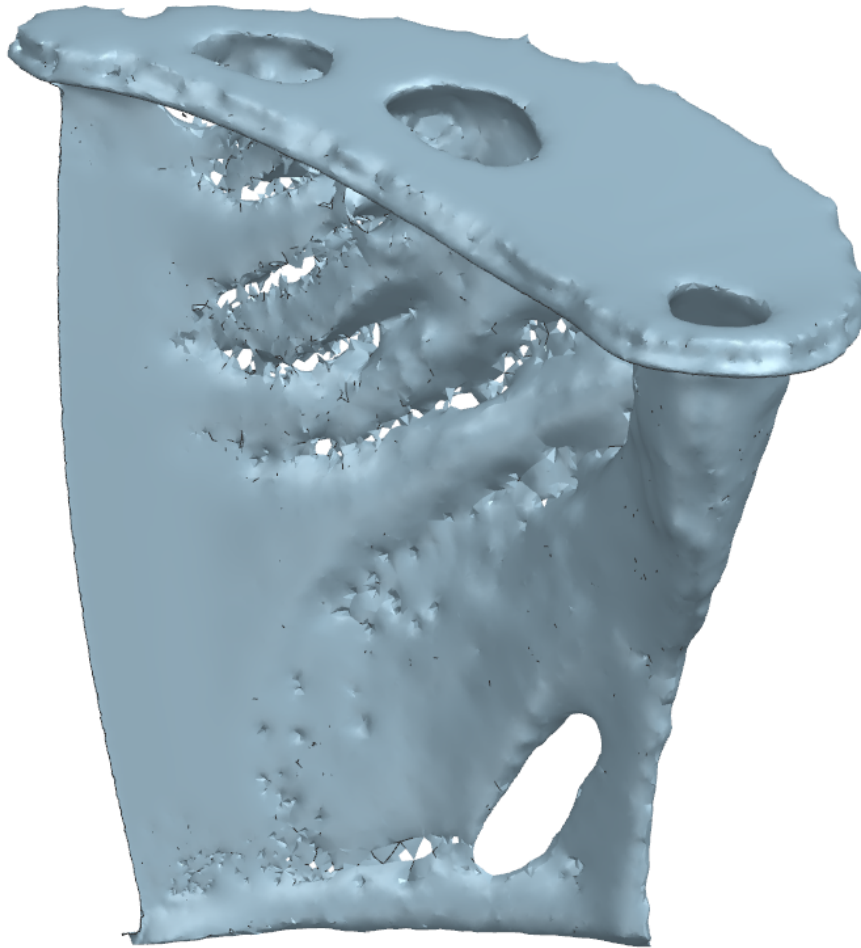




**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



# **STL files to CAD format for topology optimized structures**

Master's thesis in Product Development

Gopinath Siva

---

DEPARTMENT OF INDUSTRIAL AND MATERIAL SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2022

[www.chalmers.se](http://www.chalmers.se)



MASTER'S THESIS 2022

# STL files to CAD format for topology optimized structures

GOPINATH SIVA



Department of Industrial and Materials Science  
*Division of Product Development*  
Name of research group (if applicable)  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden 2022

STL files to CAD format for topology optimized structures  
GOPINATH SIVA

© GOPINATH SIVA, 2022.

Supervisors:

- Tina Hajali, Department of Industrial and Materials Science
- Josefine Tenselius, Siemens Energy AB
- Jan-Erik Lundgren, Siemens Energy AB

Examiner:

- Prof. Ola Isaksson, Department of Industrial and Materials Science

Master's Thesis 2022  
Department of Industrial and Materials Science  
Division of Product Development  
Chalmers University of Technology  
SE-412 96 Gothenburg  
Telephone +46 31 772 1000

Cover: STL file of the gas turbine guide vane showing geometrical issues post topology optimization

Typeset in L<sup>A</sup>T<sub>E</sub>X  
Printed by Chalmers Reproservice  
Gothenburg, Sweden 2022



# Abstract

Recently, topology optimization technique has acquired lot of interest across the industries because of its ability to develop lightweight efficient designs that also brings other functional benefits. But one of the difficulties is the complexity of the design to manufacture. Recent developments prove that complex designs can be easily manufactured using additive manufacturing. With the synergy of topology optimization and additive manufacturing, industries are aiming to gain more benefits. Siemens Energy, with the knowledge and capability to manufacture components using additive manufacturing, is now interested in implementing non-traditional design technique like topology optimization to take the next leap in the development of its products. In the process of implementation, Siemens Energy face some issues in handling the STL geometries from topology optimization. The STL files possess some defects that prevent the engineers to use the geometries similarly to other files handled in CAD system. This thesis is to develop a method to repair the STL issues within Siemens NX environment and transform the STL file to CAD format. The method would assist design engineers at Siemens Energy to fix the STL issues in future that enable the STL geometry to be used similarly to other CAD files. The thesis also explores the feasibility of automation of the repairing process in order to reduce the manual labour.

This was achieved by the exploratory study that began with literature review to explore about STL file format and various STL errors followed by exploration of Siemens NX CAD in the direction to repair STL. Later automation feasibility was investigated in NX. The method to fix STL issues is devised using use cases. STL issues were successfully addressed to develop a CAD format of the models using NX CAD. The limitations and feasibility of automation in NX was studied. NX Open tool with C# language was used in the project. A method that shall help the design engineers to fix the STL issues is proposed. In the end, future work towards automation and other possible explorations are summarised.

Keywords: Polygon modeling, STL error repair, Siemens NX CAD, Automation in NX



# Acknowledgements

First and foremost, I would like to thank Simon Hellborg for giving me the thesis opportunity and for the constant support throughout the project. Special thanks to Jan-Erik Lundgren for his continuous support, had been a constant support for sharing ideas and explaining various technical concepts. His valuable feedback and insights were significant in the project. I am thankful to Josefine Tenselius for supervising my work and for helping me to get trained in NX. I want express my deep thanks to Vivekanandan Shanmuga Sundaram for the enlightening discussions, support and friendship. I would like to extend my gratitude to the entire Additive Manufacturing team and all the colleagues who supported me in this journey.

In the University side, I want to thank Tina Hajali, my thesis supervisor at Chalmers University of Technology for her guidance and tips that she provided throughout the project. I would like to thank Ola Isaksson for accepting my request to be my examiner and for his guidance throughout the project.

I wish to express my deepest gratitude to Prakash Thiyagarajan for all his support and patience to help me in NX Open. It would never be possible to program within this short duration without his willingness to share his expertise.

- Gopinath Siva,



# Nomenclature

3MF	3D Manufacturing Format
AM	Additive Manufacturing
AMF	Additive Manufacturing Format
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASTM	American Society for Testing and Materials
CAD	Computer Aided Design
CAE	Computer Aided Engineering
DfAM	Design for Additive Manufacturing
FE	Finite Element
GRIP	Graphic Interactive Programming
IDE	Integrated Development Environment
JT	Jupiter Tessellation
KBE	Knowledge Based Engineering
PBF-LB	Powder Bed Fusion - Laser Beam
PLY	Polygon File Format
SGT	Siemens-Energy Gas Turbine
SLA	Stereolithography
SNAP	Simple NX Application Programming
STL file	STereo-Lithography file
TO	Topology Optimization

---

# Contents

<b>Nomenclature</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem Area . . . . .	2
1.3 Aim . . . . .	3
1.4 Specifications of Research Questions . . . . .	4
1.5 Demarcation . . . . .	4
1.6 Research Ethics . . . . .	5
<b>2 Theory</b>	<b>7</b>
2.1 Topology Optimization . . . . .	7
2.1.1 Topology Optimization workflow . . . . .	7
2.2 STL file . . . . .	9
2.2.1 Rules for STL files . . . . .	12
2.2.1.1 Vertex to Vertex rule . . . . .	12
2.2.1.2 Facet Orientation rule . . . . .	12
2.3 Issues in STL files . . . . .	13
2.4 Available STL repair techniques . . . . .	17
2.5 Available STL repair tools . . . . .	17
2.6 Other File Formats Available . . . . .	18
2.7 Additive Manufacturing . . . . .	19
<b>3 Research Methodology</b>	<b>21</b>
3.1 Literature review . . . . .	21
3.2 Interviews . . . . .	21
3.3 Forum Discussion . . . . .	22
<b>4 Tools</b>	<b>23</b>
4.1 Types of facet bodies . . . . .	23
4.2 Commands to handle STL issues . . . . .	26
4.2.1 Polygon modeling task environment . . . . .	26
4.2.2 Cleanup facet body . . . . .	26
4.2.3 Snip facet . . . . .	28
4.2.4 Fill Hole . . . . .	28
4.2.5 Smooth Facet Body . . . . .	30
4.2.6 Other Useful commands . . . . .	31

4.3	Automation . . . . .	32
4.3.1	NX Capabilities . . . . .	32
4.3.2	Automation tools used . . . . .	34
<b>5</b>	<b>Experimental Research</b>	<b>37</b>
5.1	Importing STL file . . . . .	38
5.2	Analyze the input . . . . .	38
5.3	Preserve the non-design space . . . . .	40
5.4	Fix the STL issues . . . . .	42
5.4.1	Repairing use case 2 . . . . .	43
5.4.2	Repairing use case 3 . . . . .	46
5.5	Automation . . . . .	49
5.5.1	Redundant facets . . . . .	50
5.5.2	Self Intersecting facets . . . . .	50
<b>6</b>	<b>Discussion</b>	<b>53</b>
<b>7</b>	<b>Conclusion</b>	<b>57</b>
<b>A</b>	<b>Thesis Workflow</b>	<b>I</b>
<b>B</b>	<b>Forum Discussion 1</b>	<b>III</b>
<b>C</b>	<b>Forum Discussion 2</b>	<b>VII</b>



# 1

## Introduction

For decades, forming and casting were used as the primary manufacturing method followed by subtractive manufacturing to process the manufactured components. While the forming and casting process provided the overall volume of the product, subtractive manufacturing processes such as milling, turning were used to provide the desired shape by removing material. In 1987, an alternative approach to this traditional manufacturing called Additive Manufacturing (AM), also known as 3D printing, emerged where successive layers of material were added along one direction to get the desired shape [1] of an object. This novel technique brought potential advantages along with its own shortcomings.

One of the benefits of AM is the ability to manufacture complex designs. AM is capable of manufacturing components that consume more energy if manufactured conventionally at a lower expense. Also AM is capable of manufacturing components that cannot be manufactured using conventional manufacturing techniques [2]. In parallel, the researches in the design domain revealed that the analysis lead design techniques capable of generating more effective designs that are lightweight but complex to manufacture [3]. Topology Optimization (TO) is one such design technique that is capable of developing designs with best distribution of material for the given set of loads and constraints. Provided the complexity of the optimized design, TO becomes the natural design technology for AM as it fully exploits the potential.

Siemens Energy is looking to use the advantage of AM and TO. But a potential problem is identified that limits Siemens Energy from fully utilizing the benefits. This section will describe the background of the problem followed by aim, limitation, delimitation and research questions.

### 1.1 Background

Siemens Energy is a global company that provides energy generation and energy harnessing solutions for small, medium and large-scale organizations [4]. Siemens Energy manufactures gas and steam turbines, hybrid power plants operated with hydrogen, power generators and transformers. From Sweden, medium gas turbines are designed, manufactured, assembled and delivered to customers all over the globe. All the components of the gas turbines were initially manufactured using subtractive and joining processes. Later, AM department was established in the organization

with the focus to replace the conventional manufacturing method with AM for certain components of gas turbine [5].

Initially AM was introduced to build prototypes in shorter lead time. By identifying the potential, application of AM was expanded to eliminate the labor intensive manufacturing of complex components that required more precision and consistency. The burner front of gas turbines were the first point of interest since it had complex system responsible for fuel feeding, internal cooling and other more functions. Initially, one EOS M280 Powder Bed Fusion - Laser Beam (PBF-LB) printer was installed at Siemens for the repair work of conventionally manufactured burners [5]. Later complete burner front of SGT700 was developed using AM. The traditionally manufactured burner front composed of 13 individual components with 18 welds were united to a single piece AM component as shown in the Figure 1.1.



**Figure 1.1:** SGT-700 burner with additively manufactured burner front

The new design of the burner front improved the cooling, allowing lower operating temperature which contributed to a longer lifespan of components and the gas turbines [5]. Siemens Energy is looking to use AM to produce other components of the gas turbines to improve the functionality of the components, example by improved cooling.

## 1.2 Problem Area

Since manufacturing constraints of traditional manufacturing is no longer applicable to AM, AM facilitates more design and manufacturing freedom. This enabled Siemens Energy to investigate new design methods like TO. By using TO, the amount of material required to build a component is optimized but also the created design is capable of fulfilling other functional demands.

Before engaging in TO, the Computer Aided Design (CAD) were manually created by design engineers at Siemens Energy. Siemens Energy used NX CAD as the CAD tool and the native CAD models were in part format. The native CAD models use Non-Uniform Rational B-Spline (NURBS) to represent the 3D object. During printing process, the native CAD files were converted into Stereolithography (STL) file format using tessellation process because the 3D printing systems use STL as the input format. A STL file describe the outer surface of an 3D object using triangular facets [6]. Usually an STL file developed from native CAD file encloses the boundary of 3D CAD model without any problem. Therefore, the STL file can be directly used for printing.

In case of designs from TO, the STL files are not created in the similar way. The STL files are developed as a result of Computer Aided Engineering (CAE) Analysis. The topology of the input CAD represented as NURBS is tessellated into user defined mesh on which CAE analysis is performed to obtain optimised geometry. The optimised geometry is extracted from the CAE solver as a STL file in this thesis work. This STL file when inspected consist of errors that prevent extraction of solid model from the geometry and also consists of unrealistic or infeasible solutions.

It is necessary to obtain the solid geometry of optimized model to directly use with other CAD files. The STL files do not provide rich description of geometry as NURBS. The tessellation of geometry into triangular facets give rise to erroneous STL geometries that do not provide information on volume, mass and it is not possible to integrate STL models with other models in CAD environment. So the underlying need is to investigate the issues in STL file so as to enable Siemens Energy engineers to handle the topology optimized geometry together with other geometries in Siemens NX CAD environment.

### 1.3 Aim

According to Siemens Energy, the output of topology optimization, the STL format, contains a number of issues and infeasible solutions such as load transfer through point contact, self intersecting facets. Despite of the geometric issues, the STL files can be printed but the output performance like structural rigidity, fluid flow through channels in the printed component will not be desirable. Therefore, the STL file need to be repaired and the issues have to be fixed to develop an error free 3D solid geometry. The aim of the thesis work is to investigate to find a practical method and make recommendations on how to convert 3D geometries from STL format to a NX representative model. Also investigate on automation of the repairing process and develop a technique that shall help in the fixing process. The overall aim of the thesis can be put into following points:

1. To analyze the Topology Optimized model and study the problems associated with the generated STL file (in geometry)
2. To investigate and recommend the best practice strategies to generate a near net

shape CAD model from STL files

3. Investigate on possibility of automation of STL repairing technique that will aid Siemens Energy engineers in converting the STL file to NX CAD geometry

## 1.4 Specifications of Research Questions

To be focused on the topic, the thesis work aims to answer the following research questions

1. What are the issues in topology optimized STL file that prevent design engineers to use it along with other CAD files in NX environment?
2. What are the steps involved in development of NX CAD model from the optimized STL file?
3. Is automation of STL repair possible? If possible, how to automate the STL repair technique?

## 1.5 Demarcation

The thesis work is performed for 20 weeks with certain boundary limits which are as followed below.

1. The scope of project is limited to propose a method to convert the STL geometry to CAD geometry in NX. Investigating on method of Topology Optimization or additive manufacturing is not included in the scope.
2. The reference to the original STL is lost while using a third party tool. Also there is a possibility of data loss or depreciation of STL quality when an third party tool is used to repair STL model and then bringing into NX environment. Therefore one other limitation is to repair the STL file within Siemen's NX environment using its built-in utilities.
3. The thesis involves development of solid NX CAD geometry from STL file. This does not include CAE analysis nor physical prototyping of the developed CAD model.
4. Throughout the thesis work, NX CAD version 2007 is used [7]. The findings from this work may or may not be applicable to other versions of NX CAD.

## 1.6 Research Ethics

The thesis work follows the six principles of Economic and Social Research Council's framework [8] for research ethics. The confidential information shared by the participants are respected and not shared outside the research group. Also the anonymity of the research participants are respected by not mentioning the personal information. The report uses data and facts of Siemens Energy with the granted permission and uses data from external references that are cited to acknowledge the work of the original author.



# 2

## Theory

For this thesis work, the output of topology optimization is a STL file. In ideal case, the STL file obtained from TO shall be directly used as input for Additive Manufacturing. This can be illustrated as a process flow as shown in the figure 2.1



**Figure 2.1:** Ideal process flow between TO and AM

The thesis work is more focused on working with the STL file. So this chapter will elaborate about STL file and briefly describe TO and AM processes to provide an holistic understanding.

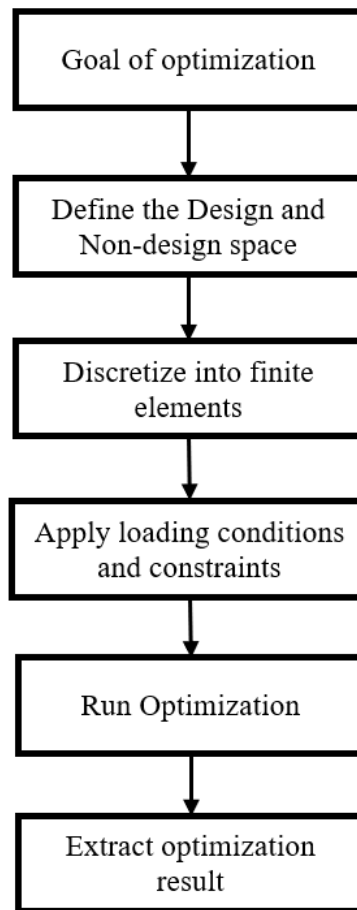
### 2.1 Topology Optimization

A conventional product development process starts with concept sketch followed by CAD design, then the CAD model is verified by performing iterations of CAE analysis and the approved design is prototyped and released for manufacturing. This human intuition based designing approach is more time consuming and usually iterations of CAE analysis and CAD model revisions need to be performed to obtain the desired result [9]. With the help of TO, the number of manual CAE iterations and CAD revisions can be minimized. According to Liu.J. and To.A.C., Topology Optimization is defined as the mathematical method that optimizes material layout within a given design domain based on loading and boundary conditions to achieve desired performance [10]. TO also facilitates in creating new ideas and concepts that outmatch the human-based design [11]. These design outputs of TO are usually unconventional and difficult to manufacture using subtractive manufacturing [2]. In recent developments, various manufacturing constraints are made available in TO solver tools so that the unconventional results can be manufactured using the specific manufacturing method.

#### 2.1.1 Topology Optimization workflow

A typical topology optimization problem starts with the objective or goal. Generally, optimization can be applied to any continuous field that can be represented mathematically. Typical examples are fluid problems and structural problems. For

structural applications, maximize stiffness and minimize mass are some of the examples of objectives that can be defined. For fluid problems, minimize pressure drop and homogenization of cross section velocity are examples of the objective functions [12]. In this study, the focus is on fluid problems. The optimization problem is subjected on a design space which is a bulk volume of material that to be optimised based on fluid flow through the material. A non-design space is a region that is excluded from optimization. For example a hole for holding a bearing is a non design space where changing geometry of the hole is not permitted. The loading conditions and constraints are defined on the non design space. The loading condition can be a force, pressure or fluid flow rate depending on the problem. Then the FEA solver is run for the desired objective. The FEA solver uses the different loading conditions to get the result. Based on the result, the material layout is changed and the algorithm is run in iterations till the result is converged. The workflow is depicted in the figure 2.2.

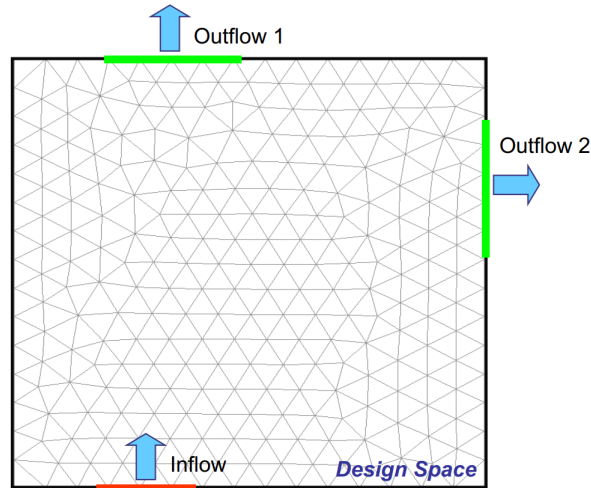


**Figure 2.2:** Workflow of Topology Optimization.

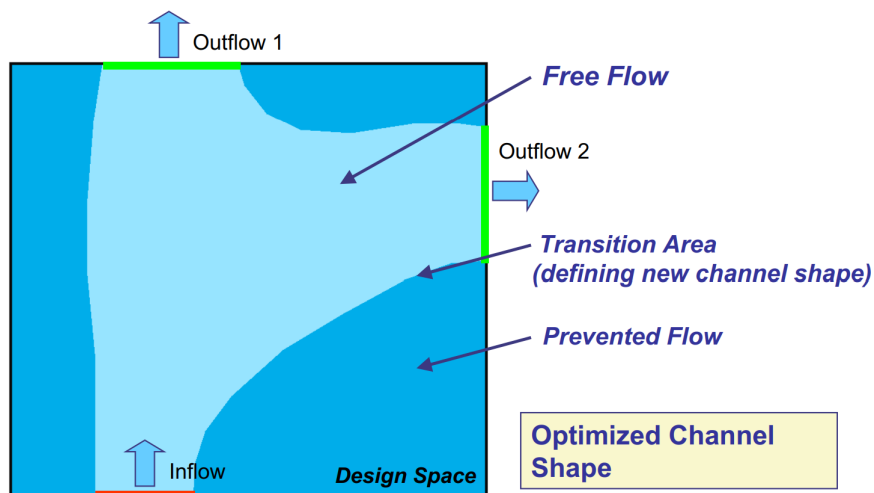
An example of 2D CFD (Computational Fluid Dynamics) based TO is illustrated in the figures 2.3 and 2.4 [12]. The square box in figure 2.3 represents the design space of a channel. The triangles represent the finite element mesh. The red region represent the fluid entry into the channel and the green region represent the fluid exits. The figure 2.4 shows the result of topology optimization. For the given



boundary conditions, the result is converged and the dark blue area in the design space represent the presence of material and the flow of fluid is restricted. The light blue region represent the void and fluid is allowed to flow through this area.



**Figure 2.3:** Input conditions of TO [12]



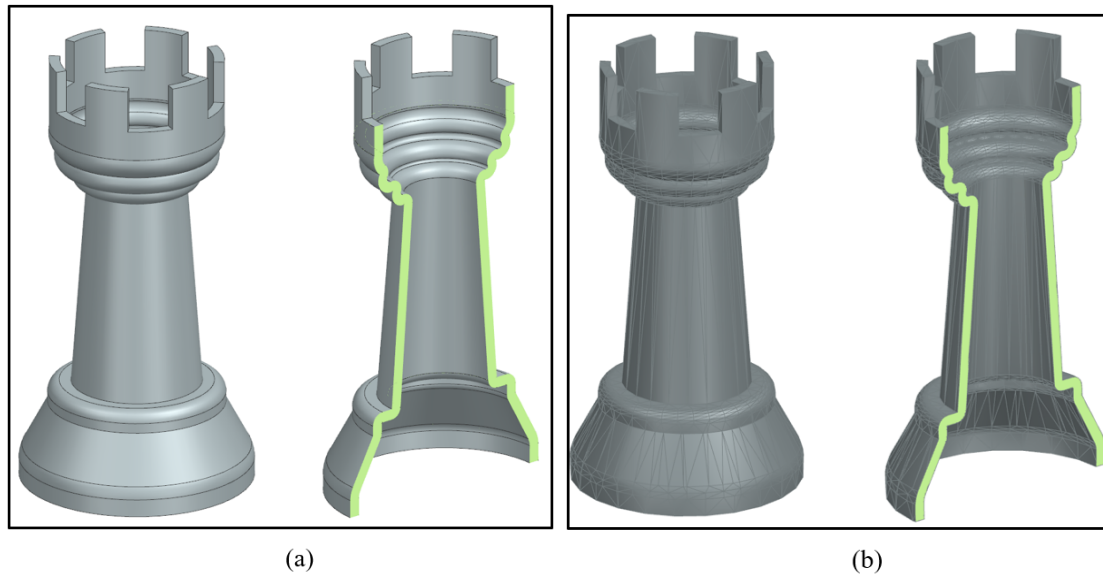
**Figure 2.4:** Result of TO [12]

The result of topology optimization can be visualized as an image as shown above and can be extracted from the solver as a basic 3D geometry if it is a 3D TO problem. Common file formats of such basic 3D geometries are STL, 3MF (3D Manufacturing Format) and PLY (Polygon File Format).

## 2.2 STL file

Stereolithography (SLA) was the first commercially available rapid prototyping technique, developed by 3D Systems, and STL file was used as the input file format for printing. Ever since STL file format has been widely used across AM industry and

it is considered to be the *defacto* industry standard [13] [14]. An STL file consists of tessellated triangles that describe only the periphery geometry of a 3D object with hollow interior[15] [16]. STL is also referred to as "Standard Tessellation Language" and "Standard Triangle Language" [6].



**Figure 2.5:** (a) Regular CAD model represented using NURBS along with the cross section view. (b) STL geometry of the same CAD model along with the cross section view (with facet edges shown). In NX CAD, Show cap option is enabled in section view (green color). The cap in section view of STL denotes that the STL file is watertight

While creating a STL file from a solid CAD geometry, the exterior of the solid object is divided into finite number of triangles using tessellation algorithm. The built-in algorithm constructs the simple boundary representation that covers the surface of the object with triangular facets[17]. Therefore, a mesh of tessellated triangles represent the solid geometry (see figure 2.5). Attributes such as units, layer, color that are available in CAD systems are ignored during tessellation process [18]. This allows the model to be light-weight, simple and easy to print.

The tessellation information of the facet can be stored in two different formats.

i) ASCII encoding :

American Standard Code for Information Interchange (ASCII) format is a character encoding format that represents data as readable text and can be read by any text editor. STL file exported as ASCII format encodes the triangle normals as  $n_x$ ,  $n_y$ ,  $n_z$  and triangle vertices as  $v_1$ ,  $v_2$ ,  $v_3$ . A typical ASCII format of a facet and an entire STL file is shown in figure 2.6.

<pre> facet normal  n<sub>x</sub>  n<sub>y</sub>  n<sub>z</sub>   outer loop     vertex  v<sub>1x</sub> v<sub>1y</sub> v<sub>1z</sub>     vertex  v<sub>2x</sub> v<sub>2y</sub> v<sub>2z</sub>     vertex  v<sub>3x</sub> v<sub>3y</sub> v<sub>3z</sub>   endloop endfacet </pre>	<pre> solid   facet 1 description   facet 2 description   .   .   . endsolid </pre>
<i>facet description in ASCII</i>	<i>STL description in ASCII</i>

**Figure 2.6:** ASCII description of each facet in a STL file is shown in the left. The vertex coordinates and normals are denoted as floating point numbers. An STL consist of numerous facets and each facet information is organized in linear fashion as shown on the image in right. Every STL file starts with "solid" as the first line and end with "endsolid" as the last line.

ii) Binary encoding :

Binary files encode the data in binary language and comparatively small in file size. Binary files contain more information than an ASCII file and are efficient for data processing [19]. A binary file starts with 80 character header followed by number of triangles in the file as a four byte unsigned integer. The information on the vertex and normals are provided as 32 bit floating point numbers. For every triangle there is a two-byte attribute count element that encodes additional information of the triangle used for checking [20]. The generic binary format of STL is displayed in the figure 2.7

```

UINT8[80] - Header
UINT32 - Number of triangles

foreach triangle
  REAL32[3] - Normal vector
  REAL32[3] - Vertex 1
  REAL32[3] - Vertex 2
  REAL32[3] - Vertex 3
  UINT16 - Attribute byte count
end

```

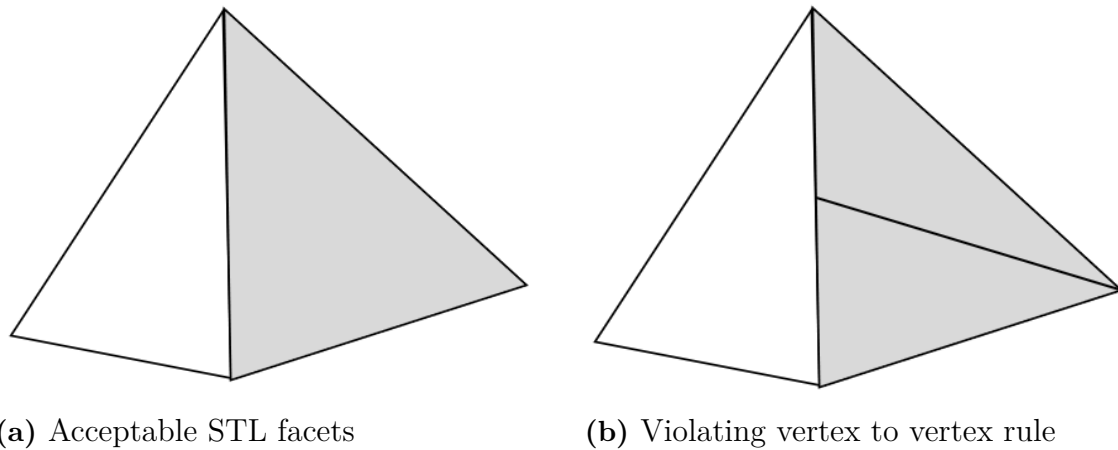
**Figure 2.7:** Binary STL format

While Binary STL file is recommended for CAD and AM, ASCII format is preferred for manual debugging in text format.

## 2.2.1 Rules for STL files

### 2.2.1.1 Vertex to Vertex rule

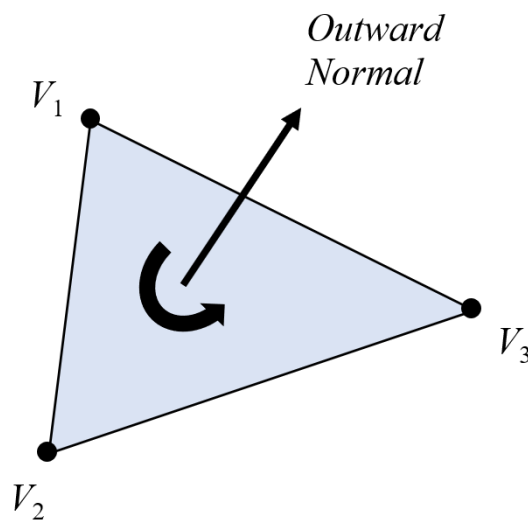
Each triangular facet must share two vertices with each of its adjacent facet, that is no facet is allowed to share two adjacent facets along one edge [21]. A vertex cannot intersect an edge of another facet as shown in the figure 2.8.



**Figure 2.8:** Illustration for vertex to vertex rule.

### 2.2.1.2 Facet Orientation rule

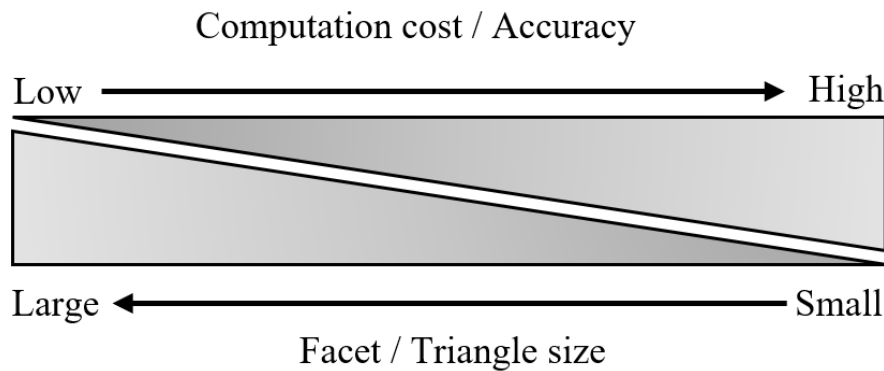
Each facet is defined by normals and coordinates of the vertices. In order to be consistent with all the facets, the direction of the normal is always considered pointing outward and the the vertices are listed in the counter clockwise direction when the facet is looked from outside of the object [22] as shown in the figure 2.9.



**Figure 2.9:** Illustration of definition of vertices and normal of a facet

## 2.3 Issues in STL files

STL files cover the periphery of a 3D CAD object using planar tessellated triangles. A planar surface can be more accurately tessellated with the triangular facets. However, to capture big curvatures of smooth and curved surfaces more accurately, the mesh is further refined. Hence, smooth surfaces are discretized into more triangular facets to achieve better approximate description of the surfaces. Excessively refined mesh can lead to large file size and more computation time while large mesh size can lead to falsified results (see figure 2.10). Hence the tessellation has to be carefully performed [23]. Similarly STL has no information on the unit of measure which leads to issues in scaling.

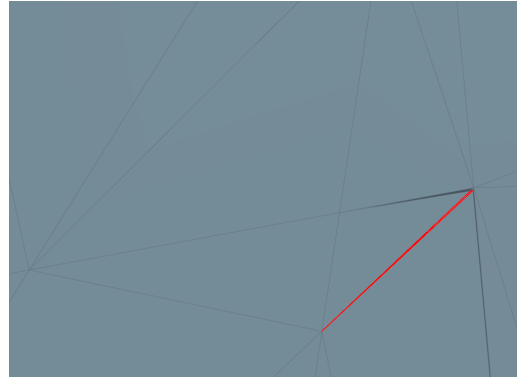
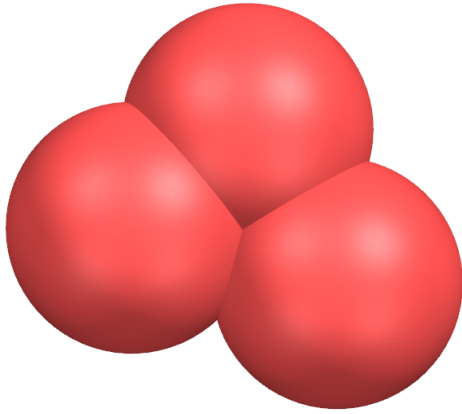


**Figure 2.10:** Relation between Facet size, computational cost and accuracy

The above mentioned problems are inherent to STL files which can be sorted by careful handling. The context of the issues in STL is defined to the irregular geometrical issues that are developed during tessellation process in STL generation. STL files are simple and compact as they contain only triangular surfaces that are developed based on vertex coordinates and normals. However, there is no continuity between the facets and because of deficiencies like redundancy in format, lack of topological information, poor approximation method along with data exchange issues, STL files are prone to errors [24]. It is reported that approximately one in every seven STL files contains various errors [25]. The errors lead to non watertight STL geometries, which lead to manufacturing and CAD handling problems.

The common STL issues are described below.

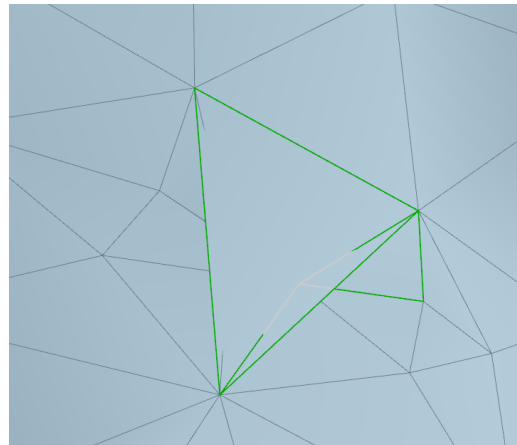
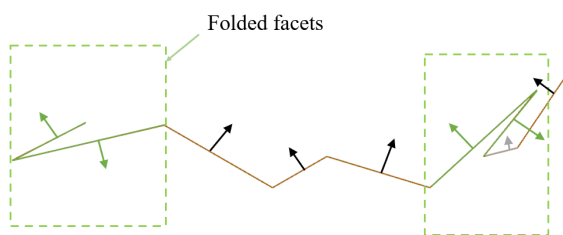
- **Degenerated facets:** Facets that result in nearly zero area are termed as degenerated facets. Usually degenerated facets are introduced during tessellation process as a result of the stitching algorithm that attempts to prevent shell puncture. The edges of the facet lie closely collinear although all of the vertices are distinct [26]. Degenerated facets were found in use cases provided by Siemens Energy and some examples of degenerated facets are shown in figure 2.11



(a) Degenerated facet with insignificant facet face (b) The red lines represent the edges of a degenerated facet

**Figure 2.11:** Degenerated facets

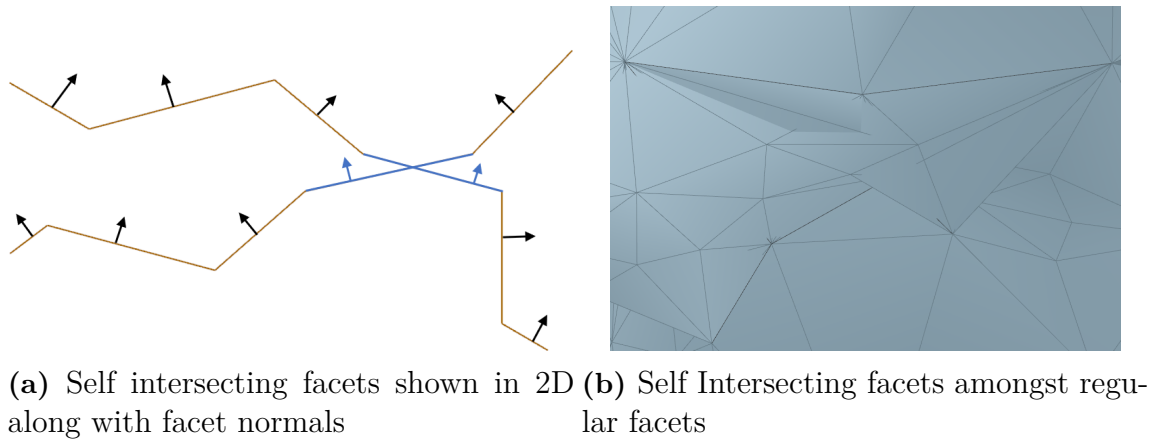
- **Folded facets:** Tessellation of surfaces with large curvatures can lead to facets that form sharp edges with the adjacent facets owing to small angle between them. These facets appear to be folded because of the small angles. This is illustrated in the figure 2.12



(a) Folded facets shown in 2D along with facet normals (b) The green edges denote the particular facet is a folded facet

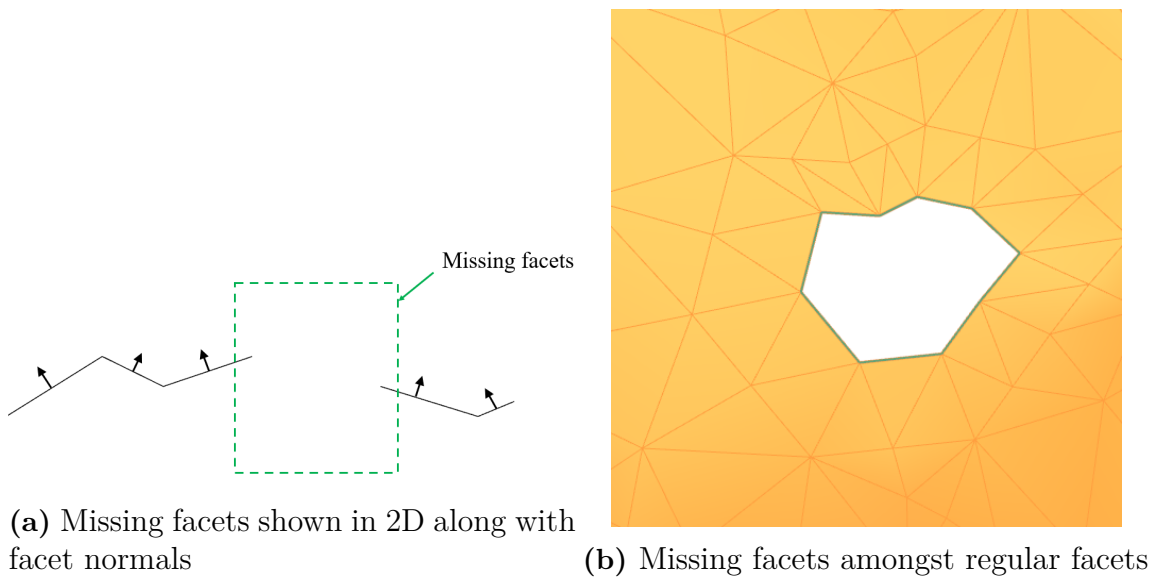
**Figure 2.12:** Folded facets

- **Self Intersecting facets:** Self intersections or overlaps are formed due to numerical round-off errors during tessellation process [26]. Since the vertex coordinates are denoted as floating point numbers in 3D space, self intersection occur if the tolerances provided are more liberal. Figure 2.13 illustrates some examples of self intersecting facets.



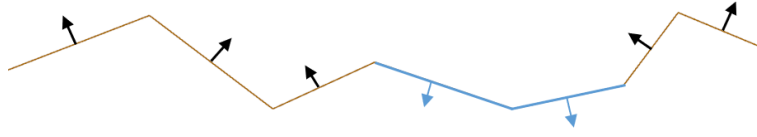
**Figure 2.13:** Self Intersecting Facets

- Missing facets: Due to numerical approximation, the tessellation algorithm may fall short of merging the facet boundaries leading to bad STL with cracks and gaps as shown in figure 2.14. This lead to an open 3D geometry that is invalid for manufacturing.



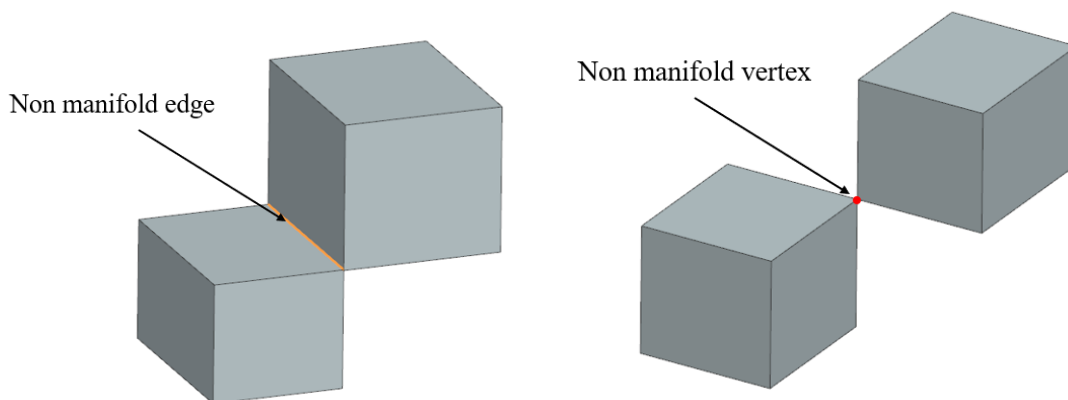
**Figure 2.14:** Missing facets

- Inconsistent normals: In some cases, the normals of the facets are flipped resulting in facets with opposite outward normal adjacent to each other. The figure 2.15 illustrates the phenomena



**Figure 2.15:** Inconsistent normals shown in 2D along with facet normals

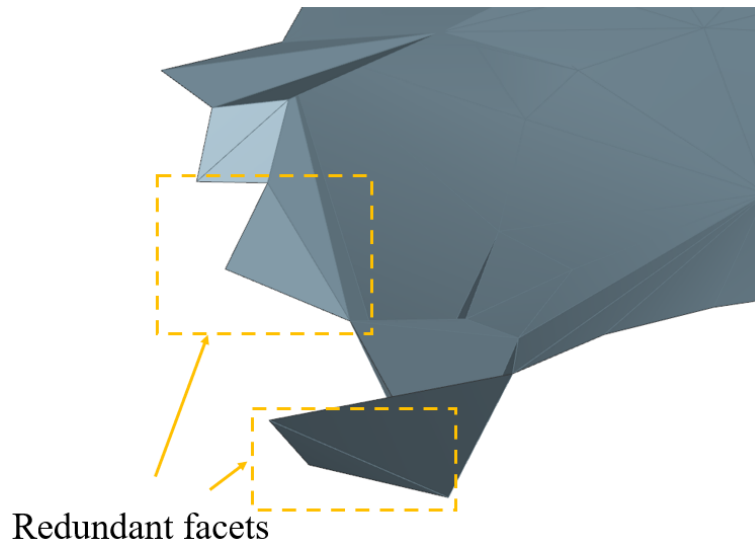
- Non-manifold geometries: Non-manifold geometries are the virtual 3D geometries that cannot exist in the real world. The non-manifold geometries are developed as a result of approximation of fine features during tessellation process. The term non-manifold geometries denote the following issues
  1. Open geometry: In CAD, a geometry with no thickness or one or more facets missing is said to be open geometry because it has no volume and cannot be manufactured.
  2. Non-manifold edges: Usually each facet edge should have only one adjacent edge. If more than two facets share an edge, then the edge is called a non-manifold edge. The non manifold edge separates two lumps of a solid block as shown in figure 2.16 and since the resulting edge is indefinitely thin, the part cannot be manufactured.
  3. Non-manifold vertex: Two separate regions of a body connected by a single point or vertex is termed as Non-manifold vertex. Similar to non manifold edges, non manifold vertex also lead to infeasible and unrealistic geometry.



**Figure 2.16:** Non manifold geometries

- Redundant facets: Facets which are isolated and do not have at least two adjacent facets tend to stick out of the topology of the 3D model. These are considered as redundant facets. The figure 2.17 illustrates redundant facets.





**Figure 2.17:** Redundant facets with two non adjacent edges

## 2.4 Available STL repair techniques

Previous research studies aimed to address the STL errors by different techniques. Snyder et al use chamfer mapping technique to close the noise induced gaps in the edges [27]. Barquet and Sharir [28] used partial curve matching technique and optimal triangulation technique to identify and fill holes found in the boundary of polyhedrons. Bohn and Wozny [29] worked on identifying the erroneous triangular facets and filled the holes with triangles. According to Y.Ito and K.Nakahashi [14], the intersections can be removed by two following approaches 1) re-triangulation near the intersection - suitable for simple geometries and 2) converting intersecting surfaces into one continuous surface before meshing - effective for complex geometries. S.Liu proposed an algorithm to follow to automate the repair of STL files. First to repair the hole followed by repairing the overlaps then repair inconsistent normals and repeat them till the STL is fully repaired [15]. A combination of the above techniques is used in repairing the STL files in this study.

## 2.5 Available STL repair tools

The STL issues detailed in the section 2.3 were attempted to repair using many commercial tools. The thesis work is focused to find repair solution within NX CAD environment however, a basic study was performed on the available tools that are dedicated to repair STL files. Some tools are free for all the users while some tools require paid license for usage. While some tools are online platform based, some are offline based. Each tool available for STL repair consists of specific benefits and shortcomings. Some commercially available STL repair tools are listed below.

### **Autodesk Meshmixer:**

A versatile and user-friendly tool that supports several repairing functions and re-

garded to be highly ranked tool for repairing mesh [30]. Meshmixer is available for free for non commercial purpose however the tool will be no longer developed by Autodesk since the functionalities are integrated to Autodesk Fusion 360 [31]. Autodesk Fusion 360 can be used for commercial purpose on paid subscription and can handle CAD geometries, CAM, CAE and PCB tools. The annual subscription fee of Autodesk 360 is nearly 5000 SEK [31].

### **MeshLab:**

A free, open-source 3D mesh processing software that is ideal for advanced STL repairs consist of rich toolset that can clean, heal, inspect, edit, render. The tool is in constant development and can handle large STL files [30]. The tool requires good level of technical knowledge to use it appropriately [32].

### **Blender:**

Defacto standard tool for 3D modeling and animation [30] that possess powerful functions to repair STL files. The tool is open source however, sophisticated to use.

### **Autodesk Netfabb:**

An advanced well-known software for repairing STL files [30]. The tool is available in Premium and Ultimate versions that include advanced features like post processing and lattice optimization. The premium Netfabb version costs around 50000 SEK for annual subscription [33]. Apart from being a standalone tool, Netfabb is also integrated Fusion 360 which also includes functionalities of Meshmixer [30].

### **Materialise Magics:**

A professional mesh preparation tool that allows extensive manual control over meshes [34]. The tool is available in subscription basis.

Other tools such as 3D Builder, Microsoft 3D tools, Simplify3D, Ultimaker Cura and PrusaSlicer are also used to repair STL issues.

## 2.6 Other File Formats Available

The thesis work with STL file format. There are other file formats that possess some more capabilities than STL file as follows.

### **PLY:**

Polygon File Format (PLY) [35] is similar to STL format developed in Stanford graphics lab. PLY file consist of vertex list, face list and list of other elements like texture coordinates, color of the object, transparency. Similar to STL file, PLY can also be stored in ASCII and binary formats.

### **AMF:**

Additive Manufacturing File format (AMF) [36] is an open file format that contains the description of model in XML format. AMF can store the information of geometry along with information on material, texture.

**3MF:**

3D Manufacturing Format (3MF) [37] is a human readable 3D printing format developed by 3MF Consortium, founded by Microsoft. 3MF contains 3D model, material, property information in zip compression along with the capability to carry digital signatures, 3d textures and print ticket. The file format is free and open to use in CAD tools. The format consist of rules that limit the user to create a 3MF format file only if the geometries are manifold without gaps, open edges.

**OBJ:**

OBJ [38] is a open source and neutral file format developed by Wavefront Technologies used to store 3D data and geometric objects. In contrast to STL, OBJ allows the user to store information of the model in three different ways. 3D model can be

- i) tessellated with polygonal faces, similar to STL or
- ii) defined with free form surface curves, curved lines or
- iii) defined as free form surfaces patches, NURBS

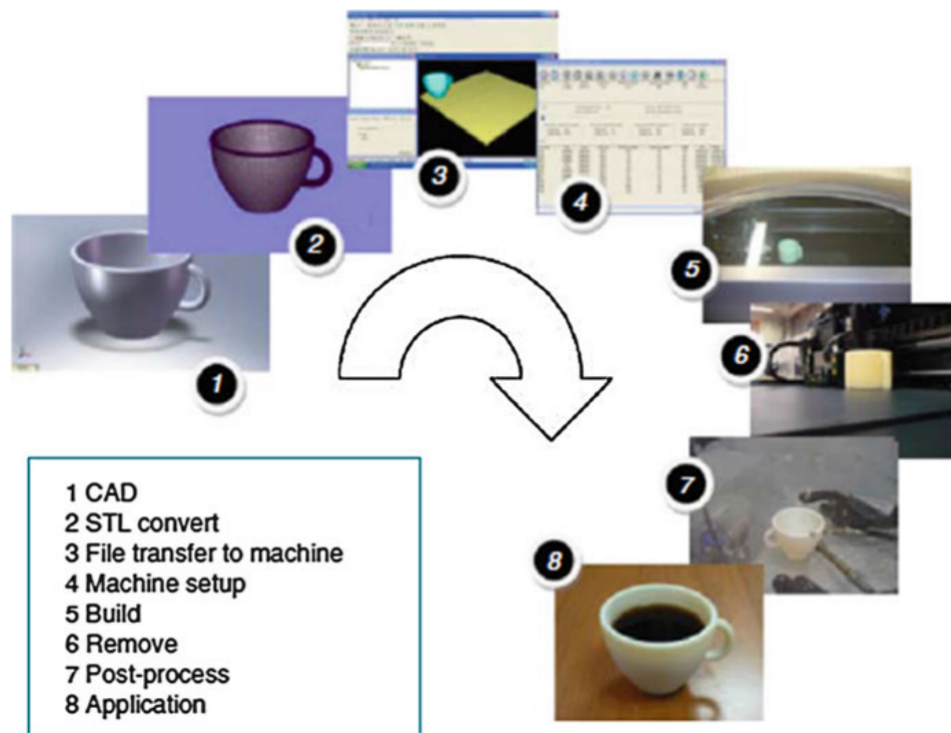
OBJ format's ability to store information as free form surfaces and surface patches allows the user to save more precise information at lower file size. OBJ format is used in aerospace industries because of the capability to store precise information using surface patches. OBJ also allows the user to store color and texture information of a model in Material Template Library (MTL) file format.

## 2.7 Additive Manufacturing

According to American Society for Testing and Materials (ASTM), Additive manufacturing is a process of joining materials, usually layer by layer to make objects from 3D CAD geometry. In consumer market, AM is also referred to as 3D printing. The major benefits of AM include reduction in product development lead time, cost reduction and ability to manufacture complex models [39]. In 1980s, the AM technique was widely used for prototyping and it was also referred to as Rapid Prototyping (RP).

According to Gibson [20], the process involved in AM can be explained in eight different stages as shown in the figure 2.18. A typical AM process starts with the CAD model to be manufactured. A solid 3D CAD geometry can be used as the input and the 3D model should comply Design for Additive Manufacturing (DfAM) principles for successful printing. The 3D model which is developed from various CAD software is converted into CAD neutral format like STL that is widely accepted by AM machines. Later the converted file is manipulated in the AM machine's build area to check the scale, position and orientation for building. One additional step here is to preprocess the input model. This includes processes like establishing support structures, determining infill percentage and similar steps. Then the AM

machine is setup prior build process which include steps like leveling the build plate, loading material, checking energy source, etc. The build process is mostly automated because the machines are capable to function without human intervention. However, some supervision may be required to check the material run out, software glitches, power fluctuations etc. After the completion of the build, the component is removed from the machine and the build plate. The build removal can be a simple process or highly technical process based on the technology involved. The extracted component may need additional cleanup like support structure removal, heat treatment, surface finish and other similar processes which are usually considered as post processing step. After the post processing technique, the part is ready to use.



**Figure 2.18:** AM process in eight stages [20]

# 3

## Research Methodology

This thesis is an exploratory study with the focus to find solution within NX CAD environment. For better understanding of thesis problem and to develop a method for repairing STL file, different research techniques were handled. Primary research techniques such as interviews, brainstorming and online forum discussions were utilised and secondary information sources including research papers, databases, company websites were explored. More information on the research methods are detailed in this chapter.

### 3.1 Literature review

Databases such as Google Scholar, Chalmers library were searched with keywords like Stereolithography files, STL issues, topology optimization, additive manufacturing to acquire more knowledge on the concepts involved in the thesis work. The literature study was also used to explore the problems and solutions in detail. The literature review aided in gaining knowledge about TO, STL files and AM. The study was focused on the rules of STL files and types of geometric issues that are commonly found in STL files. The approaches that already used to repair STL were helpful in addressing the STL issues in this thesis work. Alternative file formats and tools adapted for additive manufacturing and STL repair opens the scope for research in other areas.

### 3.2 Interviews

Throughout the project, three interviews were planned to acquire specific knowledge from domain experts. A set of questions were used to drive the initial questionnaire followed by questions from the information provided by the interviewee. The first interview was conducted with a CAD expert at Siemens energy which also included brainstorming of ideas. The focus of the interview was to enquire the unexpected issues encountered while working with NX commands. By tracking the root cause and exploring different ideas, the outcome of the interview was a robust idea to convert STL files to solid body by creating boundary spline across each cross sectional plane. Despite the idea was robust and promising, the STL issues such as intersection, gaps and the shortcomings of NX to develop surface from the curves limited the approach. The idea was not successfully developed at present, but future development can make this possible.

The second interview was aimed to explore the automation capabilities of NX from a program developer at Siemens Energy. The unstructured interview was helpful in gaining the knowledge and insights on NX Open. The main outcomes of the interview were the suggestions on NX Open, databases available for NX Open and programming language to be followed which were immensely useful in the development of the STL repair method.

The third interviewee was an automation expert with previous experience in programming for NX customization. The interview was focused on developing automation script for fixing STL issues. The unstructured interview aided in gaining insights on tool used for NX Open. The outcomes of the interview included suggestions for User Function Programs (see chapter 4), troubleshooting of programming scripts (also explained in Chapter 4) that were significant in the exploration of automation methods.

## 3.3 Forum Discussion

The forum discussions were used to address the knowledge gap in programming towards fixing STL issues. The forums dedicated to NX and NX automation based discussions were identified and served as knowledge source for exploring the possibility of automation in NX CAD. Forums such as NX Journaling, Siemens Community and Stack Overflow were used to learn programming through examples. The queries and solutions that were provided by other users of forum were the main medium of learning NX automation. The discussions initiated by other users provided idea on syntax for using NX commands. Some project specific discussions were initiated by the thesis student to pose questions related to STL repair and the suggested solutions from other forum users were tried to execute in the thesis work. The forum questions and the responses are included in the appendix B and C

# 4

## Tools

A closed geometry with mass and volume can be handled along with other CAD models in PLM system. Owing to the imperfections in the geometry of the STL files, the TO outputs were could not be handled similar to other CAD files. In order to work with other CAD models, the STL files need to be repaired such that the TO output possess mass and volume without any facet errors. To repair the STL files, an intermediate tool is required. A CAD tool is used for this purpose. Therefore, the process flow between TO and AM is not same as described in Chapter 2. The realistic process flow between TO and AM includes CAD system as shown in the figure 4.1.



**Figure 4.1:** Realistic process flow between TO and AM due to STL issues

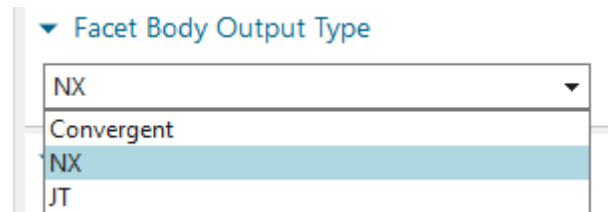
For this thesis work, NX CAD tool is used to investigate and repair the STL issues. This chapter will detail the capabilities of NX with respect to handling, editing and repairing STL and the related issues and brief some of the important commands used in fixing the STL files. The chapter will also outline the automation capabilities in NX along with specific tools used in exploring the automation technique.

### 4.1 Types of facet bodies

As introduced in chapter 2, STL files can be stored in two different formats. NX CAD can execute ASCII format and the resultant is a point cloud of vertex points. A binary STL file can be imported into NX CAD and the resultant is a 3D geometry with facets. A binary STL file can be imported into NX CAD in 3 different types namely, convergent facet body, NX facet body and JT facet body.

JT (Jupiter Tessellation) facet body is a CAD neutral data format developed by Siemens Digital Industries. JT facet bodies are lightweight bodies that are commonly used for data visualization and data exchange between multiple CAD programs [40]. NX facets offer higher level of details than JT facet bodies and used for reverse engineering, analysis and data visualization. Convergent facet bodies are recently developed that can work with any imported feature, surface and facet geometry directly without the need of conversion. Convergent facet bodies are used

for reverse engineering, analysis, optimization and 3D printing.



**Figure 4.2:** Types of facet bodies available in NX CAD

Convergent facet bodies are of two types namely convergent sheet body and convergent solid body. A convergent facet body is represented as convergent sheet body if the facet geometry consist of gaps such that the body is not watertight whereas the convergent facet body is represented as convergent solid body when the facet body is watertight. The watertight facet body possess volume and mass if assigned a material and the watertight facet model can be handled along with other CAD files. In other words, the convergent solid body can be integrated to an assembly and can be Boolean operated with other CAD models which are required by Siemens Energy engineers. Though the convergent solid bodies can be handled similar to traditionally designed CAD files, the convergent solid body can contain STL issues such as intersections and sharp facet edges that are not desired in the final design. Hence obtaining smooth convergent solid body without errors is desired to handle the STL file similar to other CAD files in Product Lifecycle Management (PLM) system.

The convergent sheet body can be distinguished from convergent solid body using the body color. Convergent sheet body is bluish grey in color while convergent solid body is grey in color similar to a solid NURBS CAD represented in NX CAD. One more method to distinguish convergent sheet body from convergent solid body is by taking the cross section in NX CAD. Since the convergent solid body is watertight, the cross section view command with show cap option enabled will have green cap that fills the gap between inner periphery and outer periphery. However, convergent sheet body does not possess green cap when the cross section view is taken.

In the part navigator entry, convergent bodies are populated in model history while JT and NX facet geometries are populated as non-timestamp geometries. The discussed points are compiled into a table 4.1. The criteria of comparison were chosen to contrast the characteristic and capabilities of different facet body types with respect to handling and repairing STL issues.



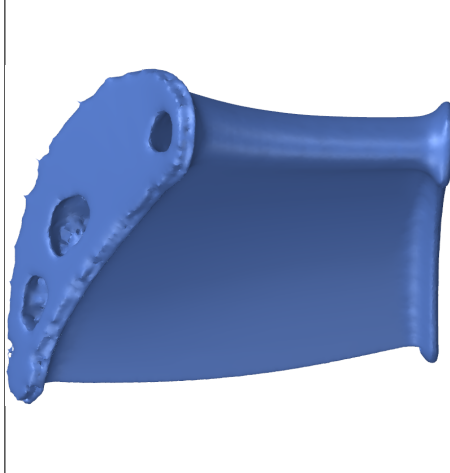
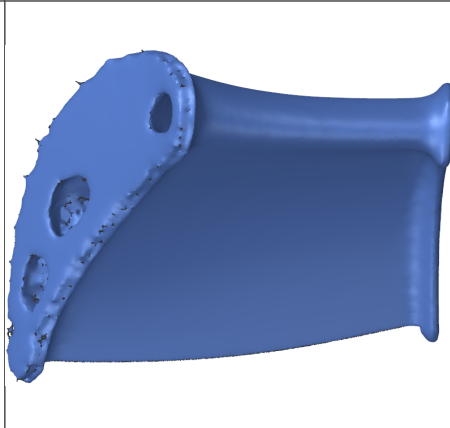
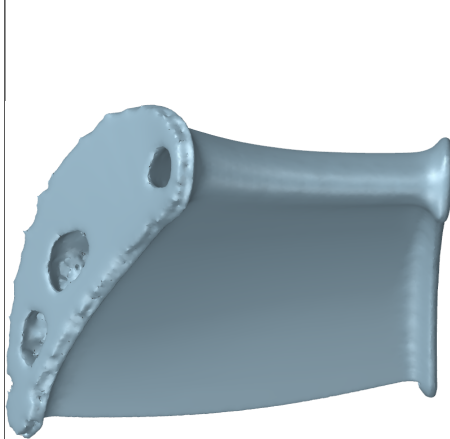
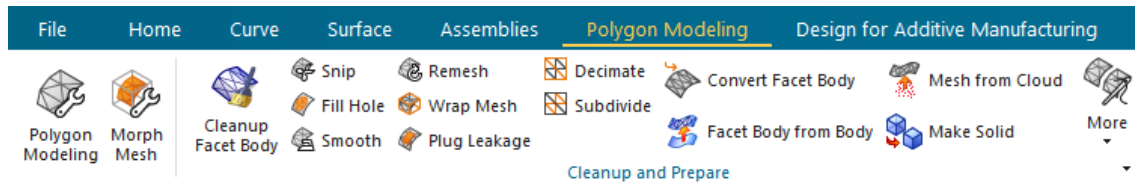
Criterion	JT facet body	NX facet body	Convergent facet body
Characteristic	CAD neutral lightweight format	Conventional facet format with higher details on data	Newly developed by Siemens that can work with surfaces, facets and bodies directly
Commonly used for	Visualization and data exchange between CAD programs	Reverse engineering, analysis and data visualization	Reverse engineering, analysis, optimization and 3D printing
Part Navigator Entry	Lightweight body in Non-timestamp geometry	Facet body in Non-timestamp geometry	Convergent body in Model History
Default representation	Blue facets with no edges shown	Blue facets with erroneous edges shown	Convergent sheet body :Bluish grey Convergent solid body: Grey
Manual repair	Not possible	All the commands available for facets can be used	All the commands available for facets and some commands of regular solid modeling can be used
Automatic repair	Not possible	All the commands available for facets can be used	Not all the commands available for facets work
Visualization			

Table 4.1: Types of facet in NX

## 4.2 Commands to handle STL issues

Most commands which work with 2D geometry, surfaces or solid geometries do not support facet bodies. NX CAD has a separate commands for facet bodies which will allow the user to work with STL files. In order to help the user, NX CAD tool consist of a dedicated toolbar for working with facet geometries. The commands to handle the facet geometries are consolidated in polygon modelling toolbar as shown in the figure 4.3.

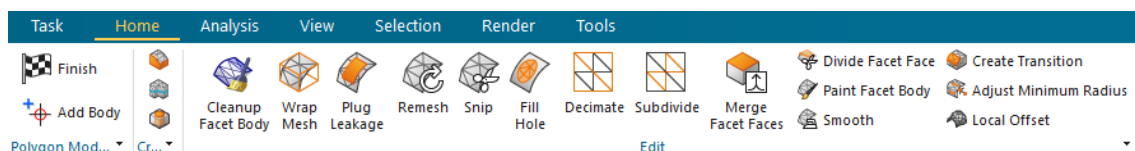


**Figure 4.3:** Polygon modeling toolbar displaying the polygon modeling task environment command and other available commands [42]

The commands used in polygon modeling are history free, meaning, the commands used do not populate as an entry in the part navigator. The part navigator consists of distinct facet bodies in the STL file and do not contain history of commands used on the STL file. This limits the ability to edit the previously executed commands.

### 4.2.1 Polygon modeling task environment

The polygon modeling toolbar consists of set of edit tools that can be used to repair facet bodies. The polygon modeling toolbar consist of polygon modeling command that leads the user to a dedicated task environment as shown in the figure 4.4. The polygon modeling task environment consolidates all the solid modeling commands such as extrude, create hole feature and surface modeling commands like divide facet face, offset and more options that are compatible with facet geometries and are not available in polygon modeling toolbar. Morph mesh task environment in the polygon modeling toolbar consist of commands that can be used to deform the existing facet body. A cage around the existing facet body can be constructed and the cage can be interactively reshaped. However, morph mesh is not used in the STL repair work.

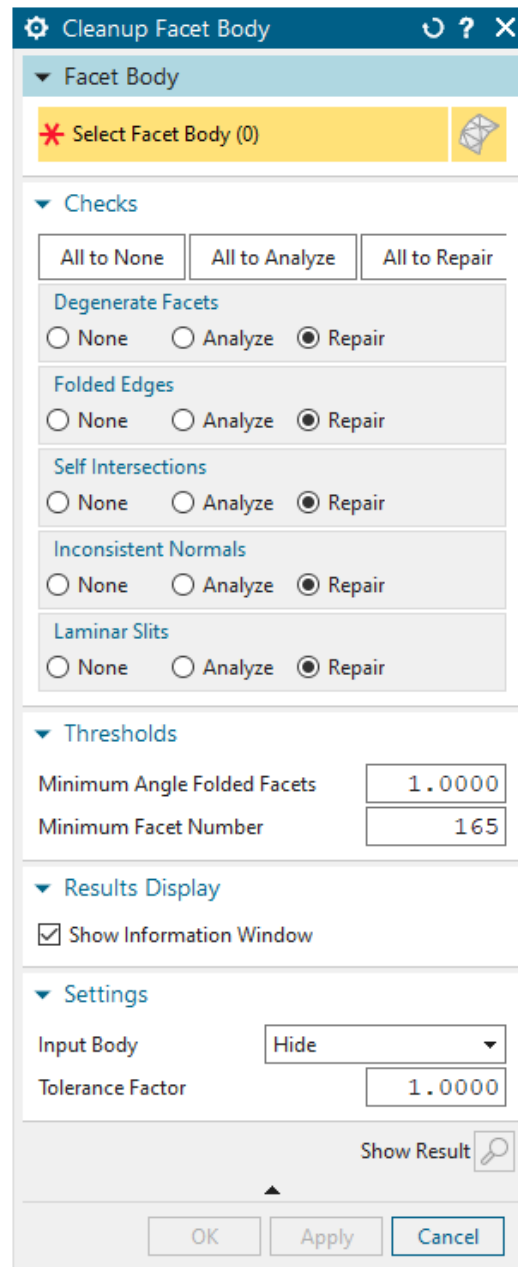


**Figure 4.4:** Polygon modeling task environment and the available commands [42]

### 4.2.2 Cleanup facet body

Cleanup facet body, available in the polygon modeling toolbar, aid in visualising and automatic repairing of the issues in facet geometry. The Cleanup facet body uses

background algorithm to identify the position of various imperfections and allows the user to choose if the identified issues need to be repaired. The command can diagnose the degenerated facets, folded facets, self intersection facets, inconsistent normals and laminar slits. Laminar slits are the imperfections that do not possess laminar continuity between the edges. The Cleanup facet body command window is shown in the figure 4.5



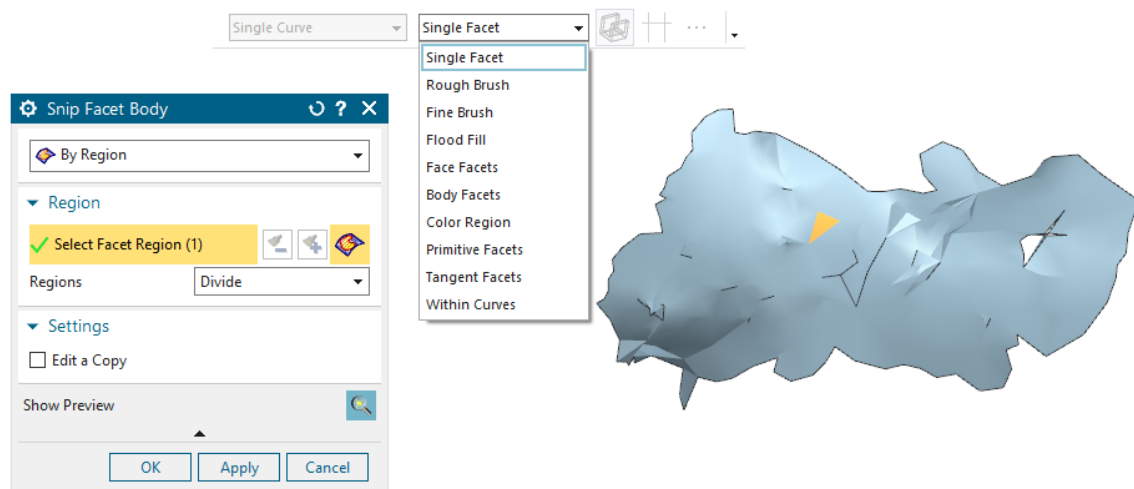
**Figure 4.5:** Cleanup Facet Body command window displaying different facet issues allowing user to decide if the issue has to be analyzed or repaired or not to be checked [42]

The command also allows the user to set two threshold values as shown in the figure 4.5. The minimum angle folded facets allows the user to set the minimum

angle between adjacent facets below which the facets are considered to be folded facets. The other user defined value is the minimum facet number. If number of facets fall below the user set threshold value, then the facet body is discarded. The cleanup facet body command also allows the user to set if the input body has to be hidden, unhidden or deleted. The tolerance factor is the global tolerance value that is multiplied with tolerances in detecting individual imperfections. In other words, every defect such as degenerate facets or folded edges possess a tolerance value for detection and it varies depending on the facet body. The tolerance factor that is set by the user is multiplied to automatically computed tolerance values. The cleanup facet body command also possess the ability to preview the result and also can populate the result in an information window.

### 4.2.3 Snip facet

Snip facet command divide the facet bodies. There are different methods of snipping. The facets can be snipped by selecting the facets directly or by drawing a closed polygon region or by projecting a curve or using a dividing plane. The selection by region method allows quick manual selection of facets using brush tools, also possess options to select tangent facets, flood fill and other options. The selected facets can either be divided into separate facet bodies or can be deleted from the body. Snip facet body command window is shown in the figure 4.6.



**Figure 4.6:** Snip Facet Body command window along with different filter selection to be used for selecting facets [42]

### 4.2.4 Fill Hole

Fill hole command is used to address the missing facets by closing holes in the faceted models. Faceted models created by optical scanning often fail to scan the entire component leading to holes. The fill hole command help in transforming the non-manifold faceted bodies to watertight manifold bodies. The fill hole command can

1) fill one hole or fill all the holes that exist in the selected facet body [42]



2) close a open hole (notch) along a laminar edge of a facet body [42]



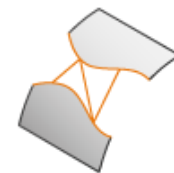
3) bridge the gap between an island of facet and a surrounded hole [42]



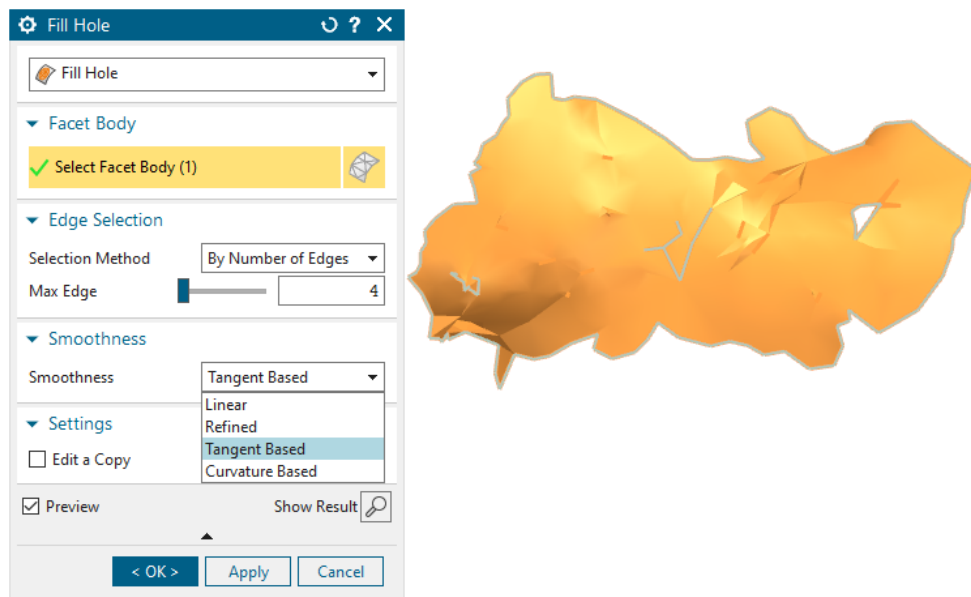
4) bridge the gap between two holes [42]



5) fill the gap between two laminar edges [42]



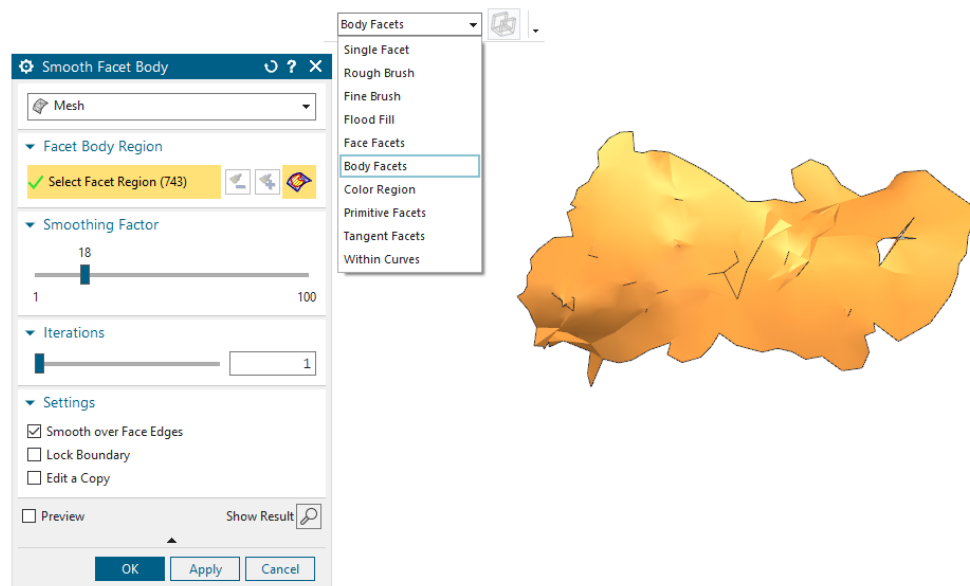
In the fill hole option, the command also allows the user to select the holes based on maximum number of edges. The holes with number of edges greater than the user specified value are not filled during the process. The option to select the smoothness allows the user to choose the transition at the the hole region. The user can select linear or refined or tangent based or curvature based continuity. The hole fill dialog box is shown in the figure 4.7



**Figure 4.7:** Fill hole command window displaying the list smoothness options [42]

### 4.2.5 Smooth Facet Body

Smooth facet body command is used to smoothen the sharp edges formed between facets and results in cleaner topology with better continuity between the facets. The dialog box of smooth facet body is shown in the figure 4.8.



**Figure 4.8:** Smooth Facet Body command window with the list of facet selection in the selection filter [42]

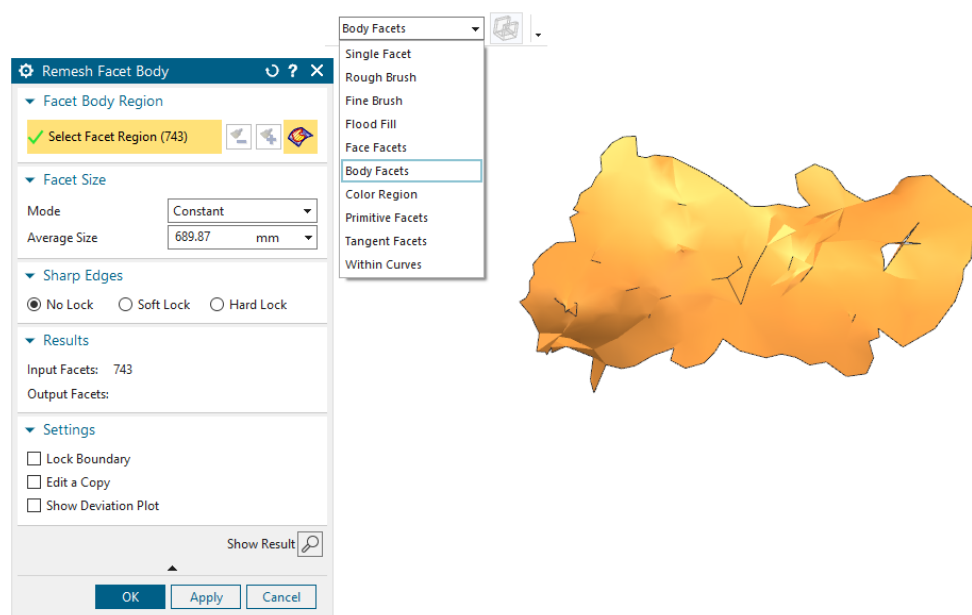
Smoothness can be defined to the free edges of the facet or the mesh. Smoothness to the edge align the edges that do not have adjacent facet for better continuity while the mesh option has the ability to smoothen entire facet body. The mesh smoothen

option also allows the user to lock boundary which ensures the free edges and its vertices to be unaltered. Both the options allow the user to choose the smoothing factor and number of iterations of smoothing algorithm to be executed on the facet body.

### 4.2.6 Other Useful commands

#### 1) Remesh Facet Body

The remesh facet body command modifies the triangular density of the mesh resulting in higher quality of topology. The remesh facet body command allows the user to customize the average size of facet and also include options to lock the sharp edges as shown in the figure 4.9



**Figure 4.9:** Remesh Facet Body command window [42]

#### 2) Convert Facet Body

The command Convert Facet Body allows a convergent facet body to be converted into a NX facet body and a NX facet body to a convergent facet body at any point in time. Owing to more functional capabilities of convergent bodies, the STL files are imported as convergent body while working. But due to automation limitation of convergent bodies, the working faceted body is converted to NX body. The convert facet body command also allows the user to choose automatic cleanup feature while converting

#### 3) Merge Disjoint and Merge Touching Facet Bodies

Merge disjoint facet bodies bridge the gap between two disjoint faceted sheet bodies resulting to a single merged faceted sheet body. The command is useful for facets that are created using optical scanning. Merge touching facet bodies command is useful in sewing two facet sheet bodies that are touching at a common edge. Faceted sheet bodies that are snipped during repair process can be united together

using both the commands. From the observation, working with merge disjoint and merge touching facet bodies can result in undesirable result like development of gaps or sharp edges near the interfaces. Therefore post executing these two commands, quality of the facet body need to be checked. If the quality is deprived, necessary corrective actions need to be taken.

### 4) Combine Facet Bodies

The overlapping faceted sheet bodies can be united to form a single faceted sheet body using combine facet bodies command.

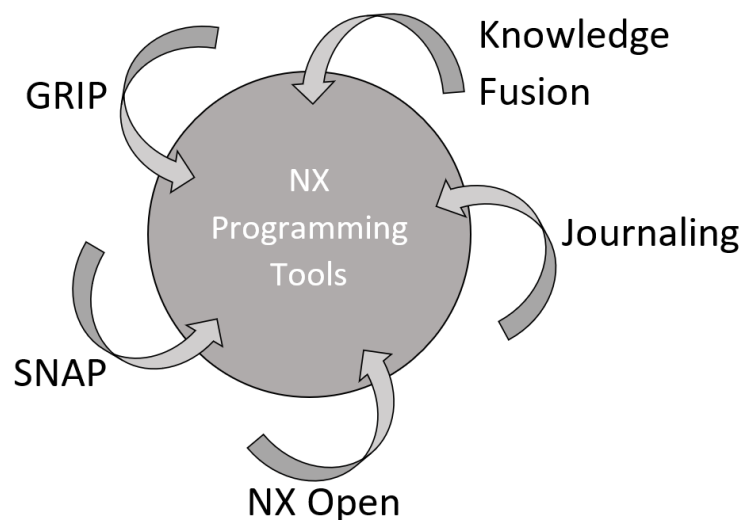
### 5) Facet Edges

Facet Edges command toggle the display of edges or the outline of the triangular facets. Switching on the facet edges facilitate the user to identify if the facets are self intersecting or presence of laminar slits and other STL issues. The command can be found in View tab -> Display group -> Effects.

## 4.3 Automation

### 4.3.1 NX Capabilities

The CAD tool of NX supports various programming applications that help in customization of NX to meet the specific needs of companies. The figure 4.10 illustrates the different automation tools supported by NX CAD.



**Figure 4.10:** Automation tools in NX [41]

Knowledge Fusion is a programming method that uses the engineering knowledge bases in conjunction with formulae and rules to develop powerful solutions [41]. The method uses rules and attributes to drive product design. In addition, external knowledge bases like spreadsheet and databases can also be used to drive the design. Knowledge fusion can be directly used in the NX user environment. All other NX

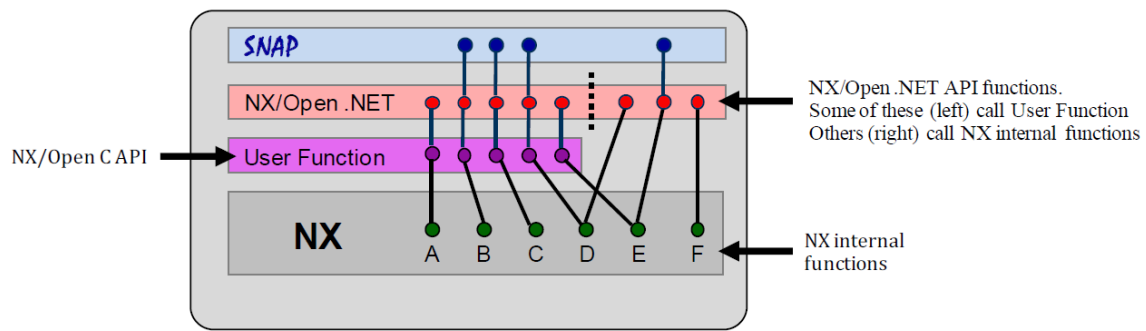


programming tools use a common object model called common Application Program Interface (API) that allows the user to develop automation using different programming languages. NX automation tools support C++, C, Java, Visual Basic and Python for programming.

NX Journaling [41] is a method that enables user to record the modelling sessions such that the recorded modelling steps can be replayed or executed on other files. The interaction made during the recorded session is available as a automation script. Journal supports programming languages like Java, C++, C, Python and Visual Basic. User can set the preferred journaling language in File -> Preferences -> User Interface. This method is very useful in automating repetitive workflows. Also journaling allows the user to edit and enhance the recorded scripts if required. Journaling is a basic automation technique that comes integrated with all Mach series of NX. The journaling method directly embed the data of the modelling session in the source code. So the script can be replayed on a different model if the hard coded values of the source code match with the working model or the journal script should be edited to remove the hard coded values. Though journaling is less powerful in terms of handling complex automation, both SNAP and NX Open functions can be executed using NX Journaling environment.

NX Open is a powerful collection of APIs that possess vast libraries that can automate complex tasks, integrate third party applications and customize the user interface of NX [41]. NX Open can work with common APIs that include NX Open for .NET, NX Open for Java, NX Open for C++, and NX Open for Python and can also work with Classic APIs like Open C, Open C++ and NX Open GRIP. NX Open initially used classic API and the functions available for Open C and Open C++ are collectively called as User Functions, UFUNC or UF programs. Over 5,000 UF programs were especially available for automation purpose and may not be found in NX CAD User Interface. The classic APIs are maintained by Siemens but no longer actively enhanced. UF programs are made available in Common API but few commands are especially available in classic API. Example: `UF_MODL_intersect_objects` is available to use in classic API but not available in Common API. But the Common API include commands that are available in UI of NX CAD which are not available in Classic API. So the library of Common API is bigger than Classic API. This is illustrated in the figure 4.11. One other advantage of using common API is that all Common API language have equal NX capabilities and user need not worry about missing functionalities.

SNAP (Simple NX Application Programming) is an easy to learn programming tool that is designed for people with little or no previous programming experience [41]. SNAP is based on NX Open but do not possess all the functionalities of NX Open. Similar to SNAP, GRIP (Graphic Interactive Programming) is an intermediate scripting language that uses English-like words to create geometry, modify existing geometry, perform file management functions and some advanced operations. GRIP is an old automation technique which is not developed anymore.



**Figure 4.11:** Capabilities of SNAP and NX Open [41]

### 4.3.2 Automation tools used

From the above mentioned programming tools, NX Open is versatile and possess the ability to work with facet bodies. NX Open is a collection of APIs and it is not an application with User Interface. The databases for each programming language containing functions and commands of NX Open are available as Reference Guides in Siemens documentations [41].

From the research, it was evident that Visual Basics and Java were more widely used programming language for NX Open. But these languages are complex and time consuming to learn. Since the thesis student had no experience in the programming, an easy and quick to learn tool was preferred. Owing to easy syntax and beginner friendly applications, Python was initially selected for programming using NX Open for Python API. Spyder was used as the Integrated Development Environment (IDE). Though the language was easy to use, it was difficult to work with NX Open packages in the IDE. Further research on selection of programming language through interviews and internet searches suggested that C# is next easy language that can be used for programming. Also Microsoft Visual Studio, an IDE for C# possess an unique feature IntelliSense that aid in programming which was very much helpful in understanding the codes.

The NX Open for C# API was used for automation. The reference guide available for C# was used to develop the programming script in Visual Studio. The script developed in Visual Studio can be saved as CS file or dlx file. The dlx file can be executed in NX CAD by loading the file using File tab -> Execute -> NX Open command. The CS file can be loaded using play option in the journal commands. These methods of loading the script are not advocated while the script is in development. Automation scripts are prone to errors or bugs if not written in the right syntax. Debugging tool available in Visual Studio is advised to use during the development phase.

The debugging option in Visual Studio allows the user to attach the code in IDE with NX CAD. The step wise execution of the script can be visualized in IDE with the corresponding input, output parameters in each line of script updating for each

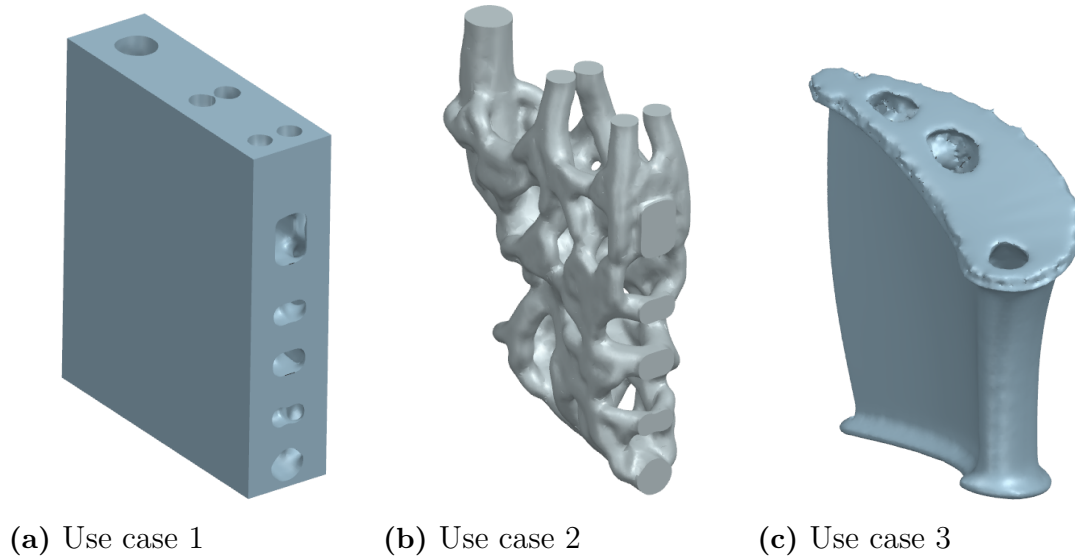
step. Debugging enables the user to find the bug and reason for the error as NX Exception. Siemens documentation possess a list of NX Exceptions with the possible reasons which can be used to work with the bug. In addition, loading the project libraries of NX Open in Visual Studio allows the user to use IntelliSense features that greatly help in programming as well as debugging.



# 5

## Experimental Research

To examine STL issues and develop a method to repair STL files, three use cases were studied. The use cases were the outputs of CFD based topology optimization. The topology optimization process was carried out in MATLAB using the script developed by research students at Linköping University. The use case one and three are the TO outputs of internal fluid flow through channels while the use case two being the fluid flow in use case one extracted as a STL file. The three specific use cases were provided based on the complexity and the number of STL issues in the input files. The STL files studied are shown in the figure 5.13.

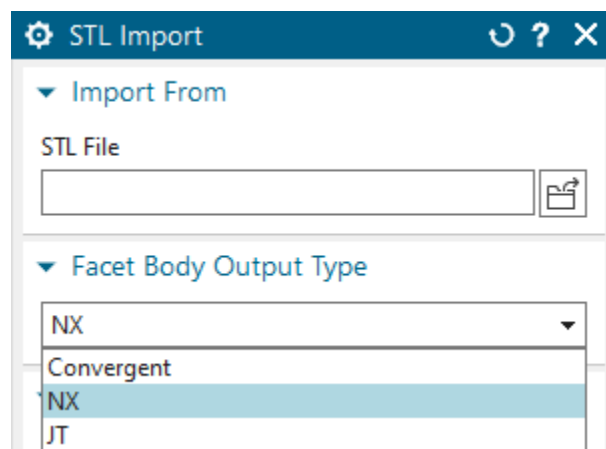


**Figure 5.1:** Convergent bodies of imported use case files

The first two use cases were generated using MATLAB script that included smoothing algorithm while the third use case did not include smoothing algorithm during optimization. The use case one and two were considered to be less complex with fairly better topology as compared to third use case which is considered to be the most complex STL file. The three use cases were executed separately to analyse and repair the STL issues. This chapter consists of sequential steps to be followed in repairing STL files.

## 5.1 Importing STL file

The binary format of the STL files were obtained as TO output from research students at Linköping University. The STL files are imported in NX CAD as convergent facet bodies since convergent facet bodies facilitates more features than NX facet and JT facet bodies. The import commands allows the user to perform automatic cleanup option. However, the automatic cleanup option lead to approximations at all regions which is not desired. Hence, STL file need to be imported without approximations and the automatic cleanup facet body option is not performed while importing. The STL import command window is shown in the figure 5.2.

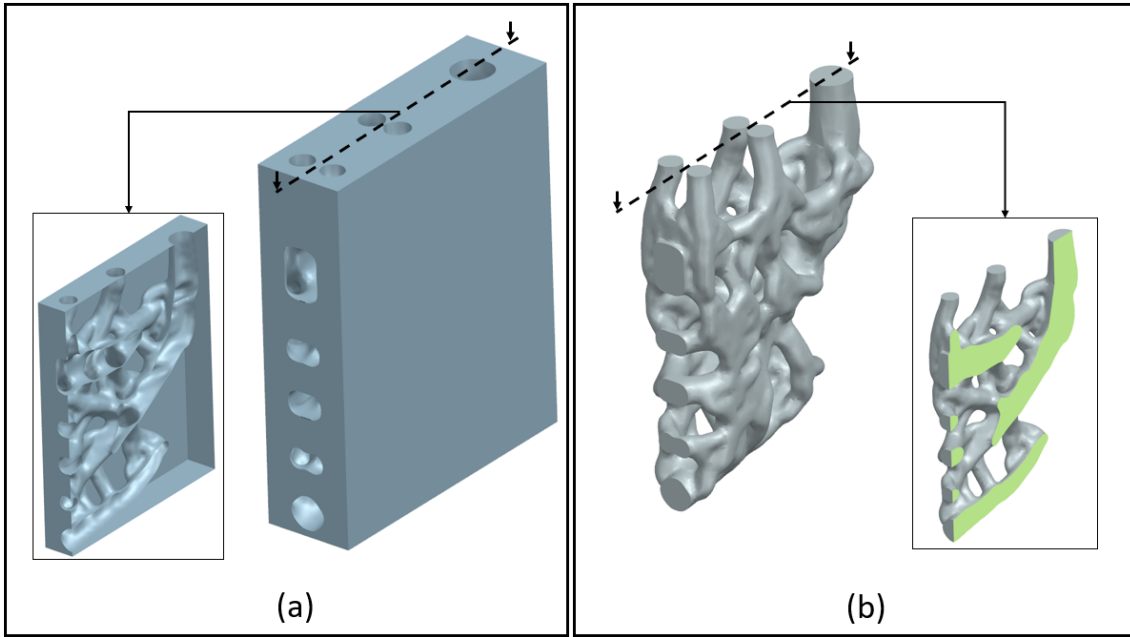


**Figure 5.2:** Import STL command window

## 5.2 Analyze the input

Before emphasising on the repair work, the STL file need to be studied on the quality of optimization, amount repairing effort required and approach perform the repair work. It is also important to understand the function of the working component to identify what geometries need to be preserved. The three use cases vary in quality of topology and approach to repair the STL files.

It was stated that the use case 2 is the inverse Boolean of use case 1. From the figure 5.3, it is evident that the use case 1 possess internally optimized channels that correspond to the external profile of use case 2. The section view with "Show cap option" in the Cap settings in NX CAD is useful to recognize if the STL is fully enclosed by facets. The presence of cap in section view of use case 2 denotes that the STL file is watertight. In contrast, the section view of use case 1 signify that the STL file is not fully enclosed by facets.

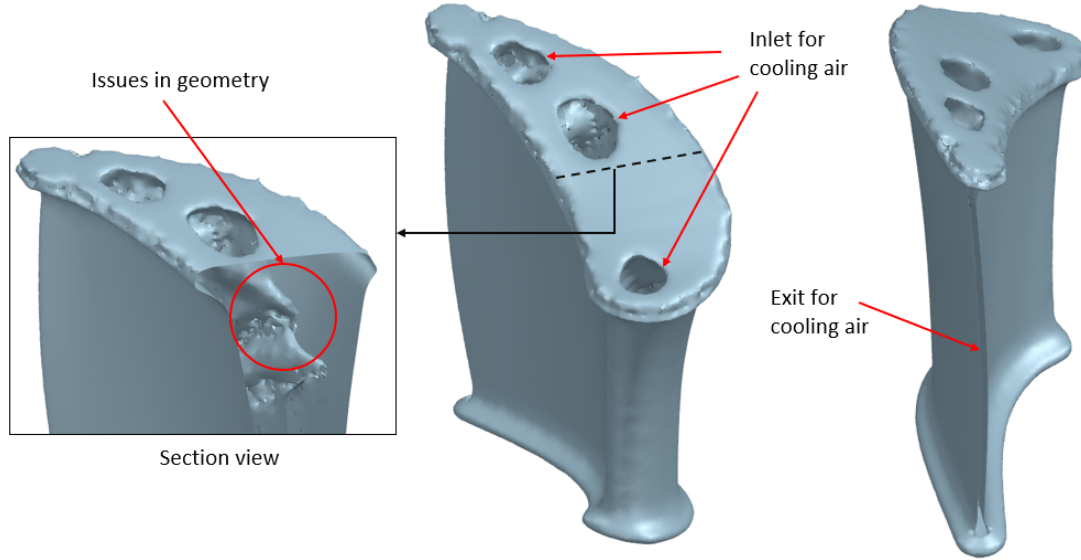


**Figure 5.3:** (a) Use case 1: CFD based TO output of a solid cuboid with specific loading and constraint conditions (b) Use case 2: The corresponding fluid flow in the cuboid extracted as a STL.

Despite being a convergent solid body, the use case 2 contains geometry imperfections. If the imperfections are minimal and under acceptable limit, the use case 2 can be directly saved as part file and can be handled like other part files. Since the STL file contains sharp edges that are not acceptable, the STL imperfections need to be repaired. Since use case 1 and use case 2 are Boolean inverse, there are two approaches in fixing STL. First approach is to repair the STL files individually and the second approach is to fix one STL and perform a Boolean inverse to get the other model. Both the approaches were followed to study the benefits and shortcomings of both techniques. However, in the report, latter approach is explained since the former method is straightforward and follows similar repair technique as use case 3 which is explained in detail in the following sections. Since the use case 1 is a standard cuboid, a solid cuboid of the same dimension can be created and subtracted from the repaired STL of use case 2.

The STL file of the use case 3 is shown in the figure 5.4. The component possess a smooth curved profile because during topology optimization, the curved profile is assigned as non-design space. The holes on top of the component are the inlet for cooling air and the rear opening forms the exit for cooling air. The issues in STL are visible in the section view. This problem can be compared to use case 1 where the exterior profiles are smooth but geometry issues found in interior channels. Since the manual STL repair process involve more manual selection of imperfect regions, one approach is to hide the external facets and work on interior channels or take cross sections to view the imperfect regions. The former method is better because the latter will consume more time and labour. From the observation, the STL files of use case 1 and 2 possess better continuity in geometry as compared to use case

3. This model possess more sharp edges, gaps and non-manifold vertices.



**Figure 5.4:** Use case 3 with the STL issues in the section view

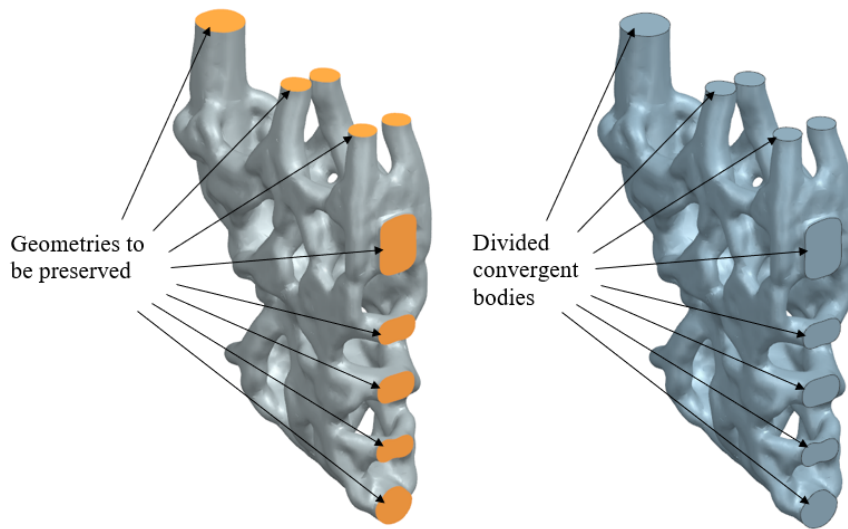
### 5.3 Preserve the non-design space

For every topology optimization problem, the loading conditions and constraints should be applied only on the non design spaces. So every TO based STL file has non design space. The non design spaces are usually smooth with no errors. If the non design spaces are not preserved, then geometric approximations like smoothing, applied during repair process will affect the geometry of non design space and they can undergo dimensional and geometric changes. This is not desirable and so it is important to identify the non design spaces and divide it from the design space.

Use case 1 will be extracted from repaired use case 2. So use case 1 is skipped from the following steps and use case 2 and use case 3 are studied in this section. Note that use case 1 is similar to use case 3. Hence while working on use case 1 individually without considering Boolean approach, the preserving non-design space was similar to use case 3.

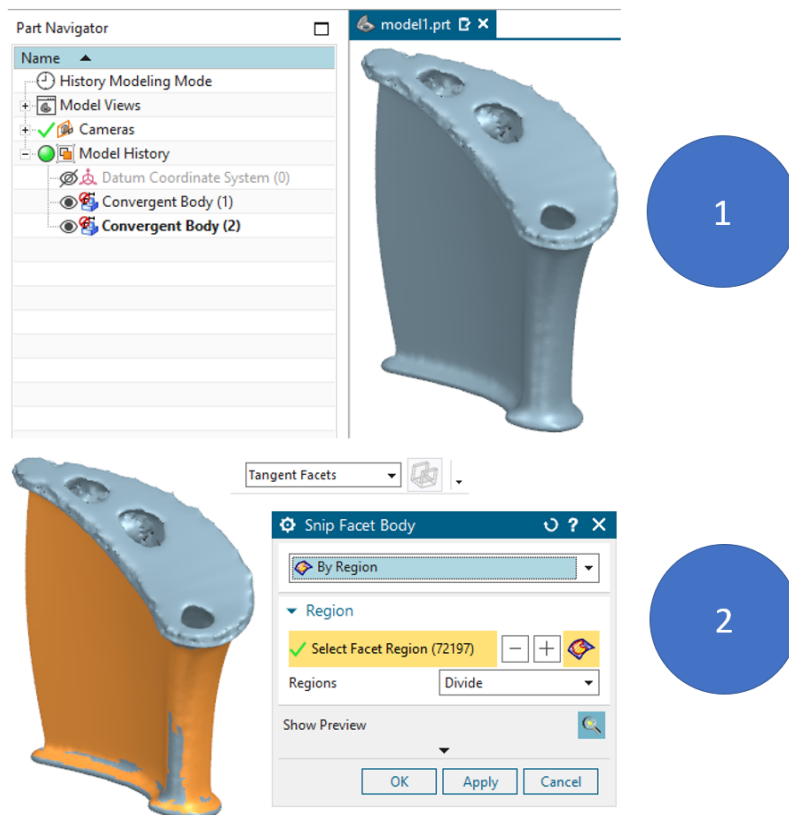
In use case 2, the flat profiles illustrated in the figure 5.5 are the non design spaces. Using snip facet command, the non design spaces can be divided. In the figure 5.5, it can be observed that the use case 2 transformed from convergent solid body to convergent sheet body.



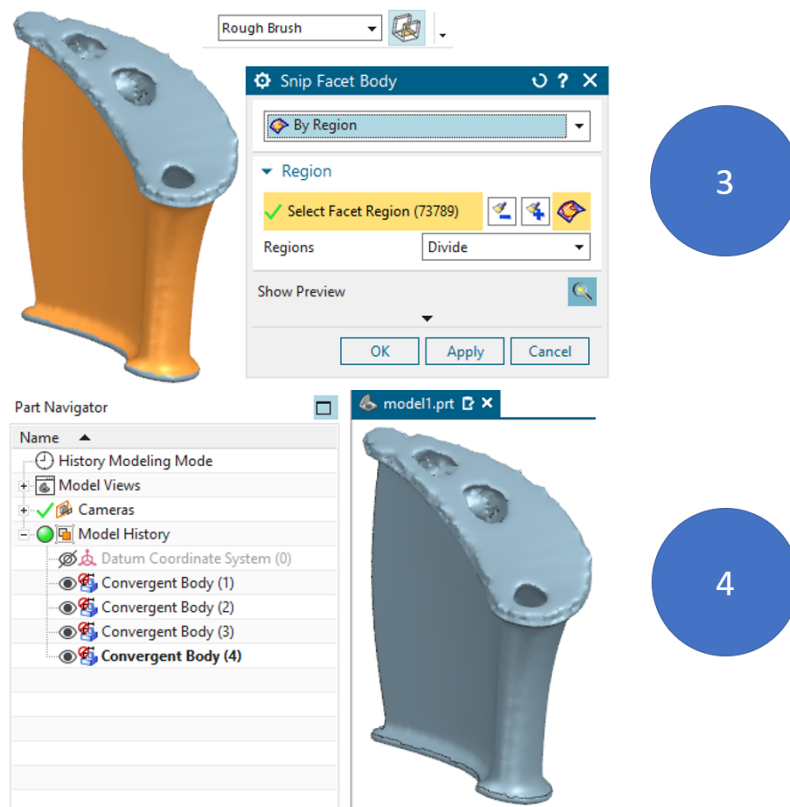


**Figure 5.5:** Dividing the non design spaces in use case 2

In case of use case 3, the curved surfaces forming an aerofoil profile is the non design space and the loading conditions of inlet air is provided that generated the internal flow channel and the inlet openings in the component. The method of preserving non design space in use case 3 is shown in the figure 5.6 and 5.7.



**Figure 5.6:** 1) Use case 3 before snipping the preserved geometry. 2) Tangent facets selected for snipping [42]



**Figure 5.7:** 3) Rough brush selected, remaining non design spaces are selected and selected regions set to 'Divide' and selected OK. 4) use case 3 facets are split and can be seen in part navigator [42]

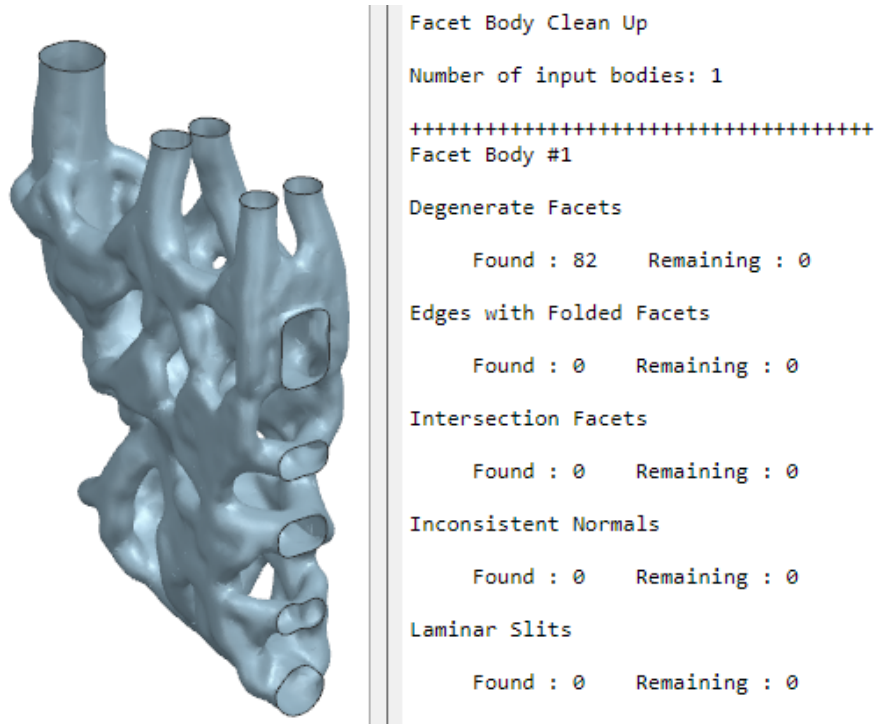
### 5.4 Fix the STL issues

The facet imperfections can be better identified when the facet edges are toggled on. The divided facet bodies can be hid in order to access the facet issues in the interior regions and to get better visibility of STL issues. The automatic command to cleanup facet body can be used to analyze and repair the STL issues. The STL issues identified by NX CAD can be visualized using the analyze command. Show result in the information window allows to see the number of imperfections in the selected faceted body. The convergent sheet body becomes convergent solid body when the number of imperfections for each type of issue is achieved zero. Though this does not hold good all times, the STL files are repaired till the number of STL issues become zero. From the observation, despite repair option in cleanup facet body command tries to fix the STL imperfections, not all issues can be automatically fixed. Manual method is used to repair the imperfections that could not be fixed by automatic clean up method

All the use cases followed a common approach till dividing the non design space. The fixing of STL issues depend on individual STL files. The approach to repair use case 2 is detailed first followed by use case 3.

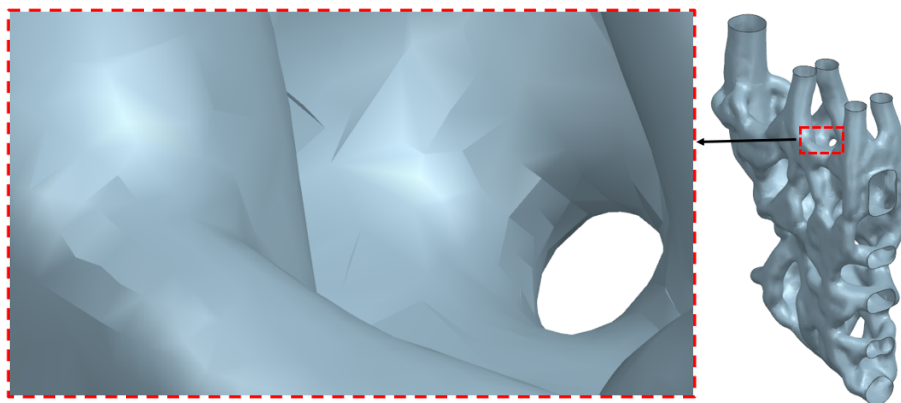
### 5.4.1 Repairing use case 2

The automatic cleanup result of use case 2 with minimum folded angle as 10 and tolerance factor as 1 is given in the figure 5.8



**Figure 5.8:** Cleanup facet body result of use case 2 [42]

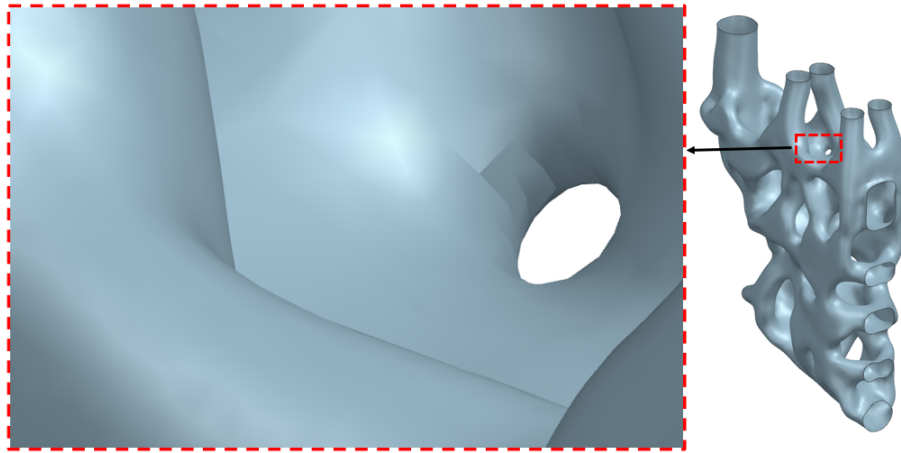
Despite of absence of STL issues, the facet body was not found to be smooth. Closer look on use case 2 is shown in the figure 5.9.



**Figure 5.9:** Rough edges of the use case 2 shown in zoomed view

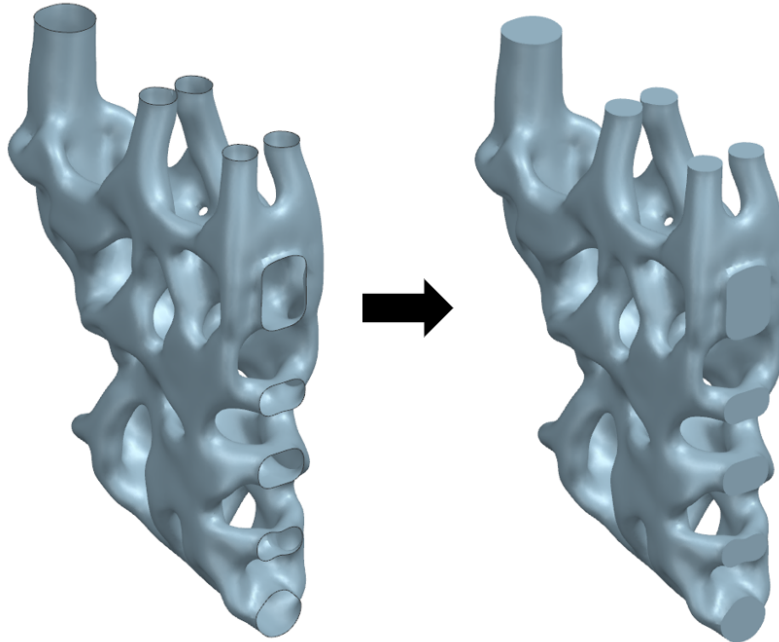
In this case, smooth facet body command can be used with the lock boundary option enabled. Lock boundary option prevents the facets that share edge with non design space from undergoing any changes. User can set the desired smoothing factor and iterations. Use case 2 was smoothened with smoothing factor of 10 and number of

iterations as 1 and the result is shown in 5.10. Localized smoothing is also possible to improve the smoothness at specific regions.



**Figure 5.10:** Use case 2 after smoothing with relatively less sharp edges

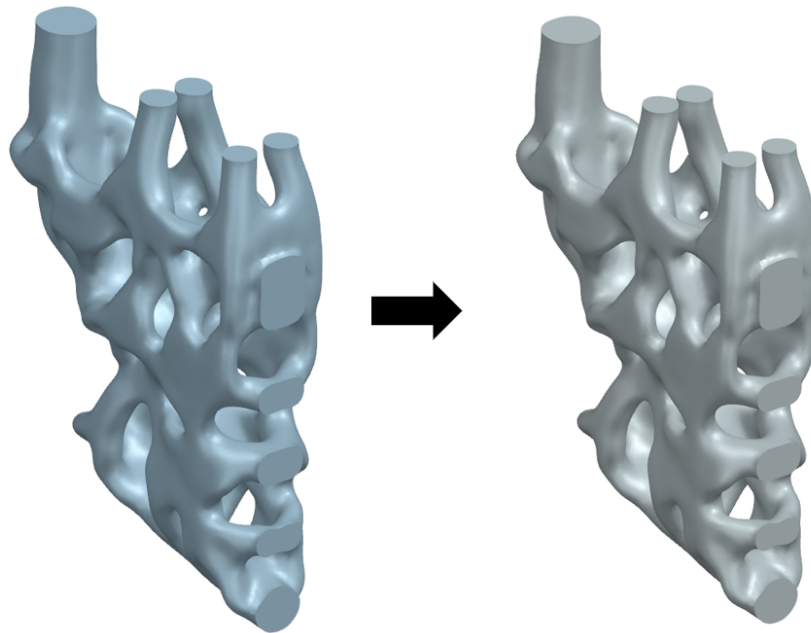
The smooth STL file with no imperfections other than missing facets can be converted into a convergent solid body using fill hole command if the profile non design space is planar or tangential. The use case 2 possess planar non design space, hence the fill hole command with linear continuity is used. The result is shown in the figure 5.11



**Figure 5.11:** The non design space area in use case 2 covered using hole fill. The convergent sheet body becomes watertight

One of the graphic errors observed during the process is the failure of automatic conversion of convergent sheet body into convergent solid body despite being watertight. The result shown in figure 5.11 is a convergent solid body but the body

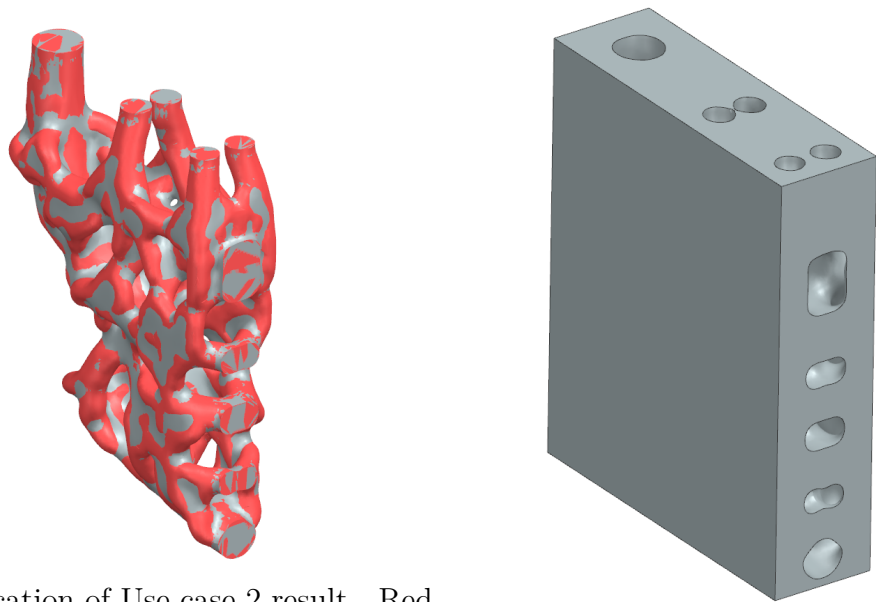
color does not represents the convergent sheet body. The source of error could not be tracked, but a way to fix is to convert the convergent facet body to NX facet body and then convert back to convergent facet body. The corresponding result is shown in the figure 5.12



**Figure 5.12:** Convergent sheet facet body converted to convergent solid body using convert facet body command

As mentioned in chapter 5.2, the convergent solid body can be used along with other parts. The convergent solid body has volume and if assigned a material, it can give the mass. Verifying the obtained result was a challenge. The model can be verified by performing a FEA analysis with same load conditions and check the performance. But a preliminary check is required to verify the quality and accuracy of the model before performing CAE test. Since the input STL do not possess mass or any other parameter to compare, one method of comparing the result is by overlapping the bodies. The visual verification of use case 2 is shown in the figure 5.13a. If the obtained result nearly approximates the net shape, then the model is acceptable. But the degree of acceptable approximation need to be defined.

The method to obtain use case 1 is to perform an inverse Boolean of use case 2 against a cuboid of the dimension in use case 1. The convergent solid body of use case 1 is shown in the figure 5.13b

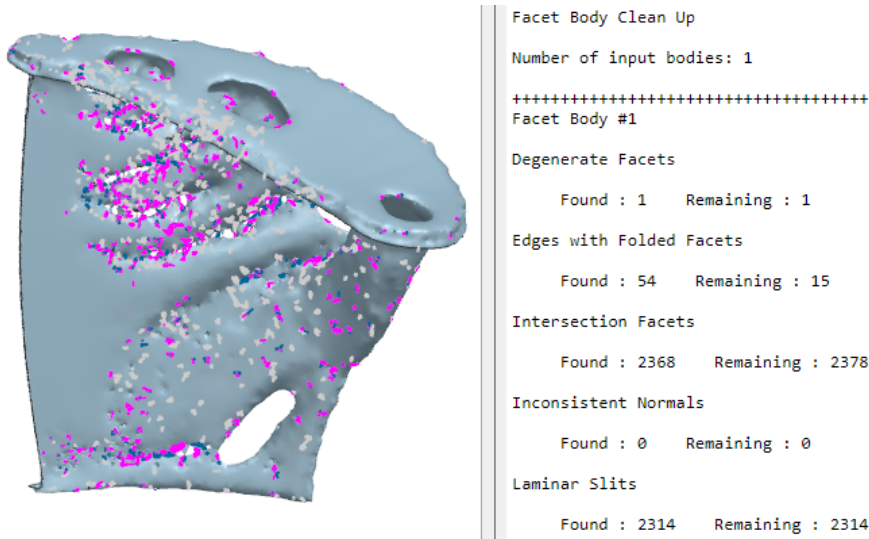


(a) Verification of Use case 2 result - Red body is input STL, Grey body is the re-paired convergent solid body (b) Use case 1 obtained from inverse Boolean of use case 2

**Figure 5.13:** Convergent bodies of use case 1 and 2

### 5.4.2 Repairing use case 3

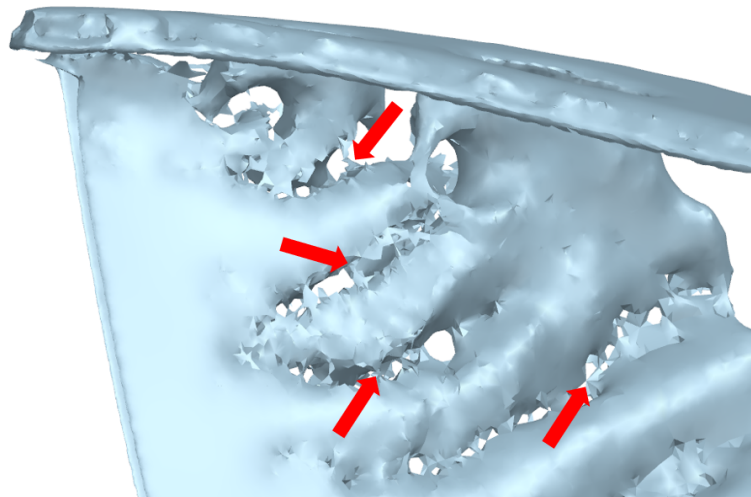
The automatic cleanup result of use case 3 is provided in figure 5.14



**Figure 5.14:** Cleanup facet body result of use case 3 [42]

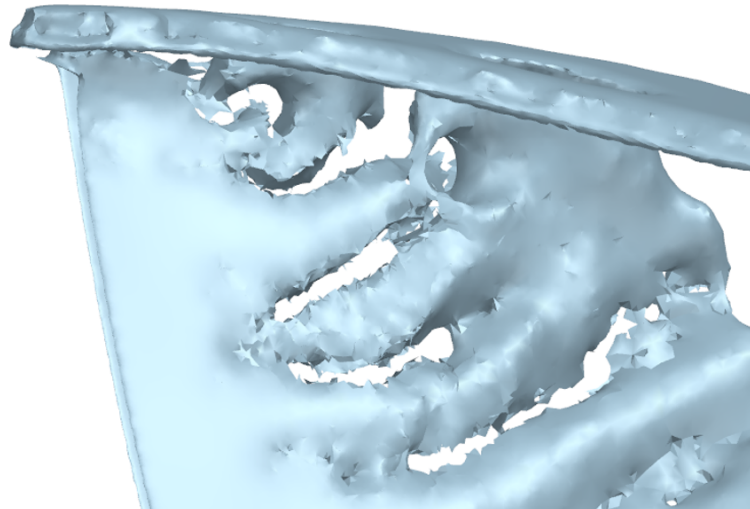
The remaining STL imperfections can be fixed manually. Commands like section view, snip, fill hole can be used to repair the geometry. Knowledge about the product is necessary to carefully repair the STL. In the use case 3, the region highlighted in the figure 5.15 consists of non manifold vertices that may act as small channels or can be dirty geometry as a result of poor optimization result.





**Figure 5.15:** Non manifold geometries in use case 3

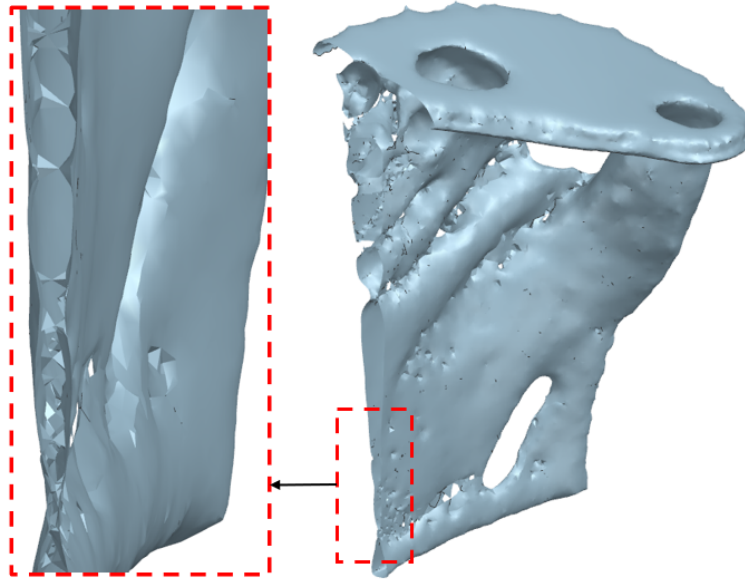
The non manifold geometries in use case 3 are decided to be snipped at places where no useful information could be sensed. The snipped facets can be suppressed or deleted. The snipped use case is displayed in the figure 5.16. The gaps created during this process filled using fill hole command.



**Figure 5.16:** Use case 3 with non manifold geometries removed

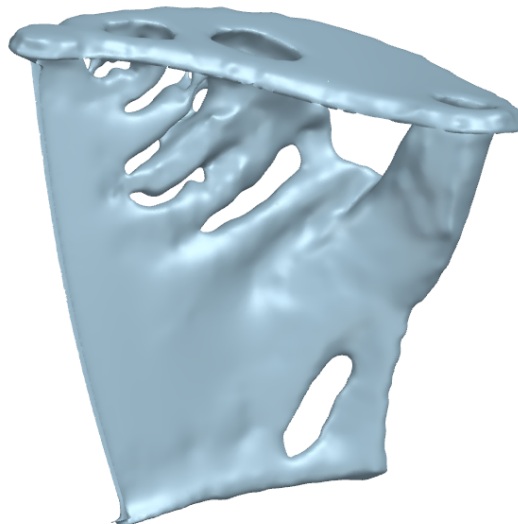
Non manifold geometries were also found in the inner channels of the STL geometry as shown in 5.17. As per the product and design expert, these non manifold geometry may signify the function of turbulators to distribute the air flow. Removing the non manifold geometries can affect the fluid flow in the channels. The presence of non manifold geometries signify the heat distribution in the region. Brainstorming to address this issue resulted in 3 different methods. 1) Create small obstructions that can function same way as the non manifold geometries so that the effect of turbulators are not compromised. 2) Create three to four approximate obstructions in the channels such that the fluid distribution remain same. 3) Narrow down the

area at the non manifold region so that the required of heat distribution is achieved. The third idea was considered since the other two ideas were difficult to implement and can lead to many iterations if obstructions are misplaced.



**Figure 5.17:** Non manifold geometries in the internal channels of use case 3

To transform the non manifold geometry, smooth facet body command was used. For better results, the non manifold regions were remeshed and then smoothed. The smoothing factor and iterations determine the internal channel area. The resultant STL file is displayed in the figure 5.18



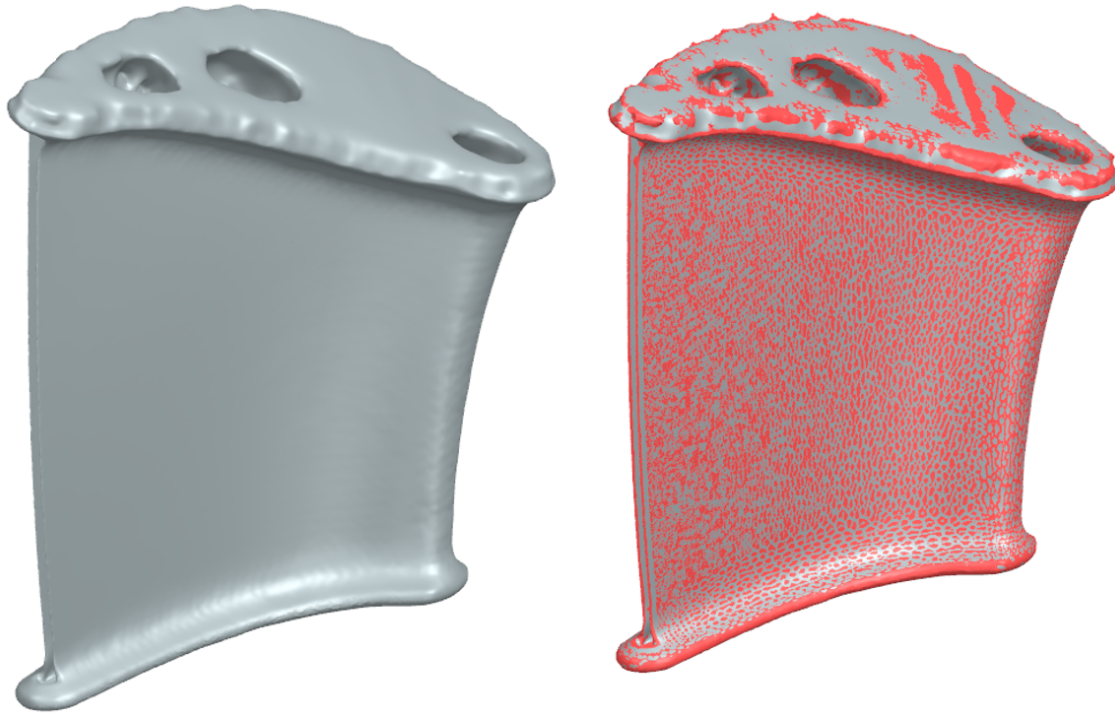
**Figure 5.18:** Smooth facet body result of use case 3

The smooth facet body command can also repair some imperfections like degenerated facets, folded facets, self intersections and non manifold geometry. Smoothing



can also lead to over approximation leading to loss of useful information. For example, smoothing of small and minute channels can lead to disappearance of the channels because of smoothing parameters being large compared to the geometry size. This is not desirable. In some cases, the dimension of features cannot be determined, one of the shortcomings of STL files, which make it difficult to predict the level of smoothing. Therefore, the level of smoothing need to be carefully handled.

Facet geometries tend to create gaps and more intersections during repairing process due to approximations. The combining process of non design space with the smooth internal channel created missing facets and self intersections. The gaps are addressed by fill hole while the self intersections were addressed by snip facet body and fill hole command. The resultant convergent solid body is shown in figure 5.19. The repaired use case 3 is verified by overlapping the raw STL file. It was found that the repaired STL traced most part of the raw STL except the erroneous areas. Since the deviation was found only at the erroneous regions and the overall function of the component is retained, the developed model was accepted.



**Figure 5.19:** Convergent solid body of use case 3 and verification of fixed geometry with raw STL. The red color body is raw STL and grey color body is the repaired solid model

## 5.5 Automation

The cleanup facet body is an inbuilt command and not all the issues could be successfully fixed. Some issues like degenerated facets, inconsistent facets could be repaired by cleanup facet body command or by smoothing facet body. Some

issues were not fixed using cleanup facet body command and this section details the automation work done to repair those STL issues.

From the study of three use cases, the most abundantly found STL issues were self-intersecting facets and laminar slits. Laminar slits get resolved if the self-intersecting facets are addressed. In the approach to repair self-intersecting facets, initially NX Journaling was used to record the script used in cleanup facet body command. However, back-end script used in repairing the self-intersecting facets could not be obtained. The algorithm used in cleanup facet body command is proprietary of Siemens NX and not accessible for the users. Hence algorithms need to be developed to fix STL issues.

User Functions commands available in Open C [43] were explored and commands specific to facet bodies were used to develop script for automation. Using the commands available, a script to remove redundant facets was developed.

### 5.5.1 Redundant facets

The facets with two or more than two adjacent facets can be identified by the commands available in NX facet package. The algorithm is given below.

---

Algorithm 1: Redundant facet

---

```
Foreach facet body in the STL file
  Foreach facet in the facet body
    Find adjacent facet for each facet
    If number of adjacent facets <2
      Remove the facet under consideration
    end if
  end foreach
end script
```

---

The automation script was developed in C# language and successfully executed on sample files. During this process, forum discussions were utilized. One of the findings include, convergent facet body do not work with 'AskAdjacentFacet' command line of NX API. It was found that NX facet body respond to all automation commands but the convergent facet bodies respond to some of the facet body commands. The redundant facets identified using the algorithm were attempted to preserve but NX facet body package was limited to delete the identified facet.

### 5.5.2 Self Intersecting facets

Self intersecting facets can be due to various reasons such as an edge sharing three facets, intersection due to numerical round off during tessellation. The issue is piercing of one facet with other facet. An algorithm to repair self intersecting facets is given below.

---

Algorithm 2: Self Intersecting facets

---

```
Foreach facet body in the STL file
  Foreach facet in the facet body
    Find the face of facet for each facet
    Ask if any edge intersect the facet face
    If any edge intersect the facet face\newline
      Remove the facet under consideration and
      the facets that possess intersecting edges
    end if
  end foreach
end script
```

---

Extensive search on 'ask if facet face is intersected by edge' were performed, forum discussions were conducted and programming experts were consulted but commands to execute the idea was not feasible. Multiple suggestions such as convert facet bodies into surfaces, create bounded plane were attempted, but the algorithm could not be successfully developed into an automation program.



# 6

## Discussion

One of the observations is the time taken to repair all the STL issues in the given use cases. The time taken to repair the STL issues for different use cases is summarised in the table 6.1. The use case 1 was considered to be inverse Boolean along the report. Effort to repair the use case 1 without considering Boolean operation was also performed to contrast the difference. From the observation, the time taken to repair the use case 1 using repaired STL of use case 2 was ten minutes and the time taken to fix issues in use case 2 was 45 minutes. Therefore, 55 minutes is the lead time to obtain error free CAD from raw STL of use case 1 using Boolean operation. The direct manual repair work on use case 1 consumed one and half hours. The difference is approach result in difference in lead time. Hence wise decision on approach in analyze step can save time in repair work.

Use case	Time taken	Approach
1	10 minutes	Boolean from use case 2
1	1.5 hour	Manual repair of raw STL
2	45 minutes	Manual repair of raw STL
3	45 hours	Manual repair of raw STL

**Table 6.1:** Time taken to repair all STL issues (first iteration)

From the observed lead times, a distinct difference in lead time can be noticed with use case 3 with respect to other use cases. The reason for taking long time to repair use case 3 is that the topology of the model is poor. The issue with optimised file is not entirely due to geometric imperfections, the output of optimization itself was poor. Comparing the given use cases, the topology of use case 1 and 2 were better and continuous as compared to use case 3. The functions of geometric representation were not clearly known while working with use case 3. For example, the non manifold geometries in use case 3 were considered to be turbulators at one region while considered to be dirty geometry in other region. Hence, the quality of raw STL also determines the amount of repair work required. In addition, it is a recommendation that the engineer need more knowledge about the product in order to mitigate the topology issues as well as geometry issues.

It was stated that the use case 1 and 2 were developed from recent TO algorithm as compared to old algorithm of use case 3. From the experimental work to fix issues, it can be observed that the STL outputs of newer algorithm are smooth, continuous and possess lesser STL issues. So the time and effort required to repair

the newer STL will be relatively less. However some attention need to be given to the minimum cross section dimension while performing topology optimization. The geometric issues in the areas of small thickness such as channels and turbulators, can lead to slender geometries after repairing and smoothing. It is recommended to maintain the minimum thickness dimension as two times the existing dimension.

NX CAD possess various features to handle STL issues. Maximum utilization of commands in polygon modeling can help in fixing STL issues. However it is to be noted that NX CAD can give rise to STL issues while fixing them. For example while stitching the preserved geometry with design space facet body as in use case 3, the command create flaws along the interface of the joint and to the other regions as well. Hence some repair work is required to fix these issues.

Regarding automation, Classic API possess more user function programs than Common API. But classic API do not possess some internal features as Classic API. It is recommended to use Common API since it will be developed for future by Siemens Digital Industry but Classic API will not be developed further. The databases for each programming language is available over internet which provides the complete list of functions and methods available in NX CAD. Given that all functionalities can be achieved independent programming language in Common API, the selection of programming language depends on the engineer's experience in programming. The user function programs in the database of Open C and Open C++ consist of examples which also help in understanding the syntax and functions. Forums are useful in finding solutions and clearing doubts.

NX CAD possess few user function programs for facet body. The NX facet body is recommended for automation purpose since convergent facet body is recently developed, not all the functions are capable to work with convergent facet body. Some issues observed while working with automation:

- 1) After execution of automation program, often the model not be recognized for cleanup.
- 2) After execution of automation program, there are graphic issues developed in working model.

The reason of the issue is not known but converting the facet body to convergent body and then back to NX facet body can help in eliminating this issue.

An insight of an interviewee is that entire repair technique cannot be automated. Some human intervention and engineering skills required to address various issues. The CAD system does not understand the reason for creating a channel or the reason to snip non manifold geometries in one region while smoothing in the other. Though this issue can be overcome if the CAE performance change can be actively visualized with the change in the CAD model, this development is not available at this point of time.

From the knowledge of repairing the use cases, a generic method is developed. The method to repair the STL issues vary depending on the input files. The generic method can be illustrated as a process flow as shown in the figure 6.1

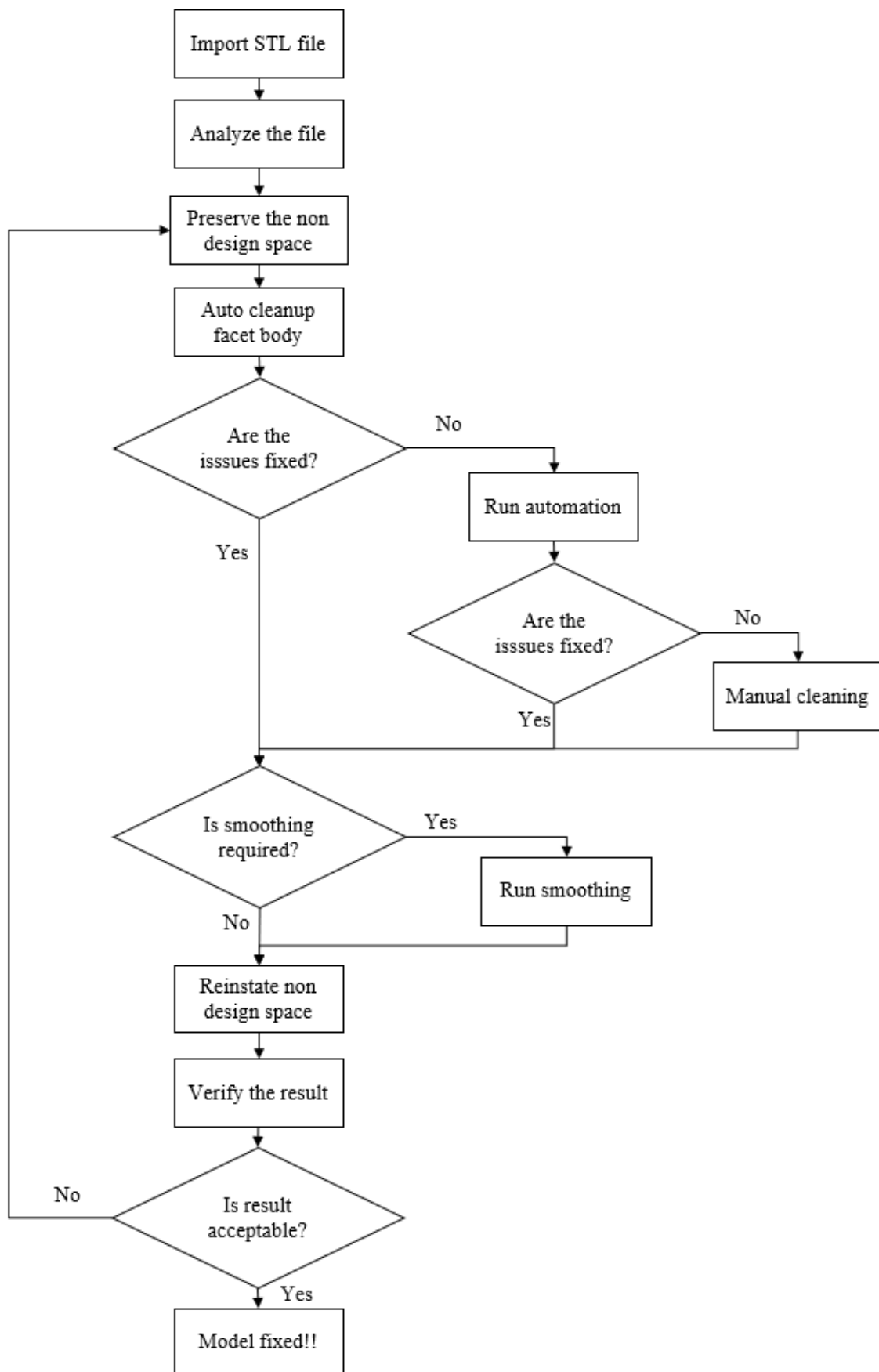


Figure 6.1: Method for repairing STL files

The first step in converting STL to CAD format for topology optimized structure is to import the binary STL geometries into NX CAD. Then analyze the imported file to determine the non design space, water-tightness, approach to repair the STL. Then preserve the non design space using snipping tool and hide or suppress. Run clean up the facet bodies command available in polygon modeling command. If all the issues are fixed by the inbuilt automatic STL repair command, check for requirement of smoothing if not subject the STL file to custom built automation programs. If not all issues fixed, run manual repair technique. When the STL is clean from all imperfections, check for smoothing. If smoothing required, run smooth body facet command. After smoothing check, use fill hole option or combine facet bodies to reinstate the design space. Verify if the model is free from errors and match the input raw STL by overlapping STLs. If the deviation is within acceptance, the new solid convergent facet body is the required CAD format of STL file.



# 7

## Conclusion

The problem in handling the STL geometries for TO structures was addressed in this work. In order to check if all the requirements of the thesis work achieved, the research questions are recalled and answered and some recommendations for future work is been provided in this chapter.

Research Questions:

1. What are the issues in topology optimized STL file that prevent design engineers to use it along with other CAD files in NX environment?

The geometric issues that are commonly found in STL files are listed in section 2.3. Not all the issues found in all STL files. The STL issues vary for each STL file.

2. What are the steps involved in development of NX CAD model from the optimized STL file?

A generic method to repair TO structures is detailed in the chapter 6. Using the method, future STL files can be repaired.

3. Is automation of STL repair possible? If possible, how to automate the STL repair technique?

Automation of STL repair process is possible in NX CAD. In some cases, human intervention is required to understand and work on the STL files. Hence not the entire process can be automated but processes that do not require human intervention can be automated. NX Open can be used to automate the repair of STL.

The erroneous TO STL use cases were converted into error free solid 3D models. Siemens Energy's requirement to build a watertight model that can be handled with Boolean operations was developed successfully. The method to assist in conversion of STL to CAD format is demonstrated. Using the generic method, future STL files can be converted by Siemens Energy engineers. The tools and commands used can be directly used for future works.

There are some limitations to the developed method. The developed technique is not robust. In other words, the STL files being repaired is subjected to change between Engineers and change between iterations. Since the method involves more

manual work, it is hard to obtain same result with different iterations. Also STL files are not similar to other CAD files which are developed parametrically using dimensions. Handling STL files is difficult, however, the method can be made more robust by automating the repair work. With limitations and benefits of different tools and capability of NX automation explained, there are good potential to develop the automation work further. Further exploration in automation can help in reduction of lead time and can reduce the manual labor.

One other limitation of the developed technique is the verification of the result. The developed convergent solid body is overlapped with the raw STL for verification. The technique is not robust. Since CFD based FEA could not be performed during the thesis work, overlap technique was performed. One approach to overcome the issue is by fixing the STL issues during TO. Since Matlab codes are used for TO, further research towards obtaining TO structures without STL issues while performing TO can eliminate the need to repair the STL files. This approach can reduce the lead time and eliminate manual effort required to repair STL files. Also the output of the technique will be more robust since it will not change with iterations unless input parameters are changed.

STL file format consist of basic information of 3D geometries. In the hierarchy of usefulness of the model, STL file forms the lowest level. Other file formats such as OBJ, 3MF as discussed in section 2.6 provide richer information than STL file formats. While STL allows the user to save with imperfections, 3MF format does not allow the user to save non manifold bodies. If the topology optimization performed in Matlab is capable of optimizing to derive 3MF format, the open geometries and laminar slits can be eliminated. OBJ format allows the user to store the model data as surface curves and surface patches (NURBS) which help in extracting accurate geometry and also eliminate the imperfections due to polygon facets. Extracting the TO output in the above format and investigating on the result can also be extended as a separate research work.

Further study to convert the non parametric solid model into a parametric CAD can help in performing changes to the optimized component. Future research to develop parametric component from STL can help immensely in the rapid development of products.

Third party tools that are specifically intended to handle STL geometric issues are available as discussed in section 2.5. Since the thesis work was focused on exploring NX capabilities, other STL repair tools were not investigated in detail. Though there are foreseen issues in using third party tools like losing reference of STL files when imported into CAD system, the other tools can be still explored to investigate if the third party tools can save more time and eases the work to repair STL file.

# Bibliography

- [1] Wohlers, T., Gornet, T. (2014). History of additive manufacturing. Wohlers report, 24(2014), 118.
- [2] Han, Y. S., Xu, B., Zhao, L., & Xie, Y. M. (2019). Topology optimization of continuum structures under hybrid additive-subtractive manufacturing constraints. *Structural and Multidisciplinary Optimization*, 60(6), 2571-2595.
- [3] Jihong, Z. H. U., Han, Z. H. O. U., Chuang, W. A. N. G., Lu, Z. H. O. U., Shangqin, Y. U. A. N., Zhang, W. (2021). A review of topology optimization for additive manufacturing: Status and challenges. *Chinese Journal of Aeronautics*, 34(1), 91-110.
- [4] About Siemens Energy. siemens. (n.d.). Retrieved May 24, 2022, from <https://www.siemens-energy.com/global/en/company/about.html>
- [5] “Siemens and E.ON reach milestone with 3D-printed burner for SGT-700 gas turbine,” 2018. [Online]. Available: <https://press.siemens.com/global/en/pressrelease/siemens-and-eon-reach-milestone-3d-printed-burner-sgt-700-gas-turbine>. [Accessed: 6-Apr-2022]
- [6] Sustainability of digital formats: Planning for Library of Congress Collections. STL (STereoLithography) File Format Family. (2019, September 12). Retrieved June 1, 2022, from <https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml>
- [7] Siemens 2007 Release Notes. NX CAD December 2021 Updates. (n.d.). Retrieved June 1, 2022, from <https://docs.sw.siemens.com/en-US/product/209349590/doc/PL20200605194735749.xid1753866/html/xid1759099/>
- [8] ESRC Framework. Ethics Guidebook. (n.d.). Retrieved May 14, 2022, from <http://www.ethicsguidebook.ac.uk/index.html>
- [9] Larsson, R. (2016). Methodology for topology and shape optimization: Application to a rear lower control arm (Master’s thesis).
- [10] Liu, J., & To, A. C. (2020). Computer-Aided Design-Based Topology Optimization System With Dynamic Feature Shape and Modeling History Evolution. *Journal of Mechanical Design*, 142(7), 071704
- [11] Chabod, S. (2021). A Topology Optimization Procedure for Assisting the Design of Nuclear Components. *Journal of Nuclear Engineering*, 2(2), 152-160.
- [12] Nitsopoulos, I., Stephan, M., & Böhm, M. (2009). An Efficient Approach for CFD Topology Optimization of interior flows. In 3rd ANSA & mETA International Conference.
- [13] - Han, D. (2019). Slicing of tessellated models for additive manufacturing based on variable thickness layers (Doctoral dissertation, Georgia Institute of Technology).

- [14] Ito, Y., Nakahashi, K. (2002). Surface triangulation for polygonal models based on CAD data. *International Journal for Numerical Methods in Fluids*, 39(1), 75-96.
- [15] Liu, S., Li, Q., Liu, J., Chen, W., Zhang, Y. (2018). A realization method for transforming a topology optimization design into additive manufacturing structures. *Engineering*, 4(2), 277-285.
- [16] Oropallo, W., Piegl, L. A., Rosen, P., & Rajab, K. (2018). Point cloud slicing for 3-D printing. *Computer-Aided Design and Applications*, 15(1), 90-97.
- [17] Szilvsi-Nagy, M., & Matyasi, G. Y. (2003). Analysis of STL files. *Mathematical and computer modelling*, 38(7-9), 945-960.
- [18] Ciobota, N. D. (2012). Standard tessellation language in rapid prototyping technology. *Sci Bull Valahia Univ*, 7, 81-5.
- [19] Wu, T., & Cheung, E. H. (2006). Enhanced stl. *The International Journal of Advanced Manufacturing Technology*, 29(11), 1143-1150.
- [20] Gibson, I., Rosen, D. W., Stucker, B., Khorasani, M., Rosen, D., Stucker, B., & Khorasani, M. (2021). *Additive manufacturing technologies* (Vol. 17). Cham, Switzerland: Springer
- [21] Chen, Y. H., Ng, C. T., & Wang, Y. Z. (1999). Data reduction in integrated reverse engineering and rapid prototyping. *International Journal of Computer Integrated Manufacturing*, 12(2), 97-103.
- [22] Huang, S. H., Zhang, L. C., & Han, M. (2002). An effective error-tolerance slicing algorithm for STL files. *The International Journal of Advanced Manufacturing Technology*, 20(5), 363-367.
- [23] Pezzoli, D. (2016). Topological analysis, healing and defeaturing of tessellated models.
- [24] Liu, F., Zhou, H., & Li, D. (2009). Repair of STL errors. *International Journal of Production Research*, 47(1), 105-118.
- [25] Roscoe, L. E., Chalasani, K. L., & Meyer, T. D. (1995, June). Living with STL files. In *Proceedings of the sixth international conference on rapid prototyping* (pp. 145-151). University of Dayton.
- [26] Leong, K. F., Chua, C. K., & Ng, Y. M. (1996). A study of stereolithography file errors and repair. Part 1. Generic solution. *The International Journal of Advanced Manufacturing Technology*, 12(6), 407-414.
- [27] Snyder, W. E., Groshong, R., Hsiao, M., Boone, K. L., & Hudacko, T. (1992). Closing gaps in edges and surfaces. *Image and Vision Computing*, 10(8), 523-531.
- [28] Barequet, G., & Sharir, M. (1995). Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12(2), 207-229.
- [29] Bohn, J. H., & Wozny, M. J. (1992). Automatic CAD-model repair: Shell-closure. In *1992 International Solid Freeform Fabrication Symposium*.
- [30] STL repair (Online amp; Offline): The best software of 2021. All3DP. (2022, June 10). Retrieved June 10, 2022, from <https://all3dp.com/2/stl-repair-fixer-tool-online-offline/>
- [31] Meshmixer is state-of-the-art software for working with triangle meshes. About Meshmixer. (n.d.). Retrieved June 1, 2022, from <https://www.meshmixer.com/>

- 
- [32] Top 7 best STL repair software for all levels. 3Dnatives. (2019, July 31). Retrieved June 1, 2022, from <https://www.3dnatives.com/en/best-stl-repair-software-240620194/>!
  - [33] Fusion 360 with Netfabb. Autodesk. (2022, June 2). Retrieved June 3, 2022, from <https://www.autodesk.com/products/netfabb/overview?term=1-YEAR&tab=subscription>
  - [34] How to repair STL files for 3D printing . Formlabs. (n.d.). Retrieved June 1, 2022, from <https://formlabs.com/asia/blog/best-stl-file-repair-software-tools/>
  - [35] Iqbal, K. (2019, September 10). Ply - polygon 3D file format. PLY - Polygon 3D File Format. Retrieved May 25, 2022, from <https://docs.fileformat.com/3d/ply/>
  - [36] Iqbal, K. (2019, September 10). AMF - additive manufacturing file. AMF - Additive Manufacturing File. Retrieved May 25, 2022, from <https://docs.fileformat.com/3d/amf/>
  - [37] 3MF Specification. 3MF Consortium. (2021, November 17). Retrieved May 25, 2022, from <https://3mf.io/specification/>
  - [38] Iqbal, K. (2019, September 10). OBJ file format. OBJ File Format. Retrieved May 25, 2022, from <https://docs.fileformat.com/3d/obj/>
  - [39] Ashley, S. (1991). Rapid prototyping systems. Mechanical engineering, 113(4), 34.
  - [40] Siemens. (n.d.). Siemens Documentation. Retrieved May 3, 2022, from <https://docs.sw.siemens.com/en-US/product/209349590/doc/PL20200605194735749.whatsnew/html/xid1851820>
  - [41] Siemens NX documentations. NX 2007 series documentation. (n.d.). Retrieved May 16, 2022, from <https://docs.sw.siemens.com/en-US/release/209349590/NX%202007%20Series>
  - [42] NX CAD. (n.d.). Version (2007). Retrieved from <https://www.plm.automation.siemens.com/global/en/products/nx/>.
  - [43] Open C Overview. Open C reference manual. (n.d.). Retrieved June 1, 2022, from [https://docs.plm.automation.siemens.com/data\\_services/resources/nx/10/nx\\_api/en\\_US/custom/](https://docs.plm.automation.siemens.com/data_services/resources/nx/10/nx_api/en_US/custom/)



# A

## Thesis Workflow

In order to provide the context of the thesis work, a workflow with some steps outside the thesis work is illustrated in the form of IDEF0 diagram in the figure A.1

The first step in the process is selection of component that to be optimized and printed. The selection of the component typically relies on cost for traditional manufacturing, scope of material reduction in the existing component, etc. The selected components are optimized using an CAE tools. ANSYS, NX CAD, Altair Inspire are some commercially available tools for topology optimization. For this thesis work, the input topology optimized STL files are provided by the research students working on TO research project at Linköping University. MATLAB script developed by research students at Linköping University is used for performing topology optimization. The input for topology optimization is a bulk volume of material consisting of design and non design spaces. The loading conditions and constraints are programmed in MATLAB. The 3D model is divided into finite number of small elements consisting of information on density of the material. As the simulations run for optimizing, the density information in each element is analysed. According to different loading conditions, the material is retained only where the load is carried by the component. The elements take density information as 0 when material is not required in a particular location and take value 1 where the material need to be present. Based on the density information, the topology of the component is obtained. The output of topology optimization is a 3D geometry in the STL format. The STL consists of number of errors and is the input for the thesis.

The literature study on different STL issues were used to diagnose the issues in the provided STL files. First step in the thesis is to perform literature study on STL and other CAD formats. Check how the STL errors are approached in previous research studies. Explore different options for repairing STL errors in NX CAD tool. Develop a method to manually repair the erroneous STL files. Verify if the repaired STL file functions similar to raw STL file. Implement the developed method on other use cases. With successful manual repair of STL, explore feasibility of automation of the work.

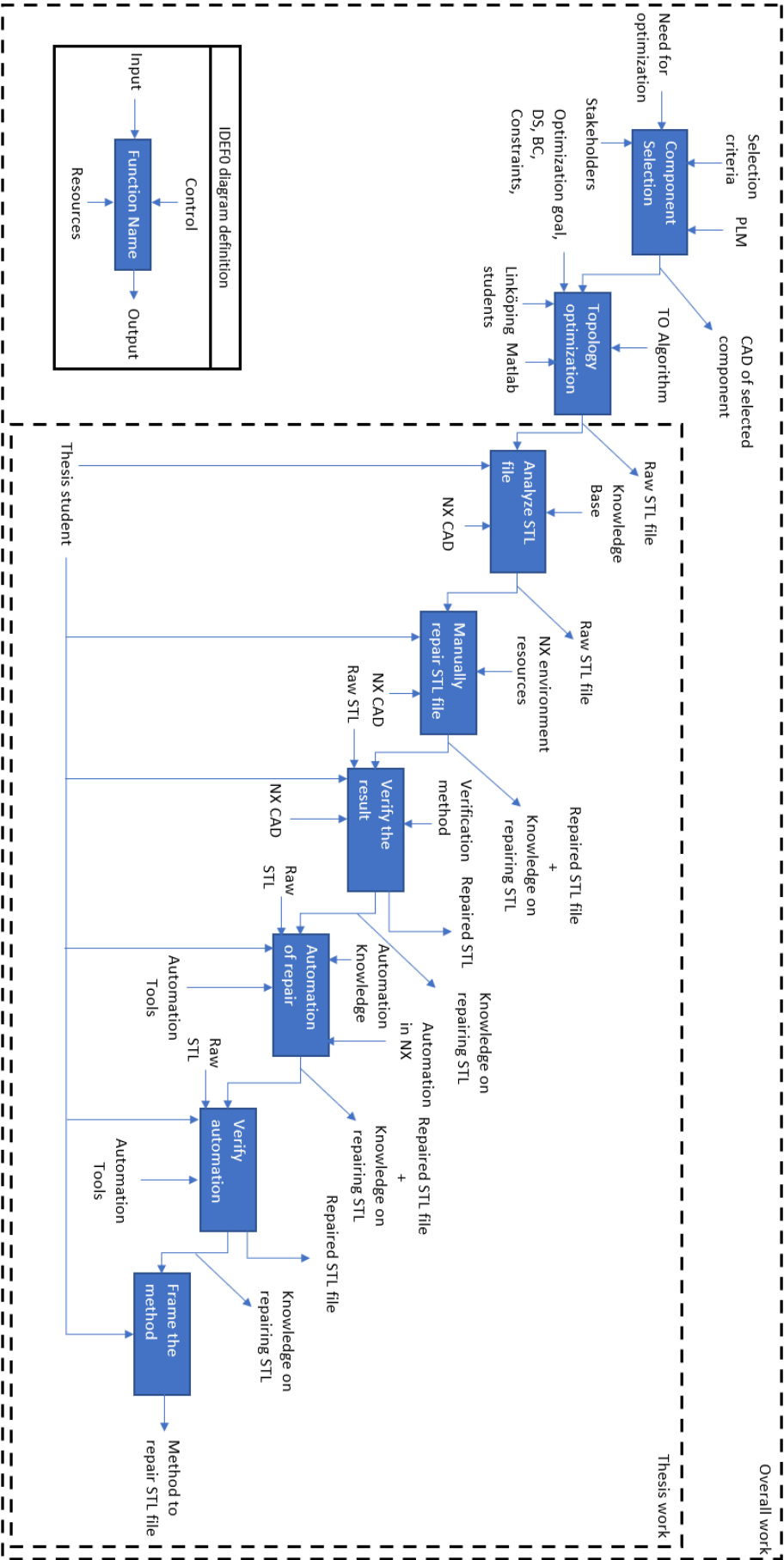


Figure A.1: Thesis workflow





# B

## Forum Discussion 1


The command AskAdjacentFacet was not working with convergent facet body. The question was raised in a public forum


### How to use AskAdjacentFacet function in NXOpen?

Asked 1 month ago   Modified 1 month ago   Viewed 36 times

  
0  


I am working with convergent facet bodies in NX and using C# in NXOpen. In the process, I am using **UFSession.Facet.AskAdjacentFacet** function to get the adjacent facets of each facet. But on using this particular command, the NXOpen throws error stating "*NXOpen.NXException: The facet object is not supported for this operation*". I went through the example given in NXOpen documentation

 ([https://docs.plm.automation.siemens.com/data\\_services/resources/nx/10/nx\\_api/en\\_US/custom/ugopen\\_doc/uf\\_facet/uf\\_facet\\_eg2.c](https://docs.plm.automation.siemens.com/data_services/resources/nx/10/nx_api/en_US/custom/ugopen_doc/uf_facet/uf_facet_eg2.c)) and used similar approach, but this error shows up any way.

 Below is the script that I tried.

...

**Figure B.1:** Stack Overflow question

Sample script attached

It was found that the convergent facet do not work with all the UFunc programs

```
public static void Main(string[] args)
{
    NXOpen.UF.UFFacet myFacet = UFSession.Facet;

    int facetID;
    int edgeID;
    int adjFacID;
    int edgeIDinAdjFac;

    int null_facet_ID = UFConstants.UF_FACET_NULL_FACET_ID;
    facetID = null_facet_ID;

    foreach (NXOpen.Facet.FacetedBody facetBody in workPart.FacetedBodies)
    {
        myFacet.CycleFacets(facetBody.Tag, ref facetID); // initialise for cycling


        while (facetID != null_facet_ID)
        {
            List<int> Adj_fac_list = new List<int>();
            for (edgeID = 0; edgeID < 3; edgeID++)
            {
                myFacet.AskAdjacentFacet(facetBody.Tag, facetID, edgeID, out adjFacID,
                if (adjFacID != UFConstants.UF_FACET_NULL_FACET_ID)
                {
                    Adj_fac_list.Add(adjFacID);
                }
            }
        }
    }
}
```

Note: I could use the same model tag and facet id in the function

**UFSession.FACET.AskNumVertsInFacet** and the script works fine. But I do not know why **AskAdjacentFacet** is not working. Can anyone help me on why there is an error and how to get this working?

**Figure B.2:** Stack Overflow question

## 1 Answer

Sorted by: Highest score (default) 

On first glimpse, the problem I see is that you have not initialized the variable `myFacet` and it's `null`. And since it's `null`, you cannot call its members.

So change a single line of the code from

```
NXOpen.UF.UFFacet myFacet = UFSession.Facet;
```

To

```
NXOpen.UF.UFFacet myFacet = UFSession.GetUFSession().Facet;
```

Share Edit Follow

edited Apr 12 at 0:31



Jeremy Caney

6,141 ● 34 ● 44 ● 70

answered Apr 11 at 19:03



Mech-Programmer

9 ● 4

Thanks @Mech-Programmer and @JeremyCaney!! I worked on the code as instructed but still I had the same error. Later I tried importing the same STL as NX facet body rather than Convergent body. The code works fine. I think since Convergent body is newly introduced by Siemens, not all UFunc commands work with it.

– Gopinath Siva Apr 12 at 10:55

[Add a comment](#)


Figure B.3: Stack Overflow question



# C

## Forum Discussion 2

Question while working with intersection facets -



**Govic** asked a question.  
April 13, 2022 at 12:27 PM

### How to extract and delete intersecting facets in a facet body?


Hi!!


I am working automation of facet clean up in NXOpen . I used "cleanup facet body" to repair the issues but found that it could not fix all the issues. The remaining issues are getting highlighted while using "Analyze" option in the builder but could extract or snip them. Running journal was not helpful. I am interested in extracting the intersecting facets. When I tried UF commands, I couldn't find command for extracting facet intersection rather tried AskFaceFaceIntersection, AskFaceSelfIntersection. But the facet geometry was not considered as the input object. I would like to know if there is a way to extract the facets those intersect in a list so that I can delete or disassociate from the parent facet body.


Thank you!!

NX Customization And Programming

NX FACET Modeling


 Like


 Answer


 Share



8 answers · 86 views

Figure C.1: Stack Overflow question

 **Elias Ghossein (Partner)**  
a month ago  
Use `UF_FACET_del_facet_from_model` to delete facet  
[Like](#) · [Reply](#)

 **Govic**  
a month ago  
Thanks!! I am using the same command for deleting the facet but need help on extraction of the intersecting facets though  
[Like](#) · [Reply](#)

 **Elias Ghossein (Partner)**  
Edited April 13, 2022 at 6:08 PM  
what do you mean by intersecting facets ?  
[Like](#) · [Reply](#)

 **Govic**  
a month ago  
  
[Expand Post](#)  
[Like](#) · [Reply](#)

**Figure C.2:** Stack Overflow question

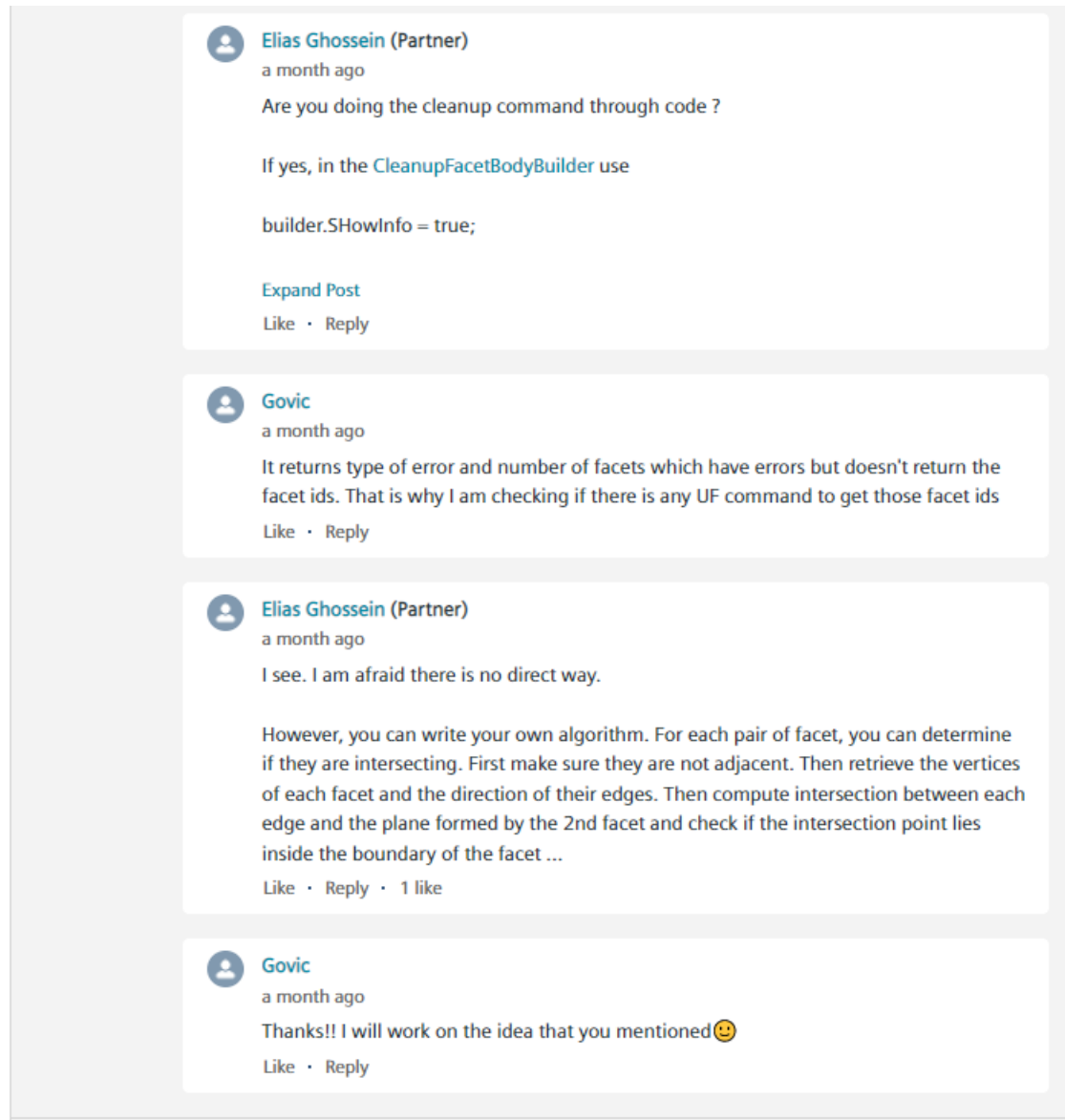


Figure C.3: Stack Overflow question

-

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY  
CHALMERS UNIVERSITY OF TECHNOLOGY  
Gothenburg, Sweden  
[www.chalmers.se](http://www.chalmers.se)



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY