EUROFUSION WPBB-CP(16) 15707

L Lu et al.

Improved Solid Decomposition Algorithms for the CAD-to-MC Conversion Tool McCad

Preprint of Paper to be submitted for publication in Proceedings of 29th Symposium on Fusion Technology (SOFT 2016)

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Improved Solid Decomposition Algorithms for the CAD-to-MC Conversion Tool McCad

Lei Lu, Yuefeng Qiu, Ulrich Fischer

Karlsruhe Institute of Technology (KIT), Institute for Neutron Physics and Reactor Technology.
Hermann-von-Helmholtz-Platz 1,76344, Eggenstein-Leopoldshafen, Germany

McCad is a geometry conversion tool developed at the Karlsruhe Institute of Technology (KIT) for the automatic conversion of CAD models into the semi-algebraic geometry representations which can then be used in Monte Carlo (MC) particle transport simulations applied in design analyses of fusion and fission reactors. A new algorithm for the decomposition of complex geometry models was designed and implemented in the McCad code. With this improvement, a weakness of the original McCad was mitigated which is due to the instability of the applied graphic kernel and the original decomposition algorithm. The improvements have been verified with some representative test models and a generic model of a DEMO fusion power reactor. The results show that the advanced McCad version with the new decomposition algorithm is more robust and provides more accurate and less complex conversion results. The decomposition process is efficient and produces correct results which are consistent with the original model.

Keywords: McCad, MC modeling, BRep-CSG decomposition.

1. Introduction

McCad is a geometry conversion tool that enables the automatic conversion of CAD models into the semi-algebraic geometry representations utilized in Monte Carlo (MC) particle transport simulations. McCad is entirely based on open-source software and libraries utilizing Open Cascade (OCC) as CAD kernel and the Qt4 and OpenGL libraries for the graphical user interface (GUI). With McCad, a CAD model can be processed, converted into MC geometry representation and output in the syntax of MC codes such as MCNP, TRIPOLI and Geant4 [1-3]. In addition, related visualization capabilities based on the coupling of McCad with the ParaView software allows the user to overlay mesh tally distributions on the CAD geometry.

The kernel algorithms of McCad conversion include two significant processes: solid decomposition and void generation. An input CAD solid with complex geometry is decomposed into a collection of disjoint and simple convex solids which can be represented by Boolean forms of primitive solids or algebraic half-spaces. Then the void spaces in and around the decomposed solids are generated and exported as filling solids. The previous version of McCad has already included an improved void filling algorithm and function [4]. However, the decomposition function implemented previously in McCad is still not very efficient and stable when applied to large and complex geometry models resulting in frequent programme crashes, and fragmentized and irregular decomposed solids. Moreover, a lot of CPU time and memory are consumed. To overcome such difficulties and improve the capability of McCad, new decomposition algorithms and functions have been developed which employ some new techniques, e.g., the triangular collision detection, assistant splitting surfaces adding and the splitting surfaces sorting. In addition, the memory management system has also been optimized that decreases the CPU time and memory consumption and accelerate the decomposition process.

The new decomposition algorithm and its implementation in McCad have been verified with some example models and components extracted from ITER device. Moreover, the EU DEMO generic model is adopted as an entire combination model to verify the complete CAD-MC conversion.

2. Solid Decomposition

2.1 Workflow

The solid decomposition function of McCad reads and decomposes a complex CAD model recursively into a combination of convex solids with a number of splitting surfaces. Therefore, creating a sufficient and suitable set of splitting surfaces is the first step of the decomposition. Most splitting surfaces can be selected directly from the boundary surfaces which are traversed and the splitting surfaces will be detected and stored in the splitting surface list. However, if the solid contains curved boundary surfaces, e.g., cylinders, cones, spheres and torus, and they are also splitting surfaces, the original boundary surfaces are not sufficient to describe the geometry eventually and some auxiliary splitting surfaces must be added for separating these curved surfaces from the solid [4].

When sufficient splitting surface set is ready, the next step is the practical decomposing process with Boolean operations. However, the sequence of the splitting surfaces utilized significantly affects the quality of the final result and the number of Boolean operations used. Fig.1 shows two different results of a simple solid decomposition with different splitting surfaces sequence. Here solution *A* needs only one Boolean operation and 2

author's email: lei.lu@kit.edu

convex solids are generated. Solution *B* needs 3 Boolean operations and 4 solids are generated. Obviously, solution A is a better solution.

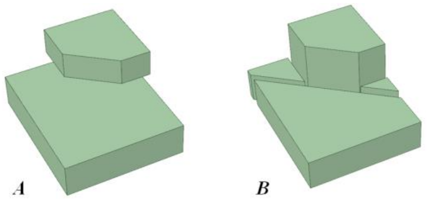


Fig. 1. Two different decomposition results employing a different sequence of splitting surfaces

Therefore, optimizing the sequence of splitting surfaces as far as possible is the most important step for generating a less complex decomposition result with fewer Boolean operations.

2.2 Requirements of improvements

Theoretically speaking, when a sufficient set of splitting surfaces is constructed, a solid could always be decomposed eventually. However, the graphic kernel OCC employed in McCad is not very stable and robust in performing Boolean operations, especially when a complex geometry is processed. Therefore, McCad's original algorithm has to be optimized for a minimum number of Boolean operations. For example, there is a simple method for detecting the splitting boundary surfaces which intersect a solid with the positive and negative half spaces of each boundary surface. If the solid has common parts with both half spaces, the boundary surface is recognized as splitting surface. However, with this method, a large number of Boolean operations are required.

Furthermore, McCad's original algorithm generates the auxiliary splitting surfaces simply with the boundary edges of the curved surface. When the radians of curved surfaces are small, this process might result in some irregular shapes of the decomposed solids, e.g., thin plates with sharp corners which are found to be one of the important reasons for causing subsequent Boolean operation errors [4]. So the second requirement is to refine the auxiliary splitting surfaces adding algorithm, avoiding the generation of irregular and error-prone solids.

In addition, the original decomposition function of McCad didn't sort the splitting surfaces but selects them randomly. This is the reason why previous decomposition results were fragmental and more CPU time was consumed. Therefore, a sorting function is required for generating a simpler and more regular result that has fewer decomposed solids and the boundary surfaces included. Another objective is to reduce the number of Boolean operations, as already noted above, and accelerate the decomposing process.

In summary, the improvements of the decomposition function aim at decomposing a complicate CAD model into simple and regular sub-solids with fewer Boolean operations to ensure the process is stable and efficient. An optimized decomposition result also affects the subsequent simulation speed with MC codes. Along these guideline new decomposition functions have been developed as shown in the following.

3. Improvements

3.1 Detect splitting surfaces with triangles collision

Instead of Boolean intersections, the triangles collision detecting algorithm is employed to classify the splitting boundary surfaces and non-splitting surfaces. The input CAD model is first meshed into a quantity of triangles and then the relative positions of the triangles and boundary surface are calculated. The boundary surface is a splitting surface if one of the triangles is colliding with it or the triangles are located at both sides of it, otherwise it is a non-splitting surface. The meshing function is a conventional base function of the geometry kernel for visualizing the geometries, which is fast and stable. In the meantime, coarse or fine triangles can be also controlled by given precisions.

The relative positions between plane and triangles are easy to be calculated. If the three vertices of a triangle are located at both sides of the boundary surface there are collisions. Therefore, the complicate collision detecting is simplified and just an issue of the numerical calculation. However, it is not sufficient to calculate the relative position between triangles and curved surfaces only with the vertices. As shown on the example in Fig.2, Triangle *A* and *B*, two vertices of them locate on the surface *S* and one vertex at the outside. The triangle *A* has a collision with *S*, but triangle *B* is considered as having no collision with *S* because the collision area is too small. So the areas of collision are also important factors for detecting that a curved surface is a splitting surface or not, and a tolerance must to be given to the collision area.

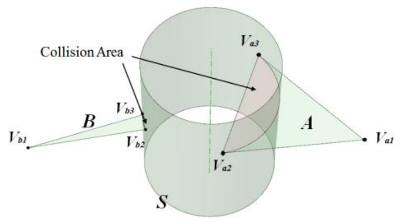


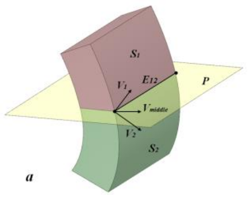
Fig. 2. The relative positions between triangles and cylinder

With the triangles collision detecting algorithm, the splitting boundary surfaces of a solid are selected without Boolean operations. In addition, the number of colliding boundary surfaces of each splitting surface is also counted as an important reference for calculating the weights of the splitting surfaces, which will be used for sorting splitting surfaces in next step.

3.2 Add Auxiliary Splitting Surfaces

The second improvement is to optimize the auxiliary splitting surfaces adding algorithm. In order to avoid the generation of irregular and low-quality decomposed solids, the improved algorithm refines the addition of auxiliary surfaces according to different radians of curved surfaces. When the radian of a curved surface is larger than 90°, one splitting surface through its boundary edges will be generated directly. When the radian is smaller than 90 degree, two splitting surfaces are generated which not only go through its boundary edges but also through the rotated axis. Thus the generation of a solid with sharp corners is avoided [4].

Furthermore, the new algorithm has also been extended to separate connected curved surfaces as shown on the example in Fig.3. In Fig.3a shows a solid containing two cylinders S_1 and S_2 that have a common straight edge E_{12}. Normally they have to be separated by a plane through E_{12}. The new algorithm calculates the normal vector V_1 and V_2 of the cylinders and generates a plane P between the two vectors and through E_{12} as a splitting surface. If two cylinders connect with an ellipse, as shown in Fig3b, a splitting plane through the ellipse is generated directly. However, if the two cylinders are connected with an edge which is a space curve, normally it is unnecessary and impossible to create a splitting surface through the space curve.



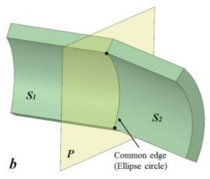


Fig. 3. Auxiliary splitting surfaces of connected cylinders

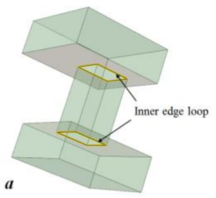
The intersections of curved surfaces are common geometric features in fusion reactor models, e.g., TF (Toroidal Field) coils and VV (Vacuum Vessel). The new algorithm analyzes the intersections of different curved surfaces, and generates different auxiliary surfaces according to the different connected edges, accordingly decomposes these models efficiently and generates more regular sub-solids.

3.3 Sort splitting surfaces

The splitting surfaces sorting algorithm employs the feature recognition technique and introduces some significant rules for calculating the weights of splitting surfaces, such as internal edge loops, and the numbers of concave edges passing through and the boundary surfaces colliding.

The first step is to find the splitting surfaces with internal edge loops. Normally a complicate CAD model is composed of several independent parts and connected with the surfaces that have internal edge loops. As shown in Fig.4a, these surfaces should be selected as prior splitting surfaces. With these surfaces, a complicate solid could be decomposed into a number of primary independent parts, and then will be decomposed further in the next steps.

In the second step, the concave edge in a solid is considered as an edge that connects two splitting boundary surfaces. So the number of concave edges that are passed through by splitting surface is another important factor for calculating the weights of splitting surfaces. If a selected splitting surface passes through more concave edges, it means that more splitting surfaces connected with these edges could be separated by this splitting surface. Thus fewer splitting processes, namely Boolean operations, are required to implement the final decomposition. In the model shown in Fig.4b, surfaces S_1, S_2, S_3, S_4 are all splitting surfaces, and E_1, E_2, E_3 are three concave edges. S_1 passes through E_1, E_2, E_3 three concave edges but S_2, S_3, S_4 pass only one, therefore, S_1 has high priority for its use as splitting surface. In effect, only one splitting process is required, otherwise more splitting processes would be necessary.



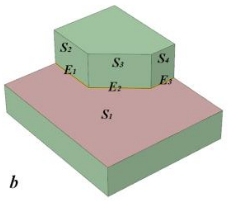


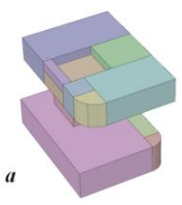
Fig. 4. Selection the priority of splitting surfaces

In the third step, and the Boolean operation between solids are essentially face-face intersections. In order to avoid potential Boolean operations errors, the sorting algorithm also records the number and types of boundary surfaces which interfere with each splitting surface. First, if a splitting surface interferes with fewer boundary surfaces and fewer curved surfaces, it will be recognized as a prior splitting surface. Second, splitting of a solid with planes has priority since it is more robust than curved surfaces.

However, although some rules are introduced for creating an optimal sequence of splitting surfaces, it is still difficult to generate a general rule which could be applied to all models, especially the models with extreme complexities. The current algorithm is applicable for most of the models with medium complexities, which are common in fusion and fission facilities and similar devices.

4. Validations

The improved decomposition algorithms and functions have been validated with example models. Fig.5a shows a simple model with the particular features such as internal loops and round corners. Fig.5b shows a representative part extracted from an ITER diagnose system model. With the new decomposition function, the models can be decomposed successfully. The volume comparison between the original and the decomposed model given in Table.1 indicates that the geometries after decomposition are consistent with the original geometries.



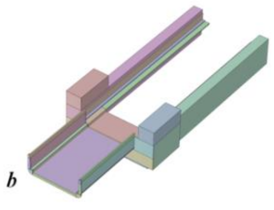


Fig. 5. Decomposition results of example models

Table 1. The volumes comparison

	Model a	Model b
Number of solids in original model	1	1
Number of solids in decomposed model	11	43
Volume of original model (mm^3)	1.58909e4	1.34349e10
Volume of decomposed model (mm^3)	1.58908e4	1.34349e10
Volume error	-6.293e-4%	1.489e-5%
CPU Time	1.68s	13.68s

Applied PC configuration:
CPU: Intel Xeon 2.4GHz
RAM: 48,000M

In addition, a generic model of a DEMO fusion power reactor, developed in the frame of the European Power Plant Physics and Technology (PPPT) programme has been employed for validation purposes. The model includes all relevant components in simplified representation. It has been decomposed and converted into a complete MCNP file. For the validation, not only the geometric shapes and volumes of solids are compared, but also the volumes of the cells of the MCNP model. Fig.6 shows the models with all components before and after decomposition, and the converted MCNP model generated with the MCNP plotter in a vertical cross section. Tables 2 compares the volumes before and after decomposition and those calculated with MCNP. There is good agreement showing that the models actually are consistent and can be used in application calculations with MCNP.

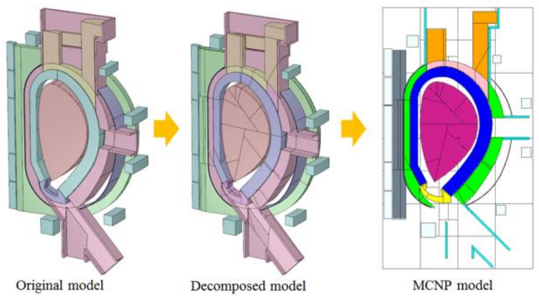


Fig. 6. Decomposition result for the EU DEMO model and MCNP model (vertical cross section drawn with the MCNP plotter)

Table 2. Volumes comparison for the EU DEMO model

Unit: mm^3	Original Model	Decomposed Model	MCNP Model
Plasma	6.6577e10	6.6577e10	6.6588e10
Blankets	5.0187e10	5.0186e10	5.0205e10
Divertor	3.4098e9	3.4098e9	3.4127e9
Vacuum Vessel	4.9632e10	4.9632e10	4.9622e10
Ports	2.2038e10	2.2038e10	2.2043e10
TF Coils & Cases	3.9370e10	3.9369e10	3.9354e10
CS Coils	2.1135e10	2.1135e10	2.1127e10
Port Plugs	1.7910e10	1.7910e10	1.7912e10
Thermal Shield	2.3043e9	2.3041e9	2.3015e9

Conclusion

A new algorithm for the decomposition of complex geometry model was designed and implemented in the McCad code. With this improvement, a weakness of the original McCad was mitigated which is due to the instability of the graphic kernel OCC and the applied original decomposition algorithm. The improvements have been validated with some representative test models and a generic model of a DEMO fusion power reactor. The results demonstrate that the new algorithm and function can be applied for the decomposition of models with medium complexity. The decomposition process is efficient and produces correct results which are consistent with the original model.

Although the current decomposition function is still not capable of processing a model with extreme complex geometry, it helps the users to improve the accuracy and efficiency of the practical modeling work. McCad's decomposition function will thus be further developed and extended in its capabilities. The improved function presented in this paper has been already integrated into McCad, verfsion 0.5.0, which is available on the source code hosting platform github: (https://github.com/inr-kit/McCad-0.5).

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed therein do not necessarily reflect those of the European Commission.

References

[1] J.F.Briesmeister, MCNP a general Monte Carlo N-particle transport code, version 4C, Report LA-13709-M, Los Alamos National Laboratory, USA, 2000.

[2] J.C.Nimal, T.Vergnaud, TRIPOLI: A general Monte Carlo code, present state and future prospects. (1990) Progress in Nuclear Energy, 24 (1-3), pp. 195-200.

[3] GEANT4 Collaboration, 2015. http://geant4.cern.ch/.

[4] L. Lu, U. Fischer, P. Pereslavtsev, Improved algorithms and advanced features of the CAD to MC conversion tool McCad, Fusion Engineering and Design, Volume 89, Issues 9–10, October 2014, Pages 1885–1888.