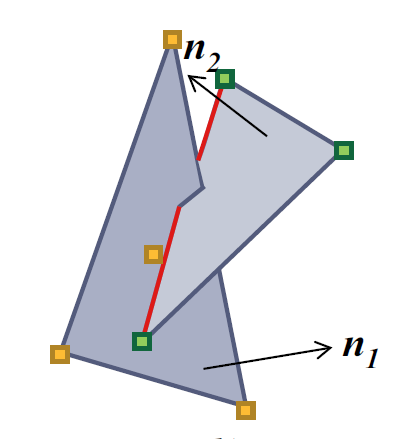
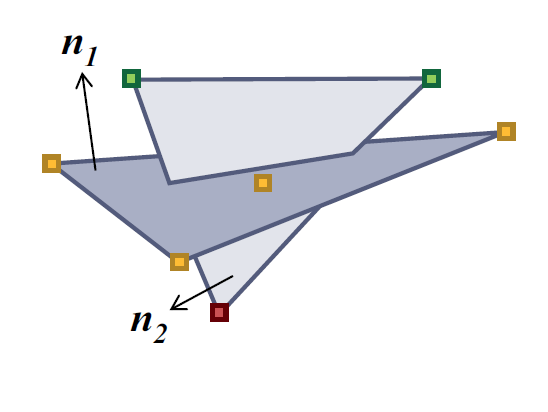
# Use of SISL Non-Uniform Rational B-Splines Library to Identify Interpenetrations

The method used by VTMS to remove surface interpenetrations removes the majority of interpenetrations but lack the accuracy to detect all interpenetrations and the methods to remove the interpenetrations were not robust. Several other techniques were also evaluated and developed to remove the surface interpenetrations within VTMS but were also not sufficiently robust. Appendix A discusses these techniques in detail. Therefore, another approach was needed to detect and fix surface interpenetrations reliably. Many references use parametric surface representations to detect interpenetration between surfaces [REFs]. One of the more popular parametric surface types is the non-uniform rational b-spline surface (NURBS). It was decided to use NURBS surfaces because of the documentation available, third-party support, and ease of implementation. The SISL library from the Department of Applied Mathematics at SINTEF ICT is a NURBS library designed for the “modeling and interrogation of curves and surfaces.” [REF] The library is used for fitting a NURBS surface to the VTMS surface geometry and the detection of the intersections between two NURBS surfaces. The library receives the faceted VTMS surfaces in the format required by the library through a process that a later section will discuss. The library then fits a NURBS surface to each VTMS tow surface and calculates all the intersections between the two surfaces. The library returns these intersections as b-spline curves. The library does not correct the surfaces to eliminate the interpenetrating regions between the tow surfaces. Also, the b-spline curves returned from the library are not always closed and may have curves that describe some or all of another b-spline curve. A later section will discuss the requirement of unique and closed intersection curves.

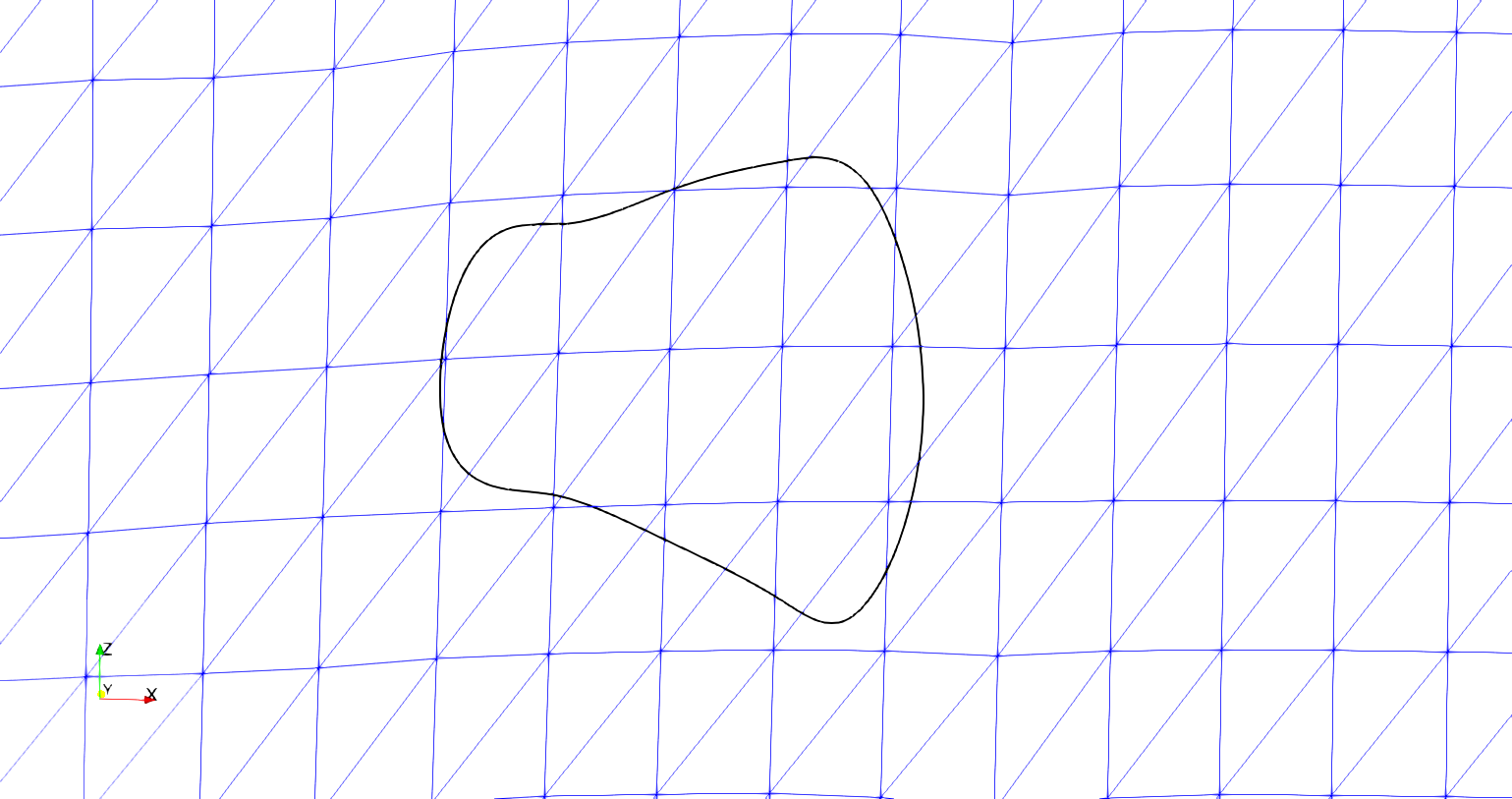
The b-spline curves obtained from the SISL library are the defined boundaries of each interpenetration region between the two tow surfaces. The main drawback of the interpenetration detection methods for the polygon surfaces is that the methods only check if a surface node is interpenetrating another surface (Figure A.a). The method used by VTMS does not check for edge-edge interpenetrations between surface mesh elements (Figure A.b) and results in incompatible meshes after correcting the interpenetrating nodes. If the surfaces are defined using NURBS, the intersection curves created by SISL outline the region (or regions) where the two surfaces interpenetrate. A definitive boundary of the interpenetration regions is more accurate and more useful than removing interpenetrations on a point-by-point basis. The intersection curves, which are also the boundaries for the interpenetration regions, are used to remove any surface mesh elements that lie within the perimeter of the intersection curves. Figure B shows two scenarios between an intersection curve and surface elements. Elements, such as those marked with an **o,** that lie entirely inside the curve are removed. Elements marked with an **x** are divided where the intersection curve intersects that element so that the resulting submesh elements are entirely inside or outside the intersection curve perimeter.

**Figure A: Two cases of interpenetrations between polygon surfaces**



1. **Vertex inside element (node interpenetration)**
2. **Edge inside element (edge-edge interpenetration)**

Figure B is only considering one surface mesh. However, both surfaces have their interpenetrating elements removed using the same intersection curve. Removing elements from both surfaces using the same curve ensures that neither surface has elements inside of the interpenetration region bounded by the intersection curve. A later section discusses the process of removing the elements from both surface meshes in a systematic manner. Once both surface meshes have their interpenetrating elements removed, a new mesh is created using the intersection curve as the boundary. The new mesh is inserted into both surfaces to replace the deleted interpenetrating elements. Now, the elements for both surfaces match perfectly inside of the intersection curve, resulting in a perfectly compatible mesh within the curve.



x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

o

o

o

o

o

o

o

x

x

x

x

x

**Figure B: Surface intersection curve on a tow surface**

## Pre-SISL data formats and SISL required inputs

A NURBS surface is a type of parametric surface and uses a curvilinear coordinate system to describe a three-dimensional surface. Hence there are only two curvilinear coordinates. For Cartesian coordinates, each of the three coordinates is found using a basis function, as shown in equations 1-3. Equation 4 is the equation for the position vector of any point using curvilinear coordinates and the basis functions.

(1)

(2)

(3)

, (4)

Figure C further shows that cartesian coordinates for point ***p*** are related to the curvilinear coordinates via the basis function . The equation or set of equations that define depend on the type of parametric representation used.



**u**

**v**

**x**

**y**

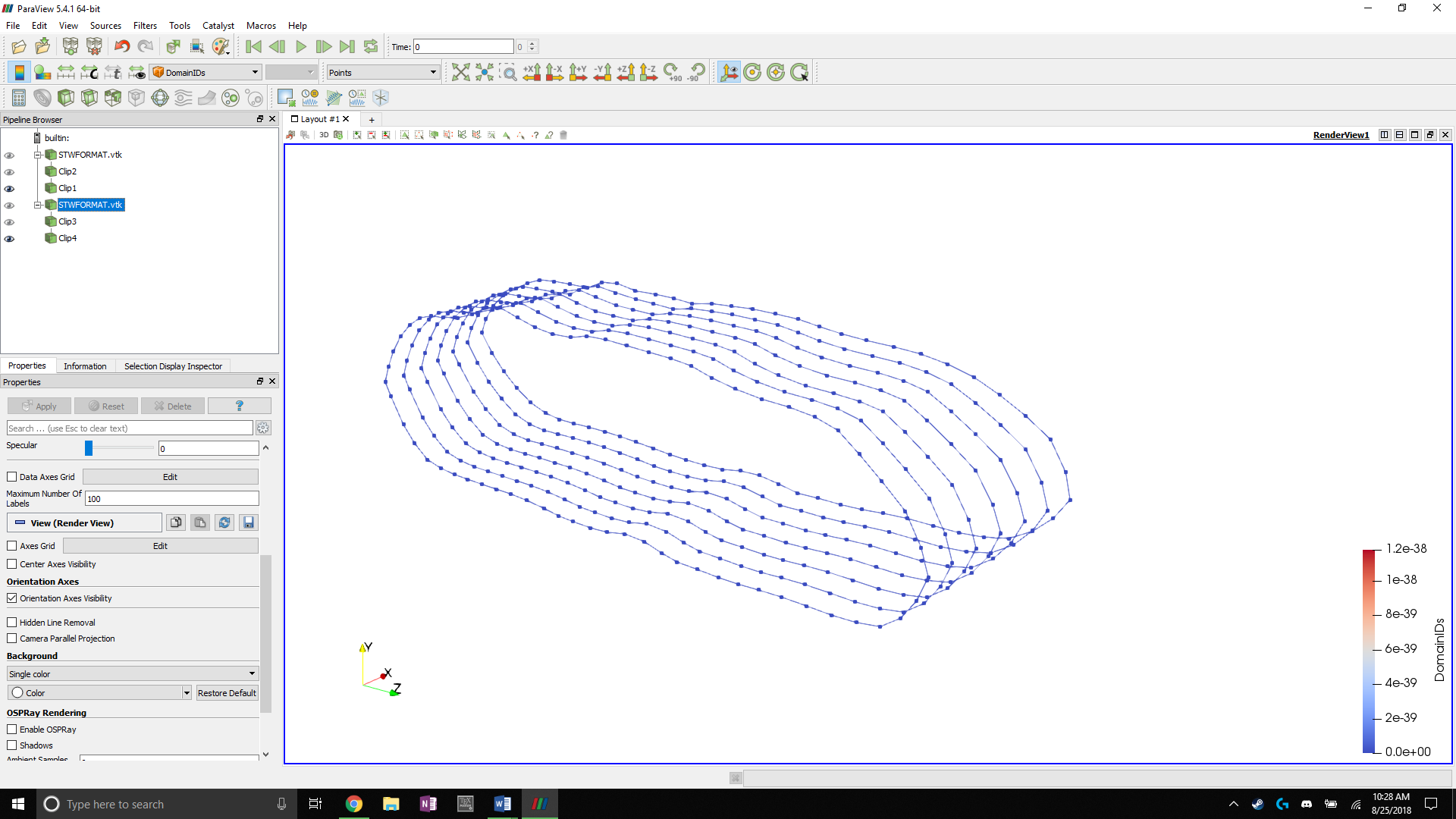
**z**

**Figure C: Relationship between curvilinear coordinate system and cartesian coordinate system**

***p***

The equations that define a NURBS surface basis function are, as previously discussed, complex. SISL handles the creation and implementation of the complex NURBS basis functions so that the user only needs to provide adequate surface data to use the library’s full functionality. SISL requires a list of points that become the control points of the NURBS surface and the number of control points in the two curvilinear directions. The first step in using the SISL library is to create control points using the VTMS surface data that is exported by VTMS and format them according to SISL requirements.

VTMS’ standard tow format (exported as the .stw file type) describes the surface as a series of polygonal cross sections where each cross-section is made up of the same number of points. These cross sections are perpendicular to the path of the tow and all points of a cross-section lie in the same plane. Each point belongs to a specific cross-section. For example, in figure D, the points numbered **1-5** all belong to cross-section ***i****.* The lines that outline the cross sections have been added to clarify which points belong to each cross-section. The points for each cross-section are numbered circumferentially in the same orientation shown for cross-section ***i***. VTMS stores all the points for cross-section ***i***in order, followed by all points in cross-section ***i + 1***, then the points in cross-section ***i + 2***, continuing until all cross-section points are saved.



**Figure D: Tow surface cross sections with starting points of each stack marked**

**1**

**2**

**3**

**4**

**5**

***i***

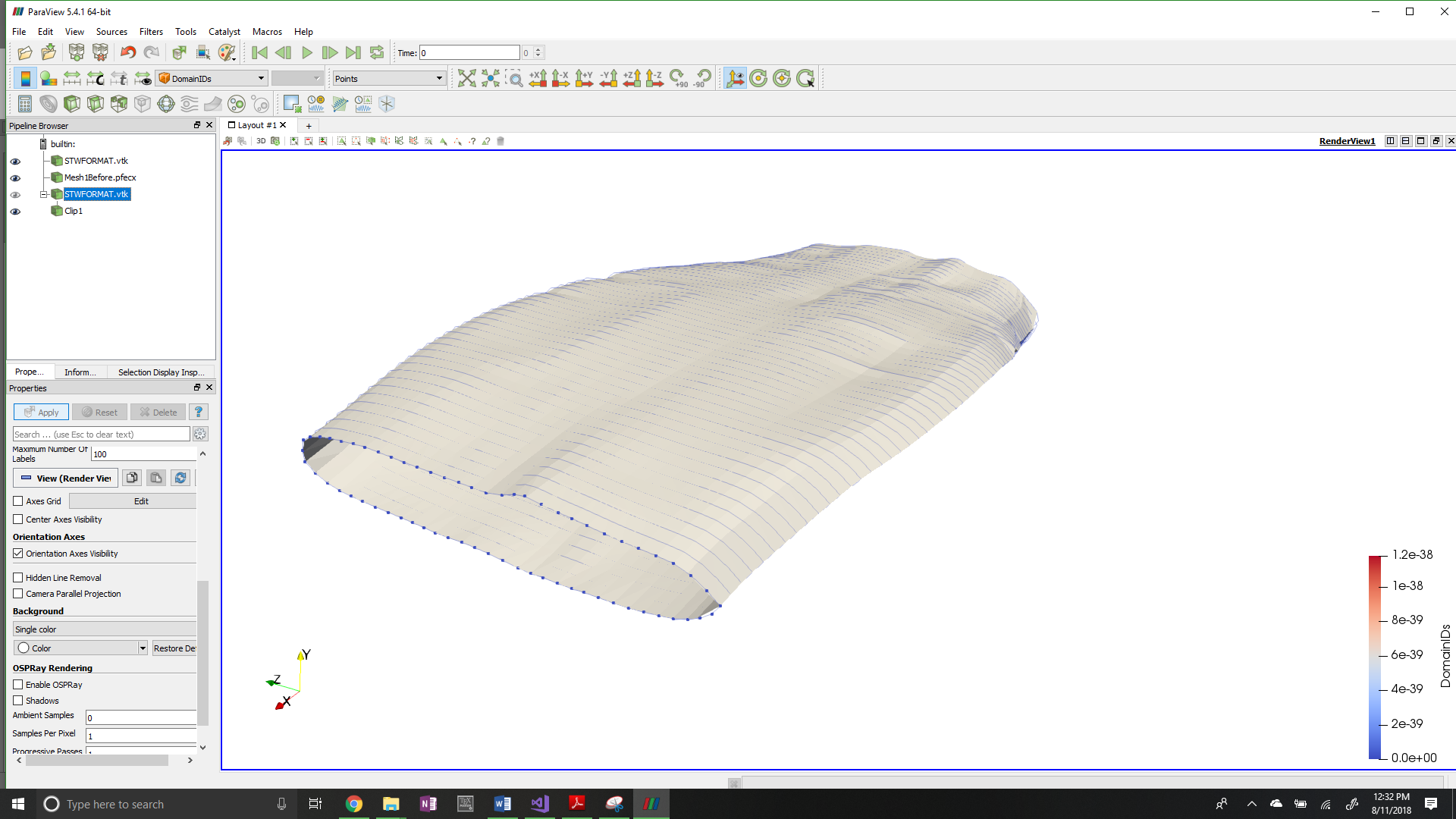
***i + 1***

***i + 2***

***i + 3***

VTMS’ systematic method of storing its surface points almost directly matches the format that SISL requires. SISL requires a set of all surface control point coordinates in a specific order of the form

where is the number of tow surface cross-sections and is the number of points per cross-section. The indexes and refer to the point on cross-section. Each point’s coordinates are listed idividually, resulting in a set that has number of values. SISL also requires the number of cross-sections and the number of points per cross-section which are used to define the range of the curvilinear coordinates.

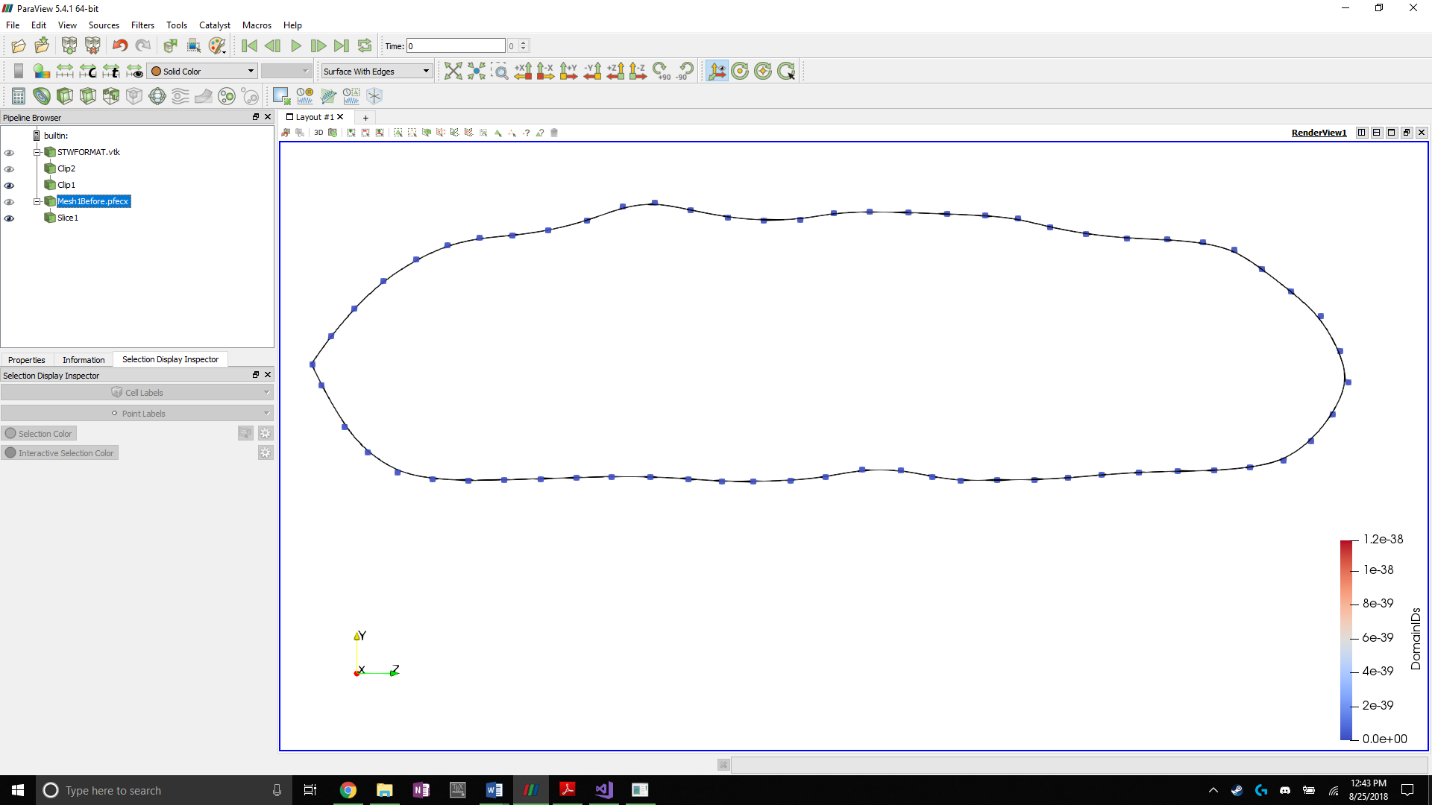
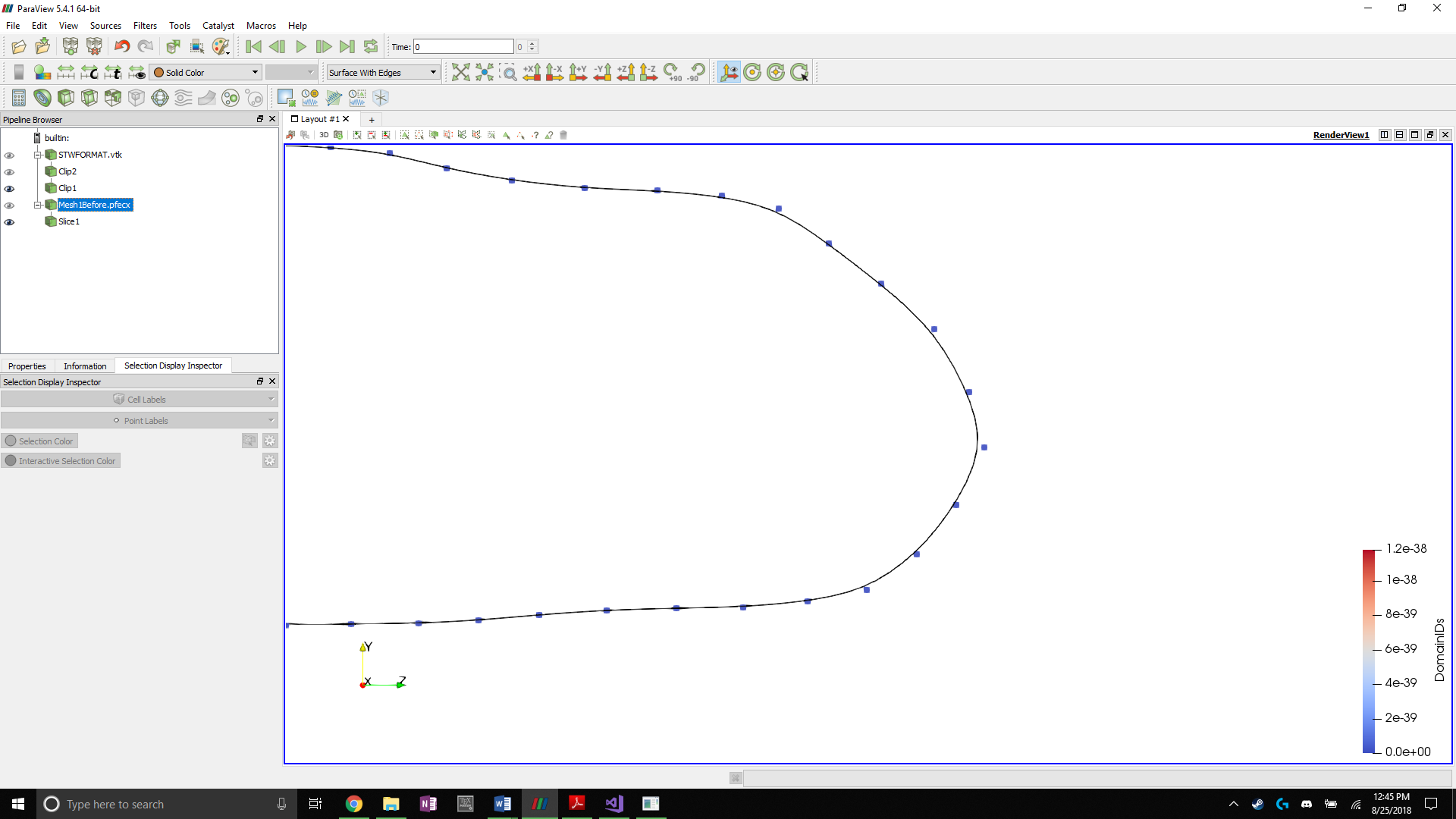


***u***

***v***

**Figure E: Curvilinear coordinate system for tow input data to the SISL library**

SISL uses the control point coordinates to fit a NURBS surface to the VTMS surface data and defines the curvilinear coordinate system shown in figure E. It is important to note that the resulting surface approximates the VTMS data. The curves that make up the NURBS surface are not required to run through the original VTMS surface points because now the original points are the control points for the NURBS surface. In general, control points rarely lie on the b-spline curve it defines. Figure F shows the original VTMS points of a single cross-section compared to the resulting NURBS curve. The figure shows that in the region where the surface curvature rapidly changes, the points that are used to control the surface do not lie on the surface. When the surface’s curvature is more constant, the points lie close to or on the surface. By observation, the SISL library captures nearly all of the tow volume as well as keeps many of the topological features (peaks and valleys of the surface) that are important for detecting surface interpenetrations. Therefore, the NURBS surface approximations are an accurate approximation of the VTMS surfaces and will result in accurate detection of the interpenetration regions.



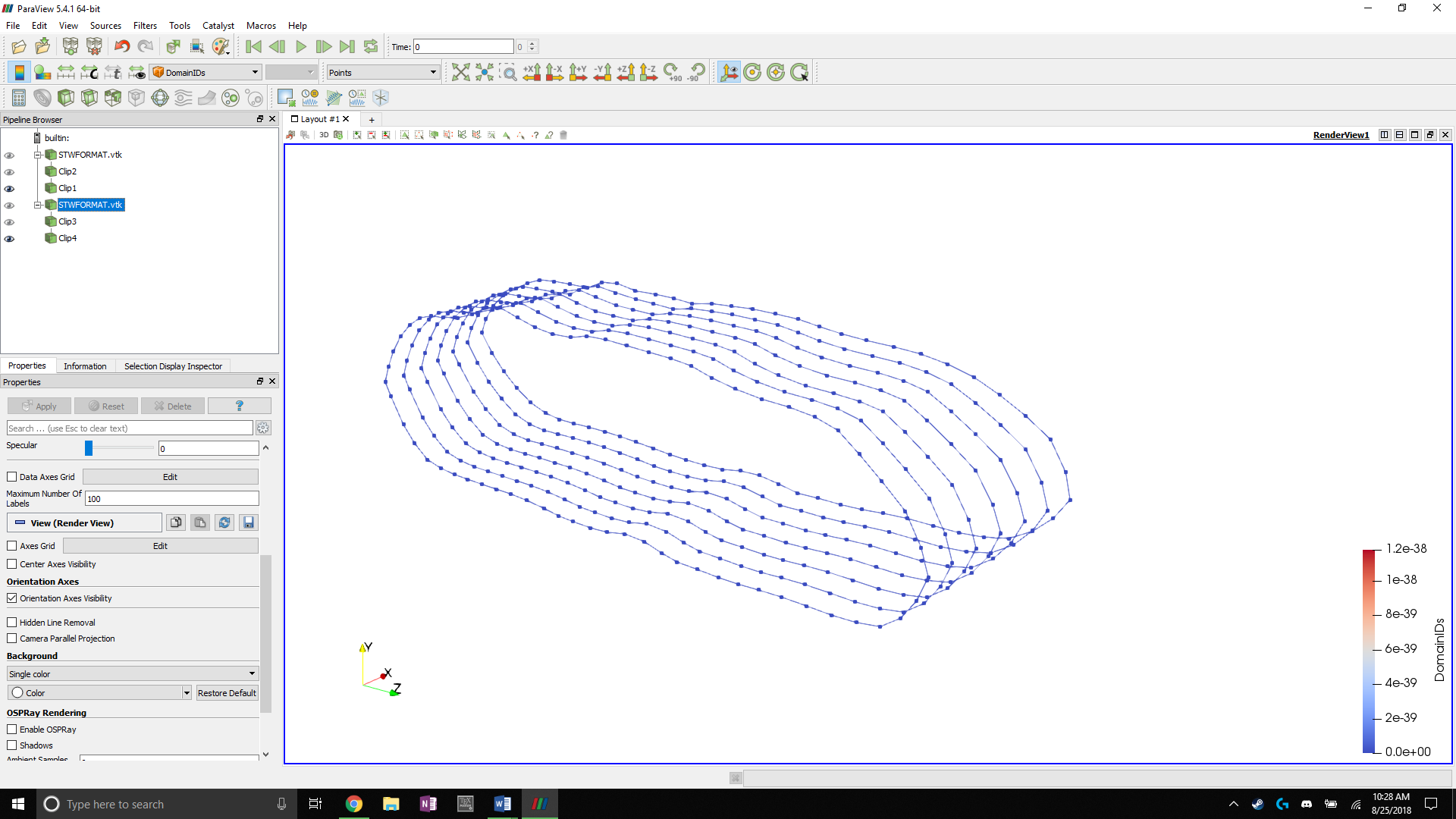
**Figure F: NURBS approximation of tow cross section with original VTMS data as control points**

Once the NURBS surfaces have been created by the SISL library, the surfaces are then used by the library to detect the interpenetration regions and return the intersection curves that outline the regions where two surfaces interpenetrate. The terms intersection curve and boundary curve can be used interchangeably. The curve that bounds the interpenetration region is the same curve that traces where the two surfaces intersect and cross into each other. The library returns these interpenetration boundary curves as b-splines. The next section discusses the linear approximation techniques used to visualize the b-spline curves and NURBS surfaces.

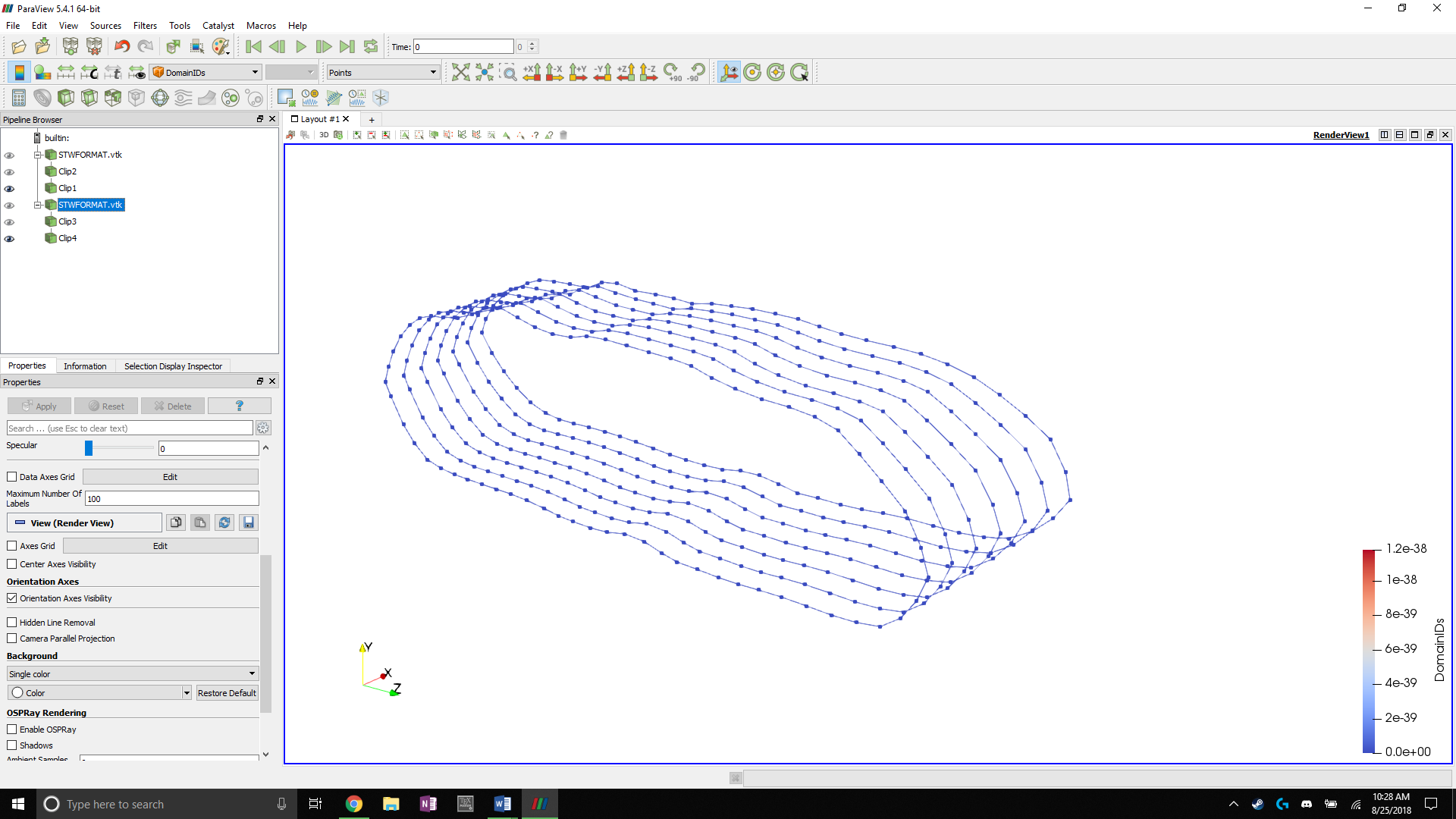
## NURBS visualization techniques

One drawback of using NURBS and B-Splines to describe a surface is visualizing the surfaces and curves. It is important to visualize the surfaces and intersection curves so that the issue of interpenetration can be fully examined. Without a proper understanding of the problem, it is incredibly difficult to develop a solution. Most visualization software suites use polygons to approximate the surface of complex shapes to render them for viewing. SISL comes with a visualization tool, but the tool cannot add or remove sections of the surfaces and curves that help to understand the interpenetration regions. Therefore, we selected Paraview as the primary visualization software. Paraview has a large amount of functionality such as surface clipping, data filtering, and other functions that help to understand interpenetrations. However, in order to use Paraview, the NURBS surfaces and b-spline curves must be discretized. After discretization, the surfaces and curves need to be systematically stored for efficient and effective use. BetaMesh is used because it allows for meshes of complex shapes to stored systematically. BetaMesh also efficient methods for quickly finding surface elements, removing duplicate mesh nodes and elements, and other methods that help to accomplish the task at hand.

The surfaces (or curves) are discretized using a surface (or curve) sampling function in the SISL library. The function returns the cartesian coordinates of the surface at a chosen set of curvilinear coordinates. The number of times the surface is sampled between the maximum and minimum curvilinear coordinate values in each direction controls the refinement of the discretized surface. The same level of refinement as the original VTMS surface data is chosen because the refinement is a good compromise between accuracy and efficiency. The surface is sampled by evenly dividing the curvilinear coordinate space in the direction by the original number of cross-sections and the direction by the original number of points per cross-section. Once the curvilinear coordinate space has been evenly divided, the surface is sampled at curvilinear coordinate pairs and the physical three-dimensional coordinates of the surface are returned. The sampling function is not creating new points on the surface. Instead, the SISL sampling function is simply converting the curvilinear coordinates to physical coordinates using the defined SISL basis functions discussed earlier. The coordinates returned from the surface sampling function are stored in a format similar to the VTMS format because it is an efficient and deliberate way of collecting coordinates. A systematic approach to collecting the coordinates is important when forming the surface elements that are used for the visualization. Once the surface is sampled, according to the refinement chosen, a method creates the surface mesh using the collection of coordinates.



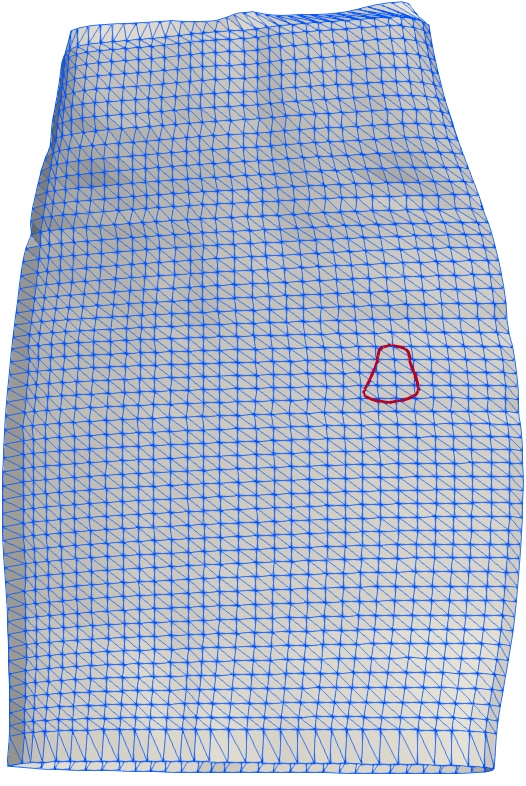
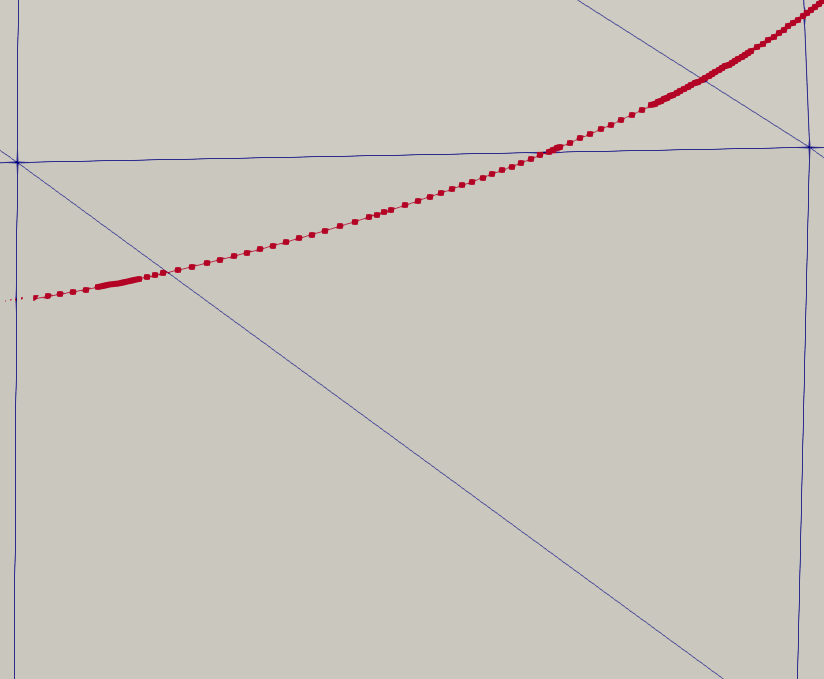
**Figure G: Cross-section polygons with connecting surface elements**



The creation of the surface mesh is simple because of the systematic ordering of the surface points. In figure D, the nodes labeled , , and are all the node on their corresponding cross-sections. Every node can be connected to the node on the proceeding cross-section because the numbering scheme is the same for each cross-section. This consistency is ensured because of the systematic way the points on the surface were collected. Then, the and the node on each cross-section can be connected to create quadrilateral elements formed by nodes , , **,** and . Two opposite corners of the quadrilateral are connected to create two triangles, as shown in figure G. These triangles are stored as triangular elements. These elements comprise to a surface mesh that is exported by a BetaMesh function to file format that are used in Paraview.

## Intersection curve visualization techniques

The SISL library also returns the intersection curves as b-splines, which must be discretized to be visualized using Paraview. The same function that samples the NURBS surface is used to translate a curvilinear coordinate into the physical space. The b-spline curves only have one coordinate, compared to the NURBS surface which has two coordinates. Therefore, discretizing a b-spline curve is easier than a surface. As before, the range of the curvilinear coordinate space is evenly divided. The curve is then sampled at a high frequency so that the meshed curve closely matches the b-spline curve. Once the physical coordinates of the points on the b-spline curve have been collected, all the points are connected by line elements to create a linear piecewise approximation of the b-spline curve. Figure H shows an interpenetration boundary curve and an example of the refinement level of the boundary curve. Figure H illustrates the point that the intersection curve approximation is at a much higher refinement than the surface approximation.

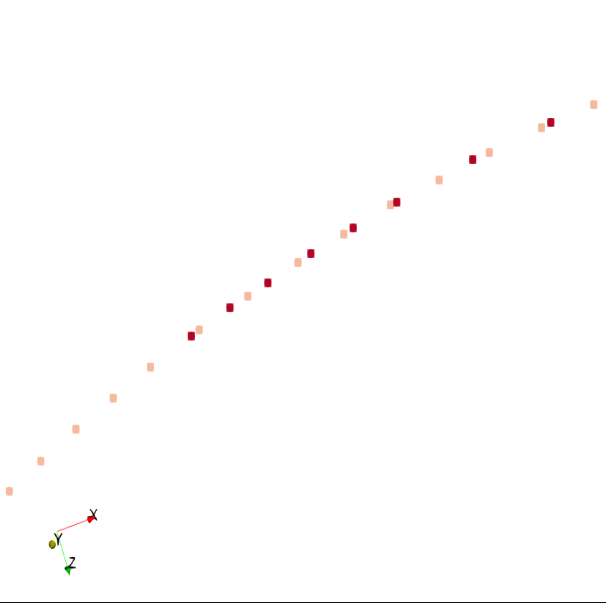
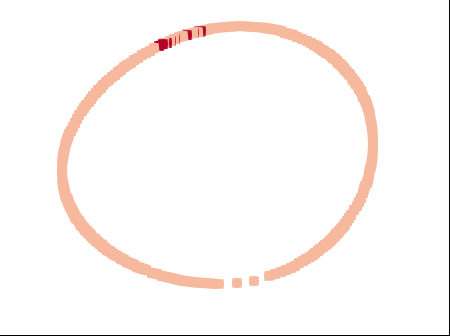


**Figure H: Surface mesh of NURBS surface with boundary curve and relative refinement of curve**

The surfaces shown in Paraview are approximations of the NURBS surface due to the discretization process that was used to visualize them. However, the points that define the surface elements were directly sampled from the surface to reduce the error in the approximation. If the refinement of the sampling is increased, the relative error between the approximation and the actual surface is reduced. The same relationship applies to the intersection curve. The higher refinement of the interpenetration boundary curve relative to the surface ensures that all interpenetrating nodes from the surfaces lie inside the intersection curve. The approximation of the NURBS surfaces as discretized meshes and the linear approximations of the intersection curves will be used to remove the interpenetrating regions of the surface meshes.

## Consolidation of intersection curve approximations

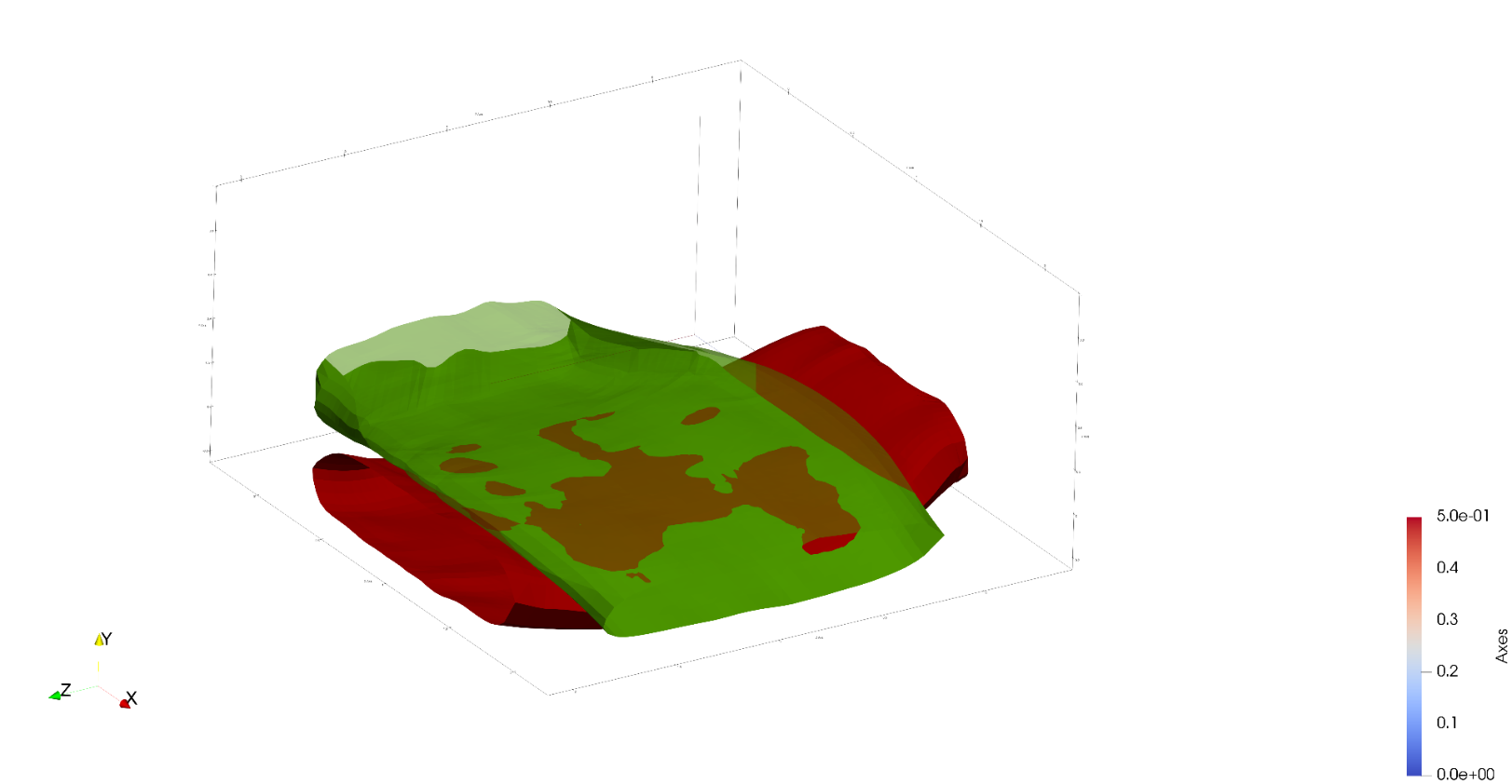
When considering the region where two surfaces intersect, it is not safe to assume that the intersection curve between the surfaces is closed. However, if the surfaces are closed in the region where they intersect, it is a safe assumption that the intersection curve between the surfaces is also closed. Furthermore, there can only be one unique intersection curve for each intersection between surfaces.



**Figure I: Non-unique meshed boundary curves with duplicated curve data**

The main drawback of using the SISL library is that the intersection curves returned from the methods are not guaranteed to be unique or closed. Therefore, the meshed curves need to be corrected before they are used to remove interpenetration elements from the surface mesh. Figure I shows an example of two surface intersection curves that were returned by SISL and converted into a line element mesh. Only the nodes that make up the curves are shown for clarity. They are described as non-unique because the smaller, open curve (dark in color) duplicates a region of the larger, closed curve (light in color).

There are multiple scenarios where two curves either overlap and duplicate data, or connect to create a closed boundary curve. It is assumed that if two surfaces interpenetrate then any interpenetration region can be described by a closed curve. Assuming a closed intersection curve is valid because if two surfaces interpenetrate then one surface should both enter and exit the opposing surface, creating a closed region of interpenetration that is bounded by a closed intersection curve. Figure J shows two surfaces interpenetrating. Using the assumption of a closed intersection curve, the intersection curves from SISL are required to be unique and closed.



**Figure J: Two tow surfaces with interpenetrating regions**

The first step is to identify which surface intersection curves are closed and which are open. When approximating the b-spline intersection curves, line elements were created between every pair of adjacent nodes sampled from the b-spline curves and saved in a list that records all of the line elements in no particular order. During this step, a map is created for each node that lists every element that is connected to the node. For a closed curve, every node will have two line segments connected to it except the first and last nodes sampled from the curve. If the curve is closed, the first and last point should be the same. However, the beginning and end points being the same is not guaranteed by SISL. A check could be made to verify if the beginning and end nodes are the same, but this would still not verify if there are duplicates of the same curve. Therefore, a systematic method of assembling the curves is required to ensure that the linear approximations of the b-splines are unique and closed. The mapping between the nodes and the nodes connected line elements is vital when assembling the curves. Each intersection curve is made of hundreds or even thousands of line segments. The curves are pieced together to ensure that opened or closed nature of the curve is known. While the curves are put together, the segments are tested for any overlaps or duplications of the same curve data. The method begins by finding a node that is connected to only one segment using the node-element map. A node that is only connected to one segment signifies either the beginning or end of a curve, both of which are ideal starting points for piecing together the curve approximations from line segments. The directionality with which the curves are pieced together (beginning to end or end to beginning) does not matter. However, it is easier to start at one end of a curve than to piece the curve together starting in the middle of the curve. The starting node and singular connected segment are added to a mesh that will describe (or store?) the linear approximation of the b-spline curve. Then, using the map between the nodes and connected elements, the remaining node on the line segment checks if there is another segment connected to that node. If so, the segment is added to the mesh of the linear approximation. Each time a segment is added, the node that is not connected to the previously added segment is checked against the first node in the mesh to see if the curve has connected back to itself, creating a closed curve. If the nodes are not the same, then the node not connected to the previously added segment checks whether it has a second segment connected to it using the node-element map. If so, then the segment is added, and the method continues. If the node does not have a second segment, and the node is not the same as the first, then the curve is known to be open. Each time the method determines that the curve is closed or open, the mesh is label as closed or open, saved, and the method begins again with a new single element node. The method continues until all the segments have been added to meshes. The result is a curve mesh list containing both open and closed boundary curve meshes.

A method compiles all of the curve meshes into a list and performs a dual-loop iteration to determine if any of the curves are duplicates. These loops choose the first closed curve (denoted curve **A**) in the list and compare the remaining curves against it. If the curve being compared (curve **B**) overlaps curve **A**, whether closed or open, it is removed from the list. Overlaps are determined by an algorithm that chooses both the start and end node of curve **B** and iterates over all the line elements of curve **A**. The overlap method uses the two endpoints (nodes **a** and **b**) of the current line element from curve **A** and the node to be checked (node **c**) from curve **B**. A vector is created to connect **a** and **b**, as well as a vector from **a** to **c**. The cross product between these two vectors is calculated, and if the cross-product is below a certain tolerance, node **c** is determined to lie on the line that goes through **a** and **b**. Figure K.a shows an example where node **c** is much closer to the line segment than **c’** and the cross-product between vectors **ab** and **ac** is much smaller than that of **ab** and **ac’.** If **ab** x **ac** is sufficiently small (less than a user-defined tolerance), the node **c** is said to overlap the line connecting nodes **a** and **b**. The dot product is then calculated between the two vectors. If the result is greater than zero but less than one, shown by **ab ● ac** in figure K, it is known that point **c** lies between the points **a** and **b**. If the dot product is less than zero (shown by **ab ● ac’**)then the point does not lie between **a** and **b**.These two checks verify that node **c** is on the line segment formed by **a** and **b**. If both checks are pass, then there is an overlap and the curve which node **c** belongs to is removed. If the test does not result in an overlap for the beginning or end node of the curve, the curve mesh is kept. The result of this method is a set of unique, closed curve meshes that are used to identify which surface elements need to be modified and removed to eliminate interpenetrations between surfaces.

**a**

**b**

**c**

**c’**

**ab x ac’**

**ab x ac**

**a**

**b**

**c**

**c’**

**ab ● ac’**

**ab ● ac**

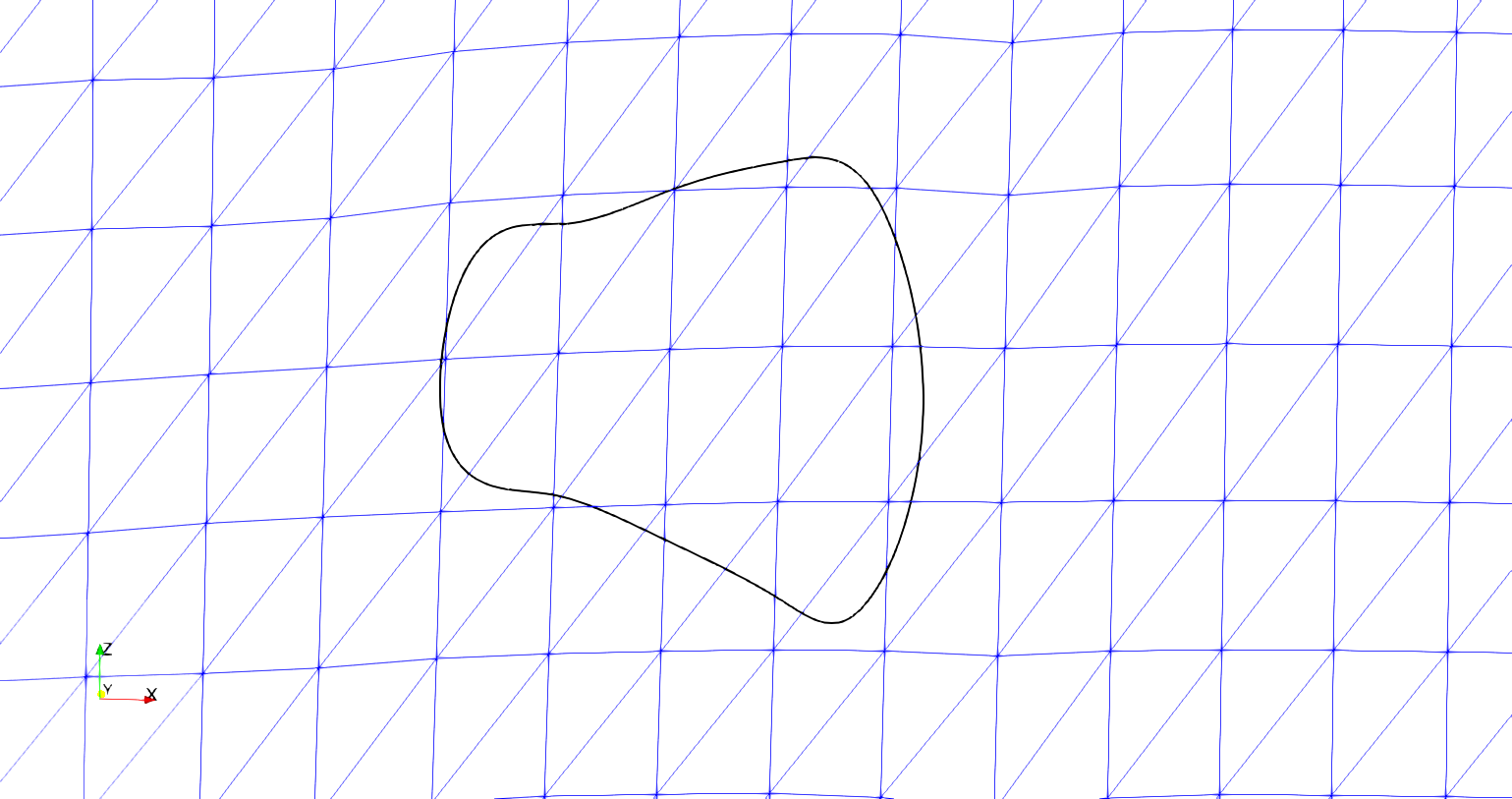
**Figure K: Cross-products (a) and dot products (b) of vectors between nodes**

**a)**

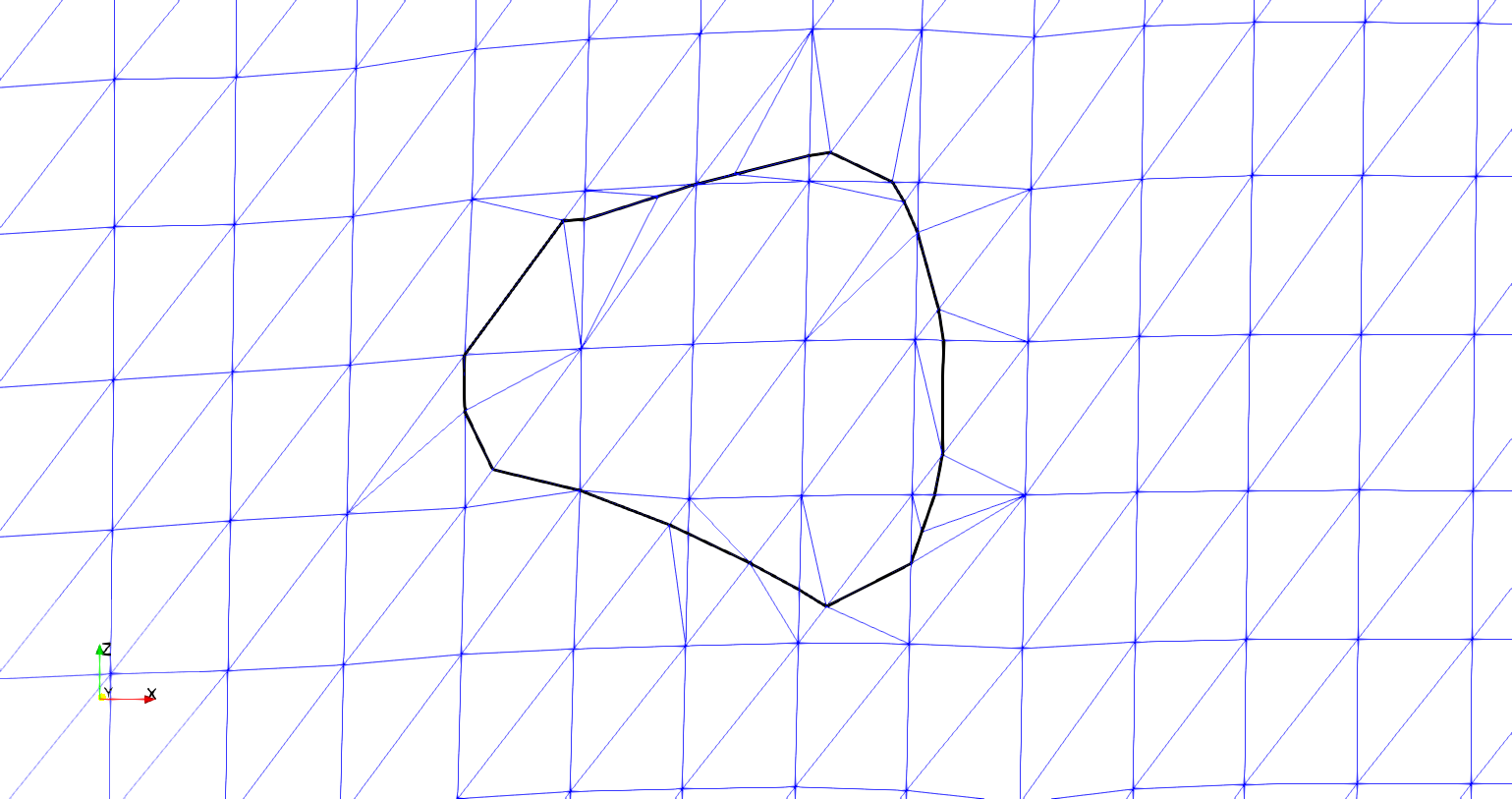
**b)**

## Detecting and removing surface elements using a linearly approximated boundary curve

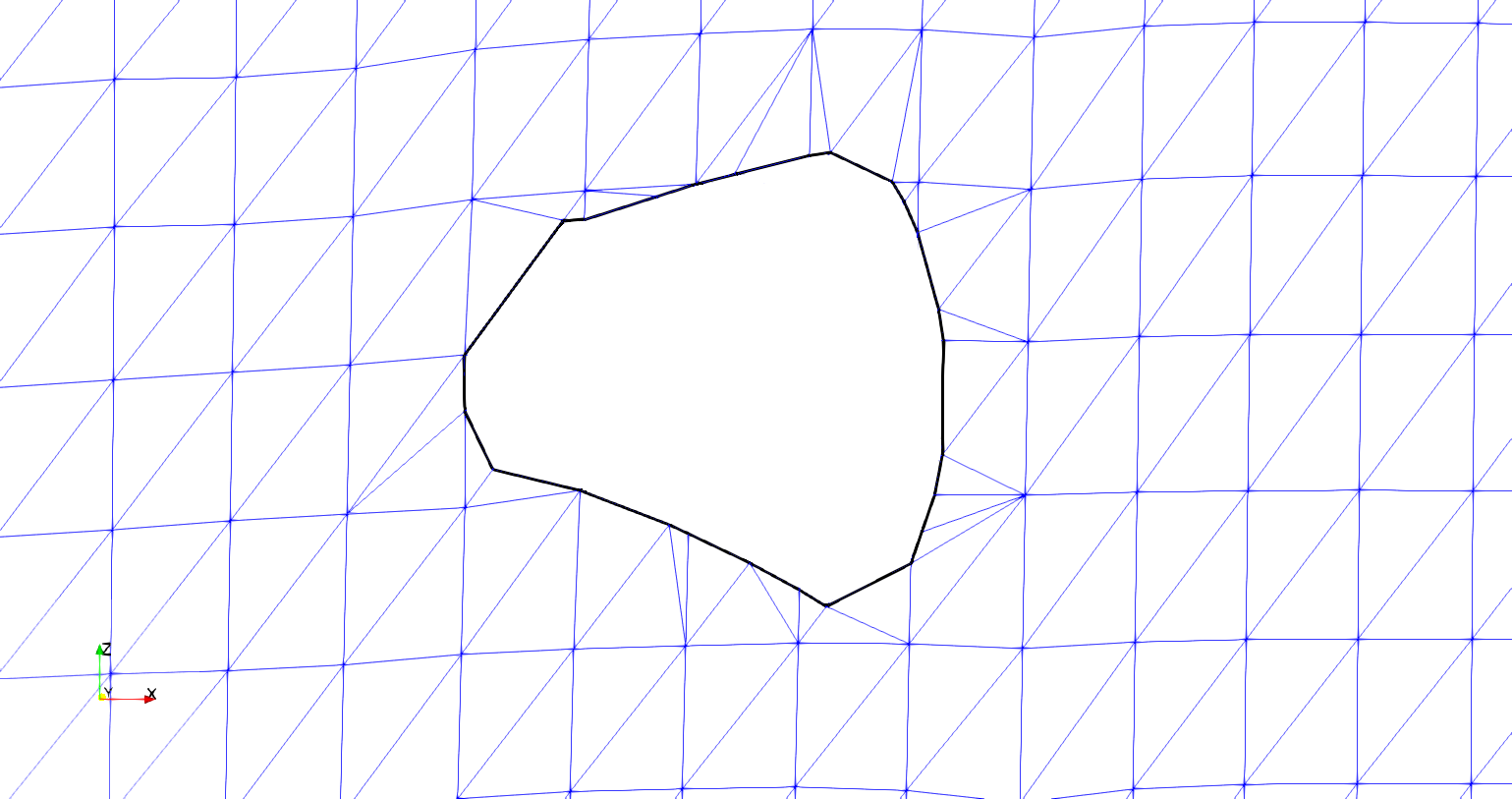
Once the surface intersection curves have been made unique and closed, the curves are used to modify the tow surface elements they intersect. First, the method used to remove the interpenetrating elements is presented using an intersection curve and one surface. In reality, the method is performed on both surfaces to remove the interpenetrating elements on each surface. However, discussing the method using only a single surface first allows for a clear description of each step of the method. Once the method has been described completely, a later discussion will show how the method is adapted to incorporate both surfaces. Figure L shows an example of an intersection curve that is used to remove the interpenetrating elements. Figure L.a show the initial linear approximation of the intersection curve overlaid on a section of the example surface mesh. Figure L.b shows a further approximation of the intersection curve when only the intersection points between the curve and the surface elements are used to define the intersection curve. This curve does not represent the original curve very well, but a later discussion will discuss how the fit is expected to be improved once a second surface is included in the method. Figure L.b also shows how the surface mesh is modified so that no surface elements lie partially in the area enclosed by the intersection curve. Elements that lie partially inside the intersection curve must be sub-divided into smaller elements so that the region inside of the intersection curve can be removed. Figure L.c shows the surface mesh after the interpenetrating elements have been removed.



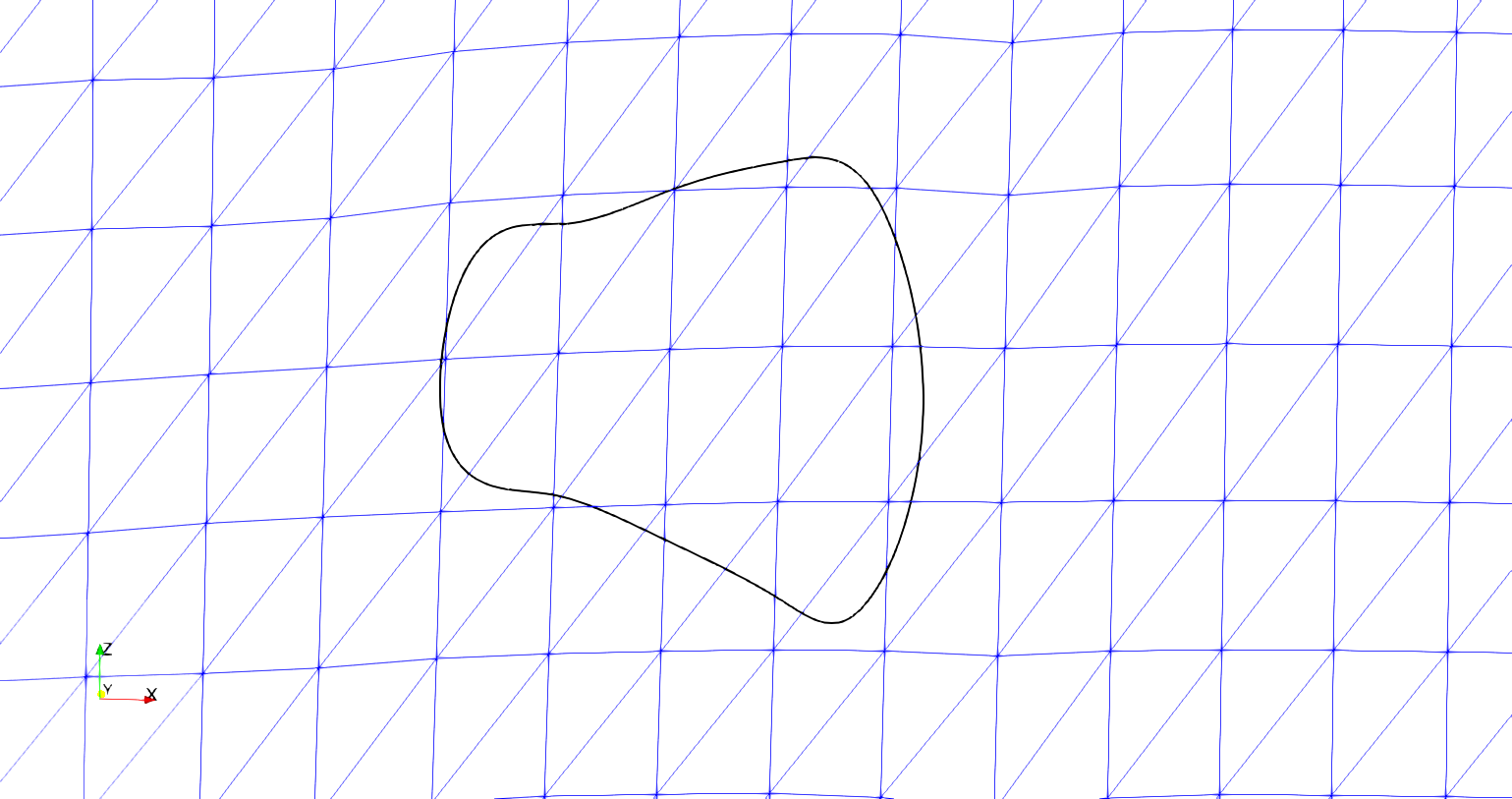
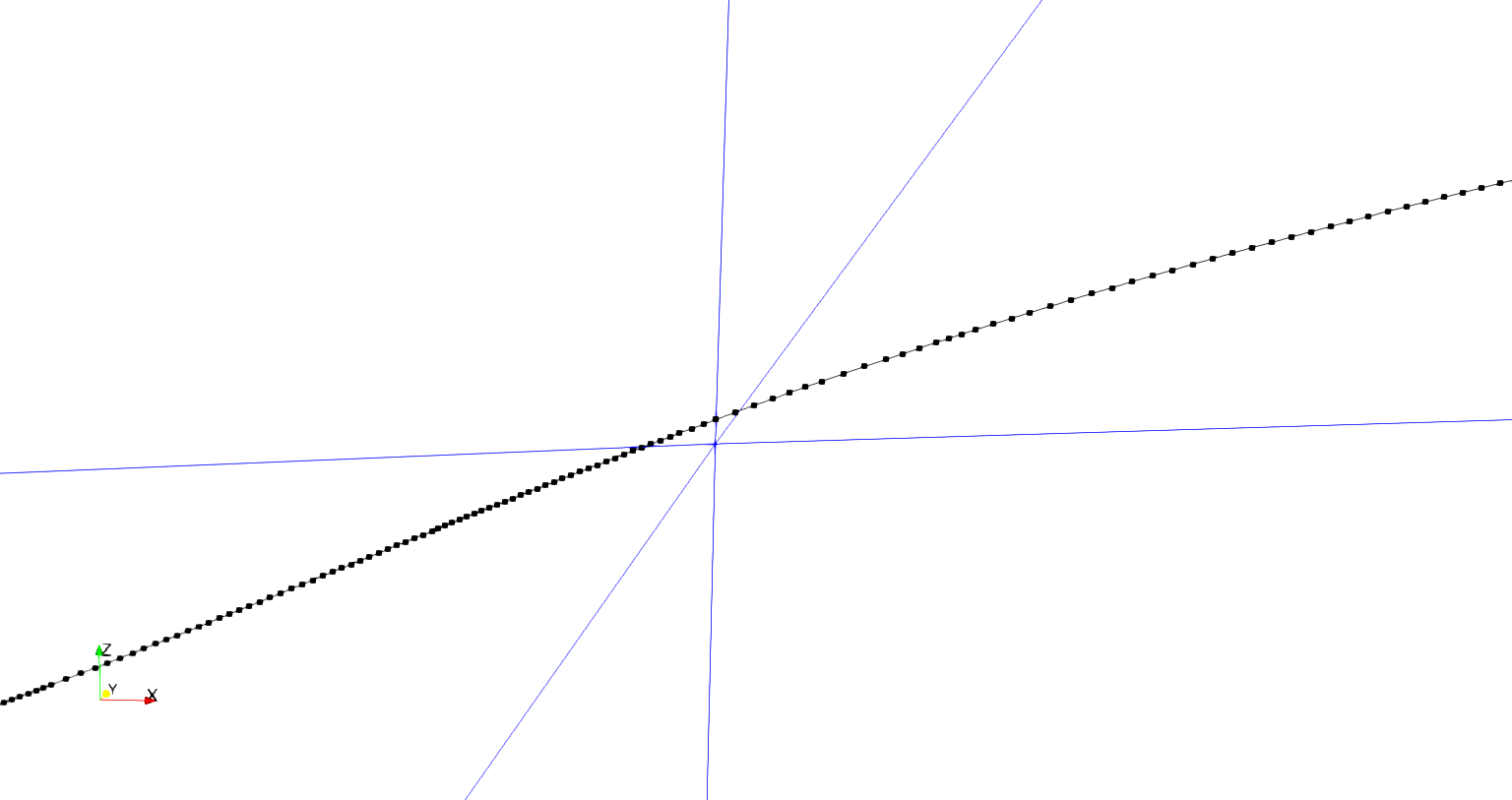
**Figure L: Boundary curve and element removal result**



1. **Initial intersection curve and surface mesh**
2. **Intersection curve with divided surface elements**



1. **Intersection curve with interpenetrating elements removed**

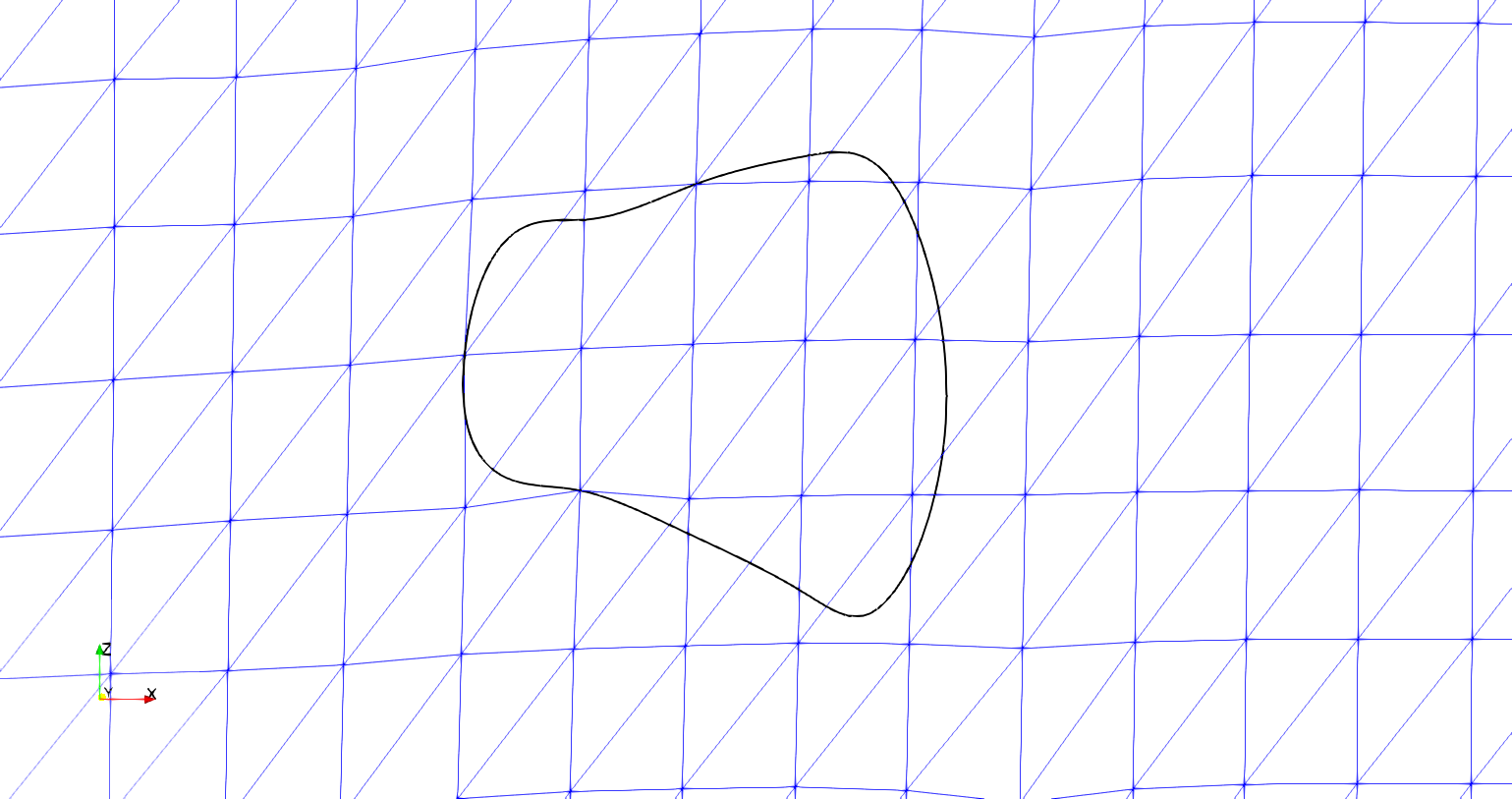


**Figure M: Intersection curve in close proximity to a surface mesh node**

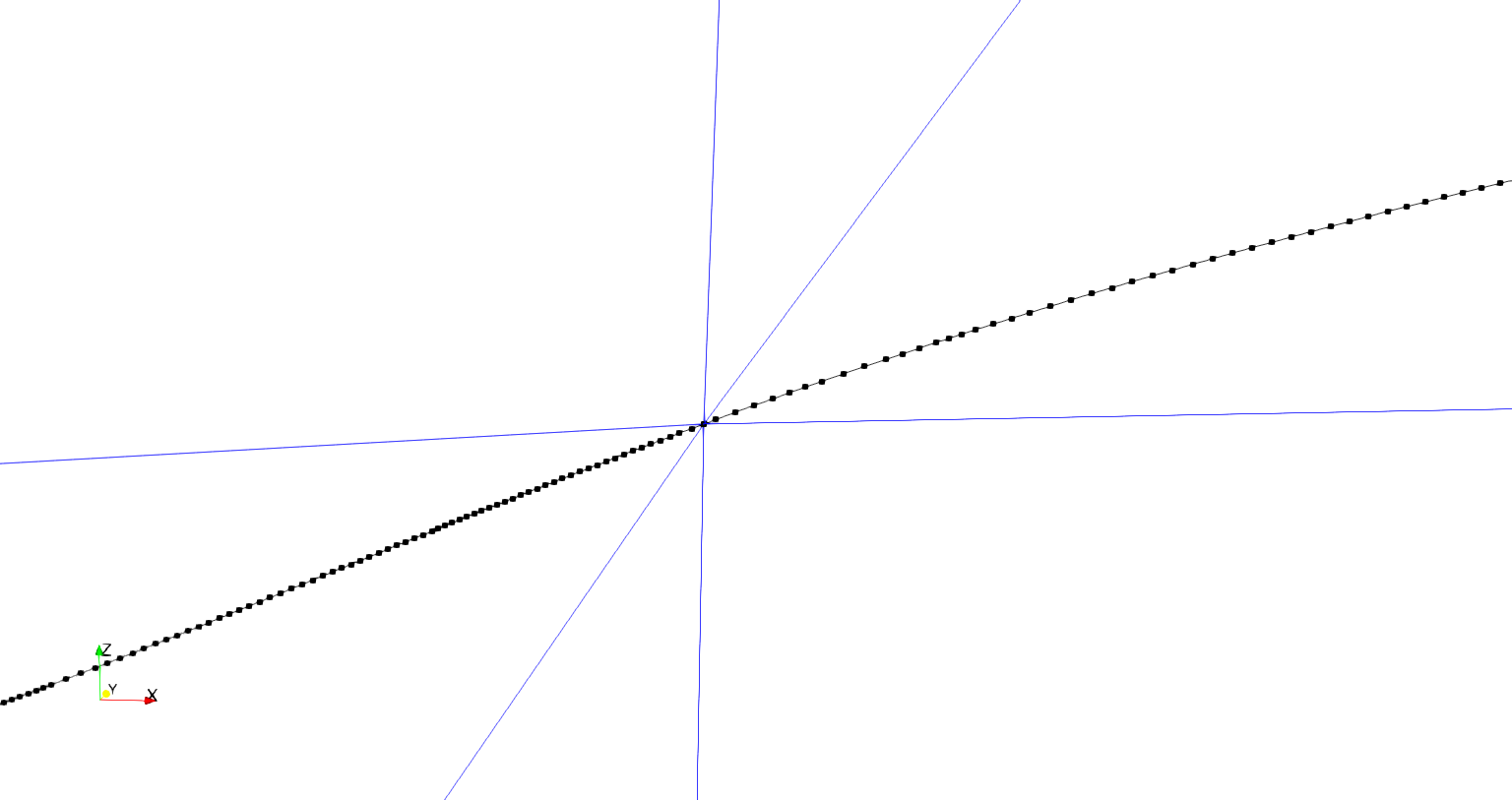
The curve must be represented in a form so that the interpenetrating elements from the surfaces can be removed. The chosen solution is to divide the surface elements where they are intersected by the intersection curve into smaller elements and remove the interpenetrating elements from the surface mesh. Once the curve is used to remove interpenetrating elements from both surface meshes, it becomes the common interface along which the two surfaces will be compatible. The first step is to move any surface node that is nearby of the intersection curve to a node on the curve itself. The surface nodes are moved to avoid the situation where the intersection curve contains a very small part of a surface element. Figure M shows a section of the upper part of the curve in figure L that is very close to a surface node on the surface mesh. In figure M there are very small regions of two surface elements that lie within the intersection region. In order to remove this small region, the element must be divided into smaller elements such that the new elements lie either entirely inside or outside of the curve. The size of these new elements is very small compared to the original element size. Enforcing compatibility between these small elements and the surrounding surface mesh elements would require a large amount of work and could negatively affect any finite element analysis conducted using the mesh.

Instead of attempting to re-mesh the surface elements to capture the very small regions that lie inside of the intersection curve, any surface nodes that are within a certain distance (determined by the user) of the intersection curve are moved to the curve, as in figure N.

Because the relative refinement of the intersection curve is very high compared to the refinement of the number of surface elements it intersects (usually two orders of magnitude higher in the number of elements), a surface node is directly moved the closest intersection curve node. A k-d tree searching algorithm is implemented to find the nearest intersection curve node to a surface mesh node. The distance between the two points is calculated and compared against an established tolerance. A larger tolerance allows for surface nodes farther away from the boundary curve to be adjusted to lie on the curve. However, a larger tolerance will affect the surface mesh more than a small tolerance that does not allow the surface nodes to move as much. The result is a surface mesh whose surface nodes that lie close to the intersection curve have been moved to the curve. Moving surface nodes to the intersection curve eliminates the localized high refinement that is caused by a small corner of the element remaining is the intersection curve.



**Figure N: Boundary curve with surface mesh nodes moved to the boundary curve**



Once the surface nodes have been moved, the intersection points between the boundary curve and the surface mesh are calculated. The Separating Axis Theorem (SAT) is implemented to detect if an intersection curve segment intersects with a surface element. The SAT is discussed more fully in Appendix A. Before the SAT can be used, the intersection curve segment and the surface element must lie in the same plane. An assumption is made that any segment that intersects a surface element is nearly planar with the element because the intersection curve lies on the intersection of both surfaces. Therefore, if an intersection curve segment intersects an element edge or lies within the element edges, the intersection curve segment should lie on the same plane as the element. However, this is not guaranteed because of the approximation of the NURBS surface as a faceted surface mesh, which changes the intersection point of the meshes slightly. Need advice for a figure here showing a vertical separation of intersection curve and surface mesh. The intersection curves do not lie perfectly on the surface, but it is difficult to get a comprehensible angle showing what is happening. Therefore, when evaluating if part of the intersection curve intersects a surface element, the curve segment is verified to be in the proximity of the element and then projected onto the surface element. Intersection curve proximity is verified by creating a vector between a point (**p**) in the plane of the element and the mid-point (**m**) of the curve segment (**s**) being checked, as in figure O. If the result is within a user defined tolerance, the segment midpoint is verified to be in or near the plane of the element. Another check is made to verify that the segment is within a certain vertical distance of the element surface by taking the dot product of the previously mentioned vectors and verifying the result is within a set tolerance (**Dot product** in figure O). The last check projects the segment (**s’**) onto the element plane and conducts the SAT. Projecting the segment onto the element plane is accomplished by determining the distance to the plane for each segment node and then moving the node along the element plane normal vector by the calculated distance to the plane of the surface element. The projection of the curve segment is shown in figure O as **s’**. The SAT is used to determine if the segment, when projected onto the surface element plane, intersects or lies within the element. Once the intersected elements are identified, the points where the intersection curve segments intersect the edges of the surface element can be calculated.

Dot product

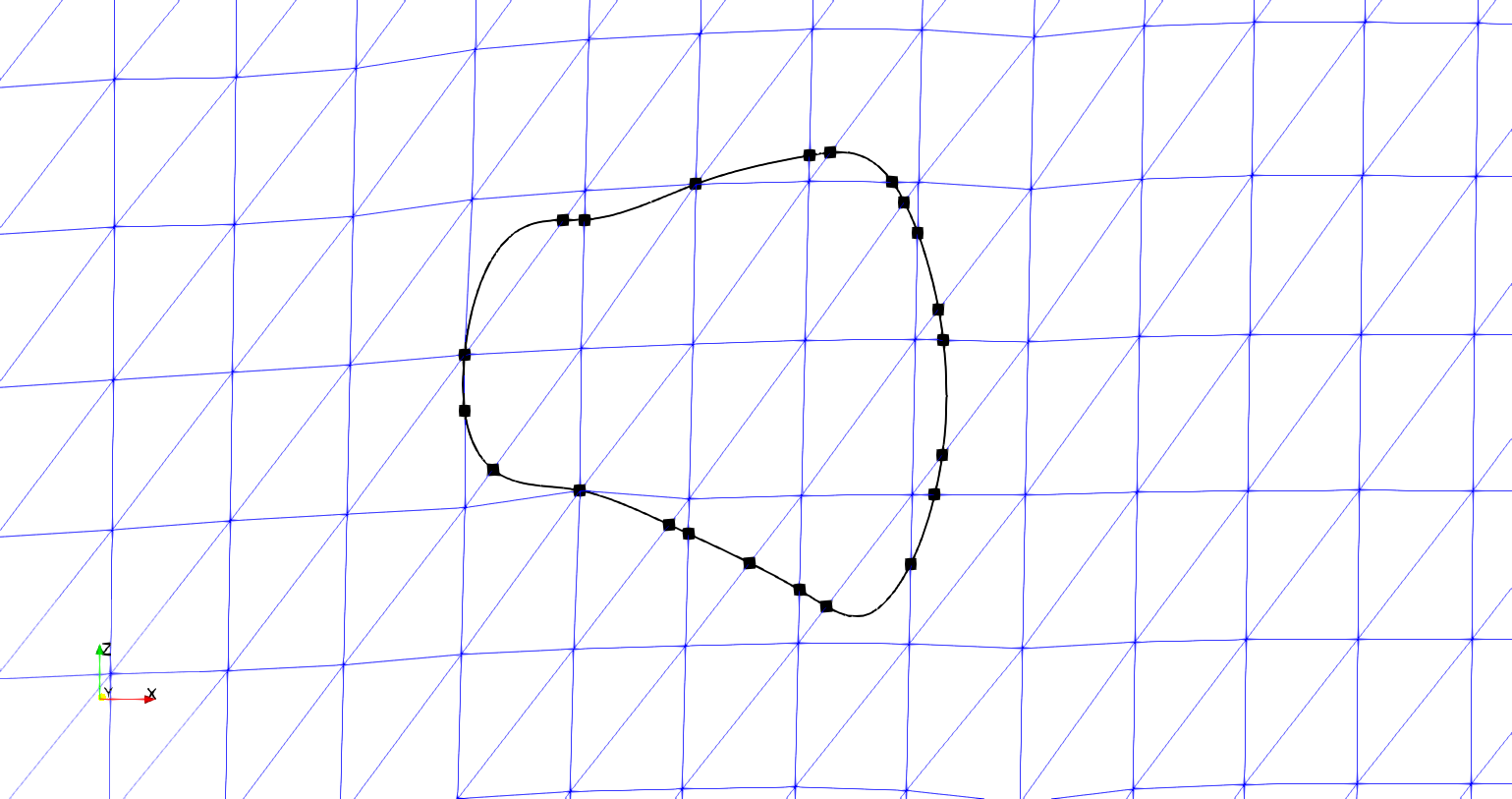
**m**

**p**

**Figure O: Illustration of proximity check algorithm**

**s**

**s’**



**Figure P: Boundary curve with marked surface element intersection points**

The result is clearly defined intersection points where the interpenetration boundary curve intersects surface element edges, shown in figure P. During this process the elements that are intersected by the intersection curves are recorded for later use in the routines that divide and re-mesh the intersected elements.

## The purpose and use of intersection points

The main purpose of dividing the intersection curves at each element intersection point is to establish a basis for compatibility between the two surface meshes. The algorithm calculates the intersection points for both tow surfaces whose interpenetrations are bounded by the curve. It is known that along this curve both surfaces have elements with edges that run directly through a point on the curve. Therefore, elements on each surface intersected by the curve will have a set of points on the curve that their edges will line up with, which is required in traditional finite elements. However, the intersection points are calculated for only one surface in this section for simplicity. The method for calculating the intersection points is the same for both surfaces.

The intersection points are calculated by first finding which curve segments intersect the edge of an element. In figure Q, curve segments **a** and **c** intersect the triangular surface element’s edges. Then, the segment is divided at the intersection points (points **1** and **2**), and the points are inserted into the curve. Every calculated element intersection point is added to the points that define the linearly approximated intersection curve. Now, the intersection curve is a base curve for compatibility between the tow surfaces. Once this compatibility is created along the intersection curve, the curve can be used to divide the surface mesh elements that the curve intersects. The element intersection points also serve a secondary purpose. The surface mesh refinement around the intersection curve would be much higher than the existing surface refinement if all of the points in the intersection curve were used to re-mesh the intersected surface elements. The intersection points identify the path the boundary curve takes through each element. By connecting the intersection points with line elements, the curve refinement can be reduced. If the curve refinement is not reduced, the algorithm that re-meshes intersected surface elements will produce high-refinement meshes to replace the original surface elements. Large changes in the refinement of a mesh can have negative consequences on a finite element analysis. Therefore, the relative refinement of any surface element that requires a new mesh should match the same level of refinement of the surrounding elements as much as possible.



**a**

**c**

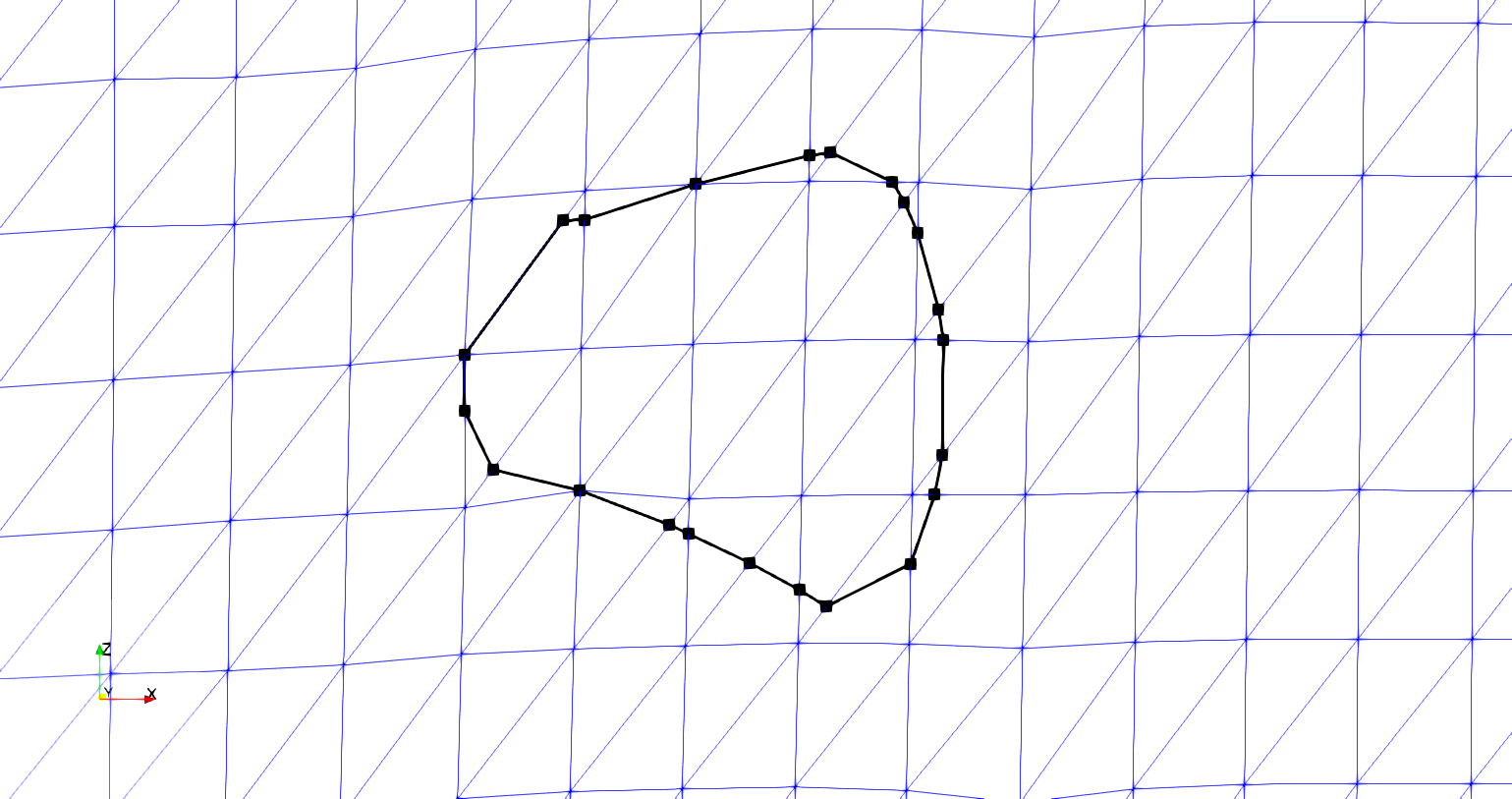
**2**

**1**

**Figure Q: Intersection curve with marked edge intersection points with a surface element**

The high refinement is caused by the number of intersection curve segments that lie inside any element that is meshed. Reducing the refinement of the intersection curve will also reduce the relative refinement of the elements that are intersected by the intersection curve. Also, because the intersection points are calculated before the boundary curve refinement is reduced, there is no loss of accuracy when removing interpenetrating nodes and elements. An iterative loop is run to remove any curve points between two consecutive intersection points to reduce the intersection curve’s refinement. The result of the method can be seen in figure R.

Figure R shows that all of the surface nodes inside the perimeter of the original intersection curve from figure L are still in the interior of the curve. Also, the relative shape of the curve is the same, indicating that the lower refinement does not overly affect the interpenetration boundary curve. Figure R is the result of capturing the interpenetration points of just one surface for illustrative purposes. However, once the intersection points from both curves have been added to the intersection curve, there will be intersection points that line within the elements in figure R as well, as shown in figure Q. I am not sure how to discuss the effect of both surfaces in tandem. I could potentially show both results simultaneously and discuss them, but I am afraid it may add more confusion.



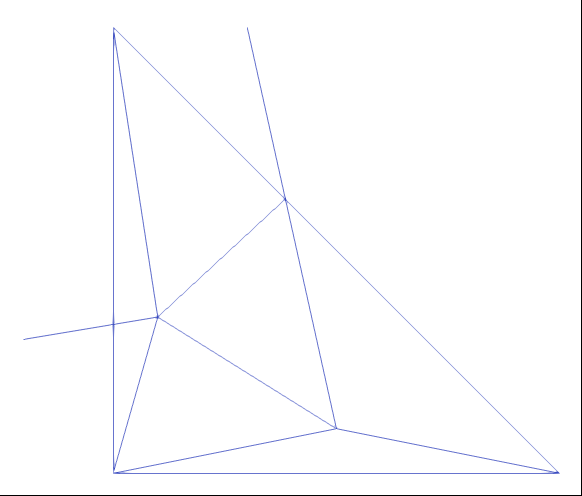
**Figure R: Reduced refinement boundary curve with marked intersection points**

Once the intersection points between the surface elements and the intersection curve of the surfaces have been used to reduce the curve’s refinement, the surface meshes element are divided where the curve intersects the element. The interpenetrating elements that lie partially inside the intersection curve are divided along the curve. The two parts of the original element are re-meshed using any intersection points that lie in the element or on the edges of the element. The new tow surface mesh created by individually adding each intersected element’s divided mesh back into the original surface mesh. The elements intersected by the intersection curve are iterated over and divided individually. The same intersection algorithm involving the Separating Axis Theorem is used to collect intersection curve segments that lie within the current surface element being divided. Once the segments have been collected, they are checked against the edges of the element to verify which intersection curve segments have endpoints on the element’s edge. Figure R shows only one segment per element, but once the intersection points from both surfaces are added to the intersection curve, there could be more than one segment per element. Figure S illustrates this point with an example element with multiple intersection curve segments.



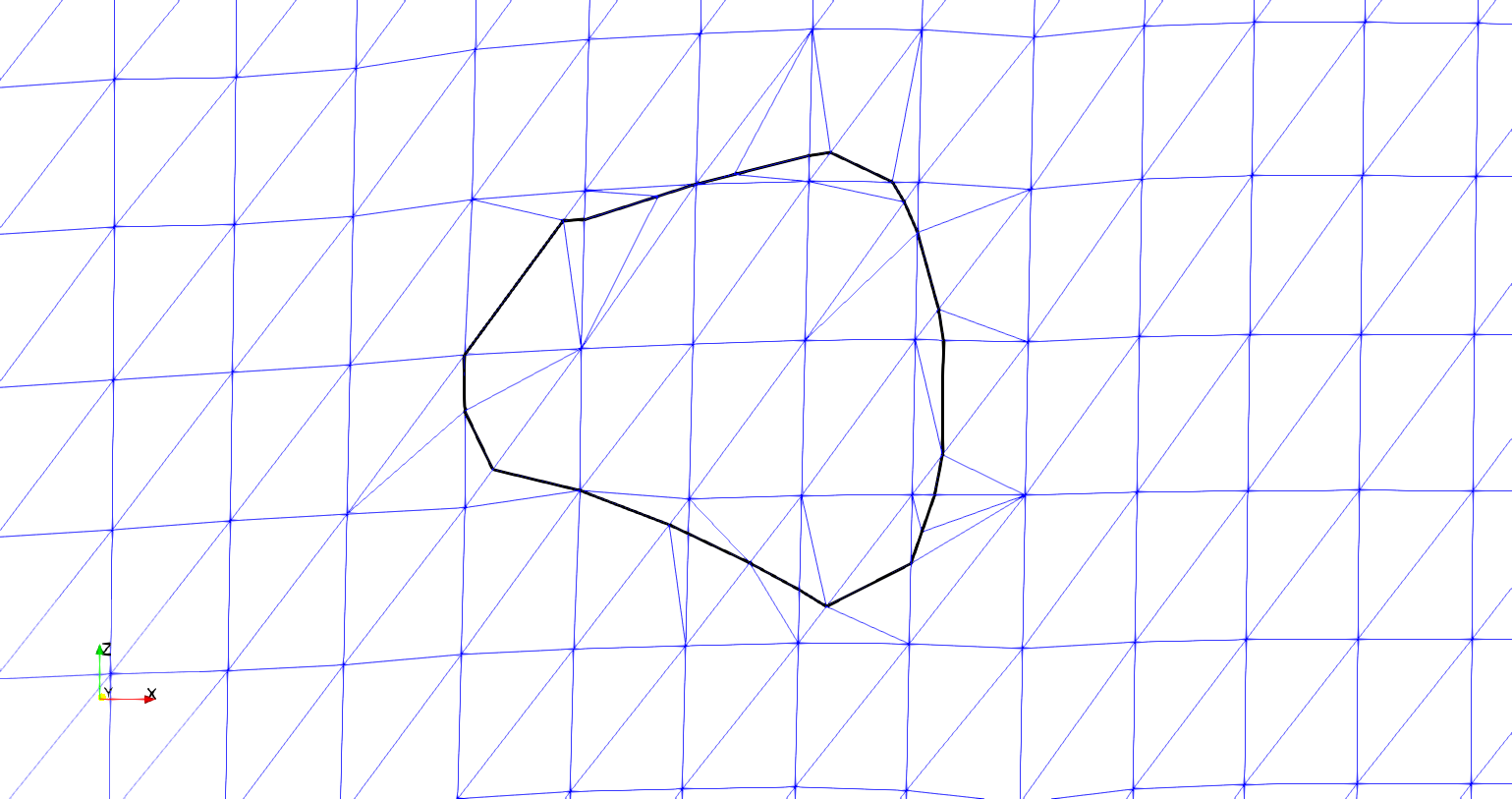
**Figure S: Example surface element with three boundary curve segments intersecting**

Figure S shows two intersection curve segments with endpoints lying on the surface element edges and one curve segment completely contained in the element. A mesh generation library called Triangle is used to create a new mesh of the surface element in figure S that includes the intersection curve segments. The library requires that all line segments, referred to as boundary segments by the library, that define the required boundaries be included in the list of segments be given to the library. These segments include all surface element edges and the intersection curve segments that lie within the element. Once the curve and element segments are given to the library, a mesh is returned, as shown in figure T.



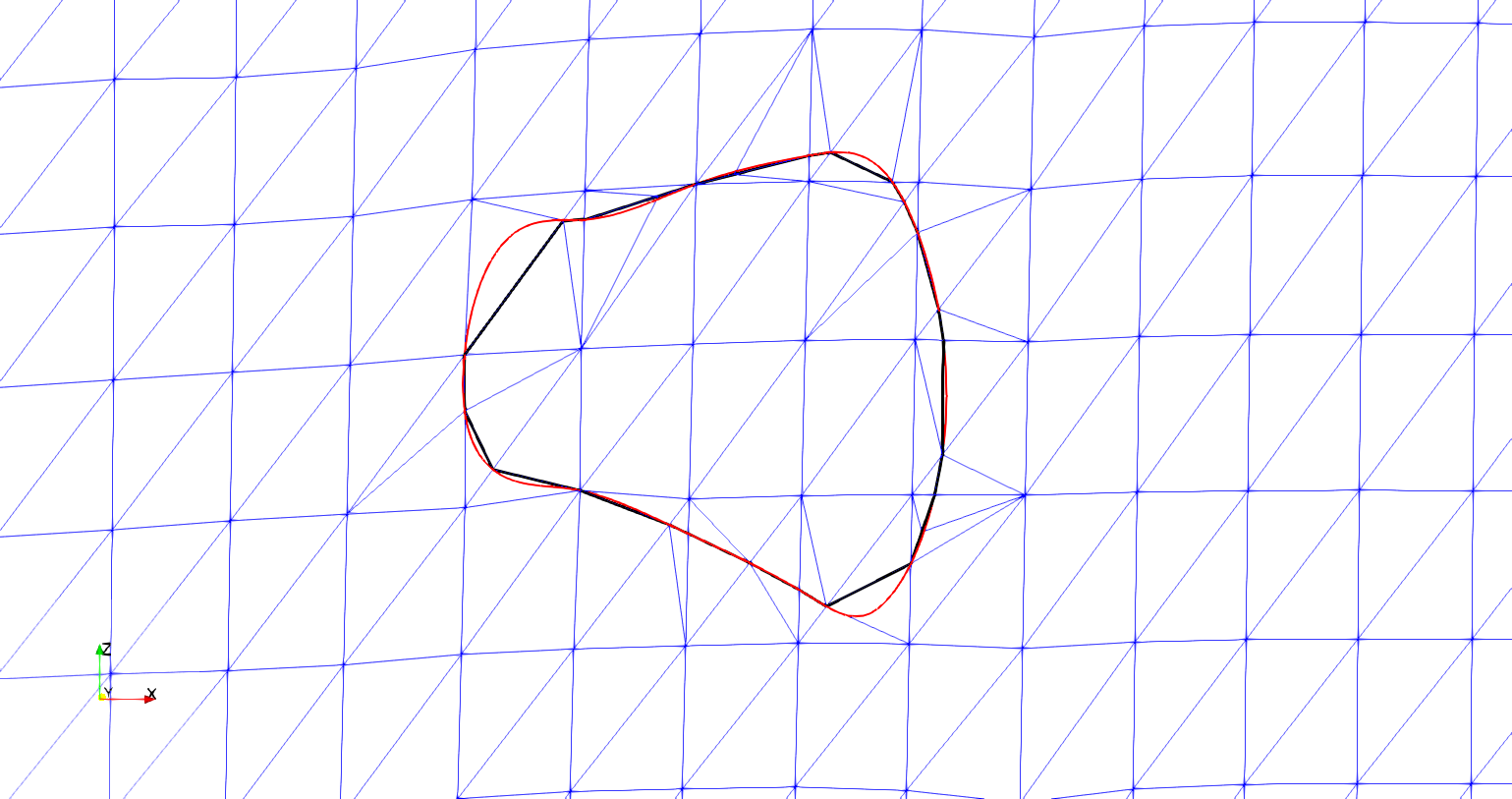
**Figure T: Sub-mesh of surface element with boundary curve**

The newly meshed element is then be added back into the surface mesh, replacing the original element. Figure U shows the result of the algorithm for the test case shown. Figure U shows both the relative refinement of the newly meshed elements as well as the reduced refinement of the intersection curve. The refinement is comparable between the sub-meshed surface elements and the untouched elements. Figure V shows the original boundary curve compared to the new curve used to re-mesh the surface.



**Figure U: Reduced refinement intersection curve and re-meshed surface elements**

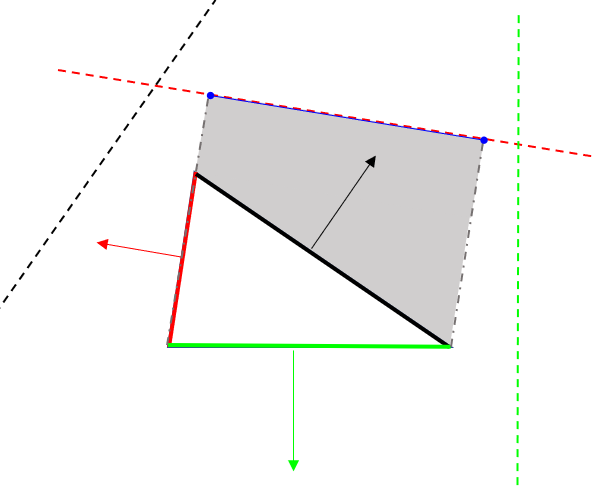
Figure V shows how the overall shape of the intersection curve does not fit the original intersection curve well. Although no surface nodes or elements are left out from the reduced refinement intersection curve, a better fitment of the original curve data is desired. Some sections of the curve are not captured when the curve has been coarsened, but it is expected that the fitment will improve with the inclusion of the intersection points from the other tow surface. The result is a mesh that includes the curve along which the surface mesh will be cut to remove the interpenetrating region. When the curve has been used to remove interpenetrating elements from both surfaces, the surfaces will then share a curve along which there is compatibility. The compatible intersection curve is the most important feature of the methods developed during this research. Previously, there has not been a method that will ensure a compatible region between any two tow surfaces. Now, the intersection curve between the two tow surfaces is also where the two surfaces are connected in a compatible manner. Using the intersection curve, a compatible mesh can be created in the interior of the intersection curve that will be used to replace both surfaces elements that have been removed. The result is a compatible mesh that both surfaces share, creating a connecting surface between the tow meshes. I think I could end with the same figure with the inner elements removed from the mesh like weve discuss as well but im not sure if it is redundant.



**Figure V: Original (red) vs reduced refinement boundary curve with new surface mesh**

# Appendix

The Separating Axis Theorem (SAT) is implemented to identify which elements are intersected and to refine the boundary curve where it intersects surface elements. The SAT starts by projecting a shape onto pre-determined axes. The projection can be thought of as the shadow of the shape on an axis. The axes are created by taking the normal direction of an shapes edge and creating an imaginary infinite line in the same direction. Figure 1 shows a projection of a triangle onto the red axis (RA) that is determined by the red edge (RE) and is parallel to the red edge normal (REN). The line segment along RA (labeled “Triangle Projection”) is the projection of the triangle onto the axis RA in Figure 1. Three axes are identified, one corresponding to each side of the triangle.



**Figure 1: Projection of a triangle on an axis**

**RA**

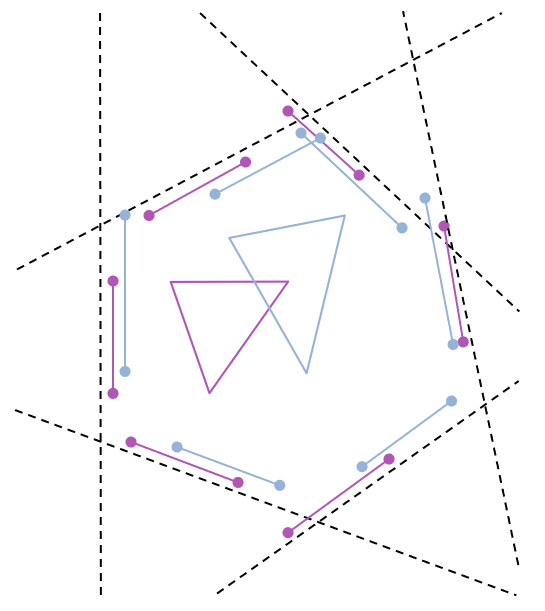
**REN**

**RE**

**Triangle Projection**

Next, the projections are tested to see if they overlap. If there is any axis on which the projections do not overlap, then the polygons do not intersect. If the projections overlap on every axis, then the polygons do intersect. A reference picture is shown in Figure 2.

In Figure 2.a, two triangles are shown to intersect. This can be verified by looking at each dotted line that represents a projection axis. Along each axis the bounds of the triangles are shown. There is no axis in which the bounds do not overlap. Figure 2.b shows the case when the two shapes do not intersect. The axes (dotted lines) are the same in 2.a and 2.b, since the orientations of the two triangles are the same, only the positioning is different. Circled are shape bounds that do not overlap in 2.b and therefore verify that the triangles do not intersect. This method is adapted so that the second shape is simply a line segment from a curve.



**a) SAT in which triangles overlap**

**b) SAT in which triangles do not overlap**

**Figure 2: Two cases for testing the Separating Axis Theorem**

