed in the source code by changing "specular=true" for specular reflections and "specular=false" for diffuse reflections.

Alternatively, these values can be hard-coded into the source code, which is sometimes easier to work with. There is a commented-out code block near the beginning where these values can be hard-coded. Of course, then the code must be recompiled each time a parameter is changed in that case.

Also, there are several *for* loops (commented out by default) to iterate over various particle absorptivities and incidence angles, which can be used to easily run many simulations in succession, for parametric studies.

Output files

Two output files are written at the end of the simulation:

1) *abs_loc_all_[*solid fraction*]_[*number of particles*]_[*number of photons*]_[*particle absorptivity*]_[*DEM time step*]_[*incidence angle*]*

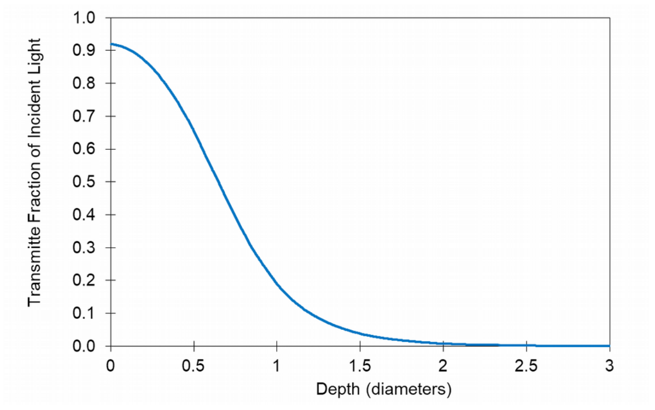
   This file contains the xyz location where each photon was absorbed, either on the surface of a particle or at the z=0 wall.

2) *zPos_ctr_[*solid fraction*]_[*number of particles*]_[*number of photons*]_[*particle absorptivity*]_[*DEM time step*]_[*incidence angle*]*

This file contains the particle IDs (line number in the *pos* file), the z position of the particle center, and the number of photons absorbed by that particle. Thus, the photon absorptions are associated with the particle centers in this case. [Note that these columns of information are written by *each processor*, so there are n_processors*n_particles lines in this file. The data can be re-shaped and combined during post-processing to find the total number of hits (from all processors) for each particle.]

In the column of particle IDs, the last ID is that of the wall, which is represented by an ID of -1, and a z-position of 0.

Post-processing can be done to find data such as the depth vs. transmission curve shown below:


Transmitte Fraction of Incident Light
Depth (diameters)