Università degli Studi di Padova
Dipartimento di Ingegneria dell'Informazione

Master's Degree in ICT for Internet and Multimedia

3D MODELING FOR SPRINGBACK COMPENSATION IN SHEET METAL FORMING

Student: Matteo Candon
Advisor: Prof. Pietro Zanuttigh
Company Tutor: Giampaolo Pagnutti

February 27^{th}, 2023

Academic Year 2022/2023

Acknowledgements

I would like to express my deepest appreciation to the Inspire Studio Die Design Team, that followed my internship and the developement of the project with constancy. This experience allowed me to get useful insights for my professional and personal growth. In particular, Special thanks to my company tutor Giampaolo Pagnutti for all the help and patience demonstrated during these months, together with Massimo Origano for the support and availability whenever I needed a hint.

I would also like to thank my Thesis Advisor, Professor Pietro Zanuttigh, who helped me in this final step of my University career.

Finally, I could not have undertaken this journey without my parents, for their constant belief in me and the support during my years in University, and in particular during the internship.

Thank you.

Contents

Abstract

In sheet metal forming, a particular issue is given by the metal deformation and bending after the forming process. This springback phenomenon is caused by the physical characteristics of the material and must be taken into consideration. In order to avoid this deformation, the parts modeled in the design phase (Die and Punch) are modified to obtain the correct final form, on which the springback is compensated. In this thesis are presented some developed software functionalities that, from the original die design and the deformed metal sheet, compute the new design in order to compensate the springback phenomenon.

In the second chapter a brief introduction to Sheet Metal Forming and springback is given, in order to motivate why it is important to develop a software capable to compute the modified forming components. A description of the main concepts of 3D modeling is then presented, followed by an overview of Parasolid©, a geometric library used by Inspire™ Studio for many of its 3D modeling functionalities. After that the details of the software developed as an Inspire™ Studio plug-in are reported, with an overview of the different requirements fulfilled and the issues that have been overcome. Finally, in the last chapters, the results obtained in the developed project are described together with the possible future integrations and developments.

Sommario

Nello stampaggio della lamiera, un importante fattore di cui tenere conto è la deformazione del metallo e la piegatura che riceve durante il processo di stampaggio. Questo fenomeno, detto *Springback* o Ritorno Elastico, è causato dalle caratterstiche fisiche del materiale lavorato e deve essere accuratamente valutato e considerato. Per evitare questa deformazione indesiderata, le parti modellate in fase di progettazione (Punzone e Stampo) sono modificate per ottenere il disegno finale corretto, ovvero in modo da compensare il Ritorno Elastico.

Il primo capitolo presenta le motivazioni dietro lo sviluppo del progetto in esame, e una descrizione dettagliata della struttura della tesi. Il secondo capitolo contiene una breve introduzione al processo di stampaggio della lamiera e al fenomeno di Ritorno Elastico, in modo tale da spiegare l'importanza dello sviluppo di un software in grado di creare modelli corretti per lo stampaggio. A seguire, vengono presentati i concetti più importanti inerenti alla Modellazione 3D, insieme ad una panoramica su Parasolid ©, una libreria di funzioni geometriche utilizzata all'interno di Inspire™ Studio per le diverse funzionalità proposte. Si passa poi alla descrizione del plug-in software creato proprio per Inspire™ Studio, con un'analisi sui requisiti dello strumento di modellazione e le problematiche affrontate durante lo sviluppo. Infine, negli ultimi capitoli, vengono descritti in dettaglio i risultati ottenuti nel progetto e le future implementazioni.

1 Introduction

1.1 Motivations and scope of the project

Sheet Metal Forming is a widely used production process, from automotive to the packaging industry. Even though most forming processes have been applied for decades, the entire process is still not fully understood. This lack of understanding results in a cost and time consuming trial and error process to determine the proper process design that leads to the desired product [1].

The leading motivation for this project is the development of a plugin for the creation of deformed surfaces. This is computed using the data obtained from the simulation of Sheet Metal Forming in Inspire Form®.

Figure 1.1 shows a screenshot from the above mentioned simulation software, where can be seen an example of the components from the Sheet Metal Forming process called Die and Punch (see Chapter 2). The mathematical input for the plug-in is computed from these components.

1.2 Organization of the thesis

The project developed during the internship at Altair started after a first period of introduction to the concepts of metal forming and 3D modeling. For this reason,

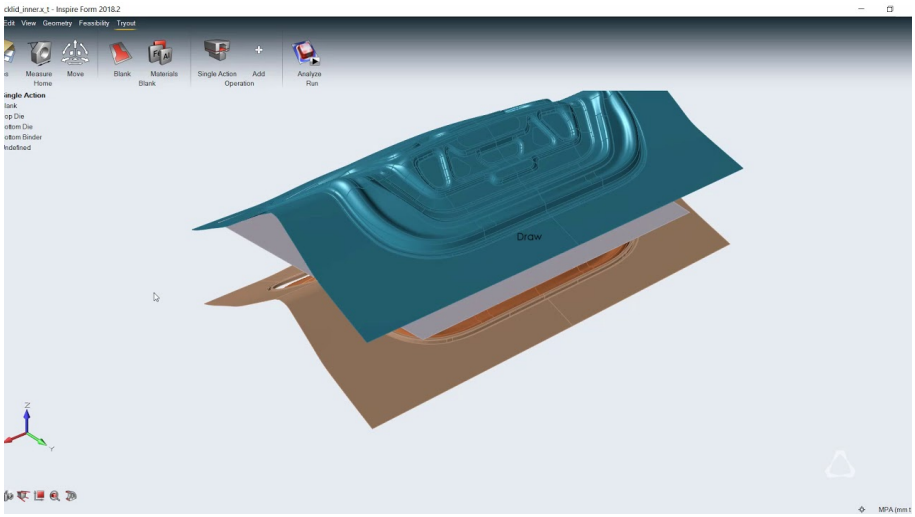

Figure 1.1: Design of forming components (Die and Punch, in brown and blue) used in Inspire Form®, together with the metal sheet in between (in gray).

before explaining the project itself, a brief introduction to these concepts and the instruments used will be presented.

Th First Chapter of this thesis describes the motivations that lead the company to develop the plug-in, together with the different objectives and the final scope.

The Second Chapter provides an overview on Sheet Metal Forming, which is the field of application of the final product obtained at the end of the project. Moreover, an explanation about Springback and Springback Compensation is presented, in order to address the issue to be solved. However, since it's not the aim of this thesis to provide a full understanding of the concept and its mechanical features, it is only briefly exposed.

Chapter Three presents some basic concepts of 3D modeling, providing a description of the main concepts used in CAD software. Following this, a presentation of the most important mathematical techniques that are used and have been helpful in the development of the plugin.

After having presented the 3D Modeling concepts, Chapter Four focuses on the geometric tool used to apply them: *Parasolid®*. It is a geometric modeling kernel owned by Siemens Digital Industries Software, and its functionalities were used for the developed plug-in.

The Fifth Chapter provides a complete description of the Modeling Tool created for this project, called CurveCompensate. In this chapter are reported all the steps that lead to the final results, starting from the first applications to the complete plugin. Following that, a detailed explanation of the issues that were met during development and how they were managed in order to reach the client's requests.

The Sixth Chapter overviews an analysis of the results obtained w.r.t. the requirements asked by the client, together with a representation of the final product.

Chapter Seven, the final section of this thesis, provides a resume of what was achieved during this study, together with the issues that have been overcome. Lastly, since the development of the project is only at its beginning, a presentation of the future implementations and possibilities is provided.

2

Sheet Metal Forming & Springback Compensation

This chapter will give an introduction to the process of Sheet Metal Forming, followed by an explanation of what is springback and the methods that are used to compensate for this phenomenon.

2.1 Sheet Metal Forming

Sheet metal forming is nowadays the most common approach for metal manufacturing, from simple cans to car bodies. The wide use of this method is due to the fact that it is the most cost-effective approach, and it can be easily automated in factories.

Sheet metal forming is defined by many different processes, but they all falls into two main categories: cutting and forming. Cutting is when the workpiece is stressed beyond its possibilities, while forming refers to when the workpiece is only bent in different shapes. For this project, only the forming is taken into consideration.

As the name suggests, the starting material is a plain metal sheet, whose thickness can range depending on the application of the final product. The sheet is then cut into individual blanks that are placed in the forming machine. This is mainly

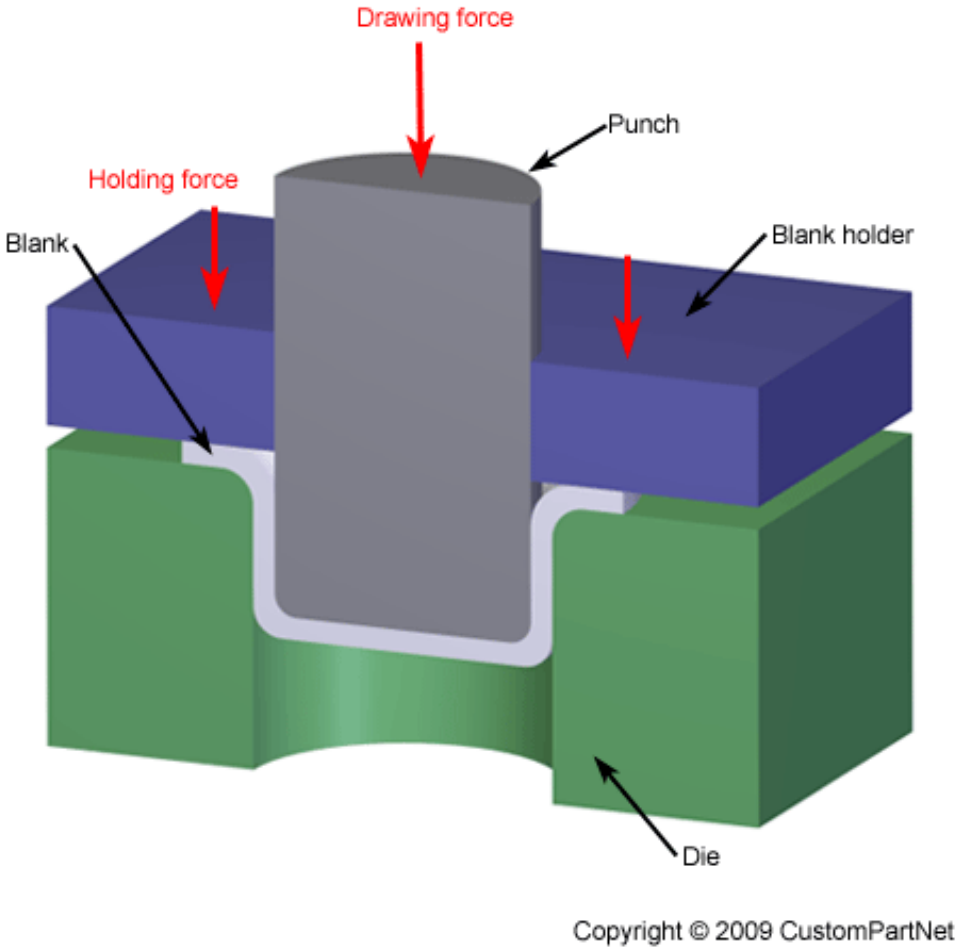

Figure 2.1: Schematic example of sheet metal forming components, together with the forces applied during the process.

composed of two parts, called *Die* and *Punch* (see Fig. 2.1): both components are shaped according to the final design. The first one, the Die (in green in the figure), is the component placed below the metal blank, and does not move during the operation. The Punch (in gray in the figure), on the other hand, presses the blank from above, imposing a force that deforms the metal into the desired final design. Depending on the final application of the product, different materials and dimensions must be taken into consideration in order to understand how to design the Die and Punch and determine the force to be applied.

2.2 Springback phenomenon

At the final stage of forming, when the load is removed and the stamping tools released, the product springs out. This deformation is caused by internal stresses (elastic deformation) and results in a change of geometry of the metal sheet. This phenomenon is called springback, and it's a significant problem in the process of sheet metal forming. To develop an accurate analysis of the final shape inaccuracies is an issue that is currently being studied, in order to obtain an effective and reliable methodology for springback prediction and compensation. The dimensional and geometrical change caused by springback is mainly influenced by the material and process variables, such as thickness of the sheet, material strength and Young's modulus [1].

Springback can be divided into categories, according to the changes in the shape and the action of forces. All these different categories (angle deviation, twist, distortion, etc.) can't be considered singularly, as they take place simultaneously during the forming process. Because of this, it is very difficult to describe how these springback behaviors influence each other and define the final result. In order to do this, many different approaches have been developed: some more simplistic, like the ideally plastic approach, others more accurate, like the plastic approach.

2.3 Springback compensation

As said above, springback is an issue that is still studied [2]. For this reason, a stable and accurate analysis of the phenomenon is required. To obtain it, numerical simulation software are used. This solution, anyway, presents both advantages and disadvantages: the advantages are in the analysis itself, that can be evaluated in different directions and compared with other models. On the other hand, the main disadvantage is the persistent low accuracy. This is given by the numerical simulation, and only a virtual analysis can lead to accurate results. The mathematical computation is made iteratively, by testing the new model and applying the new information obtained in the next step.

When an enough accurate methodology for the springback analysis is found, the next step is to find an approach to compensate for the unwanted deformation. Right now, two approaches can be considered:

- The first one, usually called spring-forward method, focuses on manual geome-

[1] The Young's modulus (E) is a property of the material that tells us how easily it can stretch and deform and is defined as the ratio of tensile stress (σ) to tensile strain (ϵ). Where stress is the amount of force applied per unit area ($\sigma = \mathrm{F/A}$) and strain is extension per unit length ($\epsilon = \mathrm{dl/l}$).

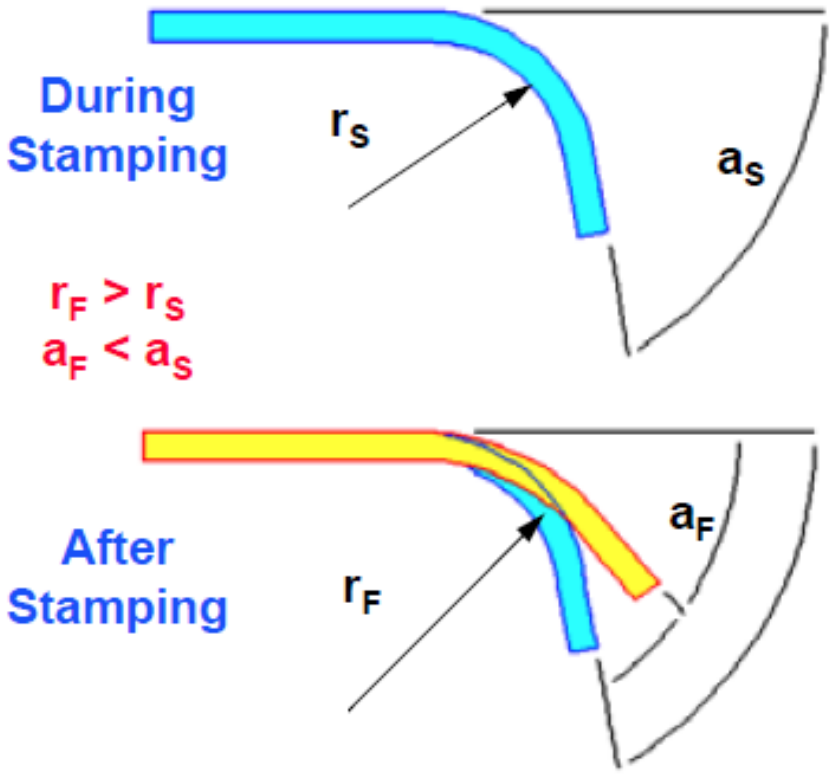

Figure 2.2: Simple visualization of springback in a one dimension case: the upper figure shows how the metal is bent during the forming process (with a radius rs and up to an angle as), while the lower figure shows what happens after the removal of the forming forces (the final bending is different and presented in yellow, where both the radius and the angle have changed w.r.t. the ideal values).

try compensation with CAD modelling. This approach is very time consuming, due to the manual surface modelling;

- The second approach, the more effective, is based on special computational modules. These modules correct the geometry after the springback analysis from the previous iteration.

Regarding this project, the idea for the inputs are the ideal surface to be obtained, and the deformed surface where springback took place. The method used to find this surface will not be analyzed here, as it's obtained from another modeling and simulation software called Inspire Form.

3

3D modeling & Inspire Studio

This chapter will give an explanation of the basic concepts of 3D modeling, like Boundary Representation and parametric modeling. In addition, the basics of the 3D environment visualization will be presented, always with a focus on the geometry involved. After that, an overview of the 3D modeling software Inspire Studio, that is one of the software produced by Altair, and the main instrument used during the development of this project.

3.1 Geometric 3D Modeling

Usually, when talking of 3D modeling, different kinds of software might come to mind. In particular, the main distinction is between Computer Aided Engineering (CAE) and Computer Aided Design. CAD refers to the use of a computer software to design and document a product's design. CAE, on the other hand, refers to an extension of CAD where the models developed are manipulated and used in physics simulations. Here, the focus is given on CAD software.

Geometric modeling is a branch of applied mathematics that focus its attention on the digital mathematical description of the geometry of the object. The CAD software can perform different kinds of operations, from the creation of the model to the application of transformations (e.g. rotation and scale). These objects might

be both two-dimensional and three-dimensional: in this second case, they are called solids. In order to create a 3D model of an object, the first step is to create a wire-frame model; an example can be seen in Figure 3.1. The faces between the different wires are then added by the software.

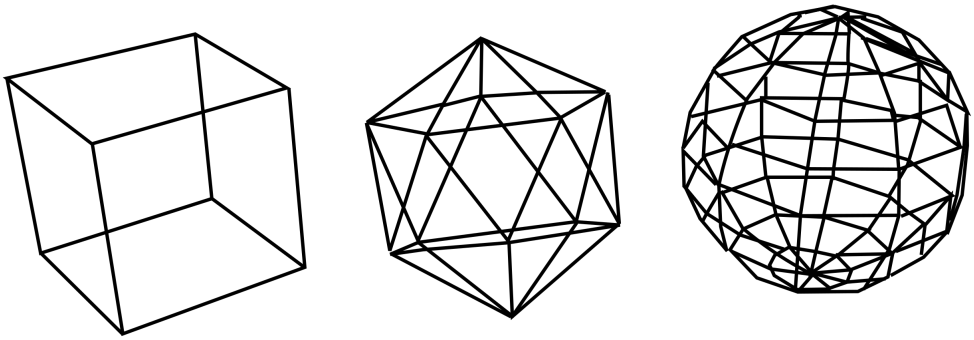

Figure 3.1: Examples of basic wire-frame models.

3.1.1 Solid Geometry

In modern CAD systems, these main methods are used for solid modeling:

- Constructive Solid Geometry (CSG);
- Boundary Representation (B-Rep);
- Parametric modeling.

However, instead of sticking to only one method, a combination of schemes and techniques is usually implemented in the representation of a solid.

3.1.1.1 Constructive Solid Geometry

In Constructive Solid Geometry (CSG), objects are represented in terms of a sequence of Boolean set operations (e.g. intersection, union) and rigid motion operators, starting with a given collection of primitive objects (e.g. cube, sphere cylinder) [3]. Given this combination of different elements, related by a Boolean operations, a simple yet efficient visualization of the process is given by a so called CSG-tree. An example is reported in Figure 3.2. This constructing method gives many different advantages: it's very concise, unambiguous (given the simplicity of the operations and of the primitives) and easy to create.

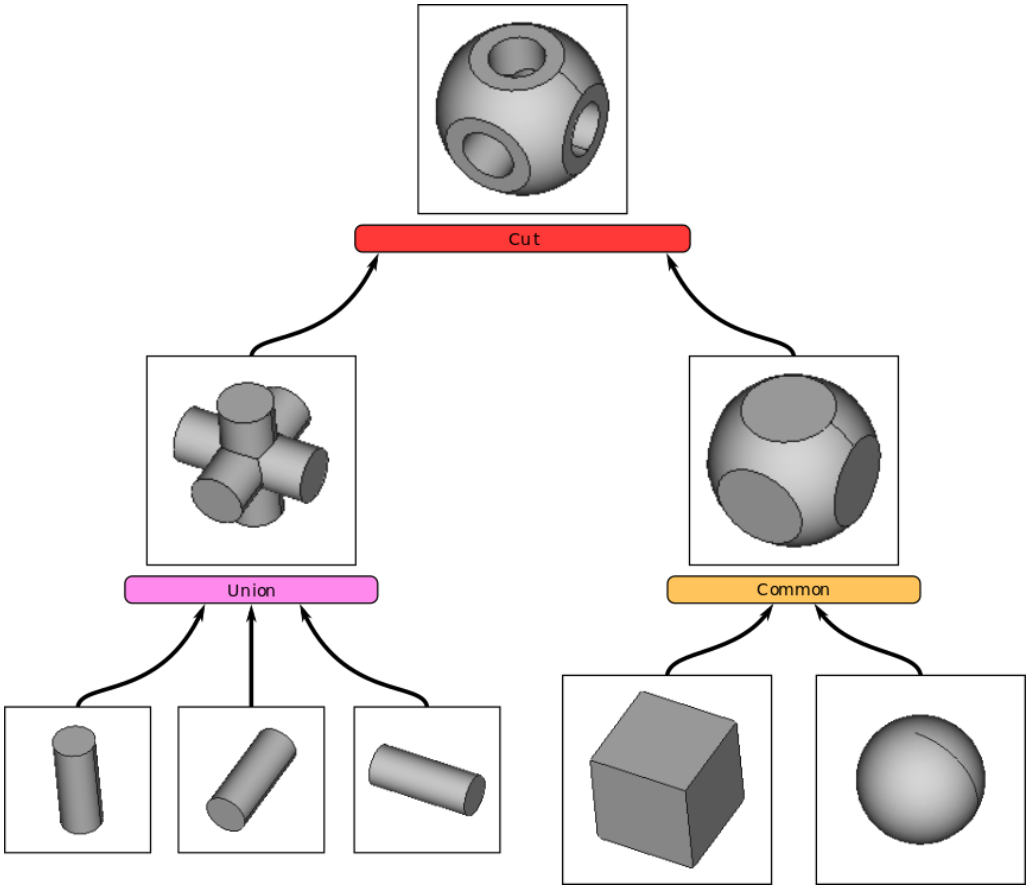

Figure 3.2: Representation of a CSG tree, from the bottom (primitives) to the final obtained object at the top.

3.1.1.2 Boundary Representation

Boundary Representation is probably the most common representation used in computer graphics. This method describe an object in terms of its surface boundaries: vertices, edges and faces. In particular, a face is restricted by its edges, and and edge is restricted by its vertices (hierarchical concept). In order to correctly represent an object given these information, a set of conditions must be confirmed. Here are reported some of them [4]:

- Each face must have (at least) three edges;
- Each edge must have two vertices;
- Each edge must belong to an even number of faces;
- Each vertex of a face must belong to precisely two edges of the face;
- All points must be distinct;

- Edges must be disjoint or intersect in a vertex;
- Faces must be disjoint or intersect in edges.

Some of these conditions are easy to verify, given the information of the element's relationships. Others (as check if two faces are disjoint) are more expensive to test.

3.1.1.3 Parametric modeling

A parametric model is defined by a set of parameters that controls the various properties of the entity. The entities with a parameter can be a physical dimension (the length of an edge), a Boolean primitive, or even a transformation. This type of modeling is very useful, since the user is able to modify each one of this parameters to obtain the desired result. In particular, when dealing with a software that implements Construction History (see chapter 3.2), the model regenerates its different parts automatically, storing the relationships between parameters of different entities.

For these reasons, parametric modeling is very useful when dealing with mechanical components that requires specific measurements and constraints. The main drawback, however, is that the structure of the model needs to be developed correctly, making the right choices by building relationships between entities and parameters. If this is not the case, the model might not support the variations to the parameters and the final result won't be the desired one.

3.1.2 Non Uniform Rational Basis-Splines

Non Uniform Rational Basis-Splines, usually referred as NURBS, are mathematical representations of 3D geometry that can describe accurately any wire shape and even surfaces. Due to their accuracy and flexibility, they are widely used in computer graphics. An example is presented in Figure 3.3.

NURBS curves are defined by four elements [5]:

- **Degree**: an integer number, usually between 1 and 5. It's strictly connected to the order of a NURBS curve (order = degree + 1). It defines the degree of the polynomial that represents mathematically the curve.
- **Control Points**: N points, where N = degree + 1, the easier way to modify a NURBS curve's shape in 3d modeling.
- **Knots**: knots are a list of degree + N - 1 numbers, each with its own multiplicity. They are a parameter that determines how the control points affects the NURBS curve

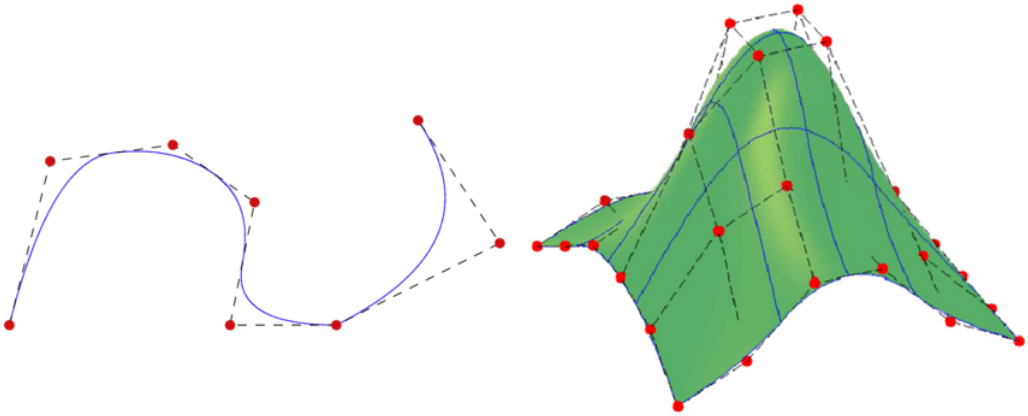

Figure 3.3: Representation of a NURBS curve (on the left) and a NURBS surface (on the right). Highlighted in red are the control points, used in 3D modeling as handles to control the shape of the curve.

- **Evaluation Rule**: it's a mathematical formula that takes a number ad assigns a point, based on all the elements above.

NURBS curves are a generalization of B-Splines and Bézier curves.

3.1.2.1 Bézier curve

Given a sequence of points p_i in R^m, with $i = 0, 1, ..., n$, the Bézier curve $p(u)$ with $u \in [0, 1]$ is defined by

$$p(u) = \sum_{i=0}^{n} B_{i,n}(u) p_i \tag{3.1}$$

where $B_{i,n}$ is referred as Bernstein basis polynomial of degree n

$$B_{i,n} = \binom{n}{k} u^i (1-u)^{n-i} \tag{3.2}$$

and n is the number of control points of the curve.

This component is actually part of the Bernstein polynomial approximation of a function f, a polynomial function defined as

$$F_n(f)(u) = \sum_{i=0}^{n} f(\frac{i}{n}) B_{i,n}(u) \tag{3.3}$$

These polynomials were used to give a constructive proof of the Weierstrass approximation theorem, which showed that every continuous function could be approximated by a polynomial.

3.1.2.2 B-Splines

Differently from Bézier curves, B-Splines do not change and recompute the entire curve when a control point is changed. There are three main types of B-Splines, distinct by their parameterisation:

- uniform B-Splines, for which the knots are equally spaced in the parameter interval;
- non-uniform B-Splines, for which the knots are arbitrarily set in the parameter interval;
- non uniform rational B-Splines (NURBS).

They are defined by a basis function $N_{i,j}(t)$, of order j and degree $j-1$ with respect to the knot vector t

$$N_{i,0}(t) = \begin{cases} 1 & \text{if} \quad t_i \leq t < t_{i+1} \quad \text{and} \quad t_i < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t) \tag{3.5}$$

where t are the values in the knot vector and $j = 1, .., p$ with p degree of the curve [3]. Then the B-Spline curve is defined by

$$C(t) = \sum_{i=0}^{n} P_i N_{i,p}(t) \tag{3.6}$$

3.1.2.3 NURBS surfaces

A Non Uniform Rational B-Spline surface of degree (p,q) is defined by

$$S(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) w_{i,j} P_{i,j}}{\sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) w_{i,j}}$$

where $N_{i,p}$ and $N_{j,q}$ are the B-Spline basis functions, $P_{i,j}$ are the control points,and the weight $w_{i,j}$ is the last coordinate of the homogeneous point $P^w_{i,j}$.

The value p defines the degree along the u direction, while q defines the degree along v. The control points describe a grid on the surface, and a change in the position of one of them leads to a shape change of the surface in both dimensions.

3.1.3 Rendering and visualization

In CAD software, the user need to visualize the objects created in a 3D environment and interact with them.

The first thing to notice is that, in order to represent points and objects, the information are stored in *homogeneous coordinates*. These are particularly handful because by adding a dimension, it's possible to avoid computations that are supposed to deal with points at infinite position. In particular, the last dimension takes value 0 or 1, where 0 indicates that the point taken into consideration lies at the infinite.

3.1.3.1 View Frustum

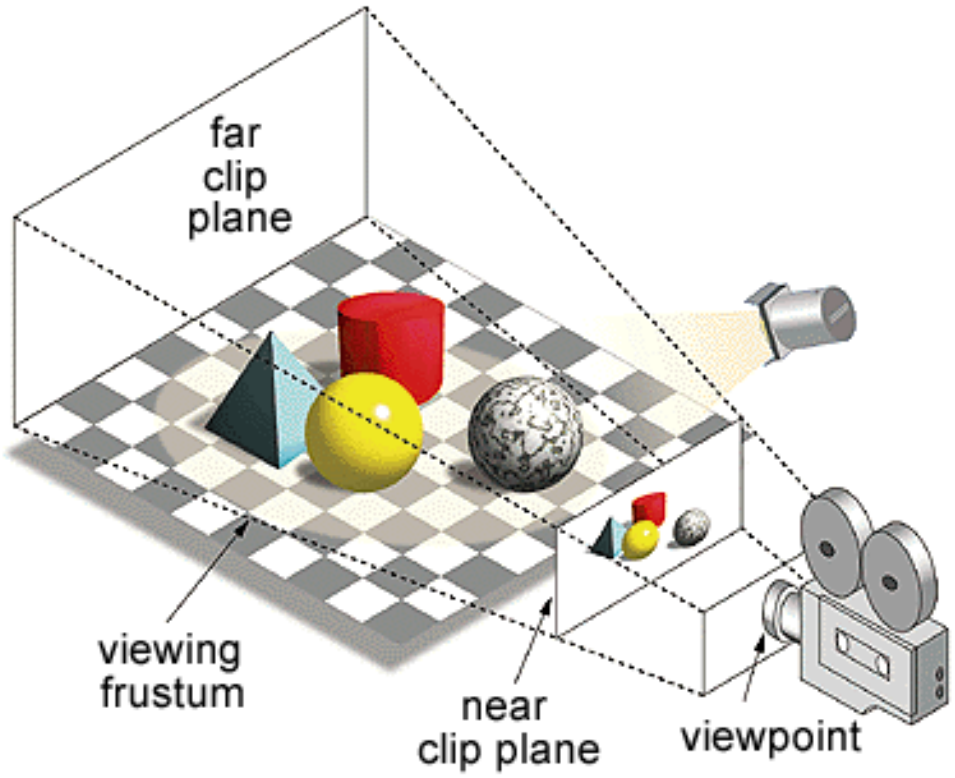

Figure 3.4: Simple visual representation of the viewing frustum: in the 3D environment is considered the position and orientation of a camera, which is used to report the point of view of the user.

Figure 3.4 represents the structure for visualization of a 3D environment, from a two-dimensional point of view. To achieve this, two possible techniques can be used: parallel projections and prospective projections. Parallel projections are useful because they keep intact distances and parallelism, but is not a realistic representation. Prospective projections, on the other hand, return a 2D image.

In order to represent a three-dimensional system on a plane, a few steps must be considered:

- First, a 3D object has to be defined (in a geometric representation);
- Then, the Point Of View (POV) is the point where the observer (the user) is positioned inside the 3D system. With this is also associated the direction of view;
- A projection plane, which is usually perpendicular to the direction view;
- A View Reference Point (VRP): it' the intersection between the projection plane and the direction view.

From all these elements, it is possible to define a 2D reference system parallel to the one associated with the observer.

3.1.3.2 Rendering

Rendering is the process of generating a picture from a 2D or 3D template. In other words, it's the operation that obtains a photo-realistic (or non photo-realistic) image from a geometrical definition of an object or a scene. It features shading, texture mapping, shadows, and other light phenomena like refraction and reflection. Another important feature of rendering is Hidden-Line, Hidden-Surface Removal (HLHSR): from the geometrical information of the 3D object and the Point Of View of the user, this algorithm removes from the 2D image those lines and surfaces that are not supposed to be visible (some examples of commonly used algorithms for HLHSR are Z-Buffer and Painter Algorithm).

3.2 Altair Inspire Studio

Inspire Studio is a 3D modeling, rendering and animation software by Altair Engineering. It is used by designers, architects and artists to create and visualize designs for a wide variety of different applications. A very important feature of this software is the construction history: this implementation allows the user to "automatically backtrack to an earlier stage of your design, reusing or modifying objects form past stages in real-time." [6]. Thanks to this, the 3D model can be refined and instantly re-generated. Figure 3.5 shows the software environment and interface.

One of the libraries used by this software is Parasolid, a geometric modeling kernel. This is better described in the next chapter.

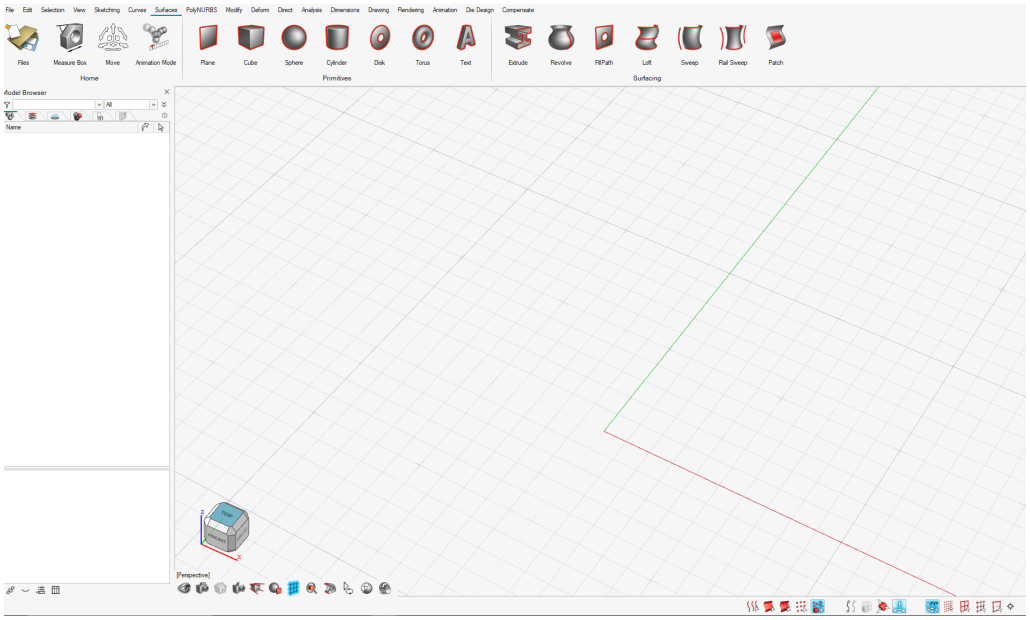

Figure 3.5: Inspire Studio interface, from the 2022.2 version. This is the version in which the modeling tool was developed.

4
Parasolid

In order to better understand how the project was developed, in this chapter is reported an explanation of the above mentioned Parasolid geometric modeling kernel. It is developed by Siemens Industry Software, and thanks to its wide variety of functions and support on many modeling techniques (e.g. solid modeling, direct editing, and free-form surface/sheet modeling) is integrated in a lot of the nowadays used 3D modeling software [7]. The concepts presented here are taken from the Parasolid On-Line Documentation Web, V34.1.153.

4.1 Entity classes

Figure 4.1 reports a visual aid to understand the structure of the Parasolid classes and entities. An entity can be defined as any identifiable component of a model. Also, any model can be represented as a network of entities that are linked by simple or complex relationships. In particular, an important distinction between entities that needs to be made. In fact, there are three different typologies:

- Geometric entities;
- Topological entities;
- Other entities.

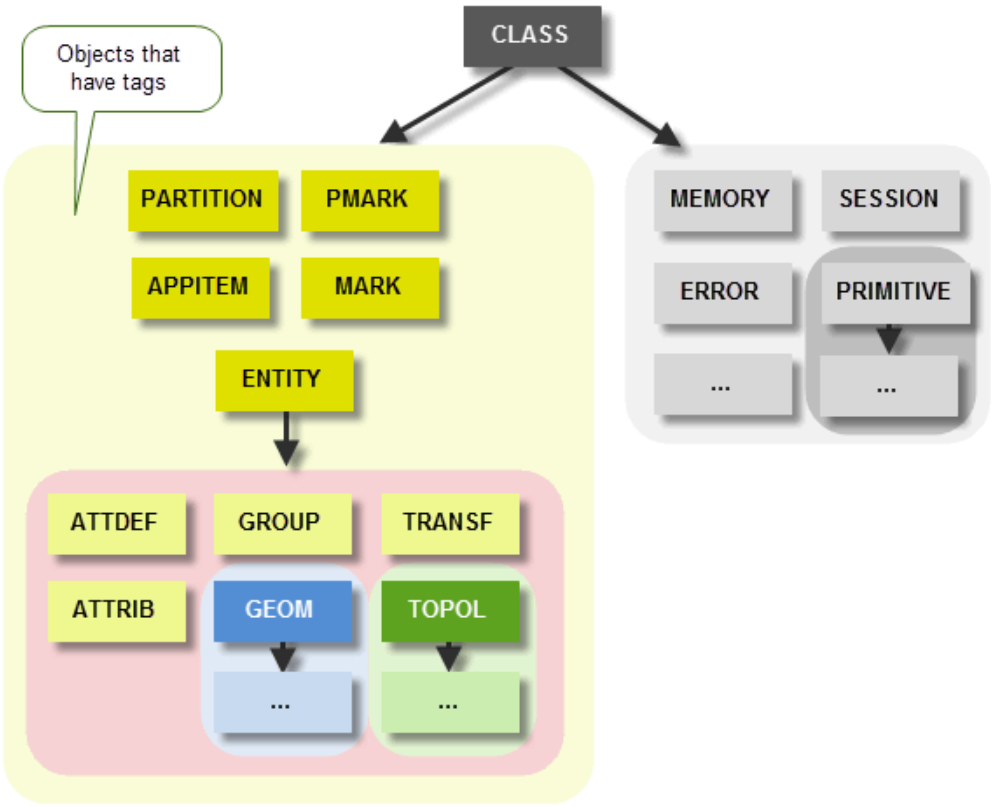

Figure 4.1: Visual representation of the Parasolid class hierarchy.

4.1.1 Topological entities

Topological entities comprise all the entities that constitute the structure of a model. These entities are classified as:

- **Body**:fundamental element in the Parasolid Kernel (better described below in section 4.2);
- **Face**: bounded subset of a surface, whose boundary is a collection of zero or more loops;
- **Edge**: bounded piece of a single curve, whose boundary is a collection of zero or more vertices;
- **Vertex**: represents a point in space, zero-dimensional analogy of a region.

4.1.2 Geometric entities

The function of geometrical entities is to specify the geometric shape of a body or its components. The geometrical entities are:

- **Lattice**;
- **Surface**: element attached to a face, or used as construction geometry for a body;
- **Curve**: element attached to edges or fin of a model;
- **Point**: element attached to vertices or bodies.

4.2 Bodies

Bodies are fundamental for modeling with Parasolid. They are composed of one or more connected entities, or even more than one body (*compound bodies*). For this reason, the first classification for bodies is between standard, compound and child. Standard refers to the basic unit of modeling, compound refers to bodies that are composed of multiple bodies. These last ones take the name of child.

Another important distinction when explaining bodies is between Manifold and Non-Manifold geometries. Manifold bodies allows disjoint lumps to exist within the logical body, whereas Non-Manifold do not allow it. In other words, a Manifold body is "Manufacturable", so it can be shaped from a single block of metal. Regarding this classification, Parasolid further distinct Manifold bodies as:

- Acorn, with zero dimensions and composed by one or more points in space;
- Wire, a one-dimensional body made of edges;
- Sheet, which is two-dimensional and can be either open or closed;
- Solid, a three-dimensional body that occupies a finite volume.

Non-Manifold bodies are simply described as General bodies.

4.3 Tokens, Tags and Identifiers

Every time an entity is generated, Parasolid associate to it a unique integer number, that calls *Tag*. These tags are different for each session, so the tag associated to a certain entity may change. For this reason, the Kernel also stores *Identifiers*, that remain unaltered between different sessions when loading the same saved file. *Tokens*, on the other hand, are numerical values used by Parasolid to define classes and other types of information regarding entities or functions used.

5

CurveCompensate Modeling Tool

This chapter provides a full report on the aim of the plug-in, together with the steps of the development and the improvements made to obtain the correct results. The initial idea for the plug-in was to develop a tool that, taking as input a polymesh surface, a matrix of points on the surface and the vectors that defines the deformation computed from the springback analysis, returns as output the deformed surface. The reason behind the use of polymesh is simply defined by the output from Inspire Form, after computing the mathematical analysis.

5.1 Initial approach and requests

Keeping in mind this starting idea, the development of the plug-in started with a simpler approach. For this reason, the surface to be deformed and studied is built with a constant section along one arbitrary direction (see Figure 5.1). This surface represents the ideal geometry of the metal sheet after the forming process. Together with the input surface, a single wire object is given: this is the Target Curve, the section of the metal sheet after the springback (obtained from Inspire Form). Since the section is constant, there is no need to share the entire deformed surface and the process to obtain it from the surface is a trivial boolean operation. Moreover, in this case the problem changes from three-dimensional to two-dimensional, making

it simpler to deal with.

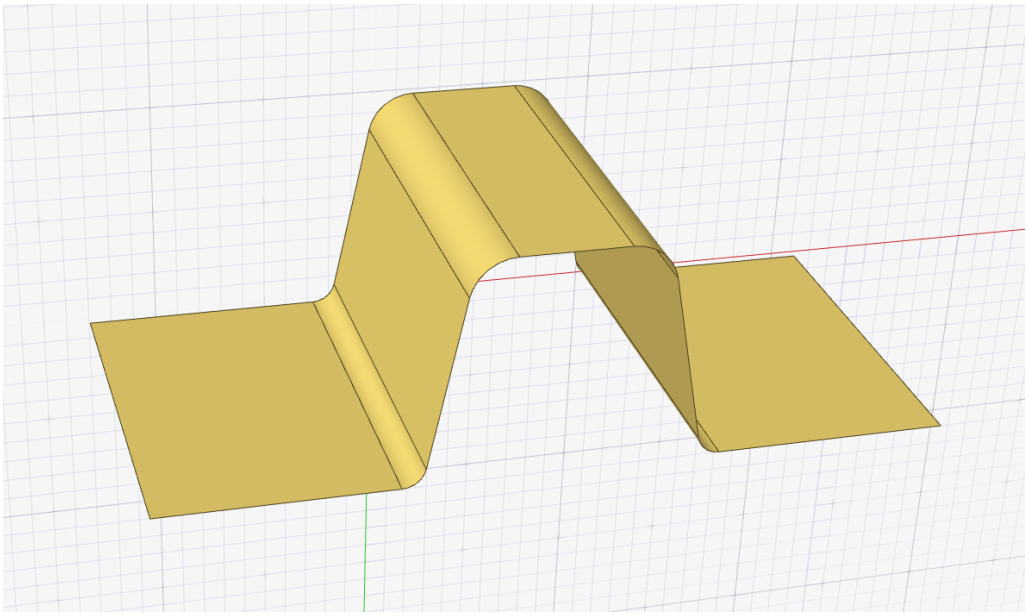

Figure 5.1: Example of input surface. This represents the ideal output geometry of the forming process.

5.1.1 Requests

Initially, the main request done was to allow the user to pick the faces on the surface and set them as **constrained**: this is done to prevent those faces to be deformed when using the tool. In order to do this, an additional algorithm was needed, because if the faces required no deformation, they are not allowed to move from the original position. This causes a problem in the creation of the deformed surface, since it needs to be not-disjoint. This will be better explained with the blending curves in Chapter 5.5.

Another implementation requested is the scale factor, called *Deformation factor* in the plug-in. This factor is modifiable by the user, and defines how the deformation takes place between the original surface and the Target Curve. For this reason, the factor had an initial range of 0-1, where 0 is the original section of the surface and 1 is the Target Curve position. In Figure 5.2 is shown the basic idea of the deformation factor between the section of the surface and the Target Curve. Later in the development, the range of this parameter has been widened to allow negative values, up to -1.0. This was required to compensate for the deformation of the springback, since it provides an opposite deformation to the one shown from the Target Curve.

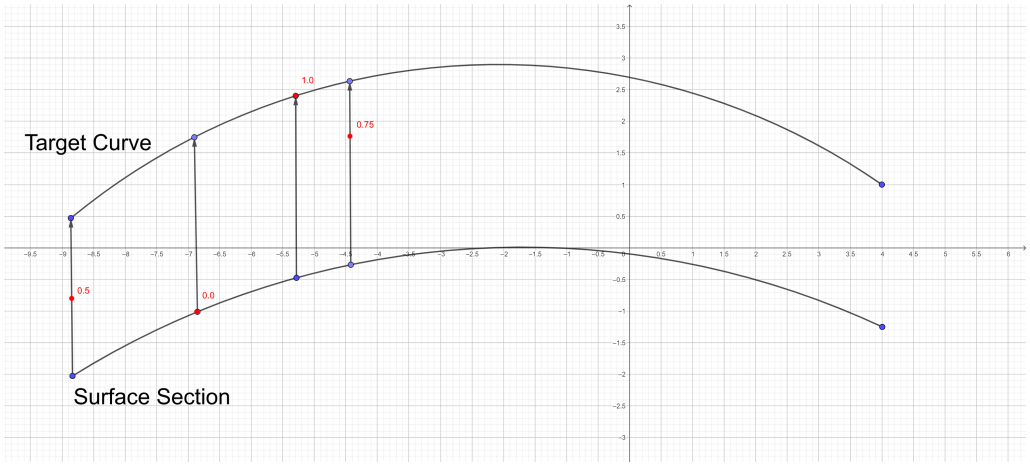

Figure 5.2: Visual representation of the deformation factor in different values. By computing the distance between the two curves, the deformation factor allows the user to decide at which distance the final deformed surface should be.

Other implementations have been made during the development, and will be presented in the next chapters.

5.2 Pre-processing of the input

To set a coordinate system for the two objects, the Target curve is set along the $z = 0$ plane, and the surface's constant section is also on the same plane. This was done only to perform a first attempt at the tool, and was later changed (see Chapter 5.3).

The first thing that has been done is to create a curve, or more precisely a wire object, that is the constant section of the original surface. To do this, a couple of functions from Parasolid were very helpful: first, a plane object (a geometric entity) was created on $z = 0$. After that, a rectangle was created based on the plane and the dimensions taken from the box of the surface. This allowed to make a topological entity, with which a boolean operation with the original surface returned a wire object. This wire is sure to be planar, and laying on the same plane of the Target Curve previously positioned. Once obtained the section, a pre-processing must be made, in order to create a correct relation between the two curves. First of all, the curves are each stored inside a body, and they are both made of one or more edges. To every edge is related a curve, that is the edge's geometrical entity. All these curves are defined by a length and an orientation. For example, let's say that an edge has vertices A and B: the related curve can be oriented to go from A to B or B to A. In order to set a usable reference system, they need to be oriented

the same way, or at least the information regarding the orientation of each curve has to be stored in a variable and used in the next steps of the plug-in. Secondly, since these curves will be used as reference system, they need to be described by a parameter, such as their length. For this reason, the curves will undergo a process of parameterisation.

5.2.1 Curves' Orientation

The algorithm's idea to create a standard orientation in each curve is pretty simple, but it required a few passages. In Algorithm 1 are reported the steps that leads to the correct storage of the ordered lists of vertices and edges on a curve. Notice also that the orientation is aligned w.r.t. the first edge found, and propagated to all the other edges.

Regarding the functions used in the algorithm, they are Parasolid inquiries functions to the topological entities in the wire body. The geometrical kernel allows to ask any vertex which edges are connected to it, and similarly ask any edge the connected vertices. From these simple functions, it was simple to scan the body. The only choice to be done is where to start scanning: for this reason, the first vertex stored is found in a preliminary scan of all the vertices, that stops when the first spur vertex (connected with only one edge) is found.

The orientation process is run through both wire bodies (Section and Target Curve). After that, for the reference system to be complete, the orientation of both curves needs to be the same in the 3D coordinate system. In order to do this, a simple check on the spur vertices was made, to verify if the chosen starting vertices were "on the same side" of the curves. This was achieved computing and comparing the distance between the first vertex of the section and both spur vertices (first and last, in the ordered list), to verify if the first vertices were also the nearest. Otherwise, the orientation of the Target Curve is reversed in order to match the one of the section curve.

5.2.2 Arc-Length Curve Parameterisation

After the creation of the ordered lists of edges and vertices, it is easier to compute the remaining data necessary for the deformation. In particular, the information that still need to be stored are the curves related to the single edge, their parameterised form and their lengths (both of the singular curves and of the entire wire bodies).

For this reason, another function was created inside the plug-in to scan through the bodies. Parasolid is again a useful tool for extracting the geometrical informa-

Algorithm 1 Set Curve Orientation

Require: Body of the curve
Ensure: Ordered list of edges, ordered list of vertices

```
N ← n                                              ▷ Number of vertices
i ← 0
while i ≤ N do
    if UnorderedListVertices[i] = spur then
        OrderedListVertices[0] = UnorderedListVertices[i]
        break
    end if
    i ← i + 1
end while
OrderedListEdges[0], nEdges =AskAdjacentEdges(OrderedListVertices[0])
                                                   ▷ There can be only one
orientation = AskOrientation(OrderedListEdges[0])
if orientation then ReverseOrientation(OrderedListEdges[0])
end if
PropagateOrientation(OrderedListEdges[0], CurveBody)
                                                   ▷ Align all edges' orientations
j ← 0
while j ≤ N − 1 do
    TemporaryEdges, nEdges = AskAdjacentEdges(OrderedListVertices[i])
    if j > 0 AND nEdges == 1 then
        break
    end if
    if j == 0 then
        verticesNear = AskVertices(TemporaryEdges[0])
                                   ▷ Always two vertices returned in the array
        OrderedListVertices[1] = verticesNear[1]
    else if j > 0 then
        if TemporaryEdges[0] == OrderedListEdges[j - 1] then
            OrderedListEdges[j] = TemporaryEdges[1]
        else if TemporaryEdges[0] != OrderedListEdges[j - 1] then
            OrderedListEdges[j] = TemporaryEdges[0]
        end if
        TemporaryVertices = AskVertices(OrderedListEdges[j])
        OrderedListVertices[j + 1] = TemporaryVertices[1]
    end if
end while
```

tion from the topological entities, and the curves are easily obtained together with their length. The most interesting step, however, is the creation of the curve approximation. The PK function creates a re-parameterised approximation of a curve, and allows two types of parameterisation: Arc-Length and Even. The first one provides a parameter value proportional to the length along the curve, so that the interval is [0,L] where L is the length of the curve. The second one, instead, normalizes the length so that the interval is [0,1]; in the case of circles, the parameterisation interval is [0, 2π] using the angle as parameter. These new parameterised curves are useful considering the possibility to pick a point along the curve by knowing the distance from the initial point of the curve. On the other hand, obtaining the parameter along the curve given the point might become more difficult, since the new curve is an approximation of the original and the point picked might not results to be belonging to the parameterised curve. For this reason, an additional step was added to the deformation function in order to ensure that the points are correctly selected and considered along the curve.

5.2.3 Deformation Function

The main idea for the deformation function is given by the Parasolid Kernel, which also provides functions for the deformation and transformation of the 3D objects. To deform the surface, single points on every face are passed to a particular function, controlled by the developer. Each point undergoes a deformation in its coordinates, and the surface is re-computed from the position of all the points. Regardless of the dimension of the surface body, the number of points easily surpasses 10.000 each time the deformation function is called, so the result is very precise.

In order to correctly position each point, the first thing to understand is where the point is w.r.t. the coordinate system previously created. So, each point is firstly projected on the section plane, resulting in a point along the section. From this, a scan of each parameterised curve is run to understand where the point is, obtaining the index of the oriented curve where the point lies and the parameter along its length. As said in the previous section, the points might not result to be belonging to the parameterised curves: this is due to the parameterisation itself, but also to the projection on the $z = 0$ plane. In fact, simple operations might results in a slight variation of the coordinates' values, making it difficult to compute the deformation. For this reason, a pre-processing of the translated point is performed: given the point and the curves of the section, a PK function is run to understand which of the curves is the nearest to the point. Once found, the point is projected on the curve itself, to ensure that the evaluation of the parameter can be performed correctly.

Once the parameter of the point along the entire section is found (sum of all the curves before the one where the point lies, and the parameter relative to the curve), it's straightforward to find the point that lies on the Target Curve where which the parameter is the same. To achieve this, the parameter is firstly normalized, dividing the value by the section's length, and then reported on the Target Curve. Since this curve might also be composed of many different edges, the uniformed parameter is multiplied by the Target Curve's length. The correct edge (and curve) where the correct point lies is found by summing the parameterised curves' lengths, following the order previously found. Once the correct index of the curve is found, an evaluation of the parameter is done to find the final point.

Now, the deformation factor is used to understand where the final surface needs to be computed. The calculations are simple, and made for both x and y coordinates. The last one is now not considered, since it's managed differently and does not change (the final point will have the same distance from the section as the point on the original surface, so the z coordinate is unchanged).

The equation for each coordinate is:

$$d_P = p_S + (p_T - p_S) * \lambda \tag{5.1}$$

where d_P is the coordinate of the final deformed point, p_S is the coordinate of the point found along the section, and p_T is the coordinate of the point found along the Target Curve at the same parameter value. λ instead refers to the Deformation Factor.

This method is working on some strong assumptions, that will later be dropped when considering a more advanced development of the plug-in.

5.3 Reference and Target Curves

After the first implementation was done, a few other requests were made to have a more complete tool. The main addition was the Reference Curve, another wire body as the Target Curve but representing the ideal final surface (similarly to the surface given itself). The main difference from the section of the original surface is the length: in fact, the Reference Curve and the Target Curve are now considered to be smaller or equal to the section. This requires that the deformation takes place only in between the two curves, leaving out the rest of the surface (as the surface might be longer than the curves). A visual representation of the two curves is given in Figure 5.3. These curves can be shorter, longer or the same length w.r.t. the surface's section, and each case requires a different management. For example, for

shorter curves a blending function is required (see Chapter 5.5), while for longer curves it's necessary to find the correct starting-ending parameter. The different approaches to the models are presented in Chapter 5.6.

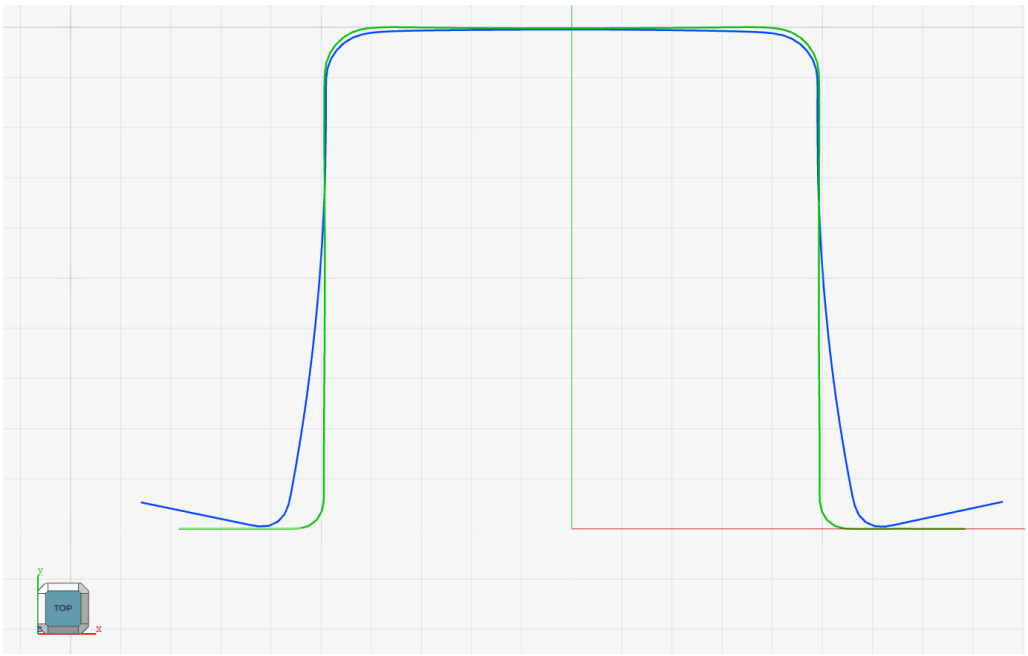

Figure 5.3: Curves used for the development and testing of the plug-in; in blue is represented the Target Curve, that is the section of the sheet deformed after springback. In green, the Reference Curve, that is the section of the sheet ideally formed.

5.3.1 Computing Curves' plane

One of the first improvements that was done after adding to the input the Reference Curve was the development of robustness to rotation and translation of the model from the $z = 0$ plane. The assumption of the Target Curve (and now also the Reference Curve) lying on that plane was dropped. Instead, the plane is computed starting from the two input curves, the only assumption is that they have to be on the same plane, otherwise the plug-in will return an error message.

The first step is to verify that the single curves are planar: this is still something that is required from the user, and the plug-in simply verifies that it is true. After that, the two planes need to be the same geometrical entity. This cannot be done directly through the Parasolid Kernel inquiry functions, as the planes might have a slight difference. For this reason, a tolerance value was considered when comparing the two entities. The planes returned from the PK functions of the singular curves are defined by three elements: a location, that is a point in space, and two vectors

that are applied to the location. One off them belongs to the plane, the other one is the normal to the plane. The first thing to verify is that the planes are at least parallel, and this is done comparing the normal vectors. If they have the same value (w.r.t. the tolerance given), the planes can be considered parallel. The second check was done computing the distance between the locations and the opposite plane: this is easily returned from a PK function. If the distance is equal to zero (w.r.t. the tolerance given), the location of the Target Curve plane belongs to the plane of the Reference Curve, and viceversa.

From the obtained plane, it is now possible to also compute the section of the input surface. This is done with the same idea presented in Chapter 5.2, but the plane is no more constrained to be $z = 0$. Instead, it is now computed every time the input is picked from the user, or it is rotated/translated.

The main difference in the deformation function is that the z coordinate can no more be considered differently from the others. For every point extracted by the surface, a translation vector was computed to move the point on the Reference-Target plane. after the computation of the final point is done, the translation vector is again applied with negative values to restore the position w.r.t. the surface.

5.4 Constraint Faces

As previously presented, one of the main implementation required from the tool is the possibility from the user to pick some of the faces of the original surface to set them as constrained. This particular implementation required some pre-processing of the inputs, mainly due to the fact that the picked faces had to be reported in the oriented reference system in order to correctly manage the points in the deformation function. Moreover, the main problem brought from constraint faces is the necessity to create a ”bridge” between the deformed components of the new surface with the undeformed components, that are the selected constraint faces. Figure 5.4 shows the picking of a constraint face in one of the models used during the plug-in development.

5.5 Surface blending

An important step in the development of the plug-in is the implementation of the blending curves. These were added to meet some requirements that were added after the first draft was done. In fact, blending appeared to be necessary to provide smoothness between different regions of the deformed surface, such as between constraint faces and deformed faces, or for surface outside the Reference-Target area.

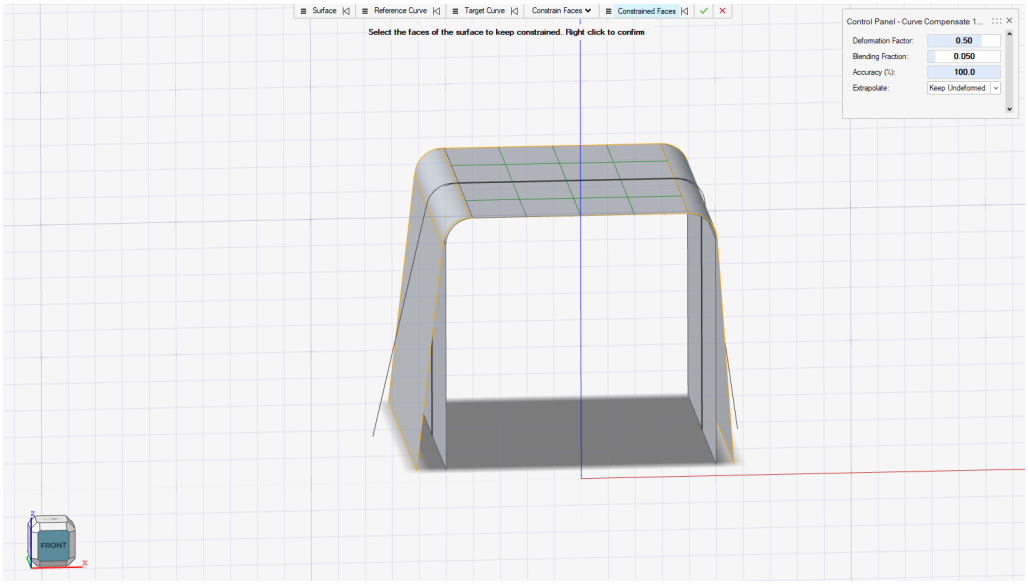

Figure 5.4: Picking of a face, that will be considered Constrained. Inspire Studio allows to have a visual representation of the elements selected, that are highlighted with a green grid.

Since the deformation works with a two-dimensional problem, the solution for the blending was developed with the same reasoning. In fact, to create a conjunction between these regions were used B-Curves (seen in Chapter 3.1.2). These curves are built by defining four control points, two knots 0,1 both with multiplicity 4. The two external points of the curve were easily obtained from the two regions that required to be blended. Notice that, since the constrained regions must not be modified, the blending takes place on the deformed region, using a fraction of total length to create the curve. This parameter, called Blending fraction, was made modifiable from the user, allowing a range from 1% to 40% of the total Reference Curve length.

Regarding the internal control points, their position were decided keeping in mind that the final deformed surface must have tangency continuity. For this reason, the second point was found starting from the first one, and performing a translation along the tangent direction found in the first point. The distance of this translation is based on the total distance between the first and last point, and is set to be a third of this length. This also means that the points change whenever the Blending fraction is modified. The third point is found with the same geometrical computation, but reversed to keep the point in between the external ones. Figure 5.6 reports a representation of these concepts, for a better understanding of the method used to achieve the best results.

In Figures 5.5 and 5.6, The Blending Curve is represented with a dashed line. In green is the Reference Curve, while in blue is the Target Curve. In yellow is

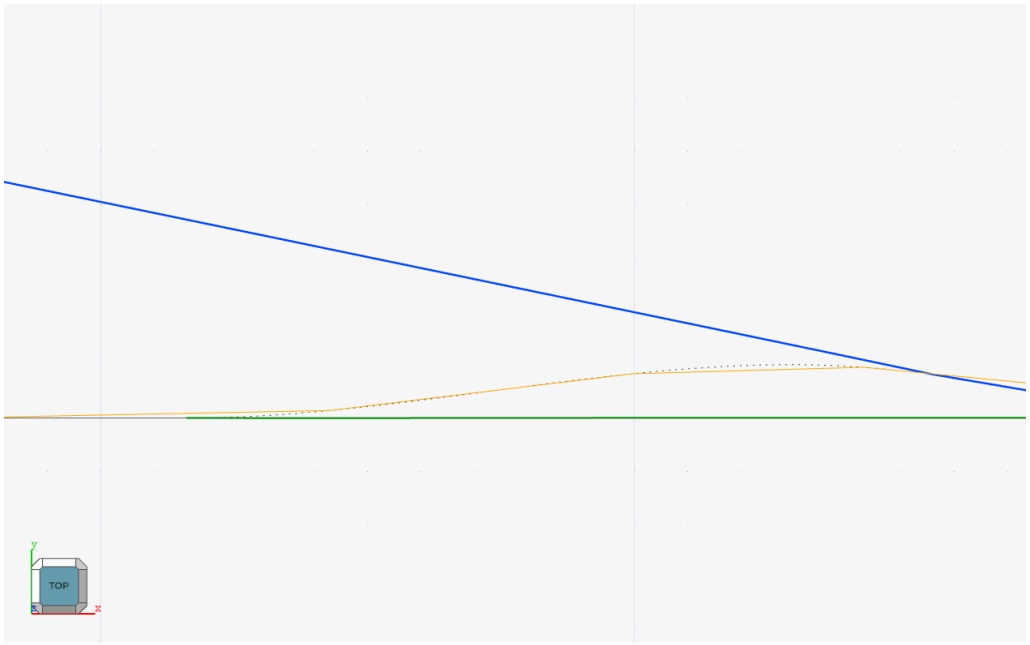

Figure 5.5: Visualization of the Blending Curve between the external area of the surface (outside the Reference-Target area) and the internal deformed region. The control points are not visible.

reported the profile of the deformed surface.

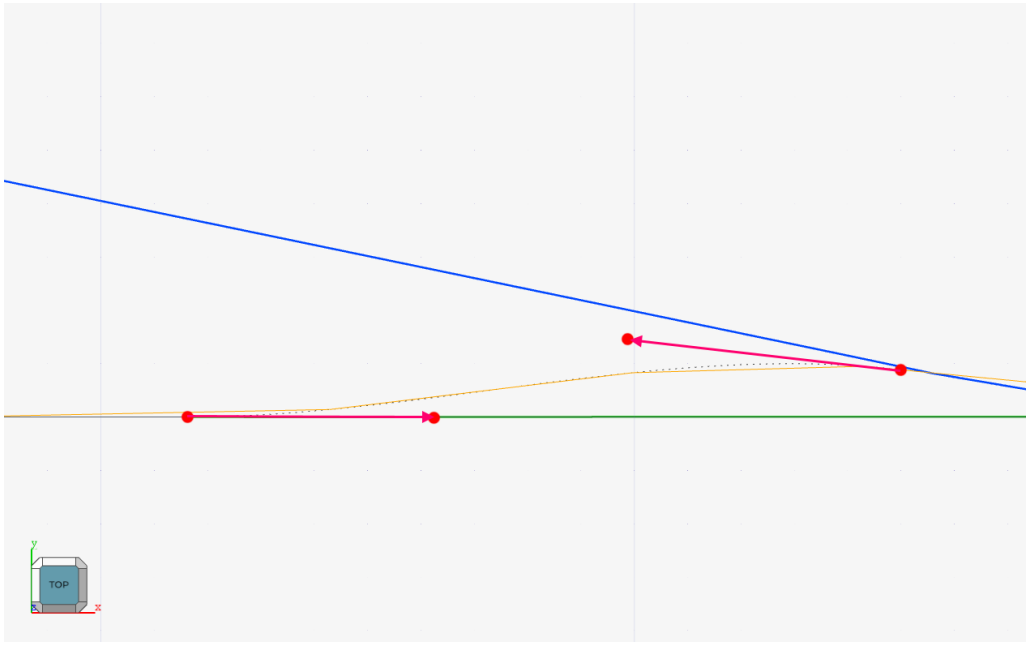

Figure 5.6: Visualization of the control points and tangent vectors used to find the internal control points starting from the external ones. In red are reported the control points and the tangent vectors used to obtain the two intenal points.

5.6 Models used

To verify if the plug-in created can be used in real applications, some models were provided during the intermediate phase of the development.

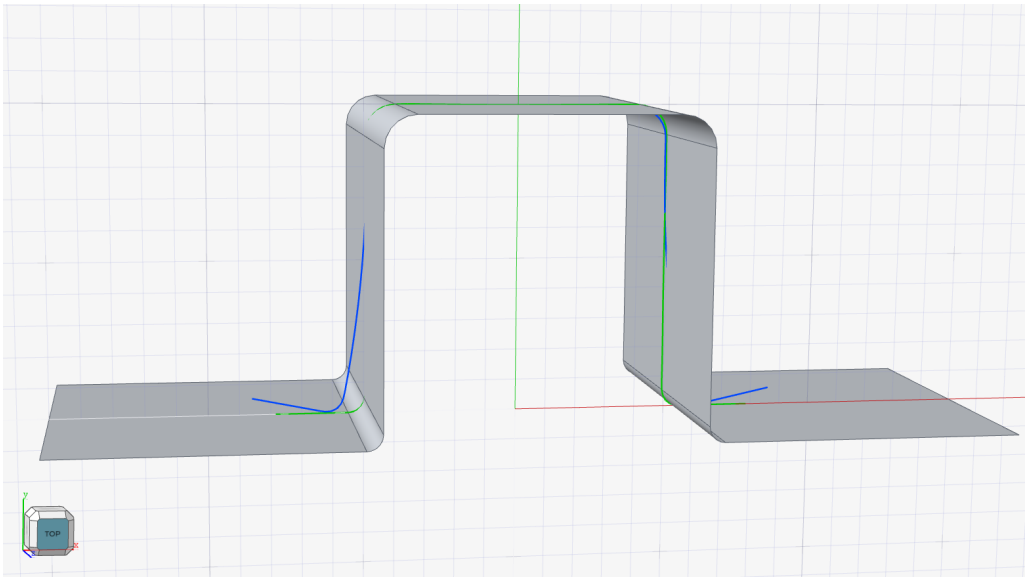

Figure 5.7: Main model used during the development of the plug-in.

In Figure 5.7 can be seen the main model provided: it is composed of the two curves, that are the same curves presented in Figure 5.3, and the surface representing the ideal final geometry of the metal sheet after the forming process. From a topological point of view, the surface is composed of nine different faces, of which five are planar and four are obtained through a blend function (different from the one developed in the plug-in). It is clear that Reference and Target curves are shorter than the section of the surface: this required an additional step in order to find the point on the section where the deformation should start and end. To do this, the reference system is based on the two curves first vertices: these, as presented in Chapter 5.2.1, are oriented in the same way so the starting point are on the same side. From this two points, a line is computed using the PK functions: a line is defined by a starting point, which in this case is the first vertex of the Reference Curve, and a direction, which is the vector obtained from the difference between the two starting vertices. Intersecting this line with the section curve of the surface, the starting and ending points can be easily found. From them, it is also possible to find the total parameter along the section, which allows to create a reference system to be used during the deformation function.

The objective for the main model is to perform the deformation only within these two parameter values, leaving undeformed the surface outside. It will also be

presented in Chapter 5.7.1 that the surface left outside will not only be undeformed, but might require other particular deformations.

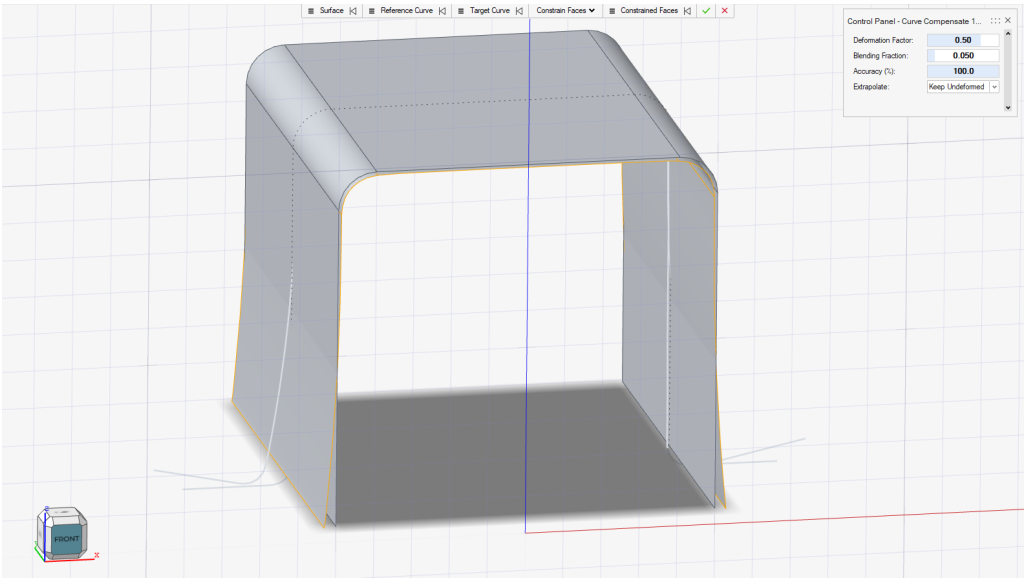

Figure 5.8: Trimmed surface model, here shown with the deformed surface computed (highlighted). The Reference and Target Curves are the same used before, even though they are not colored.

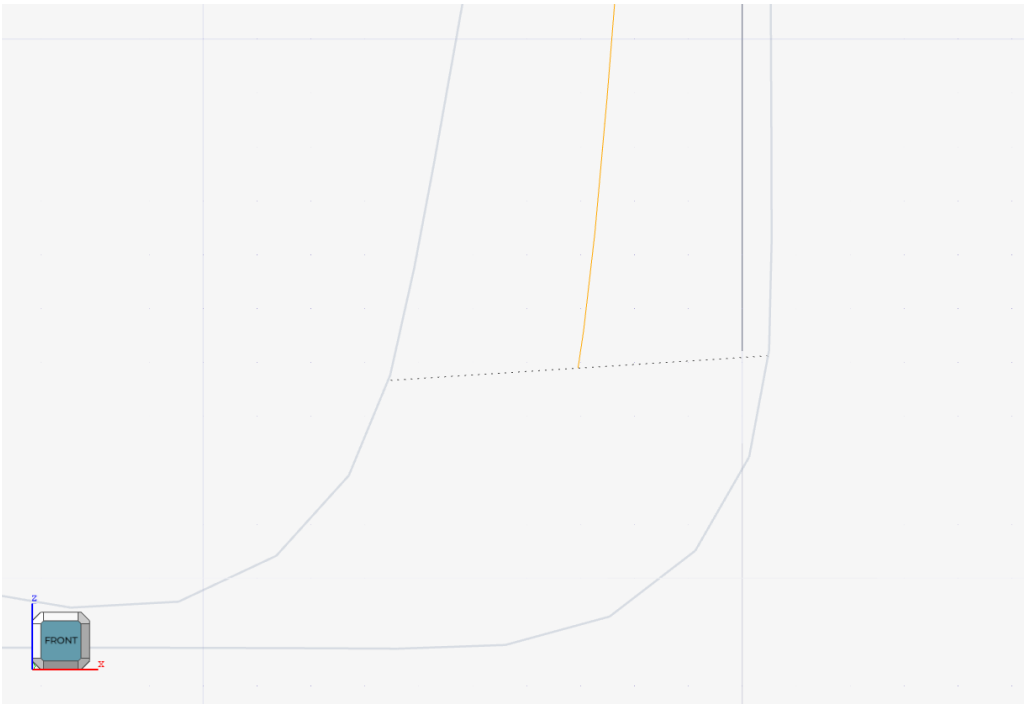

Figure 5.9: Focus on the trimmed surface model, with the focus on the side of the first vertex of the section. The dashed line is the segment between the two points on Reference and Target that have the same parameter.

Another interesting model is presented in Figure 5.8. In this model, the Reference and Target Curves are the same as the main model, but the surface has clearly been cut to exclude the first and last two faces. The requirements for this model are to perform the deformation not only considering the start and end of the Curves, but also the endings of the surface. This requires the definition of a different reference system, since the deformation is related only to the two curves and not the coordinates of the objects. So the first approach was to perform a binary search to find where the surface starts w.r.t. the parameter or Reference and Target Curves. This method was soon dropped, mainly due to the fact that the PK functions were failing to find the correct intersection with the section of the surface (this will further be analyzed in the Results, more precisely in Chapter 6.3.2). To prevent this, a simpler but still accurate enough approach was used.

Figure 5.9 reports the accuracy of the deformation starting point: the dashed line is supposed to intersect the first vertex of the surface, but it can be seen that it is slightly off. This happens because the starting parameter is computed starting from the projection of the first vertex onto the Reference Curve. This choice was based on the fact that the surface's section is supposed to be very similar to the Reference Curve, so the result should be accurate enough.

The last model considered in the development process is the one reported in Figure 5.4. This model is different from the others because both the surface and the curves are cut at both endings. In particular, in this case the Reference-Target Curves endings perfectly meet the endings of the surface. This means that no blending is required at the endings.

Moreover, this model was also used to test another implementation: if the Reference Curve is not selected, the plug-in should take as input the section of the surface. This means that the plane is computed with only the Target Curve.

5.7 Modalities

Here will be presented the different modalities built in the plug-in. In particular, two different settings were added for the user to pick: the first one is the ability to pick between the Constraint Faces mode, and the Constrain Region mode (explained in Chapter 5.7.2). The other option is related to the managing of the regions outside the Reference-Target area: these are called Extrapolate options, and are explained in detail in 5.7.1. The visual representation of these different options will be reported in the Results.

5.7.1 Extrapolate Options

The three options required by the plug-in were added in the second half of the development. The first one, *Undeformed*, was used as default before and it's the same approach as constraint faces. *Deform As Original* and *Deform As Tangent* were added to provide a more accurate representation of the metal sheet behaviour. Another thing to notice is that these options are used only when the surface is wider than the Reference-Curve area, or when the *Constrain Region* option is enabled. In fact, the extrapolation options can work both in *Constraint Faces* and *Constrain Region* mode, with different results.

5.7.1.1 Undeformed

This extrapolation option constrain the points before the starting point and after the ending point of the Reference-Curve area. Since they are constrained, the Deformation function does not modify their coordinates, and the final result is equal to the constrained faces. For this reason, the blending function is required at the start and at the end of the Reference-Curve area, to join the internal deformed surface with the undeformed regions outside.

5.7.1.2 Deform As Original

This option computes the position of the original surface point (along the section) on the start and end of the Reference-Target area. These position are then used together with the position of the first deformed points inside the area to compute a translation vector. This is then used in the Deformation function for all the points lying outside the area. The deformed region outside appears to be parallel to the original surface. As for *Undeformed*, a blending is required to match the different regions with tangent continuity.

5.7.1.3 Deform As Tangent

Deform As Tangent operates differently from the other options, as it creates a new structure to manage the external regions. In fact, when this option is enabled, the tangent of the deformed surface is computed at the start and end points of the Reference-Target area. from this, a line is computed where the final points will be laying, proportionally to their parameter along the section curve. This is the most useful option, as it represents the more accurate behaviour of the metal sheet in the external regions, which are usually not deformed. Moreover, for this option no

blending is required, as the external region is directly computed from the tangent and the continuity is granted.

5.7.2 Constrain Region

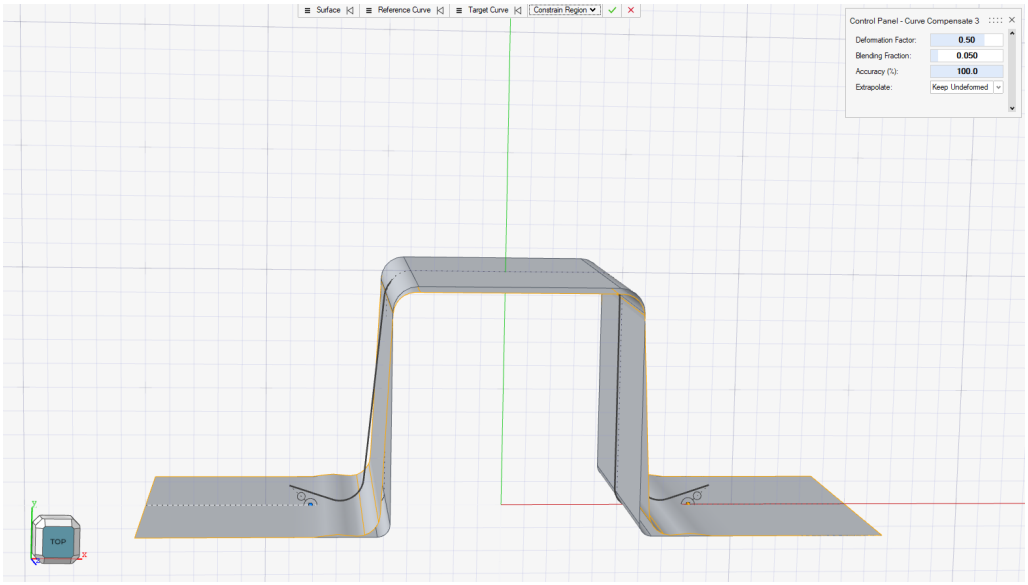

Figure 5.10: Visualization of the Constrain Region handles, at the default position (start and end of the Reference-Target area). The obtained deformed surface is highlighted.

This mode allows the user to pick the starting and ending points of the deformation. This is useful whenever the deformation is required in a smaller area than the total Reference-Target area. The picking of the points is provided through handles, which are interactive objects in the scene. As can be seen in Figure 5.10, the handles are made of three different elements: two blue points, one set to be on the section, while the other represents the deformed position of the other point. In addition, to make the points more visible to the user, they are both surrounded by a black circle. The third and final element is a dashed line is visualized between the Reference and Target points related to the handle's point. These handles are available at both ends of the deformation region.

These handles are constrained to move only along the section curve, and only within the Reference-Target area. This constraint required the development of an additional algorithm, which is related to the concept of view frustum presented in Chapter 3.1.3.1. Since the handles are moved with an interaction with the mouse, it is necessary to extract the ray passing through the mouse position on the screen, and relate it to the section curve. This is done finding the closest position of the

section w.r.t. the mouse, and updating the position of the handle when the mouse moves.

Regarding the functionality of this modality, it simply modifies the starting and ending parameters, to update the size of the deformation region. Outside this region, the remaining parts of the surface are managed exactly as before, depending on the extrapolate option selected.

6 Results

In this Chapter are reported the results obtained with the developed plug-in, applied to the models presented in Chapter 5.6. After that, some observations on the main issues encountered during the development are reported. All the results reported are performed with a positive Deformation factor, as from the default value of 0.5, with the exception of one representation with a negative value.

The deformed surface is the one highlighted, and can be confronted with the original one.

6.1 Deformation Function

In order to manage all these different modalities, options and parameters, the deformation function was constantly modified along the development of the plug-in. In fact, not only the deformation is now working between the Reference and Target Curve, but it's on an arbitrary plane computed from these curves. For this reason, the first thing that is done when the function passes a point on the original surface is to get all the necessary information about it, after translating it to the section plane. The main retrieved information are the parameter along the wire body and the index of the edge where the point is. From these two information, the point can be correctly managed into the different regions of the section, which

might be: the external area (where the point will be transformed according to the enabled Extrapolate option), the internal area (where the point will be transformed according to the normal deformation function) or a blending area (where the point will be transformed according to the computed blending curve). The final thing to manage is the presence of Constrain Faces: if the point lies on one of these faces, the coordinates are unchanged.

6.2 Obtained Deformations

Here are reported the visualizations of the deformations with different options and parameters. To be noticed that the highlighted surface (with yellow borders) is the deformed one, while the other is the original surface.

Figure 6.1 reports the deformation obtained on the main model when the Extrapolate option is set on *Undeformed*. It can be seen that the surfaces are the same outside the Reference-Target area, whereas a deformation takes place between the two curves. A blending region can be easily spotted on both sides of the region, where the curves end, from the slight gutter caused by the small Blending fraction.

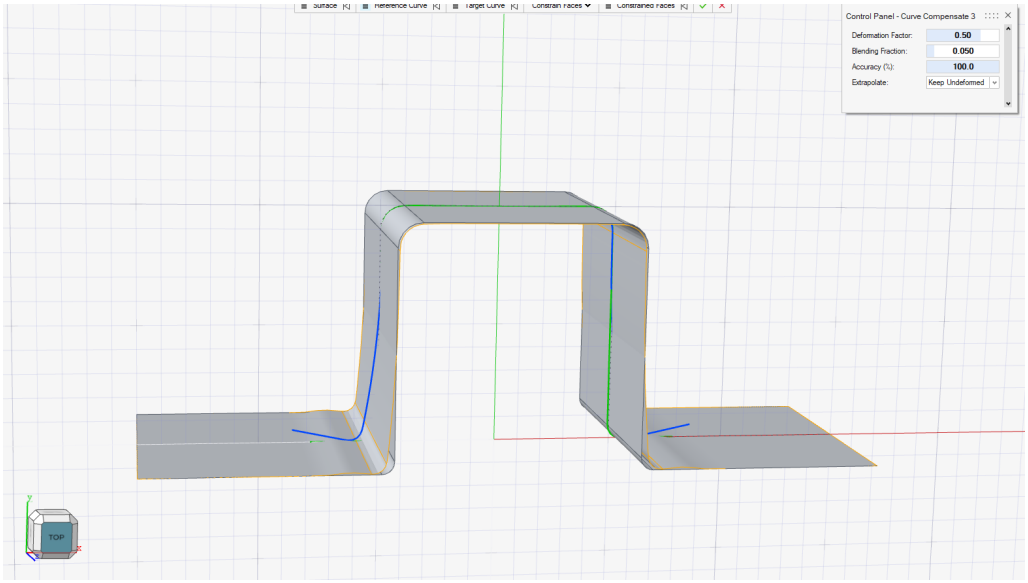

Figure 6.1: Model deformation performed with Constrain Faces mode (with no faces picked), and *Undeformed* set as Extrapolate option. For a focus on the blending, see Figure 5.5.

Figure 6.2 reports the deformation obtained on the main model when the Extrapolate option is set on *Deform As Original*. Here is also needed a blending between the internal and external area, which is still pretty smooth.

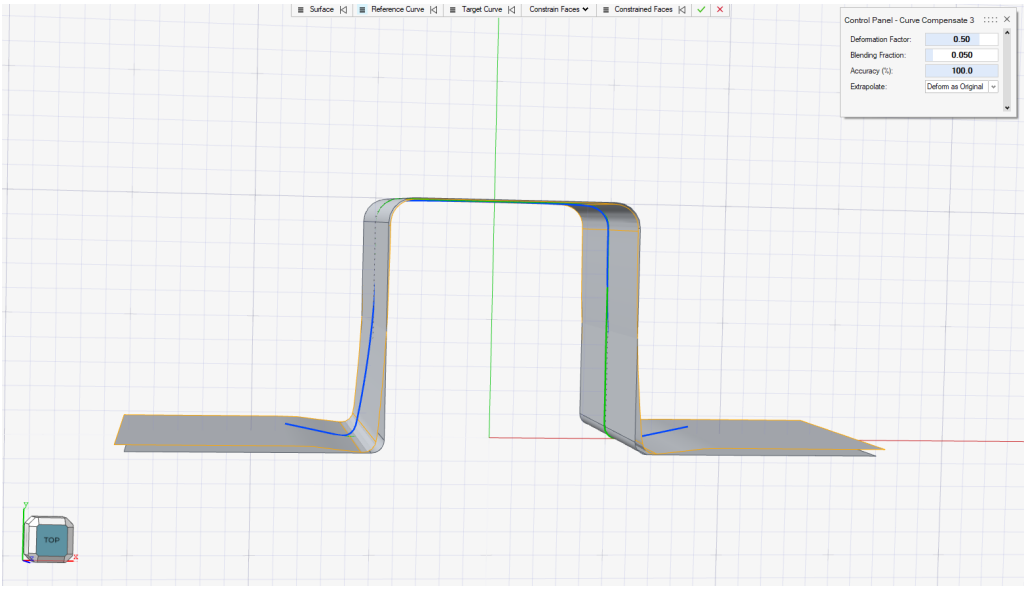

Figure 6.2: Model deformation performed with Constrain Faces mode (with no faces picked), and *Deform As Original* set as Extrapolate option.

Figure 6.3 reports the deformation obtained on the main model when the Extrapolate option is set on *Deform As Tangent*. Here the blending is not required.

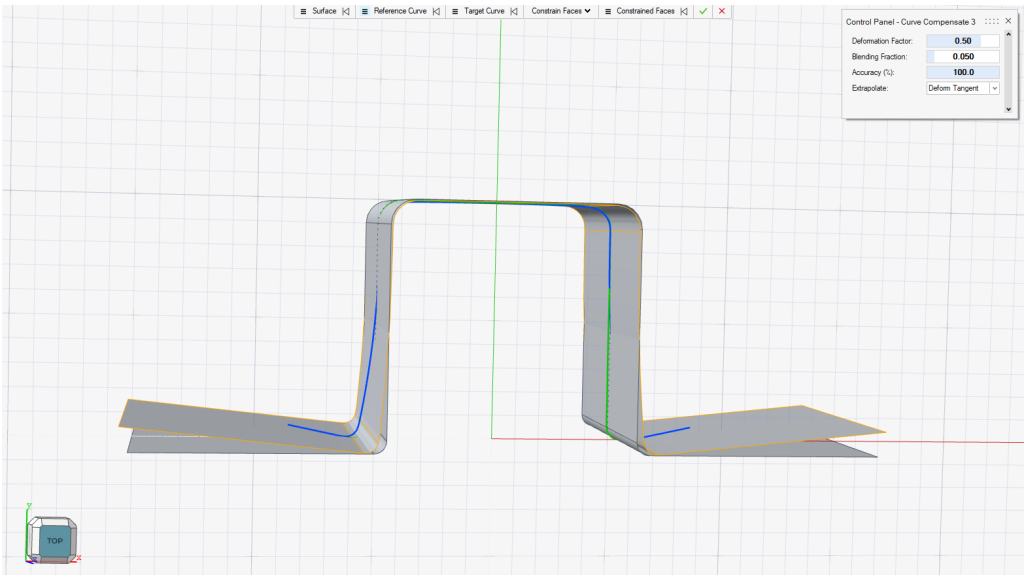

Figure 6.3: Model deformation performed with Constrain Faces mode (with no faces picked), and *Deform As Tangent* set as Extrapolate option.

The final and probably most important result is the deformation in Figure 6.4, where the Deformation factor is set to a negative value of - 0.5. To show this deformation, the Extrapolate option is set to *Deform As Tangent*. This might be the most important deformation setting because it shows the most accurate representation of

the metal sheet, with the supposed compensation for the springback.

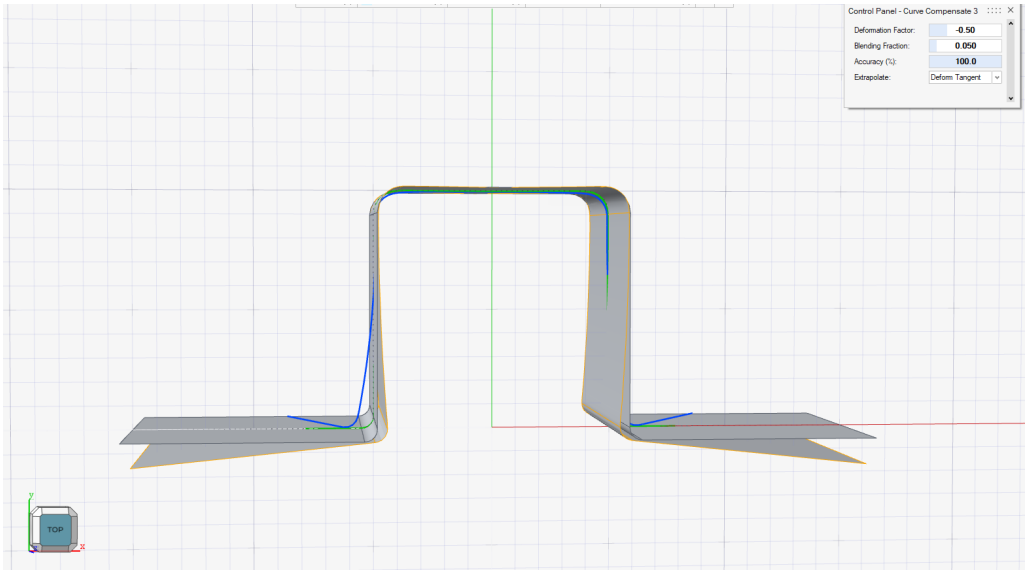

Figure 6.4: Model deformation performed with Constrain Faces mode (with no faces picked), and *Deform As Tangent* set as Extrapolate option. The Deformation Factor is set to a negative value.

6.3 Main Issues Encountered

6.3.1 Computational Burden

One of the main issues faced while developing the plug-in is the computational burden of the deformation function. Most of the computations are done in the pre-processing phase, but the high number of points to be processed on the surface remained the main cause of time loss. In addition to this, a particular attention must be given to the input curves. In fact, since these were obtained from a poly-mesh surface, when performing the boolean operation if intersection the resulting curve is composed of hundreds of small edges. More specifically, both Reference and Target curves appeared to be composed of 340 edges (Figure 6.5 reports the Reference Curve, with the visualization of every single vertex along the curve). Since the deformation required the computation of the parameter along the two curves, a for loop was implemented to scan the curves and find the points that corresponded to that parameter. This required a very long computational time, that was solved during the development with the creation of similar curves but composed only of a number of edges similar to the one of the surface's section. This was easily obtained, considering that the curve in many regions is straight and only two points

are necessary to define those regions (start and end of the segment).

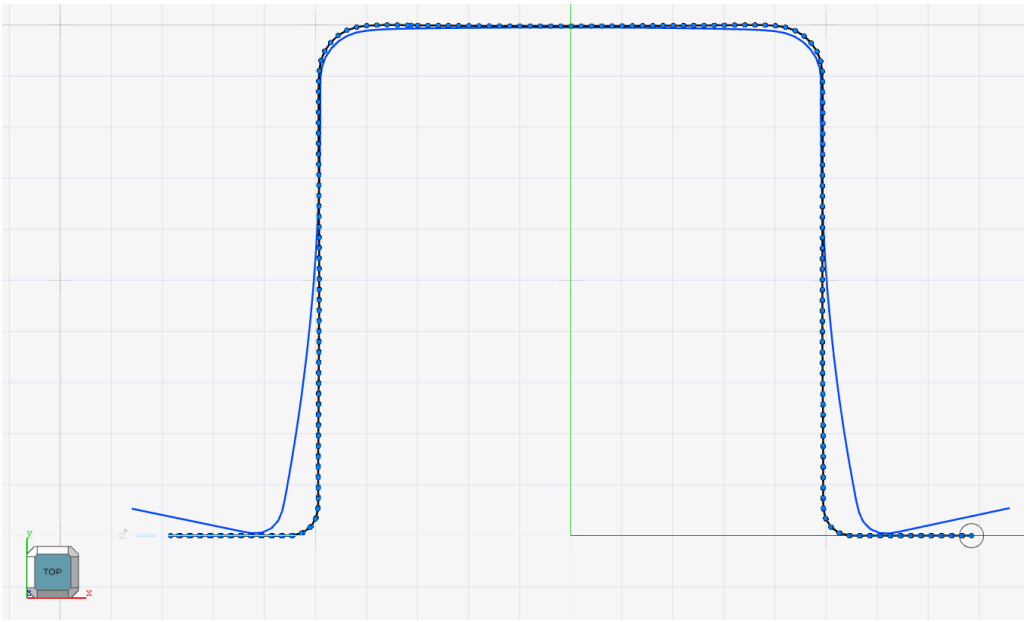

Figure 6.5: Visualization of the Reference Curve vertices (341).

6.3.2 Accuracy

In order to perform a faster deformation, an accuracy parameter was added. This parameter is modifiable by the user, and is reported in percentage. It performs a calculation regarding the tolerance used inside the Parasolid functions, which defines the precision of the results. In fact, a smaller level of accuracy performs faster but with slightly worse computations.

This parameter was required for the plug-in due to the computational burden, but the deformation fails whenever the accuracy is brought below 100%. This is probably due to the fact that a high level of precision is strictly required, using complex inputs with a high number of entities (edges). A test was performed to check this hypothesis, using Reference and Target Curves with a smaller number of edges. With these inputs, the deformation allows values of accuracy lower than 100%.

7

Conclusions & Future Works

This final Chapter provides a summary of the development steps and the final results w.r.t. the initial idea of the plug-in. After that, some missing implementations are reported that might be interesting to add to the project. After that, an introduction to the next steps of the development is presented.

7.1 Conclusions

The development of this plug-in was started to create a mathematical approach for springback compensation in metal sheet forming. The inputs given are produced by Inspire Form, a CAE software which defines how the metal is bent and deformed during the forming process. In particular, since this is the first approach to the problem, the plug-in is developed to deal with metal sheets with a constant section along one axis.

In the final form of the plug-in, the inputs are three: a Reference Curve, representing the ideal section of the final sheet, a Target Curve representing the section of the sheet deformed by the springback, and finally the third element is the surface, which is the final ideal form of the metal sheet. Initially all the three elements must be picked, but later the Reference Curve was made optional, and if missing the section of the original surface is used. The deformation is computed with a

Deformation factor, a parameter that set the position of the final surface w.r.t. the Reference and Target curves. Firstly it was bound in a [0.0,1.0] interval, which was later expanded to be [-1.0,1.0]. In the positive interval, it shows different levels of springback between the ideal surface and the total springback computed. In the negative values, the surface represents the compensated surface.

Different modes and models were used, to test different surfaces and cases. First, a management of the different models is needed: in fact, the Reference-Target Curves might be of a different size w.r.t. the section of the surface. For this reason, different kind of deformation were developed to decide how to deal with these different cases. For shorter sections, an algorithm was developed to find the correct starting and ending point of deformation between the two curves, in order to maintain the same size of the original surface. For longer sections, an additional choice is required to set how to manage the external surface. This option was called Extrapolate, and three options were given to the user:

- *Undeformed*, keeps the external surface identical to the original one, adding a blending region on the outer sides of the internal region;

- *Deform As Original*, maintain the original shape of the external surface w.r.t. the original one, adding a blending region on the outer sides of the internal region;

- *Deform As Tangent*, deform the external region in a straight line, corresponding to the direction of the outer sides of the internal region. No blending region is required.

In this cases, an important issue was to keep the tangency continuity along the surface (and so along the section), which brought to the decision of creating a blending region to join the different areas of the final surface. Another important requirement was the possibility to pick Constraint Faces, which would be treated similarly to the Undeformed regions. Also in this case, a blending is required to join the constrained face to the final surface.

The final implementation was added with the Constrain Region mode, where the user can manually pick the start and end points of deformation along the section of the original surface. This was managed with the implementation of a couple of handles, elements with which the user can interact.

7.2 Future Developments

Here are reported the following steps of the development for the plug-in, in order to achieve a complete and final result.

7.2.1 Multiple Reference-Target Curves

Since the main constraint for this project was the limit of a surface with a constant section, the following step would be considering multiple and different sections along the surface. This requires multiple Reference-Target couples, set along a dimension of the surface, to determine how the deformation should be computed. The most critical issue here is presented by the necessity of blending the surface between these different couples.

7.2.2 Shifting Vector Matrix

The final step of the plug-in would be creating the final surface starting from the original surface, and a mesh of translation vectors that are related to the points of the polymesh surface. This would allows to start from the raw information given by the springack compensation algorithm, avoiding dealing with other inputs than the surface. The only issue is given by the complex deformation algorithm to develop, which would deal with hundreds or more translation vectors. Moreover, for the points given that do not have a related translation vector, an interpolation would be required in order to understand how the surface deforms between two or more vectors. A linear interpolation cannot be applied, since it would cause tangency discontinuity along the surface, so a cubic or more advanced interpolation would be needed.

Bibliography

[1] R. L. M. B. T. Meinders, I. Burchitz, "Numerical product design: Springback prediction, compensation and optimization," *International Journal of Machine Tools and Manufacture*, vol. 48, no. 5, pp. 499–514, 2008.

[2] M. V. Tomáš Pacák, František Tatícek, "Compensation of springback in large sheet metal forming," Ph.D. dissertation, Czech Technical University in Prague, 2019.

[3] M. K. Agoston, *Computer Graphics and Geometric Modeling*. Springer.

[4] F. H. Foley, Van Dam, *Compuer Graphics: Principles and Practice*. Addison-Wesley Professional.

[5] Nurbs geometry overview. [Online]. Available: https://developer.rhino3d.com/guides/rhinopython/python-rhinoscriptsyntax-nurbs/

[6] Altair® inspire™ studio. [Online]. Available: https://www.altair.com/inspire-studio

[7] Parasolid. [Online]. Available: https://www.plm.automation.siemens.com/global/en/products/plm-components/parasolid.html