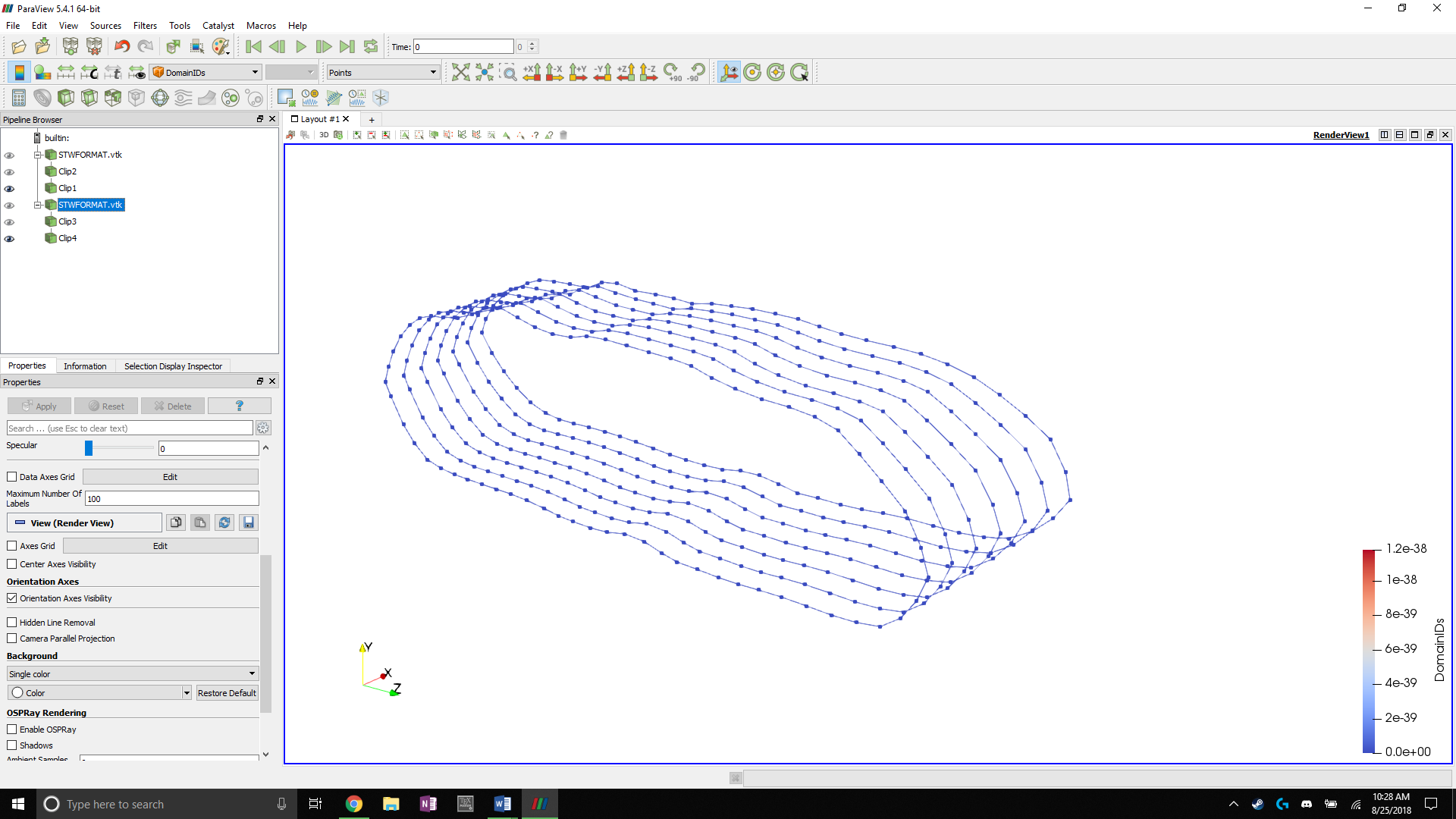
# Non-Uniform Rational B-Spline Surface Strategy

The methods implemented for the default surface formats provided by VTMS removed the majority of surface interpenetrations but lack the accuracy to detect all interpenetrations and the methods to remove the interpenetrations were not robust. Another approach was needed to reliably detect and fix surface interpenetrations. There are many references that use parametric surface representations to detect interpenetration between surfaces. One of the more popular parametric surface types is the non-uniform rational b-spline surface (NURBS). NURBS were chosen as the new surface type because of its heavy documentation, third-party support and ease of implementation. The SISL library from the Department of Applied Mathematics at SINTEF ICT supports surface fitting to external data and has intersection boundary curve detection algorithms. However, this library does not fix the interpenetrations and requires adjustment to the data that it returns. This library has been implemented in the methods that follow.

## Pre-SISL data formats and SISL required inputs

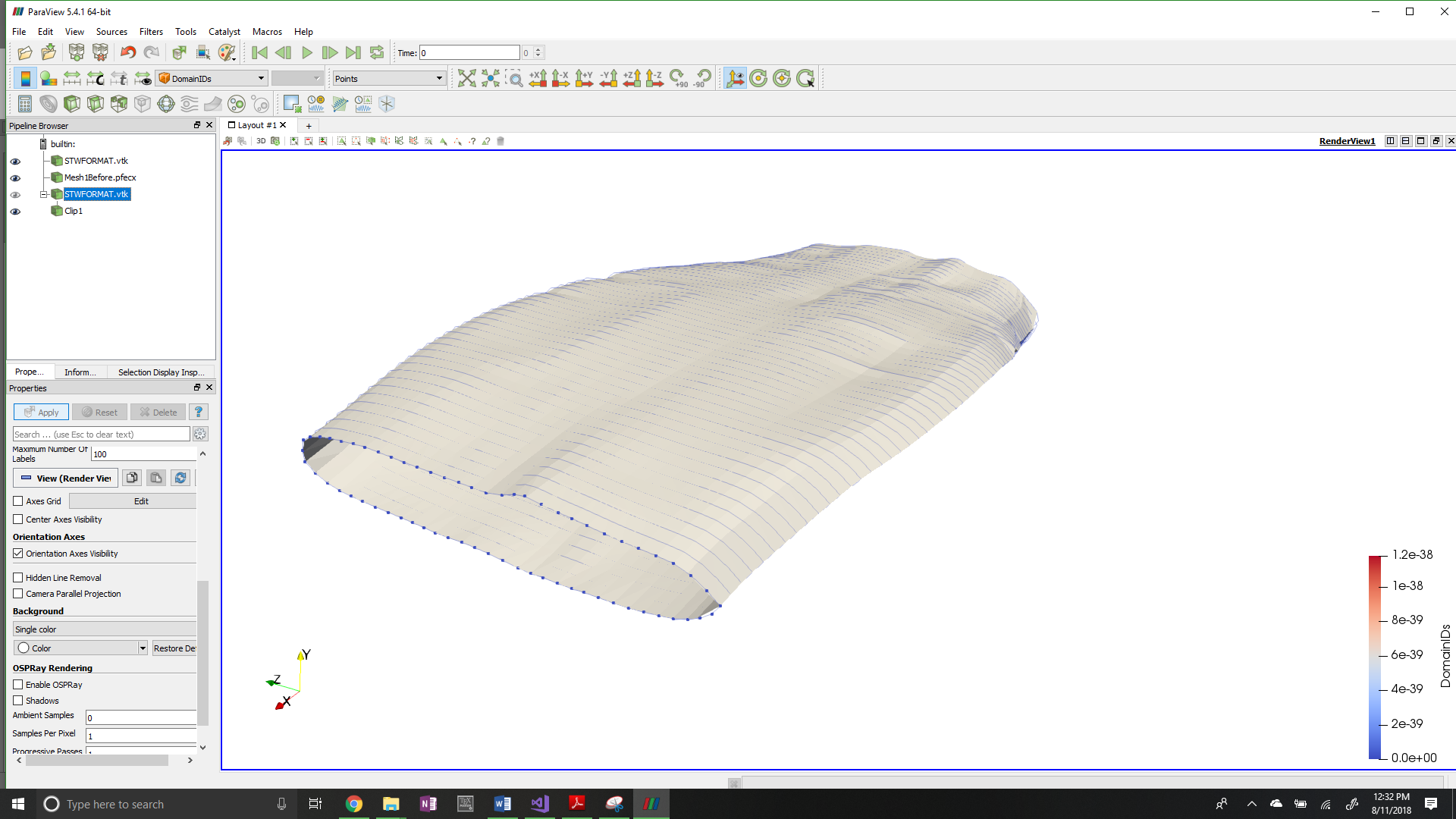
A NURBS surface is a sub-class of a parametric surface and uses a curvilinear coordinate system to describe a three-dimensional surface. NURBS use two parameter values to return the location of the surface at any point. Therefore, a way to describe the VTMS tow surface with two parameters was needed. The SISL library requires a list of all the points in the surface, and the number of points in the two parameter directions. These parameter requirements needed to be fulfilled to create a NURBS surface using the library.

VTMS has a format describing its surfaces that can be easily used with the SISL library. 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 the points of a cross section lie in the same plane. Each point that defines the surface has a cross section it belongs to and can be found using two reference values, the cross-section number it lies on and which number point it is in the cross section. The reference values are analogous to curvilinear coordinates that describe the surface. Figure A shows multiple cross sections with its points and the starting point of each stack is marked. The lines that outline the cross sections have been added to clarify which points belong to a cross section.



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

A method was developed to read the .stw file in the order that it exports the data, which lists each cross section in order and the points that make up that cross section. Each point is stored in the order that it is read from the file. SISL is then told the number of cross sections and the number of points which are used to define the range of the curvilinear coordinate in the ***u*** and ***v*** directions (Figure B). The cross-section’s points become the control points for the NURBS surface.



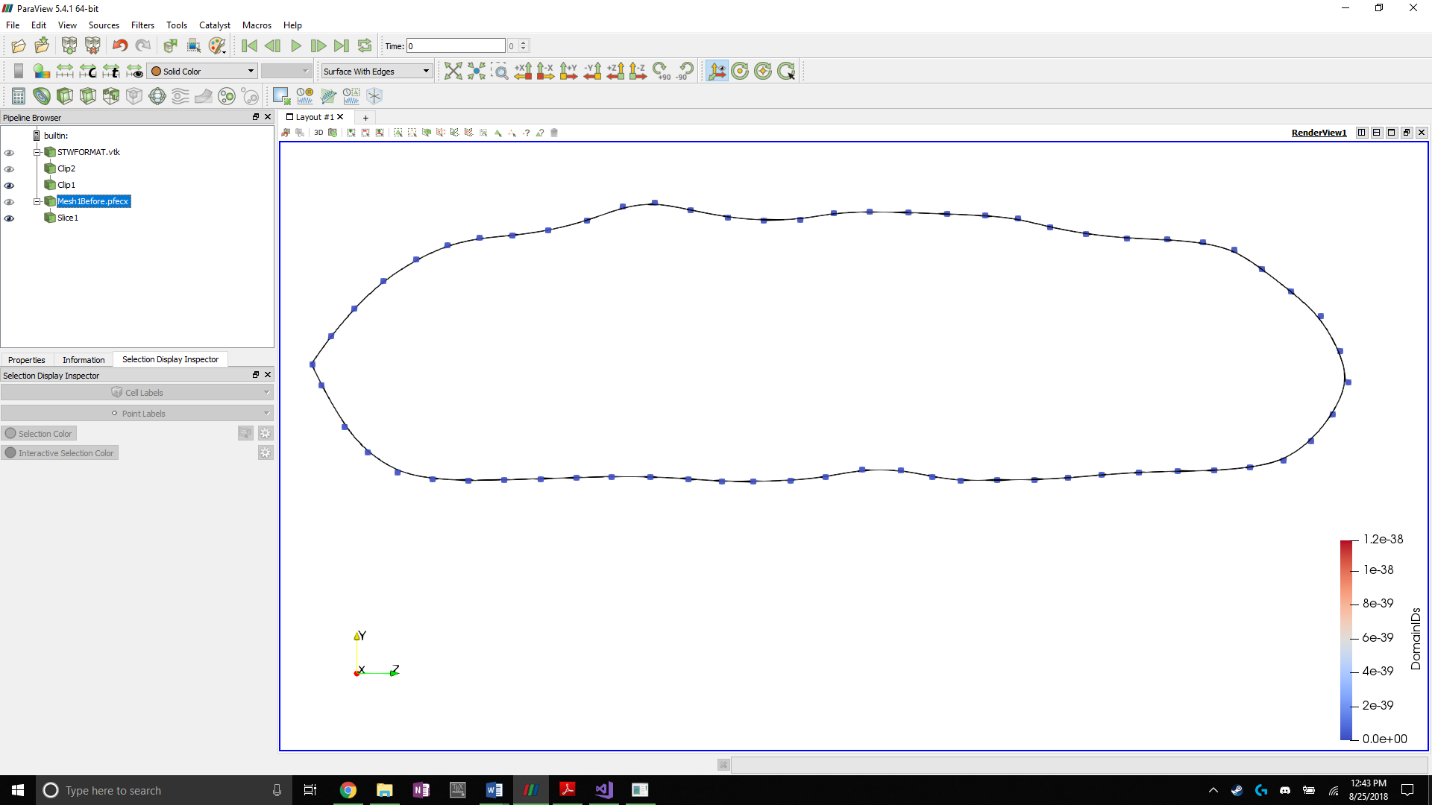
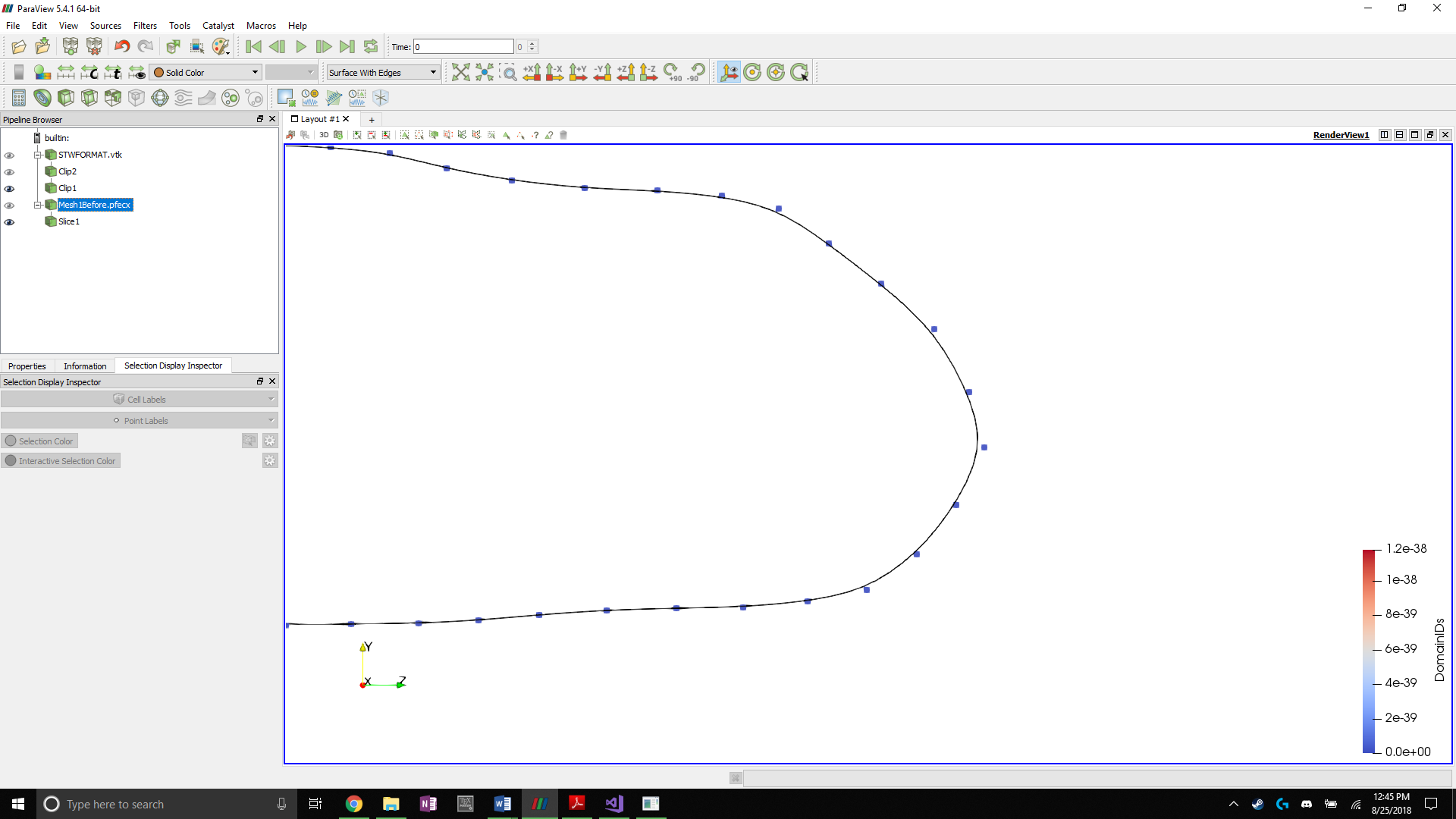
***u***

***v***

**Figure B: Parametric coordinate system for tow input data to the SISL library**

SISL has its own method that takes this data as input and converts it to a SISL definition of a NURBS surface. It is important to note that the resulting surface approximates the data sent from VTMS. This is because of how NURBS are formed by their control points. The curves that make up the surface are not required to run through the control points. In general, the control points rarely lie exactly upon the b-spline curve it defines. However, for a non-idealized tow surface produced by VTMS, the NURBS approximation is very good because the surfaces produced by the SISL library fit the original data very well. This relationship became clear when the surface created by SISL is compared it to the original VTMS surface data. Figure C 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 arcs significantly, the points that are used to control the surface do not lie on the surface but are close. Where the surface does not bend significantly 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.

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 boundary curves that outline the regions where two surfaces interpenetrate. The library returns these interpenetration boundary curves as b-splines. Multiple functions were developed to allow the surfaces and boundary curves that are being used inside of the library to be viewed.



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

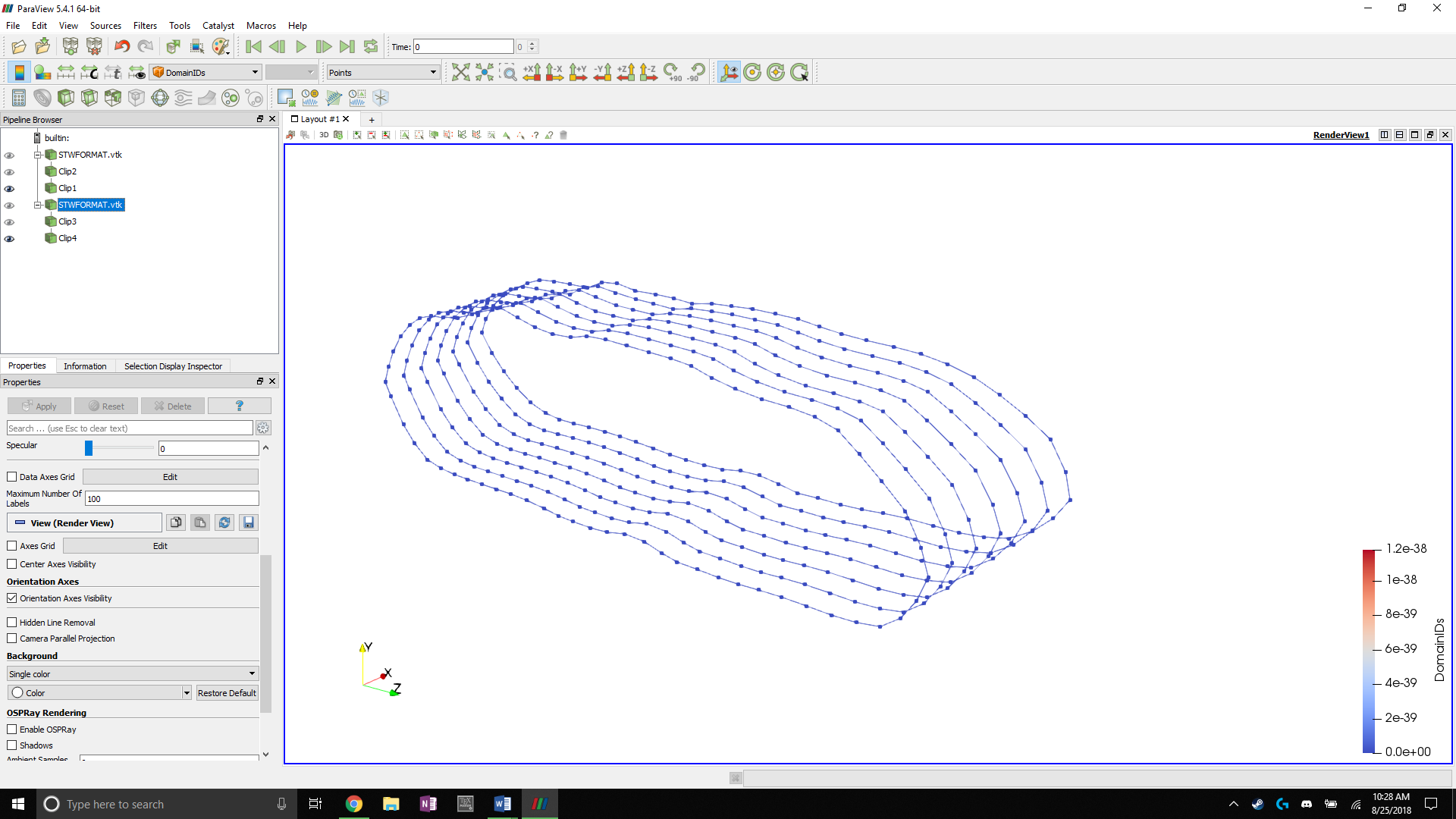
## NURBS and B-Spline viewing algorithms

One drawback of using NURBS and B-Splines to describe a surface is visualizing the surfaces and curves. Most visualization software suites use polygons to approximate the surface of complex shapes to render them for viewing. Therefore, accommodations had to be made to use the chosen viewing software. Paraview was used as the main visualization software due to previous experience using the software. SISL comes with its own visualization suite but it is used for visualization only and lacks important functionality available in Paraview that is used to show interpenetrations between surfaces. Another set of software, called BetaMesh, was used that was extremely helpful. BetaMesh is a mesh framework that is well suited for domain discretization and manipulation. BetaMesh also has modules that can conduct a wide variety of finite element analyses. However, BetaMesh primarily used for its mesh framework. BetaMesh was developed at Texas A&M University and is currently maintained by Dr. John Whitcomb and his students. I don’t know exactly know who to credit here. BetaMesh also comes with methods that allow it to directly export mesh data to Paraview for visualization. It is well suited for the task of visualizing NURBS as a polygonal surface mesh.

To use BetaMesh as a mesh framework, the NURBS surfaces and b-splines from the SISL library must be discretized. This was accomplished by using a surface (or curve) point sampling function supplied by the SISL library. The function returns the cartesian coordinates of the surface at a chosen set of curvilinear coordinates. The maximum and minimum curvilinear coordinates for both parametric directions of the NURBS surface were found to make sure the surface is sampled at its boundaries. These ranges are were used to evenly divide the parameter space to reflect the desired amount of surface refinement.

The refinement of the surface is controlled by increasing the number of times the surface is sampled in the respective parametric directions. The same level of refinement as the original VTMS surface data was chosen because the refinement is a good compromise between accuracy and coarseness. The surface was sampled corresponding to the number of cross-sections in the ***u*** direction (axial direction of the tow) and the number of points per cross-section in the ***v*** direction (circumferential direction of the tow), similar to figure A. The surface was sampled by starting at one end of the tow surface (minimum coordinate value in ***u*** direction) and sampling around the circumference of the surface (stepping from the minimum to the maximum coordinate value in ***v*** direction). The method then steps to the next coordinate value in the ***u*** direction and sample the entire circumference again. Each time the surface was sampled, the function returns a set of coordinates that was used to create a BetaMesh node object. These node objects store node data such as coordinates, local and global number, owning partitions, and multiple functions that can edit and assign the node the various attributes. These nodes were stored in a list in the order they were sampled them so that the order in which they are stored was known. This is important when forming the surface elements that are used for the visualization. Once the surface was sampled according the refinement chosen, a method was developed that creates the surface mesh using the vector of nodes.

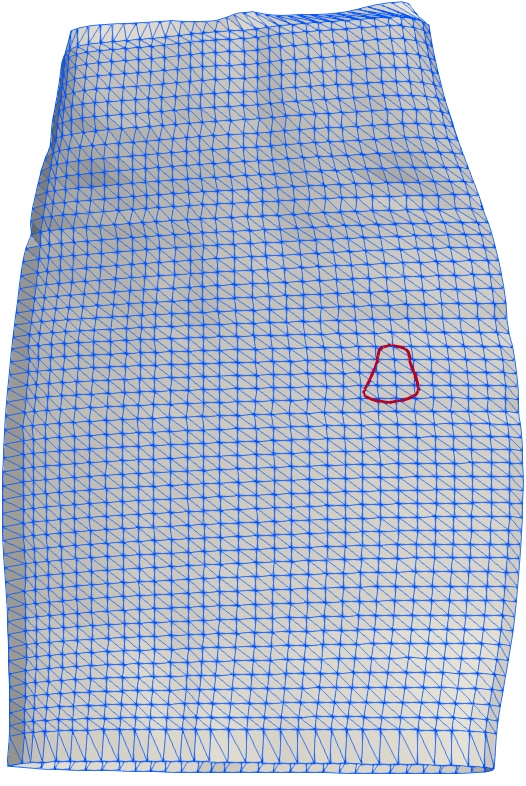
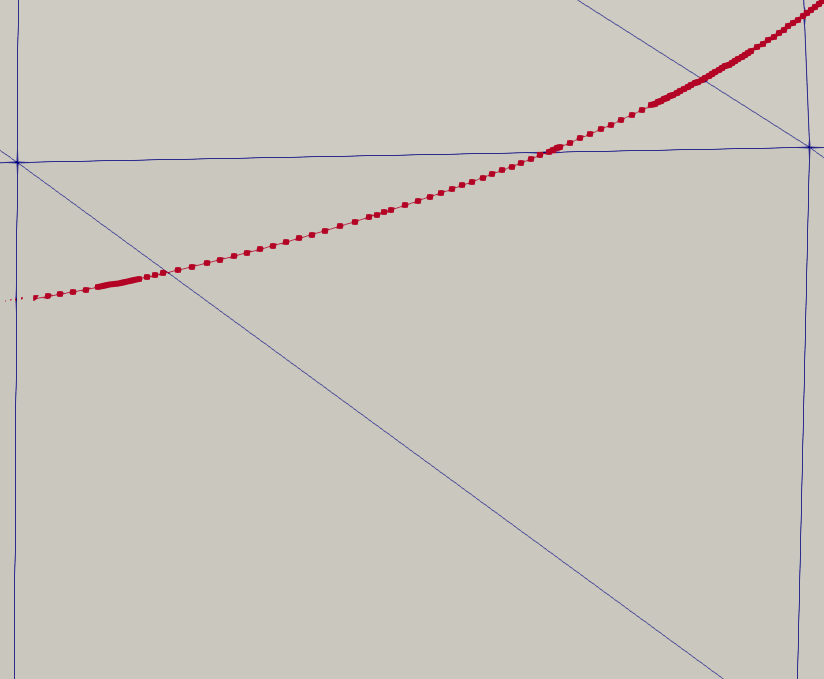
The mesh was easily created because the order that the surface was sampled and how the nodes in the vector were ordered was known. The algorithm begins by connecting two adjacent nodes on a cross-section and doing the same for the proceeding cross section. Then the same indexed points on the two cross sections are connected, creating a quadrilateral element.



Two opposite corners of the quadrilateral element are connected to create two triangular elements, which are stored as BetaMesh triangular elements. All of these elements were assigned to a surface mesh that is exported by a BetaMesh function to file formats that are used in Paraview.

The SISL library also returns boundary curves as b-splines, which were converted to be viewed by Paraview. The same function that samples the NURBS surface at regular intervals was used and coordinates of the curve were returned. For each node that is created from the returned coordinates, it is connected to the previous point found using a line element. A line element is defined as a three-dimensional, two node line segment that can be manipulated and used the same way as a triangular surface element is used. The curve was sampled at a high refinement so that the meshed curve closely resembles the b-spline. Figure E shows an interpenetration boundary curve and an example of the refinement level of the boundary curve.

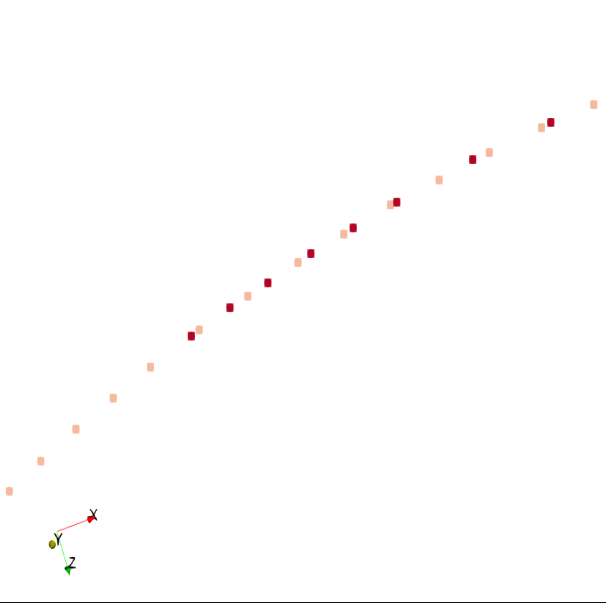
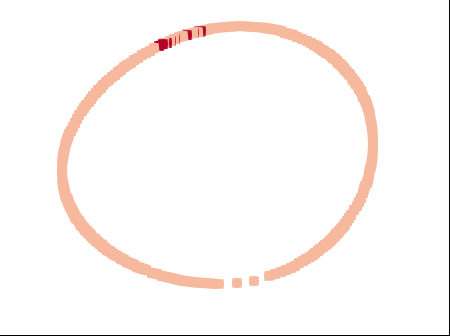
The result of these methods is the ability to visualize NURBS surfaces and curves from the SISL library. The surfaces shown in Paraview are approximations of the NURBS surface due to the discretization process that was used to visualize them. However, the points that define the surface elements were directly sampled from the surface to reduce the error in the approximation. If the refinement of the sampling was increased the relative error between the approximation and the actual surface would be reduced. The same relationship applies to the boundary curve. The higher refinement of the interpenetration boundary curve relative to the surface was to ensure that all interpenetrating nodes from the surfaces were contained by the boundary curve. Once the surfaces and boundary curves are converted, a method to implement the boundary curve into the surface was needed.



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

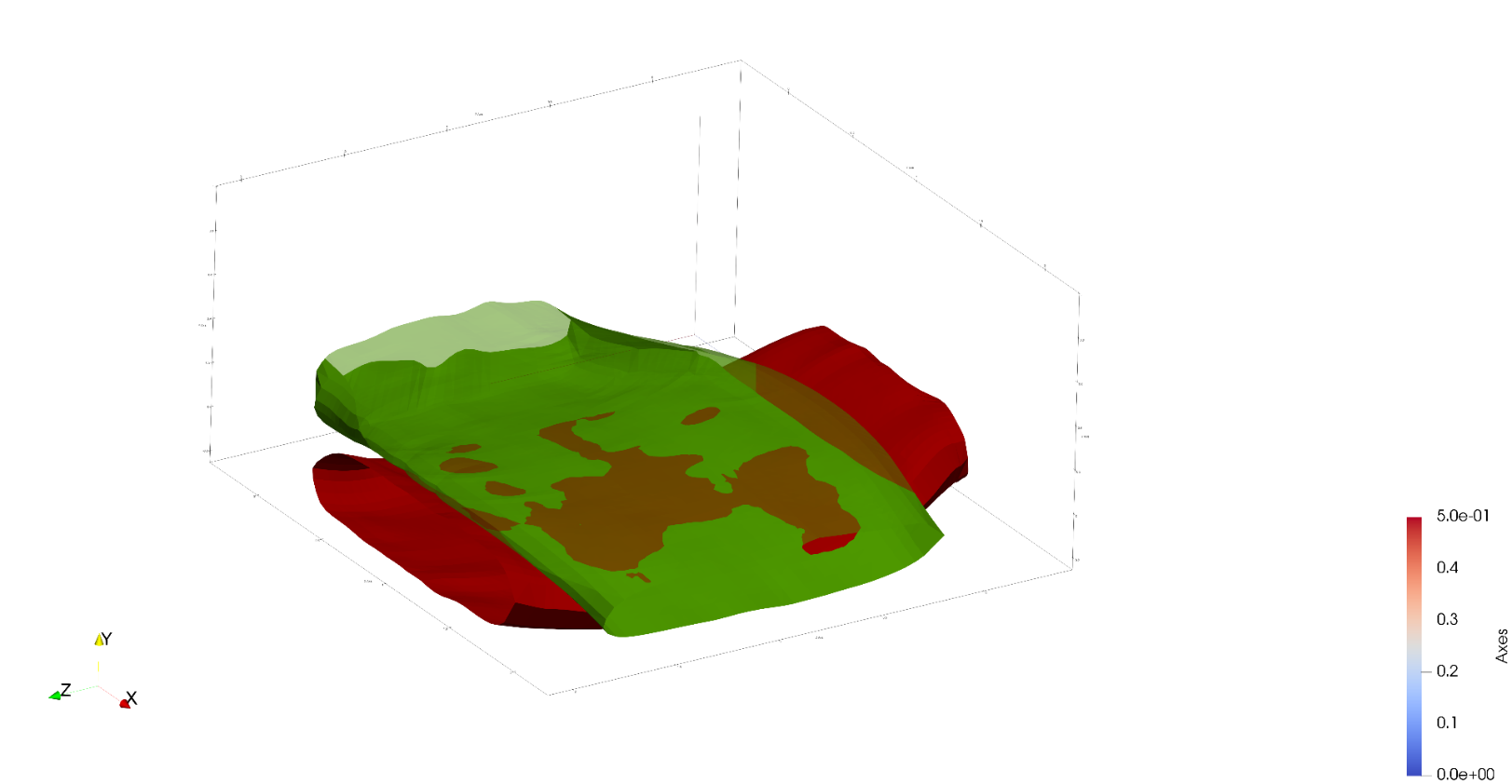
## Correction of Boundary curve meshes

The main drawback of using the SISL library is that the boundary curves returned from the methods are not guaranteed to be unique, closed, or even have shared nodes between their converted meshes due to how the sampling method works. Therefore, the meshed boundary curves needed to be corrected before they were added to the surface mesh. Figure F shows an example of two boundary 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).



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

There are multiple scenarios where two curves either overlap and duplicate data, or two open curves connect to create a closed boundary curve. An assumption is made that if two surfaces interpenetrate then any interpenetration region that they create can be described by a boundary that is closed. This is a valid assumption because if two surfaces interpenetrate in a relatively small region then one surface should both enter and exit the opposing surface, creating a closed region of interpenetration that is bounded by a closed boundary curve. Figure G can be used as a reference for two surfaces interpenetrating. Using this assumption, it is required that all the boundary curves from this method are unique and closed.



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

The first step of this method identifies which boundary curves are closed and which are open. When the NURBS curves were converted into line element meshes, line elements were created between every pair of adjacent nodes sampled from all of the curves. The line elements were not linked together but instead added to a list and ensured that each linear element was unique. These elements were also added to one large BetaMesh mesh object which allowed the usage of mesh manipulation operations, such as the ability to determine how many elements a node belonged to. Most nodes belong to a pair of line elements because they are the connection point between two individual line elements. However, some nodes only belong to a single line element, which indicates that it is the beginning or end of a curve. One of these nodes is identified as the starting point for a search loop that builds a set of boundary curves that are known to be open or closed. Using this node, the element it belongs to is selected and added it to a temporary mesh object that is used to record the current curve being defined. The second node on the element is used to find the node’s remaining parent element. The second parent element is added to the curve mesh and the process is repeated, maintaining connectivity definitions. As each new element is added to the curve mesh, it is removed from the list of line elements and checked to see if the second node of the element is at the same location as the original single parent node. If the two nodes coincide, then the curve is defined as closed and the algorithm loop is exited. The closed curve is then added to a boundary curve mesh list for later use. If the algorithm finds a node that has no second parent element and does not coincide with the original starting element, then it is an open curve. The curve is labeled as open and saved it as well. This process continued until all of the boundary curve line elements have been removed from the original list of unique line elements. The end result is a curve list containing both open and closed boundary curve meshes. This is required so that duplicate and open curves could be removed.

To determine if any of the curves duplicate data, a method compiles all of the curves into a list and performs a dual-loop iteration. These loops choose the first closed curve (denoted curve **A**) in the list and compare the remaining boundary curves against it. If the curve being compared (curve **B**) lies on 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 determined if it lies on any line element from curve **A**. The method that calculates this uses the two end points of the line element (nodes **a** and **b**) and the node to be checked (node **c**). 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 it is below a certain tolerance, node **c** is determined to lie on the line that goes through **a** and **b**. The dot product is then calculated between the two vectors. If the result is greater than zero but less than one, it is known that point **c** lies between the points **a** and **b**. These two checks verify if node **c** is on the line segment formed by **a** and **b**. This identifies an overlap and the curve which the node belongs to is removed. If the test does not result in an overlap for the beginning or end node of the curve, the curve is kept in the curve list. The result of this method is a set of unique, closed curve meshes that are used to subdivide the surface meshes along the boundary curves.