MULTP-M CODE GEOMETRY IMPORT MODULE PERFORMANCE OPTIMISATION

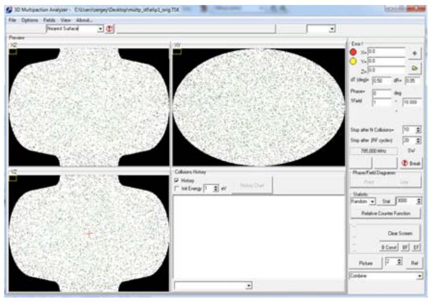
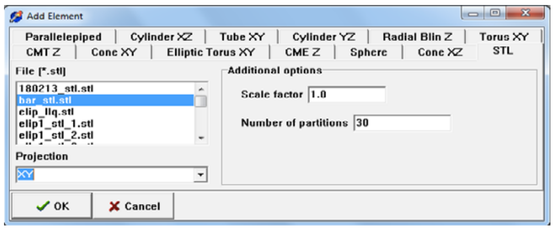
S.A. Khudyakov, M.A. Gusarova, M.V.Lalayan, National research nuclear university MEPhI, Moscow, Russia

Abstract

Introduces the new features of the module import geometry for three-dimensional modeling program multipactor MultP-M. On an example, consider an increase in the speed and accuracy of the calculation using a new algorithm for calculating the use of loading geometry format STL.

INTRODUCTION

Earlier [1] new module of geometry import for multipactor discharge simulation code MultP-M implementation and testing results were presented. This upgrade allows device under investigation geometry to be directly imported as STL file. Previously device under simulation shape was described using Boolean operations on basic geometry primitives like brick, torus, sphere etc. Results obtained using this code were compared and found coincident with known numeric, analytical and experimental data. Figure 1 shows geometry import module interface developed.

Projection: XY

Scale factor: 1.0

Number of partitions: 30

Figure 1: STL import module interface.

Tests showed [1] that new module operates correctly and could be used instead of preceding one with practically the same accuracy as it is illustrated in Figure 2. However despite of pretty effective algorithms and numeric models implementation it was found that computation time significantly grows for fine mesh models. This paper reports the simulation algorithm optimization solutions developed for MultP-M code that allow it to operate faster without detriment to accuracy.

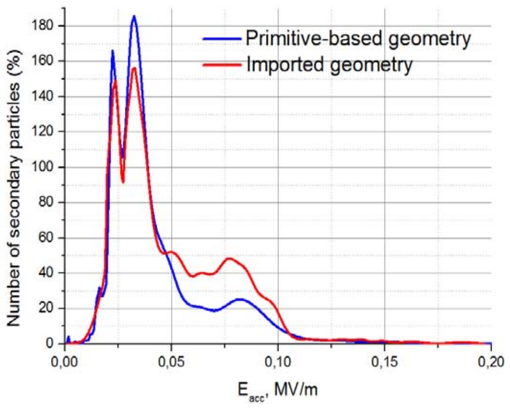


Figure 2: Simulation accuracy test.

MULTP-M AND STL FILE

While MultP-M multipactor discharge simulation code runs the main task to be solved is to determine each electron location with respect to model confines, i.e. to decide whether particle is inside the model boundaries or not. In case of boundaries defined as set of geometric primitives this task could easily solved using simple math. STL file describes 3D objects by their facets thus making this math much more complicated.

Geometry import module developed incorporates ray tracing algorithm [2] in order to get point position with coordinates (x, y, z) relative to facets specified 3D body.

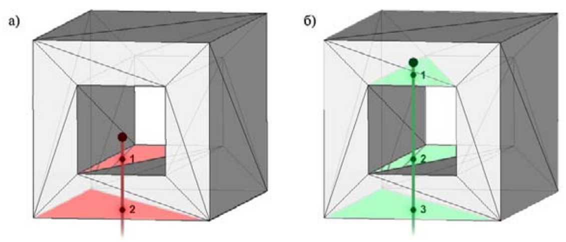


Figure 3: Different point and body collocation: a – point is outside body, ray has even number of boundary crosses; b – point is inside body, odd crosses number.

This algorithm demands 3D body under consideration to be closed. STL standard also sets the same requirement to all objects described.

So one has to develop algorithm that calculates number of ray and boundary crossings. This algorithm is

ISBN 978-3-95450-170-0

to be accurate and effective with respect to code performance.

Net crossings number is calculated by searching for possible ray intersection with every facet. In case ray crosses the plane of facet one should check whether the point lies inside facet or not. In case of STL file triangle facets are used therefore known intersection theory for barycentric (areal) coordinate system [3] suits well.

However for facets number exceeding 20,000 severe computation speed drop was noted despite of powerful algorithms implemented for checking of ray and plane intersection and if cross point belongs to facet triangle.

Thorough algorithm performance analysis and its efficacy for different facets number it was stated that ray traced form every point crosses just a few facets. Crossings number is appreciably less than total facets amount and it seldom exceeds 5 to 6. Obviously algorithm computation rate increases with diminishing facets considered number. This could be done if only facets in some area close to ray are considered.

Basic idea of algorithm developed is to split whole body into equal areas (Figure 4). First the area consisting ray source point with coordinates (x, y, z) is considered. Then algorithm checks for crossings of ray and facets only from selected area. In case of triangle facet partially lies in several adjacent areas it is simultaneously attributed to the both because only triangles of single selected area are considered.

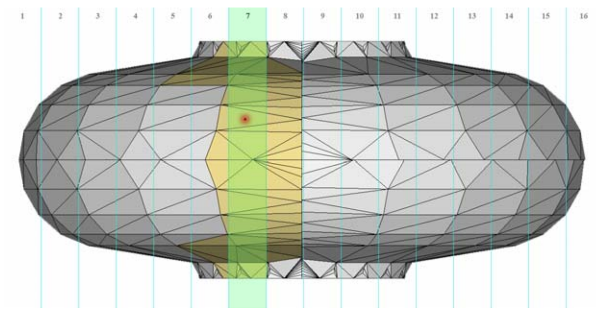


Figure 4: Area selection scheme.

Algorithm developed will run correctly in case the following conditions are met:

1. projection plane (XY, XZ, YZ) for the whole model division in areas is to be chosen;
2. ray traced from given point is to be perpendicular to the chosen projection axis.

These conditions are caused by the fact that in order to evaluate ray traced from the point and surface crossing number all facets are to be checked. Figure 5 shows computation time dependence on triangles number for two algorithms, one without optimization and the second with optimization at 50 areas.

It is clear that non-optimized algorithm run time has liner dependence on triangles number. Optimization leads to noticeable run time growth only for large models having 250,000 triangles.

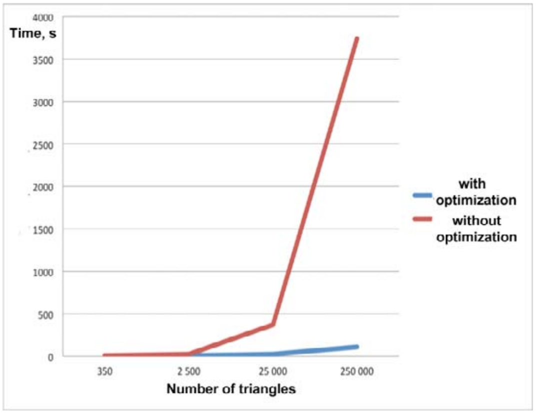


Figure 5: Different algorithms run time vs. triangles number.

It should be mentioned however that is triangles number in every area increases with total triangles number growth, so areas number is also to be increased. This fact is illustrated by the dependence of run time vs. areas number for model having 250,000 triangles shown in Figure 6.

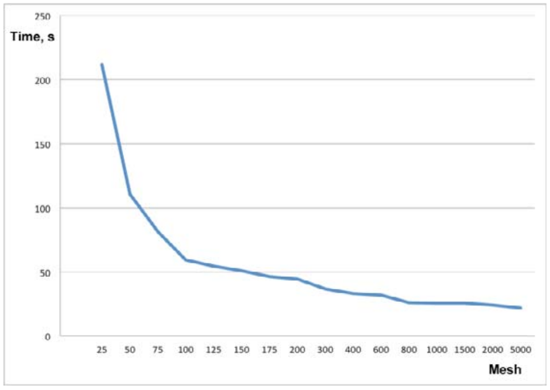


Figure 6: Computation time vs. areas number for model of 250000 triangles.

CONCLUSION

Model import module algorithm of computer simulation code MultP-M was done. This upgrade made simulated task preparation time shorter and more convenient.

REFERENCES

[1] Gusarova M.A., Petrushina I.I., Khuduakov S., The new possibility of MultP-M code, Proceedings of IPAC’14.

[2] Eric Haines, Point in Polygon Strategies. http://erich.realtimerendering.com/ptinpoly/. 2001.

[3] Steve Marschner. Ray-Surface Intersection. Cornell CS417 Spring. 2003.