International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

IMPORTING CAD MODELS INTO MONK AND MCBEND

Keith Searson, Fabrice Fleurot and Andrew J Cooper
Sellafield Ltd.
H460, Hinton House
Warrington
Cheshire, WA3 6AS
keith.searson@sellafieldsites.com

Pat Cowan
The ANSWERS Software Service, Serco
A32, Winfrith Technology Centre
Dorchester
Dorset
DT2 8DH

ABSTRACT

The direct use of Computer Aided Design (CAD) models in criticality and shielding codes has been a long standing goal for Sellafield Ltd. Such functionality could offer several advantages over the traditional method of text based modelling systems. Analysts would be able to take advantage of the advanced Graphical User Interface based modelling features provided by solid modellers, potentially reducing the costs associated with creating models in a format suitable for the analyst's criticality and shielding code.
A prototype system has been developed that allows CAD models created in Autodesk Inventor or Solidworks to be used in criticality and shielding calculations. The system is based on the ANSI Initial Graphics Exchange Specification 5.3 standard and models are exported from the CAD software in Trimmed NURBS format. The format retains much more of the model's geometrical information than a format based on solid meshing techniques and avoids many of the associated problems such as large memory costs, surface approximations and void spaces. The time consuming and complex meshing process is also avoided.
Runtime intersection calculations are performed using either a Bezier clipping process for NURBS based surface definitions, or by transforming the coordinate system through which the ray tracks for Surface of Revolution calculations. NURBS surfaces are therefore converted to Bezier form as the model is imported. In addition, the SR generatrix is, in general, converted to a 'strip tree' representation, allowing the SR intersection calculations to be performed with arbitrary generatrix shapes. Details of recent improvements to the Bezier clipping process are provided. Reduction in runtime of SR based Solidworks models over equivalent NURBS based Autodesk Inventor models is also demonstrated.

Key Words: CAD, Importing, IGES, Bezier Clipping.

1. INTRODUCTION

Criticality and shielding analysts using the Monte Carlo codes MONK [1] and MCBEND [2] currently rely on text based Combinatorial Solid Geometry (CSG) methods for describing model geometry, changing the text describing the geometry, then viewing the results of the changes in

one of several tools (VISAGE and VISTA-RAY [3], Visual Workshop [4]). This process is followed iteratively until the analyst deems the text based CSG model adequately reflects the environment under analysis. Such an approach contrasts starkly to the methods now available for creating three dimensional models, in which the user interactively 'builds' the model within a Graphical User Interface (GUI). Such modelling systems offer intuitive creation and manipulation of objects, providing immediate visual feedback, functionality that is not currently available with the MONK and MCBEND visualisation tools.

In order to give analysts access to such functionality and allow them to use previously created plant models, a system has been developed that allows CAD models to be imported and used as the geometry definition for radiation transport calculations. For Monte Carlo codes such as MONK and MCBEND, this merely requires the radiation particles to be tracked through the CAD geometry to determine the likelihood and nature of any interactions between the radiation and the material present. An overview of the system is provided in sections 2, 3 and 4. Section 5 compares the runtimes of the CAD tracking system with CSG tracking for a small number of primitive solids. The system is demonstrated in section 6, where k-effective results are calculated for three criticality benchmark models.

2. COMPATIBLE CAD FORMAT

The CAD import and tracking system described within is compatible with Solidworks 2007 and Autodesk Inventor 11. Exports from either system must be in Initial Graphics Exchange Specification (IGES) Trimmed NURBS format [5]. Solidworks uses trimmed Surfaces of Revolution (SR) in addition to trimmed NURBS surfaces in this format. A list of IGES entities understood by the system are provided in Table I.

Table I. IGES entities compatible with the system

Entity Number	IGES Description	Details
408	Singular Subfigure Instance	Defines a single instance of a defined subfigure (entity type 308).
308	Subfigure Definition	Defines a single model entity, for example a transport flask that can have multiple instances via entity type 408.
144	Trimmed Parametric Surface	Trimmed surface definition, trimmed by entity type 142, Curve on a Parametric Surface. The surface can be either a Rational B-Spline entity 128 or a Surface of Revolution entity 120.
142	Curve on a Parametric Surface	Identifies a curve with a surface. The curve lies on the associated surface. The curves themselves are usually defined by a composite curve entity 102.
128	Rational B-Spline Surface	This entity contains the definition of the NURBS surface referenced by entity 144.

126	**Rational B-Spline Curve**	**These are the sub-curves defining the trimming curves via a Composite Curve entity 102. Could also be the generatrix of a Surface of Revolution entity 120.**
124	**Transformation Matrix**	**Defines any translation or rotation to any of the entity types listed.**
120	**Surface of Revolution**	**Surface of Revolution definition containing the axis of revolution and a pointer to the generatrix, which could be formed using any of the curve entities in this table.**
110	**Line**	**Defines a line using a start point and an end point. Used as part of a composite trimming curve or as a generatrix.**
104	**Conic Arc**	**Defines an ellipse, hyperbola or parabola. Can be used as the generatrix of a surface of rotation.**
102	**Composite Curve**	**Each trimming curve is usually defined using a Composite Curve entity, which further defines a collection of individual sub-curves (type 126) or lines (type 110).**
100	**Circular Arc**	**Defines a circular arc. Used as the generatrix of a Surface of Revolution.**

3. NURBS SURFACE INTERSECTION

The method of intersecting NURBS surfaces provided in this document follows that detailed in [6] and is based on the Bezier Clipping method detailed in [7]. Note that a Bezier surface is required for the Bezier Clipping process. As the problem at hand is to calculate the intersection between a ray and a NURBS surface, the NURBS surfaces are converted to Bezier form. The conversion is achieved using a knot insertion process as the model is read in as described in [8].

3.1. Bezier Clipping

The Bezier Clipping process iteratively removes portions of a Bezier patch that the ray cannot intersect until the remainder of the patch parameter space is small enough to be considered the point of intersection. The procedural steps are:

1. Define the ray as the intersection of two orthogonal planes.
2. Project the Bezier patch control points onto a plane orthogonal to the ray planes (left, figure 1).
3. If all projected control points lie on one side of the ray planes, then an intersection cannot occur and the algorithm terminates early.
4. Calculate two vectors, $\mathbf{L}_u$ and $\mathbf{L}_v$, that represent the u and v parameter directions of the Bezier patch, respectively. The vectors are approximately perpendicular to said parameter directions (right, figure 1).

The following steps are then performed iteratively, first in the u parameter direction, then in the v parameter direction. Assuming we are dealing with the u parameter direction, we have:

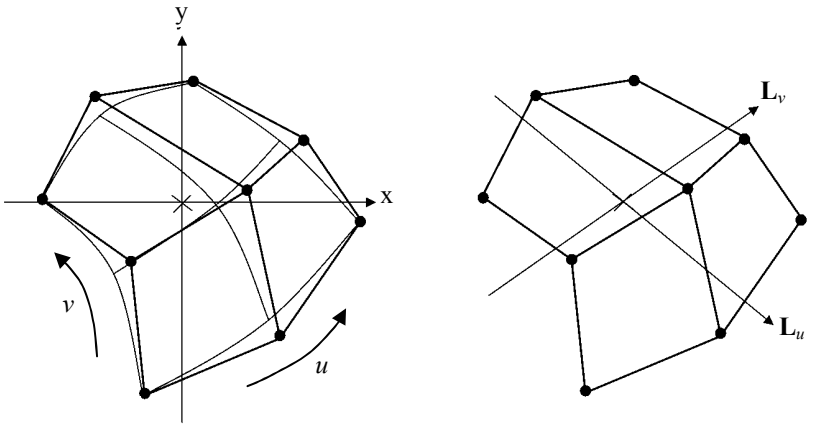

Figure 1. Left: Bezier patch under consideration with control point mesh overlaid. Ray is travelling into the page, represented by the intersection of two orthogonal planes. The planes form a local (x,y) coordinate system onto which the control points are projected. Right: $\mathbf{L}_u$ and $\mathbf{L}_v$ vectors, approximately perpendicular to u and v parameter directions of the Bezier patch.

5. The signed distances $\mathbf{d}_{i,j}$ of each of the Bezier patch control points to the direction vector $\mathbf{L}_u$ are calculated (left, figure 2).
6. Plot the points (i/n , $\mathbf{d}_{i,j}$) on a (u,d) diagram where $0 \leq i \leq n$-1 and n is the number of control points in the u parameter direction (middle, figure 2).
7. Calculate the convex hull of the plotted points and determine $u_{\min}$ and $u_{\max}$, the minimum and maximum crossing points the convex hull makes with the u axis (middle, figure 2).
8. The ray must intersect the patch between $u_{\min}$ and $u_{\max}$. Any part of the patch outside this region is 'clipped' away using the de Casteljau algorithm [9] (right, figure 2).
9. If the parameter range is reduced by less than 20%, the patch is split in half in the current direction and the algorithm is recursively applied to both halves. This enables the algorithm to deal with multiple intersections.

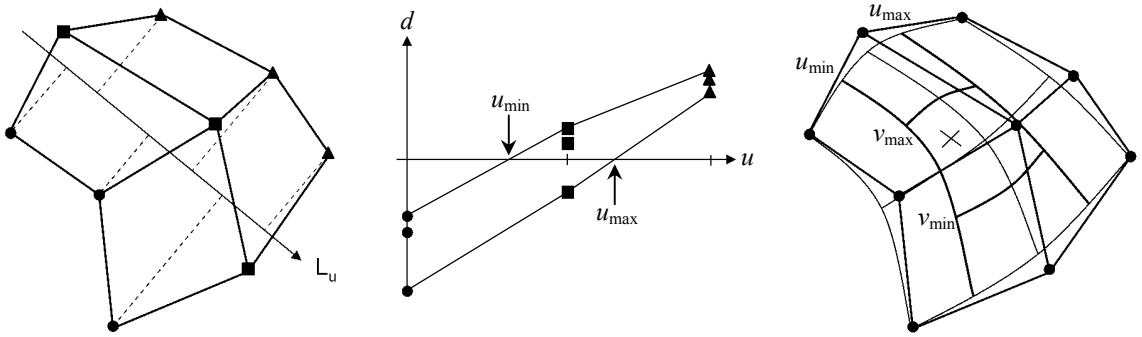

Figure 2. Left: Calculating signed distances of control points to $\mathbf{L}_u$. Middle: Signed distances plotted on (u,d) diagram with convex hull intersections at $u_{\min}$ and $u_{\max}$. Right: Portions of the patch outside $u_{\min}$, $u_{\max}$ and $v_{\min}$, $v_{\max}$ are removed.

3.2. Infinite Recursion Problem

It was found that the Bezier Clipping method can fail when the ray is travelling approximately along one of the parameter directions and the origin of the local coordinate system is within the convex hull of the projected control points. The problem is more pronounced with Bezier patches that are linear along one of the parameter directions and can be demonstrated with the aid of figure 3. Note that the ray shown in the figure misses the Bezier patch.

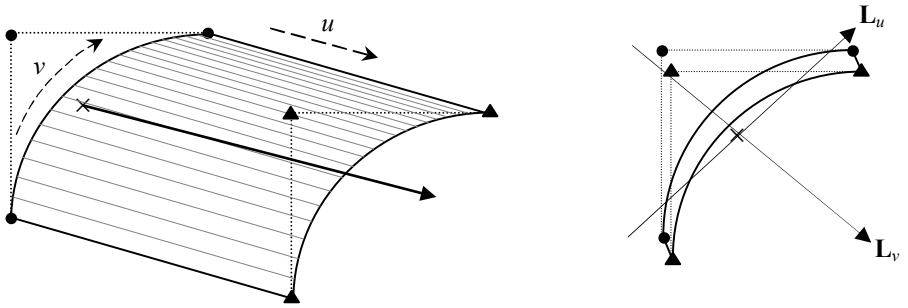

Figure 3. Left: Bezier patch of a cylinder with control point mesh, parameter directions and ray shown. Right: View along ray with projected control points shown. The $\mathbf{L}_u$ and $\mathbf{L}_v$ vectors are overlaid, approximately perpendicular to *u* and *v* parameter directions.

Considering the *u* parameter direction first, the corresponding (*u*,*d*) diagram for the situation shown in figure 3 is provided in figure 4. Note that $u_{\min}$ and $u_{\max}$ are equal to 0 and 1 respectively, and the parameter range cannot be reduced. The patch is now split in half along the current parameter direction and the algorithm applied to each half.

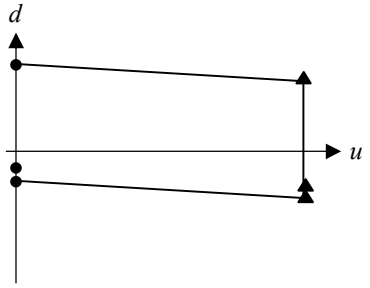

Figure 4. (*u*,*d*) diagram for ray travelling approximately along the u parameter direction within the convex hull of the projected control points.

Unfortunately, splitting the patch in such a way still gives rise to a very similar situation. The projected control points for each half of the patch look very similar to the diagram on the right of figure 3 and the (*u*,*d*) diagram results in $u_{\min}$ and $u_{\max}$ values of 0 and 1 respectively. Thus, the patch is split again along *u* and the process continues indefinitely. If limits are set on the recursion level, the problem would, for the case shown, result in a significant amount of wasted computing time. If the patch was split along the *v* parameter direction instead of the *u* parameter direction, then the problem would not have arisen and the algorithm would have quickly determined that an intersection with the patch could not occur since all of the projected control points for each half of the patch would lie on one side of one of the ray's planes and the algorithm would terminate at step 3 above.

A similar situation can arise when the ray has an intersection with the Bezier patch. If the angle of the ray in figure 3 is changed slightly so that an intersection occurs (effectively moving the

projected control points shown as triangles in the positive $\mathbf{L}_v$ direction until the ray's projected origin lies between the curves) the resulting (u,d) diagram would still look similar to figure 4. The danger now is that any recursion limit is reached and the intersection is ignored.

Essentially, when using a Bezier Clipping method for tracking purposes, it is not always appropriate to split the patch along the parameter direction under consideration when the need for splitting arises. Doing so can lead to considerable reduction in efficiency and missed intersections. In order to prevent the problem occurring, two additional variables are passed into the recursive Bezier Clipping routine. The variables, `u_split` and `v_split`, are true if the patch should be split along u and v respectively (assuming the need arises) or false if the patch should be split in a direction opposite to the current parameter direction. Both variables are initially set to true so that splitting first occurs in the current parameter direction. The difference between u_{min}, u_{max} or v_{min}, v_{max} is then monitored. If patch splitting is required (i.e. the parameter range has been reduced by less than 20%) then an additional check is made. If the parameter range has been reduced by less than 0.1%, the logical value assigned to `u_split` or `v_split` is inverted after the patch has been split. This ensures that during the next recursion, the Bezier patch will be split in the opposite direction if required.

3.3. Termination Condition

The Bezier Clipping process detailed above terminates when both Δu and Δv fall below a fixed threshold value ε, where $\Delta u = u_{max} - u_{min}$ and $\Delta v = v_{max} - v_{min}$. The termination criteria is determined in the (u,v) parameter space of the projected patch. Unfortunately, criticality and shielding models often have elements that differ dimensionally by several orders of magnitude. Assuming the parameter space is normalised in u and v, then the position in 3D (x,y,z) space for an intersection on a patch with large extents will be known less precisely than an intersection position on a patch with small extents. As an additional complication, the parameter ranges output by the CAD software when exporting in IGES format are not always normalised and often vary from surface to surface, even within a single object definition. Such variation in the known position of intersections can result in tracking errors.

The problem could be fixed by assigning a very low value to ε, but this results in unnecessary iteration in the Bezier Clipping routine and significant increases in runtime. The problem is dealt with in [6] using a method that based ε on the resolution of a virtual screen and the angle subtended by the patch as viewed from a fixed location (the paper dealt with Bezier Clipping in the context of scene visualisation). Unfortunately, when calculating intersection positions to within 10^{-6} cm from arbitrary locations within the model, a much more precise method is required.

For the CAD import system described here, each Bezier patch is assigned pre-calculated values ε_u and ε_v for each parameter. The values are determined at the CAD import stage by calculating the maximum distance, d_{max}, along the Bezier patch in both parameter spaces. In order to determine d_{max} a series of Bezier curves in each parameter direction are temporarily created based on the patch control points. The curves are then converted to strip tree format (see section 4.1), from which their lengths are easily calculated. Once the maximum length is found, it is a

trivial matter to determine the parameter based tolerances ε_u and ε_v that results in the required precision in 3D (x,y,z) space.

4. SURFACE OF REVOLUTION INTERSECTION

The methods used for the SR calculations are based on the methods in [11] and [12]. For completeness, very brief outlines of the techniques are provided next.

4.1. Strip Trees

A strip tree is a linear representation of a curve at varying levels of precision, organised in a binary tree structure. Such a representation yields various advantages in a range of applications, allowing for fast evaluation and elegant algorithms for typical 2D curve operations (e.g. displaying, intersection, length calculation). Importantly for SR intersections, strip trees can be used to represent all of the IGES curve types listed in table I, meaning that a single SR intersection routine can be developed regardless of the curve type used to represent the generatrix.

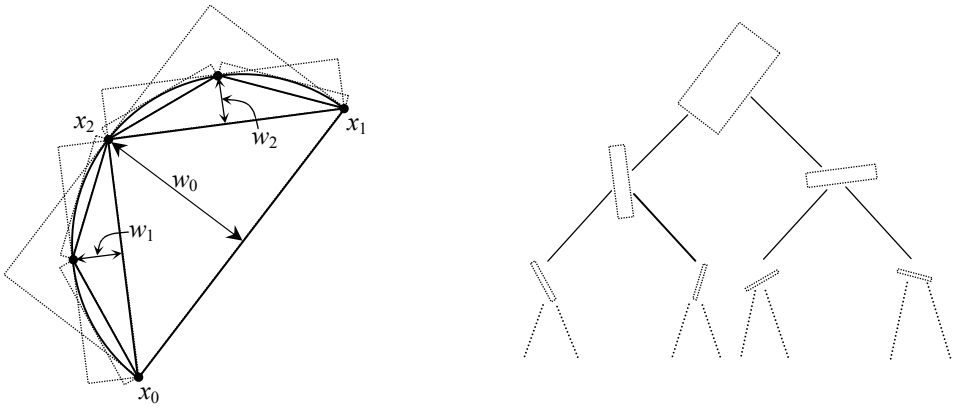

Figure 5. Left: Strip tree conversion of a curve. Right: Hierarchical representation of strips in a binary tree structure.

Figure 5 demonstrates the conversion of a circular arc (IGES entity 100) to strip tree format. The process begins by finding the maximum displacement, w_0, of the arc from a straight line drawn from the start coordinate x_0, to the end coordinate x_1. Here, w_0 is the width of the strip and is a measure of the tolerance to which the arc is represented by the straight line running from x_0 to x_1. The maximum displacement occurs at x_2 in figure 5. If w_0 is above some pre-set tolerance T, then the process continues, calculating the maximum displacements w_1 and w_2 of the arc from a line joining x_0 to x_2 and from a line joining x_2 to x_1 etc. Eventually, the curve is represented by a series of strips, arranged in a hierarchical tree structure, as shown in the diagram on the right of figure 5. For the system described in this paper, T is assigned a value of 5×10^{-7} cm.

4.2. Intersecting SR with Strip Tree Generatrix

The intersection of a SR follows two paths. If the generatrix is a line, parallel to the axis of rotation, then a simplified approach is taken that will not be detailed here. Any other generatrix is converted to strip tree format and is intersected following the method below.

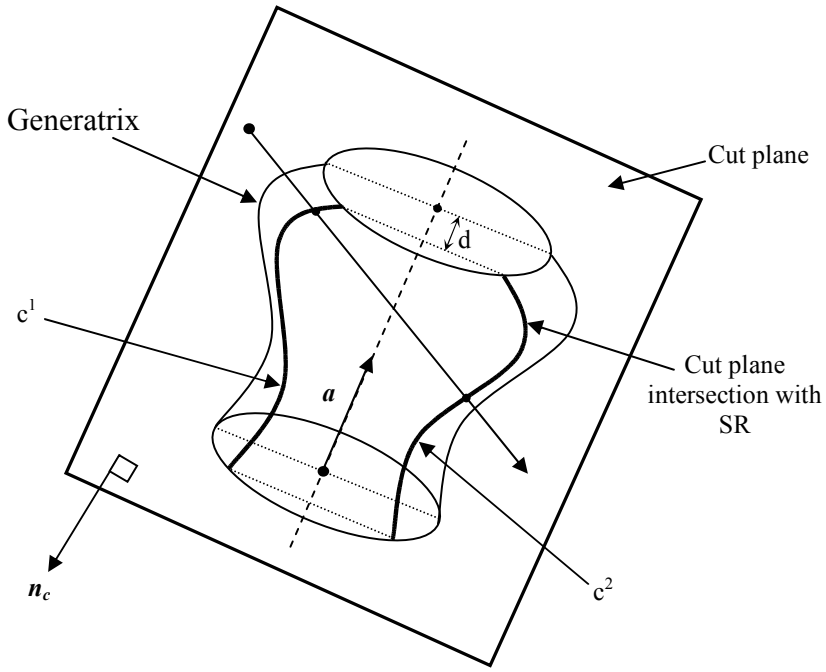

Figure 6. Surface of revolution intersected by ray travelling in cut plane.

Assuming the ray is defined as $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ where $\boldsymbol{o}$ and $\boldsymbol{d}$ are the ray's origin and normalised direction respectively, the cut plane, as shown in figure 6, is defined as a plane with normal, $\boldsymbol{n_c}$, perpendicular to both the ray's direction $\boldsymbol{d}$ and axis of rotation $\boldsymbol{a}$.

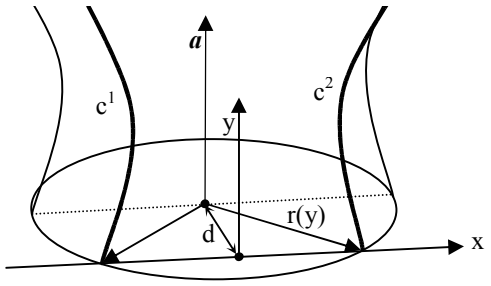

Figure 7. Local coordinate system in cut plane.

The intersection between the ray and the SR can now be defined in terms of the intersection between the ray and two curves, c^1 and c^2 in the cut plane. A local coordinate system is defined by projecting the origin of the rotation axis into the cut plane along the direction $\boldsymbol{n_c}$. The y axis direction vector is the same as the rotation axis direction vector and the x axis is $\boldsymbol{a} \times \boldsymbol{n_c}$. The arrangement is shown in figure 7. In this local coordinate system, c^1 and c^2 are given by $(-\sqrt{r(y)^2 - d^2}, y)$ and $(\sqrt{r(y)^2 - d^2}, y)$ respectively where r(y) is the radius function of the generatrix (i.e. the distance of the generatrix from the axis of rotation, measured perpendicularly from the axis). Note that these curve definitions cannot be used directly because we cannot store, in strip tree format, every possible curve that would result as the cut plane distance changes. The problem is dealt with by tracing the ray in (x^2, y) space,

where the definition of c^1 and c^2 becomes $(r(y)^2 - d^2 , y)$. The x and y components of the ray are similarly redefined and the intersection problem becomes one of solving a quadratic equation. The intersection point in 3D (x,y,z) space is found by substituting t back into the original ray definition.

5. SIMPLE PRIMITIVE TIMING COMPARISONS

Tracking times based on simple primitives created in both CAD packages are compared and the results of the comparisons are shown in table II. The timings were performed on a 3 GHz Intel dual core Xeon. The source code was compiled using Lahey Fujitsu's LF95 on Suse Linux 10.2. A total of 100,000 rays with an isotropic distribution were fired from the interior of each primitive with the elapsed time measured using the Fortran SYSTEM_CLOCK intrinsic.

The timings shown in the table were averaged over three runs and normalised with respect to the time taken to perform the same number of intersection calculations with a CSG definition of the same primitive. Note that both Inventor and Solidworks export planes in NURBS format. For the Solidworks timings shown, the planes were converted to primitive form as the model was imported. Both Solidworks and Inventor tracking methods include processing of all associated trimming curves.

Table II. Normalised timings for 100,000 ray-surface intersection calculations.

Primitive	Inventor (NURBS only)	Solidworks	CSG
Sphere	147	25	1
Cylinder	114	15	1
Box	10	4	1

6. CRITICALITY BENCHMARK TESTS

The methods detailed in sections 3 and 4 are incorporated into a test harness that includes the BINGO collision processor [13], a rudimentary input syntax for material definition and a simple scoring system for calculating k-effective. Tracking routines for a collection of simple primitives were also developed so that the criticality model could be created using the text based CSG approach for comparative purposes.

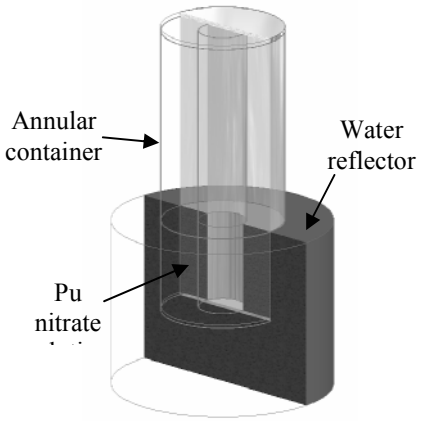

Figure 8. Benchmark model.

A criticality calculation was performed using CAD models created using Inventor's pure NURBS based output and the Solidworks hybrid NURBS/Surface of Revolution based output. The model used was based on the criticality benchmark PU-SOL-THERM-022 [14]

and consists of a plutonium nitrate solution in a water reflected annular cylinder tank with 50 cm outer and 20 cm inner diameters. An image of the model, produced by Inventor using a half-section view, is provided in figure 8. Note that the annular container was made semi-transparent in order to clarify the model geometry. The calculation results using each of the CAD models and from the traditional text based CSG definition are provided in table III.

Table III. Criticality benchmark results with normalised runtime comparison.

Model	k-effective	STDV	Runtime
Inventor	1.0256	0.00094	99.4
Solidworks	1.0243	0.00095	7.2
CSG	1.0254	0.00095	1

6.1. Additional Benchmark Modelling

Solidworks was also used to create the simplified Big Ten criticality benchmark model IEU-MET-FAST-007, as shown on the left in figure 9. The model consists of a core with average 10% ^{235}U enrichment and a large ^{238}U reflector. The benchmark k-effective value is 1.0045 with a standard deviation of 0.0007 [15]. The calculated value based on the Solidworks CAD model was 1.0002 with a standard deviation of 0.0021.

A simple spherical model, PU-MET-FAST-023, was then created in Inventor following the details provided in [16]. A quarter-section view of the model is shown on the right of figure 9 with generatrix dimensions displayed. The model consists of a graphite reflected sphere 98% ^{239}Pu with a central void. The MONK 7A k-effective result was 0.9964 with a standard deviation of 0.001 [16]. The calculated value based on the Inventor CAD model was found to be 0.9949 with a standard deviation of 0.0018.

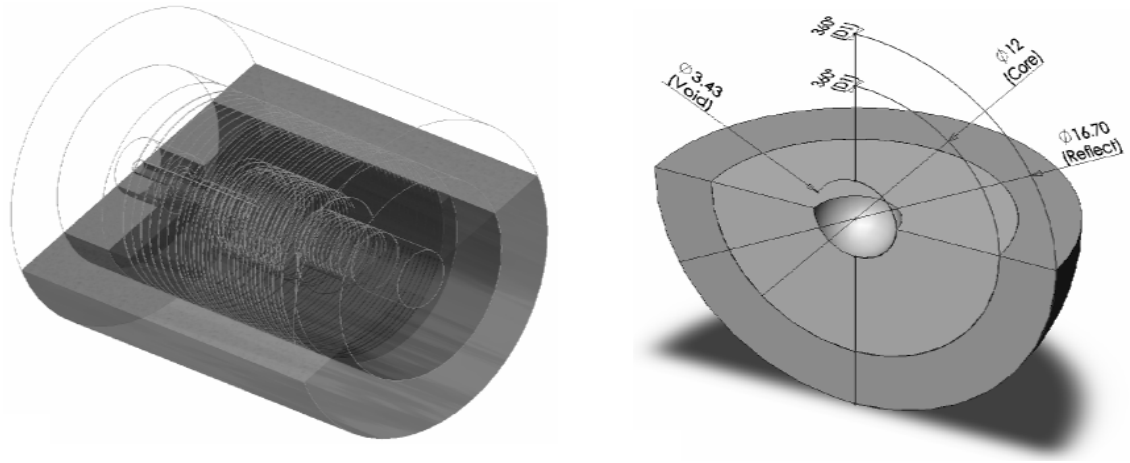

Figure 9. Left: Big Ten geometry viewed from within Inventor. Right: PU-MET-FAST-023 with dimensions displayed using Solidworks.

5. ONGOING WORK

Work is ongoing to enable the system to import and track through models in IGES Manifold Solid B-Rep Object (MSBO) format. The format only makes use of NURBS surfaces when required and will allow Inventor models to be tracked through in times similar to those found using Solidworks models. More importantly, both Inventor and Solidworks would then make use of IGES entities that will enable many elements of a typical model to be converted to CSG format, with a significant decrease in runtime.

The system has now been integrated with the MONK and MCBEND tracking routines and preliminary criticality and shielding calculations are expected soon.

6. CONCLUSIONS

An overview of the system used for importing CAD models has been presented. Details of improvements to the Bezier Clipping algorithm, necessary for the process to work when tracking from arbitrary positions in the model and to the high precision necessary for criticality and shielding calculations have been provided.

Tracking through NURBS based CAD models is found to take up to 147 times longer that equivalent models created using CSG definitions (where the models make use of the primitives listed in table II), though this increase is reduced significantly when importing Solidworks models because SR definitions are used where possible. The upper limit to the increase in runtime for such models was found to be a factor of around 25. In practice, criticality and shielding codes perform many calculations outside the tracking system (for example, collision processing and scoring) and any increase in runtime will depend on the underlying geometry of the model and the proportion of time spent outside the tracking routines. The Solidworks benchmark model PU-SOL-THERM-022 was found to increase runtime by a factor of seven when compared to an equivalent CSG model.

The system appears to be close to realising the goal of giving assessors the ability to use CAD directly in Monte Carlo codes. The factor of seven increase in runtime found for the PU-SOL-THERM-022 model is considered tolerable given the increase in efficiency of the modelling process. However, work is ongoing to significantly reduce the runtimes found when using CAD by making use of the IGES MSBO format. Information contained within the MSBO format will make it possible to convert many parts of typical criticality and shielding models into CSG elements for both Inventor and Solidworks IGES outputs.

The system also enables calculations to be performed using models containing complex NURBS surfaces, functionality that is not currently available to MONK and MCBEND users. The significant increases in runtime associated with such models may be deemed acceptable in certain situations, such as calculations involving the use of NURBS-based phantoms.

REFERENCES

1. M. J. Armishaw and A. J. Cooper, "Current Status and Future Direction of the MONK Software Package", *Proc. 8th International Conference on Nuclear Criticality Safety, St.Petersburg*, Russia, May 28 – June 1 2007, (Vol II p264)
2. P. Cowan, E. Shuttleworth, A. Bird and A.Cooper, "The Launch of MCBEND 10", *Proc. 10th International Conference on Radiation Shielding (ICRS-10) and 13th Topical Meeting on Radiation Protection and Shielding (RPS-2004)*, Funchal, Madeira Island, Portugal (May 2004)
3. M Armishaw, A Bird, H Crofts and NR Smith, "The Role of Graphical Supporting Tools for Monte Carlo Analysis", *Proc Monte Carlo 2000*, Lisbon, Portugal (October 2000)
4. D. Dewar, A. Cooper, A. Bird, P. Cowan, "Visual Workshop as the Single Model Editor for the Answers Shielding and Criticality Codes", *11th International Conference on Radiation Shielding (ICRS-11)*, Pine Mountain, Georgia, USA (2008).
5. IGES/PDES Organisation*, Initial Graphics Exchange Specification IGES 5.3*, ANSI US PRO/IPO-100-1996.
6. Robust and Numerically Stable Bezier Clipping Method for Ray Tracing NURBS Surfaces, Seidel et. al., 2005.
7. Nishita, T., Sederberg, T. W., and Kakimoto, M., "Ray Tracing Trimmed Rational Surface Patches", *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH 1990, ACM Press, 337.345.
8. Piegl, L., Tiller, W., *The NURBS Book*, 2nd Edition, Monographs in Visual Communication, Springer, 1997.
9. de Casteljau, P., *Shape Mathematics and CAD*, London: Kogan Page, 1986.
10. Efremov, A., V. Havran and H.-P. Seidel, "Robust and Numerically Stable Bezier Clipping Method for Ray Tracing NURBS Surfaces", *Proc. 21st Spring Conference on Computer Graphics*, ACM Press, New York, 127–135 (2005).
11. J. Kajiya, "New Techniques for Ray Tracing Procedurally Defined Objects", *IEEE Computer Graphics and Applications,* 2(3):161-181, July (1983).
12. D. Ballard, "Strip Trees: A Hierarchical Representation for Curves", *Communications of the ACM*, 23:310-321, May (1981).
13. S. M. Connolly, M. J. Grimstone, "The Development and Validation of a New Collision Processor for MONK"*, Proceedings of International Conference on Nuclear Criticality Safety, ICNC2003*, Tokai-mura, Japan, October (2003).
14. Leclaire, N., Poullot, G., *Plutonium (19% ^{240}Pu) Nitrate Solution in a Water-Reflected Annular Cylinder Tank (50/20 cm dia.)*, International Handbook of Evaluated Criticality Benchmark Experiments, Vol 3, NEA/NSC/DOC(95)03, September (2007).
15. Sapir, J. L., Krohn, B. J., *Big Ten: A Large, Mixed-Uranium-Metal Cylindrical Core with 10% Average ^{235}U Enrichment, Surrounded by a Thick ^{238}U Reflector*, International Handbook of Evaluated Criticality Benchmark Experiments, Vol 3, NEA/NSC/DOC(95)03, September (2007).
16. M. V. Gorbatenko, V. P. Gorelov, V. P. Yegorov, V. G. Zagrafov, A. N. Zakharov, V. I. Ilyin, M. I. Kuvshinov, A. A. Malinkin, V. I. Yuferev, *Graphite Reflected Spherical Assembly Of ^{239}Pu(δ, 98%)*, International Handbook of Evaluated Criticality Benchmark Experiments, Vol 3, NEA/NSC/DOC(95)03, September (2007).