# Description of Data Types

Understanding how VTMS describes the surfaces it produces is important to better understand what constitutes an interpenetration between surfaces in VTMS. Before a surface is created, tows are defined by digital element chains that are bundled together similar to fiber bundles, or tows. Each chain has a diameter for the spheres in the chain, which is used during the contact analysis and simulates the diameter of the fiber. This diameter is also used when creating the surface. To create a surface, VTMS creates a circle of a set diameter that is rolled along the boundary of the bundle of digital chains. As the circle, or roller, is rolled around the boundary, points of contact with the digital chains are recorded. These points are what defines the surface boundary. The roller is moved in stages down the length of the tow until the entire length of the tow has been covered. Figure A shows the cross-section of a bundle of digital chains and two rollers shown for demonstrative purposes.

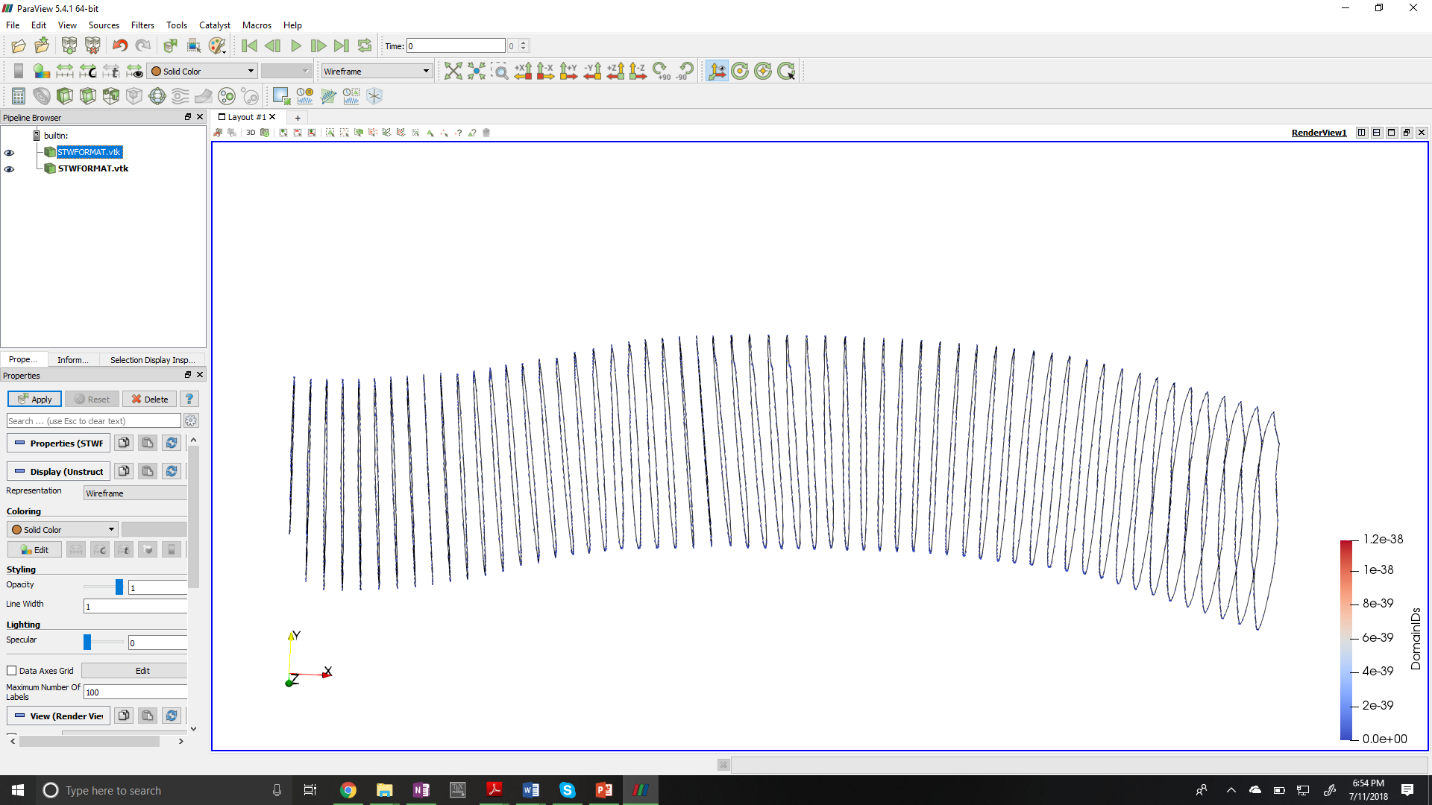
**Figure A: Digital chain bundle with varying diameter surface rollers**

Figure A shows two roller diameters to help illustrate a key point about this method for defining the surface. The smaller the roller that defines the surface, the less smooth the surface will be. The small undulations in the surface can cause small, disconnected regions of interpenetrations that will be discussed later. The smaller roller will more accurately represent the boundary of the tow but can also cause other issues that will also be covered. A large roller will lose accuracy representing the boundary but also result in a smoother surface representation that causes larger interpenetration regions that are easier to detect. Currently, we use the default value for the roller diameter which equates to a middle ground that has some smoothening to the surface.

Once the surfaces are created, VTMS exports the surface data. Currently there are two forms of exported surface data. We will discuss each format in detail.

## Standard Tow Format

One form of the exported surface data is the standard tow format. The standard tow format is exported after the relaxation steps that form the non-idealistic tow shapes. This format is made of stacks (cross sections) of the tow that are outlined by nodes. These nodes are the result of the surface roller method previously discussed and the number of nodes describing the cross-section outline is user defined. Each node has its coordinates stored in the file. The format of the file is by stack where each stack is listed in order along the length of the tow. In each stack, the nodes that make up the outline of the stack are listed with their coordinate. Figure B.1 shows the tow with the stacks normal (defined as the normal of the plane that all stack nodes lie on) in the plane of the page. Figure B.2 shows one stack from the tow with its normal perpendicular to the page. One important feature of this format is that the tow surfaces are open at the ends, resulting in a tube-like surface.



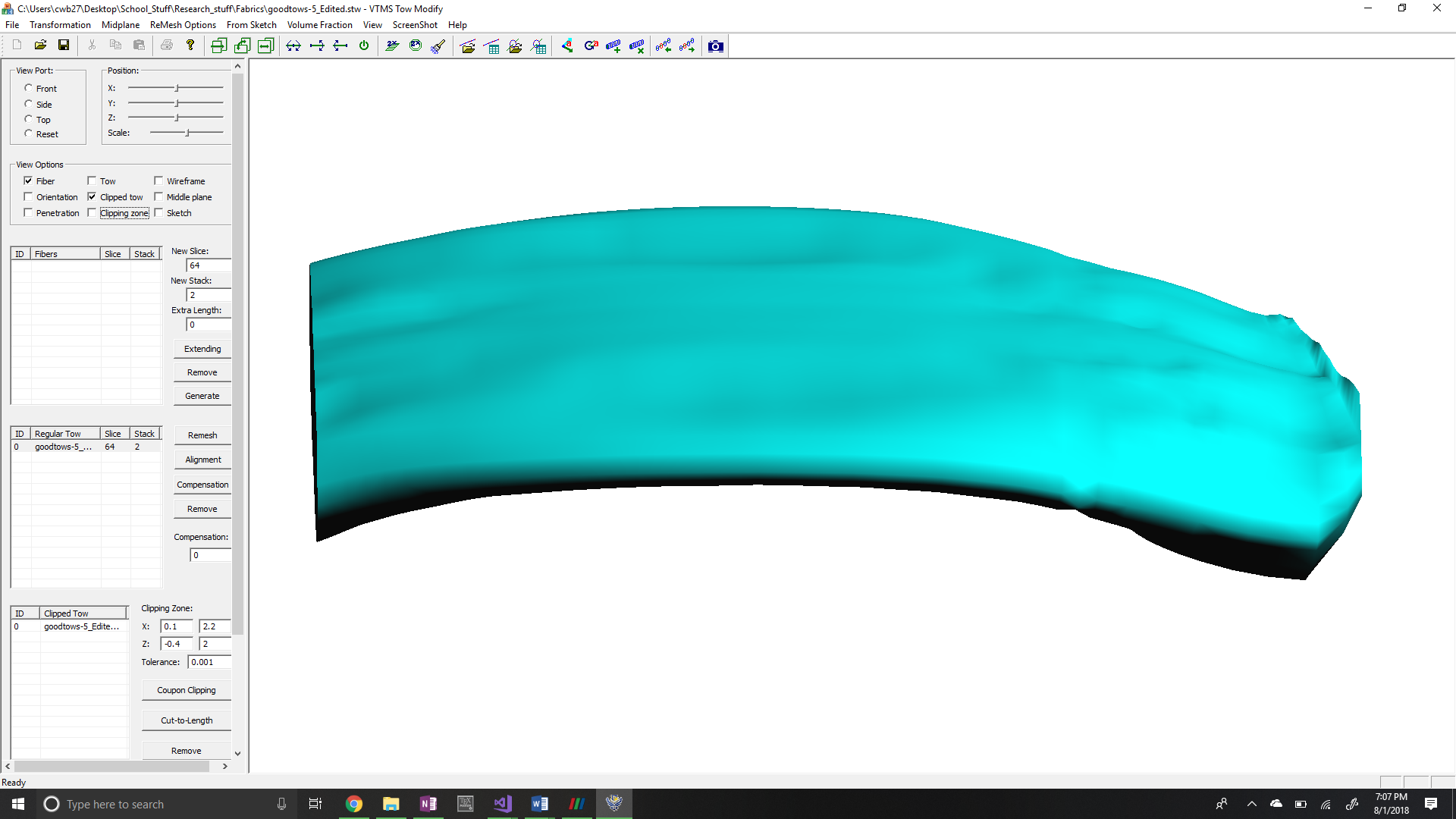
1. **Standard format normal to stack view**
2. **Standard format side view**

**Figure B: Views of VTMS standard tow format**

VTMS also has a method to visualize this data in its own viewer. It creates surface triangles that connect surface nodes to create a faceted surface. However, it does not output this information to its standard tow format. VTMS uses this format to save and load surfaces after all relaxation steps have been completed so that the surfaces may be recovered by VTMS. They can be loaded into the tow modify module of VTMS where they can then be clipped to a user-defined size. This clipping results in the other VTMS format.

## Clipped Tow Format

Once a standard tow is clipped in VTMS, the clipped surfaces can be exported in the clipped tow format. This surface format is very similar in layout to a finite element mesh in that the surface is defined by surface elements. The file exported stores each node and its coordinates followed by each element and its list of nodes that make up the corners of the element. The surface itself is primarily made up of triangular shapes but can be made up of polygons as well. It is important to note that the clipped tow surfaces are not mesh ready as they can have surface shapes that have more than 4 sides. Therefore, some preprocessing is required to make the surfaces finite element ready. One feature is that once a clipped tow is created, VTMS closes the ends of the surface creating a closed surface representation of the volume occupied by the tow. Figure C shows a clipped tow displayed in VTMS.



**Figure C: VTMS clipped tow**

The points that define the surface are the same points as the standard tow. When looking at the two surfaces side by side one can see that the stacks line up. This means that both surfaces are the same surface described in two formats and choosing one over the other does not change the surface being described.

# VTMS Surfaces Identification and Resolution

## Detection of Interpenetrations

A method was already in place inside of VTMS to detect interpenetrations. This routine uses a ray-surface intersection algorithm between nodes and surface triangles created in VTMS using the standard tow format of the surface. I plan on going into more detail for the thesis. Right now I am trying to cover only my contributions in detail so we can discuss it.

The main drawback to VTMS’ method is that it does not detect nor eliminate all interpenetrations. This is because of how VTMS detects interpenetrations using a point-by-point basis. AFRL works around this by not using a traditional finite element software to conduct their analysis but an experimental finite element code developed in-house. If we wish to use the tow surfaces that VTMS produces, this is the main issue we have to correct. A by-product of how VTMS detects interpenetrations is that once the interpenetrations are corrected, there exists a gap between the surfaces that should physically not exist if the tows come into contact. The surfaces should create a contact region between the two surfaces. This can be seen in micro-CT scans when two tows lie against each other. I am going to create this contact surface where the surfaces interpenetrate.

The first objective is to create a method that is more thorough in detecting interpenetrations than VTMS’ default method. To accomplish this, a modified minimum distance algorithm was chosen. The algorithm works as follows (refer to figure D and E):

1. First, a point and a surface are needed to compute a minimum distance. The point to test is from one tow and the surface needs to chosen from the opposing tow. Suppose there is a node **N** that needs to be tested against tow **T**. Choosing a surface works as follows:
   1. Any planar surface needs three points to define it. Therefore, three nodes from tow **T** need to be found to create a planar surface to test node **N** against. An algorithm was written that creates a temporary list for all the nodes on tow **T**. The algorithm loops over all the nodes in the list and records which node is closest to node **N**. Once a node is identified as being the closest node, it is added to another list for later use. It is then deleted from the temporary tow **T** node list so that it is no longer in the temporary list. The list is then iterated over multiple times, each time removing the next closest node until the three closest nodes from tow **T** to node **N** are found.
   2. Once three nodes are found (nodes **a, b,** and **c**), a surface element is found or created. There are two methods for doing this and the method depends on which surface format is being used.
      1. For a standard tow, the stack to which one node (see node **a** in Figure D) belongs is identified. An outward vector for this node from the tow surface is created. To do this, the stack centroid (node **d** in Figure D) is calculated from averaging the location of all of the nodes of the stack. Then a vector is drawn from this centroid to the node (node **a**). This vector defines an outward direction relative to the tow surface at this node. This vector is not perpendicular to the tow surface. This is done for the three captured nodes (**a**, **b**, and **c**). The average vector of all outward vectors is taken and is projected onto the normal vector of the surface containing all three of the closest nodes to node **N**. This results in an outward facing vector **x** (outward relative to the inside of the tow surface) that is normal to the surface that contains the three captured nodes. If the artificial element normal is used, there is no guaranteed that it is outward facing. Using the three nodes along with an outward normal, a temporary surface element can be created.



**a**

**d**

**Figure D: Node outward vector using stack centroid**

**x**

**y**

**N**

**Figure E: Surface element with outward normal vector and relative vector to node N**

**a**

**b**

**c**

* + 1. For a clipped tow, the algorithm is less involved. Because VTMS stores surface elements for a clipped tow, the element that uses all three nodes can be found and its outward normal can be used.

1. Once a surface element is found (or created), a vector (vector **y**) is created that runs from the average of the three captured nodes (**a**, **b**, and **c**) to the node of interest (node **N**). This gives the relative location of the node to the surface element.
2. The dot product is calculated between vectors **x** and **y**. This dot product will result in a positive number if node **N** is outside of the tow surface and negative if it is inside. If the node is inside of the tow surface, the node is marked as being an interpenetrating node.
3. This method is used for every node on both tows to collect all the nodes that are inside of an opposing tow.

The algorithm for detecting interpenetrating nodes for standard and clipped tows is very similar. The main difference is that a reference element has to be created for the standard tow format as there are no surface elements. Once all of the interpenetrating nodes have been found, an algorithm was developed to fix the interpenetrations for both formats.

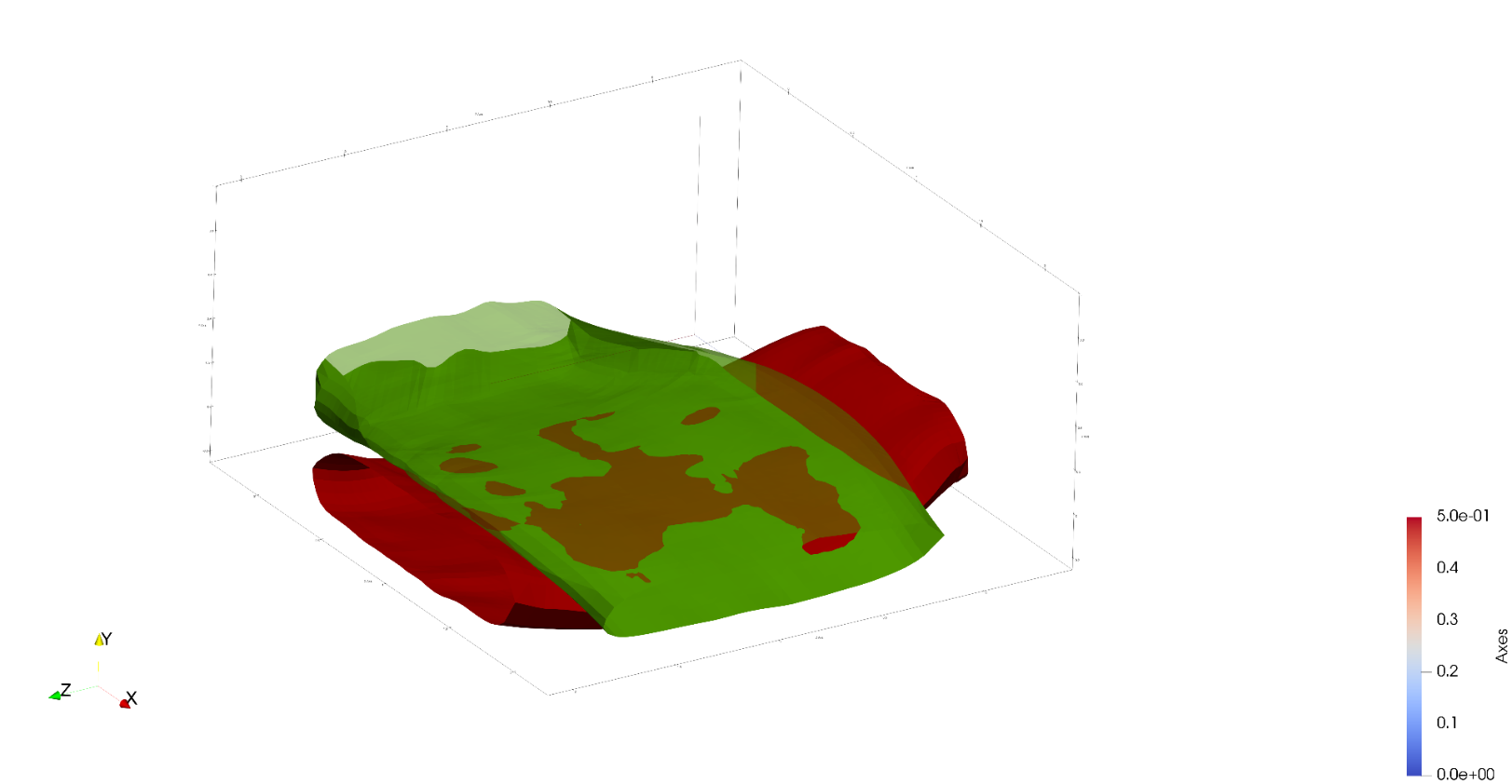
## Eliminate Interpenetration regions

Once the interpenetrated nodes were identified for each tow, a solution was implemented to solve the interpenetrations. The goal of the solutions is to remove the interpenetrations and create a compatible surface where the interpenetrations occurred. This is different than the method VTMS uses. I also plan to further discuss this in the thesis. We can have a separate discussion if you wish on this but I know we have discussed it before.

### Method 1: Artificial contact surface

The goal of this algorithm was to remove surface interpenetrations and establish a surface that is shared by both tows that will be referred to as a “contact surface”. A flat surface was chosen as it is the simplest surface to implement into the algorithm. The algorithm is as follows:

1. When two tows come in close proximity there can be multiple regions of interpenetrations that are disconnected (figure F). Therefore, the algorithm must account for this possibility. This is accomplished by implementing a sphere detection algorithm that creates a spherical detection region around a known interpenetration node. Any interpenetrating nodes that lie within this detection region are collected. These collected nodes are given a spherical detection region to identify other interpenetrating nodes. This continues until there are no more interpenetrating nodes being detected for this specific node group. This creates node groups that are part of the same interpenetration region but excludes other nodes that are not part of the same region. The result is distinct node groups for each region of interpenetration. This is important because if we were to use all interpenetrating nodes at the same time with a flat surface, there could be large distortions in the tow surfaces. Once there are defined node groups, the algorithm continues below for each individual region.



Disconnected interpenetration regions

**Figure F: Tow surface interpenetrations with emphasis on disconnected regions**

1. The algorithm begins by calculating the mean location of all the nodes using all of the collected interpenetrating nodes from the two tows. This acts as the central point of the interpenetration.
2. The algorithm then iterates over all of the interpenetrating nodes of one tow. For each node, the stack that it belongs to is saved in a list to reference later. If the stack already is in the list, then the algorithm continues to the next node in the list. We verify that the stack is not in a list using a simple find-in-list function. The result is a list that has every tow stack that has an interpenetrating node. A list is created for each tow surface so that a record of which stacks have interpenetrating nodes is kept.
3. Then, for each stack that has interpenetrating nodes, the centroid (or mean) of all the nodes that make up the stack is calculated. The purpose is to give a node that roughly determines a central point for the stack. We then order the stacks in their list so that they are in the same order as they are listed in the tow surface data. This ensures that as the list is iterated over, the stacks are in order. This important when approximating the tow path.
4. Next, the centroids of the interpenetrating stacks in the tow surface are recorded. We order the centroids in their own list such that the centroid and the stack it belongs to can be referenced at the same index for their respective list. Suppose in figure G VTMS outputs the stacks as [**A**, **B**, **C**, **D**]. We ensure that the order of the saved centroids is in the order [**a**, **b**, **c**, **d**]. This ensures that when iterating over the centroid list we know that the next centroid corresponds to the next stack in the tow surface.

***x***

**A**

**B**

**C**

**D**

**a**

**b**

**c**

**d**

**Figure G: Simplified stack representations with centroids and connecting vectors**

***z***

***y***

1. Now that all the information concerning the interpenetration region has been prepared, a surface needs to be created for the tows surfaces to share. A plane was chosen as an initial surface shape as it is the easiest to implement. The normal vector to the plane needs to be established to create the surface. To create this normal, we start by calculated the vector between centroid in the interpenetrating stack centroid list and the centroid that immediately comes after it. This is done for every centroid except the last in the list. This results in a collection of vectors [***x****,* ***y****,* ***z***] that describe the direction the tow moves along it length. This is what we refer to as the tow path. For each tow, we average the vectors that are created in this step to give the general axial direction of the tow in the region where the interpenetrations occur (figure H.a, vectors **q** and **r**). We can use these vectors to determine how the two tows are oriented relative to each other in the region where they interpenetrate. We take the cross-product of these two vectors to find a vector that is perpendicular to the two tows in the region (figure H.b, vector **s**). Because the average tow path vectors are the average of the tow path along the entire interpenetration region, the result of the cross product is not perpendicular to both tows along the entire region. However, as long as the tows are not excessively curved along the interpenetration region, the perpendicular vector is relatively orthogonal to both tows.

**q**

**r**

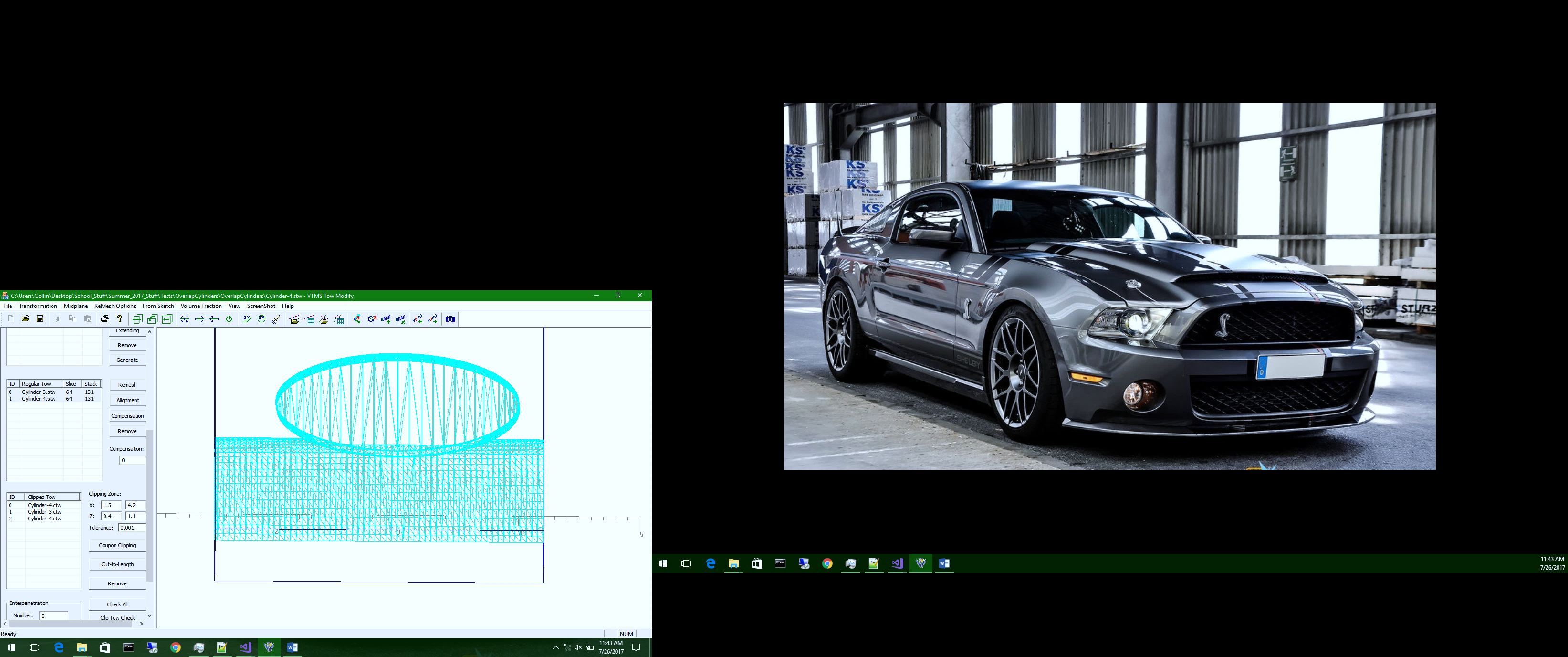
**Figure H: Visualization of plane normal vector calculation**

**q**

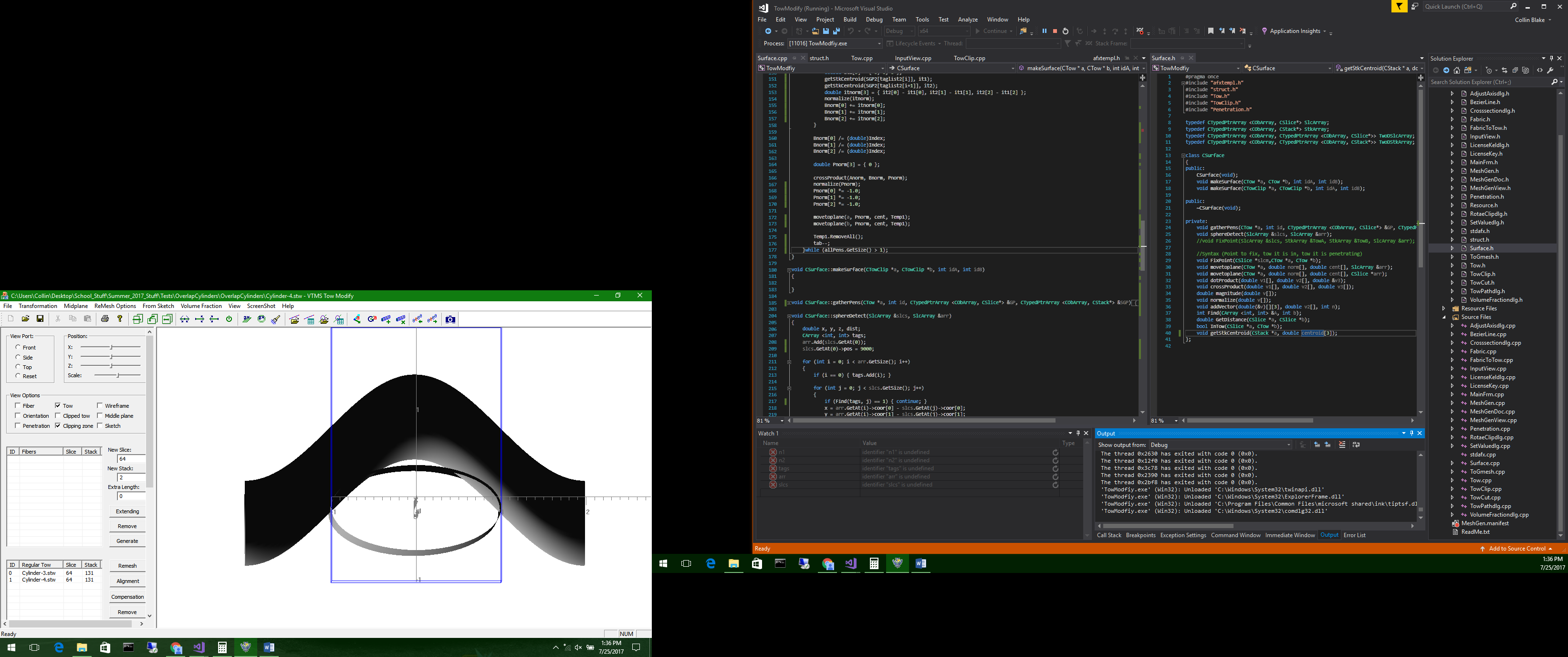
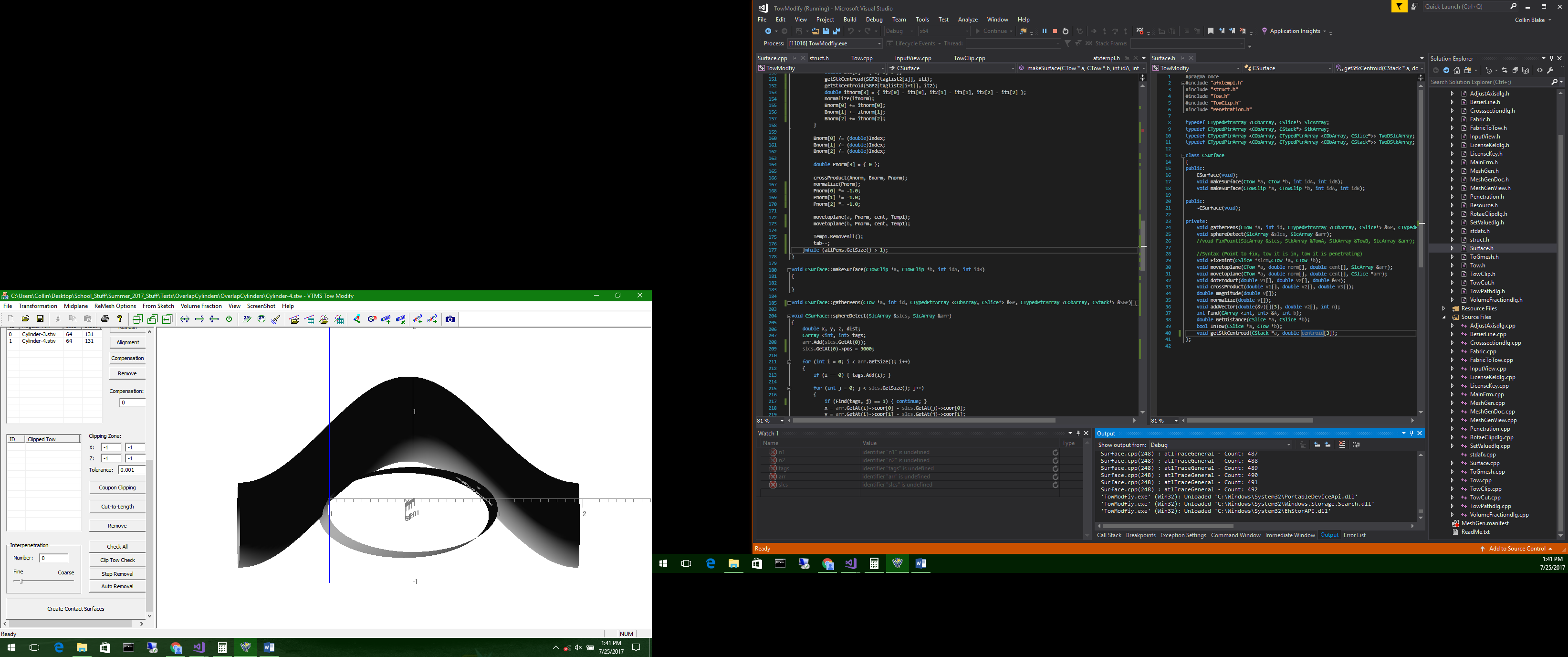
**r**

**s**

1. **Two tows with tow path vectors *q* and *r***
2. **Cross product of *q* and *r* resulting in *s* that defines a plane**
3. This perpendicular vector, along with the average, or centroid, of the interpenetration nodes, is used to create a flat that defines a plane along which the tows will share a surface. This plane is not orthogonal to both tow paths everywhere due to the curvature of the tow surfaces. However, it is a reasonable approximation for the interpenetration region.
4. Once the surface is defined, a vector from the centroid of the interpenetrating nodes to each interpenetration point is calculated. This gives the relative location of any point to the reference point (centroid) on the shared surface plane. This vector is then projected onto the normal vector of surface. The resulting vector projection (often referred to as the minimum translation vector, or MTV) accurately describes the exact translation the interpenetrating node must undergo to be on the plane. The node is then move along this vector to the plane.
5. After all the nodes have been moved, a check is made to ensure that all of the interpenetrating nodes are no longer interpenetrating. The surfaces are then saved and exported so that they can be used later.
6. There exists another case where an interpenetrating node does not detect any other interpenetrating nodes in its detection sphere. When this occurs, another algorithm begins.
   1. The algorithm conducts a similar process to how the nodes are identified as being interpenetrating (refer to figure E). The node finds the three closest nodes on the opposing tow and creates an artificial surface element to act as the interpenetrating node’ shared surface to be pulled to. The outward normal of the artificial surface element is calculated using the same method as when it is calculated during the detection phase. This normal is what the interpenetrating node uses to project onto to find its minimal translation vector. Then the interpenetrating node is moved to the surface.
7. The result can be seen below: I’m going to have to put in a couple hours of development to get better figures in paraview.



**Figure J: Mesh of tows with a shared surface**



**Figure I: Interpenetration and resolution of simple tows**

1. **Two surfaces interpenetrating**
2. **Interpenetrations fixed with common surface**

This method has undergone a couple of iterations and improvements. The initial version did not account for multiple disconnected regions and created the same plane for all regions. This resulted in large deformation of the tow surface for interpenetration regions that were relatively far from the mean location of all the interpenetrating nodes. These disconnected regions come from the surface not being smooth, which is a result of a small surface roller diameter. If the roller was larger, the surface would be smoother and there would be a larger, connected region of interpenetrations. Disconnected regions led to the implementation of the spherical searching algorithm that collects all of the interpenetrating nodes of the same region. Next, we discovered that there existed interpenetrating nodes that were not part of a region but rather a single node interpenetration. A solution was developed and is similar to the interpenetration detection algorithm in how it detects the minimal translation vector to move the node to a surface.

This method works great for multiple penetration regions between tows. The currently limitations are that this algorithm currently only creates flat surfaces between tows and does not ensure compatibility of tow meshes along contact region. This methods merit is that when the interpenetrations are fixed, the surface nodes are not simply moved beyond the boundary of the surface they penetrate. Many methods are satisfied with moving the interpenetrating nodes outside of the surface by a minimum distance. This method forces the nodes to lie on a surface so that the tows are in contact where their surfaces were interpenetrating. The result is a more realistic interaction between tows where they are in close proximity. This surface can then have some form of contact interaction imposed along this surface.

This method does have faults. When the interpenetration region is large or the tows are curved, the planar surface can change the topology of the tows excessively. Also, because the method uses a sphere-based detection algorithm for finding interpenetrating nodes, disconnected regions of interpenetrations can be grouped together unintentionally if the detection sphere is too large. This can lead to misalignment of the shared surfaces and can significantly modify the tow surfaces. We considered creating a master surface and a slave surface between two tows as it would eliminate the need for defining a surface between the two tows. However, we became concerned that if one tow in a fabric was set as a slave for every interaction, the tow volume could be affected greatly from the original volume. This would have a worse effect on the material properties of the fabric during analysis than if the surface approximation between the tows is slightly misaligned.

Enforcing compatibility is another issue with this method. It is difficult to find a node from each tow to pair and make a compatible mesh because of the stack description style. The stacks of two tows that are orthogonal rarely have nodes in close proximity. Also, moving surface nodes can result in high-aspect ratio elements and collapsed surface elements. For all these reasons, a different method was developed using the clipped tow format.

### Method 2: Node-to-Node half distance compatibility solution

This method came from the need to solve the issue of the lack of compatibility between two tows along a shared surface. The method for detecting the interpenetrations between two polygon surfaces is the same as for the standard tow. The only difference is that an artificial surface element does not need to be created. Instead, the element that contains the three found nodes is chosen and the normal recorded. To solve the interpenetrations, a separate method is implemented for the clipped tow format.

1. An interpenetrating node **a** is chosen from one surface (tow **t**). The starting node is arbitrary. Then, every interpenetrating node from the opposing tow (tow **u**) is iterated over, recording the closest node (node **b**) to node **a**.
2. A vector is created that runs from **a** to **b**. This vector gives both the direction and distance that node **b** is from node **a**. This vector is used to determine where the two nodes will be moved to so that they coincide.
3. Using the vector that connects **a** and **b**, we calculate the distance between the two nodes and divide it by two. This gives the halfway distance between the two nodes. To ensure that we do not change the topology of the surface drastically, we move both nodes to the halfway point between the two. This point lies on the vector that connects **a** and **b** and is halfway between the two nodes. This accomplishes both the task of removing interpenetrations and enforcing compatibility for the two nodes.

**a**

**b**

**t**

**u**

**a,b**

1. **Node *a* and node *b* move to halfway point**
2. **Node *a* finding the closest node (*b*)**
3. **Nodes after being moved to halfway point**

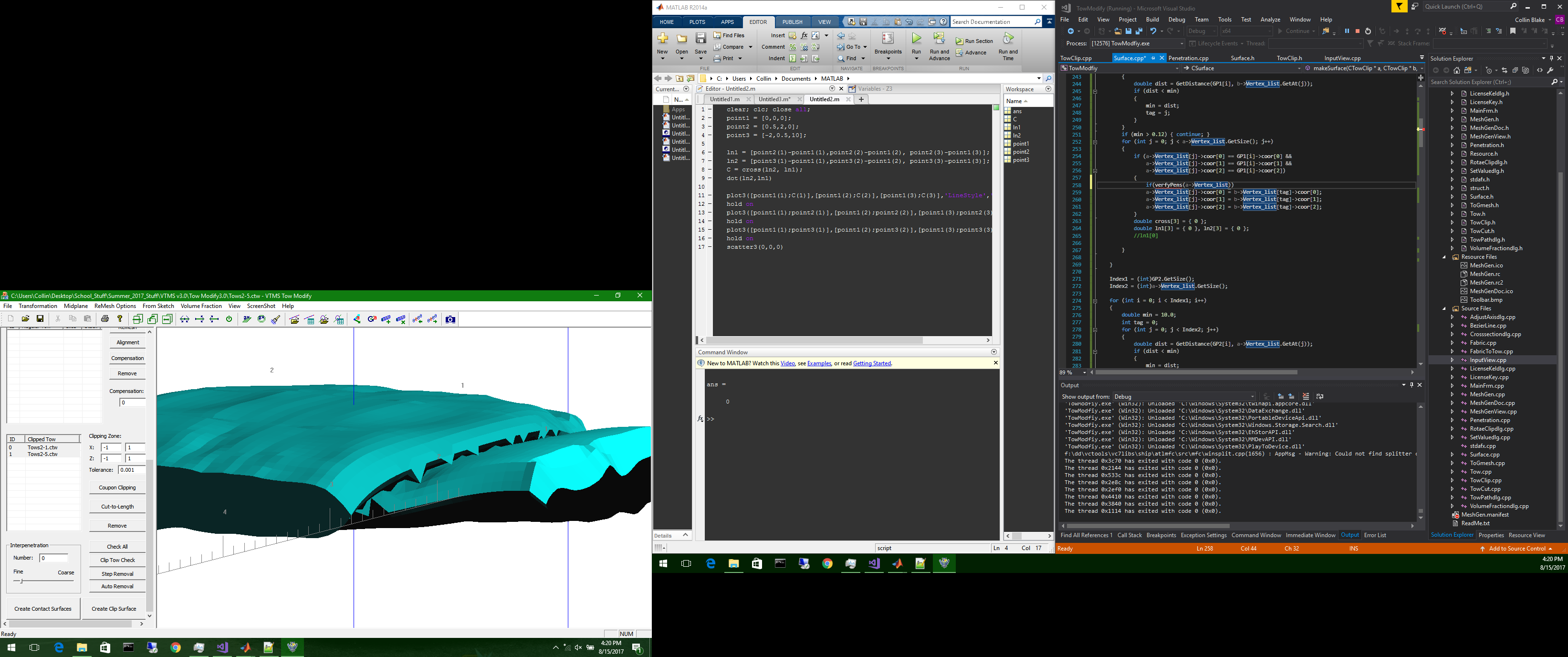
**a,b**

**Figure K: Stages of clipped tow interpenetration resolution**

**d**

1. Once compatibility is enforced for **a** and **b**, they are removed from their respective interpenetrating node lists. This is to ensure that they are not detected and moved again. The method is used until one interpenetrating node list has had all of its nodes removed. Because the interpenetrating node list for the two tows are not usually the same size, the algorithm must handle the remaining nodes in the list that is larger than the other.
2. If the remaining interpenetrating nodes are from tow **u**, the algorithm identifies the closest node (node **d**) on the opposing tow **t** to a node from tow **u**. The found node from tow **t** does not have to be interpenetrating since all interpenetrating nodes from tow **u** have already been resolved. The interpenetrating node **c** is then moved to node **d**. This is done for all the remaining interpenetrating nodes from tow **u**. The result is no interpenetrating nodes from either tow.

The result is a non-planar contact region between the two tows. The merit of this method is that when it removes the interpenetrations it simultaneously enforces compatibility. It uses a point searching algorithm and simple vector calculation. The method is also unpolished as can be seen below where some nodes are moving when they are not required to, resulting in very jagged edges. One reason the nodes may be moving is that they have been improperly identified as interpenetrating and would require improvement on how the interpenetrations are detected. There is no protection against collapsing elements as the nodes move to each other. Therefore, this method is not a full solution but rather the initial implementation of the method. This attempt at a solution was a significant factor in deciding that another method for both identifying and resolving interpenetrations was necessary. During the preliminary study of the interpenetration topic, both polygon and non-polygon surfaces were evaluated for their potential to be applied to this interpenetration problem. This method was the turning point for the need for a different approach and to use a non-polygonal surface detection method.



This concludes the accomplished methods for the discrete representation. The figures below I think could be used somewhere as well. I’m not sure where they could go but I’m saving them for now.

