NORTHWESTERN UNIVERSITY

# Modeling and Control of the
# Double-Sided Incremental Forming Process

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

Huaqing Ren

EVANSTON, ILLINOIS

September 2018

## ABSTRACT

Modeling and Control of the Double-Sided Incremental Forming Process

Huaqing Ren

Stringent demands in the sheet forming industry related to rapid and customized part realization coupled with rigorous requirements on geometric accuracy and product properties have outpaced the capabilities of traditional forming processes. As an emerging novel technology, Double-Sided Incremental Forming (DSIF) offers much higher flexibility with the complete elimination of the need for geometry-specific forming dies, and significantly reduces the forming time and cost in sheet metal production. However, challenges in achieving tight geometric tolerances at a reasonably short forming time and in maintaining structural integrity in incrementally formed parts limit the wide industrial adoption of the technology, even after decades of intensive laboratory developments worldwide. Due to the strong nonlinear nature of the DSIF process caused by material plasticity, geometric variations, and varying local contact conditions between the workpiece and the forming tool(s), only a few models succeeded in determining process inputs (such as toolpaths) to achieve the desired part geometry. However, each model was applicable only to a specific process parameter combination or geometry. Moreover, as a flexible process, IF is often performed under a vast variety of potential process parameter combinations, such as different sheet metal materials, sheet thickness, tool diameters, lubricating conditions and clamping forces, which further signify the difficulty in establishing a single generalized model that encompasses all these parameter variations.

The central idea of the thesis, in response to the above-mentioned challenges and hurdles, is to explore and realize a control framework integrated with a mechanics-based model for the

purpose of enhancing geometric accuracy and suppressing possible fracture. To bridge the gap between laboratory research and industrial applications for DSIF, the control framework is designed with high control accuracy, sufficient generalization capability and easy pragmatic instrumentation setup in mind. The envisioned framework integrates three control loops at different levels to ensure accurate tool positioning, consistent maintenance of the desired tool contact pressure, and effective springback compensations for DSIF. Specifically:

- ❖ A real-time tool compliance compensation and a real-time contact force control algorithm are introduced to compensate for tool compliance and motion error, providing a stable contact pressure and avoiding any early loss of contact between the tool and the part. The implementation of a contact stiffness model increases the algorithm's generality and robustness for different machine setups and material choices. Both algorithms are directly superimposed over the conventional position servo control loop with minimal modifications and proven to be effective over a large range of process parameters with reasonable control accuracy ($< 10\ N$ deviation for a $300\ N$ command force). The purpose of this control loop is to ensure the realization of a true DSIF process, which has demonstrated its effectiveness in achieving tight geometric tolerance, delaying fracture, and increasing fatigue life compared to single point incremental forming.
- ❖ A feedforward toolpath optimization algorithm is established during the off-line planning stage to reduce the geometric error for a newly developed Accumulative Double-Sided Incremental Forming (ADSIF) toolpath strategy. Two process parameters that determine the relevant tool positions, i.e., position angle and gap, are proposed, for the first time, to capture the bending and squeezing phenomena in ADSIF. These process parameters are

then optimized with a response surface model as a function of the local geometric wall angle. A significant error reduction of 80% is achieved using the developed response surface model. The defined optimization framework can be implemented in ADSIF without the need for any feedback sensor as the starting process condition is well-defined and consistent throughout the process. Consequently, it should be noted that this strategy for geometric control is suitable for ADSIF only. A more general strategy for DSIF is to be shown next.

- An on-line springback compensation algorithm is established, which dynamically modifies the toolpath to compensate for potential springback based on an accelerated FEM-based springback prediction model. The springback prediction model is a first-of-a-kind real-time model that links the target variable (final part geometry) to measurable process variables (forming forces). Moreover, since the FEM model can be easily adopted to different sheet materials, part geometries and boundary conditions, the predictive model can thus be easily generalized to all these parameter changes. A customized unsupervised classification algorithm is also proposed, for the first time, to extract the key control points for a given complex geometry, so that springback prediction only needs to be conducted at a limited number of selected locations. The experimental results demonstrate that springback error can be limited within the 1.5 *mm* range for the tested geometries, compared to more than a 5 *mm* geometric error for the uncontrolled parts.

To summarize, this work directly combines the control and modeling efforts to develop a systematic control framework for DSIF that improves part accuracy/contact conditions with three control loops operating at different time scales. Specifically, the introduction of feedback control into the DSIF process permits a more error-/disturbance-tolerant model, while the establishment of the process mechanics model helps to determine the proper control strategy and increases control robustness. Moreover, the framework, built on a solid foundation of process mechanics, is designed with a low implementation cost, a non-machine-intrusive setup, and a high model generality, thus guarantying a great potential of the work to be scaled up from laboratory prototypes to industrial applications. In doing so, this work successfully improves various aspects of DSIF, and paves the way for continued commercial implementation of the DSIF technology in the near future.

## ACKNOWLEDGEMENTS

I am extremely lucky to be surrounded by so many excellent people who have helped and supported me during my Ph.D. study.

I would first like to thank my advisor, Professor Jian Cao, for her continued support and guidance throughout the entirety of my graduate research. I am always inspired by her commitment to research and teaching.  I am equally thankful to Professor Kornel Ehmann, who co-advised my Ph.D. and helped me develop independent and critical thinking towards research. Throughout the years, both professors have been a tremendous source of inspiration, which motivates me to address challenging scientific problems as well as develop impactful research projects.

I would like to thank my dissertation committee members, Professor Masayoshi Tomizuka and Dr. Danielle Zeng for their truly indispensable advice, inspiration, and support throughout my journey. I would also like to thank my many other collaborators who have provided invaluable guidance, support, and discussion. Especially, I have received tremendous support and encouragement from my peers and the Advanced Manufacturing Processes Laboratory at Northwestern University: Dr. Rajiv Mahlotra, Rui Xu, Dr. Ping Guo, Dr. Peidong Han, Dr. Qiang Zeng, Marco Giovannini, Dr. Zixuan Zhang, Newell Moser, Dr. Ebot Ndip-Agbor, Mojtaba Mozaffar, Sarah Wolff, Weizhao Zhang and so on.

Last but not least, my warmest feelings and deepest thanks to my loving parents, my mom Qingyun Zhang and my dad Shiyang Ren, who have always wholeheartedly supported me during my graduate studies, and to my friend for life, my soulmate, and my wife, Xiaoman Zhang for her constant source of support and encouragement. I feel blessed to have them by my side.

This work is mainly supported by the U.S. Department of Energy under award DE-EE0005764 and Energy Efficiency & Renewable Energy program (EERE).

## DEDICATION

I dedicate this dissertation to my parents, Shiyang Ren and Qingyun Zhang, and my beloved wife, Xiaoman Zhang. Without their love and support, none of this would have been possible.

# Table of Content

# List of Figures

# List of Tables

# Chapter 1 Introduction and Background

Sheet metal forming processes have been widely utilized in various industries, such as the automotive, aerospace, medical and beverage container industries among others. In a traditional sheet metal stamping process, a set of die and punch deform an initially flat sheet metal into a final three-dimensional complex shape under a large pressure generated by the hydraulic press, as shown in Figure 1.1. The typical forming process usually takes less than 10 *s* to form one single part, and thus this process is highly efficient for high-volume production.

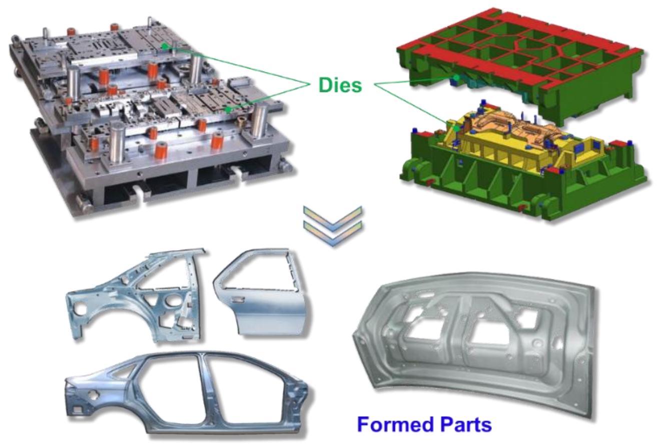


**Figure 1.1 Conventional forming process**

Traditional sheet metal stamping processes become highly burdensome regarding time, energy and cost when the production volume is lower than 5,000 (Ingarao et al., 2012). The heavy and massive die sets have to be engineered, cast, machined and then tried out several times for a given part and geometry. On the other hand, the recently growing demands for flexible, energy efficient and mobile manufacturing require flexible sheet metal forming technologies which are economically feasible for small lot production and prototyping, like decorative part manufacturing, medical implants, or architectural prototypes as shown in Figure 1.2. In the sequel, a brief description of some of the developed flexible sheet metal forming processes is first provided in Section 1.1. As one of the most efficient and effective solutions, Incremental Forming (IF) technology, especially Double-Sided Incremental Forming (DSIF), is also introduced in details. This is followed by a further discussion on the major challenges in DSIF in Section 1.2. Current solutions with their specific limitations are also reviewed in Section 1.2. Finally, Section 1.3 describes the primary objective and provides an overview of this work.

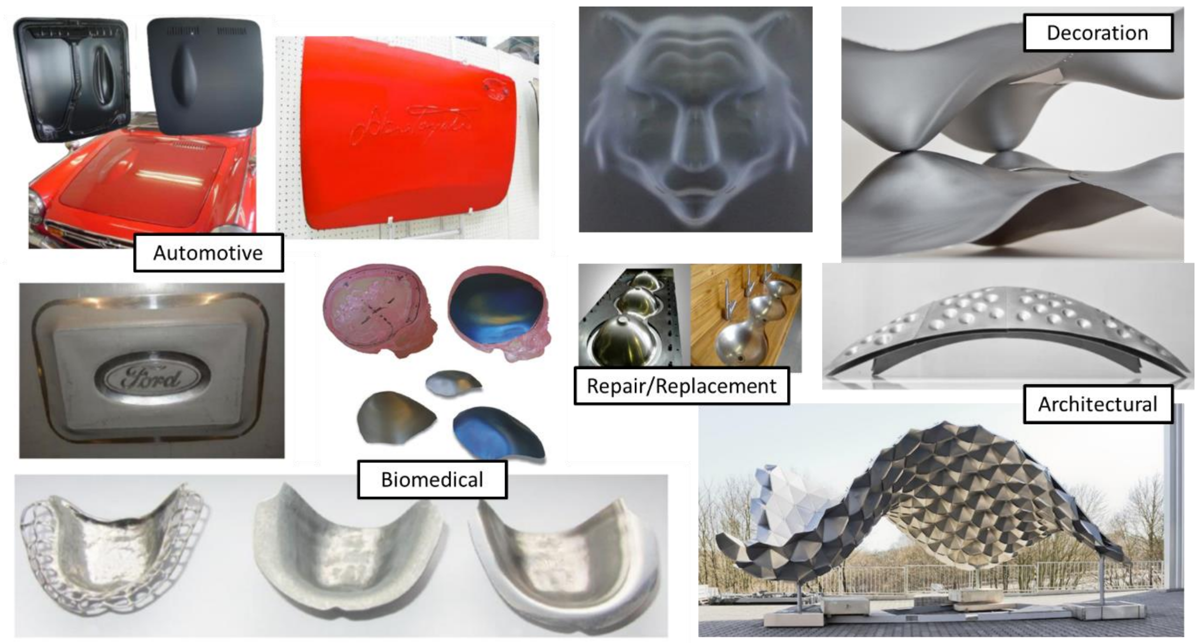


**Figure 1.2 The potential applications of flexible sheet manufacturing process (Bailly et al., 2015; Castelan et al., 2014; Emmens et al., 2010; Kalo and Newsum, 2014; Milutinović et al., 2014; Reynolds, 2015)**

## 1.1. Introduction to Incremental Forming

Incremental forming, as one of most well-developed flexible sheet metal forming processes, has drawn extensive attention during the last few years. To thoroughly understand the development of the incremental forming process, different forms of flexible sheet metal forming processes are first summarized. The invention of the incremental forming process and the different major variants of this technology are then introduced in Sections 1.1.2 to 1.1.4. Finally, the double-sided incremental forming process is introduced in Section 1.1.5 with particular emphasis on its fundamental mechanics differences with other incremental forming technologies along with its corresponding advantages.

### *1.1.1. Introduction to Flexible Sheet Metal Forming*

As the name implies, flexible sheet metal forming encompasses forming processes which do not depend on the conventional punch/die arrangement and thus shorten the production period and make the manufacturing process more "flexible." The root of flexible forming can be traced back to the original blacksmiths, who could form products ranging from simple things like nails or chains to the most complex of weapons and armor via a simple hammer and an anvil. Nowadays, more flexible sheet metal forming processes have been developed, which can be placed into four major categories as suggested in Allwood's work (Allwood et al., 2009):

- **Reconfigurable tools** in which the tool geometry is modified depending on the different target geometries. The tool geometry is adjusted to coincide with the final designed geometry if the part is formed in one stroke or the intermediate geometries in each step if the part is manufactured in multiple subsequent steps. Note that process operation is the same for all products, i.e., pressing the tools with certain displacement or pressure under limited process control. Examples include tools made from clamped strands (Finckenstein and Kleiner, 1991), modular tools (Nielsen et al., 1997) or multiple pin forming (Li et al., 1999; Liu et al., 2008; Walczyk and Hardt, 1994, 1998) as shown in Figure 1.3. The primary limitation of this process is that it usually still requires a large machine to perform the process while surface or part quality are usually compromised compared with the conventional forming methods.

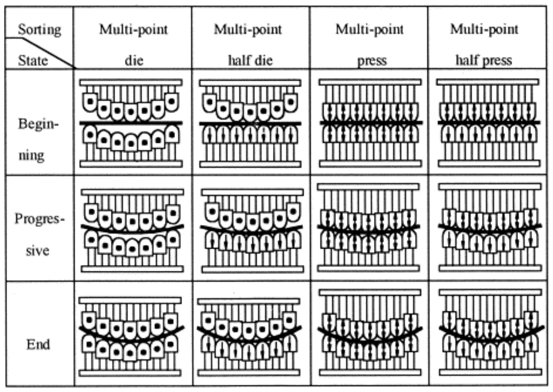


**Figure 1.3 Four methods in multiple pin forming (Li et al., 1999)**

❖ **Enhanced-Actuation** in conventional forming occurs when the conventional sheet forming process is equipped with actuation mechanisms in conjunction with the primary forming motion to improve process controllability. Examples include the utilization of clamping-force-controllable segmented blank holders in the stamping process (Lim et al., 2009), or a laser-assisted mental spinning (Klocke and Brummer, 2014) as shown in Figure 1.4. Note that due to the limitations of original sheet metal forming processes like deep drawing and spinning, the flexibility achieved in fact is insufficient for manufacturing arbitrary complex shapes.

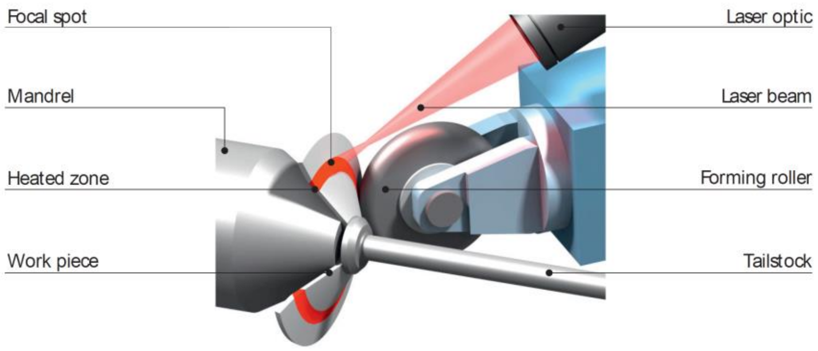


**Figure 1.4 Process principle of laser-assisted multi-pass metal spinning**

- **Stage-wise processes** in which a product passes through an array of adjustable tools in one-direction. Examples include water spray control in strip rolling (Duncan et al., 1998), roll forming lines  (Groche et al., 2008) and the use of a line array roll set in plate rolling (Cai et al., 2012). However, the process's flexibility is constrained by the number of manufacturing stages. With a finite number of stages, the geometry must be decomposable into a limited number of intermediate shapes. Most freeform geometries do not necessarily meet this requirement.
- **Mobile-tool processes** in which one or multiple small tools travel along a predefined trajectory, usually defined by a toolpath, over a workpiece, incrementally building up the desired shape. For example, incremental sheet forming with its many variants, which will be introduced in later sections, spinning (with or without a mandrel), laser forming (Li and Yao, 2000; Liu et al., 2004), and hand craft processes such as the English wheel all belong to this category. Compared to conventional forming process, the tool-sheet contact area is significantly reduced or eliminated due to the decreased size of the tool or the use of a laser, which leads to lower manufacturing loads, and naturally, smaller motion actuators for the forming machine.  Thus, the machine size is reduced, and the setup cost is lowered. On the other hand, the toolpath can be viewed as a series of defined deformations with a nearly infinite number of DOF, and thus, in principle, arbitrary geometric shapes can be manufactured if the material's maximum forming angle is not considered.

Taking the English wheel as the simplest example, a craftsperson forms compound (double curvature) curves from a flat metal sheet by merely moving the sheet in a certain pattern. The English wheel machine is usually constructed in a "C" shape, as shown in Figure 1.5. At the ends of the C, there are two wheels, with the upper one called the rolling wheel and the lower one the anvil wheel. The craftsperson passes the sheet metal between the two wheels, stretches/bends the material and causes it to form a convex surface over the anvil wheel. Moreover, with these multiple sheet passes, the English wheel can flexibly produce different panels using the same machine and has been widely used for low volume production of compound curves in coachbuilding, car restoration, and motorcycle components.

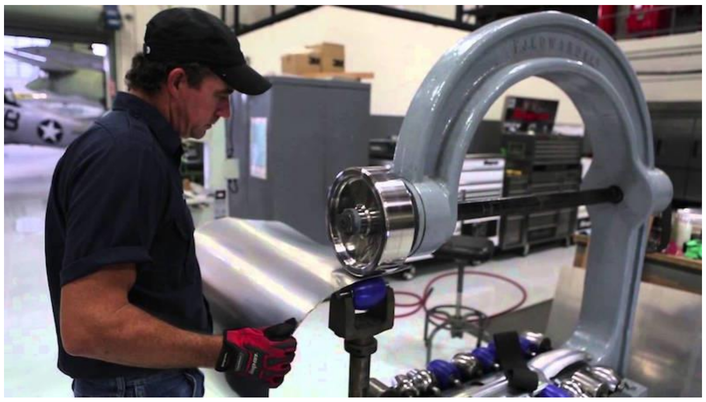


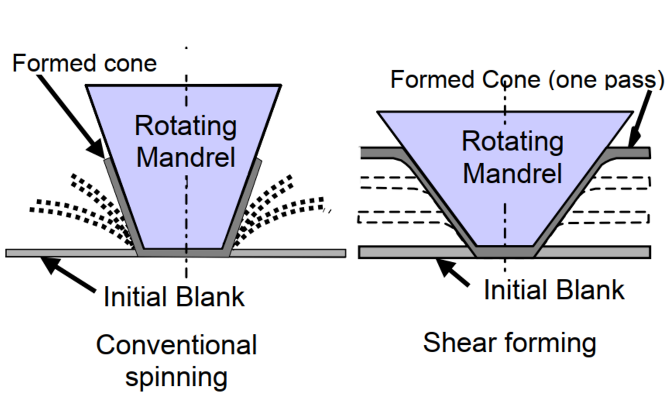
**Figure 1.5 The handling of the English wheel (Stew, 2017)**

### *1.1.2 Spinning, the Early Forerunner of Incremental Forming*

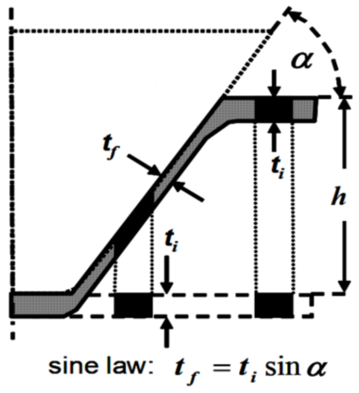
Although the English wheel can produce a complex shape by a simple machine with a low capital set up cost and short development time, the process is still manually operated, requiring high human operating skills and cannot guarantee process controllability/repeatability.

If one were to seek another “mobile-tool” type flexible sheet metal forming process with higher repeatability, spinning would be a proper example. In spinning, one or multiple clamped spinning tools approach the clamped sheet metal and deform it into the required shape. In conventional spinning the blank edge is moving inwards, and the material thickness is kept more or less constant. On the other hand, the blank edge is not moving inwards in shear spinning, and the sheet thickness is being considerably reduced (Wong et al., 2003). Note that in spinning, the tool movement is usually controlled by a servo motor instead manually, which ensures process repeatability. Moreover, a mandrel is usually utilized as a full or partial die to support the material and enforce the correct geometric shape.

One specific sheet thickness reduction principle, called the sine law, is highlighted here specifically for shear spinning, as it is also one of the fundamental principles widely adopted in the incremental forming field as introduced later. As the sine law states, the final thickness of a shear spun part can be calculated as $t_f = t_i \sin(\pi/2-\alpha)$ (Reagan and Smith, 1991), where $t_i$ is the initial thickness, $t_f$ is the final thickness, and $\alpha$ is the wall angle of the formed part, as shown in Figure 1.6(b). Note that the sine law is established based on the assumption that there is a minimal deformation of material outside the forming region, i.e., the center and the periphery of the cone part in Figure 1.6(b).



(a)



(b)

**Figure 1.6 (a) Illustration of the conventional spinning and shear spinning (Wong et al., 2003) (b) illustration of the sine law (Reagan and Smith, 1991)**

### *1.1.3 Single Point Incremental Forming*

The development of incremental sheet forming was closely related to shear spinning in its early days. Since1967, patents have been issued on variants of spinning that can be regarded as the IF process, or at least very close to that.

For example, Leszek's patent (Edward, 1967) first described a process for the manufacturing of axisymmetric sheet metal parts. The metal sheet was clamped on a rotating base and was bent and deformed by the vertical displacement of a roller, as shown in Figure 1.7. The process can be thought of as shear spinning without the requirement for a mandrel, or in another words, an initial form of axisymmetric single point incremental forming.

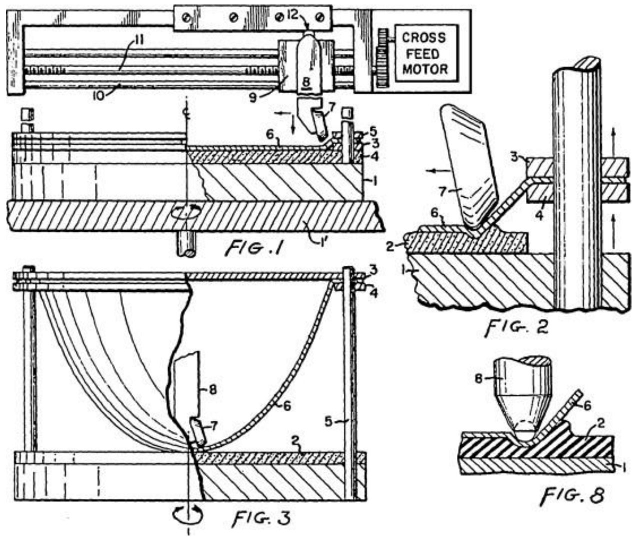


**Figure 1.7 The early axisymmetric incremental forming process (Edward, 1967)**

The first asymmetric single point incremental forming process was established around the late 1980s in Japan by Iseki as shown in Figure 1.8 (Iseki et al., 1989). Inspired by the concept of the tool-path of a three-dimensional CNC milling machine, the process successfully demonstrated the first incrementally formed non-symmetrical parts. The initial setup was straightforward and made use of a manually operated X–Y table, and then the setup was improved via a three-dimensional computer-controlled XYZ stage (Iseki, 1996), which allowed better controllability and repeatability. The process successfully incrementally formed steel, stainless steel, and titanium sheet metals of 0.7 mm thickness in various shapes.

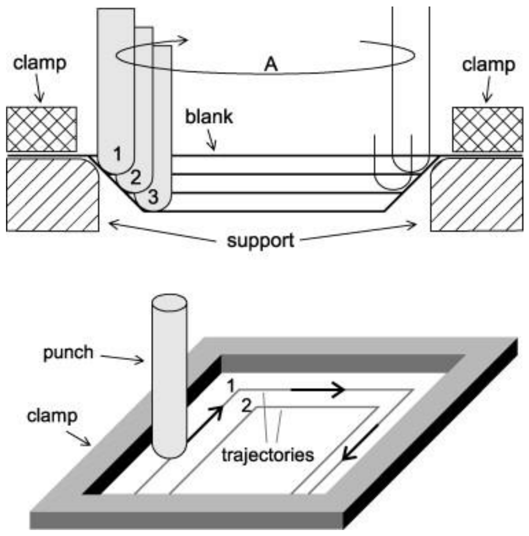


**Figure 1.8 Initial realization of SPIF by Iseki (Iseki et al., 1989)**

Iseki’s work offers a precise definition of Single Point Incremental Forming (SPIF): a single tool moves along a predefined toolpath to incrementally deform the sheet metal. The final geometry is imposed by the accumulation of sequential, localized plastic deformations. In SPIF, the avoidance of a dedicated, rigid traditional stamping system reduces the forming time and cost, especially in the production of small batches of highly differentiated parts, like in decorative part manufacturing, medical implants, or architectural prototypes. The specific advantages and disadvantages of SPIF are well summarized in the review work by Jeswiet in Table 1.1 (Jeswiet et al., 2005):

**Table 1.1 Summary of key advantages and disadvantages of SPIF**

| Advantages | Disadvantages |
|---|---|
| Generic Tooling, No Die/Punch Required | Slower Forming Speed |
| Flexible for Various Part Shape/Size | Limited Forming Angle via Single Pass |
| Suitable for Different Materials (Steel, Aluminum, Titanium, Plastics) | Low Geometric Accuracy due to Springback Error |
| Increased Formability | |
| Increased Surface Finish | |
| Small Forming Force | |
| Easy Adoption to the CNC Machine | |
| Noise Free Process | |

Note that in SPIF, the usage of only one forming tool enables the process to be performed on a typical three-axis CNC machining center. However, during the forming process, the region of the sheet metal between the tool and the clamping fixture is usually not supported. This leads to undesirable global bending that decreases the overall geometric accuracy of the part as shown in Figure 1.9 and Table 1.1 (Fiorentino et al., 2015; Malhotra et al., 2011; Micari et al., 2007).

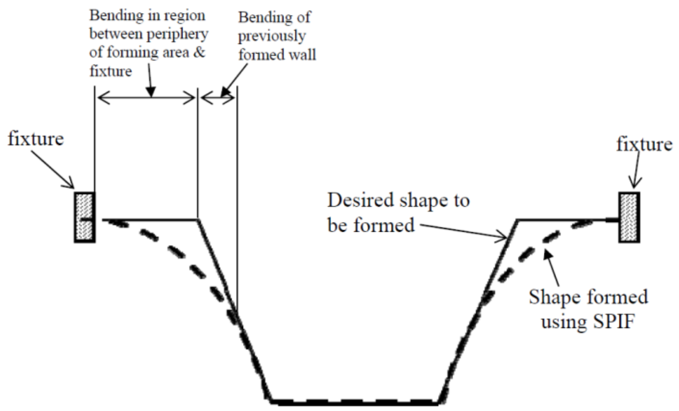

Bending in region between periphery of forming area & fixture
Bending of previously formed wall
fixture
fixture
Desired shape to be formed
Shape formed using SPIF

**Figure 1.9 Schematic showing the geometric inaccuracy in SPIF (Malhotra et al., 2011)**

### *1.1.4 Two-point Incremental Forming*

To overcome the lack of support and geometric accuracy in SPIF, Two Point Incremental Forming was suggested (TPIF). TPIF was first introduced by Powell in Europe (Powell and Andrew, 1992) and by Matsubara in Japan (Matsubara et al., 1996) as an augmented version of SPIF. In the TPIF process, a backing die is used to support the sheet, either as a full die or a partial die (a single cylinder post for example). Specifically, in TPIF, the blank holder usually moves vertically on bearings, which move the metal sheet, along the Z axis, as the forming tool simultaneously pushes into the sheet metal. As seen in Figure 1.10, this process has two points where the sheet metal is pressed, one in the upper tool and the other in the die region, hence it is called Two Point Incremental Forming. Note that the added support tool helps in supporting the undeformed region, improves the stiffness of the formed part, and thus further localizes the deformation and reduces possible geometric errors (Attanasio et al., 2008; Tekkaya et al., 2007).

However, the utilization of the die makes the TPIF not genuinely dieless and compromises the flexibility of the process.

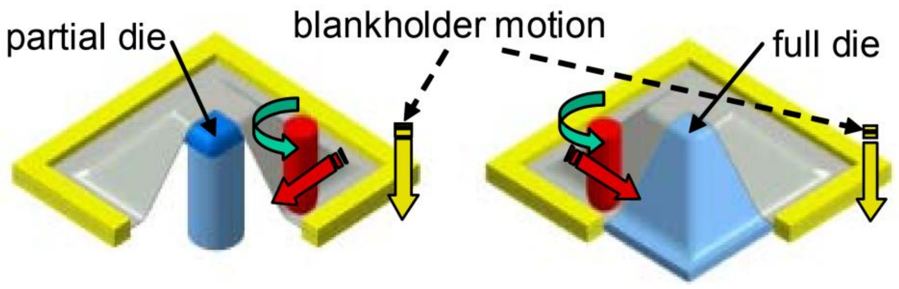


**Figure 1.10 TPIF schematics with a partial and full die (Jeswiet et al., 2005)**

### *1.1.5 Other Forms of Incremental Forming*

Before diving into Double-Sided Incremental Forming, other forms of incremental forming are briefly presented, so that a full bird-eye's view can be offered of the incremental forming field. Instead of the rigid stylus-like tool, researchers have tried to utilize different deformation media, such as laser forming, shot peening forming or water jet forming, to achieve similar localized plastic deformation.

In traditional shot peening, the bulk products are repeatedly hit by a large number of small balls, whose size typically varies from 0.125 *mm* to 5 *mm* in diameter, to create compressive surface stresses on the part. The balls are usually made of as cast steel, glass, or ceramic, considering a balance between material hardness and cost. Work by Kopp (Kopp and Schulz, 2002) has demonstrated the possibility of creating both convex and concave shapes by simultaneously using double sided shot peening.

Another new emerging alternative for the SPIF tool is high-pressure water jet based incremental forming. Water jet cutting has been widely utilized in sheet metal (or plastic and wood) cutting since the early 1980's. Researchers such as Iseki (Iseki, 2001) and Jurisevic (Jurisevic, 2003) explored the possibility of utilizing the standard waterjet cutting system for incremental forming on 0.5 *mm* thick AA6082 sheets. Successful experiments were conducted with water pressures of up to 50 *MPa*. Recent work by Shi (Shi et al., 2018) and Li (Li et al., 2014) demonstrates the possibility to form steel sheet with pressures up to the 120 *MPa*.

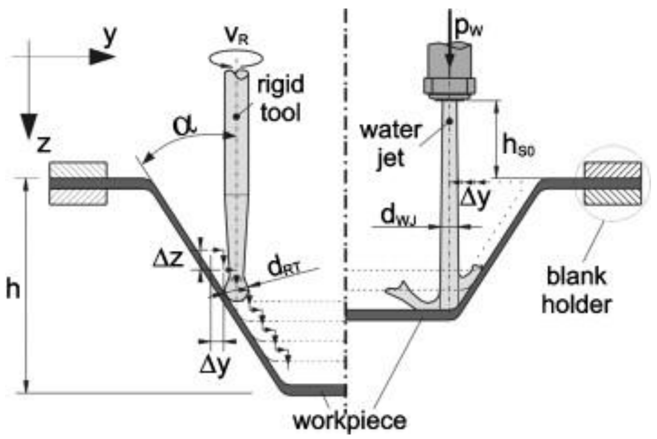


**Figure 1.11 Comparison of work the principles of SPIF (Left) and Waterjet Incremental Forming (Right) (Petek et al., 2009)**

Note that shot peening and water jet forming demonstrated a good potential to substitute the original incremental forming process with some additional benefits, like enhanced surface quality and fatigue behavior. However, the involvement of a "flexible material," i.e., either balls in the shot pinning IF or high-pressure water stream in the waterjet IF, as a forming tool, significantly builds up process complexity, and thus requires a broad research effort to investigate their fundamental mechanisms and limitations.

### *1.1.6 Double-Sided Incremental Forming*

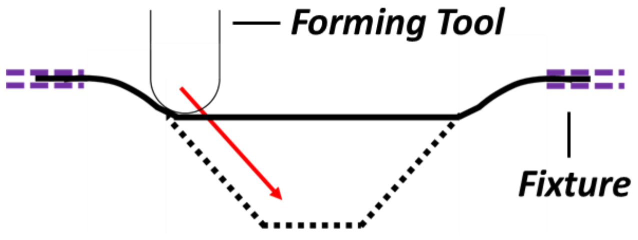
To maintain the benefits of localized deformation observed in TPIF while simultaneously avoiding the overhead costs associated with a die, Double-Sided Incremental Forming (DSIF) has been proposed(Meier et al., 2007; Wang et al., 2008). In DSIF a second tool is utilized on the other side of the metal sheet, as shown in Figure 1.12(c) that takes on the role of the backing die in TPIF.

Essentially, the DSIF process is still an evolving form of the basic SPIF, in which only one tool forms the sheet metal. The tool, called the "forming tool" in the following context, moves along a pre-defined toolpath that follows the desired surface of the CAD design. The material at the contact points on the metal sheet is bent and stretched to the target geometry. The primary role of the forming tool is to confine the deformed positions of the material, i.e., to deform the sheet material into the desired part shape. Note that the forming tool is defined here as the tool on the inner/concave side of the part feature in the normal direction to the metal sheet.

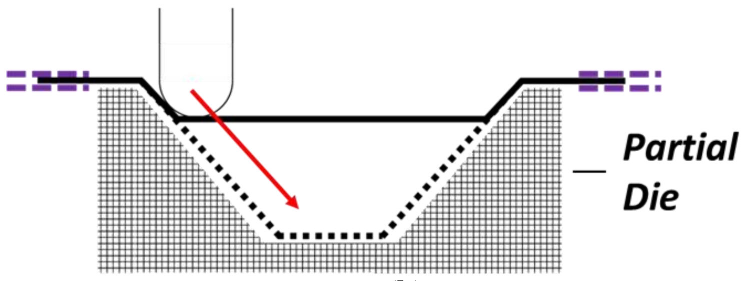
As mentioned before, the major drawback of SPIF lies in its reduced geometric accuracy caused by the unwanted global bending since only one tool is being used. To solve this problem, a second tool, usually called the "supporting tool," is utilized in the DSIF process which offers enhanced process control. From a geometric point of view, the supporting tool is defined as the tool located outside, convex side, of the part's geometric feature being formed. Compared to the forming tool which directly determines the global geometry, the supporting tool has two major effects:

❖ *Supporting Effect*: In the local contact area around the forming tool, the supporting tool acts as a local die, supporting the region of the sheet between the forming tooltip and the fixture.

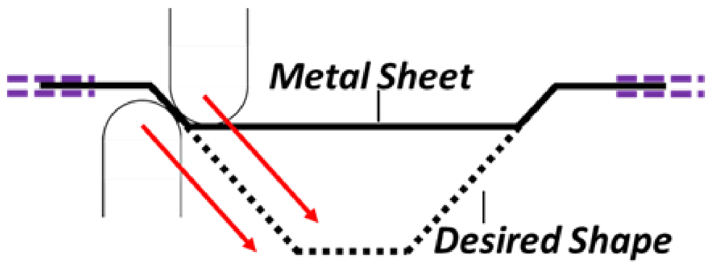
This causes the deformation to be concentrated in a local plastic region of the sheet. This stabilization of the deformation into a local region around the tool results in the deformation being much closer to the ideal desired amount than in SPIF.



(a)



(b)



(c)

**Figure 1.12 Configurations of different IF processes: (a) SPIF, (b) TPIF, (c) DSIF**

❖ *Squeezing Effect*: The squeezing of the sheet between the two tools causes the deformation to be concentrated in a small region around the tool contact area. This causes higher plastic strain and strain hardening in DSIF as compared to SPIF. The super-imposed contact pressure due to the existence of the supporting tool alters the local material stress triaxiality and enhances process formability.

Note that depending on different geometric part features, the roles of the forming and supporting tools might be exchanged even in a single part, as shown in the example in Figure 1.13, which has features on both sides of the metal sheet, i.e., convex and concave features. For the feature pointing downward, the top tool will take the role of the forming tool and the bottom tool of the supporting tool, while for the feature that points upward, the opposite applies.

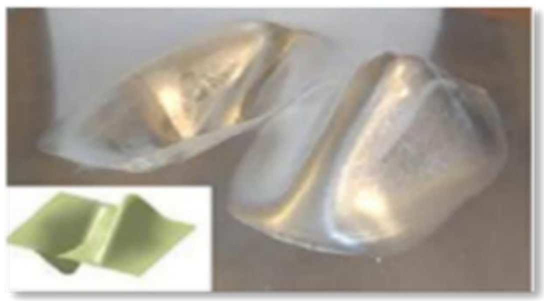


**Figure 1.13 An example part with features on both sides of the metal sheet formed by DSIF (Malhotra et al., 2012)**

## 1.2 Challenges in Double-Sided Incremental Forming

Jeswiet et al. (Jeswiet et al., 2005) report that, while most industrial parts require an accuracy of ±0.5 mm, it is observed that parts produced by IF have significantly higher dimensional inaccuracies. Although DSIF performs better than SPIF in geometric control and formability, the inaccuracy of the DSIF process remains a primary obstacle to its wider industrial adoption. The major causes of geometric errors in DSIF are related to: (1) position errors due to machine compliance or motion inaccuracy, (2) in-process springback due to tool movement or tool unloading, and (3) post process springback during the unclamping or the trimming process, i.e.:

$$\delta = \delta^1 + \delta^2 + \delta^3 \tag{1-1}$$

where $\delta$ is the total error of the final DSIFed geometric part and $\delta^1$, $\delta^2$, and $\delta^3$ are the tool position, in-process springback, and post-process springback errors.

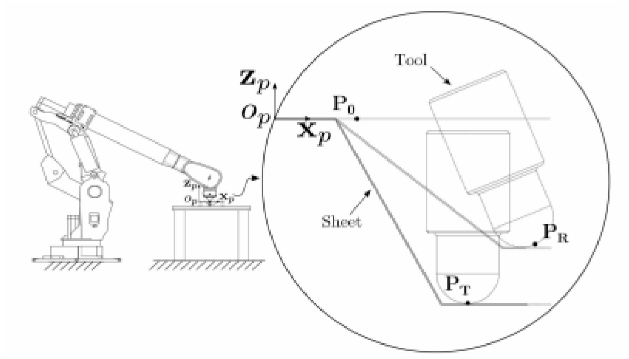
In this section, these three major geometric errors in DSIF (or SPIF) are briefly introduced as the motivation for the proposed works in Sections 1.2.1 to 1.2.3. Recent research developments to address these challenges are also presented alongside their limitations, which are mostly related to the low accuracy, inadequate generality or high capital/lead time cost. Last but not least, since the DSIF process (similar to SPIF or TPIF) is also limited in the maximum limited forming angle, i.e., material formability, the research efforts in this field will also be presented in Section 1.2.4.

### *1.2.1 Tool Position Error*

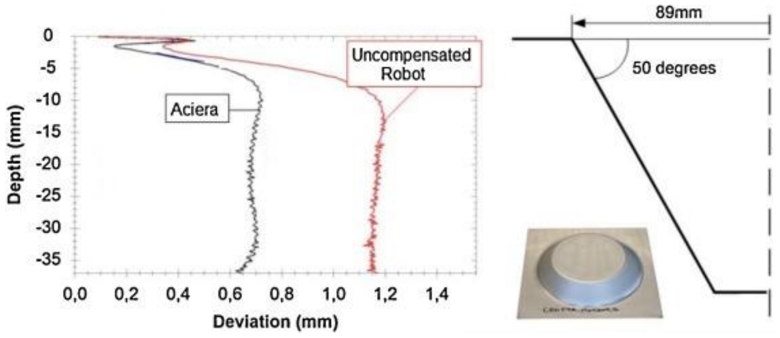
As one of the major geometric errors in DSIF, $\delta^1$ is caused by the position error of the forming tool. Various studies (Belchior et al., 2013; Belchior et al., 2014; Meier et al., 2011a) show that due to machines' limited stiffness, the forming tool deflects under the forming forces, causing a non-negligible position error, as shown in Figure 1.14. In addition, other error sources, e.g., clearances in gears and bearings, pitch errors in screws, servo errors, et al., also contribute to tool position errors. However, tool deflection remains the major contributor. This geometric error changes the shape of the tool path contour, thus leading to the geometric errors of the final part.

*To compensate for $\delta^1$, forming tool compliance compensation becomes an obligatory requirement since it is the role of the forming tool to define the part's geometric shape.*

In Meier's group (Meier et al., 2009; Meier et al., 2011a), a Finite Element Method (FEM) force prediction model was combined with a Multi-Body Simulation (MBS) model to predict the compliance value at the tool tip of the robot to accordingly compensate for the position errors, as shown in Figure 1.15. Similarly, Reddy et al. (Rakesh et al., 2016) compensate the tool compliance based on a force prediction model and assume a cantilever beam structure for the tool. In their work, forming tool deflections are compensated based on the tool's compliance modeled as a cantilever beam. Although the proposed methods are proven to be effective, they are strongly dependent on the accuracy of the force prediction model. However, a precise force prediction model is usually hard to establish due to the strong nonlinear nature of the IF process, and might lose its validity for different sheet metal materials and process parameters, such as tool diameters and the incremental depth.



**(a)**



**(b)**

**Figure 1.14 (a) Example of possible position errors in robotic incremental forming (Meier et al., 2011a) (b) Accuracy comparison between an Aciera milling machine and an uncompensated robot for a cone of wall angle 50° (Verbert, 2010)**

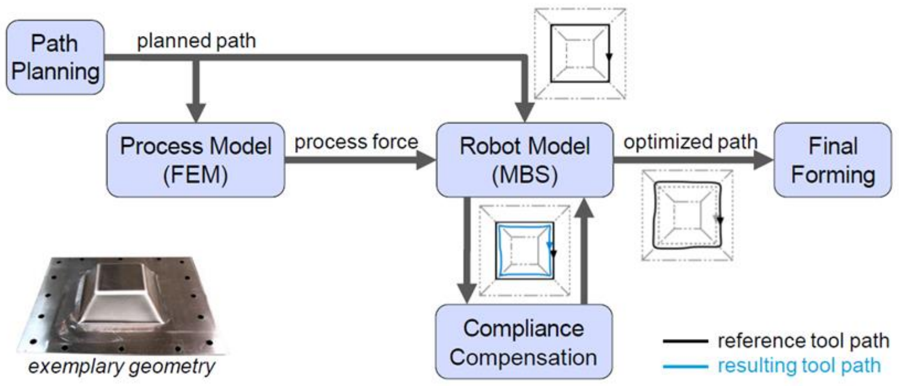


**Figure 1.15 Off-line compensation of robot-based DSIF (Meier et al., 2011a)**

### *1.2.2 In-process Springback Error*

There have been numerous efforts to correct or reduce the in-process springback error, $\delta^2$. Current research effort can be summarized under three aspects:

- **Process Enhancement**, in which a heat source like a laser or heat gun is added to the IF processes to soften the material and thus reduce global springback. For example, Duflou et al. (Duflou et al., 2007) and Göttmann et al. (Göttmann et al., 2011) improved the geometric accuracy of formed parts by laser-assisted incremental forming, where the material was locally heated by a laser beam as shown in Figure 1.16. Electrically-assisted incremental forming has also shown similar capabilities for springback reduction (Ambrogio et al., 2012; Fan et al., 2010; Meier and Magnus, 2013). These heat-assisted incremental forming processes exhibited reduced springback effects. However, they also required the integration of in-situ equipment, e.g., lasers or electrical current sources, which increase the cost of the process setup and complexity of the process control. Additionally, the introduced thermal stresses may contribute to springback after the parts are unclamped. Since the above-mentioned methods are usually energy intensive and cannot eliminate springback entirely, the focus in the academic field has moved to the utilization of toolpath modifications to compensate for springback.

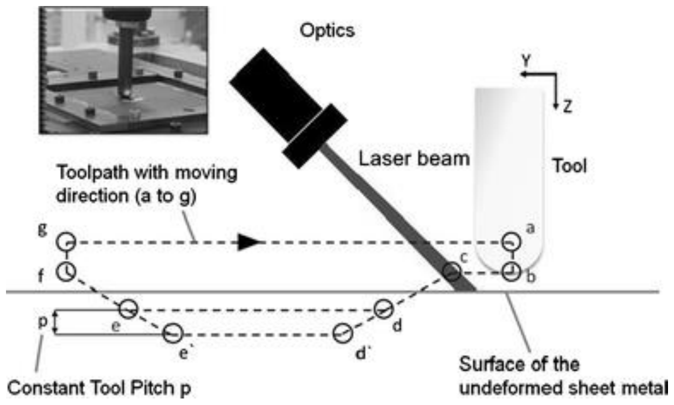

Optics
Y
Z
Laser beam
Tool
Toolpath with moving direction (a to g)
g
a
f
c
b
p
e
d
e`
d`
Constant Tool Pitch p
Surface of the undeformed sheet metal

**Figure 1.16 An example part with features on both sides of the metal sheet formed by DSIF (Göttmann et al., 2011)**

❖ **Feedforward Toolpath Optimization**, in which the toolpath is modified based on a predefined optimization model with no feedback involved. For example, Ambrogio et al. (Ambrogio et al., 2007) introduced an analytical model for truncated pyramid parts to over-deform and compensated for springback that occurred after unloading the forming tool but before removing the part from the clamps. A multi-stage forming strategy that reduced local springback in the final forming step, compared to single-stage forming, was suggested by Bambach et al. (Bambach et al., 2009). A model-based path planning system has been developed that takes into account not only the machine element compliances $\delta^1$ and also springback deviation predictions for their two-robot assisted incremental forming setup (Meier et al., 2011a). A similar compensation method was also proposed by Behera et al. (Behera, 2013; Behera et al., 2013). It utilizes multivariate adaptive regression splines (MARS) to predict springback error based on the different features of the desired geometry. Corresponding error correction functions can thus be developed, as shown in Figure 1.17. Although all these approaches have shown improvements in springback reduction without the need for any feedback devices, the use of feedforward compensation usually lacks robustness, since the springback error strongly depends on part geometry and the complex residual stress distributions in accordance with the essence of the typical incremental forming process. *In this work, a new toolpath strategy called Accumulative Double-Sided Incremental Forming (ADSIF) (Malhotra et al., 2012) will be explored, which has better process controllability regarding different geometric features. A corresponding feedforward toolpath modification framework will be proposed to optimize the tool positions and reduce the geometric error.*

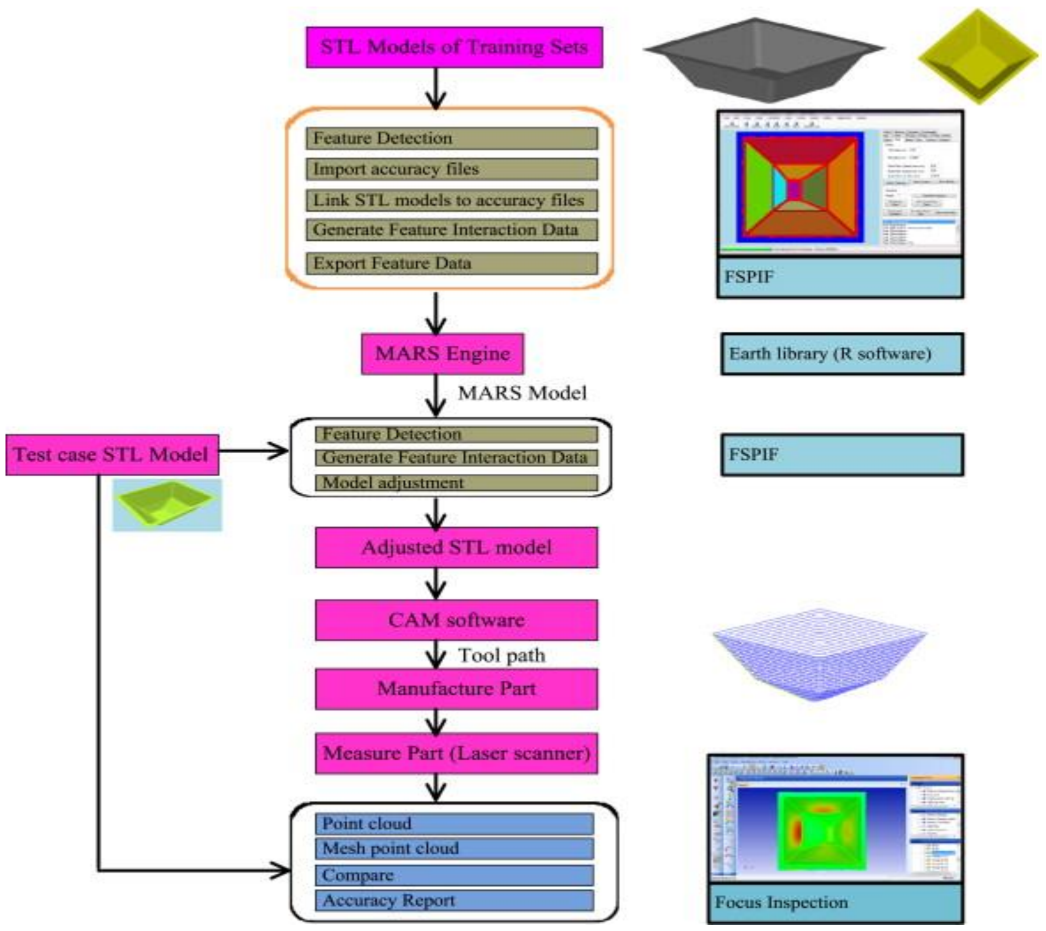


**Figure 1.17 Flowchart illustrating the procedure for the generation of MARS models and use of these models to improve part accuracy (Göttmann et al., 2011)**

- **Feedback Toolpath Compensation**, in which the part geometry is measured and utilized to iteratively compensate the error. The introduction of the concept of feedback control can greatly improve process accuracy even under model disturbances like sheet material property variations or clamping force fluctuations. For example, Hirt et al. developed a method that is iteratively correcting the CAD model by translating a scaled measure of the deviations for each point (Hirt et al., 2004). The drawback of using such a strategy requires making multiple test parts, measuring them and then applying the correction strategy, possibly in an iterative procedure, which significantly increases processing cost and time and makes the strategy unsuitable for small batch production. Fu et al. (Fu et al., 2013) developed a closed-loop toolpath correction algorithm for a truncated pyramid part using a Finite Element (FE) model to substitute for the physical trials. However, this strategy is strongly limited by the speed and accuracy of the FEM simulation, which usually takes days to run even for a simple geometry like a funnel or cone shape (Moser et al., 2016a). To address this problem, *a generic Finite Element Method (FEM) model will be established to predict the in-process springback error ($\delta^2$), which will account for different part geometries and manufacturing conditions. An on-line springback control loop will be proposed accordingly to compensate for $\delta^2$ during the forming process.*

### *1.2.3 Post-process Springback Error*

Compared to the various obstacles imposed by the in-process springback error ($\delta^2$), it has been proven, by both Bambach's and Zhang's work (Bambach et al., 2009; Zhang et al., 2016), that the post-process springback error $\delta^3$(occurring both during the clamp removal and trimming stages) can be effectively relieved by a simple annealing process. In the proposed stress-relief annealing method, the formed part is removed from the clamping system of the machine after the forming process and placed in a portable clamping device. The part and portable clamping device are then placed into a furnace and annealed, as seen in Figure 1.18. The annealing temperature and duration are chosen based on a benchmark test, in which a sheet strip is bent to a certain angle and then annealed in the furnace at different temperatures and duration to find the minimal bent angle reduction without material recrystallization. The proposed annealing method has the potential of being coupled with other springback reduction methods. Furthermore, the geometric accuracy of the part after trimming can significantly benefit from the stress-relief annealing as well. Note that this post annealing method is only conducted after the part has been formed and has no interference with the forming process itself. So it can be easily coupled with any of the above-mentioned methods for reducing $\delta^1$, $\delta^2$. *In this work, due to this simple correction of $\delta^3$, it will not be considered in the proposed work and only $\delta^1$, $\delta^2$ will be in our research scope.*

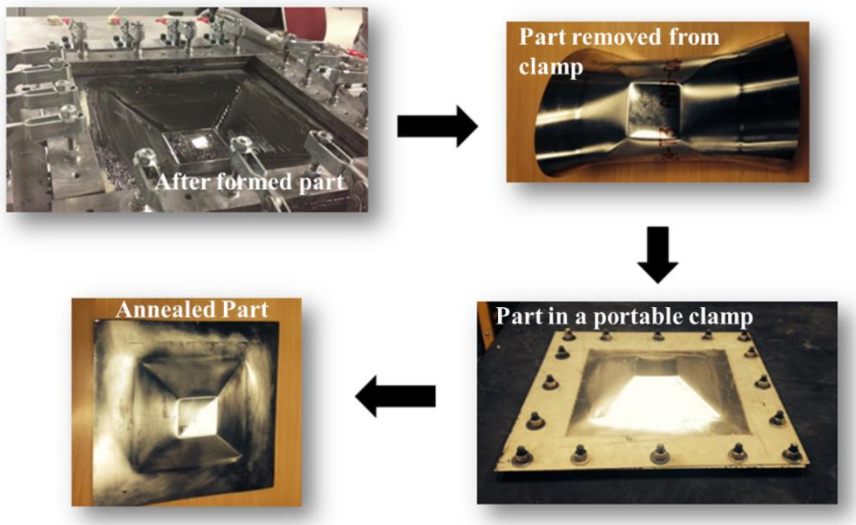


**Figure 1.18 An example of post-annealing to reduce the post process springback (Zhang et al., 2016)**

### *1.2.4 Process Formability*

In DSIF, the implementation of the additional supporting tool imposes a squeeze effect on the sheet metal, changes the stress triaxiality and improves process formability, as shown in previous research (Lu et al., 2015; Meier et al., 2011b). It has been shown that the nature of the contact maintained between the tools and the sheet improves the distribution of sheet thickness and hence, improves material formability and prevents premature fracture, as shown in Figure 1.19. However, DSIF degenerates into SPIF if contact between the supporting tool and sheet is lost. This degeneration occurs in DSIF when the tool gap between the two tools exceeds the local thickness of the sheet in the deformation region (Lu et al., 2015; Meier et al., 2007). To achieve better control of the tool gap, accurate estimates of sheet thinning and machine compliance must be considered and used for compensation in the conventional displacement-controlled DSIF system.

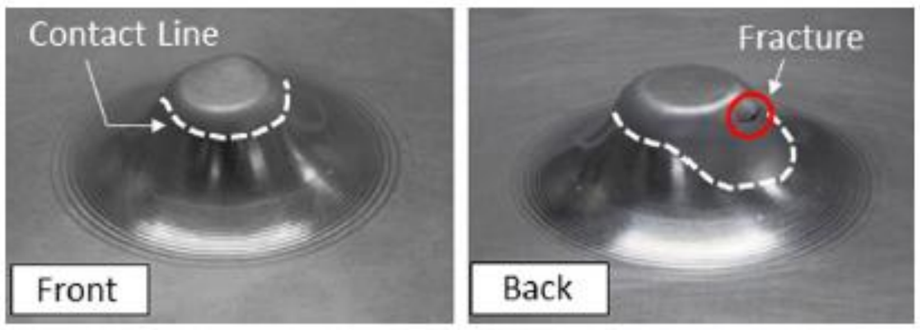


**Figure 1.19 Fracture in the region where contact is lost**

Another effort to increase material formability in the DSIF process is to increase the forming temperature during the process, similar to the process enhancement efforts mentioned in Section 1.2.2. Specifically, the Electricity Assisted Double-Sided Incremental Forming (EADSIF) rises the attention due to its simple application, low set up cost and easy temperature control (Valoppi et al., 2016). In EADSIF, electricity is directly applied to the forming and supporting tools, which imposes a certain current at the contact point on the metal sheet and locally heats up the material, as shown in Figure 1.20. However, if loss of contact happens between the supporting tool and the sheet, electric arcs will occur and damage both the sample and the tools. That being said, maintaining a stable contact condition between the tools and metal sheet is also vital for EADSIF's successful implementation.

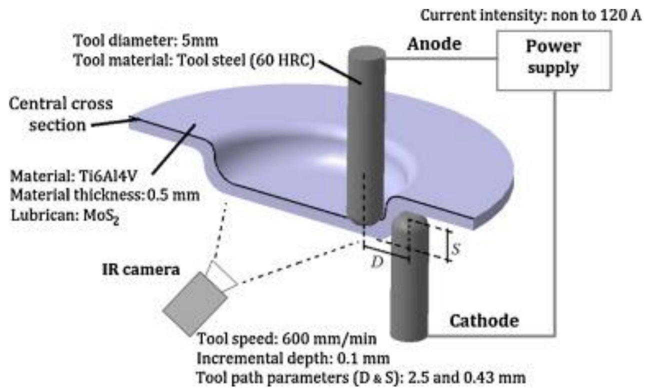

Current intensity: non to 120 A
Tool diameter: 5mm
Tool material: Tool steel (60 HRC)
Anode
Power supply
Central cross section
Material: Ti6Al4V
Material thickness: 0.5 mm
Lubrican: MoS$_2$
IR camera
D
S
Cathode
Tool speed: 600 mm/min
Incremental depth: 0.1 mm
Tool path parameters (D & S): 2.5 and 0.43 mm

**Figure 1.20 Sketch of the experimental set-up used for EADSIF(Valoppi et al., 2016)**

*To meet the aforementioned needs in DSIF and EADSIF, particular attention will be focused in this work on how to prevent the supporting tool from losing contact with, or over-squeezing, the sheet metal to enhance process formability.*

## 1.3 Objectives and Overview

As presented above, a significant amount of research can be found in the incremental forming field on tool position error compensation, springback compensation and material formability enhancement, in which the theoretical fundamentals are studied, and exploratory applications are presented. However, the industrial implementation of the proposed methods has only achieved limited success (Allwood et al., 2016; Gatea et al., 2016). To address the challenges outlined in Section 1.2, a model-based multilayer control system for the DSIF process is proposed in this work. By innovatively integrating feedback control and mechanics-based models into the DSIF process, the control system, as it will be shown, can effectively enhance part geometric accuracy and suppress possible fracture with high control accuracy, sufficient generalization capability and easy instrumentation setup practicability.

As one can see in Figure 1.21, in traditional incremental forming process setups, including Double Sided Incremental Forming (DSIF), the only implemented closed-loop controller is the tool-position servo controller of the machine, which acts to ensure that the equipment's actuators respond precisely to the commanded toolpath generated by an off-line process planning model. The DSIF process itself is being executed in an open-loop manner since no other process-related feedback information is used. The tool-position errors ($\delta^1$) and geometric errors from the in-process springback ($\delta^2$) are not considered and controlled in the above systems. Moreover, as a unique problem in the DSIF process, the contact pressure of the supporting tool is also not controlled, resulting in an early loss of contact and a decrease in material formability.

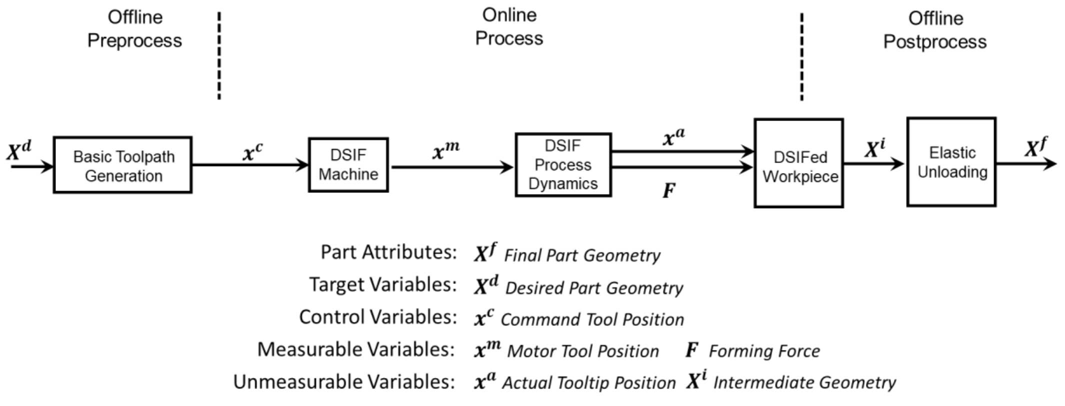

Offline Preprocess
Online Process
Offline Postprocess
$X^d$
Basic Toolpath Generation
$x^c$
DSIF Machine
$x^m$
DSIF Process Dynamics
$x^a$
$F$
DSIFed Workpiece
$X^i$
Elastic Unloading
$X^f$
Part Attributes: $X^f$ Final Part Geometry
Target Variables: $X^d$ Desired Part Geometry
Control Variables: $x^c$ Command Tool Position
Measurable Variables: $x^m$ Motor Tool Position $F$ Forming Force
Unmeasurable Variables: $x^a$ Actual Tooltip Position $X^i$ Intermediate Geometry

**Figure 1.21 Traditional open-loop controlled DSIF process**

To improve DSIF process outcomes, a system is established with three model-based control loops to achieve on-line/off-line compensation of the above-mentioned error. As shown in Figure 1.22, considering the specific desired part geometry ($X^d$), the system will optimize the toolpath parameters ($x^c$), and estimate toolpath correction values ($\Delta x$ ) based on measurable process variables, i.e., real-time tool position ($x^m$) and force ($F$), to achieve the desired part attributes, i.e., geometric accuracy and/or material structure integrity. Knowing that part attributes, e.g., final part geometry ($X^f$), are usually impossible to be directly measured in-process, the proposed work will successfully bridge the gap between the desired part attributes and the controllable process variables, which are the command positions ($x^c$) of the two tools, through the newly proposed model-based approach. As shown in Figure 1.22, the proposed control system includes:

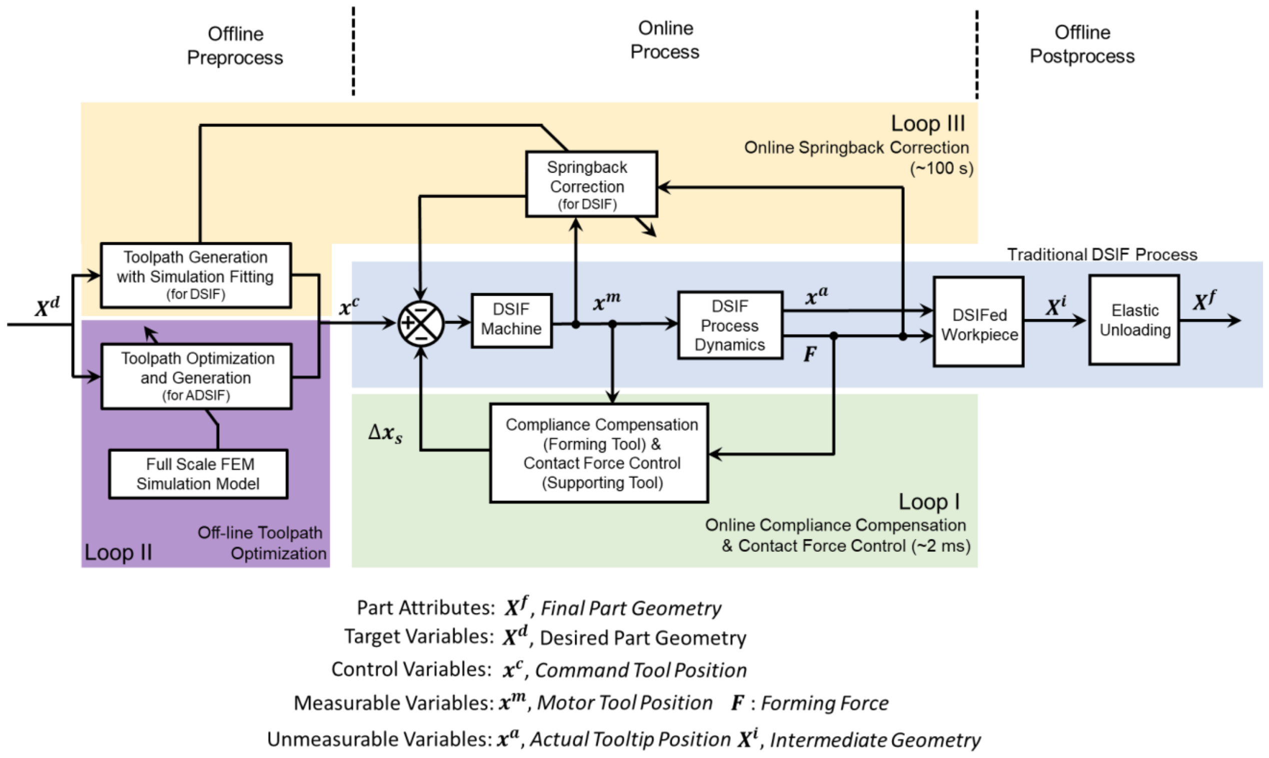

Offline Preprocess
Online Process
Offline Postprocess
Loop III
Online Springback Correction (~100 s)
Springback Correction (for DSIF)
Toolpath Generation with Simulation Fitting (for DSIF)
Traditional DSIF Process
$X^d$
$x^c$
DSIF Machine
$x^m$
DSIF Process Dynamics
$x^a$
$F$
DSIFed Workpiece
$X^i$
Elastic Unloading
$X^f$
Toolpath Optimization and Generation (for ADSIF)
Full Scale FEM Simulation Model
$\Delta x_s$
Compliance Compensation (Forming Tool) & Contact Force Control (Supporting Tool)
Loop I
Online Compliance Compensation & Contact Force Control (~2 ms)
Loop II
Off-line Toolpath Optimization
Part Attributes: $X^f$, Final Part Geometry
Target Variables: $X^d$, Desired Part Geometry
Control Variables: $x^c$, Command Tool Position
Measurable Variables: $x^m$, Motor Tool Position $F$ : Forming Force
Unmeasurable Variables: $x^a$, Actual Tooltip Position $X^i$, Intermediate Geometry

**Figure 1.22 Proposed control system for DSIF**

- ***Compliance Compensation and Force Control Algorithms (Loop I in Figure 1.22)*** to reduce tool positioning errors ($\delta^1$) caused by the tool compliance, ensure the optimum contact pressure and avoid any early loss of contact between the tool and the part. Both algorithms are directly superimposed over the conventional position servo control loop on a real-time base with minimal modifications. The establishment of the contact stiffness model increases the algorithms generality and robustness for different machine setups and material choices. The effectiveness of the approach is proven over a large range of process parameters with reasonable control accuracy ($< 10\ N$ for a $300\ N$ command force). Moreover, instead of a constant desired force value like the other implemented forming force controls (Lu et al., 2015; Meier et al., 2011b), the target force can be varied based on forming position, local wall angle or curvature.

- an ***Off-line Toolpath Optimization Algorithm (Loop II in Figure 1.22)*** which optimizes the relative tool positions of the forming tools to minimize the in-process springback ($\delta^2$). The toolpath optimization algorithm is designed specifically for an innovative toolpath strategy called Accumulated Double-Sided Incremental Forming (ADSIF) (Malhotra et al., 2012). The relative tool position of tools in ADSIF is, for the first time, defined by two input parameters, i.e., tool angle and tool gap, to capture the bending and squeezing phenomena in ADSIF. These process parameters are then optimized with a response surface model as a function of the local geometric wall angle based on a validated full-scale FEM simulation model. A significant error reduction of 80% is achieved via the developed optimization method utilized without the need for any feedback sensors.
- an ***On-line Springback Feedback Control Loop (Loop III in Figure 1.22)*** which dynamically modifies the toolpath to compensate for the potential in-process springback ($\delta^2$) based on real-time force signal feedback and a reduced-order simulation-based model. The springback toolpath correction algorithm generates a tool path adjustment vectors to compensate for the predicted springback based on the gathered forces signal during the forming process. The essential part of the on-line springback correction lies in a simplified reduced order model based on a purely-elastic Finite Element Simulation, which is being executed in nearly real-time (in seconds) as compared to the full-scale simulation, facilitating its use in the springback control loop. The feedback control loop is generalizable to different sheet materials, part geometries and boundary conditions due to the fact that FEM model can be easily adopted to all these parameters changed. A customized unsupervised classification algorithm is also utilized for the first time to extract

the key control points for given complex geometry, so that only a limited number of springback predictions are needed and the total simulation time can be further reduced.

The three control loops proposed in this work are designed to be integrated into a single system and operate together without conflicts since they are designed to execute at different update rates with different objectives, as shown in Figure 1.23. The off-line toolpath ADSIF optimization algorithm (Loop II), including toolpath generation and the required full-scale FEM simulation (shown in Figure 1.22), operates prior to the physical process without the need for on-line feedback. The springback control loop (Loop III) corrects the tool positions of both the forming and supporting tools on a semi-real-time schedule with a control loop update rate of around $10^2$ seconds, due to the limited computational speed of the simplified model. Note that Loop II and Loop III are operating on the host computer, which periodically uploads a series of toolpath commands to the motion controller. On the other hand, the compliance compensation and contact control loop (Loop I) operates on the motion controller in real-time based on the compensated tool path from the previous two loops (Loops II & III). In Loop I, the compliance compensation algorithm corrects the tool deflection of the forming tool, while the force control algorithm commands the supporting tool to follow the motion of the forming tool with the desired tool contact pressure to enhance structural integrality, i.e., suppress potential fracture of the incrementally formed part. In conclusion, the three loops are executed at different update rates (Loop I in real-time, Loop II in offline mode while Loop III in semi-real time) and different levels (Loop I in the motion controller, Loops II & III in the host computer). Such an arrangement allows them to be seamlessly integrated and simultaneously executed without interference.

It should also be noted that all three control loops are connected and empowered by different mechanics models. More specifically: (a) the contact stiffness model determines the control gain for the contact force and compliance compensation control loop (Loop I), (b) ADSIF toolpath optimization (Loop II) is established based on a response surface model which is retrieved from the full-scale FEM model and (c) the elastic FEM model in the springback compensation loop (Loop III) reads in the force signal and evaluates the springback value within seconds, which allows springback compensation to be conducted during the forming process.

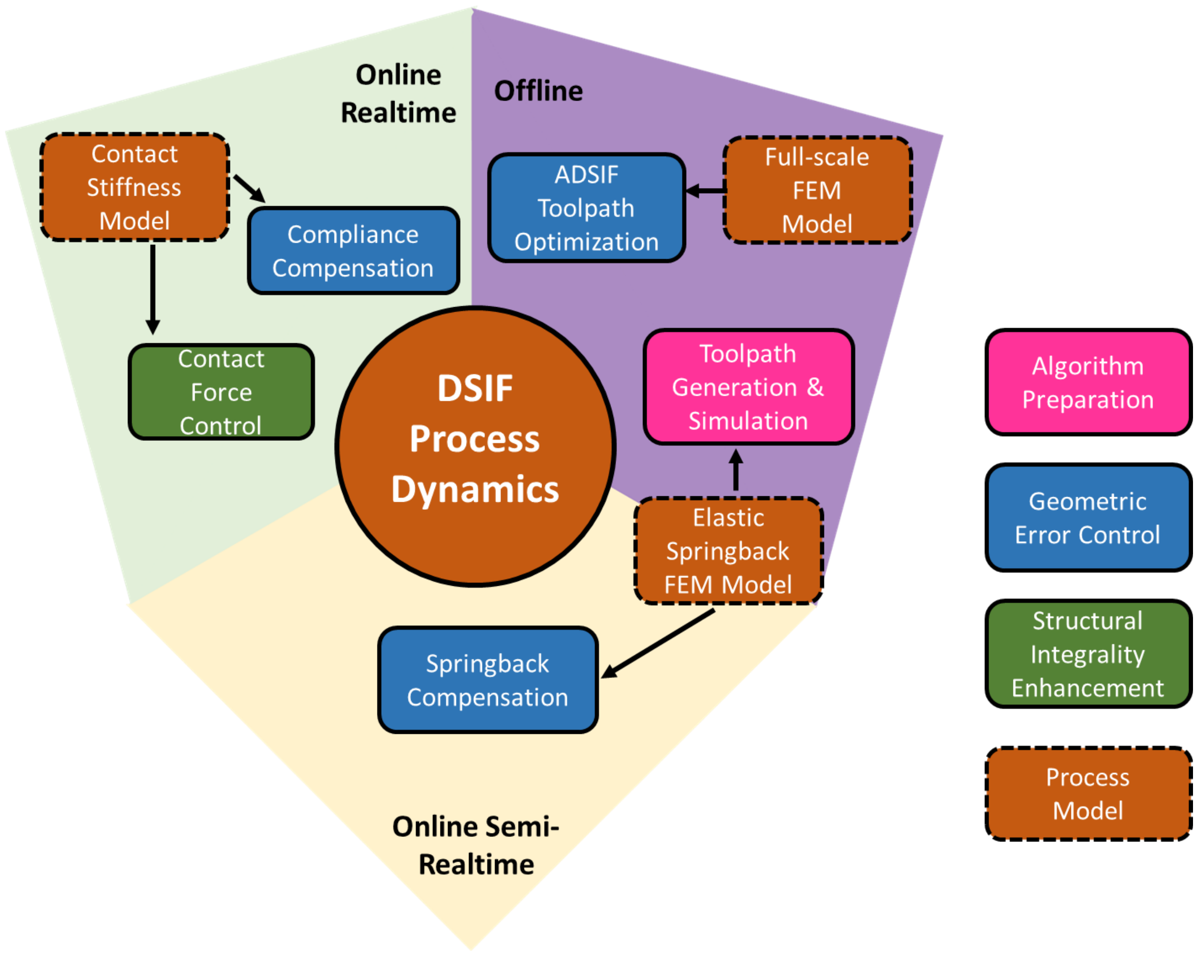


**Figure 1.23 Key components of proposed DSIF control system**

With the proposed system successfully implemented, the geometric error of the DSIF process will be continuously reduced by the multilayer tool position compensation control loops, and surface integrity will be enhanced by the supervised contact pressure control. Instead of the limiting the research focus on a specific geometry, material or other process parameters, the development/formulation of the contact stiffness and of the geometry-specific FEM springback predictive models will also significantly increase the system's generality for complex part geometries and various machine/material setups. Moreover, the control loops are designed to be seamlessly integrated into current IF procedures without significant intrusion into the original machine setup or increase in capital cost and lead time. Note that the proposed work will mainly focus on the different aspects of DSIF since DSIF has superior geometric and formability properties as compared to SPIF or TPIF. However, due to the similar nature of the different IF configurations as sequential and flexible forming processes, the proposed scheme will be easily extensible to other incremental forming processes as well. For Loop I, the tool position compensation approach can be directly implemented in SPIF and the contact force control can be applied in TPIF with minimal modifications. The springback compensation framework in Loop III is also extensible for either SPIF or TPIF. In conclusion, the proposed work is expected to significantly reduce the performance bottlenecks caused by the lack of the desired accuracy and part integrity. As a consequence, one can easily envision an expanded potential for incremental forming in a broad range of industrial applications.

The thesis is organized as follows:

Chapter 2 first introduces the compliance compensation and force contact control algorithms for DSIF, as shown by Loop II in Figure 1.22. A brief literature review regarding machine error compensation and force control will be firstly introduced. The machine and controller at Northwestern University on which the algorithms are implemented are also explained. The compliance/compensation force control algorithm is designed based on an explicit force control scheme and can be readily implemented using common motion controllers. The process contact model is also established based on sheet-tool interactions considering machine compliances. The algorithms have been proven to be suitable for complex geometries while also being capable of varying the local contact force (i.e., stress state) as a function of position.

Chapter 3 mainly proposes a toolpath optimization scheme for ADSIF (Loop II in Figure 1.22). The ADSIF process is first introduced, in which two tools are always positioned at a given horizontal plane but deform the sheet via an in-plane motion. The difference between ADSIF and DSIF is then explained not only from the deformation mechanics, but also from the process controllability point of views. Thanks to the unique advantage of ADSIF, off-line tool path optimization can be established based on a FEM-aided simulation response surface model, which is focused on the effects of the relative tool positions in ADSIF on the final part geometry. The relative tool positions are defined using the tool position angle and squeezing effect, which are related to the bending and squeezing phenomena, respectively, while the final geometry is discretized as a series of targeted wall angles. The effect of tool compliance can also be pre-compensated in the developed model. An error reduction of 80% is observed with the toolpath optimization model.

Chapter 4 presents an on-line springback compensation algorithm for conventional DSIF (Loop III in Figure 1.22). Current springback compensation efforts are first reviewed with their limitations highlighted. A FEM-based springback prediction model is first established to calculate the springback value based on the current forming forces, thus linking the target variable $X^f$ (final part geometry) to the measurable process variables $F$ (forming forces). During the forming process, the springback values of the intermediate part shape will be first evaluated at several control points based on the part geometry features, powered by an unsupervised machine learning classification algorithm. The correction value is then calculated based on the current springback values, and then the compensation is made in the subsequent toolpath to minimize the in-process springback error $\delta^2$. The results have shown that the springback error can be limited to within the 1.5 *mm* range for the tested geometries, compared with more than 5 *mm* geometric error for the uncontrolled parts.

Chapter 5 summarized the thesis by highlighting the major contributions of the work, followed by new ideas for future work.

# Chapter 2 Compliance Compensation and Force Control

To compensate for tool position errors and maintain a stable contact force during the incremental forming process, on-line control algorithms are desired to dynamically modify the tool position based on measured tool position/force signals, which is previously introduced as Loop I in Figure 1.22. In this chapter, current research results in this field will be reviewed with a particular highlight on their limitations, especially from the viewpoint of their generality or efficiency. The DSIF machine system utilized in this work is then introduced with the control system highlighted in Section 2.2. The compliance compensation and force control algorithms are then explained in detail in Section 2.3, followed by the experimental validation in Section 2.4.

## 2.1. State-of-the-Art

The major objective of this part of the proposed work is to compensate the compliance error of the forming tool and control the contact force of the supporting tool. In this section, the research efforts in the Incremental Forming (IF) field regarding these objectives will be introduced. Notably, the past modeling work in conventional forming or Single Point Incremental Forming (SPIF) offers a profound and inspiring foundation to obtain a better understanding of the underlying mechanics of the DSIF process. Besides the research conducted in IF, a brief glance will also be cast on robot and machining process control, which shares a common similarity with the proposed control scheme. The control algorithms and error compensation methods selected here for review are those that are plausible contenders that provide a base for the elements of the proposed control system for DSIF. A summary and discussion section is given to summarize the shortcomings in the current state-of-the-art and -technology, which also presents the objective for the work to be proposed.

### *2.1.1. Compliance Compensation in Double-Sided Incremental Forming*

It is known that one of the major geometric errors in DSIF is caused by the tool position error of the forming tool. The forming tool position errors change the shape of the tool path contour, thus leading to geometric error in the final part. This error can be corrected based on forming tool position compensation since it is the role of the forming tool to define the part's geometric shape.

As briefly reviewed in Section 1.2.1, the machine/tool compliance issue in IF has attracted attention from multiple research groups. Miere's group combined a Finite Element Method (FEM) force prediction model and a Multi-Body Simulation (MBS) model to predict the compliance value at the tool tip of the robot and compensate for the position errors accordingly (Meier et al., 2009;

Meier et al., 2011a). A similar work has been done in Belchior's group (Belchior et al., 2013) with the framework shown in Figure 2.1. However, both works are limited regarding the cycle time since their full-scale simulations are too time-consuming for this application, and sometimes they lack the required accuracy in force prediction.

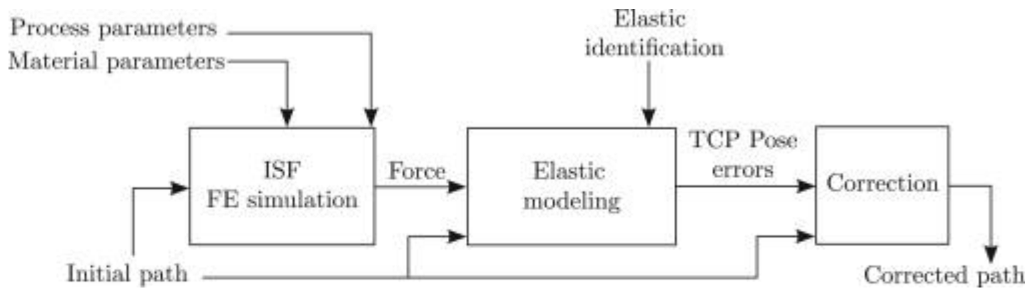


**Figure 2.1 Off-line compensation framework for robot-based DSIF adopted in (Belchior et al., 2013)**

For the DSIF process, Reddy's group (Rakesh et al., 2016) offered a framework for off-line tool compliance and sheet deflection compensation based on a force prediction model. In their work, deflections of the forming tool are compensated based on the tool's compliance that is modeled as a cantilever beam as shown in Figure 2.2, and then the supporting tool is shifted by the same amount of displacement to follow the forming tool. The major problem in their work is that the force prediction model they proposed contains too many assumptions and is hard to be generalized in real cases involving various sheet specifications and part geometries.

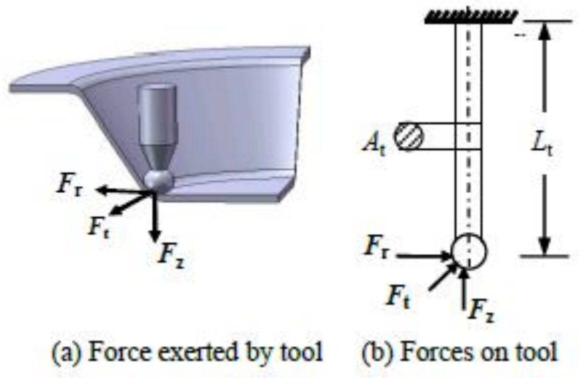


**Figure 2.2 Assumed cantilever beam model for tool compliance in (Rakesh et al., 2016)**

Besides the geometric errors caused by the forming tool, there is another particular contact error (loss of contact) resulting from the enlarged tool gap generated by the tool compliance from both the forming and supporting tools. The tool gap is defined as the minimum distance between the two tools, as shown in Figure 2.3. During the forming process, as both the forming and supporting tools are enduring the forming forces, their positions deviate from the commanded ones. This positioning error will result in an enlarged tool gap, and cause loss of the contact of the supporting tool. For example, an absolute position error of 0.1 *mm* caused by the deflection of the forming tool is usually negligible in a traditional sheet stamping operation, particularly considering the parts with dimensions around 100 *mm* or even larger. However, in a typical DSIF example, where 0.5 *mm* thick metal sheets are utilized, this 0.1 *mm* tool deflection will cause the tool gap to increase by 20 %. In most of the cases, this difference will reduce the squeezing effect between the forming tool and the supporting tool and lead to potential loss of contact.

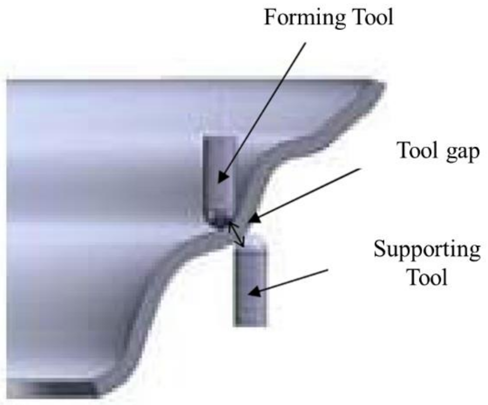


**Figure 2.3 Schematic showing the tool gap in DSIF (Malhotra et al., 2011)**

### *2.1.2. Compliance Compensation in Other Conventional Manufacturing Processes*

With reference to the previous review regarding the tool/compliance methods in IF, one can see that off-line compensation requires a force prediction model prior to the physical forming process. However, the force prediction model either requires a time-consuming FEM simulation model or may not be accurate and generic enough to offer viable force perditions for various industrial applications. Thus, an on-line compliance compensation model is desirable to compensate for the machine’s compliance errors.

Although on-line tool compensation has not been implemented in the incremental forming field, one can find similar applications related to machining processes. For example, in the adaptive CNC system proposed by Watanabe et al. (Watanabe and Iwai, 1983), the translation of the tool is utilized to perform a real-time correction for the tool compliance based on the force feedback signal, as shown in Figure 2.4.

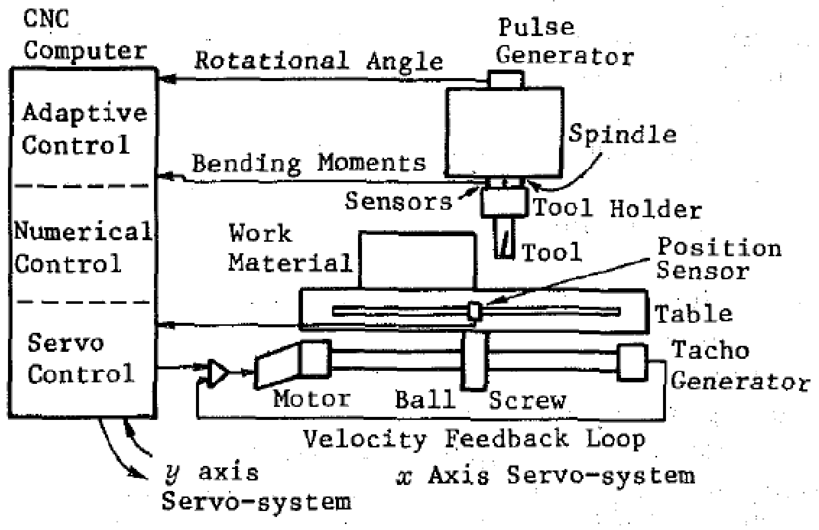


**Figure 2.4 Adaptive CNC force control system proposed in (Watanabe and Iwai, 1983)**

Similar work has also been conducted by Choi et al (Yang and Choi, 1998), in which a tool deflection compensation system is established with the ability to measure the cutting force, estimate the surface error and adjust the tool position on-line to reduce the machining errors due to the tool deflection in end milling, as shown in Figure 2.5.

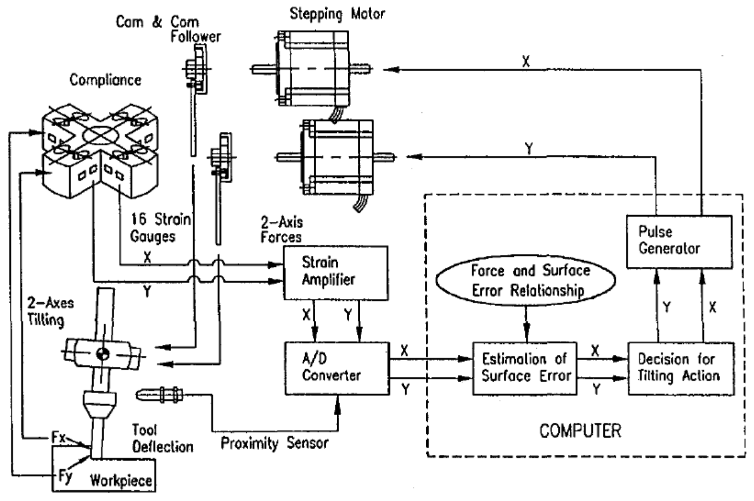


**Figure 2.5 Configuration of the tool deflection compensation system in (Yang and Choi, 1998)**

Another work proposed by Ehmann et al. in 1994 (Wang and Ehmann, 1994) expands the previous frameworks of on-line tool compensation control. Instead of assuming uniform system compliance throughout the whole workspace, a compliance matrix is measured at different locations. Then, a shape function inspired interpolation algorithm is implemented to update the compliance matrix value based on the current tool location.

### *2.1.3. Force Control in Displacement-Controlled DSIF*

For the incremental forming process, several attempts regarding forming force control have been reported since the forming force signal is one of the few common measurable variables in the process.

In 2009, Hascoet et al. proposed a force monitoring and control system for SPIF or TPIF (Rauch et al., 2009). They monitor the motor current and voltage and calculate the forming force accordingly. The calculated forming forces are being monitored throughout the process. Once the forming force exceeds a specified value, the tool path is adjusted to avoid tool collision as shown in Figure 2.6.

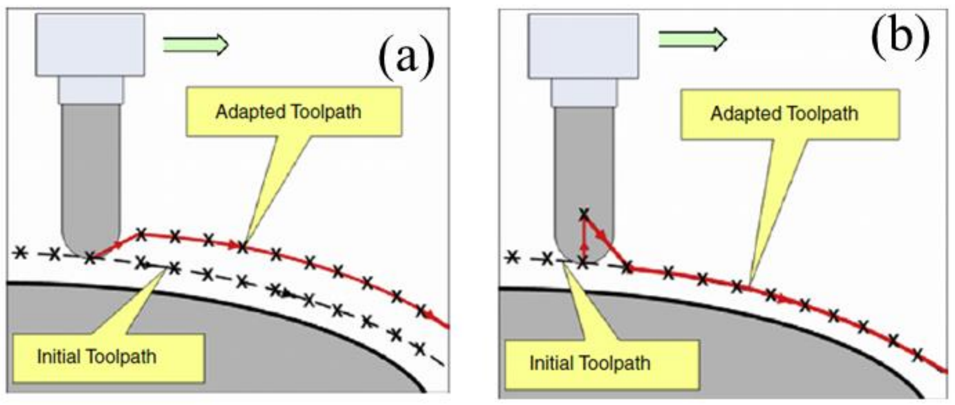


**Figure 2.6 Two different tool position change strategies proposed in (Rauch et al., 2009) to avoid collision**

Force control was also proposed as an indicator for possible fracture by Filice et al. in 2006 (Filice et al., 2006). In their work, the force is first smoothed with a low-pass filter. The change of the forming force is calculated and updated. If the force slope is lower than a preset value, fracture is likely to happen, and the process parameters need to be changed, like adopting a smaller incremental depth. However, since large force oscillations are often observed in the forming process of asymmetric parts, their proposed method seems to work only for axisymmetric parts, like a cone in their case, as shown in Figure 2.8.

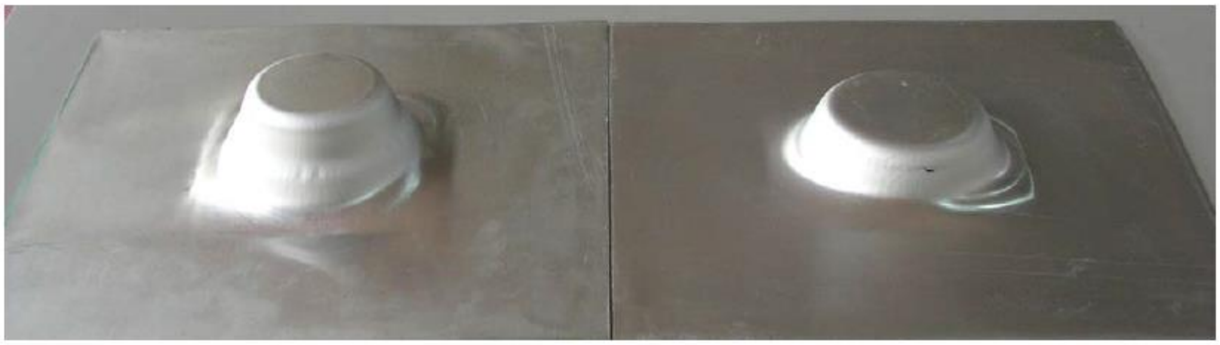


**Figure 2.7 Controlled part (left) and uncontrolled part (right) in (Filice et al., 2006)**

As reviewed in Sections 1.2 and 2.1.1, a special challenge and requirement in DSIF is how to guarantee stable contact conditions for the supporting tool. Experiments conducted on Northwestern’s displacement-controlled system (Moser et al., 2016c) show that the loss of contact condition depends on both local curvature and tool motion direction as illustrated in Figure 2.8. To compensate for the error and maintain both tools under stable contact conditions, the tool gap between the two tools is modified based on a correction function, $\boldsymbol{f}$, that considers both local curvature and motion direction. With the corrected toolpath, loss of contact is prevented, and the formability of the DSIF process is improved, as illustrated in Figure 2.9. However, the proposed method is purely empirical-based, and it would take a large number of experiments to generalize for different machine/material setups or different geometries, potentially using response surface model. In this sense, a closed-loop contact force control is preferred because it can maintain tool contact regardless of the machine configuration for various geometric features.

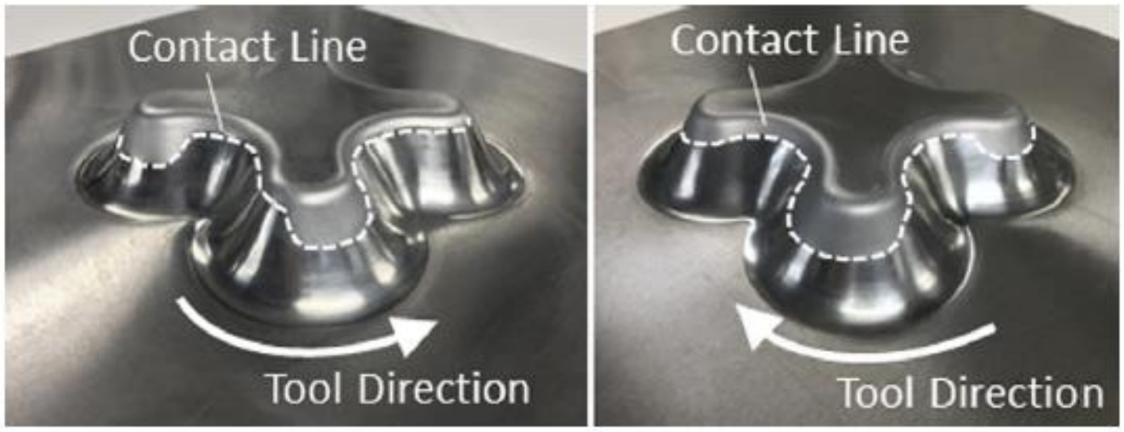


**Figure 2.8 Asymmetric loss of contact around rounded corners in (Moser et al., 2016c)**

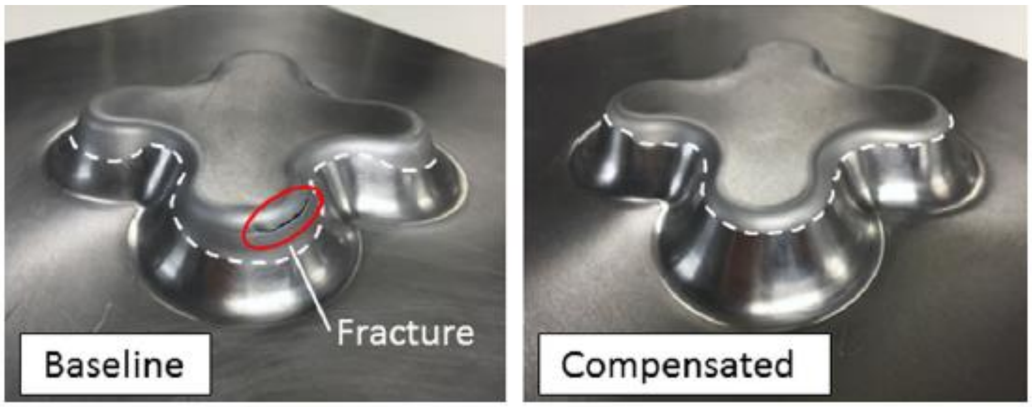


**Figure 2.9 Maintaining contact in DSIF increases material formability (Moser et al., 2016c)**

Another study by Ren et al. regarding the influence of machine compliance in DSIF also reveals a similar influence of tool contact (Ren et al., 2016). In this work, a FEM model with deformable tools is proposed, and different machine compliance values are virtually tested. It is founded that with a more rigid tool/machine setup, the forming forces become more substantial, and more uniform thickness distribution can be observed, as shown in Figure 2.10. In other words, thinning is postponed if a better contact is maintained (a larger contact force). Moreover, this research also indicates that maintaining contact is not just merely an on/off phenomenon. The target contact forces need to be determined for different cases. Otherwise, a low forming force will compromise the effectiveness of the squeeze while a high forming force will likely make the support tool slide off, which is also reported in Meier's work (Meier et al., 2011b).

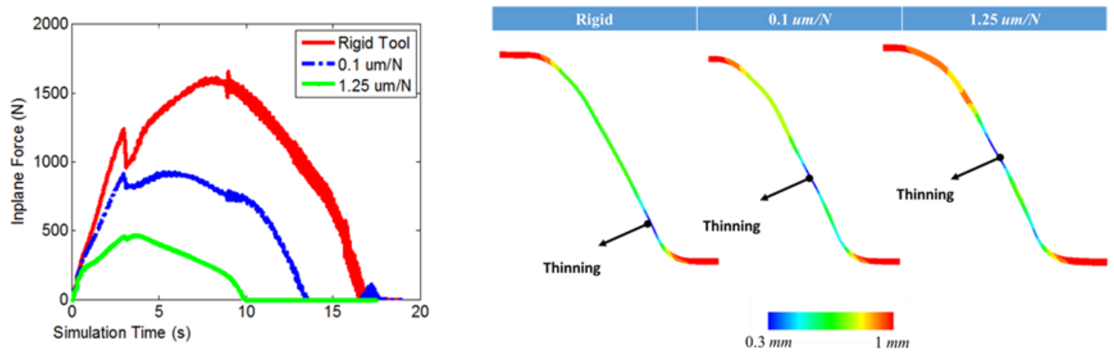


**Figure 2.10 Tool thinning conditions with different machine compliance values in (Ren et al., 2016)**

### *2.1.4. Force Control in Force-Controlled DSIF*

To overcome the obstacles of maintaining stable forming forces in purely displacement-controlled DSIF systems, researchers have implemented force control loops in certain Degrees-of-Freedom (DOF) of the machine's motion. A force-based control algorithm for the supporting tool has been implemented by both Meier et al. (Meier et al., 2011b) and Chen et al. (Lu et al., 2015). By utilizing a robot controller (Meier et al., 2011b) or hydraulic pressure actuator (Lu et al., 2015), the motion of the supporting tool is controlled to maintain a certain forming force. In their experiments, simple shapes (a funnel in (Meier et al., 2011b) as shown in Figure 2.11 and funnel/pyramid in (Lu et al., 2015) as shown in Figure 2.12) have been used in conjunction with different levels of empirical-based preselected forming forces, to determine and record the forming limits for the parts. Improvements in formability are observed in both cases. However, the magnitude of the controlled forming force needs to be further optimized based on more generic models. A low forming force will compromise the squeezing effect while a high forming force will likely make the support tool slide off as mentioned above.

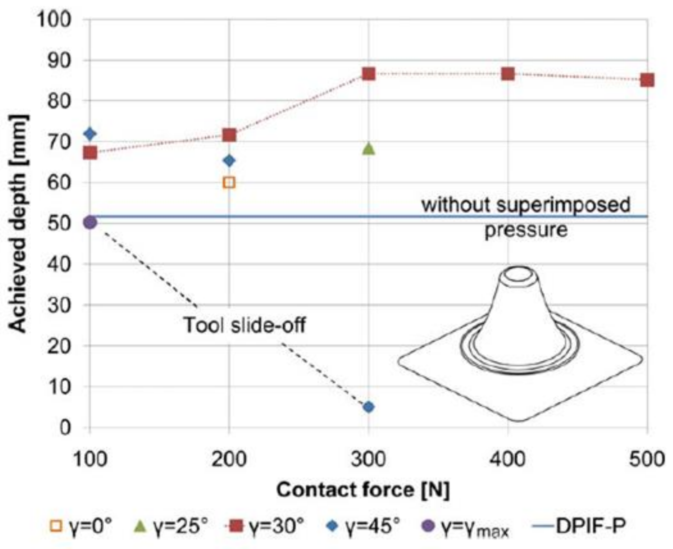


**Figure 2.11 Improved formability with imposed pressure in (Meier et al., 2011b)**

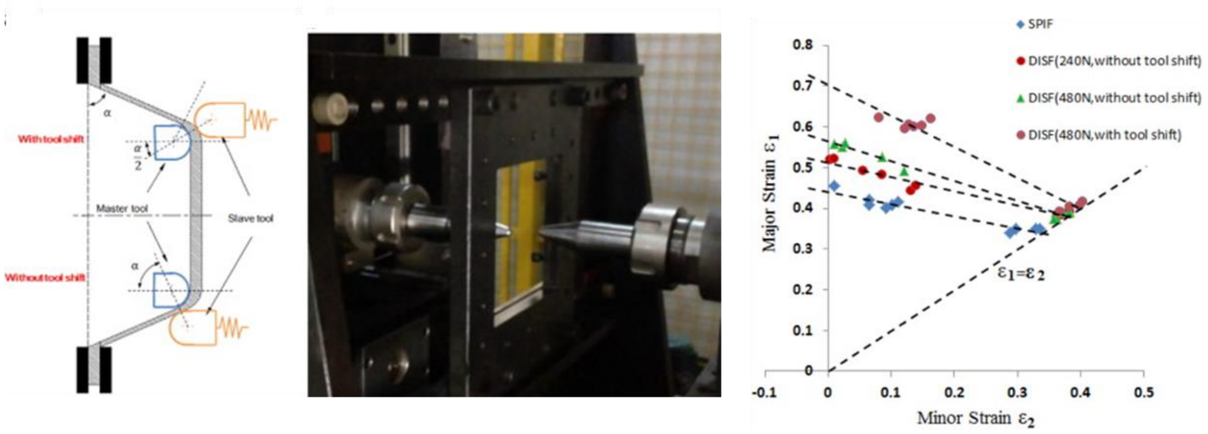


**Figure 2.12 Experimental setup and increased formability with imposed pressure in (Lu et al., 2015)**

Another issue related to the direct implementation of force feedback for the supporting tool in DSIF is that the system will likely become unstable in the directions in which no significant force-displacement interactions happen, like the tangential direction on the forming surface, as reported in the work of Meier (Meier et al., 2011b). Similar situations exist in machining processes as well, such as grinding, deburring and polishing, in which the tangential velocity of the tool along the workpiece and the force normal to the work surface need to be separately controlled (Jeon and Tomizuka, 1993). To solve this problem, Meier's group chooses to utilize position commands to control the supporting tool's motions and implement force feedback control in the direction presented by the connecting line between the two tools, i.e., the tool gap direction, mentioned in Section 2.1.1, and shown as $F_2$ in Figure 2.13.

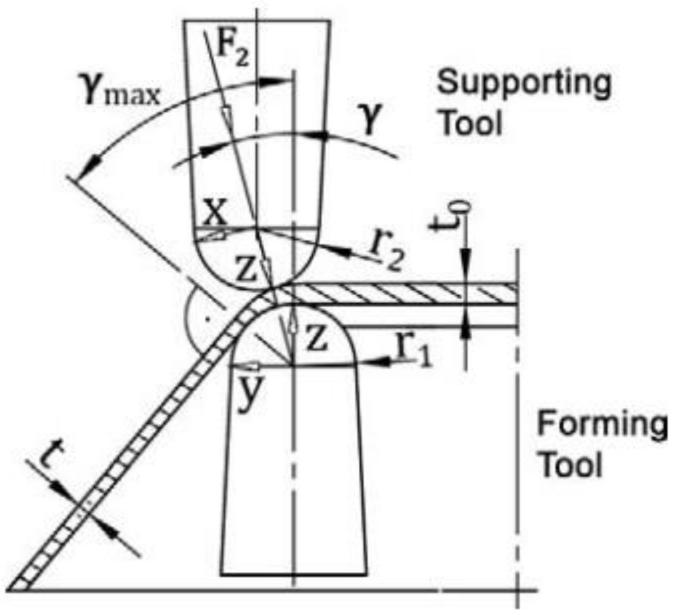


**Figure 2.13 Geometric conditions of the DSIF process and direction of the controlled force in (Meier et al., 2011b)**

### *2.1.5. Force Control in Other Conventional Manufacturing Processes*

To maintain a stable contact force level, a force feedback control loop will be needed to regulate the trajectory of the supporting tool based on the required force levels. Lots of works have been done in the past years to enable force feedback control on industrial manipulators in operations such as robotic assembly and in machining operation like grinding or polishing where the contact force is related to the material removal rate and needs to be controlled. The methods proposed by previous researchers can be a proper foundation for the force feedback algorithm developed in this work.

The article by Whitney in 1987 offers a comprehensive review of various types of continuous force feedback control systems (Whitney, 1987), all of which can be summarized by the overall architecture shown in Figure 2.18. The manipulator is commanded along a certain displacement or velocity trajectory, which is modified by motion updates derived from the force feedback strategy. Under conditions in which contact occurs between the manipulator and the target, the forces will be sensed and fed into the control strategy algorithm. This architecture can also deal with varied commanded force trajectories.

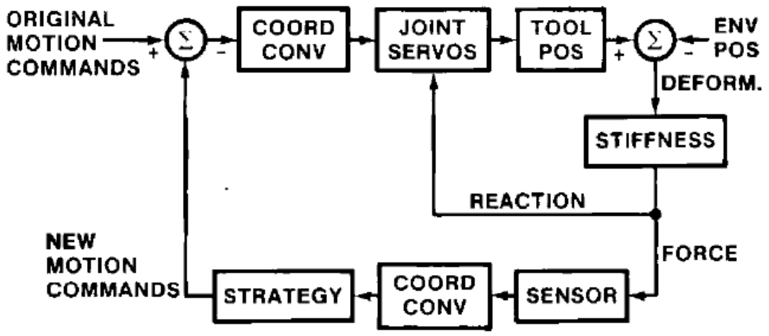


**Figure 2.14 An abstract architecture of the force feedback systems in (Whitney, 1987)**

Among the reviewed methods, the stiffness control method (Salisbury and Craig, 1982) and damping control method (Whitney, 1977) are the most commonly utilized ones due to their broad adoption in industry and their simple formation. Figure 2.15 shows a representation of stiffness control on an example of a robot arm and a zero-force target. In this diagram, **X** is the position vector while **$\tau$** is a vector of the arm joint torques, **$K_E$** is the net stiffness of all in-contact items, including both the arms (DSIF machine in the present case) and the environments (the tool-sheet interaction). In order to implement stiffness control, one can create a compliance matrix, **$K_{F1}$,** with a unit of position/force. In this case, the equilibrium contact force is given by:

$$F=[K^{-1}_{F1}]\underline{X}_D \tag{2-1}$$

Compared to stiffness control described above, damping control adopts an integrating controller in which the sensed forces give rise to velocity modifications. In this case, an admittance matrix, $\mathbf{K_{F2}}$, is required with a unit of velocity/force instead of the compliance matrix, $\mathbf{K_{F1}}$.

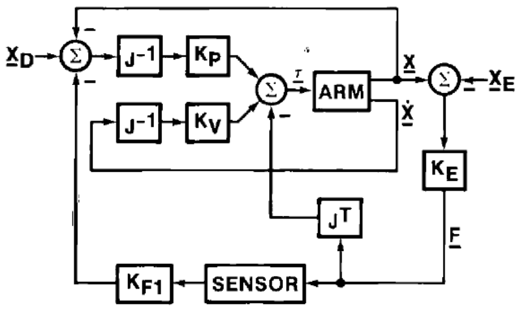


**Figure 2.15 A representation of stiffness control system in (Salisbury and Craig, 1982)**

An existing issue in implementing targeted force feedback control in DSIF is that usually not all the DOFs can be force controlled. The tool motion tangential to the sheet metal surface has to be fully position-controlled in order to impose the target geometry, while the forming force can only be controlled in the direction normal direction to the metal-tool contact surface. For example, in Meier's work, only the tool gap is force controlled while the other axes are still displacement controlled. In this case, a hybrid position/force controller is needed. This hybrid position/force control problem was first addressed by Craig et al. around 1981 (Raibert and Craig, 1981). In their work, the hybrid technique described combines force and torque information with positional data to satisfy simultaneous position and force trajectory constraints specified in a convenient task related coordinate system, as shown in Figure 2.16. A selection matrix is established to determine which global or local axes are to be controlled by force loops based on the constraint condition. Besides, coordinate transformations are also needed to transform the local into global coordinates.

Inside of the control loop, either a stiffness or damping control method, mentioned above, could be implemented.

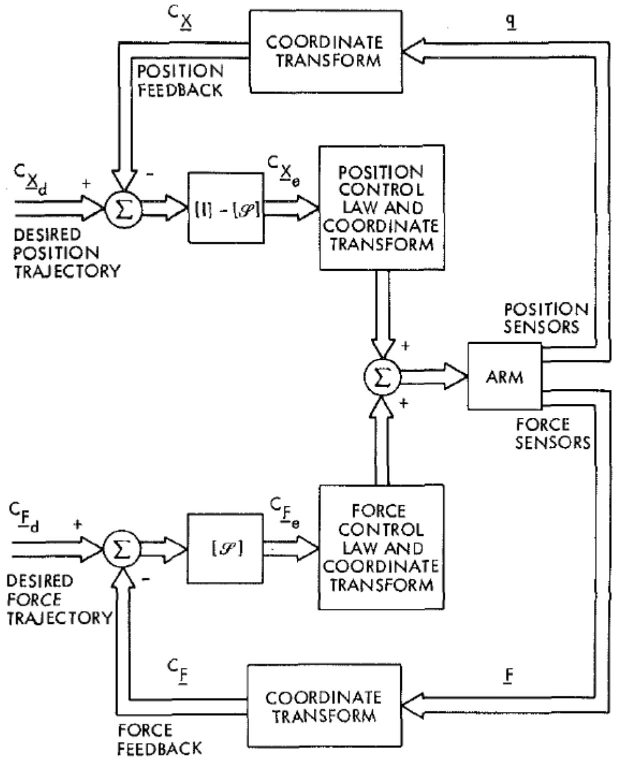


**Figure 2.16 Conceptual organization of the hybrid controller in (Raibert and Craig, 1981)**

### *2.1.6. Discussion and Research Objective Statement*

As it was discussed in the above review (Section 2.1.1), the correction of the $\delta^1$ (defined by Equation 1.1) requires an accurate position control of the forming tool. Furthermore, current off-line compensation methods usually require accurate force prediction models, which are hard to generalize for complex geometries. With respect to contact force control, the addition of the supporting tool in DSIF introduces a squeezing contact force and the opportunity to modify the local contact force and alter the stress triaxiality.  The control of the proper contact force requires an accurate control of the tool gap, which represents the distance between the two tool surfaces. Note that the traditional displacement-based control strategy in DSIF usually sets the tool gap to an assumed thinned sheet thickness, which is not capable of achieving a stable contact forming force control for generic geometric features. On the other hand, some force control loops have been successfully implemented in the DSIF process as reviewed in Section 2.2.3, but the controlled contact force is limited to a constant value and undesired tool sliding is often observed. .

In this work, particular attention is given to the development of *a robust compliance compensation and contact force control loop for DSIF (Loop I in Figure 1.22), which is validated for various materials, tool radii, and geometric features.* The general control scheme is also designed to be applicable for a broad spectrum of incremental sheet forming configurations.

## 2.2. DSIF Prototype Machine

The concept of DSIF can be conceptualized as shown in Figure 2.17 that includes tool path generation, motion control and finally the mechanical forming of the part. Among all the factors, the machine design (forming tools, clamping devices, and overall structure) and the control system (motion controller, encoder, load cell and A/D converter) will be introduced in this section. The stiffness of the machine and dynamic behaviors of the system will also be included.

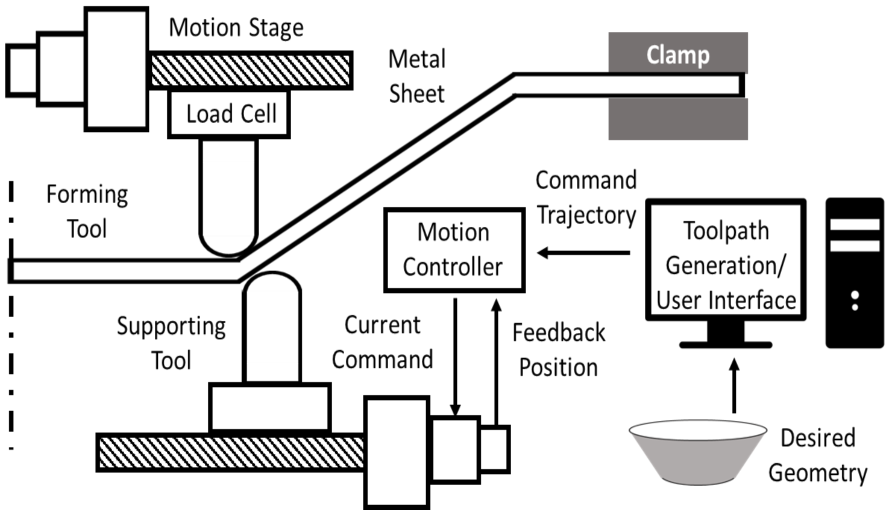


**Figure 2.17 Schematic diagram of the DSIF progress**

### *2.2.1. Machine Design*

A CAD model of the prototype DSIF machine at Northwestern University is shown in Figure 2.18. It is a customized machine with two coordinated 3-DOF serialized structures. The overall frame size is 0.9 * 0.75 * 1.9 $m^3$ constructed from 2 * 3 $inch^2$ L-extruded steel. The horizontal axes of both the forming and supporting tools are powered by two linear guides for each axis as a double gantry system, while the vertical axes are powered by a single linear guide. Each linear guide is actuated by its own servo motor with ten motors in total. Note that the application of the double gantry system can effectively prevent the twisting of the structure caused by the forming moment. Moreover, both motors in the double gantry axis are actively controlled and coordinated, instead of having a master-follower system to ensure that one guide does not get stuck. All motors are controlled by a Delta-Tau Turbo PMAC control system which will be discussed in Section 2.2.2.

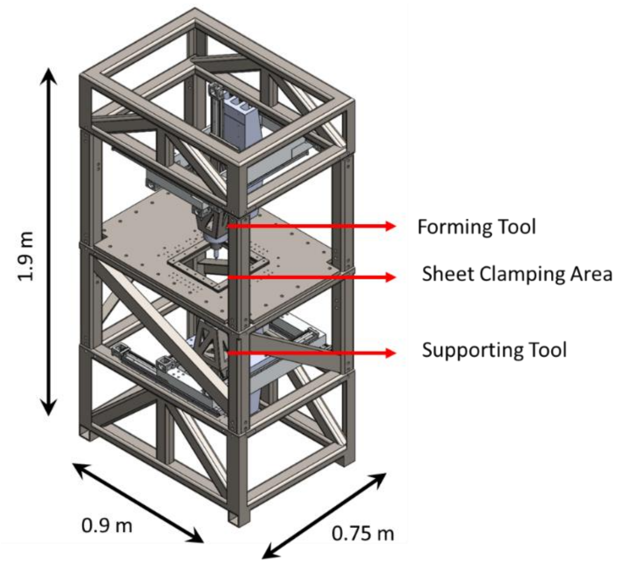


**Figure 2.18 CAD model of the DSIF prototype machine at Northwestern University**

The tool holder is a customized design based on a milling machine collet tool holder, with changeable collets to hold forming tools with different diameters. Generally, precision tools with a spherical diameter are used ranging from 5 *mm* to 10 *mm*. Multipurpose lubricant/grease infused with fluoropolymers (i.e., Teflon) is used between the tools and the sheet to minimize friction effects. The sheet fixture is designed with adjustable toggle clamps with the maximum clamping area of 0.32 * 0.32 $m^2$ and the formable center region of 0.25 * 0.25 $m^2$. The clampable sheet thickness ranges from 0.5 *mm* to 2.0 *mm*, while the maximum clamping force can reach up to 2,000 *N* in the vertical direction to ensure that no slippage of the sheets occurs.

The fabricated DSIF machine with the tool/clamp design is shown in Figure 2.19. Note that the X-Y-Z directions correspond to a right-handed Cartesian system, with the X and Y axes being the horizontal axes while the Z axis is the vertical axis. This DSIF machine can take a maximum force of 2,000 *N* and 1,500 *N* in the Z axis and X/Y axes, respectively.

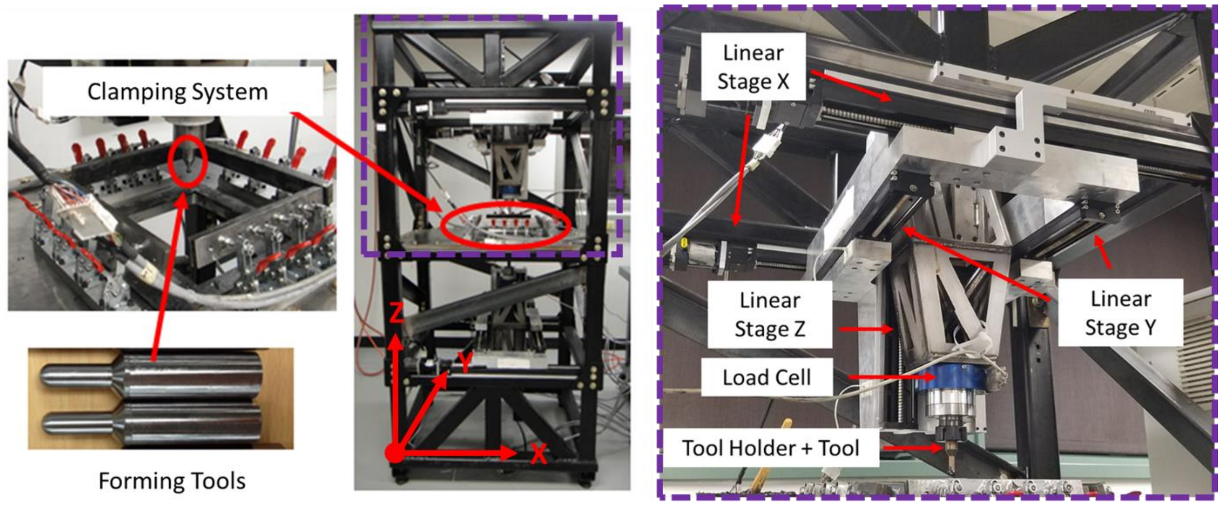
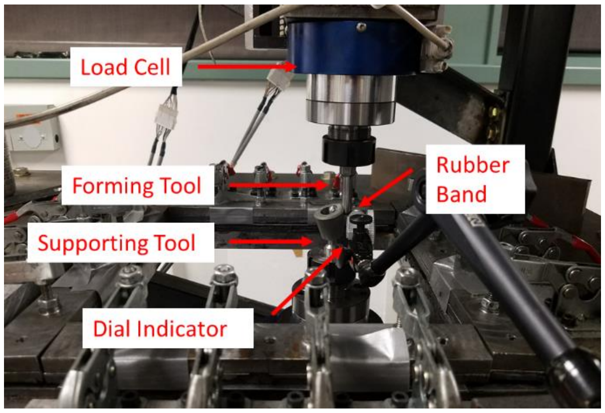


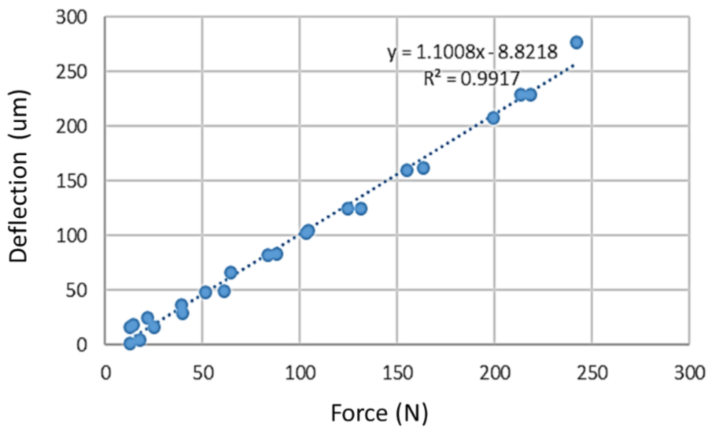
**Figure 2.19 Physical DSIF prototype machine at Northwestern University**

To implement the force control and stiffness compensation algorithm in the DSIF machine, two strain-gage-based 6-axis load cells are installed in series with the forming tools allowing the measurement of both transverse and axial loads. The signals from the load cell (JR3 Multi-Axis Load Cell 75E20) are first amplified and then input into a 12-bit multichannel A/D card with a resolution of 1 *N*, which is integrated into the Delta Tau Turbo PMAC controller, as later explained in Section 2.2.2.

In order to compensate the machine compliance, the stiffness of the forming tools must be characterized. In the setup, two tools are squeezing a rubber pad between them. The squeezing force is recorded by the load cell, while the tool deflection is measured at one of the tool tips by a dial indicator, as shown in Figure 2.20(a). A sample force-deflection curve is shown in Figure 2.20(b), which shows that the machine deflection varies proportionally to the external force, i.e., that the compliance value can be assumed as a constant in this case.



**(a)**



**(b)**

**Figure 2.20 (a) Tool stiffness measurement when two tools are squeezing a rubber band, (b) a sample force-deflection for the X axis of the supporting tool**

The tool compliances are measured for each axis of both the forming and supporting tools, at three different locations six times per each location. The characterized results are summarized in Figure 2.21. Note that the compliance value can be assumed to be constant in the workspace due to the small variations. Moreover, the compliance values of the horizontal axes (X, Y axes) are generally one magnitude larger than the ones in the vertical axis (Z axis). Due to the small compliance value (or high stiffness) in the Z axis, compliance compensation will only be performed for the in-plane deflections without compensating the Z axis.

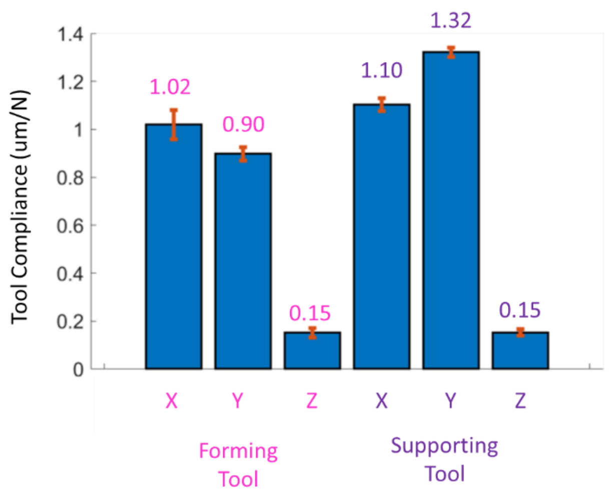


**Figure 2.21 Measured tool compliance values for different axes**

### *2.2.2. Control System*

The ten servo motors mentioned above are all commanded by a Turbo PMAC motion controller as shown in Figure 2.23, manufactured by Delta Tau Systems. The Turbo PMAC is one of the most widely used open source motion controllers with the capability to control up to 32 motors with 12 coordinate systems, typically running at a servo frequency above 2.2 *kHz*.

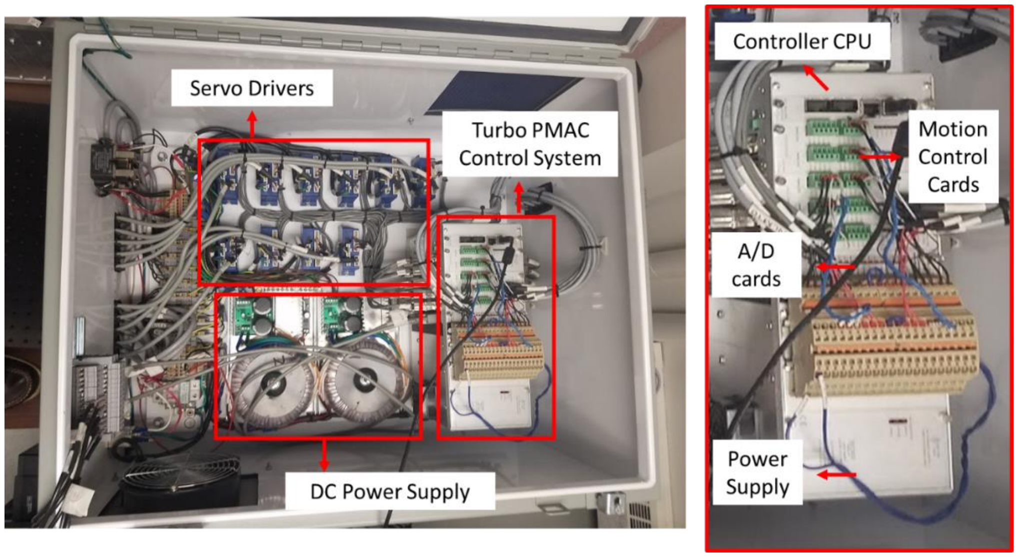
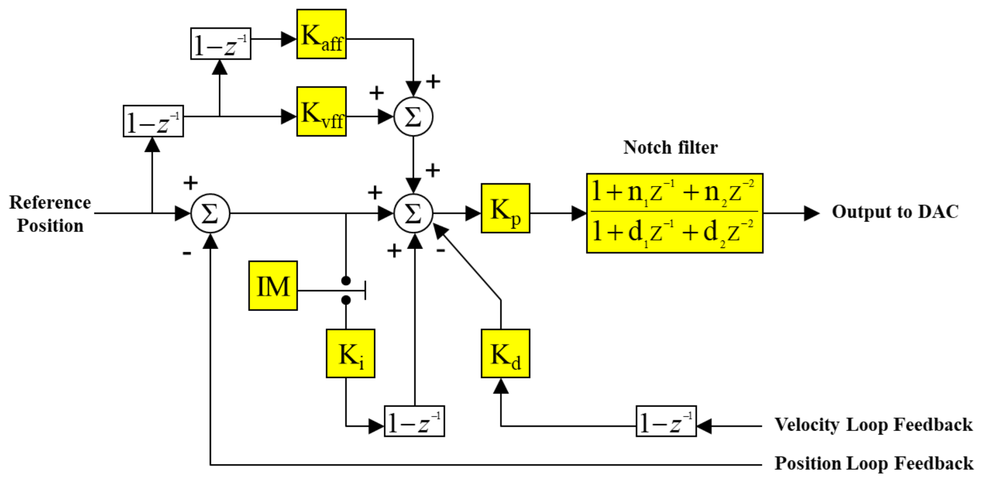


**Figure 2.22 Control system utilized in the prototype DSIF machine with a zoom-in view of the Turbo PMAC control system**

In the Turbo PMAC system, a Proportional-Integral-Derivative (PID) control loop is implemented for each servo motor control, with the control diagram shown in Figure 2.21 (PMAC, 2008). The vital tunable parameters are also listed in Figure 2.23. Note that for most applications, the notch filter is not implemented unless there is a certain frequency band in the motion signal that needs to be filtered out to avoid potential structural resonances. The velocity feedforward ($K_{vff}$) and acceleration feedforward ($K_{aff}$) terms are utilized to reduce the following error. However, the existence of the feedforward terms might cause the system to oscillate in response to any discontinuity in the command position signals, which need to be suppressed in the compliance compensation and contact force control algorithm as discussed later in Section 2.3.4. The IM (integration mode) is a clamp anti-windup scheme (Astrom and Rundqwist, 1989) to avoid any instability caused by the integration term. There also exists some saturation limiters to protect the motors, which are not shown in this diagram.



➢ **PID Parameters**

| | |
|---|---|
| $K_p$ | Proportional gain |
| $K_d$ | Derivative gain |
| $K_{vff}$ | Velocity feedforward gain |
| $K_i$ | Integral gain |
| IM | Integration mode |
| $K_{aff}$ | Acceleration feedforward gain |

➢ **Notch Filter Coefficients**

$n_1$ $n_2$ $d_1 d_2$

**Figure 2.23 PID servo loop implemented in Turbo PMAC (PMAC, 2008)**

The motion accuracy for the system is also characterized. The linear position error is first characterized via a laser interferometer (HP5529 Dynamic Calibration System) as shown in Figure 2.24. The maximum position errors in the Y axes are 2 $\mu m$ over a ±100 *mm* motion range for both tools, while the maximum position errors in the Z axis are 12 $\mu m$ over a ±100 *mm* motion range. However, the maximum position errors in the X axis are 350 $\mu m$ for the forming tool and 425 $\mu m$ for supporting tool, which cannot be neglected. Two 1-D error compensation tables have been implemented in the Turbo PMAC controller, and the position error has been reduced to 10 $\mu m$ / ±100 *mm*. The motion error is then checked with a circular motion in the X-Y plane with a 50 *mm* ball bar, which shows that the maximum radial position error is less than 15 $\mu m$ under no-load conditions. In the following context, a motion error (< 20 $\mu m$) will be assumed to be negligible compared to the tool deflection (~ 300 $\mu m$ for a typical in-plane forming force), and thus will not be further compensated.

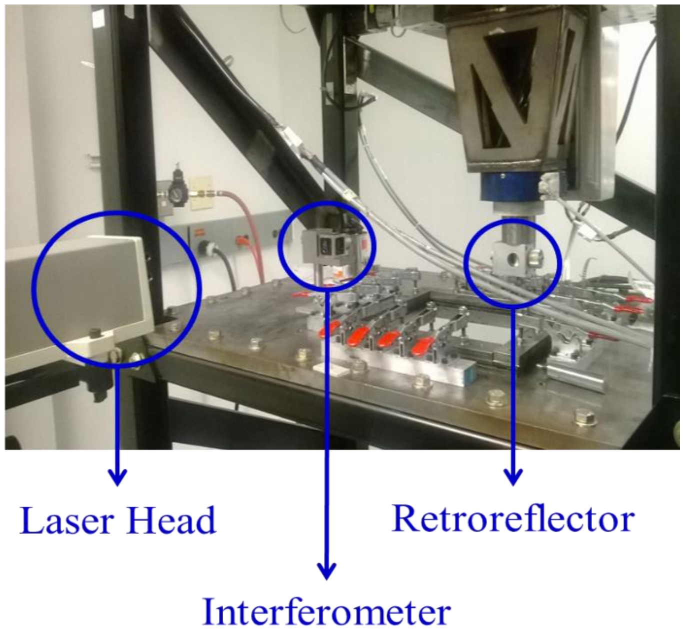


**Figure 2.24 HP5529 dynamic calibration system in the DSIF prototype machine (by courtesy of Zixuan Zhang at Northwestern University)**

## 2.3. Algorithm Design and Implementation

The essence of the compliance compensation algorithm and force control algorithm are quite similar. Both algorithms utilize the corresponding force signals, compute the respective position corrections and modify the tool command positions accordingly, either to compensate the compliance value or maintain a stable contact force. The only difference is the particular control strategy. Due to the similarity between the compliance compensation and force control algorithms, the following subsections will mainly focus on the latter one, while the explanation for the compliance compensation algorithm will be briefly covered at the end.

Before the detailed establishment of the proposed algorithm, a local tool coordinate system is first introduced to enable the position/force hybrid control of the supporting tool. Note that in the DSIF process, the addition of the supporting tool introduces a squeezing contact force and the opportunity to modify the local stress triaxiality. The forming force on the supporting tool can be decomposed into two components: the friction force, $F_t$, aligned with the tool motion, and the normal contact force, $F_n$, aligned with the local surface normal, $\boldsymbol{n}$, i.e., the vector between the centers of two tools as shown in Figure 2.25. In the global coordinate system, $F_n$ can be further decomposed into a horizontal component $F_{nxy}$ in the X-Y plane, and a vertical component $F_{nz}$ along the Z axis. In the following work, $F_{nxy}$ is set as the control variable while it is assumed that $F_{nz}$ will change about $F_{nxy}$ as $F_{nz} = F_{nxy}\, cot\varphi$, where $\varphi$ is the local wall angle. In other words, the DOF being force-controlled is the horizontal component of the normal force vector acting on the supporting tool. On the other hand, the compliance compensation in the forming tool will be accomplished in the global coordinate system, i.e., X-Y-Z system, hence, no coordinate transformation is needed.

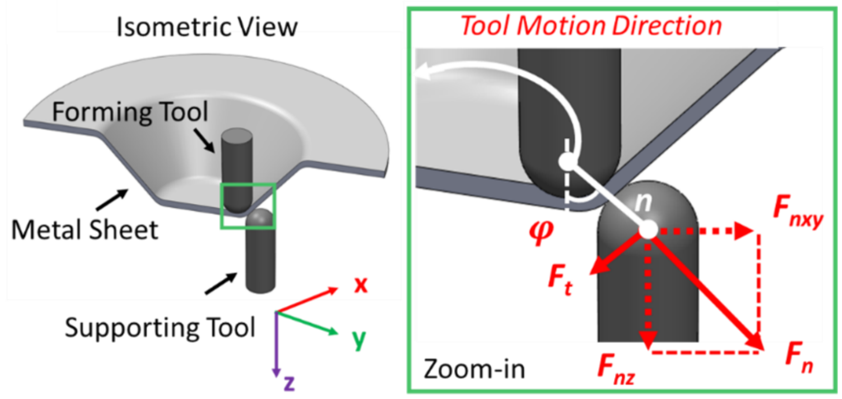


**Figure 2.25 Force decomposition on the supporting tool**

### *2.3.1. General Control Scheme – Explicit Force Control*

While various force control algorithms have been used in manufacturing processes including grinding, polishing, and milling as reviewed in Section 2.1.5, their implementation in incremental forming is still in its nascent stages. One reason is the difficulty associated with the needed modifications of conventional industrial motion controllers, since it is common to modify an existing CNC milling machine for IF processes. Standard CNC controllers tend to restrict the user’s ability to directly change the motor velocity or torque signals and only allows access to the position loop of the motion signal, which makes some of the force control schemes inapplicable. For example, both the stiffness control and the damping control reviewed in Section 2.1.5 will require a direct manipulation at the joint torque level. To overcome this constraint, we propose a generalized control scheme that utilizes explicit (or sometimes termed external) force control (Dégoulange and Dauchez, 1994; Volpe and Khosla, 1993), shown in Figure 2.26.

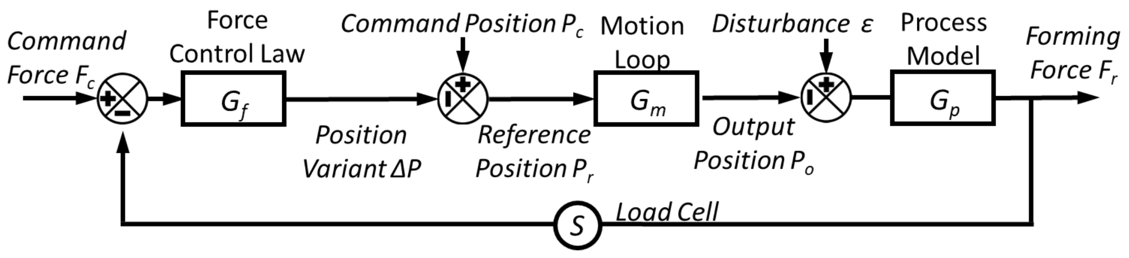


**Figure 2.26 Generalized block diagram of explicit force control for DSIF**

In explicit force control, a deviation of the tool position, *ΔP*, is first calculated by the force control law, $G_f$, from the command force, $F_c$, and the force feedback, $F_r$. Then, *ΔP* is added to the commanded position, $P_c$, leading to the new reference position, $P_r$, which is subsequently used as the commanded input to the conventional servo motion loop. In effect, the forming forces will vary in response to the position change of the supporting tool according to the process model $G_p$, and the difference between $F_r$ and $F_c$ will be minimized.

It should be re-emphasized that the only DOF force being controlled in Figure 2.26 is the horizontal normal direction, which means $F_r$ is set to be $F_{nxy}$ in our implementation. To do so, the X- and Y-force signals of the supporting tool must be projected onto the local normal to calculate $F_{nxy}$, and similarly *ΔP* must be decomposed into *ΔX* and *ΔY* before being subtracted from the motion control loop, i.e.:

$$F_{nxy} = \frac{L_x}{\sqrt{{L_x}^2 + {L_y}^2}} F_x + \frac{L_y}{\sqrt{{L_x}^2 + {L_y}^2}} F_y \tag{2-2}$$

$$\Delta X = \frac{L_x}{\sqrt{{L_x}^2 + {L_y}^2}} \Delta P, \;\; \Delta Y = \frac{L_y}{\sqrt{{L_x}^2 + {L_y}^2}} \Delta P \tag{2-3}$$

where $L_x$ and $L_y$ are the tool center distances defined in Figure 2.27. A principal characteristic of this explicit force control scheme is that the force control signal only acts as a modifier to the commanded position signal and, consequently, does not require direct interference with the inner motion control loop, $G_m$. The only modification to the original servo control system is the addition of a relative deviation (or offset) from the current supporting tool position, which can be performed by most motion controllers.

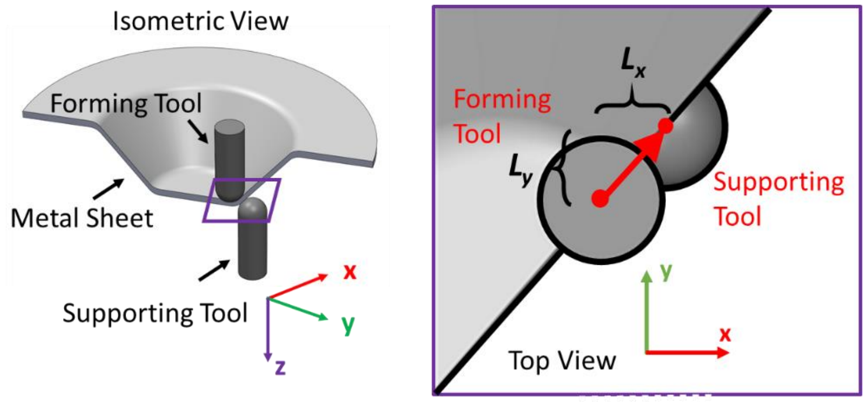


**Figure 2.27 The definition of tool center distance composing via *Lx* and *Ly***

For the compliance compensation, a similar control scheme can be utilized, as shown in Figure 2.28. The position variant, $\Delta P$, is calculated directly from the forming force signal, $F_r$, according to the compliance compensation law $G_c$, in which $G_c$ is usually a proportional term with some filters. Then, similar to the force control algorithm, $\Delta P$ is added to the commanded position, $P_c$, leading to the new reference position, $P_r$, which is subsequently used as the commanded input to the conventional servo motion loop. In this compliance compensation algorithm, the DOFs being compensated in Figure 2.28 are the X and Y axes in the global axes, instead of the tool normal direction as stated for the contact force control.

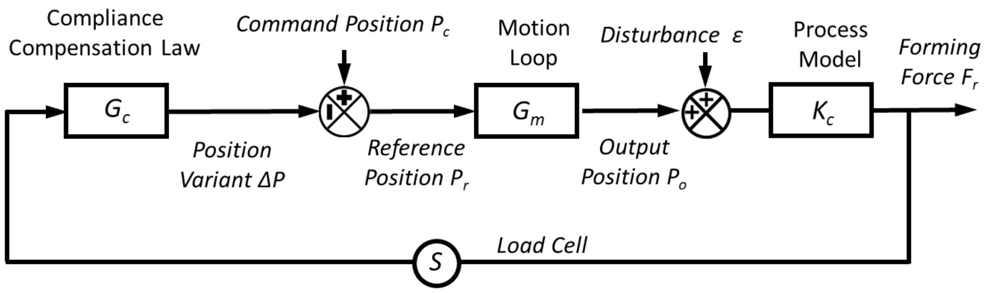


**Figure 2.28 Block diagram for compliance compensation for DSIF**

### *2.3.2. DSIF Process Contact Condition Modeling*

To model the forming process and derive the relationship between the changes of the input position and output forming forces, the process model for the local contact region, $G_p$, will be carefully examined (Figure 2.29). In this study, we assume that the tool-contact interaction can be approximated using constant stiffness springs. Additionally, the DSIF process is generally quasi-static in nature considering the low forming speed (~ 5 *mm*/*s*). More specifically, the horizontal tool velocity perpendicular to the tool motion direction, which lies in the same direction as $F_{nxy}$, is less than 5% of the total velocity in magnitude. Hence, the associated dynamic effects can be neglected even if the tool speed is significantly increased. Under these assumptions, $G_p$ can be abstracted as a serial spring system (Figure 2.29) with an overall contact stiffness, $K_c$.

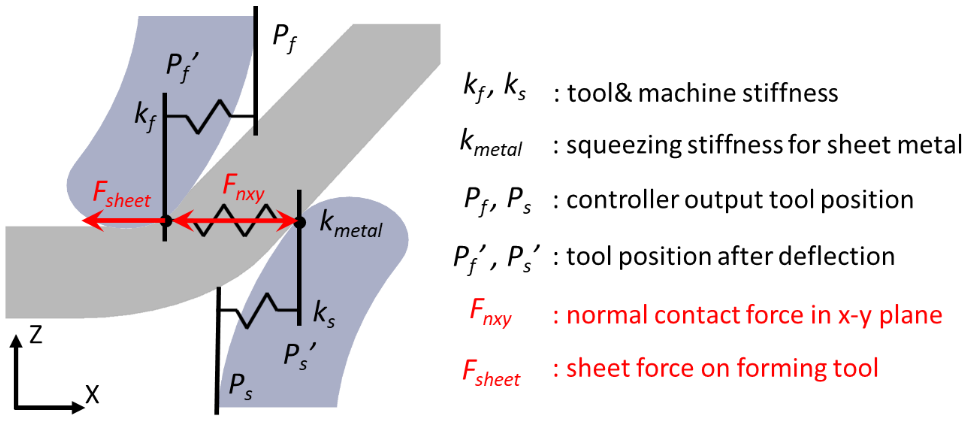

$P_f$
$P_f'$
$k_f$
$F_{sheet}$
$F_{nxy}$
$k_{metal}$
$k_s$
$P_s'$
$P_s$
Z
X
$k_f$, $k_s$ : tool& machine stiffness
$k_{metal}$ : squeezing stiffness for sheet metal
$P_f$, $P_s$ : controller output tool position
$P_f'$, $P_s'$ : tool position after deflection
$F_{nxy}$ : normal contact force in x-y plane
$F_{sheet}$ : sheet force on forming tool

**Figure 2.29 Local spring contact system representing the process model**

To derive a detailed representation of $K_c$, the following spring relations can be readily observed from Figure 2.29:

$$F_{nxy} \cong k_{metal}\left(T_c - P_s' + P_f'\right), \tag{2-4}$$

$$F_s = k_s(P_s - P_s') \quad \text{and} \quad F_s = F_{nxy}, \tag{2-5}$$

$$F_f = k_f(P_f - P_f') \quad \text{and} \quad F_f = F_{nxy} + F_{sheet}, \tag{2-6}$$

where the subscripts $s$ and $f$ correspond to the supporting and forming tools, respectively. $P_s$ and $P_f$ are defined as tool positions in the horizontal normal direction, denoted as the X axis in Figure 2.29, so that the normal contact force will decrease if $P_f$ decreases or $P_s$ increases due to an enlarged tool gap. The sheet force, $F_{sheet}$, is related to the forming tool's independent interaction with the sheet, as one could imagine that there would be an applied force even when the supporting tool loses contact. Hence, in Equation 2-6, the resultant forming force, $F_f$, is additively decomposed into a force caused by the supporting tool ($F_n$) and the force applied by the sheet ($F_{sheet}$). While $F_{sheet}$ may vary due to the structural stiffness, material yield stress, etc., this analysis assumes that $F_{sheet}$ is independent on the tools' deflections in the local contact region. $k_f$ and $k_s$ are the tool stiffness for the forming tool and supporting tool respectively, which are the reciprocal of the measured values shown in Figure 2.21. On the other hand, $k_{metal}$ is the squeezing stiffness for the sheet metal, which depends on the sheet metal material and tool radius. Note that in this analysis $T_c$ denotes the horizontal sheet thickness between the tool contact positions. Itis the minimal horizontal distance between the upper and lower surfaces of the workpiece, i.e., the horizontal component of the total current sheet thickness.

To solve for the overall contact stiffness, $K_c$, the intermediate variables $P_s'$, $P_f'$, $F_s$ and $F_f$ can be eliminated from Equations 2-4 to Equation 2-6 to obtain:

$$F_n = K_c(\varepsilon - P_s)\ , \tag{2-7}$$

$$K_c = 1/\left(\frac{1}{k_f} + \frac{1}{k_{metal}} + \frac{1}{k_s}\right), \tag{2-8}$$

$$\varepsilon = -F_{sheet}/k_f + P_f + T_c\ \ , \tag{2-9}$$

As shown by Equation 2-9, ε is influenced by the sheet force, forming tool position, and sheet thickness, which explains why it could be quite challenging to pre-compensate it in typically position-controlled DSIF. With the process model determined by Equation 2-8, the control system, shown in Figure 2.26, can now be fully defined as:

$$F_r = \frac{G_f G_m K_c}{1+G_f G_m K_c} F_c - \frac{G_m K_c}{1+G_f G_m K_c} P_c + \frac{K_c}{1+G_f G_m K_c} \varepsilon, \tag{2-10}$$

$$\Delta P = G_f(F_c - F_r) \tag{2-11}$$

### *2.3.3. Control Strategy Design and Stability Analysis*

From the system of equation defined by Equation 2-10, a simple integral control scheme is proposed for $G_f$, in part, to easily implement the control scheme on most standard CNC controllers. Additionally, the advantages of integral control are its low-pass nature and, more importantly, zero steady-state errors even under large disturbances (caused by $\varepsilon$ and $P_c$) for a constant desired contact force, $F_c$. While alternative control schemes could have been used, such as proportional-integral control, the addition of the proportional term is discarded in this study due to its tendency to cause overshoot or system instability, which will be shown later in the section, while the noisy nature of the force signal makes derivative control impractical.

Before determining a suitable value of the integral gain, $K_i$, the motion loop, $G_m$, of the physical machine controller must be either made available by the manufacturer, or as in our case, physically identified. The system position response is recorded for a given step excitation signal. The motor system is then identified as a third-order system. The measured and simulated response signals along with the command signal are shown in Figure 2.30.

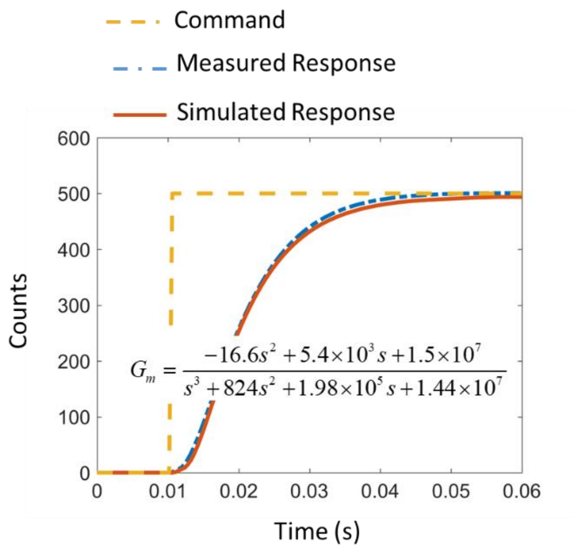


**Figure 2.30 Measured and simulated system responses**

Given the $G_m$, the simulated step responses for various values of $K_i$ are graphed corresponding to the whole force control system (Equation 2-10), as shown in Figure 2.31. The selection of the specific value of the integral gain, $K_i$, depends on the contact stiffness, $K_c$. In our specific setup, a $K_i$ of 24 $s^{-1}/K_c$ was found to be a good choice. It is also noted that the addition of the proportional term (green curve) will cause an oscillatory response with even a slightly larger 90% rise time. As mentioned in Section 2.2.1, the DSIF prototype machine, in which the algorithm will be implemented, utilizes a serial XYZ configuration. The $K_c$ in the X-Y plane is measured as 0.45±0.05 $N/\mu m$ resulting in a $K_i$ of 53 $\mu m N^{-1} s^{-1}$, i.e., 24 $s^{-1}/K_c$. On the other hand, $K_c$ in the z-direction (7.2 ± 0.6 $N/\mu m$) is one order of magnitude higher than that in the X-Y direction. To account for this difference, different $K_i$ values would be required along the different axes if $F_{nz}$ and $F_{nxy}$ are controlled simultaneously. However, in the current case, the designed algorithm only directly controls $F_{nxy}$, while $F_{nz}$ will change proportionally to $F_{nxy}$ as assumed. This implementation is simple and, as it will be seen, demonstrates sufficient accuracy in the experimental verification part presented in Section 2.4.

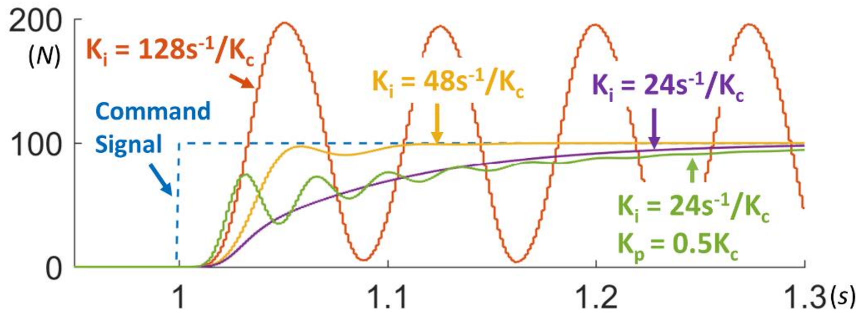


**Figure 2.31 Simulated response of a force step for different integral gains**

The system performance is also simulated for a pure promotional controller (i.e., $K_i$ = 0) as $G_f$ as shown in Figure 2.32. It shows that while the larger $K_p$ causes more overshoot and/or system instability, the smaller $K_p$ generally leads to a more substantial steady-state error. Therefore, a simple integral term is selected as the $G_f$. Note that due to the noisy nature of the force signal, the $G_f$ implemented will include some low-pass filters as well as some range limiter for safety purposes to be introduced later in Section 2.3.4.

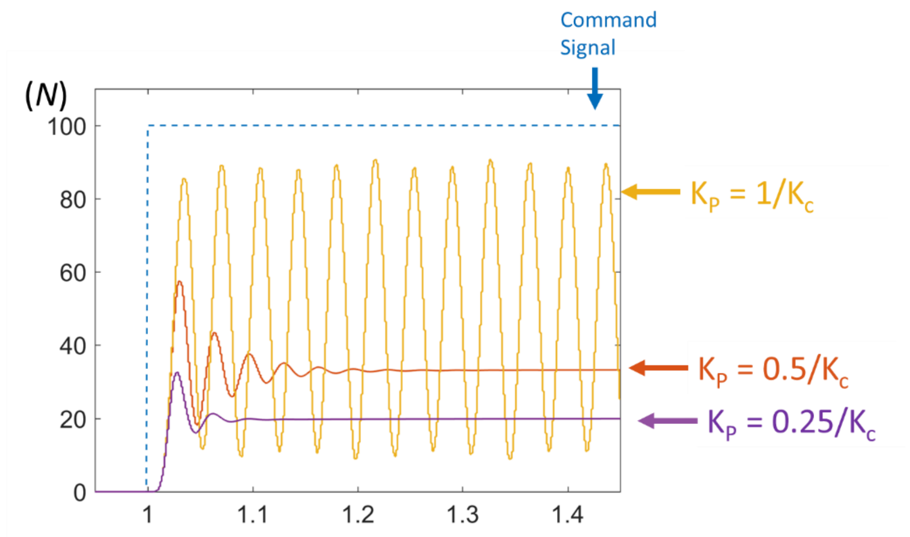


**Figure 2.32 Simulated response of a force step for different proportional gains**

For the compliance compensation algorithm, $G_f$ can be set as a general proportional gain $K_p$ and equal to $1/k_f$, and thus the position variant defined in Figure 2.28, $\Delta P$, will be equal to $F_r /k_f$, which corresponds to the deflection of the forming tool under the forming force. Similar to the force control algorithm, a low pass filter and range limiter are also implemented for robustness and safety purposes. Note that there also exists a certain stability bound for the choice of $K_p$. From Figure 2.28, the system response of a compliance compensated system can be written as:

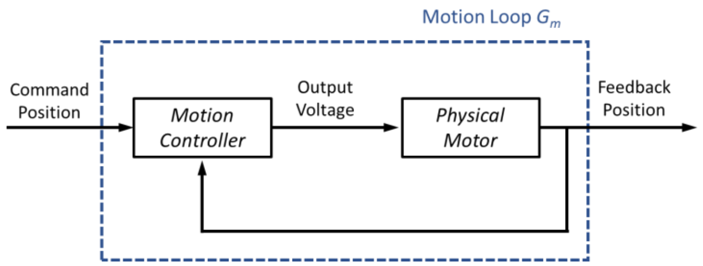
$$F_r = \frac{G_m G_p}{1 - G_m G_p K_p} P_c \tag{2-12}$$

where the forming force $F_r$ is represented by a function of command position $P_c$. The process model $G_p$ can be represented as contact stiffness $K_c$, while $G_m$ is characterized as a third order system as shown in Figure 2.30. The system's stability can be determined by solving the pole values in Equation 2-12. It is derived that one necessary condition to guarantee that all the poles are located in the left plane is that $K_p \leq 1/K_c$, i.e., the implemented proportional gain must be lower than the overall compliance of the system. Note that this condition can be easily met if the compensation algorithm is only implemented on the forming tool, where $K_p$ is set to be the reciprocal of the stiffness of the forming tool ($1/k_f$) while the overall compliance of the system ($1/K_c$) is equal to $1/k_f + 1/k_{metal} + 1/k_s$ according to Equation 2-8. However, if both the forming and the supporting tools need to be compensated, the equivalent $K_p$ needs to be set to be equal to the sum of the measured values of $1/k_f$ and $1/k_s$. Considering potential tool stiffness measurement errors and the large value of $k_{metal}$, the implemented $\mathrm{K_p}$ ($1/\overline{k_f} + 1/\overline{k_f}$) might become larger than $K_c$ ($1/k_f + 1/k_{metal} + 1/k_s$), where the upper bar in the bracket denotes measured values that often contain some variations. Note that this potential instability also rationalizes the choice of the implementation of the force control loop and of the compliance compensation loop on different tools.

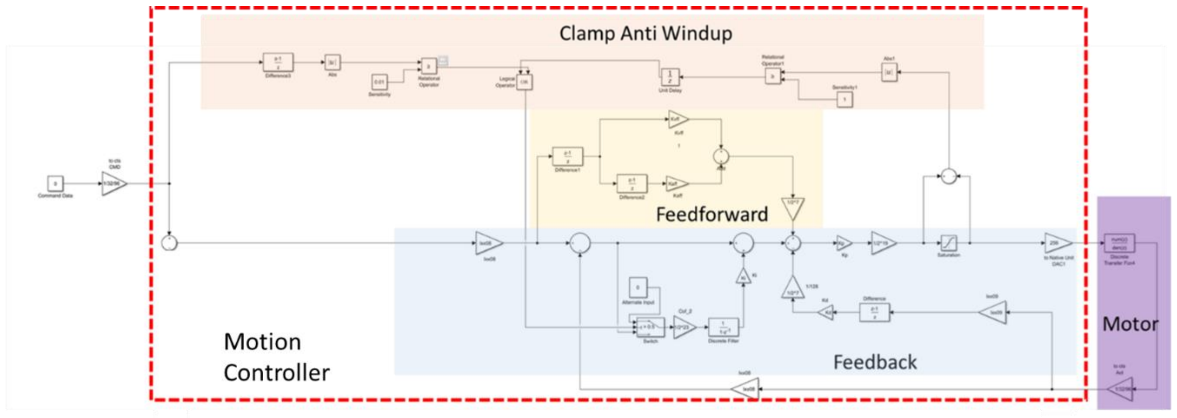
### ***2.3.4. System Simulation and Implementation***

To verify the choice of the control law and determine the detailed configurations of the low-pass filters, a simulator has been developed in MATLAB Simulink® as a discrete time model to simulate the behavior of the DSIF control system, i.e., the motion loop model $G_m$ in Figure 2.26 and Figure 2.28. Considering that the motion loop model $G_m$ is in fact composed of the dynamics behaviors from two parts, i.e., the motion controller and the physical motor, the simulator is designed to model both of their actions. The motion controller portion of the simulator takes the command position as input and outputs the voltage signal to the motor, while the motor portion of the simulator receives the voltage signal and returns the feedback position to the motion controller part, as shown in Figure 2.33(a). However, from the user point of view, the simulator is simply a black box, the input of which is the command position signal and the output is the motor position. Compared to the simple third-order model in Section 2.3.4, the simulator better accounts for the system nonlinearity and offers a more realistic test environment for the design of the force control algorithms, since the complex motion algorithm inside the motion controller is explicitly constructed.

To establish the simulation model, the motion controller portion is established based on the Delta Tau PMAC system manual (PMAC, 2008) while the motor actuators are physically characterized as a third order system. The whole model is a discrete time-based model with a sampling frequency of 2.2 *kHz*, the same as the system frequency in the Turbo PMAC controller. Note here that the motor actuators in the simulation model include not only the motor and the associated inertia but also the motor drivers and the encoders. The whole simulation model can be seen in Figure 2.33(b), with the detailed block diagram shown in Appendix B. It can be seen that the clamp anti-windup scheme and the feedforward terms have been implemented in the diagram, both of which might cause system nonlinearity in response to certain discountinous command position signals as shown later.



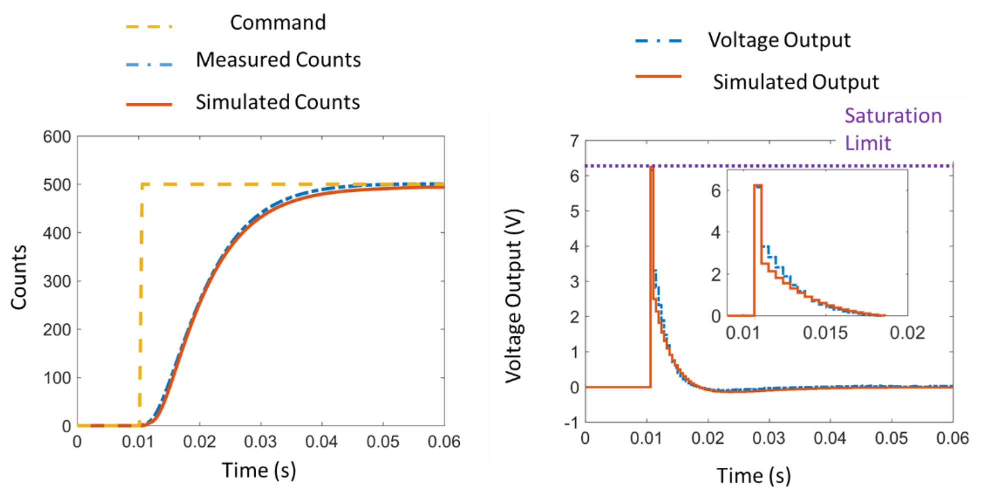
**(a)**



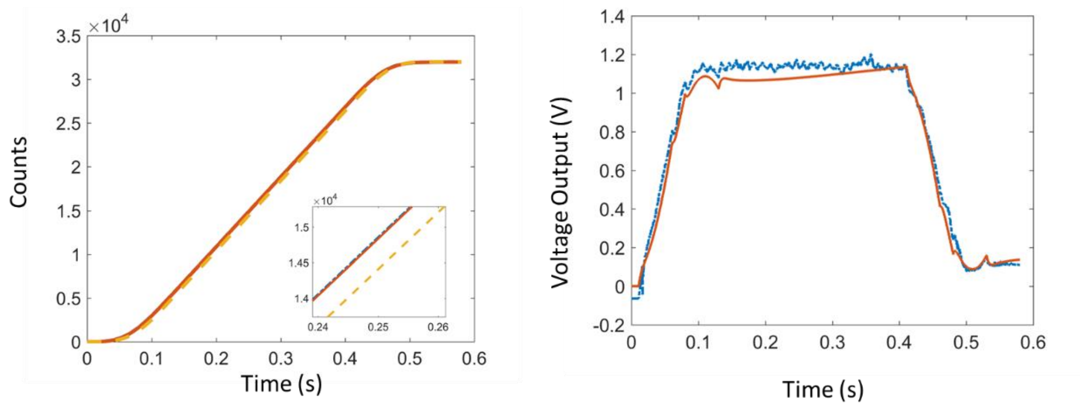
**(b)**

**Figure 2.33 (a) Overall structure of the simulator, (b) detailed block diagram of the simulation algorithm**

The established simulation model is then validated via step and ramp excitation command position signals for the X axis motors. The simulated results are compared with the experimental ones in Figure 2.34. Both the position feedback, as the output of the motor, and the voltage signal, as the output from the motion controller, are compared. A reasonable agreement can be observed between the experimental and the simulated signals.



**(a)**



**(b)**

**Figure 2.34 Measured and simulated system responses for the simulation model (a) a step signal, (b) a ramp excitation**

It should be noted that for the Heaviside step function *H(t₀)*, the velocity/acceleration of the command signal is equal to zero most of the time. However, since the step signal *H(t₀)* is discontinuous, the velocity/acceleration terms will become nearly infinite (or extremely large in the discrete analysis) at transition points when *t* is equal to *t₀* (which is 0.01 *s* in Figure 2.34(a)).

Thus a saturated voltage output signal will be triggered, as shown in the voltage output in Figure 2.34(a).

With the developed simulation model of the behavior of $G_m$, the whole control loop shown in Figure 2.26 can now be established and verified. The process model is assumed as a contact stiffness $K_c$ as presented in Section 2.3.2 while the force control algorithm is implemented as a cascaded loop in the same MATLAB Simulink environment as shown in Figure 2.35. The system model assumes a constant contact stiffness $K_c$ without any coordinate transformation. The force control loop runs at an update frequency of 440 *Hz* which is slower than the position servo loop execution frequency (2.2 *kHz*). The $F_r$ force is calculated based on a measured $K_c$ (0.45 *N/µm* as mentioned in Section 2.3.3) with respect to the output position $P_o$ with added white noise (signal-to-noise ratio of 1%), to simulate the real force feedback signal. The force error is then calculated and smoothed using a low-pass FIR filter with a cut-off frequency of 40 *Hz*, chosen to be slightly higher than the bandwidth of the position servo loop identified before (30 *Hz*). Note that a FIR filter with a high cut-off frequency cannot effectively smooth the low-frequency noise, while if the cut off frequency is lower than the position servo loop bandwidth, the filter could potentially decrease the system's response speed. Finally, using the filtered force, the position variations are computed by the integral controller. Two saturation-type limiters and one rate limiter are also incorporated for protection purposes along with a dead zone limiter to make the algorithm less sensitive to any remaining noise in the force signal.

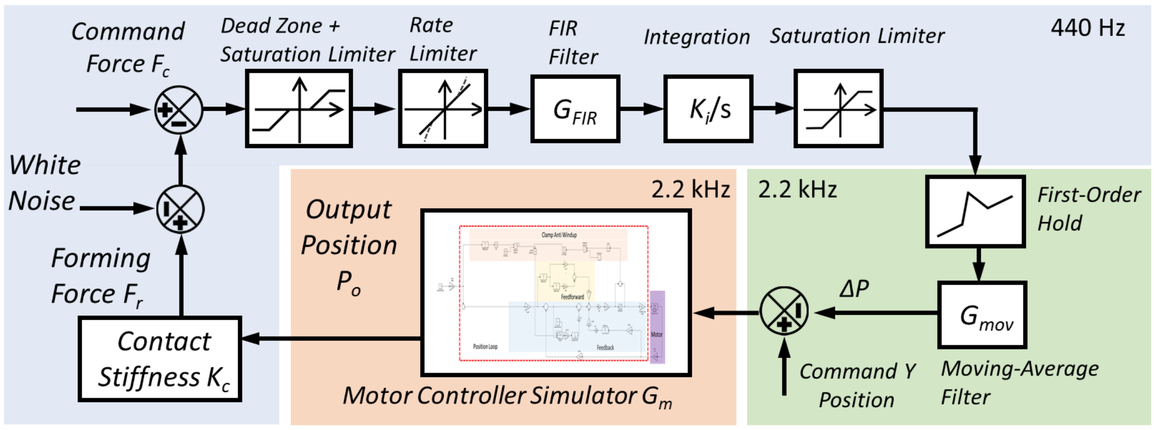


**Figure 2.35 Force control algorithm design**

Since the force control algorithm is running at a slower rate than the position servo control loop, a zero-order hold is usually required to input the calculated position variant into the motor controller. However, the zero-order hold cannot grantee the continuity of the input signal, and thus the command position signal will become a series of step input signals at a frequency of 440 *Hz* and trigger unwanted voltage output saturation and even system oscillations. Instead, the implemented first-order hold with a moving average filter can ensure the smoothness of the input position command signal, and thus avoid possible instability. It should be noted that the rate limiter and the FIR filter defined in the force control algorithm itself (the light blue region in Figure 2.34) also help to improve the smoothness of the signal and reduce any possible discontinuity. A comparison between the raw command position signal (without any filter/interpolation), the intermediate signal (with the rate limiter and FIR filter) and the final signal (with the rate limiter, FIR filter, the first-order hold and the moving average filter) is shown in Figure 2.36.

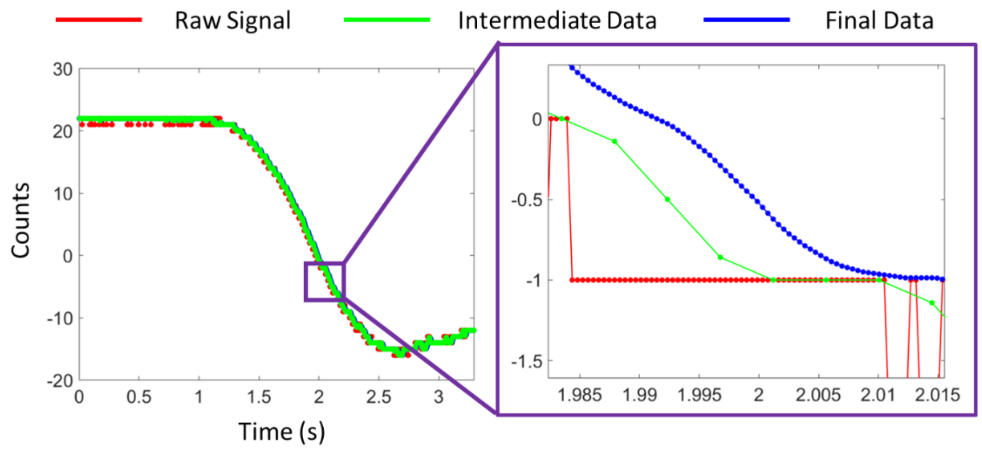


**Figure 2.36 The command position signal comparison among the raw/intermediate-filtered/final-filtered data**

With all the filter and safety limits correctly designed, the algorithm shown in Figure 2.35 is expanded to include the coordinate transformation procedure since the $F_r$ signal is not directly available in real applications. Instead of directly reading the forming force $F_r$ as stated in Figure 2.35, the $F_x$ and $F_y$ force components are first acquired and then composed into the $F_{nxy}$ component in the normal direction. Similarly, the calculated position variant $\Delta P$ also needs to be decomposed back into global coordinates so that the command positions of the X and Y axes can be successfully updated. The detailed force composition and motion decomposition equations are given by Equations 2-2 and 2-3. The whole implementation of the proposed explicit force controller is shown in Figure 2.37 with the final filter stage omitted for simplicity.

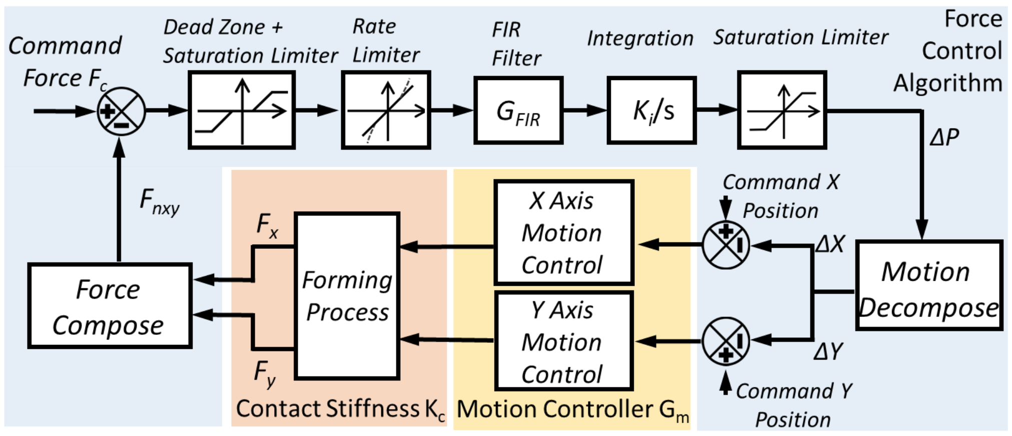


**Figure 2.37 Implementation of the proposed explicit force controller**

For the compliance compensation, a similar algorithm is implemented with the same design as the limiters/filters. The first-order hold and the moving average filter are also utilized for compliance compensation. The overall implementation of the compliance compensation algorithm can be seen in Figure 2.38. Note that the compliance compensation system utilizes a negative proportional gain term instead of the integration gain. Moreover, there is no force composition or motion decomposition involved in compliance compensation.

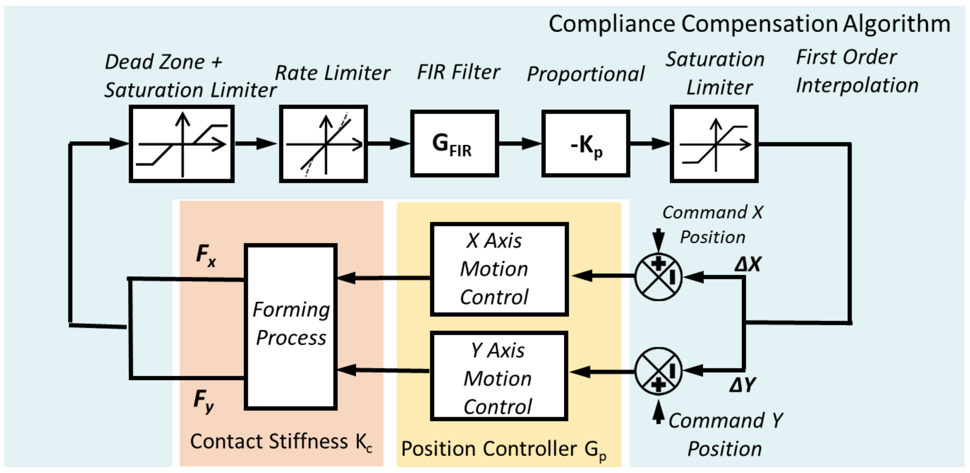
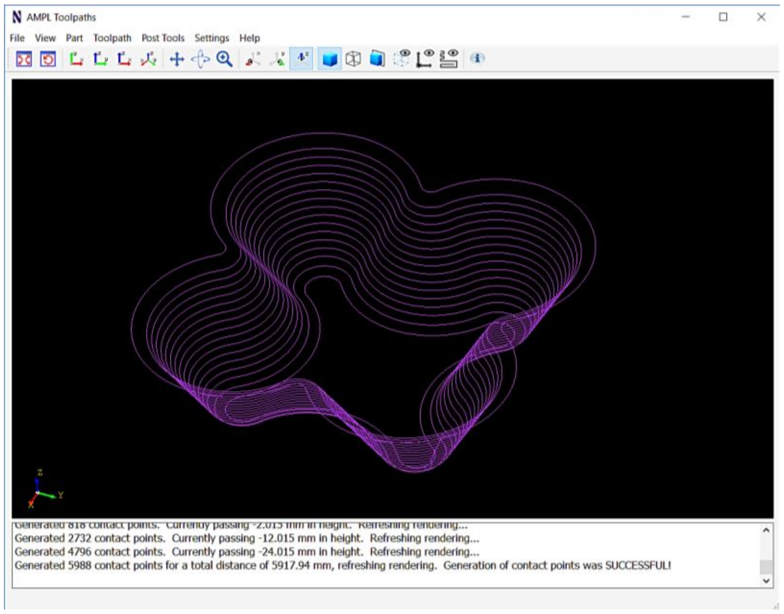


**Figure 2.38 Implementation of the proposed compliance compensation algorithm**

Both algorithms have been implemented as PLC (Programmable Logic Controller) programs that operate synchronously with the motion program. To accelerate program execution speed, the whole PLC program is written in a compiled format where all the variables memory is pre-allocated and only fixed point mathematic operations are allowed. With the complied PLC successfully implemented, a full cycle of force control can be finished within 150 $\mu s$ without interfering with the servo cycle (450 $\mu s$). The full force control and compliance compensation PLCC program can be found in Appendix B.

## 2.4. Experimental Validation

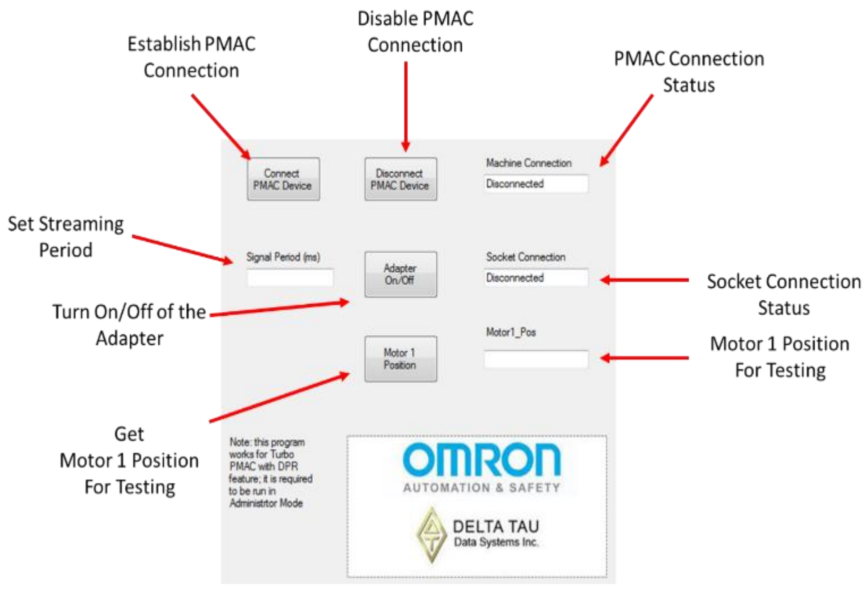
To examine the performance of the developed algorithms, verification experiments were conducted to assess the generality and robustness of the proposed scheme and its assumptions. The toolpaths utilized in the experiments were generated using a custom in-house codebase, nicknamed AMPL Toolpath (Moser, 2018), that has been built upon the C++ Open Cascade library, the interface of which is shown in Figure 2.39. The AMPL Toolpath software takes the input CAD model as an IGS or STA file and outputs the tool contact points with the tool normal directions. The tool tip positions can then be generated based on tool contact position, normal direction, sheet thickness and the tool radius. The detailed schematics of tool tip generation can be found in the work of Malhotra et al. (Malhotra et al., 2011) and will not be restated here.



```
Generated 818 contact points.  Currently passing -2.015 mm in height.  Refreshing rendering...
Generated 2732 contact points.  Currently passing -12.015 mm in height.  Refreshing rendering...
Generated 4796 contact points.  Currently passing -24.015 mm in height.  Refreshing rendering...
Generated 5988 contact points for a total distance of 5917.94 mm, refreshing rendering.  Generation of contact points was SUCCESSFUL!
```

**Figure 2.39 The user interface of the AMPL Toolpaths Software with a generated toolpath (by courtesy of Newell Moser at Northwestern University)**

During the experiments, the forming forces and tool positions are simultaneously gathered and stored on the host computer for later analysis via an in-house developed MATLAB or C++ application, as shown in Figure 2.40. The MATLAB/C++ application utilizes a COM interference called PCOMServer to communicate with the Turbo PMAC. A unique functionality called Dual Port Ram (DPR) is utilized in the PMAC, where both the PMAC and the host computer (the PC where the signals are gathered) can write/read at the same time (PMAC, 2008; Turbo, 2008). In the communication protocol, the Turbo PMAC continuously uploads the data to the host computer, including the current tool position, the forming forces, and position compensation values in each axis. Note that the X, Y, Z axes of the supporting tool are denoted as U, V, W in Turbo PMAC for easy implementation.



Machine Connection: Disconnected

Socket Connection: Disconnected

Motor1_Pos

Note: this program works for Turbo PMAC with DPR feature; it is required to be run in Administrator Mode

**Figure 2.40 The user interface of the C++ application to communicate with the Turbo PMAC**

In the following sections, the experimental validation will be conducted for contact force control (Sections 2.4.1 and 2.4.2) and compliance compensation (2.4.3). Especially, to verify the algorithm's generality, the force control algorithm has been tested with different materials, tool diameters, and command force levels as stated in Section 2.4.2.

### *2.4.1. Stabilized Contact Condition and Enhanced Material Formality*

As a first test for the force control algorithm, a 65$^o$ cone is formed with 1 *mm* thick aluminum AA1100-O sheet with a command force of 120 *N*. The tool speed is 5 *mm/s* with 9 *mm* diameter tools. The toolpath is a spiral toolpath with a 0.5 *mm* incremental depth, which represents the distance between two subsequent toolpath loops in the Z direction (Malhotra et al., 2011). The side view of the physical formed part is shown in Figure 2.41 with the $F_{nxy}$ force contour. The early loss of contact is prevented in the force-controlled case. Moreover, the force-controlled case also maintains much more stable contact force conditions while in the uncontrolled case an uneven force distribution and loss of contact line (purple dashed-dotted line in Figure 2.41) occur.

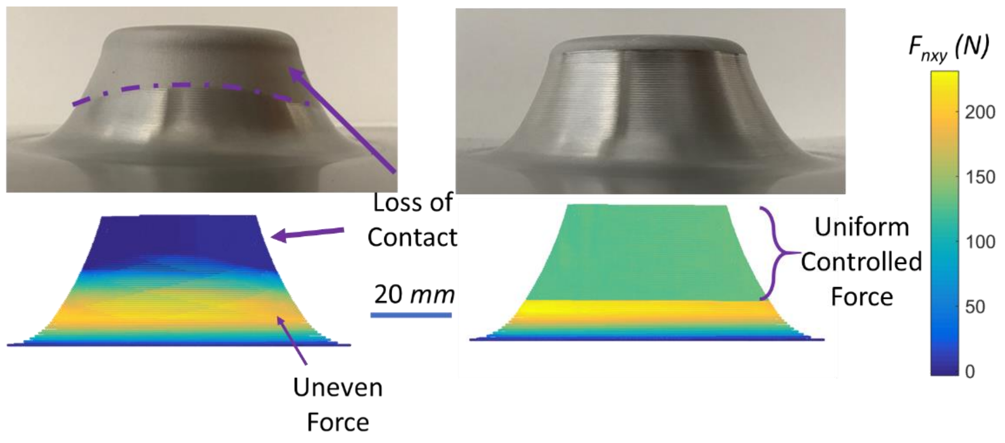


**Figure 2.41 Comparison between the position-controlled (left) and force-controlled (right) cones**

It should be noted that the level of the command contact force will influence the success of the physical experiment. For example, if the command force is equal to 200 *N*, the formation of aluminum chips is observed, and the surface integrity of the formed part is damaged, as shown in Figure 2.42. This principle of the chip formation is similar to the cutting process, i.e., the high shear stress caused by the local tangential contact force exceeds the ultimate shear strength and thus breaks the material. To avoid this shearing effect, the contact force needs to be maintained below a certain level for a given sheet material and a given tool diameter.

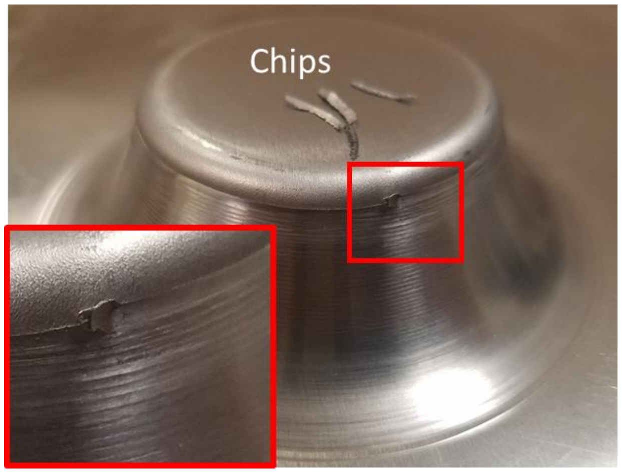


**Figure 2.42 Shearing effect with an excessive forming force**

To further test the influence of force control on formability, a second funnel shape geometry with an entry wall angle of 65°, shown in Figure 2.43, was formed with a 9 *mm* diameter tool and 1 *mm* thick AA 5757-O sheet. Note that the yield strength of AA5754-O (~ 122 *MPa*) is considerably higher than that of AA1100-O (~ 20 *MPa*), and thus a larger contact force is allowable in the process. The other process parameters are the same as in the first cone experiment. When using just position control, the supporting tool progressively loses contact with the sheet leading to failure at a depth of 19.0 *mm*. Subsequently, the part was reformed with force control activated after the initial fillet region in order to maintain $F_{nxy}$ equal to 200 *N*, which increased the formability depth to 22.1 *mm* (15% deeper). This is likely due to the squeezing pressure through the thickness that is maintained longer under force control (Lu et al., 2015).

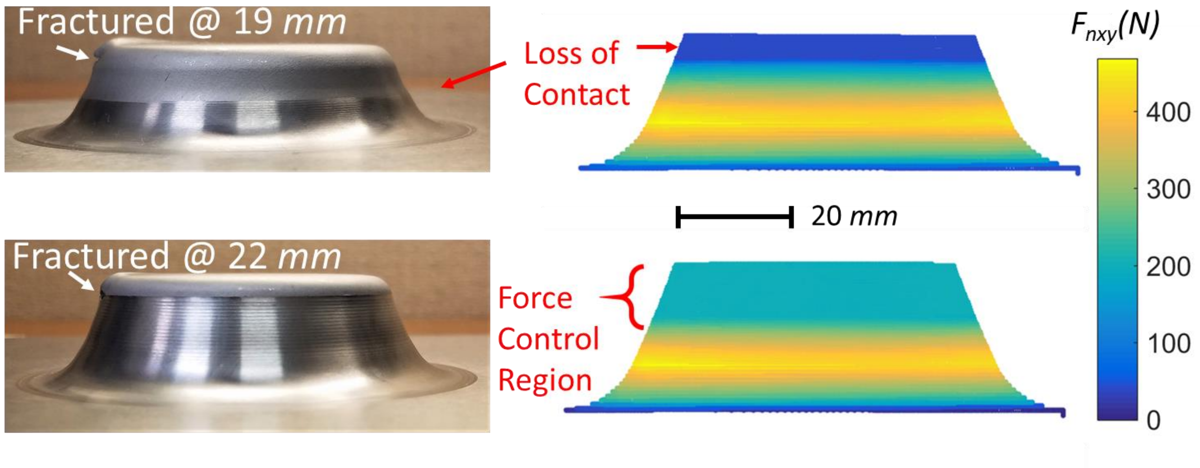


**Figure 2.43 Increased forming depth when using force control**

### *2.4.2. Generality and Robustness of the Contact Force Control Algorithms*

As a rapid prototyping or a low-volume production technology, DSIF is expected to handle a variety of situations, such as different geometries, tool diameters, materials, and toolpath strategies. Additional three experimental cases, A through C, were conducted besides the above-mentioned two to include as many variations as possible. The process parameters used are listed in Table 2.1.

**Table 2.1 The experimental parameter summary**

| **Test Case** | **A** | **B** | **C** |
|---|---|---|---|
| Shape | Cone | Fin | Pyramid |
| Aluminum Alloy | 2024-T3 | 1100-O | 5754-O |
| Strategy | ADSIF | DSIF | DSIF |
| Sheet Thickness | 0.5 *mm* | 1.0 *mm* | 1.0 *mm* |
| Tool Radius | 2.5 *mm* | 5.0 *mm* | 5.0 *mm* |
| Inc. Depth | 0.1 *mm* | 0.5 *mm* | 0.2 *mm* |
| Command $F_{nxy}$ | 180 *N* | 100 *N* | 100-200 *N* |
| Wall Angle $\varphi$ | 40° | 60° | 45° |

| | | | |
|---|---|---|---|
| Tool Speed | $5\ mm/s$ | $5\ mm/s$ | $5\ mm/s$ |

First, Accumulative-DSIF (ADSIF), a variant of DSIF where the tools remain in the plane of the initial sheet and spiral outwards starting from the center (Malhotra et al., 2012) is considered. ADSIF was used to form a cone with and without force control as Case A (Figure 2.44). While ADSIF has shown great potential to reduce geometric errors in the final part, it is highly sensitive to any tool misalignments or motion errors which lead to large cumulative errors due to rigid body motion. This effect is clearly seen in Figures. 2.44(b)–(c) where a truncated 40° cone was formed using position control resulting in a tilted base and geometric errors of near 2.0 *mm.* On the other hand, the formed cone that utilized force control was able to maintain an axisymmetric force history by effectively compensating for any intrinsic misalignment or motion errors in the process. When using force control, the contact forces varied less than $\pm 10\ N$, compared to a variation of 80 N in the case of position control, and the geometric error was below 1.0 *mm*. A detailed discussion of the ADSIF technology will also be presented in Chapter 3 along with the ADSIF toolpath optimization technology.

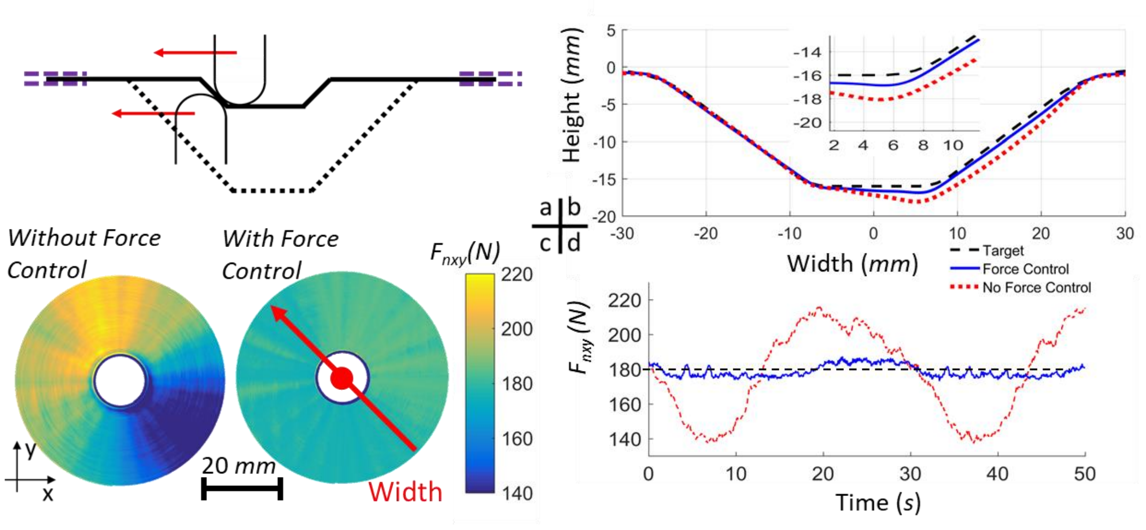


**Figure 2.44 Formed cone parts a) position controlled with complex loss of contact region, b) force controlled with stable contact**

To demonstrate the applicability of force control in DSIF for general geometries with varying geometric curvatures, a fin part was formed for Case B, which is designed as a complex shape which includes distinct common geometry features, such as sharp corners, straight walls, and a concave curve, as illustrated in Figure 2.45. This complex shape is a challenge for position control due to the difficulty of predicting sheet thinning (i.e., tool gap needed to maintain the contact), which in addition to machine compliance, often leads to substantial force fluctuations around the corners and the loss of tool contact (Figure 2.46(a)). When using force control, contact was successfully maintained without detailed attention to the influences imposed by the complex geometry Figure 2.46(b).

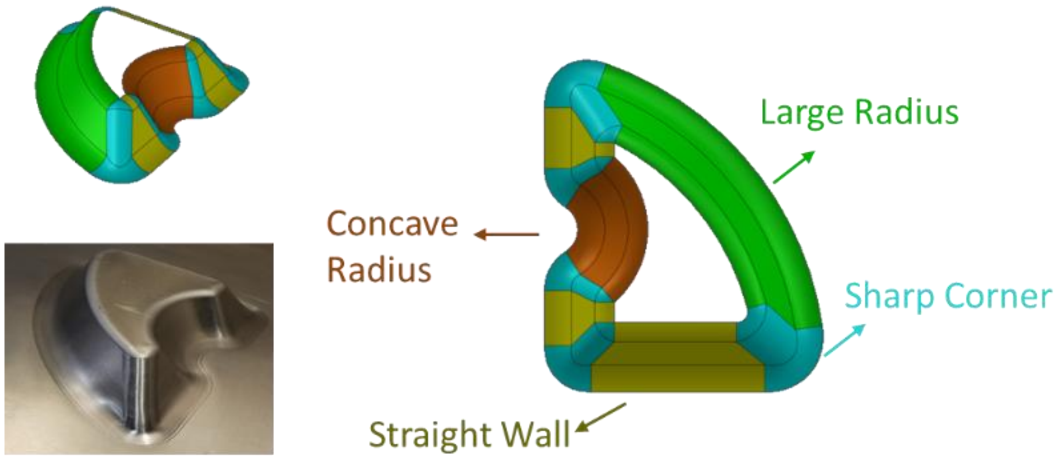


**Figure 2.45 Included complex geometric features in Case B**

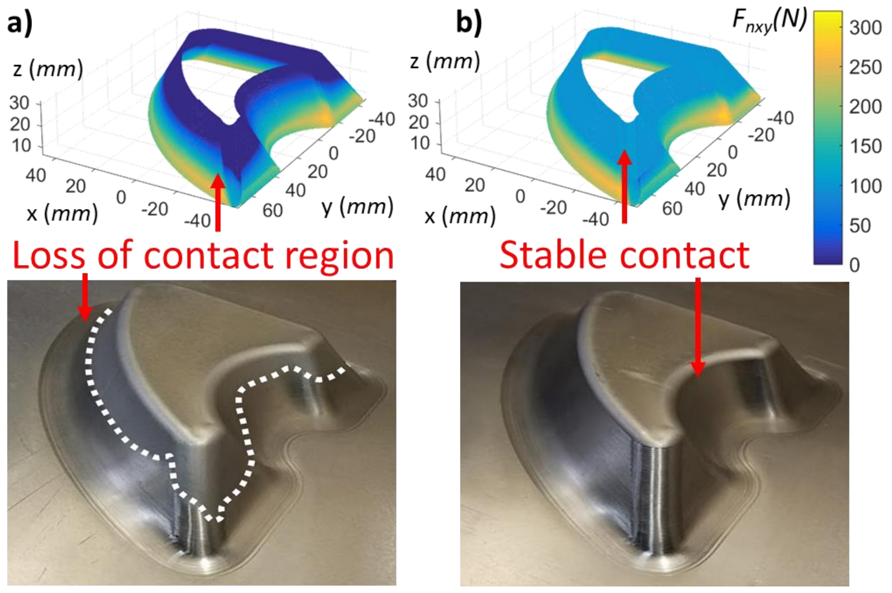


**Figure 2.46 Formed fish fin parts: (a) position controlled showing loss of contact region; (b) force controlled with stable contact**

To fully explore the influence of the complex geometry on the detailed force control accuracy, the force history in a single revolution is also examined in Figure 2.47. As one can see, force fluctuation become more significant in the sharp corners, but the overall error is still being controlled to be less than ±10 *N*, which is the same as observed in Case A.

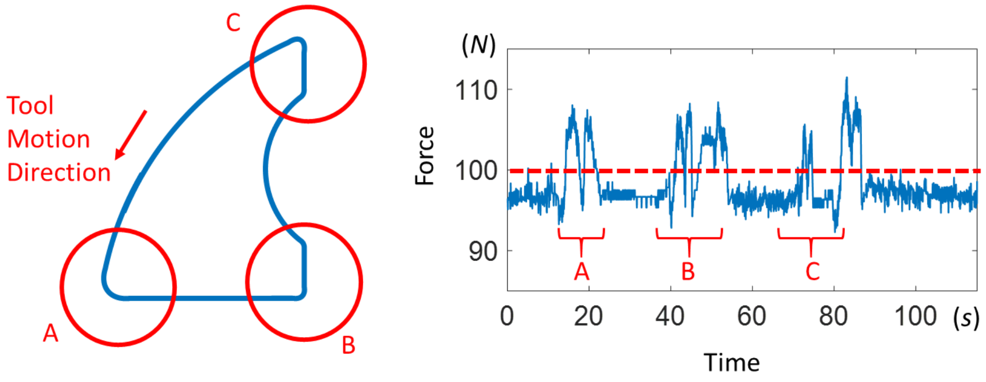


**Figure 2.47 Force history of $F_{nxy}$ for one revolution in Case B**

$F_{nz}$ is also examined in Figure 2.48 for the same revolution. To verify the assumption mentioned at the beginning of Section 2.3 that $F_{nz} = F_{nxy}\ cot\varphi$, it is also plotted as the blue curve. It is observed that $F_{nz}$ is also changing rapidly in the sharp corners, but the assumption still holds if the variation of $F_{nxy}$ is considered.

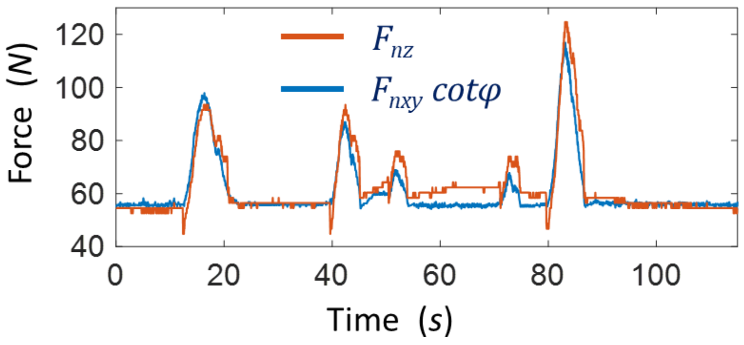


**Figure 2.48 Force history of $F_{nz}$ and $F_{nxy}cot\varphi$ for one revolution in Case B**

In Case C, a 45° pyramid part was formed with the commanded force varying between subsequent walls to demonstrate the algorithm's capability to change the contact force during forming and, thereby, providing the flexibility to control the stress state as a function of position (Figure 2.49). The X-Y force histories are shown in Figure 2.49(c), where the errors are $\pm 10$ *N*; similar to those in Cases A and B. Furthermore, the Z-contact force also varies in relation to the controlled X- and Y-forces, i.e., as $F_{nz} = F_{nxy} \cot(45°) = F_{nxy}$, verifying again the validity of the simplification assumed at the beginning of Section 2.3.

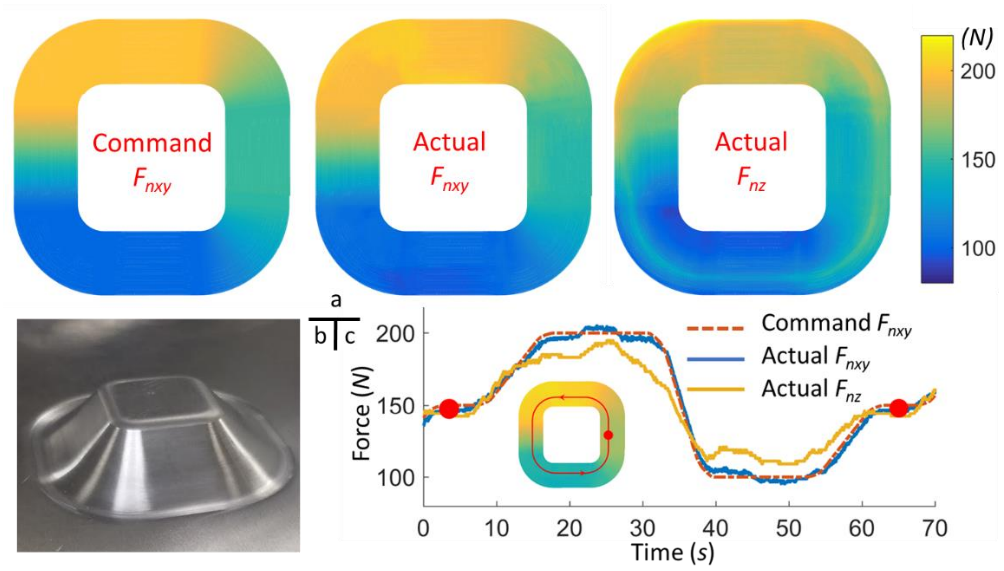
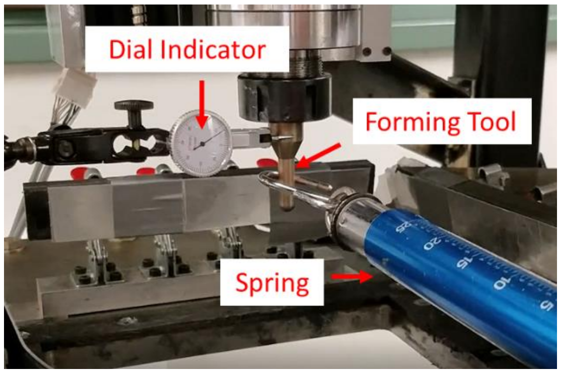


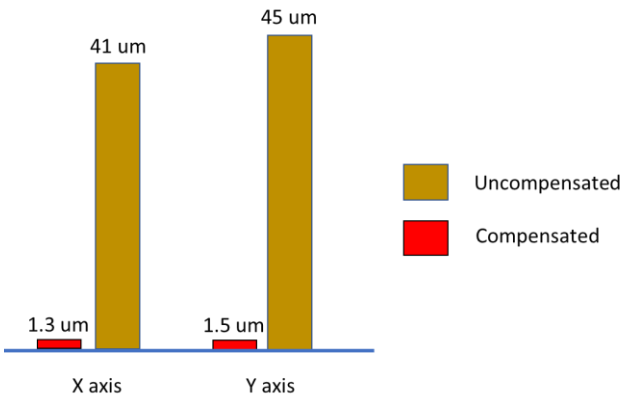
**Figure 2.49 (a) Top view of the commanded and actual force vs. position; (b) formed 45° pyramid part; (c) force history for one revolution**

### *2.4.3. Tool Compliance Compensation*

The tool compliance compensation algorithm is also validated with several physical tests. First, a simple tool pulling test was conducted to examine the tool deflection under specific loads. The schematic and the results of the tool deflection experiments are presented in Figures 2.50(a) and (b). The experiments show that the compensation algorithm can reduce the deflection up to 95%.



**(a)**



**(b)**

**Figure 2.50 (a) Experimental setup with tool pulling test (b) tool deflection measurement under a 100 *N* pulling force**

Next, a 50$^o$ cone shape is manufactured from AA 5754-O with the forming tool compensated following the control diagram in Figure 2.38 with the experimental results shown in Figure 2.51. The compensation experiments were terminated when loss of contact occurred, hence its formed height was lower than the pre-described shape. As it can be seen from the figure, a delay of loss of contact was observed, and the forming forces increase. However, without the supporting tool's contact force control, loss of contact is still inevitable, most likely due to the mispredicted sheet thickness in the toolpath generation procedure. Note that in the figure an additional case, i.e., partially compensated, is added, in which $K_p$ is set equal to half of the measured compliance value of the X and Y axes. That means, only half of the tool deflection caused by the forming force was compensated by the compliance compensation algorithm. As expected, the loss of contact in this partial compensated case occurred earlier than the fully-compensated case.

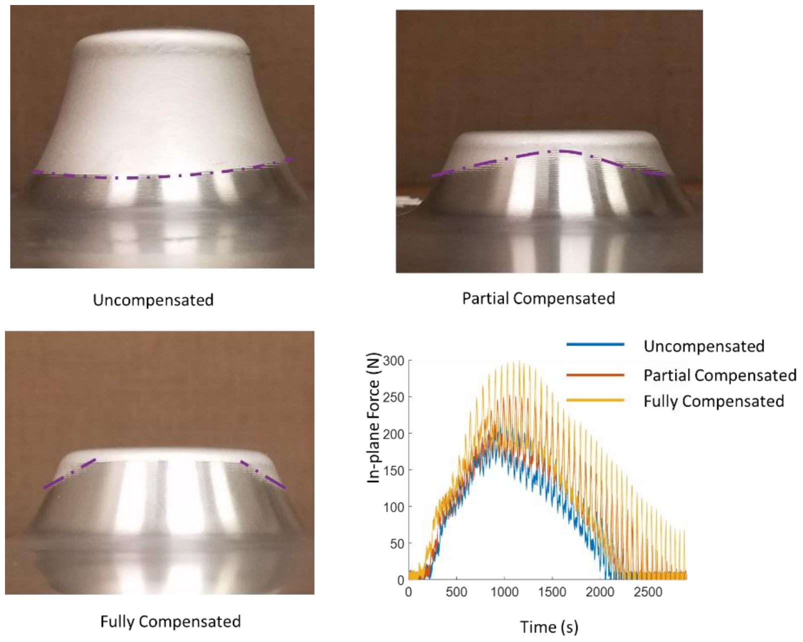


**Figure 2.51 Side view of the truncated cone shapes for different tool compliance compensations (full/partial/uncompensated) and their corresponding in-plane (X-Y) forming forces on the forming tool**

As for the improvement of geometry accuracy due to tool compliance compensation, it will be explained later in Chapter 4 together with the in-process springback error compensation algorithm, since the tool position error is hard to be decoupled from in-process springback errors in the measurements.

## 2.5. Summary and Discussion

A general compline compensation/contact force control algorithm has been proposed and implemented for the DSIF process. With the forming process controlled, one can prevent the loss of tool contact and partly control the local stress state to enhance the material’s formability and geometric accuracy. The proposed control algorithm can be implemented using common motion controllers in commercial machine tools. Its generality and effectiveness have been demonstrated via forming complex geometries, various thicknesses, and different material types. The work paves the way for wide adoptions of DSIF technologies. In future developments, a new mechanics-based model would help to determine the target forming forces for different materials and geometries along with the consideration of fatigue, surface finish, and geometric accuracy. Adaptive control schemes could also be considered to account for variations in the contact stiffness during forming. Moreover, although the compliance compensation algorithm and force control algorithm is designed to be able to execute simultaneously, the current setup only allows one of the algorithms to run due to the computational capability constraints of the motion controller. Thus, the integration of the tool compliance compensation and force control algorithm needs to be further explored. A detailed discussion of the future work will be covered in Chapter 5.

# Chapter 3 Off-line Toolpath Optimization for Accumulative-DSIF

In Accumulated Double-Sided Incremental Forming (ADSIF), two hemispherical tools impart local deformations to the sheet via their programmed in-plane spiral motion. The depth of the part is achieved via the accumulation of the rigid body motion of the already-formed portion of the part. Unlike Single Point Incremental Forming (SPIF) and Double-Sided Incremental Forming (DSIF), ADSIF does not impose forces on the already-formed part and, therefore, has the potential of achieving higher geometric accuracy, as reviewed in Section 3.1. A systematic toolpath optimization method (Loop II in Figure 1.22) will be introduced in Section 3.2 to study the influences of the relative tool positions on the local formed shape and the final geometry, which is essentially the accumulation of all previously imposed local deformations. The effect of tool deflection/compliance related to part geometry is then studied for the ADSIF process in Section 3.3. At the end of Section 3.3, a complete off-line tool optimization algorithm is established with tool compliance considered, based on which an axisymmetric part with varying wall angles was successfully formed. It was confirmed that ADSIF demonstrates improved geometric accuracy compared to conventional Double-Sided Incremental Forming (DSIF).

## 3.1. State-of-the-Art

### *3.1.1. Introduction to Accumulative-DSIF*

As reviewed in Chapter 1, Incremental Forming (IF) is a flexible sheet forming process and has received much attention in the low volume and rapid prototyping fields. Simple generic tooling is utilized to incrementally deform the sheet material within a local region following a predefined toolpath. The final part shape is achieved by the accumulation of all the previously-imparted local deformations. DSIF is an augmented version of the SPIF process by adding a second tool on the opposite side of the sheet (Maidagan et al., 2007; Meier et al., 2007; Wang et al., 2008). With the second tool as a local support, DSIF is capable of constraining the deformation in a local region and thus achieves a more accurate geometry than SPIF can. DSIF may degenerate to SPIF if the contact between the supporting tool and the sheet is lost. This degeneration may occur in DSIF if the tool gap between the two tools exceeds the current thickness of the sheet in the deformation region (Malhotra et al., 2011; Meier et al., 2011b). To prevent the loss of contact, an accurate model for predicting the sheet thickness distribution will be needed, which is still an active research area particularly for multi-pass incremental forming processes (Bambach et al., 2009) or for an arbitrary geometry (Ambrogio et al., 2005; Bambach, 2010).

To address the above-stated issue, a contact force control loop has been developed as described in Chapter 2. Although the algorithm is designed to be implemented in most standard controllers, the machine is still required to be equipped with load cells and A/D converters, which might not be desirable due to setup and maintenance costs.  To be able to maintain contact with a pure position-controlled machine setup, a novel strategy called Accumulated Double-Sided Incremental Forming (ADSIF) was proposed by Malhotra et al. (Malhotra et al., 2012). In ADSIF, unlike DSIF, two tools are always positioned at a given horizontal plane to deform the virgin material via an in-plane motion. The depth of the part is accumulated via rigid body motions of the already-formed part of the sheet, as shown in Figure 3.1. Additionally, the tool motion is designed to move from inward to outward; and consequently, the formed portion is displaced in the direction normal to the in-plane motion by rigid body motion (see Figure 3.1).

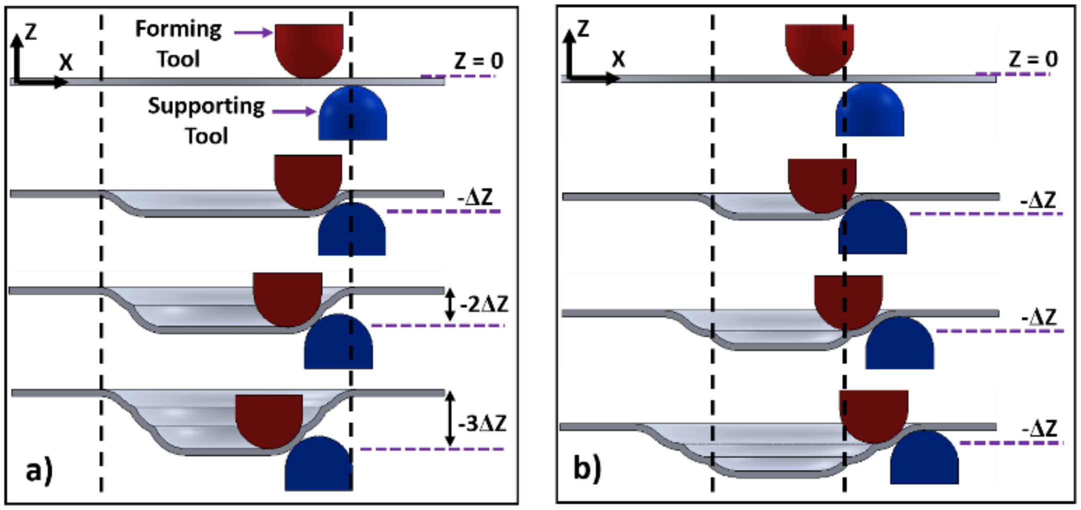


**Figure 3.1 Difference between DSIF (a) and ADSIF (b) toolpaths**

The key strength of ADSIF is that two tools are always forming the virgin material and the current forming state experiences negligible influence from the formed part so that a highly localized deformation can be maintained. Thus, once the tools are initially placed in contact with the sheet, there will be no loss of contact during the whole forming process. As shown in Figure 3.2, forming forces are stable and remain engaged in ADSIF after the initial stage, which means no loss of contact occurs. Furthermore, it has been observed that higher formability and better fatigue behavior of the sheet metal part can be achieved through ADSIF compared to SPIF and DSIF (Malhotra et al., 2012; Xu et al., 2014b).

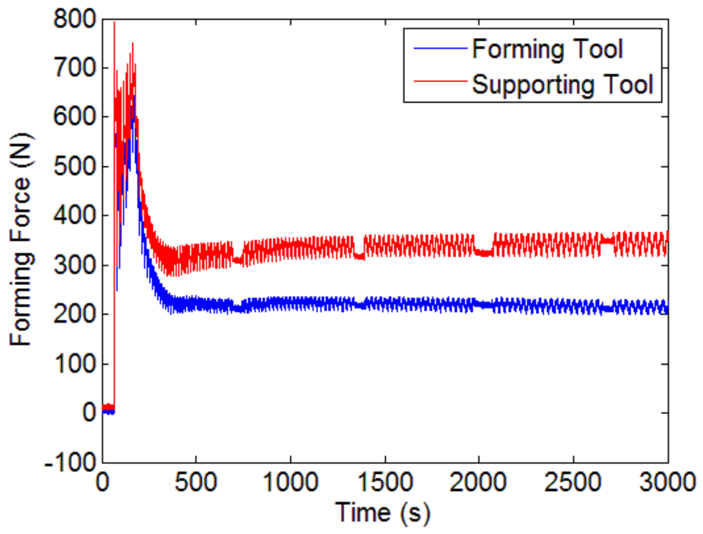


**Figure 3.2 Typical forming force histories in ADSIF**

This novel tool motion strategy offers the ADSIF process the potential of achieving better performance in terms of geometric accuracy and material formability as demonstrated in (Malhotra et al., 2012). However, ADSIF also raises several challenges. As discussed before, the depth of a part is achieved by the accumulation of rigid body motions of the formed part instead of being directly controlled by the forming tools. There has not been a direct relationship formulated between the desired shape and toolpath parameters. Currently, a trial-and-error based design approach is required for determining the toolpath parameters in order to achieve a specified geometry. Another challenge in ADSIF is that the final part shape is essentially an accumulation of previously deformed steps, which implies that any error will also be accumulated. Consequently, in ADSIF a small error per single loop will dramatically induce the geometric inaccuracy in the final part. Both challenges reveal that the toolpath in ADSIF needs to be carefully optimized to achieve the desired geometry.

### *3.1.2. Enhanced Controllability of Accumulative-DSIF*

Before diving into the details of the toolpath optimization algorithm, another comparison is explained in this section between conventional DSIF and ADSIF from the control aspect. The path planning and control of the IF process, or of similar flexible sheet forming manufacturing processes reviewed in Section 1.1, can often be summarized as a control optimization problem, i.e., how to choose the proper design parameters to achieve the minimal geometric error between the target and formed geometries (Allwood et al., 2009). Note that the geometric error consists of the tool position error $\delta^1$, in-process springback error $\delta^2$ and post process error $\delta^3$ as suggested by Equation 1-1. But, only the in-process springback error, as a major geometric error $\delta^2$, will be addressed in this chapter, as well as in Chapter 4, since $\delta^1$ can be compensated by compliance compensation as described in Chapter 2 while $\delta^3$ can be largely eliminated by a simple annealing operation, as reviewed in Section 1.2.3.

Considering the traditional DSIF system depicted in Figure 3.3 (same as Figure 1.9), the optimization problem can be stated in the following mathematical form:

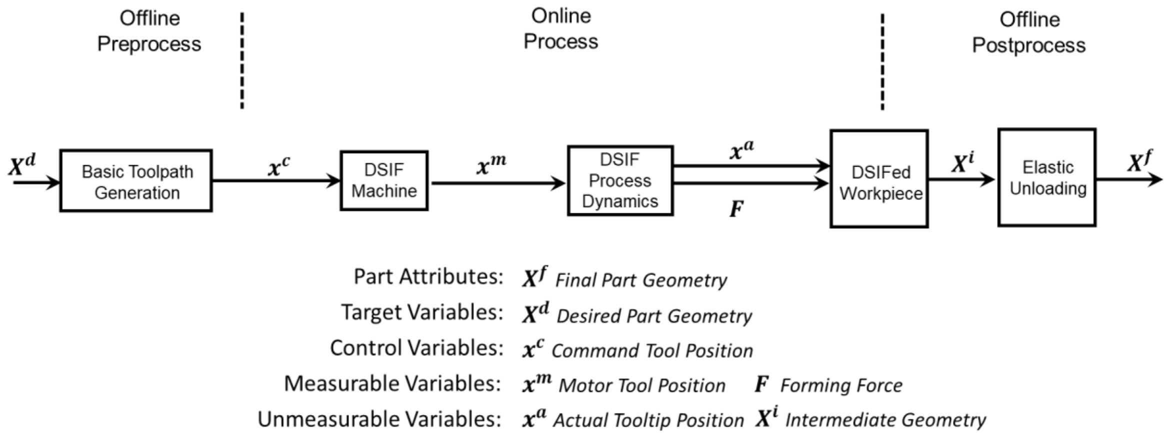

Offline Preprocess
Online Process
Offline Postprocess
$X^d$
Basic Toolpath Generation
$x^c$
DSIF Machine
$x^m$
DSIF Process Dynamics
$x^a$
$F$
DSIFed Workpiece
$X^i$
Elastic Unloading
$X^f$
Part Attributes: $X^f$ Final Part Geometry
Target Variables: $X^d$ Desired Part Geometry
Control Variables: $x^c$ Command Tool Position
Measurable Variables: $x^m$ Motor Tool Position $F$ Forming Force
Unmeasurable Variables: $x^a$ Actual Tooltip Position $X^i$ Intermediate Geometry

**Figure 3.3 Traditional DSIF process**

$$\min_{x^c} \quad J = \sum_{i}^{N} \left\| X_i^f - X_i^d \right\|^2 + L(X_i^f, x^c{}_i) \tag{3-1}$$

$$subject\ to: X_N^f = f(X_{i=1,2,..,N-1}^f, x_{i=1,2,..,N}^c) \tag{3-2}$$

$$b(X_i^f, x^c{}_i) \le 0 \tag{3-3}$$

where the $X^f$, $X^d$ and $x^c$ mean the final part geometry, desired part geometry and command tool position in the toolpath, as represented in Figure 3.3. Note that the system is represented by a discrete form, and the subscript $i$ means the sequence number of a discrete point in the toolpath for $x^c$ and the corresponding part geometry ($X^f$ or $X^d$). The discretization of the system is utilized instead of a continuous form since the generated toolpath in DSIF is by nature a sequence of discrete points, and also the whole process is a quasi-static process in which dynamic effects can be neglected.

Equation 3-1 states that the aim of the IF process is to achieve the desired geometry at every control point, measured by the summation of the Euclidean distance between the target and formed geometry ($X^d$ and $X^f$). On the other hand, $L$ stands for a penalizing function to minimize any sudden tool motion changes or large forming forces. Equation 3-2 represents the process model $f$, where the formed geometry $X^f{}_N$ evolves with the input tool position $x^c$ and the previously built geometry $X^f{}_{i = 1,2,...,N}$. In Equation 3-3, the $b$ function designates the process constraints that limit the process window. For instance, the material forming limit restraints the final achievable geometry ($X^f$) or the machine structure constraints the possible motion range of the physical actuators ($x^c$).

The above equations (Equations 3-1 to 3-3) express an idealized statement without considering disturbances and model uncertainty. The problem could be solved with a perfectly accurate model *f*. However, given the nonlinearity of DSIF caused by material plasticity, geometric variations and varying local contact conditions between the metal and the forming tool(s), it is almost impossible to establish a generalized model to minimize the geometric error for arbitrary complex geometries. More importantly, the traditional DSIF process demonstrates strong history dependency, where the current forming geometry not only depends on the current input command tool position, but also all the previous input tool positions and achieved geometry. That is due to the fact that the plastic deformation always happens on the already-built part in DSIF, as shown in the truncated cone formed with DSIF in Figure 3.4, where the $N^{th}$ contour toolpath always imposes on the formed part from $(N-1)^{th}$, $(N-2)^{th}$, …, i.e., the history of the forming process. This strong history dependence of the conventional DSIF process means that Equation 3-1 needs to be minimized over all toolpath points. However, solution times for a single run of a credible model (such as a full-scale model) of a typical DSIF process are currently measured in days or even weeks (Moser et al., 2016a). Moreover, the large number of toolpath points also means a large number of Degrees Of Freedoms (DOFs) to be planned or optimized, so the space of possible paths is enormous for a full-scale model to iterate over, which makes the FEM-based toolpath optimization impractical.

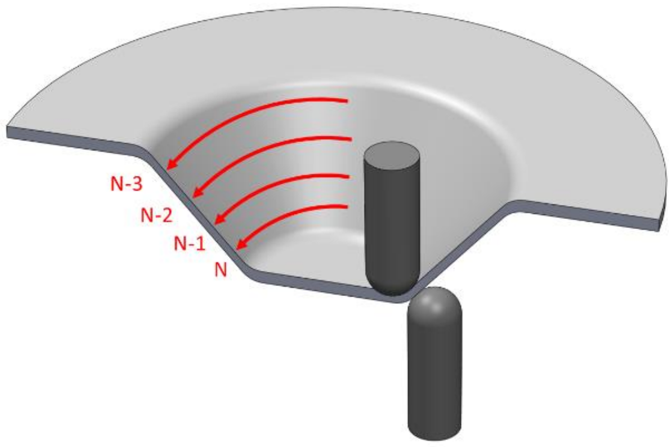


**Figure 3.4 A demonstration of the DSIF process where the N$^{th}$ contour toolpath is formed on the formed part from the 1$^{st}$ to the N-1$^{th}$ contour**

The complexity and strong history dependency stated above suggest that it would be impossible to create a perfect model *f* and conduct toolpath optimization in practice in a typical DSIF process. On the other hand, the unique nature of ADSIF makes it possible to simplify the process and eliminate this history dependency. As reviewed in Section 3.1.1, the critical difference between DSIF and ADSIF is that the tools are always forming virgin material in ADSIF, and the influence of the already built part becomes negligible.  As shown in Figure 3.5, for a similar truncated cone shape, the tools are always traveling outwards and forming the virgin material without interference due to the already formed paths (N-1, N-2,…).

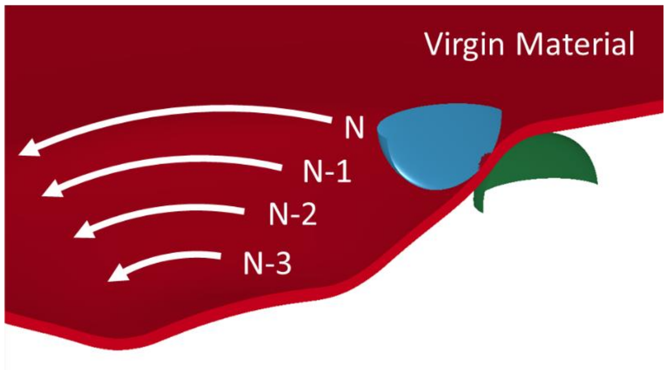


**Figure 3.5 A demonstration of the ADSIF process where the Nth contour toolpath is formed on the virgin material**

Regarding ADSIF, the process model now can be simplified and the formed part, for a given tool position, only expressed by the current command tool position as:

$$X_N^f = f(x_N^c) \tag{3-5}$$

If the function *f* can be established and *L* is negligible (as in some cases geometric accuracy is the only variable of interest), the toolpath optimization problem (Equation 3-1) can be simplified, i.e.;

$$x_N^c = f^{-1}(X_N^D) \ \ for\ i = 1, 2, \ldots, N \tag{3-6}$$

in which the optimal command position is retrieved by solving the inverse function of *f* at each point on the toolpath.

To establish the model *f* or the inverse function $f^{-1}$, the specific relationship between the relative tool position and the resulting local geometry needs to be formulated. Smith et al. utilized an explicit simulation model to study the mechanics of ADSIF and compared it with SPIF (Smith et al., 2013). It is reported that ADSIF is characterized by a bending-unbending deformation with squeezing of the sheet. However, due to the number of weeks required to finish a simulation, only one ADSIF simulation with a specific relative tool position was studied. The influence of different tool path parameters was not considered. Xu et al. and Zhang et al. proposed a method to combine ADSIF and DSIF parameters to enhance the geometric accuracy and reduce the forming time (Xu et al., 2014a; Zhang et al., 2015). However, still only one set of tool positions was studied.

Ndip-Agbor et al. proposed a systematic framework to design the relative tool position in ADSIF based on a simplified simulation model (Ndip-Agbor et al., 2016), in which *D* and *S* are utilized to define the horizontal and vertical distances between two tools. However, the determination of the *D* and *S* parameters does not only depend on the target geometry, but is also related to some key process parameters, such as sheet thickness and tool diameter. Therefore, their approach is difficult to be generalized beyond the specific case studied in (Ndip-Agbor et al., 2016).

### 3.1.3. Discussion and Statement of Research Objectives

In the following sections, *a systematic methodology for determining a process model for ADSIF and understanding of the effects of relative tool position on the formed geometry will be established based on the desired part geometry* by: (1) defining the design variables for tool positions based on the deformation mechanism, which alleviates the difficulty mentioned above (Section 3.2.1); (2) establishing a simulation model to be used as a means for conducting numerical experiments (Section 3.2.2); and (3) exploring the design space (Section 3.2.3). The resulting formed angle and forming force concerning the relative tool positions predicted by the numerical models are presented in Section 3.2.4 to Section 3.2.6, with an experimental verification case presented in Section 3.2.7.

In addition to the above aspects, *the influence of the tool compliance will also be considered in the toolpath optimization framework.* In Section 3.3, an analytical study is presented to demonstrate the dramatic effects that tool compliance can have on ADSIF (Section 3.3.1). Then, a simulation investigation is performed to verify the influence of the tool compliance in Section 3.3.2. With all ingredients ready, a complete off-line tool optimization algorithm is established with the tool compliance considered in Section 3.3.3. Additionally, a validation experiment is conducted that demonstrates that ADSIF improves geometric accuracy as compared to DSIF for the conical part of interest (Section 3.3.4).

## 3.2. Accumulative-DSIF Optimization Methodology

In this section, the relative tool positions for a given ADSIF toolpath are defined in Section 3.2.1 to have a better means of interpreting the formed geometry with respect to tool positions. A high-resolution explicit simulation model will be presented in Section 3.2.2. The design space is explored using different toolpath parameters through a simple cone geometry in Section 3.2.3. Note that in ADSIF, the deformed part is essentially an accumulation of all previous local geometries. Therefore, a part with complex geometry, viewed from the standpoint of each local deformation, can be represented by a sequence of cone geometries with infinitesimally small height with each of the cones having its own instant wall angles. For this reason, the methodology represented here applies to the forming of any complex geometry.

### *3.2.1. Definition of the Control Variables*

In ADSIF, two tools, a forming tool, and a supporting tool, are positioned to move outward while their Z-level coordinates are maintained. The tool motion trajectory of the forming tool can be easily achieved by projecting the trajectory of the forming tool, as in a SPIF case, into a horizontal plane, parallel to the plane of the initial sheet. Through the forming process, the tip points of the forming tool are placed in this plane as shown in Figure 3.6. If the distance between the projection plane and the initial sheet surface is decided, the toolpath for the forming tool is then fully determined. The supporting tool is placed relative to the forming tool accordingly determined by the relative tool positions.

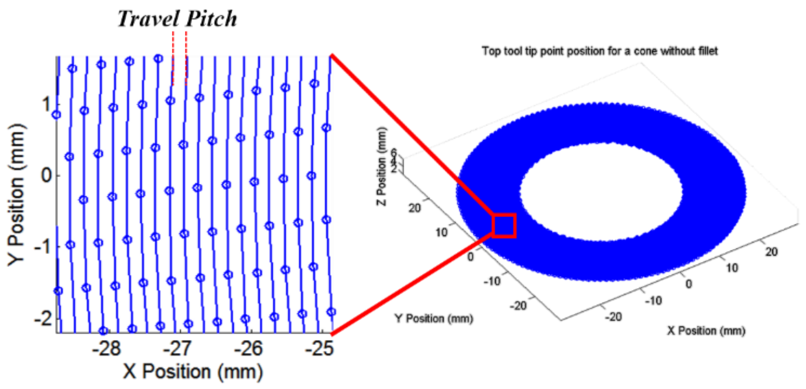


**Figure 3.6 Top tool tip points in ADSIF for a single cone without a fillet**

The tool positions are drawn in the plane normal to the tool motion direction as shown in Figure 3.7. The *projection distance* represents the distance between the tip point of the forming tool and the initial top surface of the workpiece. The *position angle*, $\theta$, represents the angle between the vertical direction and the line connecting the centers of the forming tool and of the supporting tool. Another parameter called the *normalized tool gap* ($\overline{T_g}$) is defined by the tool gap divided by the initial sheet thickness, in which the tool gap is the minimum distance between the two tools.

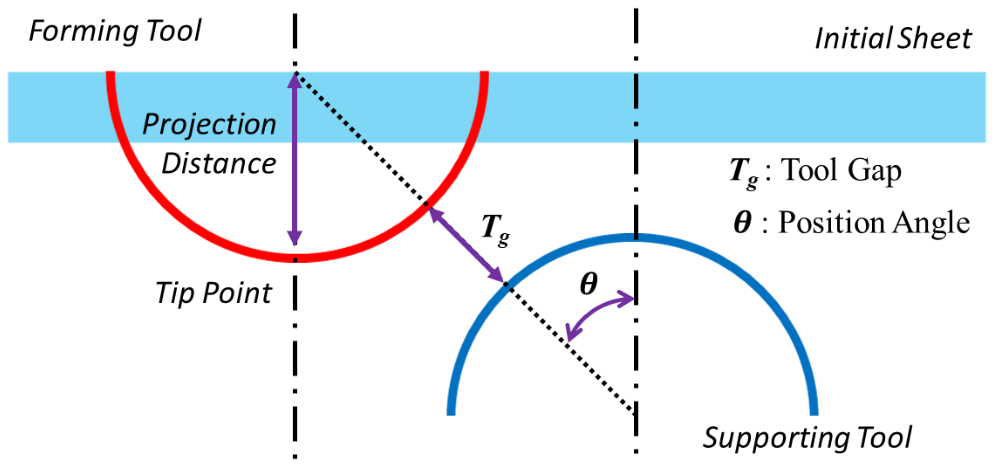


**Figure 3.7 Schematic of toolpath parameters**

To investigate the influence of relative tool positions, only $\theta$ and $\overline{T_g}$ are considered. Basically, $\theta$ represents the magnitude of bending of the sheet under deformation. A large $\theta$ means that the supporting tool is placed more upwards and bends the sheet more around the tool. $\overline{T_g}$ directly links to the magnitude of the squeeze applied to the sheet material.

The tool path parameters which are not considered in this work are the *projection distance* and the *travel pitch*. In ADSIF, a large plastic deformation happens surrounding the forming tool and the supporting tool, while a small projection distance, for example, 1.6 *mm* of projection distance, which will be utilized later in the study, for a 1 *mm* thick and 300 *mm* wide square-shape sheet, mostly induces the elastic deformation only. Travel pitch defines the distance between two sequential loops as shown in Figure 3.6. A small travel pitch reduces the forming force, enhances the geometric accuracy but dramatically increases the forming time. Hence, in this study, a moderate value of 0.2 *mm* is chosen to simplify the model.

### *3.2.2. Simulation Model Setup*

A simulation model is used as a tool to perform numerical experiments, i.e., finding the different geometric shapes as a result of various $\theta$ and $\overline{T_g}$. Using numerical experiments is preferred because of three reasons: (1) Simulations are capable of offering much more information than experiments. Stress and strain histories are very hard to measure in experiments but can be obtained directly from simulations. (2) All simulation results are history tractable, which means data is available at any time interval whereas only the final results can be measured in the experiments. In ADSIF or DSIF, the two tools in combination with the high levels of plastic strain make it impractical to use Digital Image Correlation (DIC) and/or strain gages for measuring the strain. In addition, the use of machine grease for tool lubrication makes it difficult to record etched grids over the course of an experiment. (3) Simulations are free from experimental uncertainties, such as extra tool deflection due to machine compliance, although they have their own uncertainties, such as the material constitutive model. Nevertheless, simulations can demonstrate the trends more decisively.

A simulation model is established in LS-DYNA as shown in Figure 3.8 using the explicit integration method. The sheet (250 x 250 x 1, units: *mm*) is modeled by 8-layer solid elements with reduced integration to accurately capture the bending and compression mechanics. The forming tool and the supporting tool, whose diameters are 10 *mm*, are modeled using rigid shell elements. In total, the model contains 510,868 elements. The mass scaling method is used, and the tool speed is increased from 5 *mm*/s used in the experiments to 750 *mm*/s in the simulations to speed up the simulation process. Meanwhile, artificial damping is added to eliminate the dynamic effects since ADSIF is a quasi-static process. The sheet material, aluminum alloy AA5754-O, is modeled by the von-Mises yield criterion and the Voce hardening law. Note that a more sophisticated model probably should be used in the simulations to better capture the material behavior in ADSIF. However, one practical reason for choosing the simplified material model is that the current model took 120 hours to finish in an 8-processor cluster. Nevertheless, the simulation was calibrated by tuning different contact penalty factors and time steps to compare the cone geometry obtained in experiments as shown in Figure 3.9. Notice that although the dimensions of the cone are different between the simulation and the experiment, this simulation model is capable of accurately predicting the key geometric features and the wall angle.

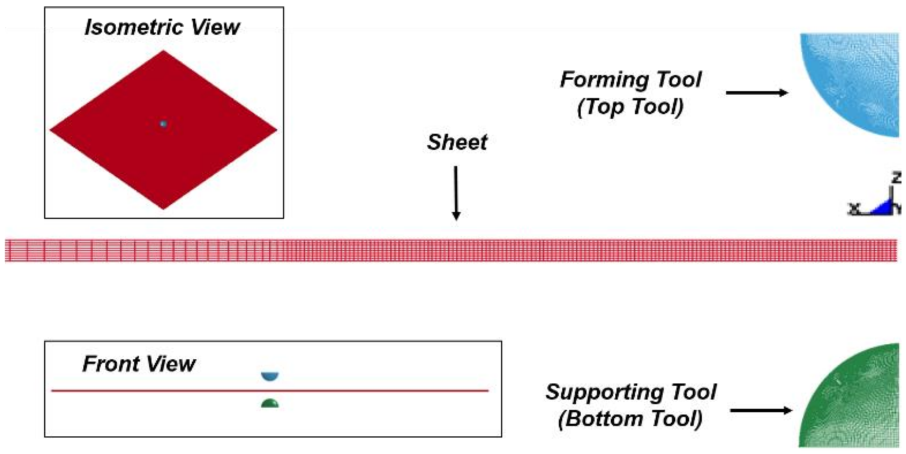


**Figure 3.8 Simulation model**

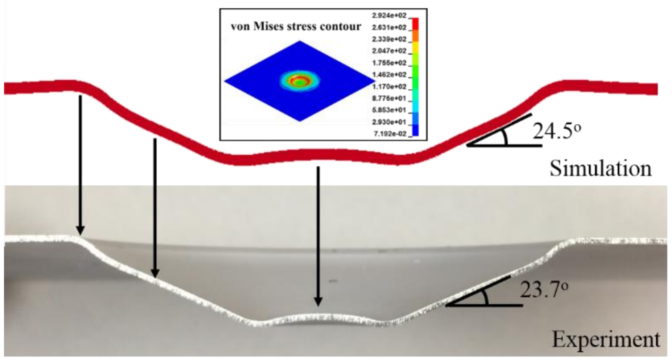


**Figure 3.9 Cross-section comparison between the simulation and the experiment**

### *3.2.3. Design Space of Toolpath Parameters*

The forming tool and the supporting tool are moved spirally outward. Correspondingly, a simple cone is formed. This axisymmetric shape enables an easy connection between the formed angle and relative tool position parameters. The observed result can then be generalized to more complex geometries. The inner diameter of the cone used in this study is 24 *mm*, while the outer diameter is 48 *mm*. The projection distance is kept at 1.6 *mm*, and the incremental pitch is set to 0.2 *mm*.

The input variables for process design are $\theta$ and $\overline{T_g}$ which stand for the relative tool position as stated before. The 24 simulations performed for different levels of $\theta$ and $\overline{T_g}$, with 4 levels of $\theta$ and 6 levels of $\overline{T_g}$ are shown in Figure 3.10. However, when the tools squeeze the metal sheet more than 40%, local buckling happens which causes the simulations to become unstable. Thus, only 19 of the 24 simulations results are recorded and analyzed, which will be presented in the next section.

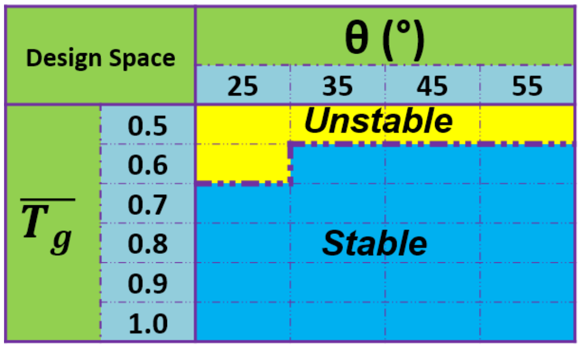


**Figure 3.10 Design space and applicable simulation results**

### *3.2.4. Response Surface Model of the Wall Angle*

In an effort to study the effects of different toolpath parameters on geometry, i.e., the process model *f* which defines the relationship between the local wall angle and relative tool positions, two new indicators are defined, namely, the *stable angle* and the *peak angle,* which are determined from the wall angle profile of the final geometry. The final wall angle along the radius of the cone is shown in Figure 3.11. In the forming region (12 *mm* < r < 24 *mm*), the *stable angle* is defined as the plateau value as indicated by the dashed line and the *peak angle* is the maximum wall angle indicated by the arrow in Figure 3.11. In this section, the resulting angles, stable and peak angles, from different relative tool positions will be presented respectively, followed by the discussion on the forming force results in Section 3.2.6.

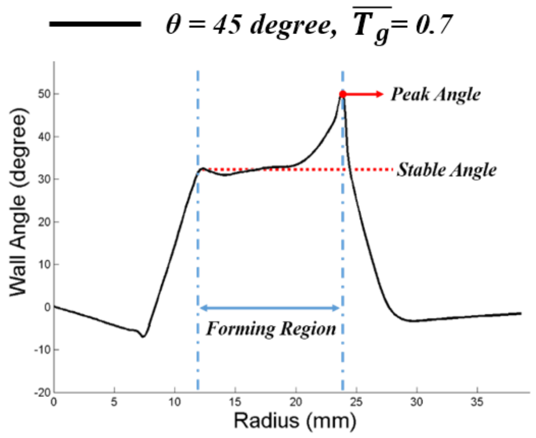
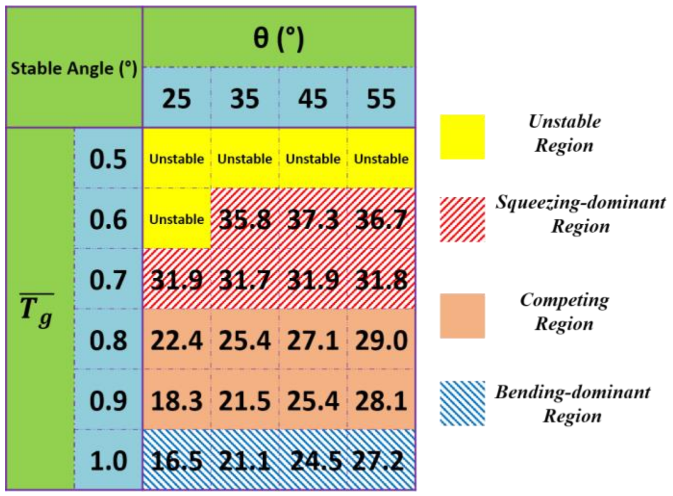


**Figure 3.11 Wall angle plot along the radius of the final geometry**

Figure 3.12 shows the different stable angles obtained from all 19 simulation cases. The general investigation reveals that $\theta$ is responsible mainly for the amount of sheet metal bending about the supporting tool, while $\overline{T_g}$ corresponds to the induced pressure on the material. From the trends in Figure 3.12, it becomes apparent that different combinations of $\theta$ and $\overline{T_g}$ lead to three different regions of interest: a squeezing-dominant region, a competing region, and a bending-dominant region.



| Stable Angle (°) | | θ (°) | | | |
|---|---|---|---|---|---|
| | | 25 | 35 | 45 | 55 |
| $\overline{T}_g$ | 0.5 | Unstable | Unstable | Unstable | Unstable |
| | 0.6 | Unstable | 35.8 | 37.3 | 36.7 |
| | 0.7 | 31.9 | 31.7 | 31.9 | 31.8 |
| | 0.8 | 22.4 | 25.4 | 27.1 | 29.0 |
| | 0.9 | 18.3 | 21.5 | 25.4 | 28.1 |
| | 1.0 | 16.5 | 21.1 | 24.5 | 27.2 |

| | |
|---|---|
| 🟨 | *Unstable Region* |
| ▨ | *Squeezing-dominant Region* |
| 🟧 | *Competing Region* |
| ▧ | *Bending-dominant Region* |

**Figure 3.12 Simulation results for different tool positions**

In the squeezing-dominant region, $\overline{T_g}$ is less than 0.8 and the wall angle is primarily influenced by the chosen $\overline{T_g}$. It can be seen that the stable angle increases as $\overline{T_g}$ decreases; however, it maintains the value even if $\theta$ is increased from 25° to 55°. On the other hand, the bending-dominant region is defined when $\overline{T_g}$ is equal to 1.0. This means that the tool gap is equal to the initial sheet thickness and no squeeze exists during the process. In this region, the wall angle is mainly influenced by bending and stretching between the forming tool and the supporting tool. Notice that in the bending-dominant region, the stable angle varies almost linearly with $\theta$. In the competing region, both squeezing and bending are influential factors. Increasing either $\overline{T_g}$ or $\theta$ will increase the achieved wall angle.

The simulation results for the stable angle are plotted as a response surface in Figure 3.13. The blue dots represent the simulation results. The mesh surface is linearly interpolated based on the simulation data. From the surface, it is apparent that when $\overline{T_g}$ is relatively small, increasing $\theta$

results in a higher stable angle. However, when $\overline{T_g}$ decreases, $\theta$ has a nearly negligible influence, i.e., squeezing dominated.

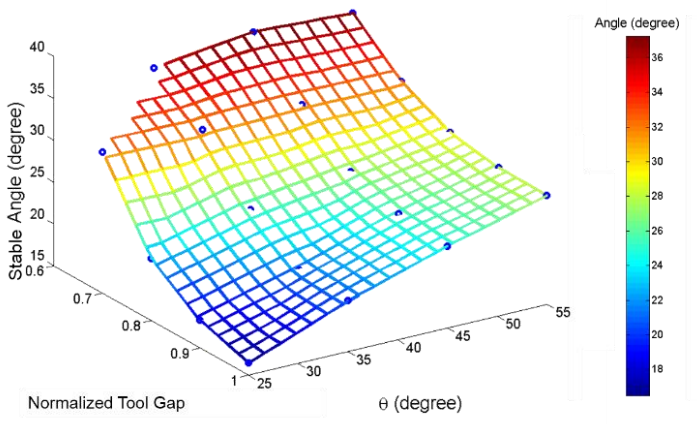


**Figure 3.13 Response surface for the stable angle regarding $\overline{T_g}$ *and* $\theta$**

Note that the response surface presented in Figure 3.13 is, in fact, a discrete form of the process model *f* as mentioned in Section 3.1.2. For any given combination of $\overline{T_g}$ and $\theta$, one can look up or interpolate the corresponding stable wall angle, i.e., the achieved local wall angle. Inversely, if a certain wall angle is given, the corresponding $\overline{T_g}$ and $\theta$ can also be found by intersecting the response surface in Figure 3.13 with a z plane at the desired wall angle value. Every point on the intersection line represents a possible solution. This inverse table look up operation equals to solving the inverse function $f^{-1}$, which constitutes the principle of the toolpath optimization algorithm. For a complex geometry, one only needs to discretize the whole part into a sequence of toolpath point and iteratively look up the optimal $\overline{T_g}$ and $\theta$ for each point according to the response surface.

Another thing to note for the established response surface is that the upper bound of the stable angle for the entire surface is 37.3$^o$ according to the simulation results, which implies that the maximum achievable angle for ADSIF is around 40$^o$ with the current experimental setup, i.e., 1.0 *mm* AA 5754-O sheet is formed with two 5 *mm* radius tools with a 0.2 *mm* pitch. This particular upper bound is smaller as compared to what was observed in the previous literature where it was shown that 55$^o$ was achievable (Malhotra et al., 2012). The possible reasons of the lower maximum angle are: (1) the current travel pitch in the toolpath is eight times of that in (Malhotra et al., 2012), and (2) the sheet thickness is twice of that used in the previous study. A larger travel pitch results in a bigger tool contact region, and in doing so, causes more material to be deformed per increment. The enlarged force due to the larger travel pitch causes forming instability at some combinations of $\overline{T_g}$ and $\theta$, which in fact limits the maximum achievable angle. Furthermore, a thicker sheet requires higher forming forces. Finally, AA2024-T3, a commonly used material for the aerospace industry was used in (Malhotra et al., 2012) whereas AA5754-O, a popular material in the automotive industry, is used here. The exact effect of material types on the maximum formable angle in ADSIF (without considering the fracture limit) is not known yet and will be a topic for future study.

### ***3.2.5. Angle Variation Analysis***

As we discussed in the last section, the response surface, shown in Figure 3.13, can be utilized to inversely choose the optimal tool position parameters of the ADSIF process regarding the desired wall angle. For example, if a 35$^o$ cone is desired, the mesh surface in Figure 3.13 is intersected with a Z-plane at the 35$^o$ stable angle, and multiple solutions can be found on this intersection line. The existence of multiple solutions for the same desired wall angle value allows a further optimization by other criteria, such as by minimizing the angle variation or forming forces as stated below. Note that an added selection criterion is, in fact, a penalty function, which is denoted as $L$ in Equation 3-1.

The profiles of the wall angle along the cone radius for three cases are plotted in Figure 3.14: (1) When $\theta$ = 25 º, $\overline{T_g}$ = 0.7, the supporting tool moves downward and inward relative to the forming tool, and the squeezing effect dominates; (2) When $\theta$ = 45 º, $\overline{T_g}$= 0.8, the supporting tool offers both a large bending and squeezing effect; and (3) When $\theta$ = 45 º, $\overline{T_g}$= 1.0, there is no squeezing and only bending takes place.

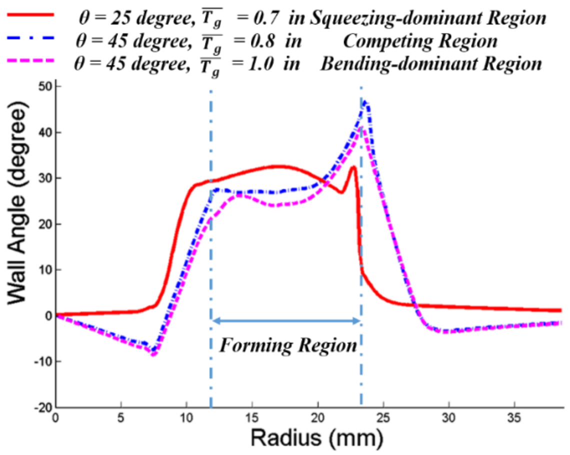


**Figure 3.14 Wall angle profiles for different $\overline{T_g}$ and $\theta$**

As shown in Figure 3.14, the profile of the wall angle along the radius is not constant in any of the three cases. The wall angle varies along the cross-section. The stable angle corresponds to the already deformed material. However, material immediately deformed by the tools experiences a significantly larger wall angle, which is referred as the maximum or peak angle. The difference between the peak angle and the stable angle, i.e., *angle variation*, can be significant and should be minimized. The stable angle is the overall wall angle achieved and determines most of the final geometry, whereas the peak angle determines the wall angle in the region where the two tools are currently forming the part. If one considers a more complex geometry in which the desired wall angle varies by some amount along the cross-section, then it is apparent that a considerable geometric inaccuracy is highly likely to occur in the transient region if the peak angle is much larger than the stable angle. Hence, the *angle variation* should be minimized to achieve the desired geometric accuracy.

In Figure 3.15, the angle variation is plotted as a function of $\overline{T_g}$ and $\theta$. Recall that the angle variation is the difference between the peak angle and the stable angle. The angle variation is mainly dependent on $\theta$, which implies a smaller $\theta$ will offer a more constant wall angle. This could be viewed as a design criterion. In other words, a smaller $\theta$ should be considered when trying to avoid a large angle variation. More generally, the relative tool positions in the squeezing-dominant region with a small $\theta$ is preferable since this is the only region which offers a large stable angle with a small $\theta$.

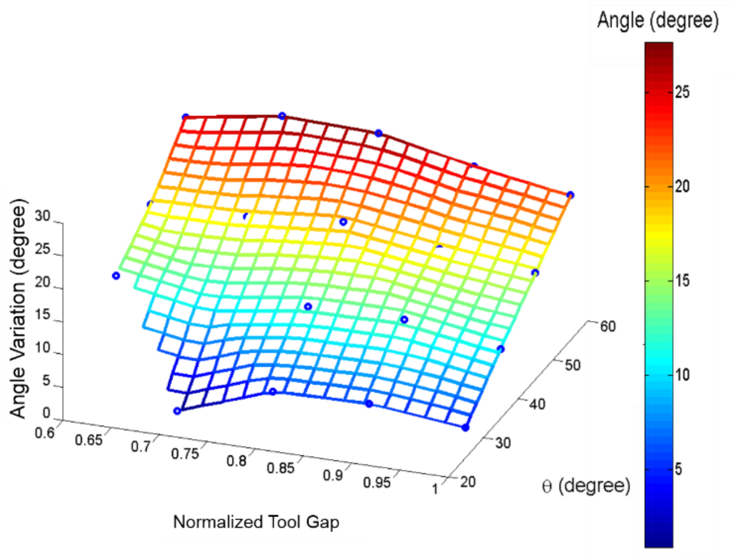


**Figure 3.15 Response surface of angle variations**

The large difference between the peak angle and the stable angle is because the achievable angle is limited by a relatively low upper bound using the current setup (maximal achievable wall angel < $40^o$ as mentioned in Section 3.2.4). So, when $\theta$ is larger than the achieved angle, the supporting tool first over bends the sheet around the forming tool, inducing a large wall angle. Then, the material is squeezed by both tools, inducing the maximum peak angle. After the tools continue moving outward, the material is bent reversely by the forming tool, which decreases the wall angle as shown in Figure 3.16. On the other hand, if $\theta$ is small, the overbending and reverse bending processes are minimized, and the only influential factor is due to squeezing.

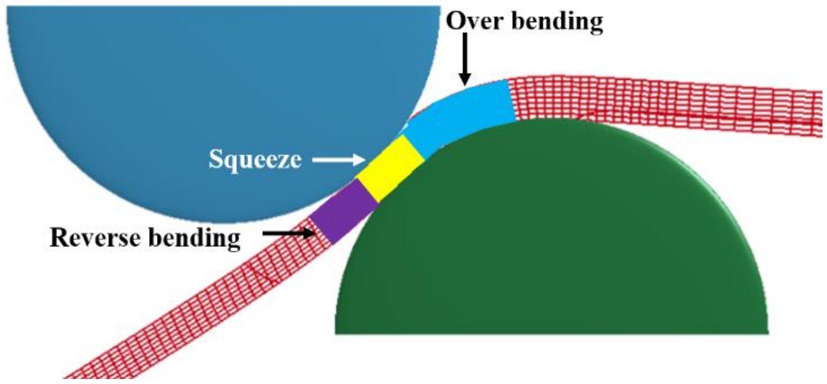


**Figure 3.16 ADSIF deformation process**

The history of the sheet thickness at a specific radius (r = 21 *mm*) is plotted in Figure 3.17, in which one tool loop implies that both tools have traveled 360$^o$ along the tool trajectory. When the parameters are chosen such that bending is dominant ($\theta$ = 45$^o$, $\overline{T_g}$ = 1.0), the reduction in sheet thickness primarily occurs in the over-bending and reverse-bending stages. However, as the squeeze becomes more apparent ($\theta$ = 45$^o$, $\overline{T_g}$= 0.8), a dramatic thickness reduction is observed as the material is essentially extruded through the tools. Finally, when $\theta$ is small ($\theta$ = 25$^o$, $\overline{T_g}$ = 0.7), over bending is minimized, and thickness reduction occurs in solely one step. Notice, however, that a slight increase of thickness is actually present before the thickness is reduced during the squeezing process.

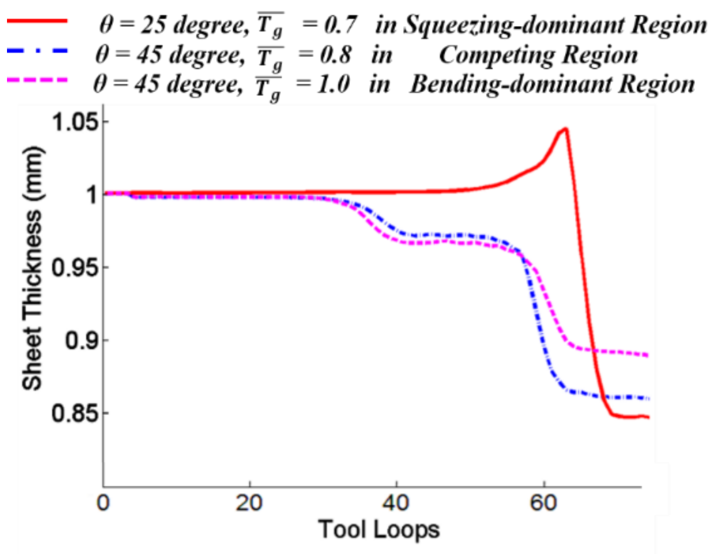


**Figure 3.17 Representative thickness change at one radial location (r = 21 *mm*) because of different $\overline{T_g}$ and $\theta$**

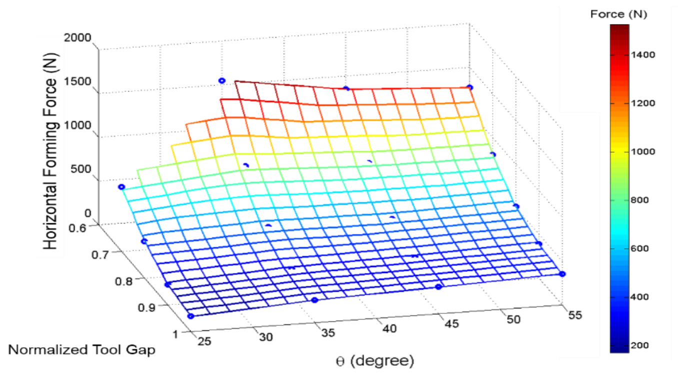
Note that the response surface presented in Figure 3.15 can directly serve as a discrete form of penalty function $L$. In other words, among all the feasible solutions for $\theta$ and $\overline{T_g}$, one should choose the optimal parameters by looking up the corresponding angle variation for each pair of $\theta$ and $\overline{T_g}$ via Figure 3.15. The one with the minimal angle variation should be selected as the optimal solution. Otherwise, one can also simply choose the one with a small $\theta$, since a strong correlation between $\theta$ and the angle variation has been observed.

### *3.2.6. Forming Force Analysis*

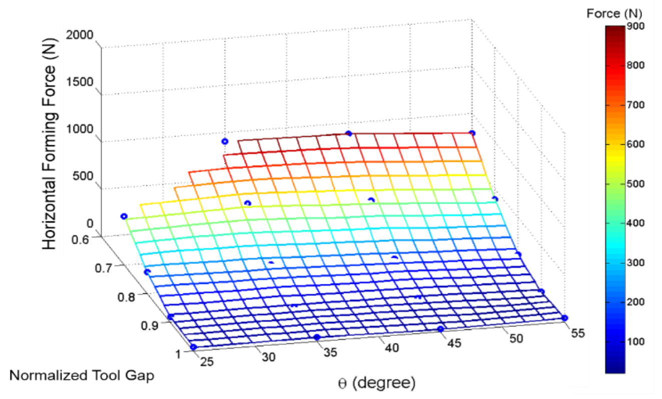
The forming force is usually a pivotal indicator of the mechanics occurring within the forming process, specifically with regards to fracture and buckling, and thus it can serve as another penalty factor $L$ to minimize. Also, in practice, the forming force is a crucial factor when considering a machine's compliance. Large forming forces will deflect both the machine and the tools, causing tool position deviations. Additionally, a large forming force may suggest local instabilities/buckling in front of the tools. For the DSIF prototype machine at Northwestern University, the horizontal compliance of the machine is much larger than the vertical component as presented in Section 2.2.1. Hence, the horizontal forming forces are selected as the minimization variables in the following analysis. The variation of the vertical forming forces for different $\overline{T_g}$ and $\theta$ shares a similar trend with the horizontal forces and will not be presented here.

As shown in Figure 3.18, the horizontal forming forces increase from 100 $N$ to more than 1,500 $N$ as $\overline{T_g}$ is decreased from 1.0 to 0.6. This reveals that $\overline{T_g}$ is the major factor in regard to the increase in the horizontal forming forces. When the sheet metal is dramatically squeezed by the forming and the supporting tools, due to the incompressibility of plastic deformation, a large hydrostatic pressure is generated between the tools and the sheet. These large forces cause material instability, such as buckling, when $\overline{T_g} < 0.6$. Thus, the stable angle is restricted by the maximum amount of feasible squeeze.

The influence of $\theta$ on the forming force is small compared to $\overline{T_g}$. In the bending-dominant region and competing region ($\overline{T_g} \geq 0.8$), the forming tool force is maintained at low levels; less than 500 $N$ for all levels of $\theta$ values. On the other hand, the bottom tool force decreases slightly when $\theta$ is increased. In the squeezing region ($\overline{T_g} \leq 0.7$), the forces are maintained at a high level regardless of the change in $\theta$.



**(a)**



**(b)**

**Figure 3.18 Mesh surface for the horizontal forming force of (a) forming tool, (b) supporting tool**

To minimize the forming forces, a small $\overline{T_g}$ should be avoided. In other words, the tool gap should be larger in order to reduce forming forces. More generally, both large $\theta$ and small $\overline{T_g}$ are capable of forming a large wall angle. In the bending-dominant region, a higher wall angle is achievable primarily due to the large $\theta$ rather than a small $\overline{T_g}$. The increased $\overline{T_g}$ should be considered if one desires relatively low forming forces. If the forming force is a major concern in the design stage, then it is preferable that the tool position parameters, $\overline{T_g}$ and $\theta$, be chosen in the bending-dominant region or the competing region.

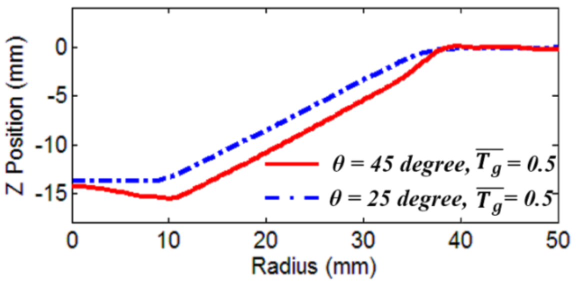
Similar to the results presented for the angle variation, the response surface presented in Figure 3.18 can also be directly utilized as a discrete form of the penalty function $L$. To optimize the tool path parameters based on the forming forces, the corresponding forming forces for all the feasible solutions of $\theta$ and $\overline{T_g}$ can be predicted based on Figure 3.18(a) or Figure 3.18(b). The solution with the minimal angle variation should be selected as the optimal solution. Otherwise, one can also simply choose the one with a small $\overline{T_g}$, since a strong correlation between $\overline{T_g}$ and the forming force has been observed as stated above. It is noteworthy that the response surface model presented in Figure 3.18 can also serve as an off-line forming force prediction model in the compliance compensation method mentioned later in Section 3.3.3.

### ***3.2.7. Experimental Verification***

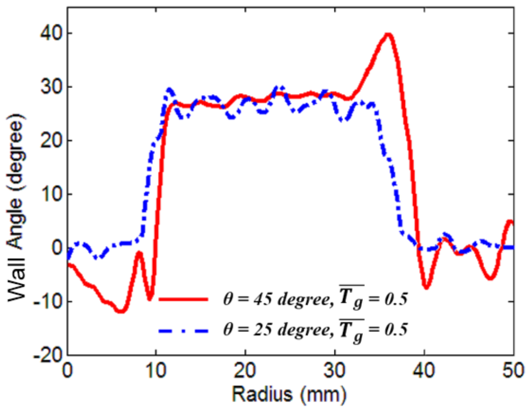
The simulation results in the previous sections provided a detailed analysis of the effects of the relative tool positions on part geometry. In this section, experimental studies are used to validate the analytical model, particularly with a focus on the general trends.

Several experiments were conducted to verify the simulation predictions. The forming tool positions are kept the same for all toolpaths used. Varying $\theta$ and $\overline{T_g}$ change the supporting tool positions. The travel pitch, tool diameters, sheet thickness and all other parameters are maintained with respect to the simulations. During experimentation, the machine compliance is a key factor which could not be ignored. The large force in the forming process deflects the tools thereby enlarging the tool gap (increasing $\overline{T_g}$), which will be explained in detail in Section 3.3. However, currently the experimental results are only analyzed qualitatively to verify the general trend observed in the simulation analyses.

Experimental results for both the cross-section view with $\theta$ = 45 $^o$ and $\overline{T_g}$ = 0.5 and wall angle profile along the radius are plotted in Figure 3.19. Note that the variation in wall angle shown in Figure 3.19(b) is artificial due to the post-processing error while laser scanning the part. The stable angle was taken to be approximately 28$^o$. Referring to $\theta$ = 45$^o$, considerable over bending exists: the peak angle was measured to be 12$^o$ more than the stable angle. If a small $\theta$ (i.e., $\theta$ = 25 $^o$) is utilized, there is no over bending phenomenon offered by the supporting tool because $\theta$ is approximately the same as the stable angle. With the exception of the peak angle, the general angle profile is very similar between the two experimental scenarios. This reveals that in the squeezing-dominant region where $\overline{T_g}$ is small, the stable angle is maintained, even for different $\theta$.



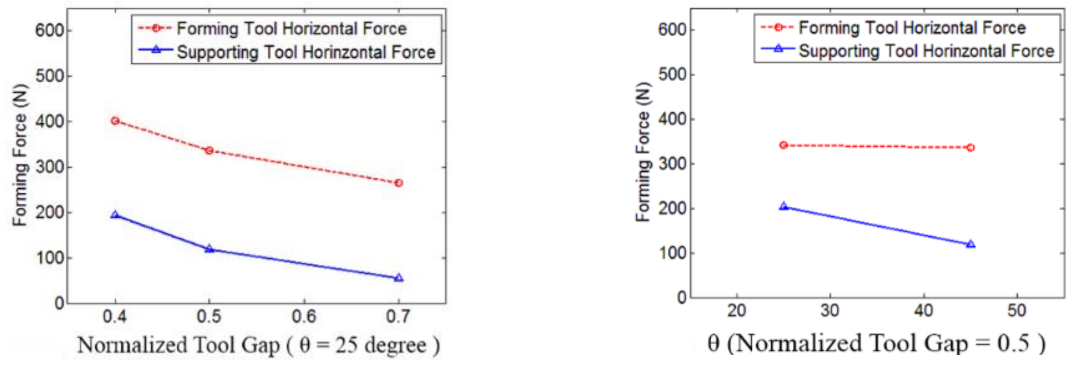
**(a)**



**(b)**

**Figure 3.19 Experimental results (a) cross section, (b) angle variation in cross section**

The trend of the forming force can also be validated by the experimental results. As shown in Figure 3.20, when $\overline{T_g}$ is decreased while $\theta$ is kept the same, the forming forces dramatically increase. However, when $\theta$ is changed, the forming tool force does not vary much while the supporting tool force decreases slightly. These results coincide well with the trends found from the simulation responses regarding the forming forces. Note that the forming forces in the experiments are smaller than the simulation ones because the tool compliance is not negligible in the experimentation. The analysis of the effect of tool compliance will be discussed in Section 3.3, while the current subsection is more focused on qualitatively analyzing the general trends as previously discussed.



**(a)** **(b)**

**Figure 3.20 Horizontal forming forces in the experiments**

## 3.3. Off-line Compliance Compensation in ADSIF

As reviewed in Chapters 1 and 2, the machine tool utilized in the DSIF process is deflectable and the deflections will induce tool position deviations, which lead to either geometric errors and/or loss of contact. For ADSIF, a similar effect can be noted when comparing the simulation and experimental results presented in Section 3.2.7. In the simulation model, the tool is treated as a rigid body without any compliance. However, machine compliance and tool compliance always exist in the real case. The large forming forces will deflect both the machine elements and the tools, which change the relative tool position and enlarge the tool gap compared to the desired input. This enlarged tool gap will reduce the squeeze effect between the two tools and affect the final formed geometry. It is noteworthy that the tool compliance induced error in ADSIF is caused not only by the absolute position deviation from the forming tool but more importantly, by the relative tool position change of both tools, since $\theta$ and $\overline{T_g}$ directly determine the local geometry of the formed part.

To compensate the error caused by tool compliance, the tool compliance compensation algorithm mentioned in Chapter 2 can be implemented on both the forming and supporting tools, so that all the possible deflections will be automatically corrected for on a real-time basis. However, implementation of the tool compliance compensation always involves additional setup and maintenance costs for the load cell, A/D cards and so forth. Tha addition of a force signal into the system would violate the initial objective of ADSIF, which is to maintain tool contact and tight geometry control with a pure displacement control strategy. Furthermore, the implementation of the tool compliance algorithm is not recommended on both of the tools simultaneously due to potential stability issues. The reason for the potential instability is the uncertainty in the characterization of the tool stiffness value, as already discussed in Section 2.3.4.

The objective of this work is to establish a general method that reveals the influence of tool deflections in the ADSIF process and compensate the error via a pure displacement control method. Here, tool deflection includes the deflections due to the overall compliance of both the forming tools and the machine. In Section 3.3.1, an analytical study is presented to demonstrate the dramatic effects that tool compliance can have on ADSIF. Then, a simulation investigation is performed to verify the influence of the tool compliance in Section 3.3.2. A full toolpath optimization scheme is stated in Section 3.3.3. Additionally, a validation experiment is conducted that demonstrates that ADSIF improves geometric accuracy as compared to DSIF for the conical part of the interest in Section 3.3.4.

### *3.3.1. Analysis of Tool Compliance Induced Error*

In ADSIF, the squeeze and bending induced by the two tools deform the metal locally and result in the final shape of the part. The squeeze effect mainly depends on the tool gap between two tools while the bending effect depends on the relative angle between the tools as stated in Section 3.2. Different combinations of the tool gap and the positioning angle will inherently have a substantial effect on the final geometry. Hence, accurate positioning of the tools is necessary. To put things in dimensional perspective, the thickness of the virgin sheet metal is approximately 0.5 to 1 *mm*, which by definition, is in the same range as the tool gap. Additionally, the total tool deflection is *of the same order of magnitude* as the tool gap for common load types, which is why it is imperative that tool and machine compliance are accounted for.

To further investigate the specific relationship between tool deflection and the change of the relative tool position parameters, $\overline{T_g}$ and $\theta$, and an analytical derivation are introduced and presented. In the following derivations, $\overline{T_g}$ is simplified to $T_g$ for notational convienience. Before the detailed derivation, several assumptions are first introduced with detailed explanations:

*Assumption 1:* The primary tool deflection occurs in the horizontal plane, which is perpendicular to the tool shaft. The compliance along the tool shaft is assumed negligible. The shaft of a forming tool is viewed as a cantilever beam under a point load at the end, and thus the major displacement is in the horizontal plane due to bending. If the machine compliance is of or beyond the magnitude of the tool compliance, this assumption needs to be validated by experimental measurements. Note that this assumption is validated for the NU DSIF prototype machine as discussed in Section 3.2.1

*Assumption 2:* The horizontal forming force affecting transverse tool bending will be predominately in the radial direction (see Figure 3.21). This is particularly true when there is an induced squeeze on the material between the tools along the radial direction. The tangential forming forces, on the other hand, are related to friction and assumed to be small due to the use of lubricants or a rotating tool.

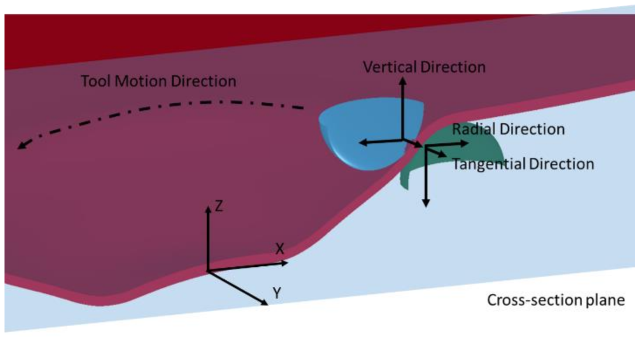
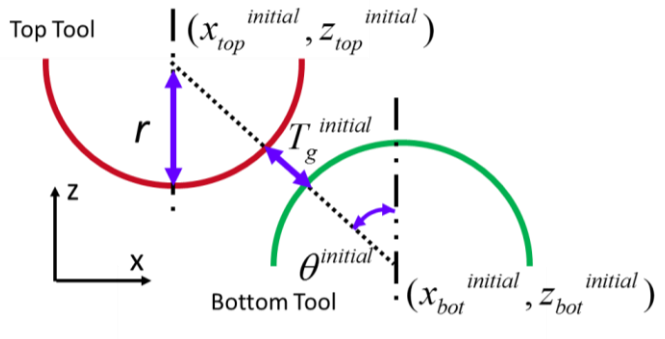


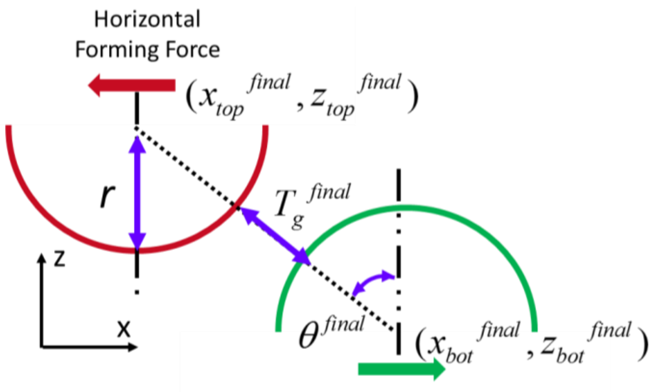
**Figure 3.21 Definition of different directions in ADSIF**

*Assumption 3:* The tool deflection caused by compliance is at least one order of magnitude smaller than the radius of the tool. In a typical incremental forming setup, the radius of the forming tool is usually at least five times larger than the sheet thickness while the tool deflection is comparably smaller than the sheet thickness.

Given the above assumptions, the tool deflection can be analyzed from a 2D study of the X-Z cross-section (Figure 3.22). The notation chosen to illustrate the initial and resultant tool positions after accounting for tool deflection is given in Figure 3.22(b). Notice that this study highlights the importance of relative tool position, rather than the absolute position of the two tools since these relative toolpath parameters have the most significant influence on tool deflection and, consequently, part geometry.


Top Tool
$(x_{top}^{initial}, z_{top}^{initial})$
$r$
$T_g^{initial}$
z
x
$\theta^{initial}$
Bottom Tool
$(x_{bot}^{initial}, z_{bot}^{initial})$

**(a)**


Horizontal Forming Force
$(x_{top}^{final}, z_{top}^{final})$
$r$
$T_g^{final}$
z
x
$\theta^{final}$
$(x_{bot}^{final}, z_{bot}^{final})$

**(b)**

**Figure 3.22 Configurations of relative tool position in ADSIF (a) before tool deflection and (b) after tool deflection**

Next, the mechanics of the process is investigated in terms of relative tool positions while accounting for variations due to compliance. The tool gap before and after tool compliance can be presented as:

$$T_g^{initial} = \sqrt{(x_{bot}^{initial} - x_{top}^{initial})^2 + (z_{bot}^{initial} - z_{top}^{initial})^2} - 2r \tag{3-6}$$

$$T_g^{final} = \sqrt{(x_{bot}^{final} - x_{top}^{final})^2 + (z_{bot}^{final} - z_{top}^{final})^2} - 2r \quad (3\text{-}7)$$

where $r$ is the radius is the tool.

As stated by *Assumption 2*, the tool deflection in the vertical direction is negligible, i.e.:

$$z^{final} = z^{initial} \quad (3\text{-}8)$$

On the other hand, the tool deflection in the horizontal direction, i.e., $\Delta$ is not negligible and it is defined as the change of the X coordinate of the tool due to the forming force in the X direction. Note that $\Delta$ is composed of the deflection of both tools ($\Delta_{top}$ and $\Delta_{bot}$ ) since both of the tools are under the effect of forming forces.

Substituting Equation 3-8 into Equation 3-7, one arrives at:

$$T_g^{final} = \sqrt{(\Delta + x_{bot}^{initial} - x_{top}^{initial})^2 + (z_{bot}^{initial} - z_{top}^{initial})^2} - 2r \quad (3\text{-}9)$$

By noting the similarity between Equation 3-9 and Equation 3-6, one can write:

$$T_g^{final} = \sqrt{(T_g^{inital} + 2r)^2 + 2\varepsilon(x_{bot}^{initial} - x_{top}^{initial}) + \Delta^2} - 2r \quad (3\text{-}10)$$

Considering the geometrical relationship between the position angle and the tool position as shown in Figure 3.22(a), one can arrive at:

$$x_{bot}^{initial} - x_{top}^{initial} = (T_g^{initial} + 2r)\sin\theta^{initial} \quad (3\text{-}11)$$

Substituting Equation 3-11 into Equation 3-10 results in the relationship between tool gaps before and after tool deflection:

$$T_g^{final} = \sqrt{(T_g^{inital} + 2r)^2 + 2\Delta(T_g^{initial} + 2r)\sin\theta^{initial} + \Delta^2} - 2r \quad (3\text{-}12)$$

According to *Assumption 3*, the tool deflection is one magnitude smaller than the tool radius so that a further simplification can be done by Taylor expansion of Equation 3-12.

$$T_g^{final} - T_g^{initial} = \Delta \sin \theta^{initial} \tag{3-13}$$

By examining Equation 3-13, it can be seen that the influence of tool deflection on the tool gap is quite significant. The in-plane compliance will enlarge the tool gap, and thus reduce the squeeze effect imposed by the two tools. Another point is that the effect of the tool compliance is also influenced by the position angle. When a larger relative position angle is imposed, the tool gap will be altered to a larger degree after a deflection. Note that Equation 3-13 also indicates the basic principle of tool compliance compensation in ADSIF: instead of counting for the toolpath positions deviation in every global axes, only the tool gap needs to be compensated. That being said, for a given tool deflection Δ, the input tool gap should be decreased by the amount of $\Delta\sin\theta$ according to Equation 3-14., i.e.:

$$T_g^{compensated} = T_g^{\,0} - \Delta \sin \theta \tag{3-14}$$

where the superscript *0* means the direct value derived from the response surface model as shown in Figure 3.13 in Section 3.2.4, while the *compensated* means the value that after compensation.

Next, the relative position angle is studied to determine how it is affected by compliance. The tangential value of the position angle before and after deflection can be presented as:

$$\tan \theta^{initial} = \frac{x_{bot}^{intial} - x_{top}^{initial}}{z_{top}^{initial} - z_{bot}^{initial}} \tag{3-15}$$

$$\tan \theta^{final} = \frac{x_{bot}^{final} - x_{top}^{final}}{z_{top}^{final} - z_{bot}^{final}} \tag{3-16}$$

By substituting Equations 3-8, 3-9 and 3-10 into the above equations, the difference between $\tan\theta^{final}$ and $\tan\theta^{iniital}$ can be derived as :

$$\tan\theta^{final} - \tan\theta^{initial} = \frac{\Delta}{z_{top}^{initial} - z_{bot}^{inital}} \tag{3-17}$$

In a manner similar to Equation 3-13, the denominator of Equation 3-17 can be presented as a function of the position angle $\theta$, initial tool gap $T_g^{intial}$ and tool radius $r$, following the geometrical relationship presented in Figure 3.22(a):

$$\tan\theta^{final} - \tan\theta^{initial} = \frac{\Delta}{(T_g^{initial} + 2r)\cos\theta^{initial}} \tag{3-18}$$

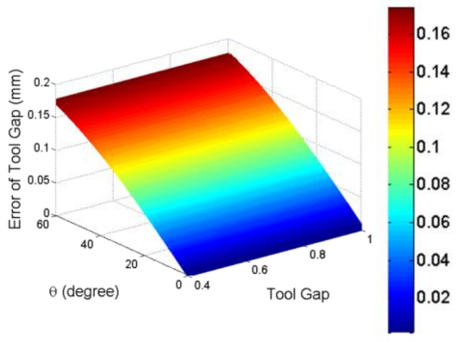
In *Assumption 3*, it is assumed that tool deflection is one order of magnitude smaller than the tool radius, i.e., $\Delta << r$. Hence, Equation 3-18 can be simplified to:

$$\tan\theta^{final} \approx \tan\theta^{initial} \tag{3-19}$$

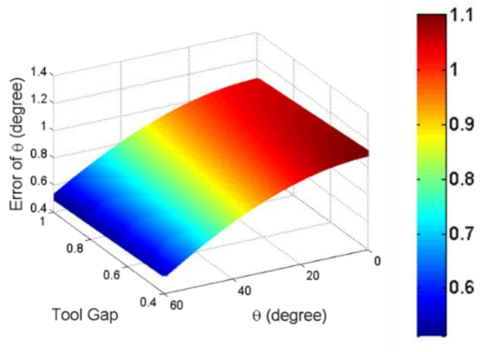
By examining Equation 3-21, it is seen that the influence of the machine compliance on the position angle is negligible, especially when the tool deflection is small compared to the tool radius.

The change in the tool gap and position angle can also be viewed in the following example. Considering a typical ADSIF setup with two 10 *mm* diameter steel tools and 1 *mm* thick sheet, the corresponding increments in $T_g$ and $\theta$ caused by a 0.2 *mm* horizontal tool deflection are presented in Figure 3.23.The deviation is defined as the difference between the tool gap (or position angle) before and after tool deflection. As shown in Figure 3.23(a), a tool deflection of 0.2 *mm* will cause an increment upwards of 0.17 *mm* depending on the position angle. On the other hand, the deviation in position angle is small.

These preliminary conclusions, i.e., that the position angle remains the same while the tool gap is enlarged due to the effect of tool compliance, will be useful when considering compliance compensations for the generated toolpaths. The tool gap should be compensated based on the forming force and the current position angle as shown by Equation 3-15. On the other hand, the position angle does not need to be separately compensated since it is not strongly dependent on machine compliance.



(a)



(b)

**Figure 3.23 Deviation of (a) tool gap and (b) position angle caused by a 0.2 *mm* tool deflection (i.e., Δ= 0.2 *mm*)**

### *3.3.2. Simulation Verification with a Deformable Tool*

The Finite Element Method (FEM) has been widely utilized to study the incremental forming process so as to develop a better understanding of the fundamental deformation mechanics. The strain-stress history of deformation, the fracture theory and/or the thickness distribution have all been within the scope of existing FEM analyses. However, most of the simulation models treat the forming tool as a rigid body. In this section, a simulation model which is capable of modeling a deformable tool is established in order to investigate the detailed influence of tool deflection.

The high-level procedure of the simulation model parallels the techniques presented by Smith et al. (Smith et al., 2013) and the one utilized in Section 3.2.2. The model was established in LS-DYNA using explicit time integration. The sheet was modeled using 8-layers of reduced-integration solid elements through the thickness, while the hemispheric tool tips were modeled as rigid shell elements. The remainder of each tool was modeled with deformable beam elements. Same as the one presented before, the von-Mises yield criterion and the Voce hardening law were utilized to approximate the sheet metal material, AA5754-O in this case.

In order to capture the tool deflection due to compliance in the simulation, a deformable tool body composed of elastic beam elements was attached to rigid shell elements on either side of the tool (Figure 3.24). To be clear, the tool hemisphere is still modeled with rigid elements to maintain a stiff contact between the tool surface and sheet metal. The deformable tool body was modeled by five linear elastic beam elements, and the compliance was determined by the area moments, the total length of the tool and the Young's modulus of the tool material. The compliance can be changed by merely adjusting the Young's modulus of the beam material. In our case, the Young's modulus is set to be 200 *GPa* and the length of the tool-shaft was chosen so as to achieve an effective compliance of 1 *um*/*N*, which is approximately equal to that of the DSIF prototype machine setup at Northwestern University. Additionally, a velocity boundary condition was enforced on the rigid tool holders instead of the displacement taken directly from the toolpath. A $C_0$ continuous velocity profile was necessary in this model to avoid infinite acceleration between toolpath points, which could cause instability. A relative damping coefficient was also added between the tool body and the rigid body to damp out any artificial oscillations arising from the mass scaling and increased tool speed.

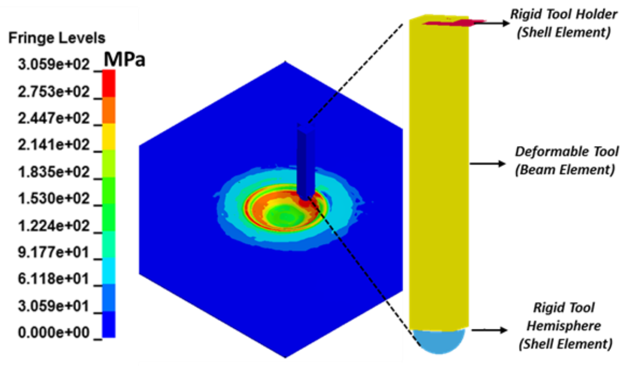


**Figure 3.24 Von-Mises stress contour in the simulation model**

Several simulation cases were conducted for a simple truncated cone shape consisting of a 12 *mm* inner radius and 24 *mm* outer radius. This cone shape, as shown in Figure 3.24 and Figure 3.25, can well represent the relationship between the geometric wall angle and the input toolpath parameters. The tool diameter was chosen to be 10 *mm*, and the initial thickness of the metal is 1 *mm*. The cross-section of the simulation with rigid and deformable tools corresponding to the same initial tool gap and position angle ($T_g = 0.7$ *mm* with $\theta = 45^o$) is presented in Figure 3.25. Note that with a deformable tool, the achieved wall angle is smaller than that observed from the rigid tool, which is primarily due to the enlarged tool gap caused by the tool deflection.

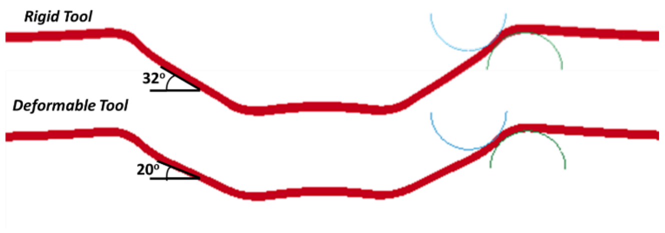


**Figure 3.25 Cross-section comparison between rigid tool and deformable tool simulation models**

The influence of the deformable tools is even more dramatic if the initial (i.e., desired) tool gap is chosen to be relatively small. The case shown in Figure 3.26 is with $T_g = 0.5\ mm$ and $\theta = 25^o$. If a rigid tool model is implemented, the small tool gap will induce a local out-of-plane buckling instability in front of the tools and cause an unrealistic phenomenon. On the other hand, if a deformable tool is implemented, the tool gap will be enlarged, and a realistic geometry will be simulated.

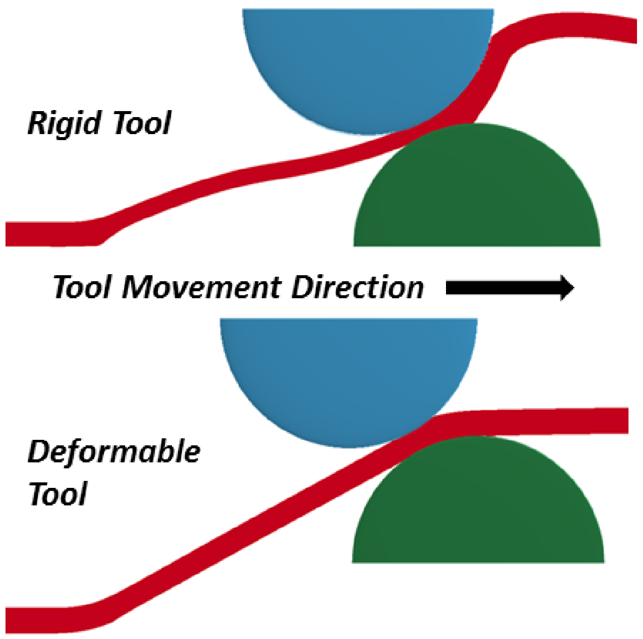


**Figure 3.26 Local deformation comparison between rigid tool and deformable tool simulation models**

### *3.3.3. Toolpath Planning Algorithm Considering Tool Compliance*

With the influence of tool compliance on the relative tool position as explained above, a detailed toolpath planning procedure considering tool compliance compensation is established as shown in Figure 3.27. For a given desired part, the local wall angle is extracted for each point in the contour or spiral toolpath, the operation of which can be successfully implemented by the AMPL toolpath software (Moser, 2018) as mentioned in Section 2.4.

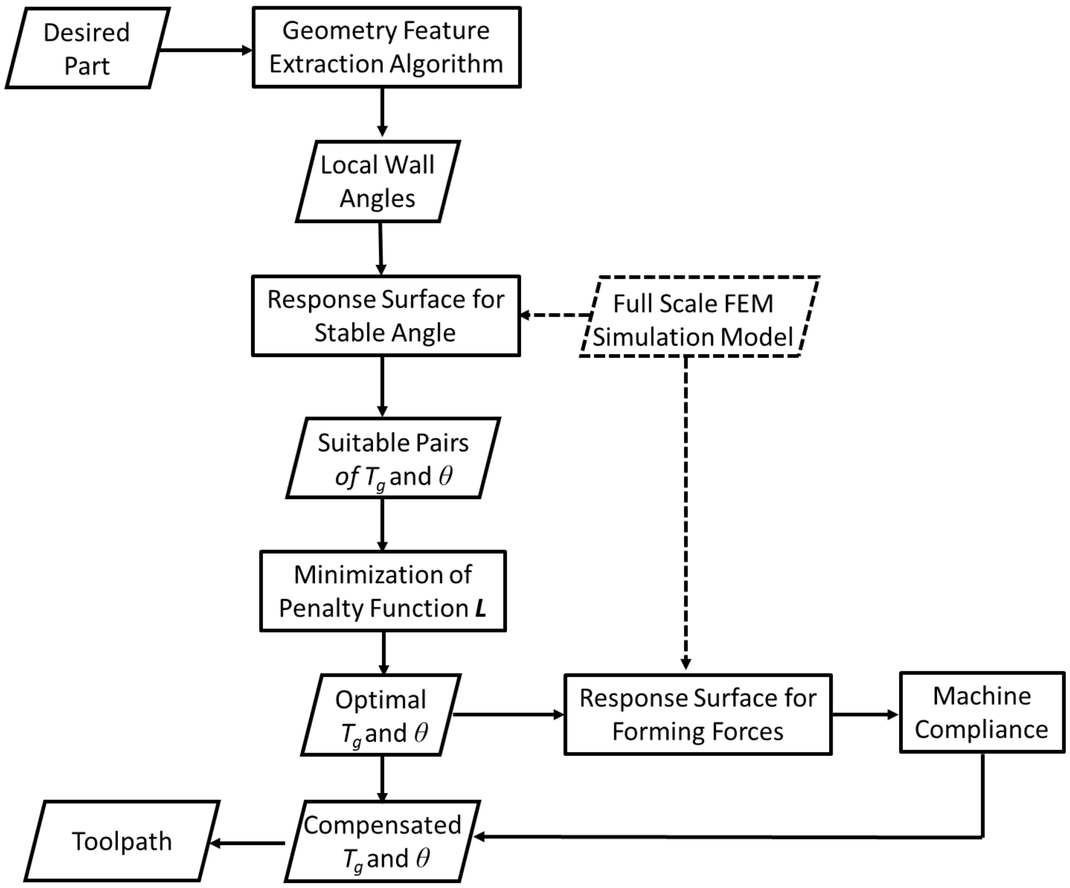

Desired Part
Geometry Feature Extraction Algorithm
Local Wall Angles
Response Surface for Stable Angle
Full Scale FEM Simulation Model
Suitable Pairs of $T_g$ and $\theta$
Minimization of Penalty Function ***L***
Optimal $T_g$ and $\theta$
Response Surface for Forming Forces
Machine Compliance
Toolpath
Compensated $T_g$ and $\theta$

**Figure 3.27 Toolpath planning algorithm flowchart for ADSIF**

Then for each value of local wall angle, the possible pairs of $T_g$ and $\theta$ are inversely looked up based on the response surface model, given in Section 3.2.3. Note that multiple solutions will exist, and further minimization is desired to choose the proper value. As discussed before, either the forming force or the angle variation could be set as the penalty function $L$ for the minimization operation. Another possible criterion for the optimization operation is the use of continuity between sequential tool path points to avoid large motion oscillations.

From the observation in Section 2.3.5 and 2.3.6, it is learned that among all the possible solutions, the smaller tool gap (and small position angle) will reduce angle variation while inducing large forming forces. On the other hand, large position angles will induce significant angle variations but maintain the forming forces at a reasonable value, as demonstrated in Figure 3.28 where the possible solutions are obtained from the intersection with a z plane at the desired wall angle value according to Section 3.2.4. Note that the above observation means that the angle variation and forming force are two conflicting criteria which cannot be simultaneously satisfied. In the following experiment, a tradeoff between these two criteria is enforced via setting the forming force as the minimization criterion while the maximum angle variation being limited to 5$^{o}$. However, there exists no ideal penalty function *L*, and the detailed employment should be determined by specific user requirements.

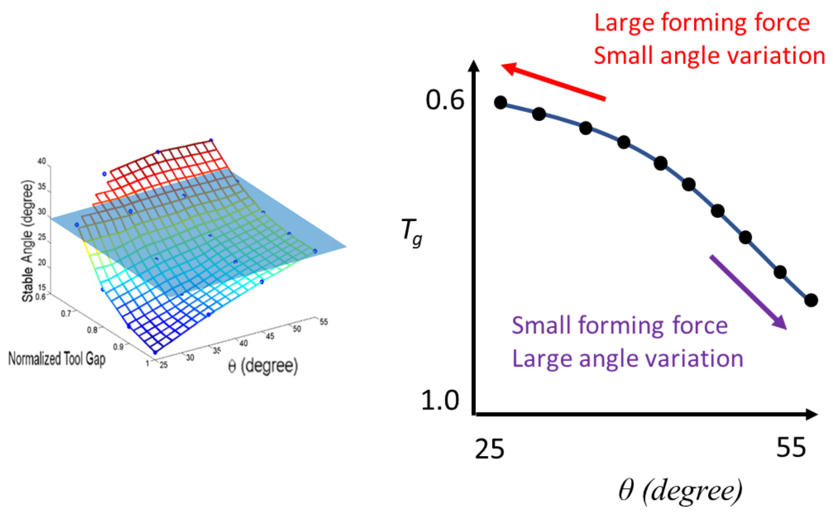


**Figure 3.28 A conceptual presentation of the choice of different $T_g$ and $\theta$ among all possible solutions**

Once the optimal $T_g$ and $\theta$ are retrieved, the corresponding forming forces can also be predicted via the FEM-simulation based response surface model of the forming forces. The tool deflection $\Delta$ is then determined by the specific machine compliance value, and $T_g$ can be compensated based on Equation 3-15. Once $T_g$ and $\theta$ are calculated for all the toolpath points, the whole toolpath planning algorithm for ADSIF is finished.

This proposed algorithm is a pure feedforward controller in the sense that no feedback signal is involved at all and the algorithm can be implemented off-line prior to any physical forming operation. The user only needs to run a finite number of simulations to define the response surface for the stable angle and forming forces. Then the whole framework can be implemented into a complex geometry with different machine stiffness and tool radius/sheet thickness (if the ratio of thickness versus tool radius is kept the same). To utilize the algorithm for different materials and/or pitch sizes, a similar FEM model might need to be run to retrieve a new response surface. It is noteworthy that the simulation is not the only way to establish the response surface model. For example, one can form similar cones with different $T_g$ and $\theta$ combinations, record the forming forces, and then establish a pure empirical-based model.  However, these multiple physical tests might be costly to conduct and some experimental errors, which are usually induced by the clamping force variation, tool misalignment, etc, cannot be excluded as in the FEM simulation model.

### *3.3.4. Experimental Verification*

In the following experimental verification, the same experimental setup conditions were used as in the simulation study: tool diameter of 10 *mm* and 1 *mm* sheet of AA5754-O. The forming area is approximately 250 by 250 *mm*. An axisymmetric part with varying wall angles (Figure 3.29) was chosen in an effort to demonstrate the effects on geometric accuracy improvement by implementing the ADSIF toolpath planning algorithm developed above. What is also emphasized in this experimental study is the influence of machine compliance on the resulting part geometry when using ADSIF.

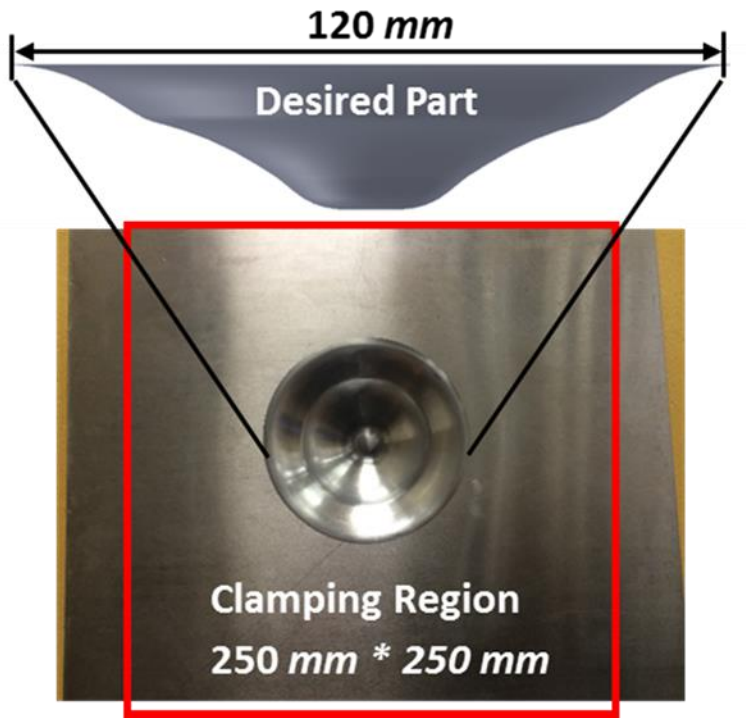


**Figure 3.29 Experimental setup to show the formed geometry and the clamping region**

First, an uncompensated ADSIF toolpath is generated for this axisymmetric part, in which the tool path parameters were derived from the response surface model without any compensation operation. Then, for each point of the uncompensated toolpath, the $T_g$ is reduced based on the predicted forming forces and measured machine stiffness. In most regions of the compensated toolpath, $T_g$ decreased to account for the total tool deflection, as shown in the comparison of the $T_g$ along the part radius between uncompensated and compensated ADSIF toolpath in Figure 3.10. Conventional DSIF without any compensation was also trialed and then compared against the developed ADSIF toolpaths.

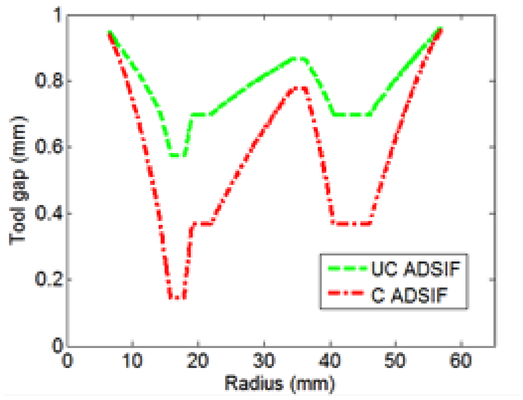
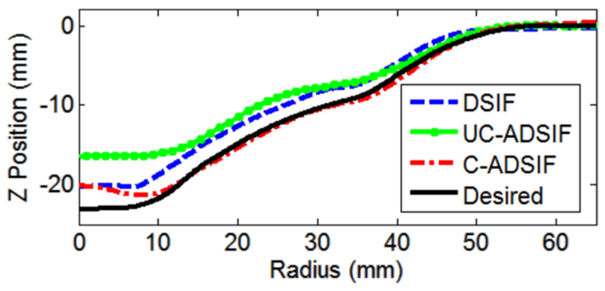


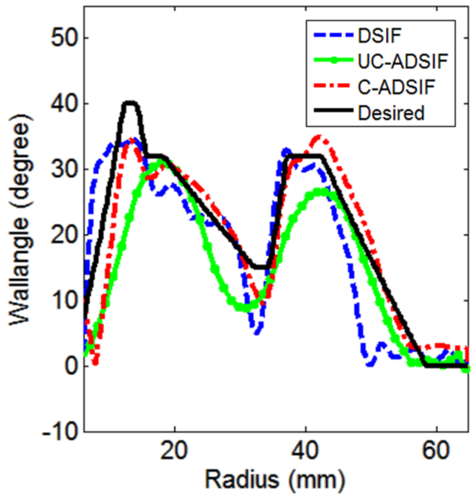
**Figure 3.30 Tool gap ($T_g$) vs. part radius location (from the part center) for the Uncompensated-ADSIF (UC-ADSIF) and Compensated-ADSIF (C-ADSIF)**

The comparison of the cross-sections between the conventional DSIF part and the compensated/uncompensated ADSIF part is presented in Figure 3.31(a) along with the desired geometry. Due to global bending and machine compliance, the conventional DSIF part exhibited geometric errors as high as 4 *mm*, while the overall geometric profile of compensated ADSIF agrees well with the desired geometry, especially along the wall. However, if the toolpath parameters of ADSIF are uncompensated, the part is much shallower than the desired one, and the geometric accuracy is largely compromised.

The wall angle corresponding to the radial position is plotted in Figure 3.31(b). The wall angles between DSIF and compensated ADSIF are somewhat similar, though the compensated ADSIF toolpath does a better job in capturing the desired geometry in the transition zones where the wall angle suddenly changes. On the other hand, the uncompensated ADSIF part generally has a lower wall angle profile, due to the enlarged tool gap ($T_g$) related to the uncompensated forming forces.



(a)



(b)

**Figure 3.31 Comparison with experimental results for DSIF, Uncompensated-ADSIF (UC-ADSIF) and Compensated-ADSIF (C-ADSIF) regarding (a) cross section view (b) angle variation**

## 3.4. Summary and Discussion

This chapter has presented a detailed analysis of the effect of relative tool positions of ADSIF on the final part geometry. The relative tool positions are defined using $\theta$ and $\overline{T_g}$, which are related to the bending and squeezing phenomena, respectively. A simple cone shape with 19 different tool positioning parameters were simulated and analyzed, based on which the response surface model is retrieved. The details deformation mechanics and the forming forces are also investigated.

This chapter also presents a study on the effect of relative tool position and tool deflection on geometric accuracy in the ADSIF process. The influence of tool compliance on these toolpath parameters is first analyzed using static analysis. Additionally, a simulation method is proposed to model the tool compliance phenomenon. Combining the response surface model with the compliance compensation method, a full toolpath planning algorithm is proposed, and experiments were conducted to demonstrate the effectiveness of the algorithm. Note that the algorithm is capable of controlling the geometric error without the involvement of any feedback device or physical trial and error. To utilize the model for different process parameters, such as the pitch size, a similar FEM model just needs to be executed to retrieve the necessary response surface model.

In the future work, the focus will be placed on investigating the influence of other factors such as different materials and sheet thicknesses, which can enhance the generality of the proposed algorithm. In the simulation, a more sophisticated material model that includes kinematic hardening should also be implemented. A thorough discussion of the future work will be presented later in Chapter 5.

# Chapter 4 On-line Springback Compensation Algorithm for DSIF

In Chapter 3, a toolpath optimization framework has been introduced to improve the part's geometric accuracy and reduce the in-process springback error for Accumulative Double-Sided Incremental Forming (ADSIF). However, ADSIF is limited in terms of the maximal achievable formable angle due to potential bulking. For instance, for a 1.0 *mm* thick AA 5754 sheet with a 5.0 *mm* radius tool and a 0.2 *mm* pitch size, the maximum formable angle is observed to be only $42.3^o$ as shown in Section 3.2.5. To form a high wall angle part with the maximal wall angle larger than the limit angle of $42.3^o$ for the ADSIF example mentioned above, a conventional DSIF process has to be utilized. However, in this case, to maintain a tight geometric accuracy and control the in-process springback error is always a challenging task due to the strong nonlinearity and history dependency of the Double-Sided Incremental Forming (DSIF) process. To address this problem, an on-line springback compensation algorithm for conventional DSIF (Loop III in Figure 1.22) is proposed in this chapter. In the compensation algorithm, the introduction of a FEM springback prediction model enables the calculation of the springback error regardless of the complexity of the part geometry, while the construction of the feedback loop informs the current state of the process and thus avoids the involvement of any process history dependency. In the following, current research on the springback error control will be reviewed in Section 4.1, including not only in Incremental Forming (IF) but also in other conventional metal forming processes. Section 4.2 will first introduce the general framework of the springback compensation algorithm, followed by two crucial ingredients of the algorithm: the FEM-based springback prediction model and the control point determination algorithm. The Section 4.3 will discuss the detail implementation of the algorithm and its experimental verifications. Finally, Section 4.4 will summarize this chapter and suggests possible future work.

## 4.1. State-of-the-Art

### *4.1.1. Springback Compensation Method in Incremental Forming*

As reviewed in Section 1.2, springback is one of the major geometric error sources in the IF process. Considerable research has been conducted to reduce the geometric error by modifying the input toolpath either based on an open-loop process planning model or closed-loop geometric feedback.

Filice et al. tried to over form the part to compensate for springback of the part in 2004 (Ambrogio et al., 2004). However, the determination of the over formed shape and the value was purely based on empirical guesses, and their application was limited to a cone-like axisymmetric part. To generalize this approach, Behera et al. proposed a feature-based framework to automatically detect different features for a complex geometry (Behera et al., 2014). Different compensation methods and toolpath strategies were applied to different features as shown in Figure 4.1 so that the shape error for an asymmetric part could be well compensated. It was claimed that a database would provide users detailed compensation functions according to feature size or depth. However, the database is still purely empirical-based and can only function well if a similar feature has been formed before.

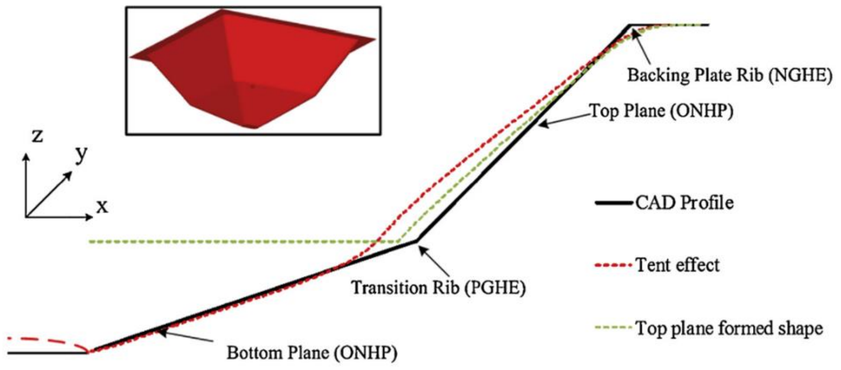


**Figure 4.1 Different kinds of geometric errors of different features**

In 2017, Reddy et al. expand the feature concepts for the SPIF and DSIF processes (Lingam et al., 2017). They also studied the influence of the feature sequence experimentally as shown in Figure 4.2, but no general conclusions have been made.

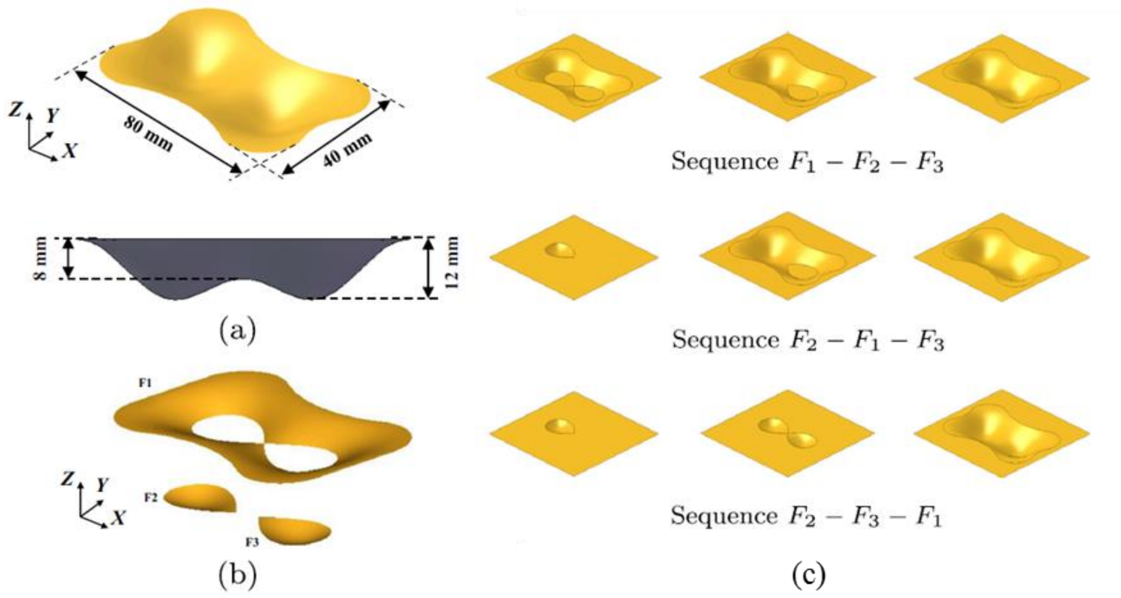

80 mm
40 mm
8 mm
12 mm
(a)
F1
F2
F3
(b)
Sequence $F_1 - F_2 - F_3$
Sequence $F_2 - F_1 - F_3$
Sequence $F_2 - F_3 - F_1$
(c)

**Figure 4.2 A sample shape with three features and possible feature sequences**

Despite the above-mentioned essentially open-loop attempts, some closed-loop springback control systems have also been implemented in the IF process. In 2005, Hirt et al. first proposed using the scanned geometry to compensate for geometric errors as shown in Figure 4.3 (Hirt et al., 2004). Note that the process is conducted off-line, and several trial and error steps are required to fully compensate the geometric error.  This makes the whole feedback process inefficient in both cost and time.

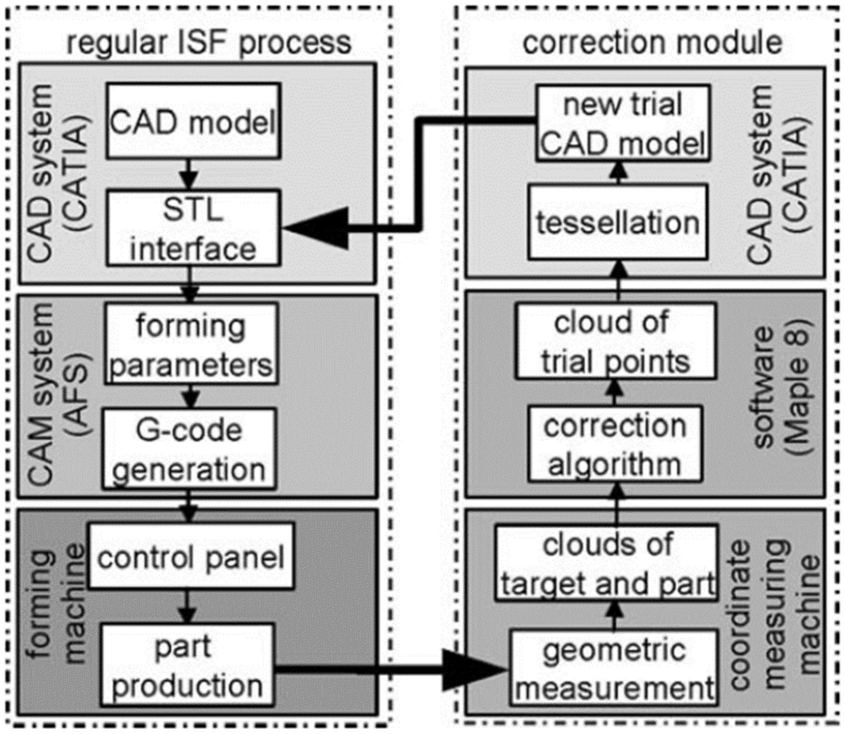


**Figure 4.3 Off-line geometric compensation from (Hirt et al., 2004)**

Following the same philosophy, Gong et al. utilized a wavelet transformation to determine the compensation function as shown in Figure 4.4 (Fu et al., 2013). They also introduce some simulation works as part of the iterative process for their algorithm to reduce the possible cost associated with physical trials. However, the efficiency of the method is still limited by the speed of the required geometric measurements and numerical simulation.

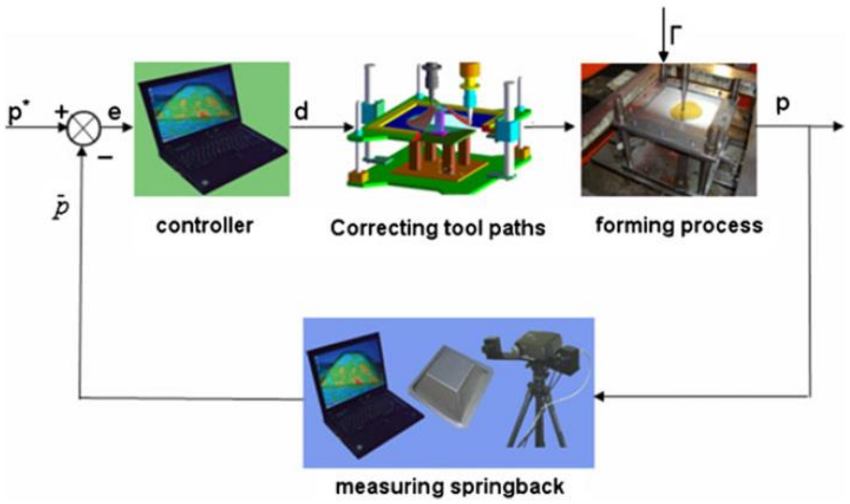


**Figure 4.4 Off-line geometric error compensation from (Fu et al., 2013)**

To avoid multiple iterations and long processing times associated with off-line control, Allwood et al. and Duncan et al. proposed on-line feedback control for SPIF (Allwood et al., 2009; Hao and Duncan, 2011). The scanned geometry is compared with the desired one at several discrete time steps, and then compensation is applied to the subsequent forming process. This proposed method can be viewed as a discrete form of the Model Predictive Control (MPC) (Garcia et al., 1989; Kothare et al., 1994). According to Allwood et al., the difficulty in the proposed work is to determine the spatial impulse responses of the process, which means how the geometry will be changed given a certain compensation value change. Their currently proposed linearized model can only work for simple axisymmetric components like a truncated cone or funnel shape, as shown in Figure 4.5.

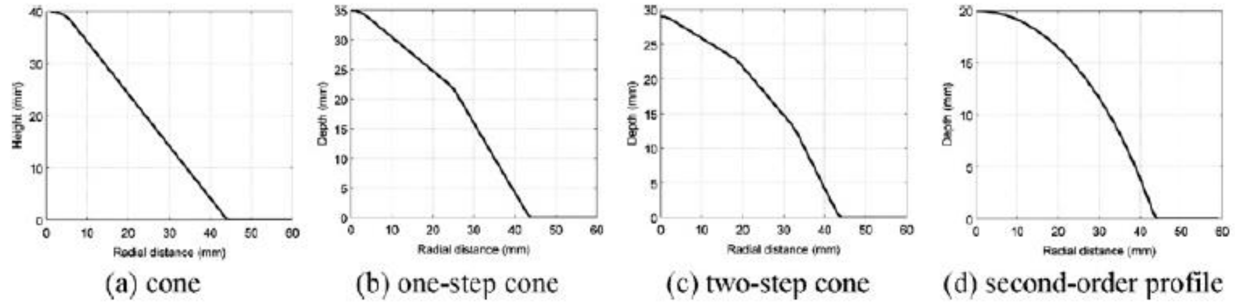


**Figure 4.5 Considered axisymmetric shapes in (Allwood et al., 2009)**

The MPC framework has also been utilized for on-line springback compensation in SPIF and TPIF by Lu et al. (Lu et al., 2016; Lu et al., 2017a; Lu et al., 2017b). The work adopted a similar set up as in Allwood's work (Allwood et al., 2009; Hao and Duncan, 2011), in which the real-time geometry measurement for the intermediate shape was retrieved by a laser scanner as shown in Figure 4.6. However, instead of an empirically defined spatial impulse responses such as in (Allwood et al., 2009; Hao and Duncan, 2011), the process model utilized in their work is rather an assumed linear interpolation function. This predefined process model is rather simple and questionable in terms of its generality for complex geometries. Moreover, the utilization of a 3D Digitizer is usually costly and time-consuming, while the accuracy is often limited due to the reflective surface of the typical sheet metal material, the existence of a lubricant during forming and the common misalignment between the measured and scanned geometries. For instance, a typical white light 3D profiler with a 400 *mm* measurement range and measurement accuracy of 250 $\mu m$ will cost more than \$ 10,000 (Mikolajek et al., 2014). Such a profiler also takes up a considerable space (around 300 *150* 100 $mm^3$) as shown in Figure 4.7(a) and is hard to directly integrate into most of DSIF systems considering that the forming tools and associated motion mechanisms occupy both sides of the machine and obstruct a clear view of the part. This is also the case with the current DSIF prototype machine at Northwestern University as introduced in Section 2.2.

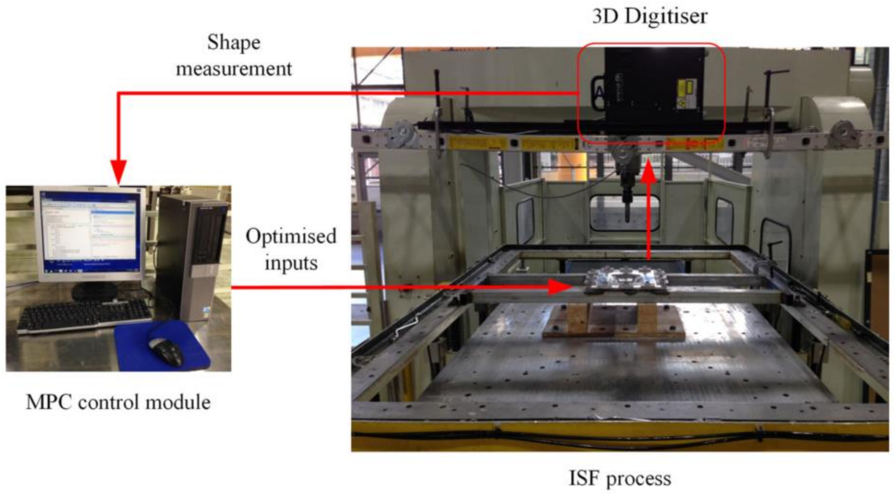
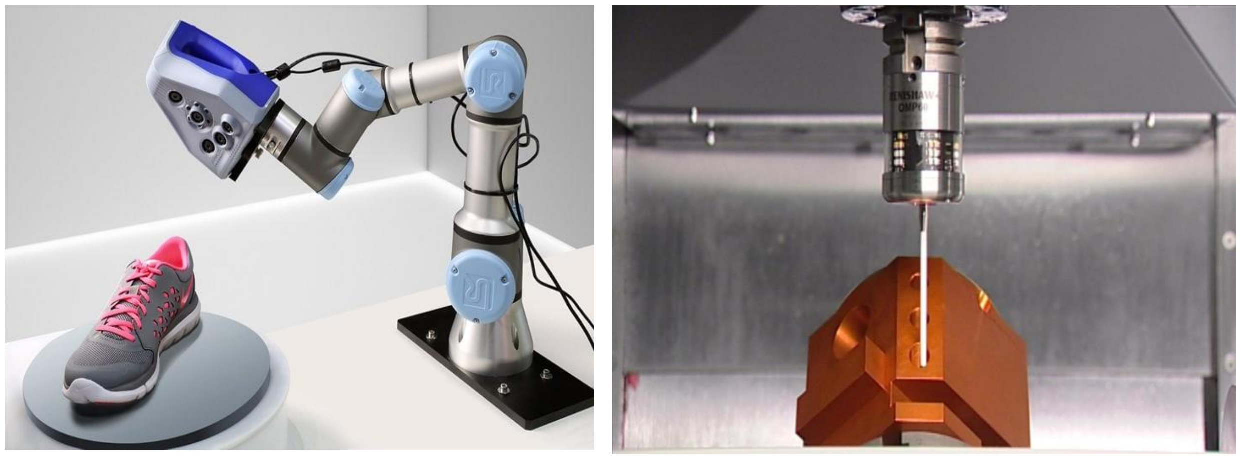


**Figure 4.6 IF closed-loop feedback control set up in (Lu et al., 2016)**

Note that there also exists some other form of geometric error measurement system in the market which could be utilized as an on-line springback measurement tool. One standard on-site measurement method is the touching-based method, such as the use of touch probes (Figure 4.7(b)) in CNC machines or Coordinate Measuring Machines (CMMs). The touch probe is usually less sensitive to the variation of the surface texture and lubrication. However, to retrieve a full 3D profile of the part, multiple attempts are needed at different touch locations which are usually quite time-consuming. Moreover, the employment of a touch probe in the forming process usually requires a change of tools or additional motion actuators, which further increases the setup time or capital cost.



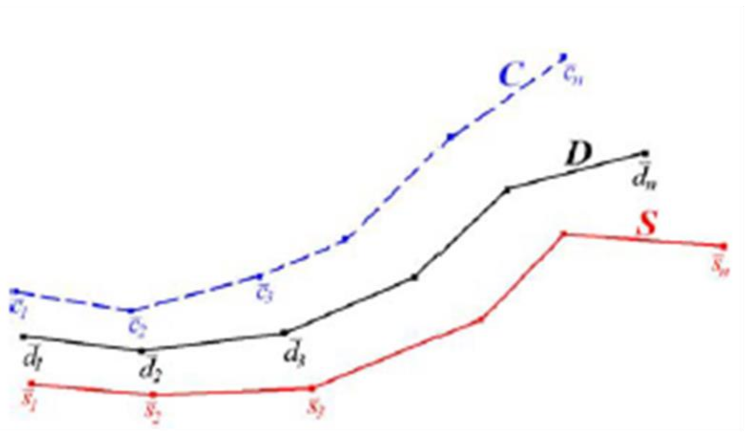
**(a)** **(b)**

**Figure 4.7 Example applications of (a) 3D object scanner (Mikolajek et al., 2014) (b) a touch probe (McMurtry et al., 1992)**

### *4.1.2. Springback Compensation Methods in Conventional Forming*

As a widely recognized existing problem in almost all forming processes, springback compensation has been a key research topic not only in IF but also in conventional sheet stamping or deep drawing processes. In this subsection, existing compensation methods in conventional forming are briefly reviewed to provide potential insights for springback compensation in IF.

There exist two common methods for springback compensation in conventional sheet metal forming processes, the Displacement Adjustment (DA) method proposed by Wagoner (Gan and Wagoner, 2004; Gan et al., 2004) and the Springforward (SF) method proposed by Karafillis (Karafillis and Boyce, 1992; Karafillis and Boyce, 1996). The DA method is a pure geometry-based method, in which the geometry of the forming shape is simply displaced in the opposite direction to the geometrical error, as shown in Figure 4.8. In the DA method, the geometry of the forming shape (**S**) needs to be measured and compared with the desired shape (**D**) to determine the value of the compensation error. Usually, optical shape measurements are utilized to offer a comprehensive error analysis for the whole part. However, optical measurements are usually a slow process and take an effort to align the scanned geometry with the desired shape. Also, particularly for DSIF, the additional tool makes optical measurements even harder. To solve this measuring issue and also the avoid material costs associated with physical iteration experiments, FEM simulation has been widely utilized during the past decade as an alternative for springback prediction of arbitrary shapes. While apparently simple in concept, the accuracy of springback prediction has proven to be limited for various reasons, such as numerical sensitivity, physical sensitivity, and inadequately characterized material behavior under reverse loading and unloading conditions.



S: Formed Shape;
D: Desired Shape;
C: Compensated Shape;
a: Compensation Factor;
j: Iteration Number;

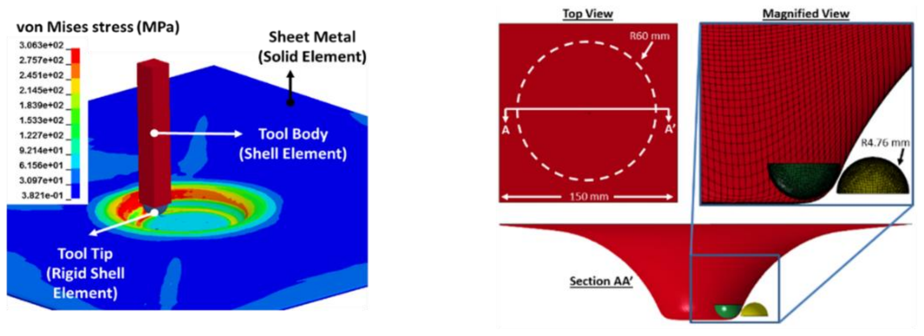
$$\mathbf{C} = \mathbf{D} - a(\mathbf{S} - \mathbf{D})$$

or

$$\mathbf{C}^{j+1} = \mathbf{C}^{j} - a(\mathbf{S}^{j} - \mathbf{D})$$

**Figure 4.8 Principle of the Displacement Adjustment Method (Gan and Wagoner, 2004)**

However, even assuming that springback can be accurately predicted, currently available simulation schemes for DSIF usually take days to generate reliable results, which is far beyond the capability of industry, even with advanced computational power. For example, an accurate solid element-based simulation model, shown in Figure 4.9, takes ten days to finish for a simple cone with a 60 mm diameter with 8 CPUs (Ren et al., 2016). Even a simplified shell element based model is 120 times slower than the real forming process if a single CPU is utilized (Moser et al., 2016a).



Full Scale Solid Element Model

Reduced Domain Shell Element Model

**Figure 4.9 NU developed full-scale simulation models (Moser et al., 2016a; Ren et al., 2016)**

Instead of simply compensating springback based on geometric deformation, the SF method uses the cause of springback, the internal stresses or the forces. The procedure is presented by the block-diagram in Figure 4.10 where $\mathbf{C_n}$ denotes the forming shape in the n-th iteration and $\mathbf{D}$ is the desired shape. Figure 4.11 highlights the difference between the DA (Figure 4.11(a)) and SF methods (Figure 4.11(b). Unlike the DA method, the SF method predicts the part springback value based on a simplified purely elastic simulation in Step 3 in Figure 4.11 (Lingbeek et al., 2008), which could be much faster in terms of computational speed. If the full-scale simulation in Step 2, which is utilized to derive the current load can be avoided, the SF method can be much more computationally efficient than the DA method.

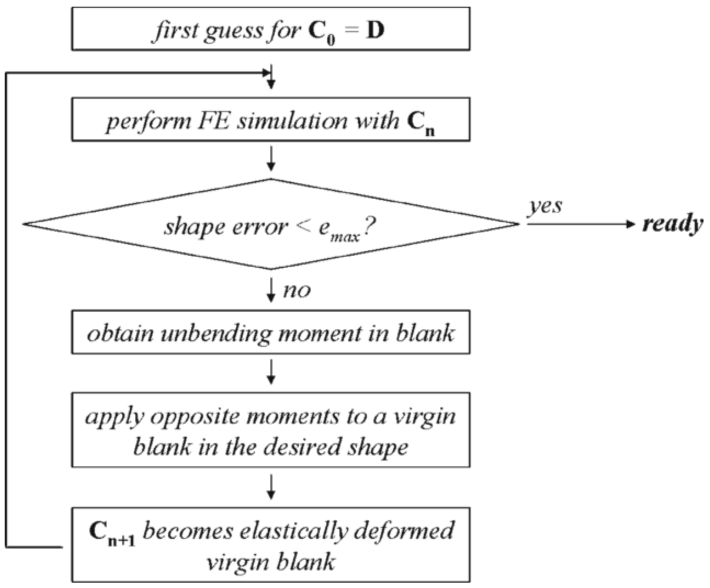

first guess for $\mathbf{C_0} = \mathbf{D}$
perform FE simulation with $\mathbf{C_n}$
shape error < $e_{max}$?
yes
ready
no
obtain unbending moment in blank
apply opposite moments to a virgin blank in the desired shape
$\mathbf{C_{n+1}}$ becomes elastically deformed virgin blank

**Figure 4.10 Principle of the Springforward (SF) method (Karafillis and Boyce, 1996)**

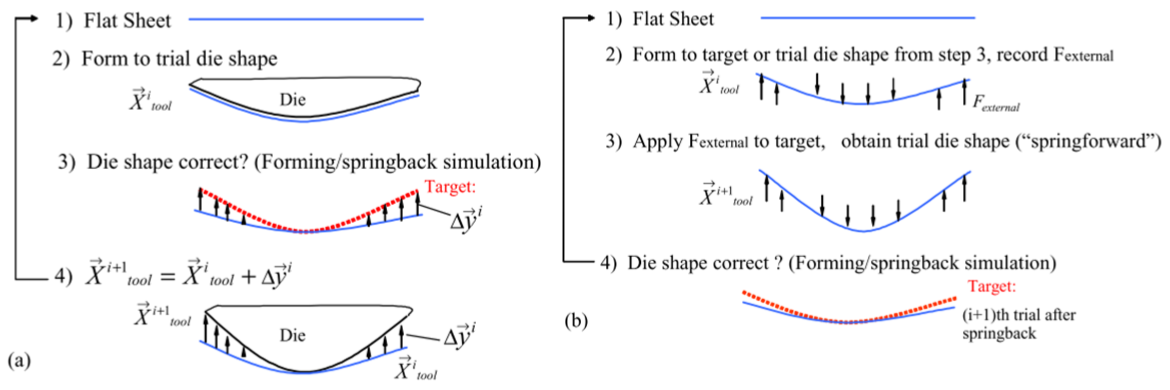

1) Flat Sheet
2) Form to trial die shape
$\vec{X}^i_{tool}$
Die
3) Die shape correct? (Forming/springback simulation)
Target:
$\Delta\vec{y}^i$
4) $\vec{X}^{i+1}_{tool} = \vec{X}^i_{tool} + \Delta\vec{y}^i$
$\vec{X}^{i+1}_{tool}$
Die
$\Delta\vec{y}^i$
$\vec{X}^i_{tool}$
(a)
1) Flat Sheet
2) Form to target or trial die shape from step 3, record Fexternal
$\vec{X}^i_{tool}$
$F_{external}$
3) Apply Fexternal to target, obtain trial die shape ("springforward")
$\vec{X}^{i+1}_{tool}$
4) Die shape correct ? (Forming/springback simulation)
Target:
(i+1)th trial after springback
(b)

**Figure 4.11 Flowchart of the DA (a) and SF method (b) (Lingbeek et al., 2008)**

### *4.1.3. Discussion and Statement of Research Objectives*

From the reviewed work, it can be concluded that despite the various attempts made to compensate the springback error without any trial and error tests, most of them are limited to simple geometric forms or past empirical observations for certain geometries and materials, which make the methods challenging to generalize for arbitrary geometric shapes and various process setups. For example, on the deep complex "shamrock" part shown in Figure 4.12, one can observe that geometric deviations in the Z-direction can be as significant as 25 % due to in-process springback. It should also be noted that the geometric error at the bottom is not uniform along the edge of the bottom surface. The reason is that the local stiffness and boundary conditions are usually different at different locations and for different features, thus causing different springback values. This phenomenon imposes the need for *a general springback compensation method to account for the different influences of features and process parameters*. The principle and the architecture of the proposed springback compensation method is firstly introduced in Section 4.2. A FEM-based simulation model to calculate the part springback value and a control point determination algorithm to select the most representative points for springback evaluation will also be discussed in Section 4.2 as part of the proposed work, followed by Section 4.3 discussing the detailed implementation of the proposed compensation algorithm and its experimental verification.

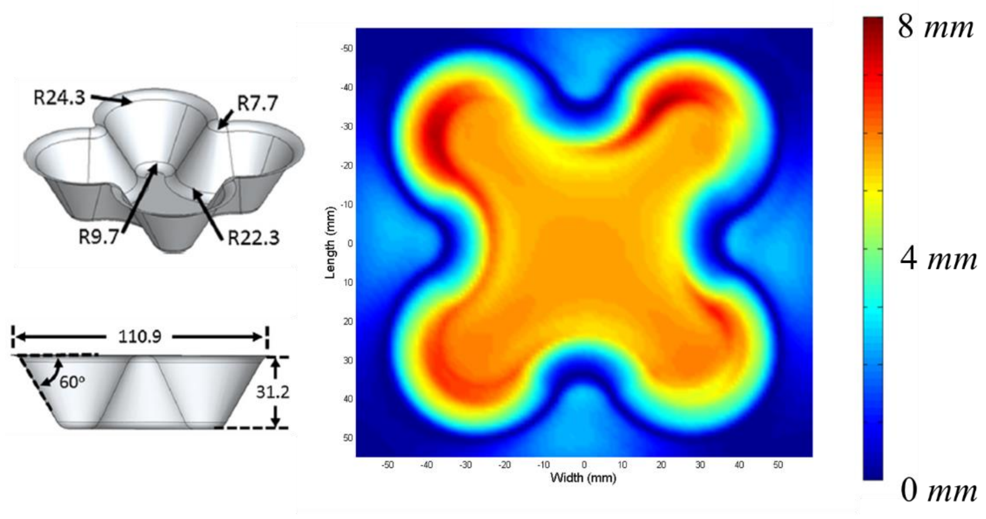


**Figure 4.12 Geometric error in the Z-direction for a shamrock part**

## 4.2. Algorithm Architecture and Design

### *4.2.1. Overall Architecture of the Springback Compensation Algorithm*

The path planning and control of DSIF, or similar flexible sheet forming manufacturing process reviewed in Section 1.1, can often be summarized as an optimization control problem. The governing equations have been discussed in Section 3.1.2 and are restated here:

$$\min_{x^c} \quad J = \sum_{i}^{N} \left\| X_i^f - X_i^d \right\|^2 + L(X_i^f, x^c{}_i) \tag{4-1}$$

$$subject\ to: X_N^f = f(X_{i=1,2,..,N-1}^f, x_{i=1,2,..,N}^c) \tag{4-2}$$

$$b(X_i^f, x^c{}_i) \le 0 \tag{4-3}$$

where the $X^f$, $X^{d,}$ and $x^c$ designate the final part geometry, desired part geometry and command tool position in the toolpath. Equation 4-1 states that the aim of the DSIF process is to achieve the desired geometry at every control point, measured by the sum of the Euclidean distance between the target and formed geometry ($X^d$ and $X^f$). $L$ stands for a penalty function, such as the sum of geometry changes or total forming forces. Since $L$ is also being minimized in Equation 4-1, the unwanted geometry changes or large forming forces are meant to be suppressed by solving this optimization control problem. Equation 4-2 represents the process model, $f$, where the formed geometry $X_N{}^f$ evolves with the input tool position $x^c$ and the previously built geometry $X^f{}_{i = 1,2,...,N}$. Finally, the $b$ function in Equation 4-3, defines the process constraints that limit the processing window.

It has been discussed in Section 3.1.2 that the strong nonlinearity and history dependency in DSIF make it extremely difficult to establish the process model, $f$, and optimize the whole manufacturing process in an open loop mode. However, this target could be achieved if on-line sensors could measure the workpiece state, and the tool path could be regularly re-planned taking into account the current state of the workpiece. With respect to geometry control, the current state of the workpiece means the geometry measurement $\overline{X^f}$ of each intermediate shape of the formed part, or the springback value, denoted as $\bar{S}$ , for the current state, where the upper bar denotes that the value is a measured one and might contains some uncertainties or errors. Note that in this case, unlike the springback error $\delta^2$ which is defined as the difference between the formed and target geometry, the springback value $S$ is defined by the difference between the commanded tool position $x^c$ and final part geometry $X_{f,}$, i.e.:

$$S_i = x_i{}^c - X_i{}^f \tag{4-4}$$

Substituting Equation 4-4 into Equation 4-1, if the penalty function $L$ is omitted, the whole optimization problem can be simplified as:

$$\min_{x^c} \quad J = \left\| x_i^c - S_i - X_i^d \right\|^2 \ for\ every\ i \tag{4-5}$$

where *i* means the sequence number of the toolpath point.

In an on-line feedback system, the springback value $S$ can be directly measured on-line as $\bar{S}$ , and then the optimal commanded tool position should simply be the sum of the target geometry and the measured springback value, i.e.:

$$x_i^c = \overline{S_i} + X_i^d \tag{4-6}$$

Following the above derivation, the overall architecture of the springback system can be established as shown in Figure 4.13, in which the springback value $\bar{S}$ is measured on-line and the command position is generated based on Equation 4-6. Specifically, at each sampling time, the tool position ($\overline{x^m}$) and forming forces ($\bar{F}$) are measured. The elastic deformation at the tool location, calculated by the fitted load-displacement function *g* from the off-line simulation, which will be introduced in Section 4.2.4, is stored as the in-process springback value $\bar{S}$. Based on this measured springback error, the commanded tool position ($x^c$) will be shifted by $\bar{S}$ from the desired geometry $X^d$ according to Equation 4-6.

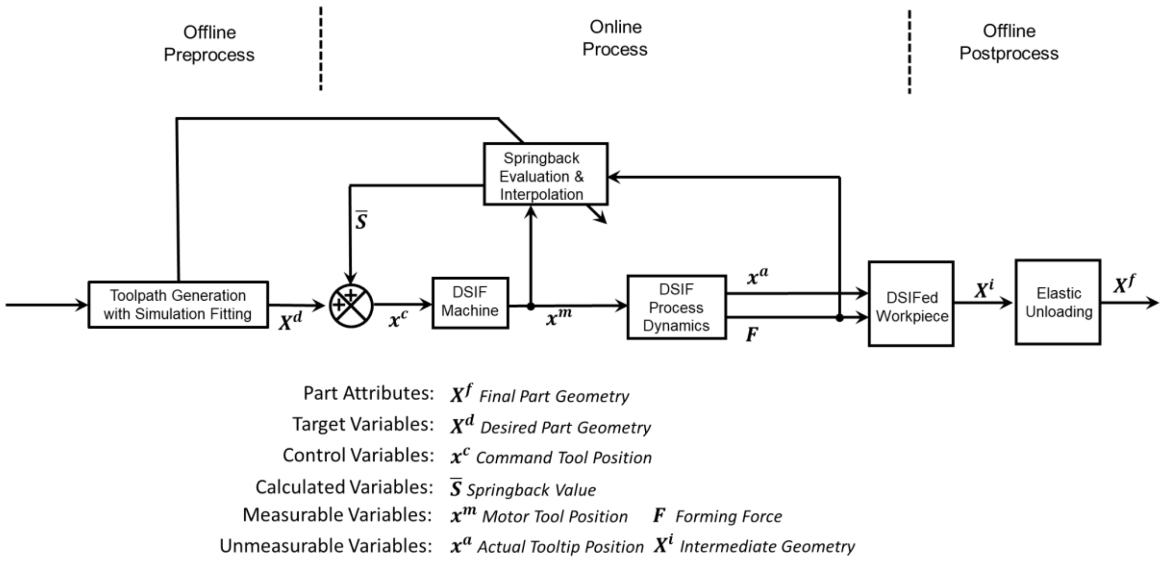

Offline Preprocess
Online Process
Offline Postprocess
Springback Evaluation & Interpolation
$\overline{S}$
Toolpath Generation with Simulation Fitting
$X^d$
$x^c$
DSIF Machine
$x^m$
DSIF Process Dynamics
$x^a$
$F$
DSIFed Workpiece
$X^i$
Elastic Unloading
$X^f$
Part Attributes: $X^f$ Final Part Geometry
Target Variables: $X^d$ Desired Part Geometry
Control Variables: $x^c$ Command Tool Position
Calculated Variables: $\overline{S}$ Springback Value
Measurable Variables: $x^m$ Motor Tool Position $F$ Forming Force
Unmeasurable Variables: $x^a$ Actual Tooltip Position $X^i$ Intermediate Geometry

**Figure 4.13 Architecture for on-line springback compensation algorithm**

As one can see from the Figure 4.13, the essence of the proposed springback compensation algorithm is to measure the springback value efficiently and accurately, so that the command toolpath can be updated during the forming process without any delay. However, as reviewed in Section 4.1.1, the current geometry measurement methods are either too costly or lack of enough efficiency, while the overall measurement accuracy is often limited due to the reflective surface finish of the metal sheet materials and the existence of forming lubricant in DSIF process.

To address these limitations, a new springback measurement method is proposed in this section, which innovatively links the measurable variables (force and position) to the desired part attributes (predicted geometric shape change). The proposed springback measurement method utilizes the forming force signal and calculates the springback value based on FEM simulations, to ascertain the springback value of the part being-formed in real-time without any intrusive machine setup changes or expensive devices (except a single load cell). On the other hand, a control point determination algorithm is also proposed. This proposed control point determination algorithm can intelligently select a finite number of discrete points to represent each individual complex features for a given geometry, so that the evaluation of the springback only needs to be conducted on these selected points instead of the whole toolpath trajectory, which significantly reduces the required computational time for the FEM simulation model and increases the efficiency of the whole springback compensation algorithm.

In the following subsections, a physical touch-based springback measurement method, called the tool lifting method, is first introduced in Section 4.2.2 due to its simplicity and similarity to the FEM-based method. Inspired by the tool lifting method, the underlying principle of the FEM-based springback measurement is thoroughly discussed in Section 4.2.3 and Section 4.2.4 along with its detail implementation. Finally, a control point determination algorithm is introduced in Section 4.2.5.

### *4.2.2. Tool Lifting Method for Springback Measurement*

In the IF process or any other metal forming process, it is typical that any deformation of the sheet metal at room temperature undergoes both elastic and plastic deformation. After the metal workpiece is removed from the tool or deformation implement, the elastic deformation will be released, and only the plastic deformation will remain, as shown in Figure 4.14. The release of the elastic deformation in the tool unload phase causes in-process springback $\delta^2$ while the release in the clamp removal phases causes post-process springback $\delta^3$, as reviewed in Section 1.2.2.

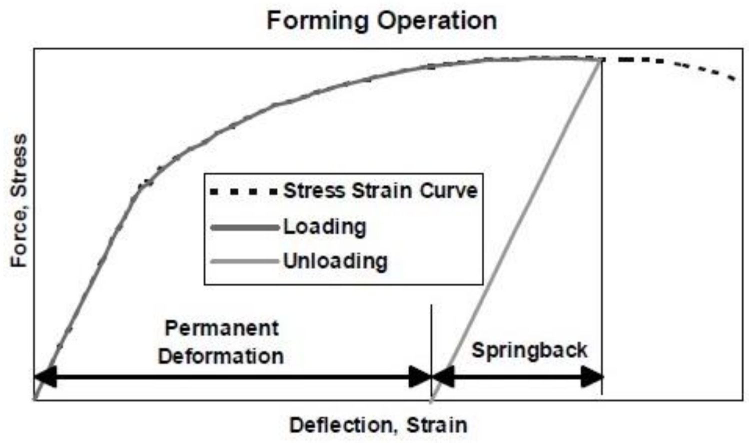


**Figure 4.14 The principle of springback in sheet metal forming (Hu et al., 2002)**

To measure the in-process springback error in the IF process, a tool lifting operation is proposed that does not require any specialized tool or scanner. Instead of only lifting up the forming tool after the whole part is formed, the tool motion is paused during the forming process, and the forming tool is lifted up at the measurement location of interest. While the forming tool is moving up gradually, the forming force and the corresponding tool position are simultaneously recorded. Once the recorded force is equal to zero, the forming tool loses contact with the metal sheet and the part reaches its final geometry. The difference between the initial tool position and the final loss-of-contact position will be taken as the in-process springback error at this specific location, as shown in Figure 4.15. Finally, the tool is moved back to its original position and resumes the paused forming operation. Note that the only tool utilized in this measurement method is the forming tool and thus it can be utilized for not only DSIF, but also SPIF and TPIF processes.

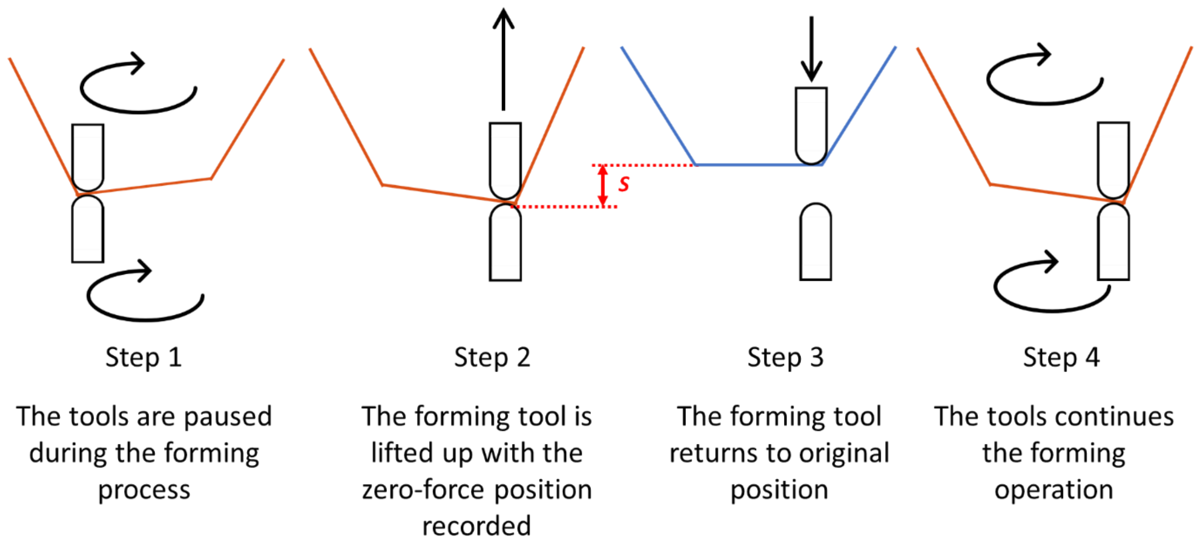

S
Step 1
The tools are paused during the forming process
Step 2
The forming tool is lifted up with the zero-force position recorded
Step 3
The forming tool returns to original position
Step 4
The tools continues the forming operation

**Figure 4.15 A schematic showing the tool lifting operation procedure**

In Figure 4.15, the distance between the zero-force and initial position, as stated earlier, is the springback value *S*. Note that this operation mimics the reverse operation of the touching probe, but without the need of a specific tool. Instead, the force sensor mounted behind the tool is utilized in combination with the position encoder of the tool motion axis to measure the tool travel distance before the tool is losing contact with the part. Two samples of the load-displacement curve during the tool lifting (Step 2 in Figure 4.15) and tool returning (Step 3 in Figure 4.15) at two different forming depths are shown in Figure 4.16. The $F_0$ stands for the forming force between the forming tool and sheet when the motion of the tool is paused. Note that tool lifting takes place in the Z-axis direction as shown in Figure 4.15 and Figure 4.16 since the springback error in X- or Y-axis is assumed to be negligible in our analysis, which will be explained in the Section 4.3.3. However, if the springback value in X- or Y-axis is perceptible and needs to be accounted for, the direction of tool lifting movement must be changed to align with the direction of the total force, which is composed of all the forming forces in X, Y, Z directions.

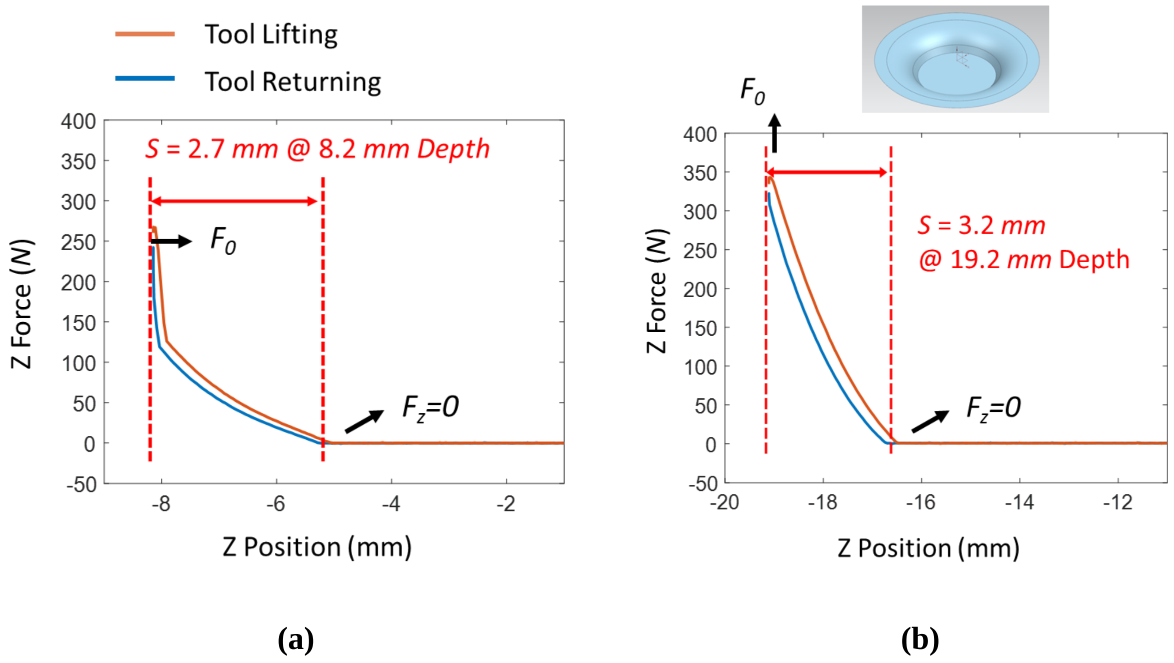
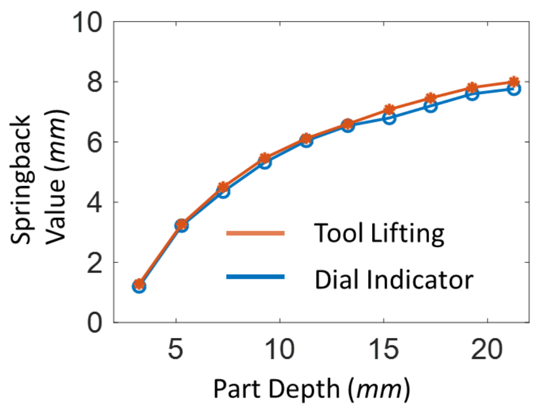
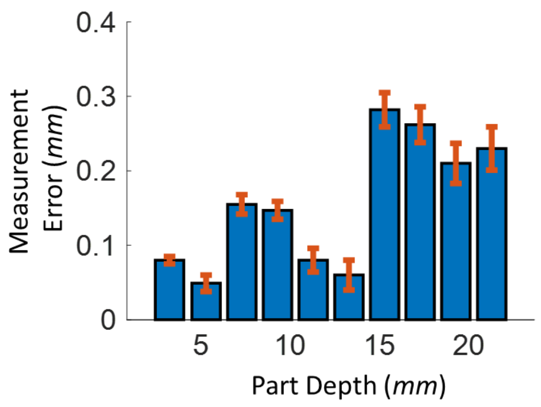


**Figure 4.16 Two examples of force-load curves of the tool lifting method for a cone at different depths**

To further verify the accuracy of the tool lifting method, a 45° truncated cone is formed with 1 *mm* thick AA5754-O sheet. The forming process has been paused each time when the forming tool traveled around 2 *mm* deeper in the Z direction. The springback value measured by the tool lifting operation is compared with the one measured by a dial indicator, which was mounted to touch the bottom surface of the cone shape. Each measurement has been repeated four times. The springback measurement results are compared in Figure 4.17(a) while the difference between the two measurement methods is shown in a bar plot in Figure 4.17(b).





**(a)** **(b)**

**Figure 4.17 (a) The comparison between the springback values measured by tool lifting and dial indicator, (b) difference between two measurements and the corresponding standard deviation**

It can be seen that as the part is formed deeper, the springback value measured increases while the difference between these two measurement methods remains at a reasonable value (< 300 $\mu m$). Considering that the overall part depth is larger than 20 $mm$, the accuracy offered by the tool lifting operation is quite sufficient (< 2% for the ratio of error/part height) as a springback measurement method. The small standard deviation (< 20 $\mu m$) also demonstrates the good repeatability of the measurement method.

Note that in this subsection, the tool lifting method is explained as an introduction to the simulation-based springback method, as the tool liftering operation itself perceptibly represents the underlying mechanics of the in-process springback phenomenon, i.e., the elastic response of the formed sheet due to the change of the forming forces. The following simulation-based springback method is indeed established based on the mechanics of the process, to measure the springback value of the sheet metal by calculating how much the sheet metal is traveling under the measured forming force.

### *4.2.3. Simulation-based Springback Measurement*

Although measuring the springback value via the tool lifting operation does not need any specific tools or setup, the measurement efficiency of this method is limited since by tool lifting only a single point can be measured each time and is not suitable for a complex geometry if the springback error at multiple points is of interest.

To address the above-stated issue, a FEM-based springback measurement method is proposed in this subsection. The method is quite similar to the tool returning operation described as Step 3 in the previous section in Figure 4.14, except that the whole procedure is conducted in a FEM simulation environment. In this method, the springback value is measured at individual discrete points at each time instant. For a single measurement, the target geometry is first truncated at the same depth as the Z coordinate of the measurement point of interest. Then, the truncated geometry is meshed, and the boundary condition, thickness field, and the material property assigned. Once the above procedure is completed, the forming force ($F_0$) is measured and applied to a particular node of the meshed surface with identical coordinates to the point of interest where the load is measured. The above-mentioned mesh geometry, boundary condition, thickness field and the loading condition constitute a complete elastic simulation which takes seconds to execute. Once the simulation ends, the displacement of the loading point is retrieved as the springback value $S$ for the part. The whole procedure is presented in Figure 4.18. Note that since the meshed surface of the springback prediction FEM simulation is generated based on the individual part geometry and process/machine setup as mentioned above, the influence of different geometric features and material choices is implicitly taken into consideration by the model. This implies that the proposed springback prediction model will be applicable for various complex part geometries and different machine/material setups.

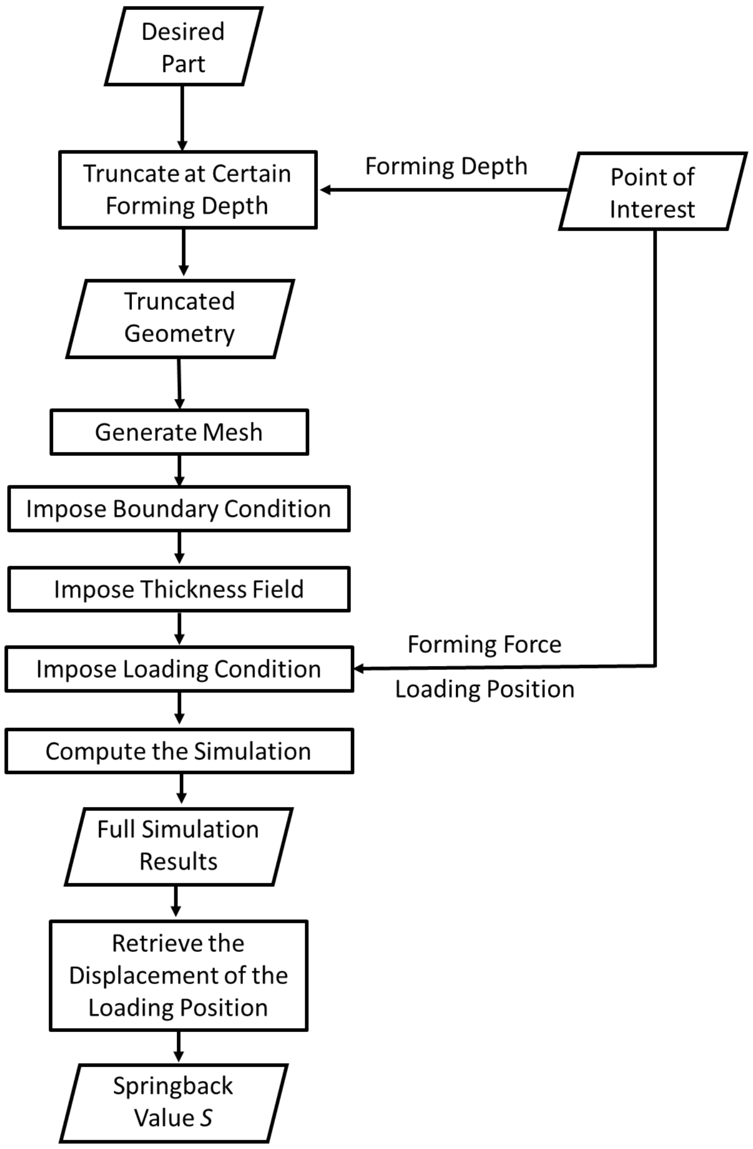


**Figure 4.18 The procedure of the simulation-based springback measurement method**

It is noteworthy in this method that the whole FEM simulation model is meshed based on the desired geometry ($X^d$), which is truncated at a specific depth as shown at the Z coordinate of the measurement point of interest in Figure 4.19. However, in the physical forming process, it is the final geometry ($X^f$) that responds to the removal of the tool loading, but not the desired geometry. If the final geometry is significantly different from the desired geometry, it is highly possible that the structural stiffness of the part will be different. This, in turn, could induce some measurement error in the springback measurement as it will be shown later in Section 4.3.2.

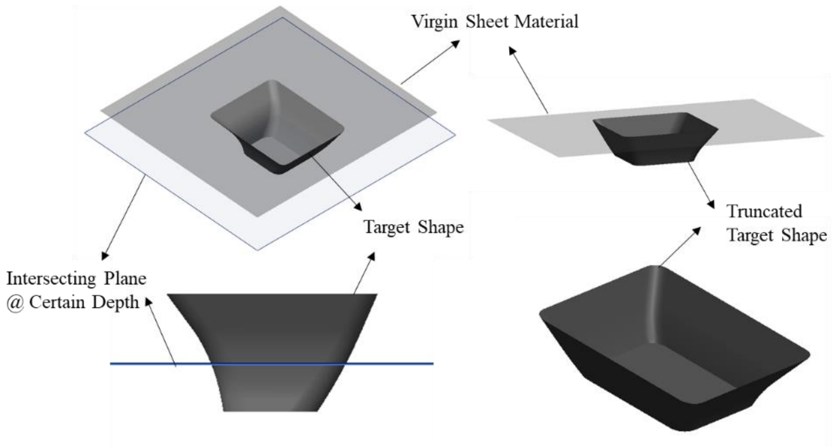


**Figure 4.19 An example of part truncated geometry at a certain depth**

An illustrative example is further provided for the shell-like geometry shown in Figure 4.20(a). First, the part shape is directly extracted from the CAD part at a certain depth, i.e., the Z coordinate of the point of interest (Figure 4.20(b)). The part is meshed (Figure 4.20 (c)), boundary conditions, the sheet thickness (following the sine law assumption (Young and Jeswiet, 2004)) and material properties are applied to the different elements (Figure 4.20(d)). Then, the load is applied (that is measured in-process), the simulation model calculations conducted (Figure 4.20(e)), and the elastic response retrieved by inquiring the displacement vector of the loaded node (Figure 4.20(f)). The displacement of the loading point corresponds to the predicted springback value *S*. If the springback values need to be evaluated at multiple points along the tool path, this method as defined in Figure 4.18 will be repeated. By doing so, the continuous change of the springback value during the part forming process can thus be measured.

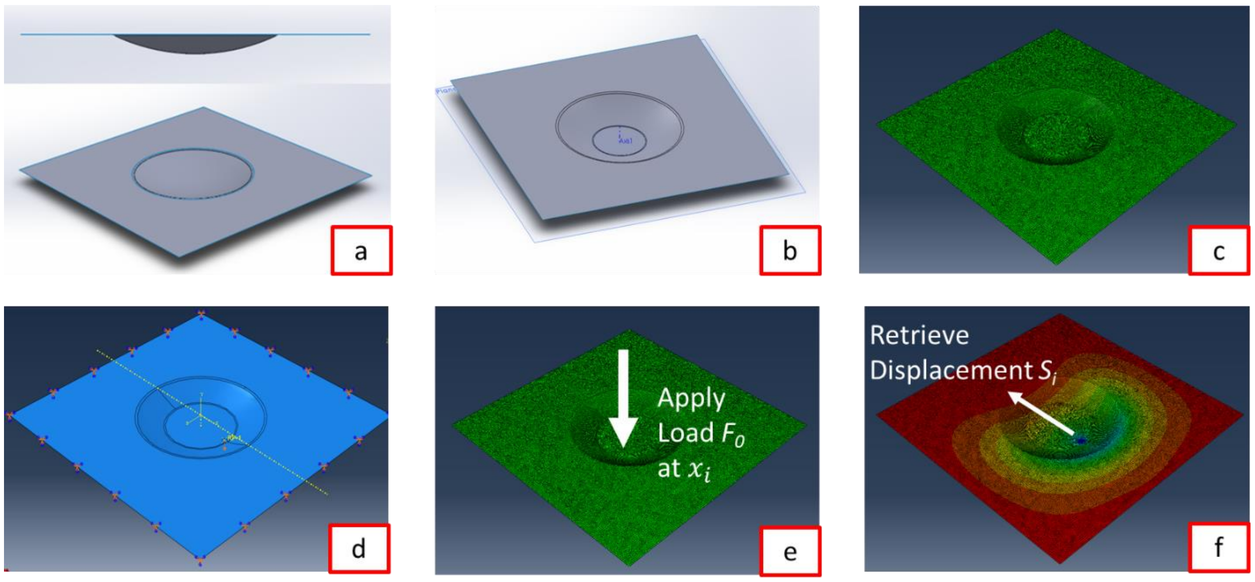


**Figure 4.20 An example of the accelerated springback prediction model (*$x_i$* designates the coordinates of the measured point of interest)**

To compare the calculation time of this simulation-based springback prediction method with the full-scale simulation method reviewed in Section 4.1.1, the execution time for each step (from Step a to Step e in Figure 4.20) for the shell-like geometry example is recorded and shown in Table 4.1. Note that the whole example is executed on a 16-GB-ram personal laptop. The simulation, including the preprocessing, execution, and postprocessing, is completed with one CPU. The specific software utilized in each step, is also given in the table.

As shown in Table 4.1, the whole springback calculation method (including the simulation and also the pre/post-processing of the simulation model) can be effectively finished within 4 *s*, while a full-scale simulation will take more than 80,000 *s* to complete. This dramatic reduction of the computation time is due to the fact that springback is a pure elastic deformation problem. Without sophisticated plasticity and/or contact algorithms involved, the simulation can indeed be finished on the order of seconds instead of hours or days. However, to implement springback compensation during the forming process, the springback value at multiple points along the toolpath needs to be evaluated. If all the steps in Table 4.1 are executed during the forming process, the above-stated execution period will limit the overall bandwidth of the springback compensation algorithm, which is not acceptable in the present application and will be addressed in the following subsection.

**Table 4.1 Benchmark test for the simulation-based springback prediction method**

| **Step** | **Procedure** | **Software** | **Time (*s*)** |
|---|---|---|---|
| b | Truncate Target Geometry at Certain Depth | UG NX | 0.3 |
| c | Generate Mesh | ABAQUS | 0.9 |
| d | Impose Boundary Condition/Thickness Field | MATLAB + ABAQUS | 0.1 |
| e | Gather Force Signal/ Impose Loading Condition/ Compute Simulation Model | MATLAB + ABAQUS | 2.2 |
| f | Retrieve Simulation Result | MATLAB | 0.02 |
| | Total | | 3.5 |

### *4.2.4. Load-Displacement Function g*

One simple but effective solution to further shorten the computational time during the forming process is to fit a load-displacement function off-line from the simulation results, and only utilize this response function to calculate the springback value during the forming process. For example, a sample load-displacement curve measured from a pyramid geometry is shown in Figure 4.21. It can be observed that the simulated load-displacement curve can be well fitted with an exponential function, *g*. Moreover, the other measured load-displacement curves such as the one presented early in Figure 4.16(a) or later in Figure 4.34 (in the experimental verification section) can also be fitted with the same function form with an R-square value larger than 0.99. Note that the form of the fitted function *g* is not limited to an exponential function, but it can be of any form (even a piece-wise linear function) as long as it can capture the simulated load-displacement data.

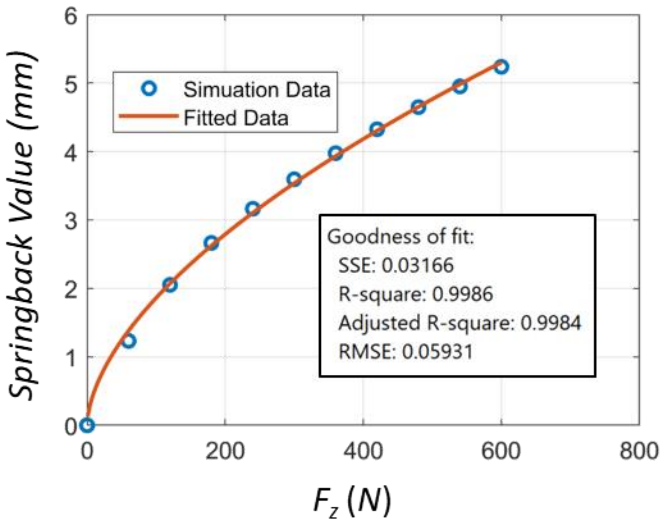


**Figure 4.21 The sample load-displacement curve and its corresponding exponential function fit**

Note that the function *g* in Figure 4.21, to be called the load-displacement function in the following content, in fact presents the detailed relationship between the measured force and the corresponding springback value at the point of interest for the specific geometry, i.e.:

$$S_i = g(F_i) = a_i F_i^{b_i} \tag{4-7}$$

If function *g*, i.e., parameters *a* and *b* in Equation 4-7, is fitted from the simulation results prior to the physical experiments, springback can be immediately calculated once the force signal is measured. This enables a much faster on-line springback measurement than running all the simulations during the physical forming process. To utilize this special characteristic of the function *g*, the springback measurement procedure has been modified as shown in Figure 4.22. Note that in original procedures shown in Figure 4.18, all the operations are executed in the on-line stage during the forming process, when any delay in springback measurement will cause the idle of the machine and thus elongate the manufacturing time. On the other hand, in the modified procedures, the most of procedures will be finished prior to the physical forming process (i.e., the toolpath generation stage as shown in Figure 4.13). During the forming process, the only computational task for the on-line springback measurement method is to measure the forming forces and calculate the springback value based on the load-displacement function *g,* which often takes less than 0.03 *s* to finish. The benefit of such modification is not only the avoidance of any potential manufacturing idle time due to the long simulation period, but also the potential further decrease of total simulation time, since now the simulation procedure (shown in the blue background in Figure 4.22) can be computed on a remote cluster with parallel computing in the off-line stage.

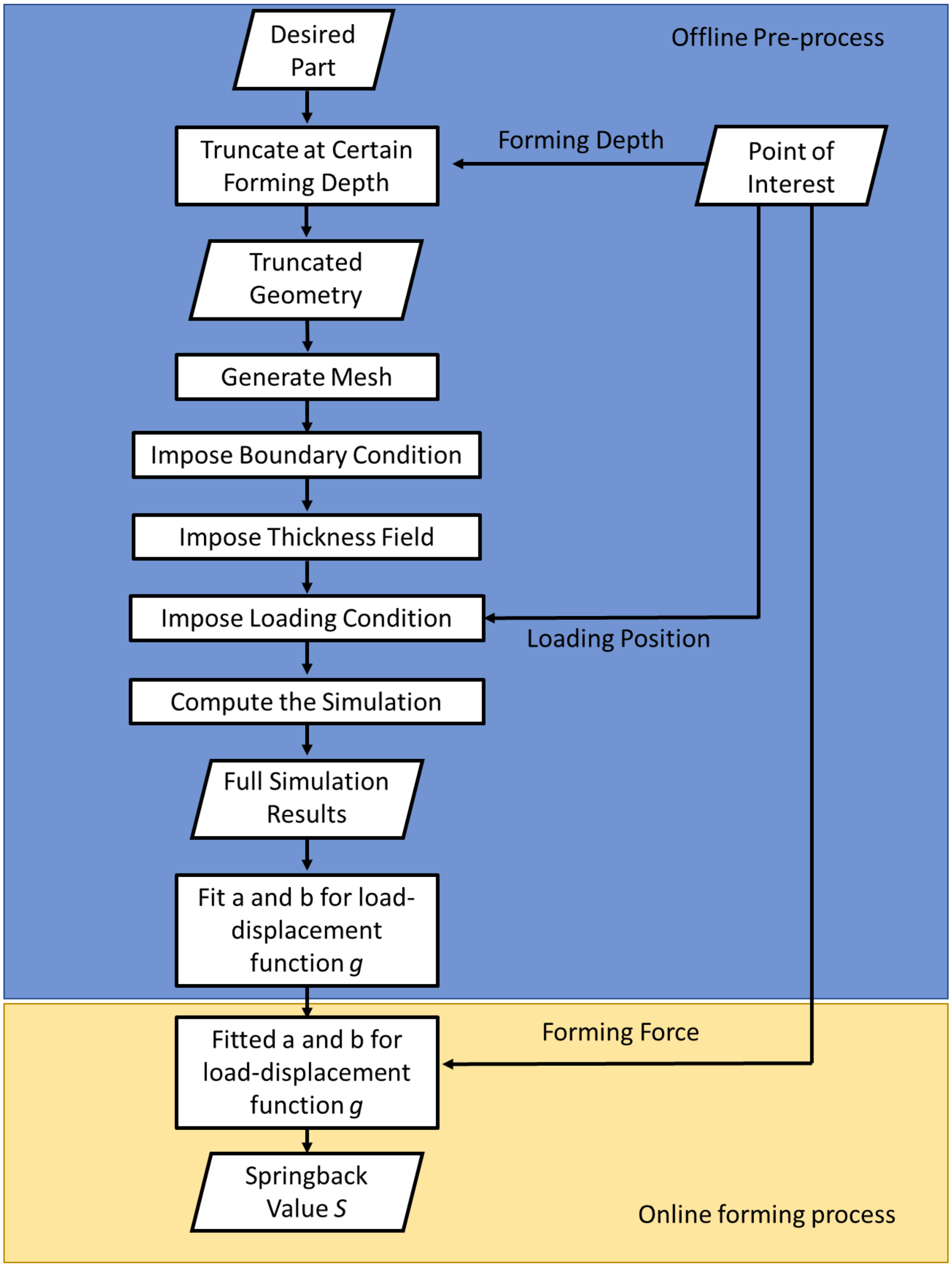
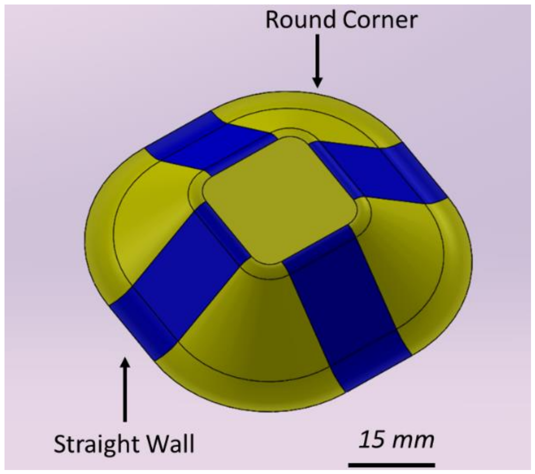


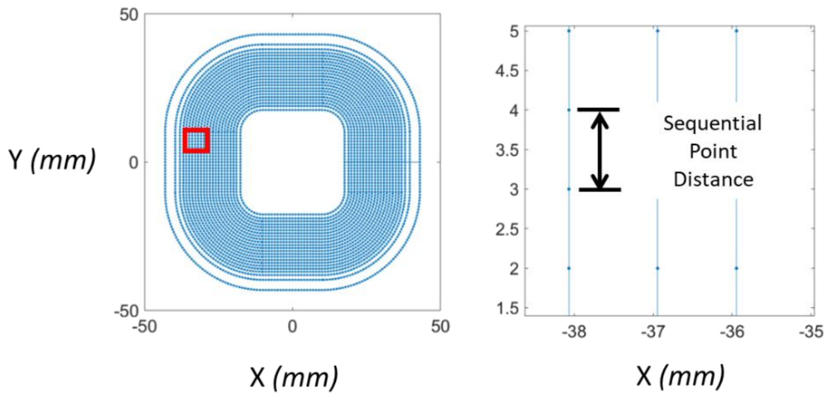
**Figure 4.22 Modified procedure of the simulation-based springback measurement method**

### *4.2.5. Control Points Determination Algorithm*

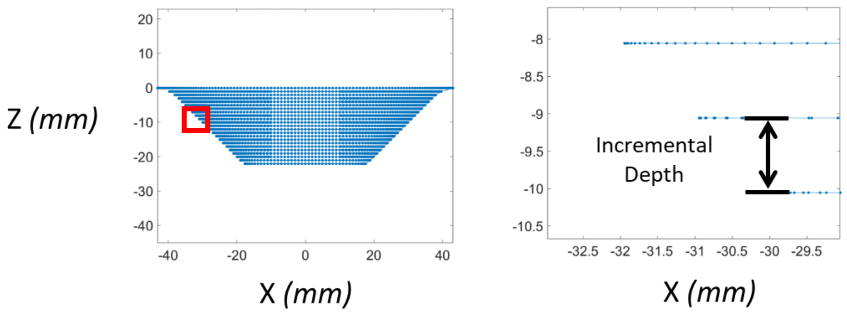
In the typical DSIF process, the tool trajectory is usually defined by a sequence of discrete toolpath points at a predefined distance. The number of the toolpath points depends on the part size, incremental depth and sequential point distance, the definition of which can be found in Section 2.4.1. For example, Figure 4.23(a) and (b) show that more than 5,000 points will be needed to adequately define the toolpath for a 40*40*20 $mm^3$ sized pyramid with a 1.0 *mm* incremental depth and 1.0 *mm* sequential point distance. On the other hand, a springback simulation to calculate the springback error for a single point requires three to four seconds to execute (even though the majority of the procedure will be running in the off-line mode as stated in the previous section). If the springback value needs to be evaluated at every point, the total calculation time could be enormous (more than 4 hours for the pyramid part shown in Figure 4.23), especially for certain automotive industrial parts whose sizes are easily one or two magnitudes larger than the demonstrated pyramid part.


Round Corner
Straight Wall
15 mm

**(a)**


Y (mm)
X (mm)
Sequential Point Distance
X (mm)

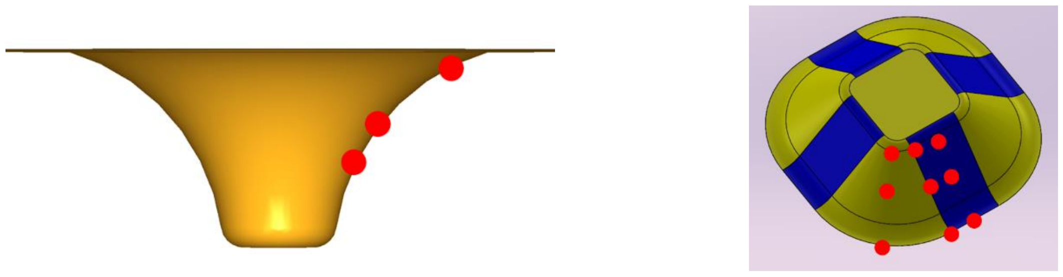
**(b)**


Z (mm)
X (mm)
Incremental Depth
X (mm)

**(c)**

**Figure 4.23 (a) A pyramid shape with round corners and straight walls (b) top view of the toolpath points generated (c) side view of the toolpath points generated**

A natural way to avoid such repeated computations of the springback evaluation is to evaluate the springback values only at some critical points which fully present the whole geometry. Such points are referred as control points in the following context, while the springback values at the other toolpath points will be interpolated based on the calculated springback results at the control points. For a truncated cone or funnel shape, only one control point for each tool revolution will be needed due to the axisymmetric structure of the shape, as shown in Figure 4.24(a). However, for a more complex part containing multiple features like the pyramid shape above, a minimal number of three points would be needed to present the corners and the straight walls as shown in Figure 4.24(b). Note that in both Figure 4.24(a) and (b), multiple control points are plotted at different depths (or tool motion revolutions) since the springback values (or the load-displacement functions) are usually different at different forming depths.



(a) (b)

**Figure 4.24 A possible selection of control points for (a) a funnel shape, (b) a pyramid shape**

As it can be seen from Figure 4.24(b), the control points are located at the center of each feature and the boundary between different features. However, how to identify each feature in a complicated part is a challenging problem. Moreover, under some circumstances in DSIF, there exists only toolpath points but no analytical representation of the target geometry, and from the toolpath's point of view, a geometry feature is essentially a cluster of nearby discrete points that share similar geometric characteristics. That being said, the essence of the discrete control points determination algorithm is a discrete point clustering problem, which can identify different point cluster belonging to different features. Note that there exists a limited prelabeled data set for such a problem (i.e., discrete point sets with labeled geometry features for sheet metal parts), and thus only unsupervised clustering algorithms will be considered in this work.

Following the above-mentioned analysis, a full control point determination algorithm is developed as shown in Figure 4.25. The toolpath is first separated into contour loops, i.e., toolpath revolutions. In each contour loop of the toolpath, the wall angle and/or in-plane curvature, which represent the local geometry, are extracted for every individual discrete toolpath point. These extracted numbers, referred to as toolpath feature indexes, in this context are directly linked to the local geometric feature. Based on the extracted toolpath feature indexes, the discrete point clustering algorithm will divide all the toolpath points into a finite number of subgroups, each of which represents a standalone feature. The number of features can either be determined by external user input or the algorithm itself. After the clustering operation, the central and boundary tool position points will be selected as control points for this contour. This whole operation is repeated for every contour loop until the whole toolpath is successfully processed.

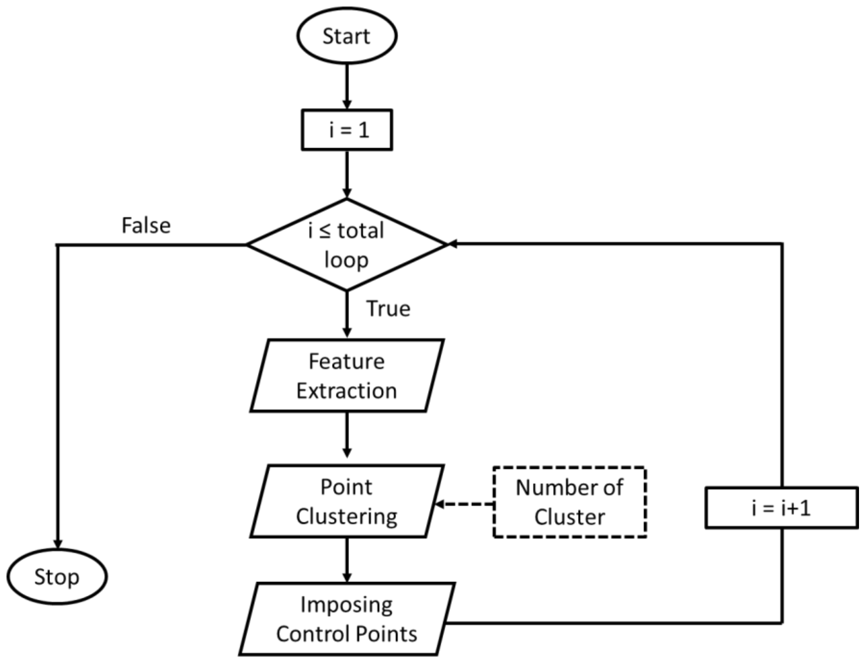


**Figure 4.25 Flowchart of the complete control points determination algorithm**

In the feature extraction algorithm, three feature indexes are selected to represent the local geometry: 1) wall angle $\beta$, 2) the radius of the in-plane curvature $\rho_1$, 3) the radius of the out-of-plane curve $\rho_2$. The wall angle denotes how steep the geometry feature is as introduced. The in-plane curvature radius is the radius of the local fitted circle via the points of interest and its adjacent ones in the X-Y plane (Figure 4.26), while the out-of-plane curve radius is the radius of the local fitted circle which is normal to the tool motion direction (Figure 4.27). The fitting algorithm utilized is the direct least square fitting algorithm that is proven to be robust even when only a small portion of the circle is given (Bjorck, 1996; Pratt, 1987). Note that all these three features can be directly derived from the discrete points and do not require any analytical presentation of the geometry

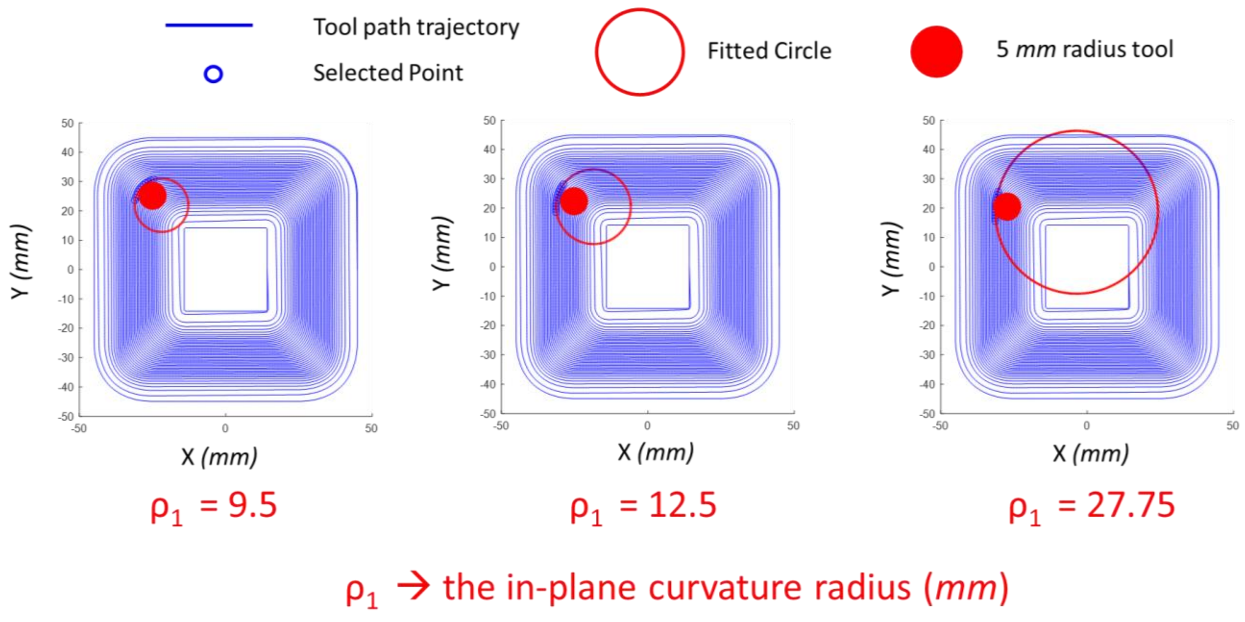


**Figure 4.26 Fitted circles and corresponding in-plane curvature radii of a pyramid toolpath**

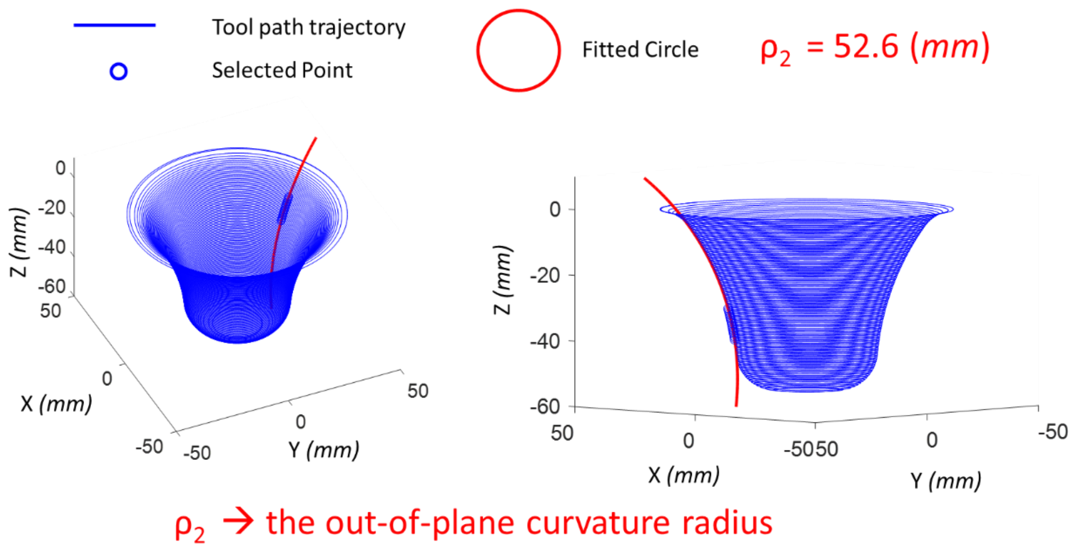


**Figure 4.27 Fitted circles and corresponding out-of-plane curvature radii of a funnel toolpath**

With the extracted feature, the clustering algorithm is then conducted to group the discrete points into separate clusters. The specific clustering algorithm implemented is agglomerative hierarchical clustering, which classifies the whole data set by grouping small clusters into larger ones. The basic algorithm is quite simple, i.e.:

---

**Step 1.** Start with each point in a cluster of its own

**Step 2.** Until there is only one cluster

(a) Find the closest pair of clusters

(b) Merge them

**Step 3.** Return the tree of cluster-mergers

---

The tree of cluster-mergers is sometimes called the hierarchy. The root of the tree is the unique cluster that gathers all the data sets, the leaves being the clusters with only one element, i.e., the tool position in the present case. A sample of the clustered tree is shown in Figure 4.28, in which five data samples (A, B, C, D, E) are classified via agglomerative hierarchical clustering. As one can see, as the merging operation is repeated, the remaining number of groups is decreasing until only one cluster includes all the samples. After the whole clustering procedure, if one wants to classify the whole dataset into three group, the results would be (A, B), (C), (D, E) while the results would be (A, B), (C, D, E) if only two groups are desired. A detailed discussion of this clustering algorithm can be found in literature and is out of the scope of this current thesis (Day and Edelsbrunner, 1984; Steinbach et al., 2000).

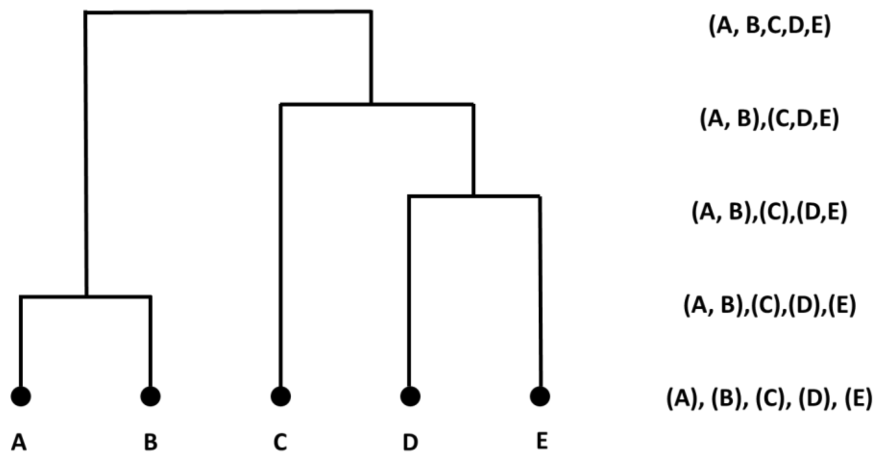


**Figure 4.28 A sample hierarchy tree generated via agglomerative hierarchical clustering**

One critical nature of the agglomerative hierarchical clustering is that it allows the addition of hard constraints into the merging stage, i.e., certain subgroups are allowed to be merged only if they meet some specific constraints (Davidson and Ravi, 2005). This feature is advantageous in this application as one can imagine that only geometrically connected points can be merged together into one feature. That being said, the constraint in the present case of discrete toolpath point clustering is the connectivity between each discrete point.

Another critical aspect of the agglomerative hierarchical clustering algorithm is that it allows an automated determination of the number of the clustered groups by maximizing a particular parameter called the CH index (Anderson, 2001; Caliński and Harabasz, 1974). The CH index is a non-dimensional index that denotes the ratio of between-clustering variation *B* over within-clustering variation *W*. Thus, a specific cluster number with a maximal CH index means that similarity among the elements in the same cluster is guaranteed while the difference between various clusters is maximized.

Given the number of the cluster *K*, and the total number of elements in the whole data set *n*, the detailed definition of the CH index, the within-clustering variation *W* and the between-clustering variation *B* can be expressed as follow (Caliński and Harabasz, 1974) :

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)} \tag{4-8}$$

$$W(K) = \sum_{k=1}^{K} \sum_{C(i)=k} \left\| X_i - \overline{X}_k \right\|^2 \tag{4-9}$$

$$B(K) = \sum_{k=1}^{K} K \left\| \overline{X}_k - \overline{X} \right\|^2 \tag{4-10}$$

where *C* is the specific clustering, and *X* is the sample point defined by the feature indexes such as $\beta$, $\rho_1, \rho_2$ (not the coordinate of the point). $\overline{X_k}$ is the average of the points in cluster *C* and $\bar{X}$ is the overall average for the whole data set, whose detailed mathematical expressions are (Caliński and Harabasz, 1974):

$$\overline{X_k} = \frac{1}{n_k} \sum\nolimits_{C(i)=k} X_i \tag{4-11}$$

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{4-12}$$

where $n_k$ is the number of the sample points in a specific cluster.

With the two above-mentioned methodologies, the clustering algorithm is shown Figure 4.29. Note that the presented agglomerative hierarchy clustering is one of the steps in the whole control point determination algorithm (Figure 4.25), where it takes the extracted feature indexes $\beta$, $\rho_1, \rho_2$

from the previous step and outputs the cluster label for each point, based on which the control points will be assigned, as shown by the dashed line in Figure 4.29.

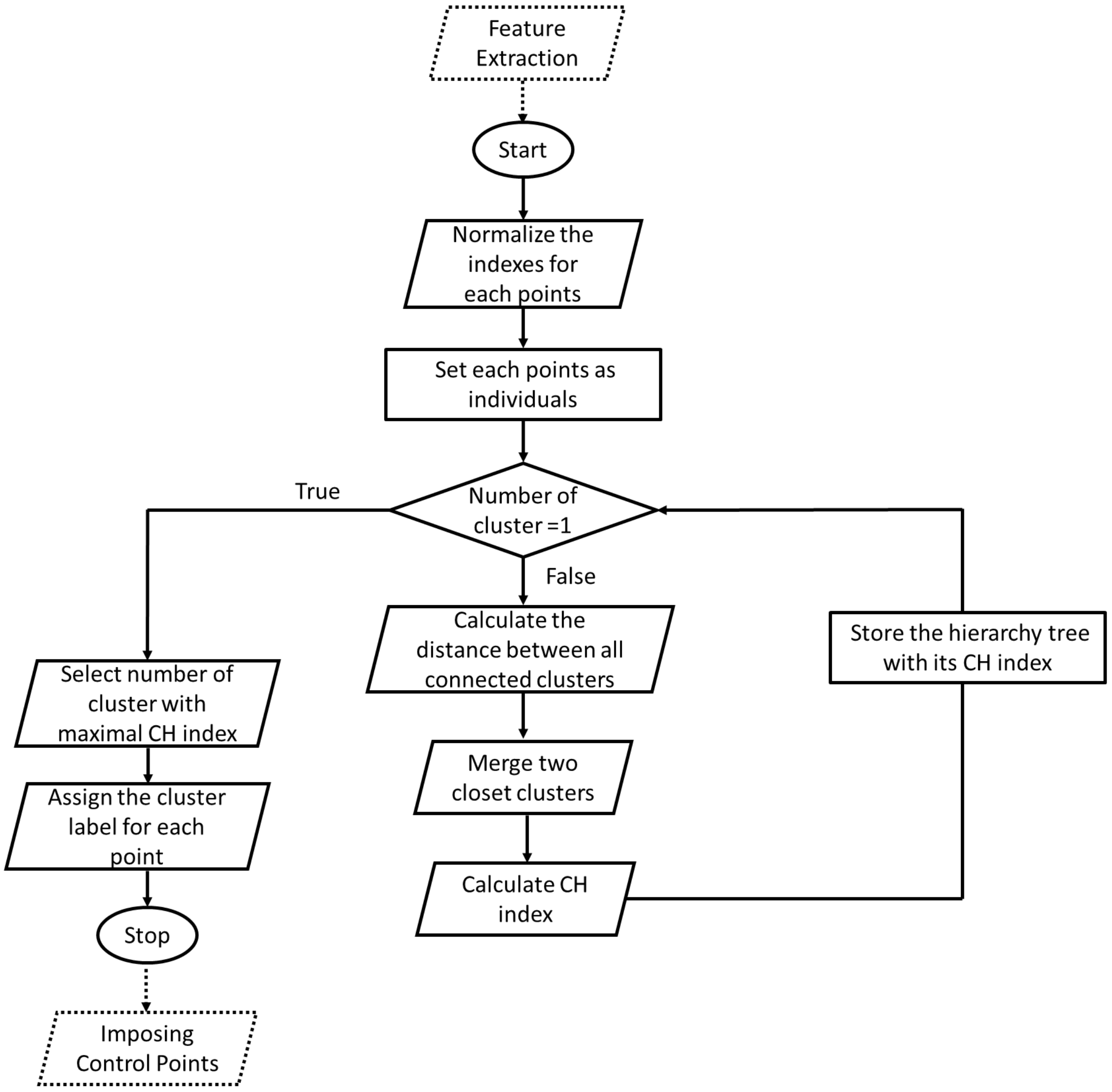

Feature Extraction
Start
Normalize the indexes for each points
Set each points as individuals
Number of cluster =1
True
False
Calculate the distance between all connected clusters
Merge two closet clusters
Calculate CH index
Store the hierarchy tree with its CH index
Select number of cluster with maximal CH index
Assign the cluster label for each point
Stop
Imposing Control Points

**Figure 4.29 The flowchart of the utilized agglomerative hierarchy clustering algorithm**

As seen from Figure 4.29, the extracted features are first normalized. Then, each element, i.e., the tool points, is set as an individual cluster. Then, the algorithm calculates the distance between any connected sub-cluster and merges the two closest ones. Note that by only merging connected regions, the connectivity constraint is implicitly imposed and the elements in each cluster are guaranteed to be adjacent to each other.

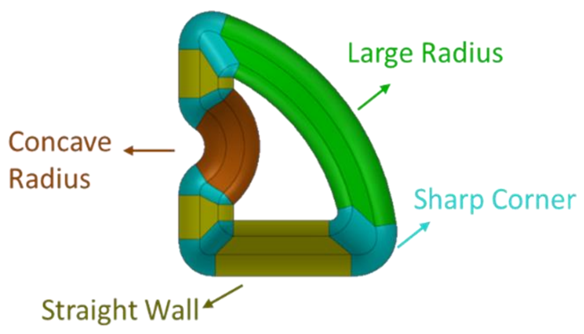
After the merging step, the number of clusters is decreased by one, and their corresponding CH index and hierarchy tree are recorded. When all the sub-clusters have been merged into one single cluster, an optimal cluster number is determined via finding the minimal CH index value. The specific label for each point is then assigned based on the stored hierarchy tree. Note that if a user assigns a fixed number of clusters beforehand, the algorithm will terminate early when the number of clusters is equal to the predefined value, and the label for each point will be directly assigned without the need for minimizing the CH index.

In the typical clustering algorithm, there exist multiple ways to define the distance between two sub-clusters. In our calculation, a ward distance is utilized due to its robustness and generality for different applications (Batagelj, 1988). The detailed representation of the ward distance between two groups A and B can be defined as (Finch, 2005):

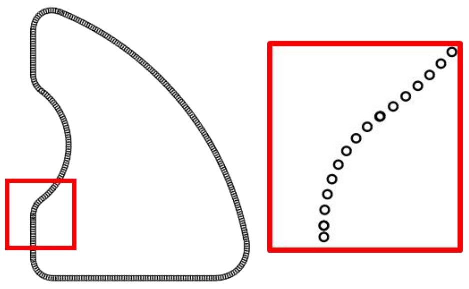
$$d_{ward}(A,B) = \frac{n_A n_B}{n_A + n_B} \left\| \bar{X}_A - \bar{X}_B \right\|^2 \tag{4-13}$$

where $n_A$ and $n_B$ are the number of elements in clusters A and B while $\overline{X_A}$ and $\overline{X_B}$ are the center of elements in clusters A and B, whose definitions are similar with Equation 4-11.

One example of the clustering algorithm is shown below, in which one revolution of the tool position for the fin part is analyzed (Figure 4.30(a)). As it can be seen from the Figure 4.30(b), the identified clusters, denoted in different colors, successfully represent the multiple complex features contained in the part shape, such as the concave/convex curvatures, sharp corners or straight walls. For comparison, the same toolpath contour is also processed by a standard k-mean algorithm (Alsabti et al., 1997; Hartigan and Wong, 1979), which is one of the most widely used algorithms in the unsupervised classification problem. Primarily, the k-mean algorithm aims to partition the elements into k clusters, in which each element belongs to the cluster with the nearest mean called k-mean. The details of the k-mean algorithm's implementation can be found in many literature sources (Alsabti et al., 1997; Hartigan and Wong, 1979) and will not be covered in this thesis. It can be observed that the standard k-mean clustering fails to capture some of the sharp corners while the large radius portion is divided into several small clusters.

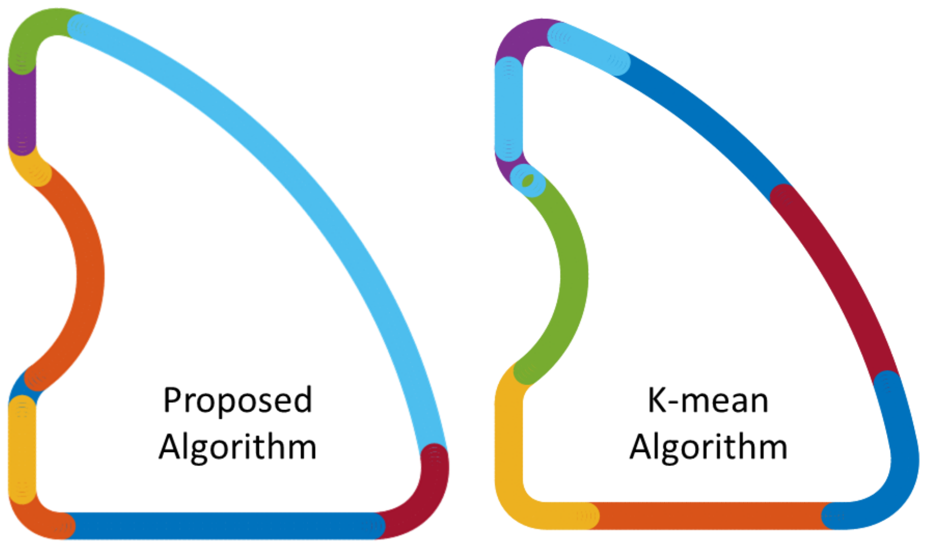


Analytical representation of the fish fin geometry



One revaluation of the fish fin toolpath points

**(a)**



**(b)**

**Figure 4.30 (a) Fish fin goeometry and its discrete toolpath (b) clustered results comparison between the proposed algorithm and standard k-mean algorithm**

Using the developed clustering algorithm, the control points can be successfully determined and assigned at the center or boundary of each cluster. For example, the control points for the above fin part are presented in Figure 4.31. The blue circles denote the central control points of each feature cluster while the red circles represent the control points located at the boundary of the cluster. It is reemphasized here that compared with the cluster results in Figure 4.30, the final points selected are the combination of the control points determined in each contour.

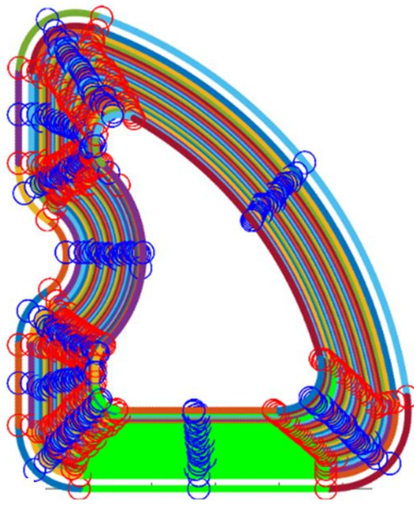
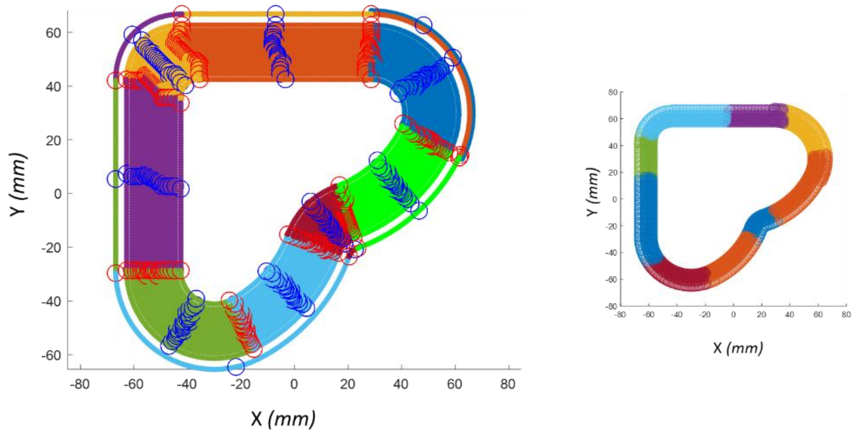
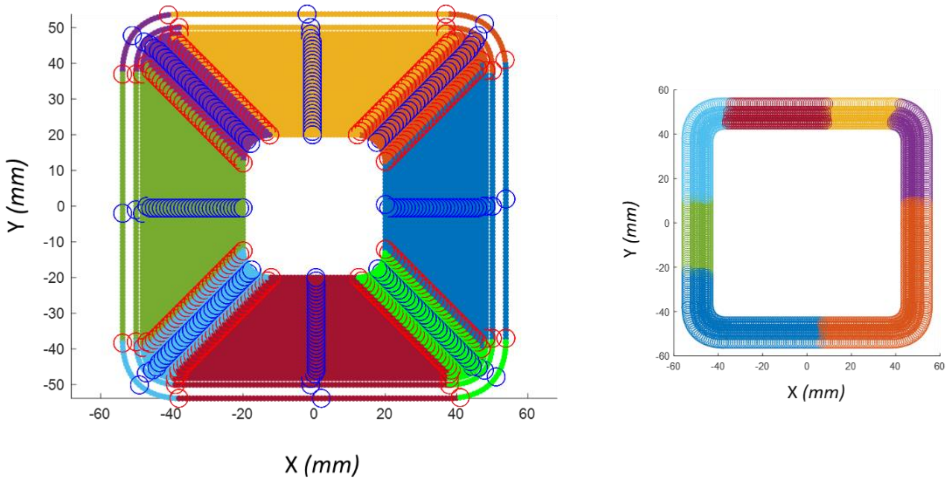


**Figure 4.31 Selected control points in the top view for the fin part**

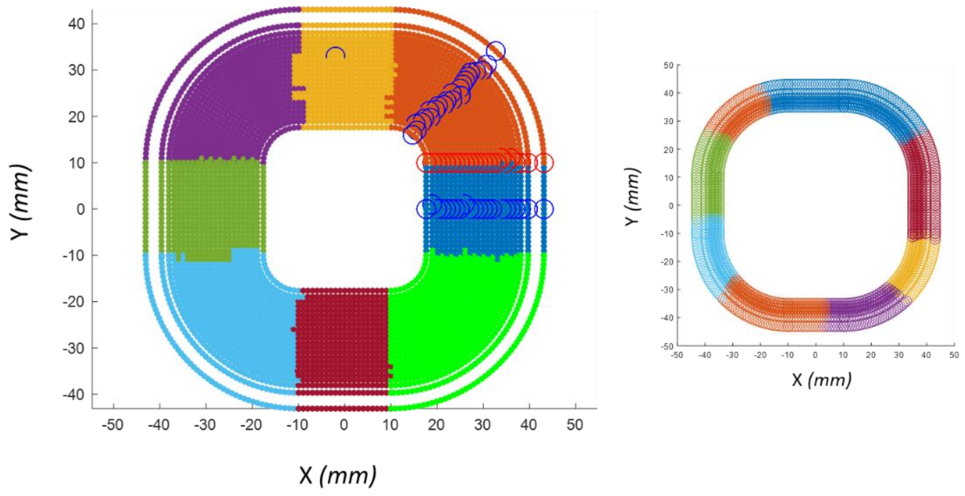
To further verify the generality of the developed selection algorithm, additional geometries have been tested, and the results are shown in Figure 4.32. Note that for each geometry, the clustered results from the standard k-mean algorithm are also shown for comparison. It can be observed that while the proposed agglomerative hierarchy clustering algorithm can successfully identify different geometric features, the k-mean algorithm usually tends to misidentify sharp corners. By recalling that there exist more than 5,000 tool path points for the pyramid shape shown in Figure 4.32(c), the fact that the selected control points by the algorithm can successfully represent the whole toolpath with less than 60 control points to a dramatic reduction in computational cost for the springback identification simulation method described in Section 4.2.3 and 4.2.4.



**(a)**



**(b)**



**(c)**

**Figure 4.32 Determined control points for various geometries (a) heart shape (b) sharp corner pyramid (c) round corner pyramid**

## 4.3. Algorithm Implementation and Experimental Verification

### *4.3.1. On-line Springback Compensation Algorithm Implementation*

With the springback measurement method and the control point determination algorithm explained above, the detailed on-line springback compensation algorithm is established as shown in Figure 4.33. Note that while the springback identification and compensation portion of the algorithm is running during the forming process (the green part in Figure 4.33), the toolpath generation, control point determination, and load-displacement function fitting are conducted off-line before the forming process.

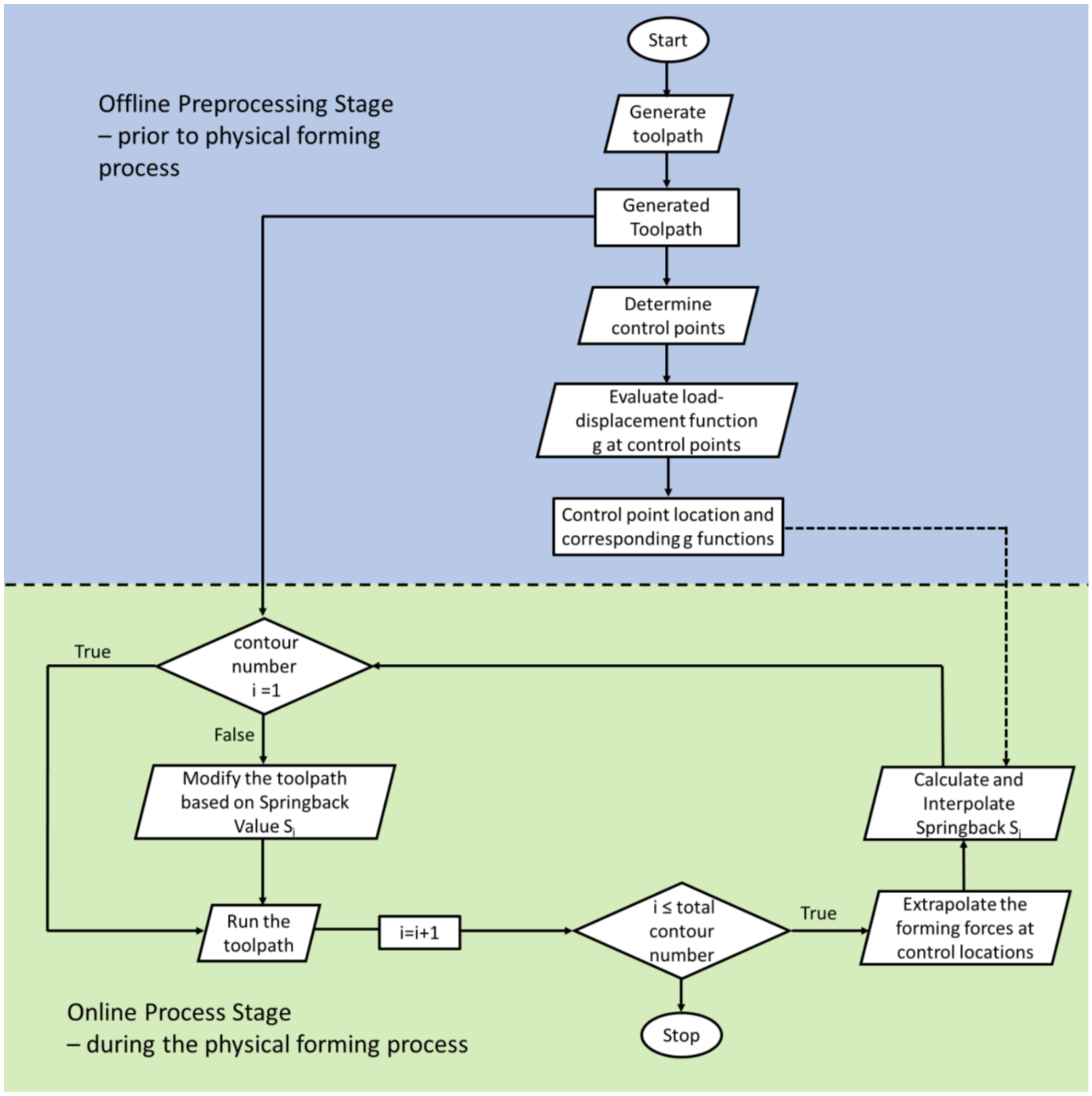

Offline Preprocessing Stage – prior to physical forming process
Start
Generate toolpath
Generated Toolpath
Determine control points
Evaluate load-displacement function g at control points
Control point location and corresponding g functions
True
contour number i =1
False
Modify the toolpath based on Springback Value $S_i$
Calculate and Interpolate Springback $S_i$
Run the toolpath
i=i+1
i ≤ total contour number
True
Extrapolate the forming forces at control locations
Online Process Stage – during the physical forming process
Stop

**Figure 4.33 The flowchart of the on-line springback compensation algorithm**

During the preprocessing stage, the toolpath is first generated by common toolpath generation software such as AMPL Toolpath (Moser, 2018) described in Section 2.4.1. With the discrete toolpath points available, the control points are identified via the control point determination algorithm introduced in Section 4.2.5. Then, the load-displacement function *g*, described in Section 4.2.4, is evaluated using these control points via an elastic FEM simulation generated based on the truncated geometry as shown in Section 4.2.3. The parameters of function *g* and the corresponding control point positions will be stored in a database for springback evaluation later during the on-line compensation stage.

During the forming process, the toolpath is discretized into individual contour loops, and the control actions are taken for each contour loop. For the first loop, the original toolpath is uploaded into the motion controller, and the forming operation is conducted. The forming forces and the tool positions are recorded. Then, the forming forces at the control points which belong to the next contour loop are extrapolated based on the forming force history, and the springback values at the control points are evaluated based on the function *g*, whose parameters (*a* and *b* defined by Equation 4-7) have been identified in the off-line stage. In the next step, the springback values of the remaining toolpath points are interpolated based on the control point results. With the springback error, *S*, identified, the toolpath of the next loop will be shifted. To guarantee the continuity of the compensated toolpath, the change in the compensation value per unit toolpath length is limited to 50 $\mu m/mm$ and a moving average filter is imposed on the modified toolpath to smooth it further out. Finally, the modified toolpath is uploaded to the motion controller and the whole springback compensation algorithm repeats until the last contour loop. It is noteworthy that in the current experimental setup (250 * 250 *mm* forming region), each contour loop requires around 100 *s* to form if the tools are moving at 6 *mm*/*s*. On the other hand, the computational time required to evaluate the springback values and modify the toolpath only takes less than 0.2 *s*, which means that the increase in the total forming time due to the on-line springback compensation algorithm is negligible (< 0.1%).

Note that the springback compensation algorithm shown in Figure 4.33 is executed at the period of each toolpath contour loop, i.e., the modification made on next toolpath contour loop (loop *i*) is based on the information (forces and positions) measured in previous loops (loop i-1, i-2, …). That being said, to calculate the springback value *S* in the next loop, according to Equation 4-7, a force prediction model is needed to determine what will be the force in the next loop ($F_i$) based on the current force history ($F_{i-1}$, $F_{i-2}$, …). In the current algorithm, linear force extrapolation is utilized as the force prediction model defined as:

$$\begin{aligned} & F_2 = F_1 \\ & F_i = 1.5F_{i-1} - 0.5F_{i-2} \quad if\ i > 2 \end{aligned} \tag{4-14}$$

where $F_i$, $F_{i-1}$ and $F_{i-2}$ denote the force values extracted for a given control points at during different loops. It is noteworthy that although a more sophisticated nonlinear force prediction model could be advantageous to achieve a more accurate force prediction, it is shown later in Section 4.33 that the current springback compensation method can achieve a reasonable accuracy with the simple force extrapolation method implemented.

As we discussed before, the off-line preprocessing stages of the algorithm is implemented as automated scripts to minimize the user time and increase efficiency. Similarly, the on-line portion is also implemented with automated MATLAB script, and the target part can be formed with minimal user interaction. The interface between the off-line and on-line stage is enabled by a database of fixed format and a JSON configuration file which defines additional parameters, such as the file storage path. To summarize the whole procedure, a table of utilized software, corresponding scripting languages and the corresponding output data formats are summarized in Table 4.2. The detailed code of the algorithm implementation can be found in Appendix D.

**Table 4.2 Summary of software/scripting language and data formats utilized in the algorithm**

| Step Name | Software | Automation Script | Output Data Format |
|---|---|---|---|
| Toolpath Generation | AMPL Toolpath | Python | CSV |
| Control Point Determination | MATLAB | MATLAB | JSON + CSV |
| Load-displacement Function Identification | ABAQUS/NX/ MATLAB | Python/MATLAB | JSON + MAT |
| Force Extrapolation/Springback Identification | MATLAB | MATLAB | MAT |
| Toolpath Modification/Upload | PCOMMServer/ MATLAB | MATLAB | TXT |

### *4.3.2. Experimental Verification for the Springback Measurement Method*

To verify the proposed springback measurement method, the springback values predicted by the simulation model are compared with the measured ones for several different geometries. First, a 250*250 $mm^2$ 1.0 *mm* thick aluminum sheet is clamped and its elastic response at the center point is recorded, i.e., the sheet is pushed down by the tool in the center of the sheet. The load is measured by the load cell on the tool while the displacement at the center of the sheet is identified through a dial indicator. The load-displacement curve is compared with the one obtained in the simulation model as shown in Figure 4.34. In Figure 4.34, two load-displacement curves are retrieved from the simulation with different boundary conditions: the orange one corresponds to a “plate clamped with an immovable edge” and underestimates the springback value for a certain load, while the blue one corresponds to a “plate clamped with the edge free to move” and agrees well with the experimental results. Note that when the plate is fully clamped there are no rotational DOFs allowed on the boundary, while when the edge is immovable/free to move translational DOFs on the boundary of the sheet are constrained/permitted accordingly.

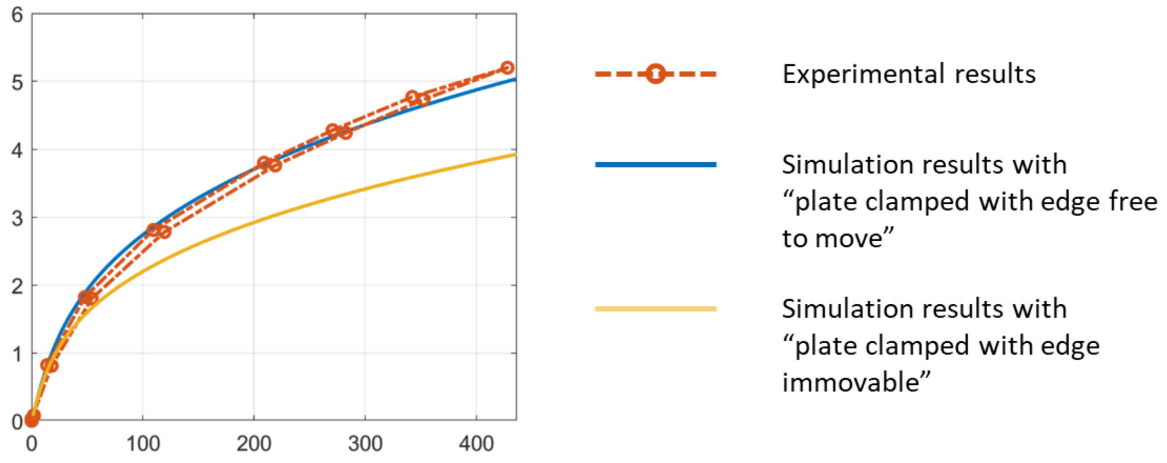


**Figure 4.34 Load-displacement for a square aluminum plate under a central load**

The above phenomenon, i.e., that different boundary conditions cause different elastic responses of the metal sheet, was also studied by Timoshenko et al. (Timoshenko and Woinowsky-Krieger, 1959). In the cited work, Timoshenko derives a generalized relationship between the center load and center displacement for a pure elastic circular plate with different boundary conditions, i.e.:

$$\frac{w_0}{h} + A\left(\frac{w_0}{h}\right)^3 = B\frac{Pa^2}{Eh^4} \quad (4\text{-}15)$$

where $h$ is the thickness of the sheet, $E$ is the Youngs modulus of the material, $a$ is the radius of the sheet while $w_o$ is the central displacement for a central load $P$. On the other hand, $A$ and $B$ are two non-dimensional parameters that vary according to different boundary conditions as shown in Table 4-3. The value of $w_o$ under a 400 *N* center load ($P$ = 400 *N*) is also shown in the Table for a 1 *mm* thick aluminum sheet. The radius is chosen to be 141 *mm* so that the circle has the same area as the 250*250 $mm^2$ square sheet used in the experiments. Note that in the experimental results, the sheet will deflect approximate 5.0 *mm* under a 400 *N* center load.

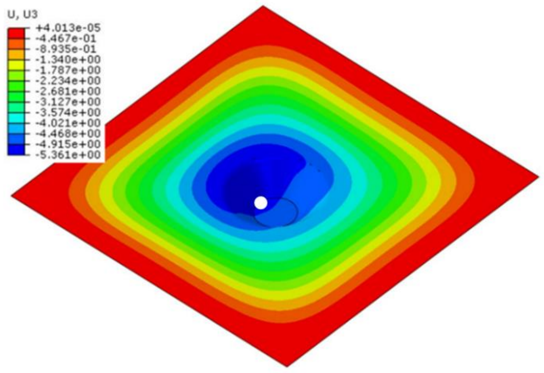
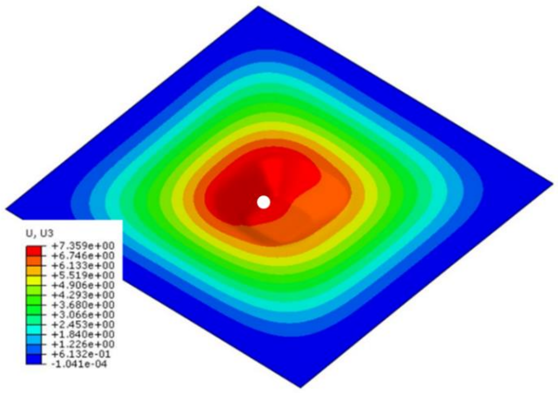
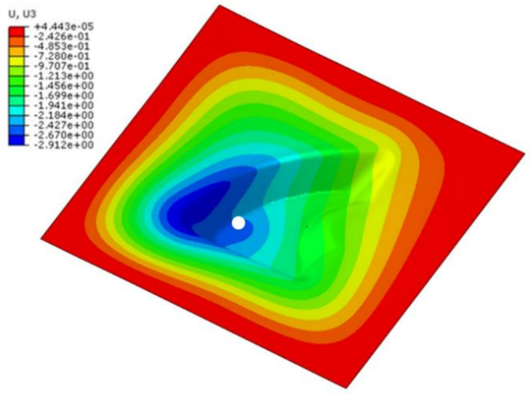
**Table 4.3 Different A/B values for different boundary conditions**

| **Boundary Condition** | **A** | **B** | **$w_o$ ($P$ = 400 $N$)** |
|---|---|---|---|
| Plate clamped with edge immovable | 0.443 | 0.217 | 3.61 *mm* |
| Plate clamped with edge free to move | 0.200 | 0.217 | 4.64 *mm* |
| Plate simply clamped with edge immovable | 1.430 | 0.552 | 3.33 *mm* |
| Plate simply clamped with edge free to move | 0.272 | 0.552 | 5.92 *mm* |

It can be seen from both Figure 4.34 and Table 4.3 that the boundary condition of “plate clamped with edges free to move” agrees best with the experimental results compared to other boundary conditions listed in Table 4.3, and thus the same boundary condition will be utilized in the rest of the work. For other machine setups and clamping systems, a similar test should be done so that the correct boundary conditions can be calibrated and chosen accordingly.

Next, three different geometries are chosen to verify the accuracy of the simulation-based springback prediction model. All these geometries are formed on 1 *mm* thick 250*250 $mm^2$ square aluminum alloy sheet with 10.0 *mm* diameter tools. For each geometry, the forming force is extracted during the experiment at a certain location. Then, the springback value at this point is measured with the simulation model according to the procedure presented in Section 4.2.4. Finally, the simulated results are compared to the scanned results obtained by a ROMER Absolute Arm with an integrated laser scanner. A detailed comparison is shown in Table 4.4 with the displacement contours of the simulation results shown for each case. The springback measurement location is highlighted with a white dot in the contour plots. The table shows that the simulation will generally underestimate the springback error by around 200 to 400 $\mu m$. As discussed above, this error could be explained by the fact that the simulation utilizes the target geometry to generate the simulation model instead of the real formed part, since the latter one is unknown during the forming process. The difference between the formed part and the target geometry will result in different structural stiffness values and thus different springback value calculations as discussed in Section 4.2.3. Nevertheless, the overall error range of this simulation model is of the same magnitude as for the tool lifting operation, and thus this simulation-based springback measurement method is generally efficient and can be implemented in the springback compensation loop (< 3% for the ratio of error/part height). Last but not least, the springback value is measured/calculated in this table only in the Z-axis for easy demonstration. However, the whole framework can easily be expanded to the X and Y axes as well. The only modification needed is to change the load force from $F_z$ to $F_x$ and $F_y$ in the load imposing step in the simulation model (Step e in Figure 4.20).

**Table 4.4 Simulation model verification with different geometry**

| **Shape** | **Springback Value (*mm*)** | | **Simulated Displacement Contour (*mm*)** |
|---|---|---|---|
| Truncated Cone | Scanned | 5.76 |  |
| | Simulated | 5.37 | |
| Truncated Pyramid | Scanned | 5.27 |  |
| | Simulated | 5.09 | |
| Fish Fin | Scanned | 3.59 |  |
| | Simulated | 3.21 | |

A thorough sensitivity analysis is also conducted for the pyramid geometry to test the influence of the different mesh type, size and order on the simulation results. The variation is shown in Table 4.5. Considering both simulation speed and meshing efficiency, the reduced integration first-order triangular shell element is chosen as the element type for the simulations (highlighted in red in Table 4.5).

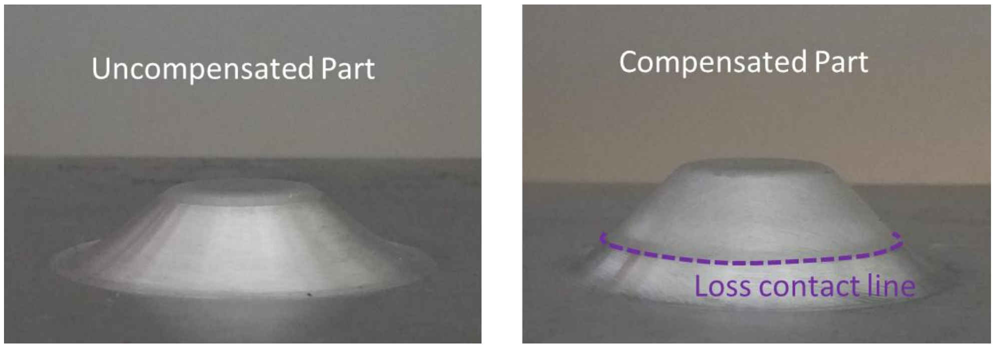
**Table 4.5 Sensitivity analysis for the springback prediction model**

| Sensitivity Check | Springback Variation |
|---|---|
| Mesh Size (0.5 - 1.0 - 2.0 *mm*) | < 0.1 *mm* |
| Element Type (Shell/Solid element) | < 0.05 *mm* |
| Element Shape (Triangular/Quadrilateral) | < 0.05 *mm* |
| Shape Function Order (Reduced Integration First-Order/Full Integration Second-Order/ Reduced Integration Second-Order) | < 0.02 *mm* |

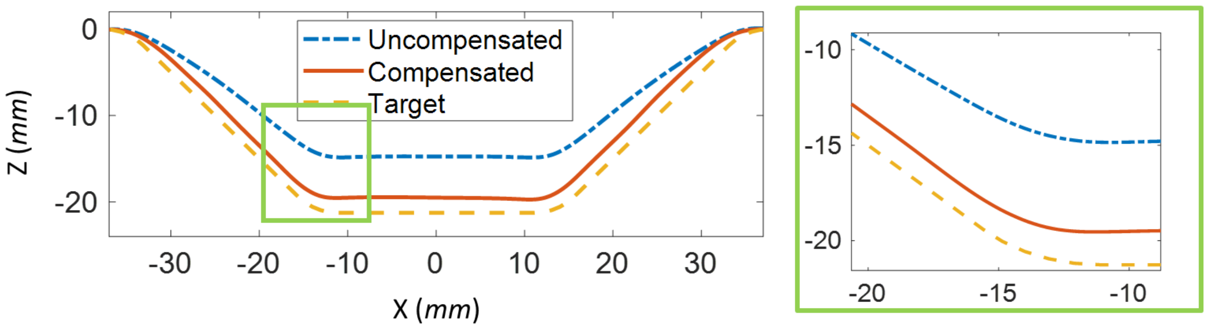
### *4.3.3. Experimental Verification of the Springback Compensation Algorithm*

To validate the proposed springback compensation algorithm, two experiments are conducted using a 1.0 *mm* thick AA5754-O sheet. The tool radius is 5.0 *mm* while the overall formable region is 250 * 250 $mm^2$. In both experiments, the incremental depth is 0.2 *mm* while the sequential point distance is 0.5 *mm*.

A 21 *mm* deep axisymmetric cone is firstly formed as an initial evaluation for the algorithm. In this experiment, 20 control points at successive 1.0 *mm* depth values are identified, and their corresponding load-displacement functions in the Z direction fitted. Two parts are formed with and without the proposed springback compensation algorithms. The formed geometries are then scanned and compared with the target geometry. A comparison of the cross-section views of the top surface for the part is presented in Figure 4.35(a) for the physical parts and Figure (b) for the scanned results. As one can see, with the springback compensation algorithm, the maximal geometric error is reduced from 5.7 *mm* to less than 1.5 *mm* while the average geometric error decreased from 3.1 *mm* to 0.8 *mm*. However, the part is still undercompensated for about 1.5 *mm*. This discrepancy could be partly due to the fact that the simulation method tends to underestimate the springback error as shown in Table 4.4, partly due to the error caused by the linear force approximation stated in Equation 4-14. It is also observed that the springback compensation algorithm induces an early loss of contact, which can be relieved if the force control algorithm is implemented for the supporting tool as shown in Chapter 2.



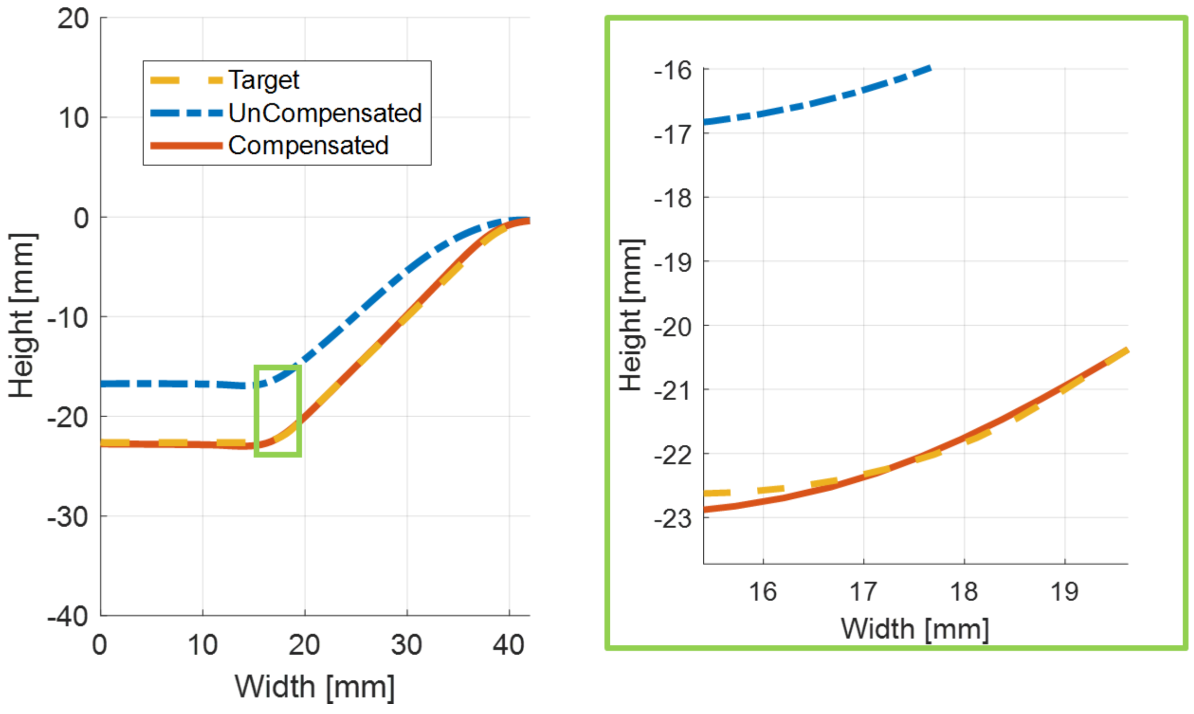
**(a)**



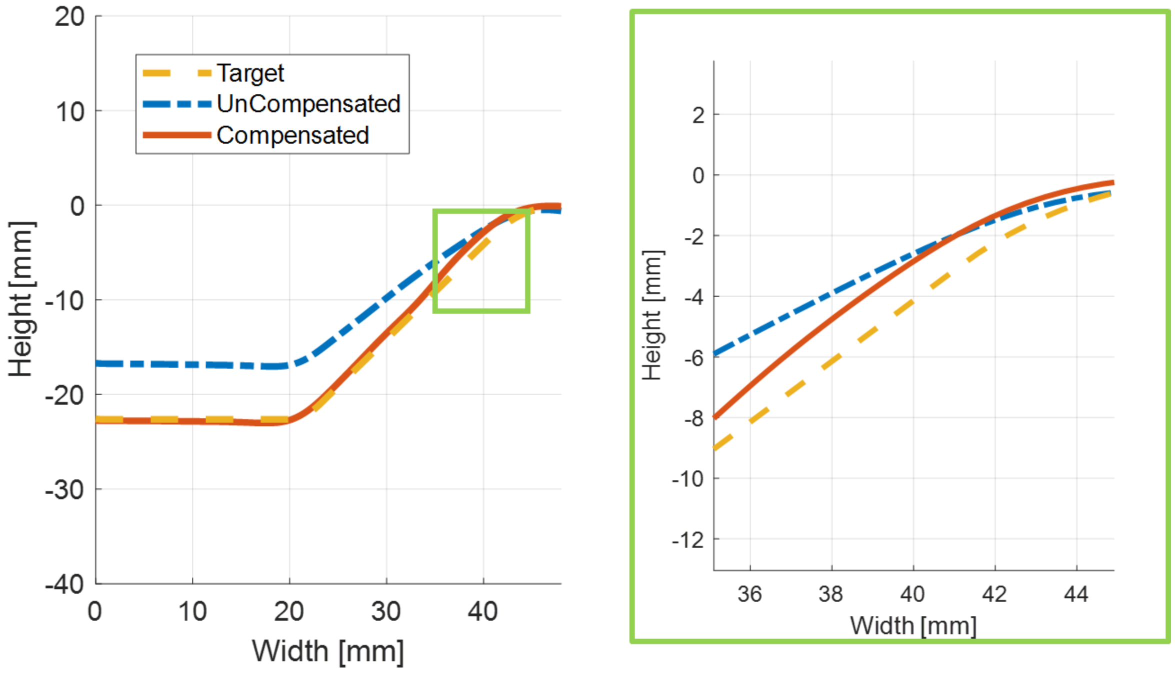
**(b)**

**Figure 4.35 Cross-section comparison for (a) the physical part and (b) scanned results of an axisymmetric cone part**

The second formed geometry is a round corner pyramid shown in Figure 4.24 to test the algorithm's capability for non-axisymmetric shapes. In this experiment, the load-displacement functions at 60 control points are defined and utilized for the evaluation of the springback error in the Z direction. Two parts are again formed with and without the proposed springback compensation algorithm. The cross-sections of the parts are then compared along the X-axis as well as along the diagonal axis (45$^{o}$ from the X-axis), as shown in Figure 4.36. It is observed that the geometric accuracy in both cross-sections has improved, i.e., the maximal geometric error is reduced from 4.8 *mm* to less than 1.1 *mm* while the average geometric error decreased from 2.6 *mm* to 0.2 *mm*. It is also observed that the algorithm compensates accurately at the side wall (Figure 4.36(a)) while undercompensates in the corner (Figure 4.36(b)).



**(a)**



**(b)**

**Figure 4.36 Cross-section comparison for the pyramid part along (a) X axis, (b) diagonal axis (45$^o$ from X axis)**

To further improve geometric accuracy, the compliance compensation algorithm, described in Chapter 2, is implemented in the top tool to form the cone shape, combined with the on-line springback compensation algorithm. Although the maximal geometric error remains almost the same as in no-compliance compensated case, a further decrease of the average geometric error to around 0.6 *mm* is observed, mostly due to the improvements in the sidewall region, as shown in Figure 4.37.

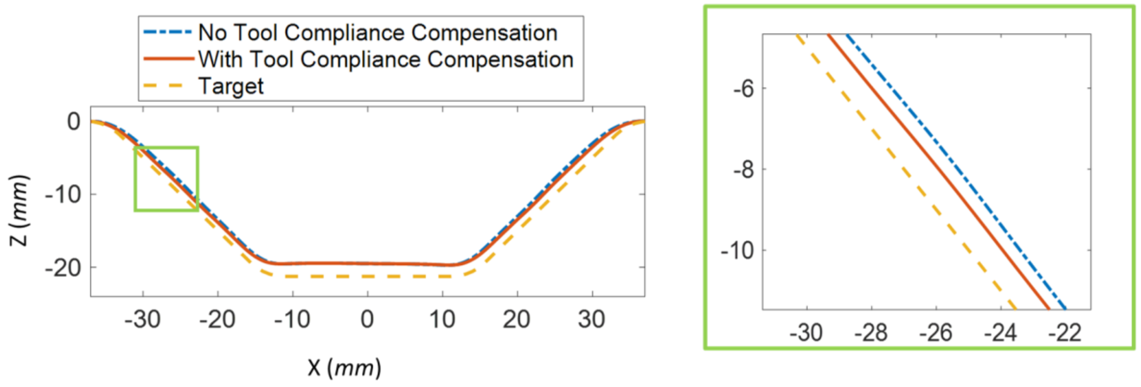


**Figure 4.37 Cross-section comparison for an axisymmetric cone part with/without tool compliance compensation (the springback error was compensated in both cases)**

It is noteworthy that in the springback compensation algorithm only the springback error in the Z-axis is considered while the ones in the X and Y axes are assumed to be negligible. The choice of this can be rationalized by the fact that the horizontal forming forces in the X and Y axes are usually only one third of the Z forming forces, while the structural stiffness in the X and Y axes is much higher than in the Z axis, as verified by the FEM simulation results shown in Table 4.6. Nevertheless, the algorithm can easily be expanded to the X and Y axes if springback in the horizontal directions cannot be neglected.

**Table 4.6 Different springback values for the cone part under load in different axes**

| **Loading Direction** | **Springback Value when F = 200 *N*** | **Springback Value when F = 600 *N*** |
|---|---|---|
| Z | 2.71 *mm* | 5.09 *mm* |
| X/Y | 0.62 *mm* | 1.11 *mm* |

## 4.4. Summary and Discussion

In this chapter, an on-line springback compensation algorithm is proposed. The key innovation of the proposed springback compensation system is the combination of a simulation-based springback prediction model with a springback compensation loop, which innovatively links the measurable variables (force and position) to the desired part attributes (predicted geometric shape change). Thus, springback error can be evaluated and compensated for during the forming process, and no physical or numerical trial and errors are required. In this way, both production lead time and cost of the DSIF process can be further reduced while, at the same time, higher geometric accuracy can be attained as shown in Section 4.3.3. In the future, more complex geometries need to be tested and the non-localization geometric errors, such as the pillow effect, need to be further explored, as discussed later in Section 5.3.

# Chapter 5 Summary and Future Works

In this work, a multilayer on-line/off-line control system for the DSIF process is proposed, which aims to enhance its geometric accuracy and suppress possible fracture. At first, Double Sided Incremental Forming is introduced as a novel flexible forming process (Section 1.1). However, in the DSIF process, maintaining tight geometric tolerances and structural integrity (avoiding fracture) of the incrementally formed parts is a major challenge due to the complex nonlinear nature of the deformation mechanisms of the process (Section 1.2). Thus, three feedback control loops, operating at different levels are proposed (Section 1.3): (1) the machine compliance compensation and contact force control loop (Loop I in Figure 1.22) to compensate tool position errors and maintain stable tool contact with the part (Chapter 2), (2) the off-line toolpath parameter optimization algorithm for ADSIF (Loop II as shown in Figure 1.22) to maintain geometric accuracy (Chapter 3), and (3) the on-line springback control loop (Loop III in Figure 1.22) to compensate for in-process springback (Chapter 4). All these three control loops are designed to be directly implemented in most of the current DSIF setups with a low implementation cost and minimal machine intrusion. Built on the solid foundations of process mechanics, the framework's generality is guaranteed for various complex geometries and process parameter combination. In doing so, this work successfully improves various aspects of DSIF, and paves the way for continued commercial implementation of the DSIF technology in the near future. It should be noted that all the different control loops (Loops I, II, III) are designed to be integrated together so that all these functionalities can be achieved simultaneously. In the system, compliance compensation and force controllers in Loop I are executed in real-time in the local controller. They are designed to correct the real tool tip positions based on force feedback. On the other hand, toolpath optimization for ADSIF and springback compensation for DSIF (Loops II and III) are operated offline or semi-real-time on the host computer. The command position of the tools will only be updated periodically without interfering with the real-time force and compliance compensation control.

In this chapter, a summary of each of the above-mentioned work will be presented as well as how it leads up to future work in the area of DSIF.

## 5.1. Compliance Compensation and Force Control

### *5.1.1. Research Summary*

Chapter 2 describes a compliance compensation algorithm to reduce tool positioning errors ($\delta^1$) caused by the tool compliance, and a force control algorithm to ensure the optimum contact pressure and avoid any early loss of contact between the tool and the part. Both algorithms are directly superimposed over the conventional position servo control loop on a real-time base with minimal modifications to the original process setup and servo control algorithm. The establishment of the contact stiffness model increases the algorithms' generality and robustness for different machine setups and material choices. Based on the contact stiffness model, a high-validity simulation model has been established and the algorithms' performance and robustness has been verified in this simulation environment. Finally, the developed algorithms have been tested through virous experiments, and the effectiveness of the approach was proven over a large range of process parameters. The following are the key achievements:

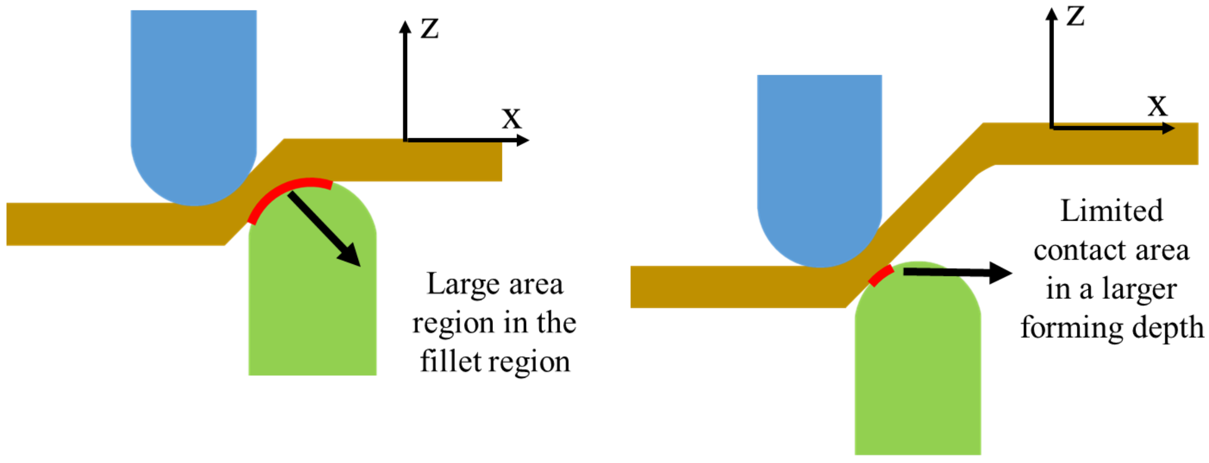
- A control scheme called explicit force control has been implemented in both the compliance compensation and force control algorithms. In explicit force control, the force control signal only plays the role of a modifier to the command position signal without further interfering with the inner position control loop, which enables the algorithms to be performed in most CNC or motion controllers.
- Considering the quasi-static nature of the DSIF process, A spring contact system has been established to model the local tool-sheet interaction during the forming process. The contact stiffness of the spring system depends on the tool compliance for both the forming tool and supporting tool and the squeezing stiffness of the sheet metal.
- A simple integral controller is implemented in the force controller due to its low-pass nature and, more importantly, zero steady-state errors even under large disturbances for a constant desired contact force. On the other hand, a negative proportional controller is implemented for the compliance compensation algorithm. The stability limits for the integral gain and proportional gain are also analyzed for each algorithm.
- A discrete time model has been established in MATLAB Simulink® to model the behavior of the control system in DSIF. Both the force control algorithm and compliance compensation algorithm have been tested via the simulator. The details of the filter and limiter design have also been validated in the simulation environment.
- The simulated algorithms were then implemented in a Delta Tau Turbo PMAC controller via a complied PLC program. The force control algorithm has been proven to be able to maintain a force accuracy of ±10 *N* for a total 300 *N* range while the compliance compensation algorithm can reduce the tool's deflection up to 95%. The

generality and effectiveness of the algorithm has been demonstrated via complex geometries with various thickness and material types. A pyramid part was also formed with the command contact force varying between adjacent walls.

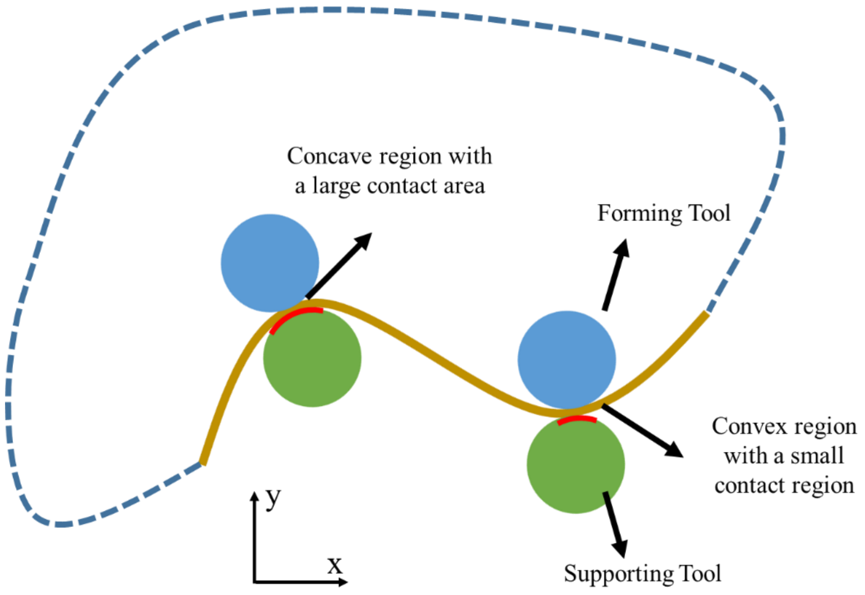
### *5.1.2. Recommendation for Future Work*

For the force control algorithm developed in Chapter 2, the contact force can be controlled during the forming process given a target forming force. However, the specified forming force value is still determined on a trial-and-error base and may not be the optimal for the specific forming material or part geometry. For instance, if a soft material is being formed, such as the cone part formed with AA1100-O in Section 2.4.1, it would be possible that the target force may be too high, and then the tools will over squeeze the metal sheet and damage the surface finish. On the other hand, if the target force is too low, the contact pressure might be inadequate to inhibit possible thinning or fracture. Moreover, the local geometry could also influence the local contact area, and thus alter the desired forming force if the same contact pressure is needed. For example, the effects of forming depth and in-plane curvature on the contact area of the supporting tool are shown in Figure 5.1, in which a larger contact area can be observed at the initial fillet or a concave region.

To guarantee optimal forming conditions and avoid over- or under-squeeze, a force prediction model will be required to derive the target force values based on different sheet metal material, machine setups or the part's geometric features. In the force prediction model, the contact force can be evaluated based on the integration of the stress tensors over the contact area. The stress filed can usually be estimated based on the yield stress of the local material, which is determined based on the strain history calculations and the materials' hardening behavior. On the hand, the contact area, assumed to be represented by the interference between the outer surface of the tool and the surface of the part, can will be calculated based on purely geometric considerations.


Z
X
Large area region in the fillet region
Z
X
Limited contact area in a larger forming depth

**(a)**



**(b)**

**Figure 5.1 The effect of local geometry on contact area due to different (a) forming depth (cross-section view) and (b) in-plane curvature (top view)**

Another possible enhancement is that implementation of adaptive control into the current algorithm. In the simplest application one could detect what is the following error of the current force signal in real-time and alter tool motion speeds depending on the level of detected force error. This strategy could be utilized for some large simple parts with sharp corners, such as the pyramid parts, in which the tool can accelerate at the flat edge and slow down once it enters the corners. A more advanced version of the adaptive controller would be to vary the integral gain of the algorithm based on the contact stiffness of the system, which is characterized based on an on-line process identification model. By doing so, any variation of the contact stiffness during the forming process could be well accounted for, and the overall performance of the system could be further improved.

## 5.2. Off-line Toolpath Optimization for Accumulative-DSIF

### *5.2.1. Research Summary*

Chapter 3 describes an off-line toolpath optimization loop which optimizes the relative tool positions of the forming tools in ADSIF to minimize the in-process springback ($\delta^2$). The toolpath optimization algorithm is designed specifically for an innovative toolpath strategy called Accumulated Double-Sided Incremental Forming (ADSIF). The relative tool position of the tools in ADSIF is, for the first time, defined by two input parameters, i.e., tool angle and tool gap, to capture the bending and squeezing phenomena in ADSIF. These process parameters are then optimized through a response surface model as a function of the local geometric wall angle based on a validated full-scale FEM simulation model. The influence of the horizontal tool deflection on the relative tool positions is also investigated, and a complete toolpath design algorithm is proposed via the developed optimization method utilized without the need for any feedback sensors. Finally, the proposed algorithm is validated on an axisymmetric part, and a significant geometric error reduction has been observed. The followings are the chief findings:

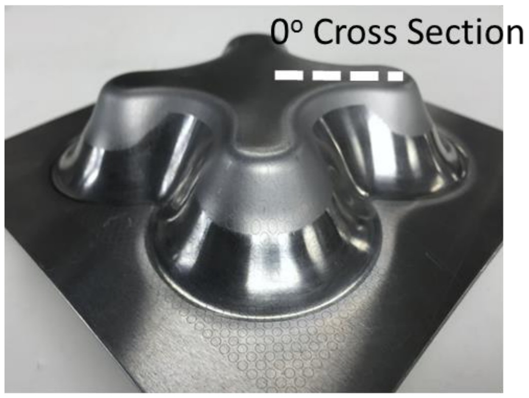
❖ Two control variables, called tool angle and tool gap, are first-time selected to define the relative tool positions of the forming tools in ADSIF. Primarily, the tool angle presents the level of the bending effect during the ADSIF, while the tool gap suggests the magnitude of the squeezing of both tools applied to the sheet material.

- A full-scale FEM simulation model has been proposed to study the effects of the two process parameters. The simulation model is established based on the explicit integration scheme with the sheet metal being modeled with reduced integration solid elements. A global damping is applied in the simulation model with accelerated tool motion speeds to reduce the simulation computational time.
- The response of the wall angle due to variations in position angle and tool gap shows that the design space can be divided into three different regions: squeezing-dominant region, bending-dominant region, and competing-region. When the tool gap is small ($\leq 0.7$), the wall angle is primarily dependent on the level of the tool gap. When the tool gap is larger than 1, the final geometric shape is formed by bending and stretching instead of squeezing. In the middle competing region, both tool gap and position angle influence the final geometry. The response surface retrieved can be used inversely to determine the position angle and tool gap values for the desired wall angle.
- The effects of the angle variation and forming force can also be investigated through the simulation results. Generally, it is observed that a large position angle will cause a large angle variation while the small tool gap will induce a large forming force. Either of these two indicators could be utilized as a minimization function to further optimize the possible solutions defined by the response surface model.
- The influence of the tool deflection on the tool gap and angle variation is then verified based on an analytical model as well as a simulation model. It has been shown that the tool deflection enlarges the tool gap while tool angle deviations caused by tool deflection are negligible.

- ❖ A full toolpath off-line optimization algorithm is proposed to optimize the tool gap and tool position angle. The target geometry is first discretized into a sequence of desired wall angles. For each desired wall angle, the tool gap and angle variation are then determined based on the response surface model, which is retrieved from the full-scale simulation model. The process parameters are then optimized based on either the angle variation, forming a force, or the combination of both. Finally, the tool gap in the selected solution will be decreased to compensate for the effects of the deflection.
- ❖ The proposed algorithm is finally implemented in ADSIF of an axisymmetric varying-wall angle part. The part formed via the proposed ADSIF toolpath optimization method is compared against the one manufactured by the conventional DSIF method to show the effectiveness of the algorithm.

### ***5.2.2. Recommendation for Future Work***

For the toolpath optimization algorithm proposed in Chapter 3, the accuracy of the simulation model determines the accuracy of the response surface model, and thus the veracity of the whole algorithm. To improve the simulation accuracy, the physical effects such as kinematic hardening, material anisotropy, and the Bauschinger effect need to be considered in the material model, so that the deformation mechanisms relevant to the DSIF process can be better captured. For example, instead of a monotonic strain path, the material point usually undergoes a complex bending – unbending process with both tools deforming the sheet in both DSIF and ADSIF. For example, a nonlinear strain path has been observed in the full-scale simulations of the shamrock part (Moser et al., 2016b) as shown in Figure 5.2.



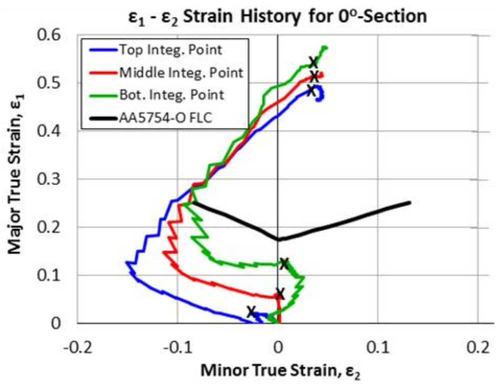
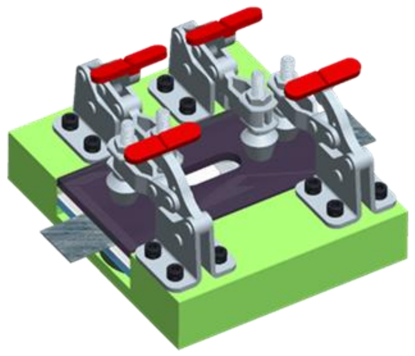
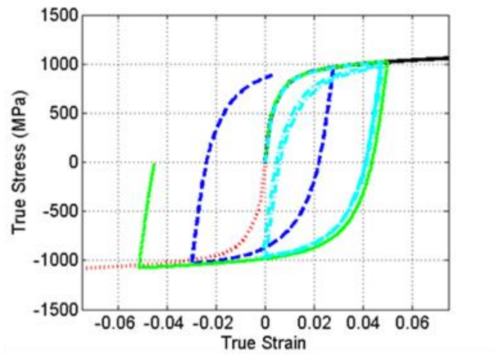


**Figure 5.2 Example of a nonlinear strain path in DSIF (Moser et al., 2016b)**

Currently, at Northwestern University, the author is developing a low-cost, transparent fixture to measure the kinematic hardening behavior of thin sheet metal as shown in Figure 5.3. The proposed fixture has proven to be capable of characterizing high strength steels like DP980 and aluminum alloy AA7075-O. Future efforts need to be put on the characterization work of AA5754-O, which is a typical material utilized in the current ADSIF setup, and the development of advanced material models, which can accurately capture the kinematic hardening behavior observed through the experimental data.


NU Designed Compact Tension-Compression Test Fixture



Tension-Compression Curve for DP980

**Figure 5.3 NU designed material test fixture and sample curves for DP980**

## 5.3. On-line Springback Compensation Algorithm for DSIF

### *5.3.1. Research Summary*

Chapter 4 describe an on-line springback compensation algorithm which dynamically modifies the toolpath to compensate for the potential in-process springback ($\delta^2$) based on real-time force signal feedback and a simulation-based model. The springback toolpath correction algorithm measures the force signals, predicts the springback value during the forming process and generates tool path adjustment vectors for the ongoing toolpath. The essence of the algorithm is a springback prediction function derived from a purely-elastic Finite Element Simulation, which can be executed in nearly real-time (in seconds) as compared to the full-scale simulation, facilitating its use in the springback control loop. A customized unsupervised classification algorithm is also proposed, for the first time, to extract the key control points for a given complex geometry, so that only a limited number of springback predictions is needed and the total calculation time can be further reduced. Finally, the proposed algorithms are validated through an axisymmetric cone and a truncated pyramid. A marked improvement of the corresponding geometric accuracy has been observed in both cases. The following are the main accomplishment:

- An exponential function is fitted through a pure elastic FEM simulation method to calculate the local springback value based on the measured forming forces. In the simulation model, the meshed surface of the interested part is generated based on the individual target geometry and machine setup, which implies that the proposed springback prediction model will be applicable for various complex part geometries and different machine/material setups.
- A control point determination algorithm is proposed to intelligently select the most representative locations, at which the exponential functions are evaluated via the simulation model. The algorithm is based on a discrete unsupervised clustering algorithm, which classifies a series of discrete tool position points into different groups. The control points are then located on the boundary or the center of each group. The algorithm can be executed without the need for any human interaction, and it has been proven to be effective for complex geometries with various geometric features.
- Based on the simulation-based springback prediction method and control points determination algorithm, an on-line force control algorithm is proposed to compensate for the in-process springback error during the forming process. In the algorithm, the tool path is compensated on a trajectory contour-wise base, i.e., the force signal measured at one contour is be utilized to calculate the springback error and tool

compensation value for the next loop. The whole algorithm is automated through a sequence of MATLAB and Python scripts, which minimizes human interruption during the execution.

- The proposed methods have been tested via the axisymmetric cone geometry and a truncated pyramid geometry. Considerable reduction of the geometric error can be observed in both cases verifying, thereby, the capability of the on-line springback compensation algorithm.

### *5.3.2. Recommendation for Future Work*

In the compensation algorithm described in Chapter 4, the main assumption is that the modification to a specific toolpath point will only change the being-formed local part geometry. However, there exist some global effects in DSIF that are not taken into account of, as the tool-sheet interaction will influence not only the local sheet, but also induce further deformation in either the already-formed part or the unformed virgin material. Two examples of these global effects, such as global bending and the pillow effect, are shown in Figure 5.4. As one can see, the pillow effect happens in the center of the part while some global bending appears around the outer periphery. Note that these regions are never in direct contact with the tools throughout the forming process. Instead, the deformation is caused by the forming forces in the surrounding area. To compensate for such global geometric errors, one of the simplest option is to reduce the forming forces near the center region or the periphery by utilizing small forming tools or small incremental depth. Another option is to expand the tool path so that the tools will not only form the original part , i.e., the side wall for the truncated in Figure 5.4, but also the center/periphery regions where these global deformations happen, and thus any unwanted geometric error could be corrected. However, how to identify the global deformation region for a complex geometry and how to generate the toolpath in this region are still challenging and requires future investigation.

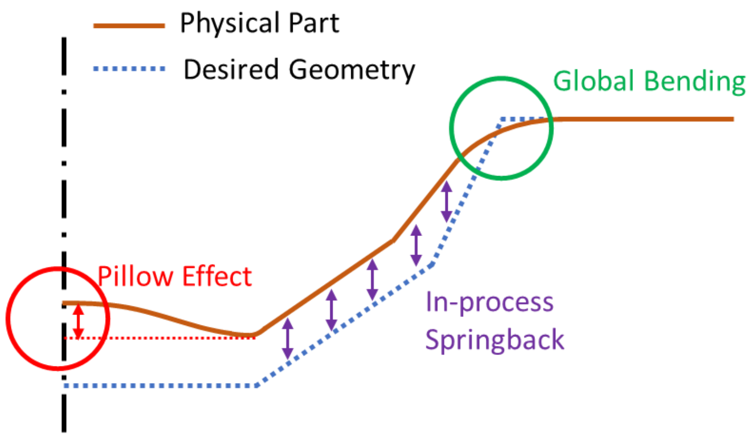


**Figure 5.4 An illustration of possible geometric errors in a DSIF physical part compared with the desired geometry**

Another possible enhancement is the establishment of a more seamless integrated user interface (and associated programs) which combine all the three control loops, i.e., the force control and compliance compensation, tool path optimization for ADSIF and springback compensation for DSIF. Although all these three algorithms are designed to be able to work together to improve the geometric accuracy and structure integrity, they are still now executing separately and require different programs/scripts to start and execute. To simplify the required user interaction and streamline the operation procedure, a unified user interface will be necessary. In the future, users can generate the toolpath, choose the compensation strategy and form the part in any connected machine by merely following the instructions of the interface. Such a user interface could serve as a prototype for a potential open-source or commercial software, which will advance the broader industrialization of the DSIF process.

Finally, due to its high manufacturing flexibility, the DSIF process is extremely suitable for small batch or even single product manufacturing. This, however, also requires that the DSIF process/machine be adaptable to various customer needs and process options, such as different material, part size and geometric complexity. In this case, the information about the process options must be provided by the potential customer and transferred to the production manager and local manufacturer (or machine operator) in an efficient way. On the other hand, the information regarding the forming of each individual part also needs to be transferred from the local machine to the production manager and customers, to enable a better process management and resources allocation. Therefore, a cloud-based manufacturing platform is desired to empower and manage these information transformations. At Northwestern University, a prototype platform, i.e., an Operating System for Cyberphysical Manufacturing, is under construction to meet the above-mentioned requirements, and will be introduced in detail in Appendix A.

# References

Allwood, J.M., Duncan, S., Cao, J., Groche, P., Hirt, G., Kinsey, B., Kuboki, T., Liewald, M., Sterzing, A., Tekkaya, A., 2016. Closed-loop control of product properties in metal forming. CIRP Annals 65, 573-596.

Allwood, J.M., Music, O., Raithathna, A., Duncan, S.R., 2009. Closed-loop feedback control of product properties in flexible metal forming processes with mobile tools. CIRP annals 58, 287-290.

Alsabti, K., Ranka, S., Singh, V., 1997. An efficient k-means clustering algorithm.

Ambrogio, G., Costantino, I., De Napoli, L., Filice, L., Fratini, L., Muzzupappa, M., 2004. Influence of some relevant process parameters on the dimensional accuracy in incremental forming: a numerical and experimental investigation. Journal of Materials Processing Technology 153, 501-507.

Ambrogio, G., Cozza, V., Filice, L., Micari, F., 2007. An analytical model for improving precision in single point incremental forming. Journal of Materials Processing Technology 191, 92-95.

Ambrogio, G., Filice, L., Gagliardi, F., 2012. Formability of lightweight alloys by hot incremental sheet forming. Materials & Design 34, 501-508.

Ambrogio, G., Filice, L., Gagliardi, F., Micari, F., 2005. Sheet thinning prediction in single point incremental forming, Advanced materials research. Trans Tech Publ, pp. 479-486.

Anderson, M.J., 2001. A new method for non-parametric multivariate analysis of variance. Austral ecology 26, 32-46.

Astrom, K.J., Rundqwist, L., 1989. Integrator windup and how to avoid it, American Control Conference, 1989. IEEE, pp. 1693-1698.

Attanasio, A., Ceretti, E., Giardini, C., Mazzoni, L., 2008. Asymmetric two points incremental forming: improving surface quality and geometric accuracy by tool path optimization. Journal of materials processing technology 197, 59-67.

Bailly, D., Bambach, M., Hirt, G., Pofahl, T., Della Puppa, G., Trautz, M., 2015. Investigation on the producibility of freeform facade elements made of sheet metal as self-supporting structures by means of incremental sheet forming. Steel Institute VDEh, Düsseldorf 15.

Bambach, M., 2010. A geometrical model of the kinematics of incremental sheet forming for the prediction of membrane strains and sheet thickness. Journal of Materials Processing Technology 210, 1562-1573.

Bambach, M., Araghi, B.T., Hirt, G., 2009. Strategies to improve the geometric accuracy in asymmetric single point incremental forming. Production Engineering 3, 145-156.

Batagelj, V., 1988. Generalized Ward and related clustering problems. Classification and related methods of data analysis, 67-74.

Behera, A., 2013. Shape feature taxonomy development for toolpath optimisation in incremental sheet forming. PhD Thesis. Katholieke Universiteit Leuven ISBN 978-94-6018-733-9.

Behera, A.K., Lauwers, B., Duflou, J.R., 2014. Tool path generation framework for accurate manufacture of complex 3D sheet metal parts using single point incremental forming. Computers in Industry 65, 563-584.

Behera, A.K., Verbert, J., Lauwers, B., Duflou, J.R., 2013. Tool path compensation strategies for single point incremental sheet forming using multivariate adaptive regression splines. Computer-Aided Design 45, 575-590.

Belchior, J., Guillo, M., Courteille, E., Maurine, P., Leotoing, L., Guines, D., 2013. Off-line compensation of the tool path deviations on robotic machining: Application to incremental sheet forming. Robotics and Computer-Integrated Manufacturing 29, 58-69.

Belchior, J., Leotoing, L., Guines, D., Courteille, E., Maurine, P., 2014. A process/machine coupling approach: application to robotized incremental sheet forming. Journal of Materials Processing Technology 214, 1605-1616.

Bjorck, A., 1996. Numerical methods for least squares problems. Siam.

Cai, Z.-Y., Li, M.-Z., Lan, Y.-W., 2012. Three-dimensional sheet metal continuous forming process based on flexible roll bending: principle and experiments. Journal of Materials Processing Technology 212, 120-127.

Caliński, T., Harabasz, J., 1974. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods 3, 1-27.

Castelan, J., Schaeffer, L., Daleffe, A., Fritzen, D., Salvaro, V., Silva, F.P.d., 2014. Manufacture of custom-made cranial implants from DICOM® images using 3D printing, CAD/CAM technology and incremental sheet forming. Revista Brasileira de Engenharia Biomédica 30, 265-273.

Davidson, I., Ravi, S., 2005. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results, European Conference on Principles of Data Mining and Knowledge Discovery. Springer, pp. 59-70.

Day, W.H., Edelsbrunner, H., 1984. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of classification 1, 7-24.

Dégoulange, E., Dauchez, P., 1994. External force control of an industrial PUMA 560 robot. Journal of Field Robotics 11, 523-540.

Duflou, J., Callebaut, B., Verbert, J., De Baerdemaeker, H., 2007. Laser assisted incremental forming: formability and accuracy improvement. CIRP Annals-Manufacturing Technology 56, 273-276.

Duncan, S.R., Allwood, J.M., Garimella, S.S., 1998. The analysis and design of spatial control systems in strip metal rolling. IEEE Transactions on control systems technology 6, 220-232.

Edward, L., 1967. Apparatus and process for incremental dieless forming. Google Patents.

Emmens, W., Sebastiani, G., van den Boogaard, A.H., 2010. The technology of incremental sheet forming—a brief review of the history. Journal of Materials Processing Technology 210, 981-997.

Fan, G., Sun, F., Meng, X., Gao, L., Tong, G., 2010. Electric hot incremental forming of Ti-6Al-4V titanium sheet. The International Journal of Advanced Manufacturing Technology 49, 941-947.

Filice, L., Ambrogio, G., Micari, F., 2006. On-line control of single point incremental forming operations through punch force monitoring. CIRP annals-Manufacturing technology 55, 245-248.

Finch, H., 2005. Comparison of distance measures in cluster analysis with dichotomous data. Journal of Data Science 3, 85-100.

Finckenstein, E., Kleiner, M., 1991. Flexible numerically controlled tool system for hydro-mechanical deep drawing. CIRP annals 40, 311-314.

Fiorentino, A., Feriti, G.C., Giardini, C., Ceretti, E., 2015. Part precision improvement in incremental sheet forming of not axisymmetric parts using an artificial cognitive system. Journal of Manufacturing Systems 35, 215-222.

Fu, Z., Mo, J., Han, F., Gong, P., 2013. Tool path correction algorithm for single-point incremental forming of sheet metal. The International Journal of Advanced Manufacturing Technology 64, 1239-1248.

Gan, W., Wagoner, R., 2004. Die design method for sheet springback. International Journal of Mechanical Sciences 46, 1097-1113.

Gan, W., Wagoner, R., Mao, K., Price, S., Rasouli, F., 2004. Practical methods for the design of sheet formed components. Journal of engineering materials and technology 126, 360-367.

Garcia, C.E., Prett, D.M., Morari, M., 1989. Model predictive control: theory and practice—a survey. Automatica 25, 335-348.

Gatea, S., Ou, H., McCartney, G., 2016. Review on the influence of process parameters in incremental sheet forming. The International Journal of Advanced Manufacturing Technology 87, 479-499.

Göttmann, A., Diettrich, J., Bergweiler, G., Bambach, M., Hirt, G., Loosen, P., Poprawe, R., 2011. Laser-assisted asymmetric incremental sheet forming of titanium sheet metal parts. Production Engineering 5, 263-271.

Groche, P., Beiter, P., Henkelmann, M., 2008. Prediction and inline compensation of springback in roll forming of high and ultra-high strength steels. Production Engineering 2, 401.

Hao, W., Duncan, S., 2011. Optimization of tool trajectory for Incremental Sheet Forming using closed loop control, Automation Science and Engineering (CASE), 2011 IEEE Conference on. IEEE, pp. 779-784.

Hartigan, J.A., Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, 100-108.

Hirt, G., Ames, J., Bambach, M., Kopp, R., 2004. Forming strategies and process modelling for CNC incremental sheet forming. CIRP Annals-Manufacturing Technology 53, 203-206.

Hu, J., Marciniak, Z., Duncan, J., 2002. Mechanics of sheet metal forming. Elsevier.

Ingarao, G., Ambrogio, G., Gagliardi, F., Di Lorenzo, R., 2012. A sustainability point of view on sheet metal forming operations: material wasting and energy consumption in incremental forming and stamping processes. Journal of Cleaner Production 29, 255-268.

Iseki, H., 1996. Practical development of press-molding machine with small punching tool, Proc. 5th Int. Conf. on Advanced Technology of Plasticity, p. 935.

Iseki, H., 2001. Flexible and incremental bulging of sheet metal using high-speed water jet. JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing 44, 486-493.

Iseki, H., Kato, K., Sakamoto, S., 1989. Flexible and incremental sheet metal forming using a spherical roller. Proc. 40th JJCTP 41, 44.

Jeon, D., Tomizuka, M., 1993. Learning hybrid force and position control of robot manipulators. IEEE Transactions on Robotics and Automation 9, 423-431.

Jeswiet, J., Micari, F., Hirt, G., Bramley, A., Duflou, J., Allwood, J., 2005. Asymmetric single point incremental forming of sheet metal. CIRP Annals-Manufacturing Technology 54, 88-114.

Jurisevic, B., 2003. Incremental sheet metal forming with a high-speed water jet, Proc. IDDRG 2003 Conf., pp. 139-148.

Kalo, A., Newsum, M.J., 2014. An investigation of robotic incremental sheet metal forming as a method for prototyping parametric architectural skins, Robotic Fabrication in Architecture, Art and Design 2014. Springer, pp. 33-49.

Karafillis, A., Boyce, M., 1992. Tooling design in sheet metal forming using springback calculations. International journal of mechanical sciences 34, 113-131.

Karafillis, A.P., Boyce, M.C., 1996. Tooling and binder design for sheet metal forming processes compensating springback error. International Journal of Machine Tools and Manufacture 36, 503-526.

Klocke, F., Brummer, C.M., 2014. Laser-assisted metal spinning of challenging materials. Procedia Engineering 81, 2385-2390.

Kopp, R., Schulz, J., 2002. Flexible sheet forming technology by double-sided simultaneous shot peen forming. CIRP Annals-Manufacturing Technology 51, 195-198.

Kothare, M.V., Balakrishnan, V., Morai, M., 1994. Robust constrained model predictive control using linear matrix inequalities, American Control Conference, 1994. IEEE, pp. 440-444.

Li, J., He, K., Wei, S., Dang, X., Du, R., 2014. Modeling and experimental validation for truncated cone parts forming based on water jet incremental sheet metal forming. The International Journal of Advanced Manufacturing Technology 75, 1691-1699.

Li, M., Liu, Y., Su, S., Li, G., 1999. Multi-point forming: a flexible manufacturing method for a 3-d surface sheet. Journal of Materials Processing Technology 87, 277-280.

Li, W., Yao, Y.L., 2000. Numerical and experimental study of strain rate effects in laser forming. Journal of manufacturing science and engineering 122, 445-451.

Lim, Y., Venugopal, R., Ulsoy, A.G., 2009. Improved part quality in stamping using multi-input multi-output (MIMO) process control, American Control Conference, 2009. ACC'09. IEEE, pp. 5570-5575.

Lingam, R., Prakash, O., Belk, J., Reddy, N., 2017. Automatic feature recognition and tool path strategies for enhancing accuracy in double sided incremental forming. The International Journal of Advanced Manufacturing Technology 88, 1639-1655.

Lingbeek, R., Gan, W., Wagoner, R., Meinders, T., Weiher, J., 2008. Theoretical verification of the displacement adjustment and springforward algorithms for springback compensation. International journal of material forming 1, 159-168.

Liu, C., Li, M., Fu, W., 2008. Principles and apparatus of multi-point forming for sheet metal. The International Journal of Advanced Manufacturing Technology 35, 1227-1233.

Liu, C., Yao, Y.L., Srinivasan, V., 2004. Optimal process planning for laser forming of doubly curved shapes. Journal of manufacturing science and engineering 126, 1-9.

Lu, B., Fang, Y., Xu, D., Chen, J., Ai, S., Long, H., Ou, H., Cao, J., 2015. Investigation of material deformation mechanism in double side incremental sheet forming. International Journal of Machine Tools and Manufacture 93, 37-48.

Lu, H., Kearney, M., Li, Y., Liu, S., Daniel, W.J., Meehan, P.A., 2016. Model predictive control of incremental sheet forming for geometric accuracy improvement. The International Journal of Advanced Manufacturing Technology 82, 1781-1794.

Lu, H., Kearney, M., Liu, S., Daniel, W.J., Meehan, P.A., 2017a. Two-directional toolpath correction in single-point incremental forming using model predictive control. The International Journal of Advanced Manufacturing Technology 91, 91-106.

Lu, H., Kearney, M., Wang, C., Liu, S., Meehan, P.A., 2017b. Part accuracy improvement in two point incremental forming with a partial die using a model predictive control algorithm. Precision Engineering 49, 179-188.

Maidagan, E., Zettler, J., Bambach, M., Rodríguez, P., Hirt, G., 2007. A new incremental sheet forming process based on a flexible supporting die system, Key Engineering Materials. Trans Tech Publ, pp. 607-614.

Malhotra, R., Cao, J., Beltran, M., Xu, D., Magargee, J., Kiridena, V., Xia, Z.C., 2012. Accumulative-DSIF strategy for enhancing process capabilities in incremental forming. CIRP Annals-Manufacturing Technology 61, 251-254.

Malhotra, R., Cao, J., Ren, F., Kiridena, V., Xia, Z.C., Reddy, N., 2011. Improvement of geometric accuracy in incremental forming by using a squeezing toolpath strategy with two forming tools. Journal of manufacturing science and engineering 133, 061019.

Matsubara, M., Tanaka, S., Nakamura, T., 1996. Development of incremental sheet metal forming system using elastic tools: Principle of forming process and formation of some fundamentally curved shapes. JSME international journal. Ser. C, Dynamics, control, robotics, design and manufacturing 39, 156-163.

McMurtry, D.R., Jarman, T.B., Bennett, S.J., 1992. Probe head. Google Patents.

Meier, H., Buff, B., Laurischkat, R., Smukala, V., 2009. Increasing the part accuracy in dieless robot-based incremental sheet metal forming. CIRP annals 58, 233-238.

Meier, H., Laurischkat, R., Zhu, J., 2011a. A model based approach to increase the part accuracy in robot based incremental sheet metal forming, AIP conference proceedings. AIP, pp. 1407-1412.

Meier, H., Magnus, C., 2013. Incremental sheet metal forming with direct resistance heating using two moving tools, Key Engineering Materials. Trans Tech Publ, pp. 1362-1367.

Meier, H., Magnus, C., Smukala, V., 2011b. Impact of superimposed pressure on dieless incremental sheet metal forming with two moving tools. CIRP Annals-Manufacturing Technology 60, 327-330.

Meier, H., Smukala, V., Dewald, O., Zhang, J., 2007. Two point incremental forming with two moving forming tools, Key Engineering Materials. Trans Tech Publ, pp. 599-605.

Micari, F., Ambrogio, G., Filice, L., 2007. Shape and dimensional accuracy in single point incremental forming: state of the art and future trends. Journal of Materials Processing Technology 191, 390-395.

Mikolajek, M., Machacek, Z., Koziorek, J., 2014. Modern sensor technology for alphanumeric recognition in metallurgy industry. Elektronika ir Elektrotechnika 20, 3-7.

Milutinović, M., Potran, R.L.M., Vilotić, D., Skakun, P., Plančak, M., 2014. Application of single point incremental forming for manufacturing of denture base. Journal for Technology of Plasticity 39, 15-23.

Moser, N., 2018. AMPL Toolpaths. Advanced Manufacturing Processes Laboratory, Northwestern University.

Moser, N., Pritchet, D., Ren, H., Ehmann, K.F., Cao, J., 2016a. An efficient and general finite element model for double-sided incremental forming. Journal of Manufacturing Science and Engineering 138, 091007.

Moser, N., Zhang, Z., Ren, H., Ehmann, K., Cao, J., 2016b. An investigation into the mechanics of double-sided incremental forming using finite element methods, AIP Conference Proceedings. AIP Publishing, p. 070021.

Moser, N., Zhang, Z., Ren, H., Zhang, H., Shi, Y., Ndip-Agbor, E.E., Lu, B., Chen, J., Ehmann, K.F., Cao, J., 2016c. Effective forming strategy for double-sided incremental forming considering in-plane curvature and tool direction. CIRP Annals 65, 265-268.

Ndip-Agbor, E., Smith, J., Ren, H., Jiang, Z., Xu, J., Moser, N., Chen, W., Xia, Z.C., Cao, J., 2016. Optimization of relative tool position in accumulative double sided incremental forming using finite element analysis and model bias correction. International Journal of Material Forming 9, 371-382.

Nielsen, L., Lassen, S., Andersen, C.B., Groenbaek, J., Bay, N., 1997. Development of a flexible tool system for small quantity production in cold forging. Journal of materials processing Technology 71, 36-42.

Petek, A., Jurisevic, B., Kuzman, K., Junkar, M., 2009. Comparison of alternative approaches of single point incremental forming processes. Journal of materials processing technology 209, 1810-1815.

PMAC, T., 2008. PMAC2 Software Reference Manual. DELTA TAU Data Systems.

Powell, N., Andrew, C., 1992. Incremental forming of flanged sheet metal components without dedicated dies. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 206, 41-47.

Pratt, V., 1987. Direct least-squares fitting of algebraic surfaces, ACM SIGGRAPH computer graphics. ACM, pp. 145-152.

Raibert, M.H., Craig, J.J., 1981. Hybrid position/force control of manipulators. Journal of Dynamic Systems, Measurement, and Control 103, 126-133.

Rakesh, L., Amit, S., Reddy, N., 2016. Deflection compensations for tool path to enhance accuracy during double-sided incremental forming. Journal of Manufacturing Science and Engineering 138, 091008.

Rauch, M., Hascoet, J.-Y., Hamann, J.-C., Plenel, Y., 2009. Tool path programming optimization for incremental sheet forming applications. Computer-Aided Design 41, 877-885.

Reagan, J., Smith, E., 1991. Metal Spinning. Lindsay Publications, Bradley, IL.

Ren, H., Moser, N., Cao, J., 2016. Simulation and Analysis of Double-Sided Incremental Forming Considering Machine Compliance, 10th International Conference and Workshop on Numerical Simulation of 3D Sheet Metal Forming Processes, Bristol, UK.

Reynolds, K., 2015. How the toyota new global architecture will impact mass-car building.

Salisbury, J.K., Craig, J.J., 1982. Articulated hands: Force control and kinematic issues. The International journal of Robotics research 1, 4-17.

Shi, Y., Zhang, W., Cao, J., Ehmann, K., 2018. An Experimental and Numerical Study of Dieless Water Jet Incremental Micro-Forming. Journal of Manufacturing Science and Engineering.

Smith, J., Malhotra, R., Liu, W., Cao, J., 2013. Deformation mechanics in single-point and accumulative double-sided incremental forming. The International Journal of Advanced Manufacturing Technology 69, 1185-1201.

Steinbach, M., Karypis, G., Kumar, V., 2000. A comparison of document clustering techniques, KDD workshop on text mining. Boston, pp. 525-526.

Stew, J., 2017. Best English Wheel Reviews Hand Tools.

Tekkaya, A.E., Shankar, R., Sebastiani, G., Homberg, W., Kleiner, M., 2007. Surface reconstruction for incremental forming. Production Engineering 1, 71-78.

Timoshenko, S.P., Woinowsky-Krieger, S., 1959. Theory of plates and shells. McGraw-hill.

Turbo, P., 2008. USER MANUAL, Programmable Multi Axis Controller, Delta Tau Data Systems. Inc.< http://www. deltatau. com, 292-294.

Valoppi, B., Egea, A.J.S., Zhang, Z., Rojas, H.A.G., Ghiotti, A., Bruschi, S., Cao, J., 2016. A hybrid mixed double-sided incremental forming method for forming Ti6Al4V alloy. CIRP Annals 65, 309-312.

Verbert, J., 2010. Computer aided process planning for rapid prototyping with incremental sheet forming techniques. Katholieke Universiteit Leuven, Belgium.

Volpe, R., Khosla, P., 1993. A theoretical and experimental investigation of explicit force control strategies for manipulators. IEEE Transactions on Automatic Control 38, 1634-1650.

Walczyk, D.F., Hardt, D.E., 1994. A new rapid tooling method for sheet metal forming dies, Proceedings of the 5th international conference on rapid prototyping, pp. 12-15.

Walczyk, D.F., Hardt, D.E., 1998. Design and analysis of reconfigurable discrete dies for sheet metal forming. Journal of Manufacturing Systems 17, 436-453.

Wang, S., Ehmann, K.F., 1994. Compensation of geometric and quasi-static deformation errors of a multi-axis machine. Transactions of NAMRI/SME 22, 283-289.

Wang, Y., Huang, Y., Cao, J., Reddy, N.V., 2008. Experimental study on a new method of double side incremental forming, ASME 2008 International Manufacturing Science and Engineering Conference collocated with the 3rd JSME/ASME International Conference on Materials and Processing. American Society of Mechanical Engineers, pp. 601-607.

Watanabe, T., Iwai, S., 1983. A control system to improve the accuracy of finished surfaces in milling. Journal of dynamic systems, measurement, and control 105, 192-199.

Whitney, D.E., 1977. Force feedback control of manipulator fine motions. Journal of Dynamic Systems, Measurement, and Control 99, 91-97.

Whitney, D.E., 1987. Historical perspective and state of the art in robot force control. The International Journal of Robotics Research 6, 3-14.

Wong, C., Dean, T., Lin, J., 2003. A review of spinning, shear forming and flow forming processes. International Journal of Machine Tools and Manufacture 43, 1419-1435.

Xu, R., Ren, H., Zhang, Z., Malhotra, R., Cao, J., 2014a. A mixed toolpath strategy for improved geometric accuracy and higher throughput in double-sided incremental forming, ASME 2014 international manufacturing science and engineering conference collocated with the JSME 2014 international conference on materials and processing and the 42nd North American manufacturing research conference. American Society of Mechanical Engineers, pp. V002T002A082-V002T002A082.

Xu, R., Shi, X., Xu, D., Malhotra, R., Cao, J., 2014b. A preliminary study on the fatigue behavior of sheet metal parts formed with accumulative-double-sided incremental forming. Manufacturing Letters 2, 8-11.

Yang, M., Choi, J., 1998. A tool deflection compensation system for end milling accuracy improvement. Journal of manufacturing science and engineering 120, 222-229.

Young, D., Jeswiet, J., 2004. Wall thickness variations in single-point incremental forming. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 218, 1453-1459.

Zhang, Z., Ren, H., Xu, R., Moser, N., Smith, J., Ndip-Agbor, E., Malhotra, R., Xia, Z.C., Ehmann, K.F., Cao, J., 2015. A mixed double-sided incremental forming toolpath strategy for improved geometric accuracy. Journal of Manufacturing Science and Engineering 137, 051007.

Zhang, Z., Zhang, H., Shi, Y., Moser, N., Ren, H., Ehmann, K.F., Cao, J., 2016. Springback reduction by annealing for incremental sheet forming. Procedia Manufacturing 5, 696-706.

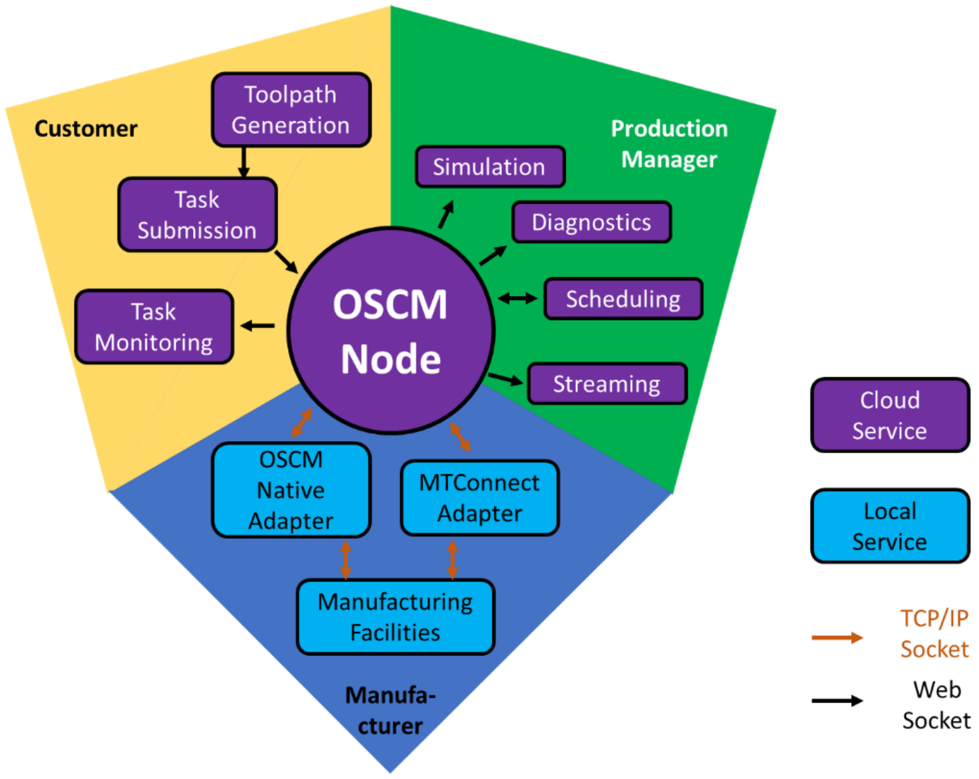
# Appendix A: Operating System for Cyberphysical Manufacturing in DSIF

Due to the high customization potential of the DSIF process (or of any other IF process), it is critical for the DSIF process to react quickly to any change in customer demands regarding various product requirements, as reviewed earlier in Section 5.3.2. In this Appendix, a manufacturing-as-a-service platform called Operation System for Cyberphysical Manufacturing (OSCM) under development is briefly described, to show how the DSIF process can be provided as a flexible manufacturing service to not only local manufacturers, but also to production managers and the final customers. Moreover, the OSCM platform can also integrate other manufacturing processes, such as machining or additive manufacturing, so that a more complex part can be manufactured with a chain of manufacturing processes, in which DSIF is one critical element. The OSCM platform is a joint project between the University of Illinois at Urbana-Champaign (UIUC), Northwestern University (NU) and the Digital Manufacturing and Design Innovation Institute (DMDII). The basic details of its implementation will be explained below.

In this Appendix, the overall structure of the OSCM platform will be first discussed in Section A-1. A user interface will be shown in Section A-2 with the detail code for the server application shown in Section A-3.

## *A-1 Overall Structure of the OSCM Platform*

The OSCM platform is a cloud-based platform that links the local manufacturing facilities of different manufacturers to the production manager (or supply chain manager) and the end customer. The core of the platform is a cloud-based server that connects all the associated services and stores the user data information. The OSCM node first gathers information from the individual manufacturing facilities through the OSCM adapter or MTConnect adapter. The gathered information is then streamed to other services like diagnostics or monitoring applications. On the other hand, the customers can generate the toolpath based on the desired part, and submit the toolpath, i.e., the manufacturing task to the OSCM node. Based on the tasks received from the customers and the facilities availability collected from the local manufacturers, the production managers distribute the tasks in a manner to minimize the overall lead time and transportation cost. Once the task distribution is determined, the production manager sends the toolpath to the assigned manufacturing facility through the OSCM node, and the customer can watch the manufacturing progress through the task monitoring services. The whole OSCM Node is shown in Figure A-1, in which the cloud services and local services are denoted by different colors. The information flow direction between each service and the OSCM node is also shown by the arrow.



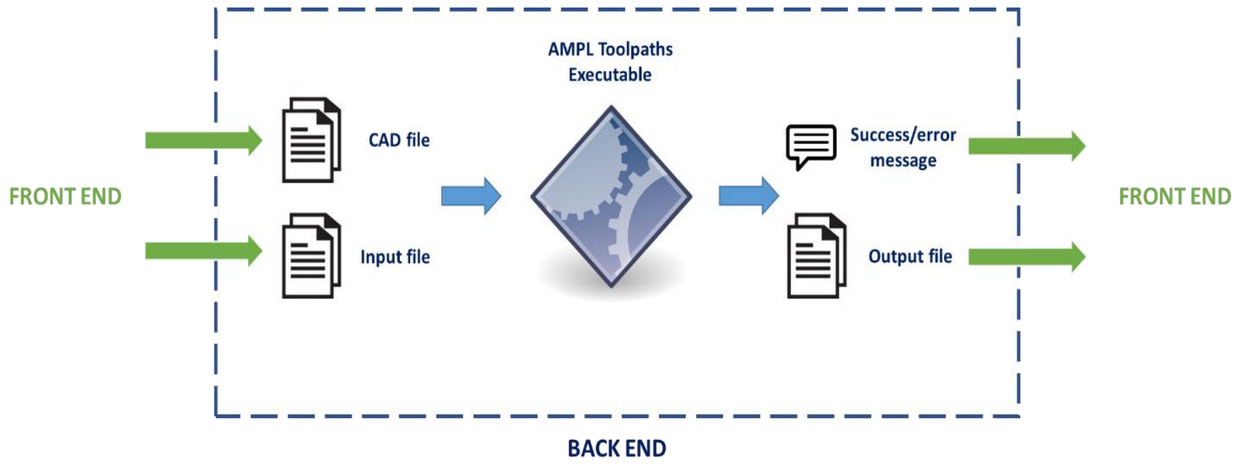
**Appendix Figure A-1 Overall architecture of the OSCM platform**

Note that in the OSCM platform, the manufacturing facilities are not limited to the DSIF process machine, but also could include other manufacturing processes. Thus, the whole OSCM platform has the potential to create a huge accessible manufacturing capability throughout the manufacturing industry. Once entirely built, the OSCM platform will allow production managers to easily and efficiently access the capacity and capabilities of individual manufacturing resources in the system, enable the customers to efficiently enter, execute and complete individual manufacturing tasks, and permit the manufacturing service providers to easily and conveniently make their resources available on the network.

## *A-2 User Interface Example for the OSCM Platform*

In developing of the OSCM platform, all the services are constructed around the network base so that users can access any service without the need for any specific software or hardware. Here, the cloud-based tool generation service is provided as one illustrative example. The development of this toolpath generation service allows the users of the DSIF process to generate their customized machine codes. It represents an example of a cloud service that can be added to the OSCM platform. The executable of the software can be treated as a black box, and the OSCM platform can gather inputs to the software from the users, run the software and return the outputs, which will be accessible to the users. This can be done by any third-party software that can run in the command prompt mode by altering only a few lines of the Toolpath asset code.

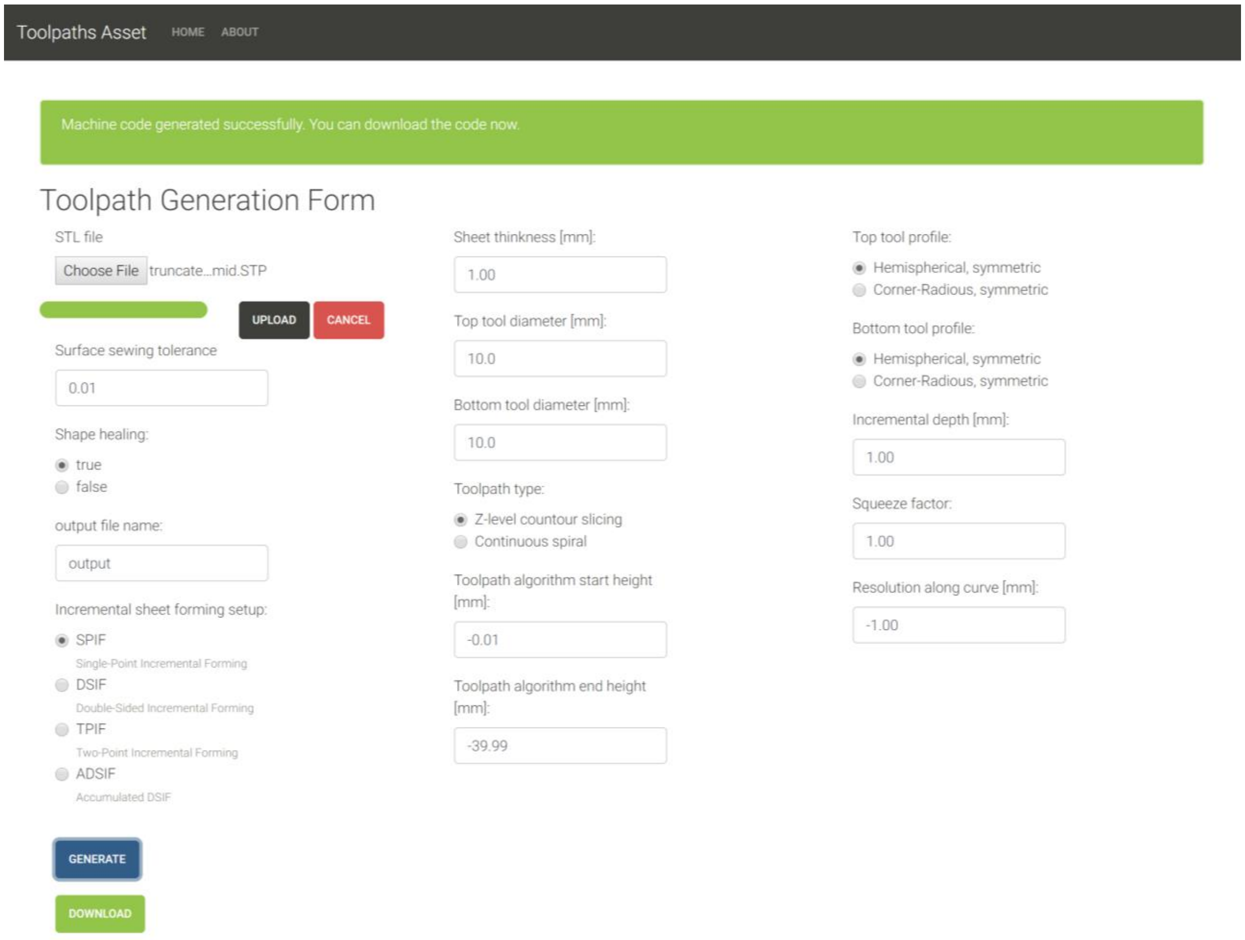
The program which is adopted as the base of this service is an in-house C++ code with a graphical user interface. First, a version of the toolpaths C++ code, called AMPL Toolpath as described in Chapter 2, is developed, which enables the direct execution of the program using the command prompt. The overall schematic of the AMPL Toolpath is demonstrated in Figure A-2.



**Appendix Figure A-2 Schematic of the AMPL Toolpath and its data-flow**

The inputs to the AMPL Toolpaths executable are: (i) the final geometry CAD file in STL format and (ii) the toolpath generation parameters and setups. The latter is created from the user-interface input and stored in a JSON file, which is then passed to the back-end for the execution of the program. Once the program is executed, the machine code output and a message (that determines the success or failure of the process) are transferred back to the front end. If the operation is successful, the user would be able to download the machine code.

The front-end of the toolpath generation service is programmed in JavaScript to enable the toolpath generation operation in a user-friendly interface. As shown in Figure A-3, the programmed front end allows users to upload CAD files, fill in the parameters of the process, generate the machine code and download the results within a single web page.


Toolpaths Asset HOME ABOUT

Machine code generated successfully. You can download the code now.

Toolpath Generation Form

STL file: Choose File truncate...mid.STP

UPLOAD CANCEL

Surface sewing tolerance: 0.01

Shape healing: [✓ true] false

output file name: output

Incremental sheet forming setup:
[✓ SPIF] Single-Point Incremental Forming
DSIF Double-Sided Incremental Forming
TPIF Two-Point Incremental Forming
ADSIF Accumulated DSIF

GENERATE

DOWNLOAD

Sheet thinkness [mm]: 1.00

Top tool diameter [mm]: 10.0

Bottom tool diameter [mm]: 10.0

Toolpath type: [✓ Z-level countour slicing] Continuous spiral

Toolpath algorithm start height [mm]: -0.01

Toolpath algorithm end height [mm]: -39.99

Top tool profile: [✓ Hemispherical, symmetric] Corner-Radious, symmetric

Bottom tool profile: [✓ Hemispherical, symmetric] Corner-Radious, symmetric

Incremental depth [mm]: 1.00

Squeeze factor: 1.00

Resolution along curve [mm]: -1.00

**Appendix Figure A-3 Front-end of the Toolpaths service (by the courtesy of Mojtaba Mozaffar in Northwestern University)**

## *A-3 C++ Application for the OSCM Native Adapter*

In the following subsection, a C++ based OSCM adapter code is listed to show the working principle of the TCP/IP connection that links the local facilities with OSCM node.

```
#pragma once

```

```
#include "stdafx.h"
//#include "Form1.h"
#include "parson.h"
#include <iostream>
#include <string>
#include <winsock2.h>        // needed for executive.h, Winsocket.h
#define WIN32_LEAN_AND_MEAN  // prevents windows.h from including winsock v1
#include <windows.h>         // needed for winsocks2.h
#include "Winsocket.h"

struct axis
{
    float X;
    float Y;
    float Z;
    float U;
    float V;
    float W;
};

struct offset
{
    float X;
    float Y;
    float Z;
    float U;
    float V;
    float W;
};

typedef struct
{
    int    sequence;           // current sequence number in the program
    axis   position;           // current position of the machine in the native coordina
te system (G53)
    offset origin_offset;      // origin_offsets for coordinates systems G54-
G59 as per G92 (same as H760, H761, H762)
    float  feed;               // feed in MM/min or  IN/min
    float  speed;              // spindle speed in RPM
    int    tool;               // tool number
    offset tool_offset;        // current tool X, Y and Z offsets (same as H800- H860)

    float  tool_radius;        // current tool raduis (same as H800- H860)
    int    coolant;            // DISABLE or ENABLE
    int    motion_mode;        // linear, CW, CCW, RAPID, DWELL
    int    plane;              // XY, YZ or XZ
    int    distance_mode;      // ABS or REL distance mode
    int    feed_mode;          // inverse time, units per minute or units per revolution
 mode
    int    units;              // MM or IN
    int    radius_comp;        // DISABLE, LEFT_TOOL or RIGHT_TOOL
    int    length_comp;        // DISABLE, ENABLE
    int    coordinate_system;  // G54 - G59
    int    retraction_mode;    // ORIGIN, SPECIFIED
    int    path_mode;          // EXACT, STOP, BLENDING
} Machine_vector;

```

```c

/************************************************************
Global variables
************************************************************/

Machine_vector machine_struct;
Machine_vector * machine_vector = &machine_struct;


/************************************************************
Function defs.
************************************************************/
char *serialized_machine_vector(void)
{
    JSON_Value *root_value = json_value_init_object();
    JSON_Object *root_object = json_value_get_object(root_value);
    char *serialized_string = NULL;

    char *axes[] = { "X", "Y", "Z", "U", "V", "W" };
    char *status[] = { "enabled", "disabled", "emergency" };
    char *feed_mode[] = { "inverse", "units per min", "units per rev" };
    char *planes[] = { "xy", "xz", "yz" };
    char *units[] = { "in", "mm" };
    char *radius_comp[] = { "off", "tool left" , "tool right" };
    char *length_comp[] = { "disable", "enable" };
    char *distance_mode[] = { "abs", "rel" };
    char *motion_mode[] = { "rapid", "linear", "cw", "ccw", "dwell" };
    char *coolant[] = { "on", "off" };

    json_object_set_string(root_object, "type", "machine_vector");

    /* print sequence number */
    json_object_set_number(root_object, "sequence_number", 0);

    json_object_dotset_string(root_object, "axes.axis_1.name", axes[0]);
    json_object_dotset_string(root_object, "axes.axis_1.status", "enabled");
    json_object_dotset_number(root_object, "axes.axis_1.native_position", machine_vector->position.X);
    json_object_dotset_number(root_object, "axes.axis_1.position", machine_vector->position.X);
    json_object_dotset_number(root_object, "axes.axis_1.target_position", machine_vector->position.X);

    json_object_dotset_string(root_object, "axes.axis_2.name", axes[1]);
    json_object_dotset_string(root_object, "axes.axis_2.status", status[0]);
    json_object_dotset_number(root_object, "axes.axis_2.native_position", machine_vector->position.Y);
    json_object_dotset_number(root_object, "axes.axis_2.position", machine_vector->position.Y);
    json_object_dotset_number(root_object, "axes.axis_2.target_position", machine_vector->position.Y);

    json_object_dotset_string(root_object, "axes.axis_3.name", axes[2]);
    json_object_dotset_string(root_object, "axes.axis_3.status", status[0]);
    json_object_dotset_number(root_object, "axes.axis_3.native_position", machine_vector->position.Z);
```

```
      json_object_dotset_number(root_object, "axes.axis_3.position", machine_vector->position.Z);
      json_object_dotset_number(root_object, "axes.axis_3.target_position", machine_vector->position.Z);


      json_object_dotset_string(root_object, "axes.axis_4.name", axes[3]);
      json_object_dotset_string(root_object, "axes.axis_4.status", "enabled");
      json_object_dotset_number(root_object, "axes.axis_4.native_position", machine_vector->position.U);
      json_object_dotset_number(root_object, "axes.axis_4.position", machine_vector->position.U);
      json_object_dotset_number(root_object, "axes.axis_4.target_position", machine_vector->position.U);

      json_object_dotset_string(root_object, "axes.axis_5.name", axes[4]);
      json_object_dotset_string(root_object, "axes.axis_5.status", status[0]);
      json_object_dotset_number(root_object, "axes.axis_5.native_position", machine_vector->position.V);
      json_object_dotset_number(root_object, "axes.axis_5.position", machine_vector->position.V);
      json_object_dotset_number(root_object, "axes.axis_5.target_position", machine_vector->position.V);

      /*json_object_dotset_string(root_object, "axes.axis_6.name", axes[5]);
      json_object_dotset_string(root_object, "axes.axis_6.status", status[0]);
      json_object_dotset_number(root_object, "axes.axis_6.native_position", machine_vector->position.W);
      json_object_dotset_number(root_object, "axes.axis_6.position", machine_vector->position.W);
      json_object_dotset_number(root_object, "axes.axis_6.target_position", 20.0);*/

      json_object_dotset_number(root_object, "tool_radius", 5.0);
      json_object_set_string(root_object, "coolant", coolant[1]);
      json_object_set_string(root_object, "motion_mode", motion_mode[1]);
      json_object_set_string(root_object, "units", units[1]);
      json_object_set_string(root_object, "distance_mode", distance_mode[0]);

      /* serialize string*/
      serialized_string = json_serialize_to_string_pretty(root_value);

      /* free memory */
      json_value_free(root_value);

      return serialized_string;
    }




    namespace PMAC_Adapter {

      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Windows::Forms;
```

```
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace PCOMMSERVERLib;


        /// <summary>
        /// Summary for MyForm
        /// </summary>
        public ref class MyForm : public System::Windows::Forms::Form
        {
        public: PmacDeviceClass pmacdoc;
                int dwDevice, status;
                bool bSuccess, bDeviceOpen;
                int port = 7171;
                Winsocket *mySocket = new Winsocket(port);


        private: System::Windows::Forms::Button^  tbDisconnect;
        public:
        private: System::Windows::Forms::Label^  label1;
        private: System::Windows::Forms::Button^  btGetResponse;
        private: System::Windows::Forms::TextBox^  tbMotor1Pos;
        private: System::Windows::Forms::Label^  label2;
        private: System::Windows::Forms::Button^  btTimerUpdate;
        private: System::Windows::Forms::Timer^  timerStreamData;



        private: System::Windows::Forms::Label^  label4;
        private: System::Windows::Forms::TextBox^  tbPeriod;
        private: System::Windows::Forms::Label^  label3;
        private: System::Windows::Forms::TextBox^  tbSocketConnection;
        private: System::Windows::Forms::PictureBox^  pictureBox1;
        private: System::Windows::Forms::Label^  label5;






        private: System::Windows::Forms::TextBox^  tbConnected;
        public:


        public:
            MyForm(void)
            {
                InitializeComponent();
                //
                //TODO: Add the constructor code here
                //
            }

        protected:
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            ~MyForm()
            {
                if (components)
                {
                    delete components;
                }
            }
        private: System::Windows::Forms::Button^  btConnectDevice;
        private: System::ComponentModel::IContainer^  components;
        protected:

        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>


    #pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                this->components = (gcnew System::ComponentModel::Container());
                System::ComponentModel::ComponentResourceManager^  resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));
                this->btConnectDevice = (gcnew System::Windows::Forms::Button());
                this->tbConnected = (gcnew System::Windows::Forms::TextBox());
                this->tbDisconnect = (gcnew System::Windows::Forms::Button());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->btGetResponse = (gcnew System::Windows::Forms::Button());
                this->tbMotor1Pos = (gcnew System::Windows::Forms::TextBox());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->btTimerUpdate = (gcnew System::Windows::Forms::Button());
                this->timerStreamData = (gcnew System::Windows::Forms::Timer(this->components));
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->tbPeriod = (gcnew System::Windows::Forms::TextBox());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->tbSocketConnection = (gcnew System::Windows::Forms::TextBox());
                this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
                this->label5 = (gcnew System::Windows::Forms::Label());
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();
                this->SuspendLayout();
                //
                // btConnectDevice
                //
                this->btConnectDevice->Location = System::Drawing::Point(34, 31);
                this->btConnectDevice->Name = L"btConnectDevice";
                this->btConnectDevice->Size = System::Drawing::Size(85, 48);
                this->btConnectDevice->TabIndex = 0;
                this->btConnectDevice->Text = L"Connect PMAC Device";
                this->btConnectDevice->UseVisualStyleBackColor = true;
                this->btConnectDevice->Click += gcnew System::EventHandler(this, &MyForm::btConnectDevice_Click);
                //
                // tbConnected
                //
                this->tbConnected->Location = System::Drawing::Point(305, 50);
                this->tbConnected->Name = L"tbConnected";
                this->tbConnected->RightToLeft = System::Windows::Forms::RightToLeft::No;
                this->tbConnected->Size = System::Drawing::Size(120, 20);
                this->tbConnected->TabIndex = 1;
                this->tbConnected->Text = L"Disconnected";
                //
                // tbDisconnect
                //
                this->tbDisconnect->Location = System::Drawing::Point(168, 31);
                this->tbDisconnect->Name = L"tbDisconnect";
                this->tbDisconnect->Size = System::Drawing::Size(85, 48);
                this->tbDisconnect->TabIndex = 2;
                this->tbDisconnect->Text = L"Disconnect PMAC Device";
                this->tbDisconnect->UseVisualStyleBackColor = true;
                this->tbDisconnect->Click += gcnew System::EventHandler(this, &MyForm::tbDisconnect_Click);
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(305, 31);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(105, 13);
                this->label1->TabIndex = 3;
                this->label1->Text = L"Machine Connection";
                //
                // btGetResponse
                //
                this->btGetResponse->Location = System::Drawing::Point(168, 221);
                this->btGetResponse->Name = L"btGetResponse";
                this->btGetResponse->Size = System::Drawing::Size(85, 51);
                this->btGetResponse->TabIndex = 4;
                this->btGetResponse->Text = L"Motor 1 Position";
                this->btGetResponse->UseVisualStyleBackColor = true;
                this->btGetResponse->Click += gcnew System::EventHandler(this, &MyForm::btGetResponse_Click);
                //
                // tbMotor1Pos
                //
                this->tbMotor1Pos->Location = System::Drawing::Point(305, 237);
                this->tbMotor1Pos->Name = L"tbMotor1Pos";
                this->tbMotor1Pos->RightToLeft = System::Windows::Forms::RightToLeft::No;
                this->tbMotor1Pos->Size = System::Drawing::Size(120, 20);
                this->tbMotor1Pos->TabIndex = 5;
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(305, 211);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(64, 13);
                this->label2->TabIndex = 6;
                this->label2->Text = L"Motor1_Pos";
                //
                // btTimerUpdate
                //
                this->btTimerUpdate->Location = System::Drawing::Point(168, 131);
                this->btTimerUpdate->Name = L"btTimerUpdate";
                this->btTimerUpdate->Size = System::Drawing::Size(85, 51);
                this->btTimerUpdate->TabIndex = 7;
                this->btTimerUpdate->Text = L"Adapter On/Off";
                this->btTimerUpdate->UseVisualStyleBackColor = true;
                this->btTimerUpdate->Click += gcnew System::EventHandler(this, &MyForm::btTimerUpdate_Click);
                //
                // timerStreamData
                //
                this->timerStreamData->Interval = 1000;
                this->timerStreamData->Tick += gcnew System::EventHandler(this, &MyForm::timerStreamData_Tick);
                //
                // label4
                //
                this->label4->AutoSize = true;
                this->label4->Location = System::Drawing::Point(31, 131);
                this->label4->Name = L"label4";
                this->label4->Size = System::Drawing::Size(91, 13);
                this->label4->TabIndex = 10;
                this->label4->Text = L"Signal Period (ms)";
                //
                // tbPeriod
                //
                this->tbPeriod->Location = System::Drawing::Point(34, 150);
                this->tbPeriod->Name = L"tbPeriod";
                this->tbPeriod->Size = System::Drawing::Size(100, 20);
                this->tbPeriod->TabIndex = 11;
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Location = System::Drawing::Point(305, 131);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(98, 13);
                this->label3->TabIndex = 13;
                this->label3->Text = L"Socket Connection";
                //
                // tbSocketConnection
                //
                this->tbSocketConnection->Location = System::Drawing::Point(305, 150);
                this->tbSocketConnection->Name = L"tbSocketConnection";
                this->tbSocketConnection->RightToLeft = System::Windows::Forms::RightToLeft::No;
                this->tbSocketConnection->Size = System::Drawing::Size(120, 20);
                this->tbSocketConnection->TabIndex = 12;
                this->tbSocketConnection->Text = L"Disconnected";
```

```
          //
          // pictureBox1
          //
          this->pictureBox1->Image = (cli::safe_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.Image")));
          this->pictureBox1->InitialImage = (cli::safe_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.InitialImage")));
          this->pictureBox1->Location = System::Drawing::Point(149, 328);
          this->pictureBox1->Name = L"pictureBox1";
          this->pictureBox1->Size = System::Drawing::Size(297, 151);
          this->pictureBox1->TabIndex = 14;
          this->pictureBox1->TabStop = false;
          //
          // label5
          //
          this->label5->ImageAlign = System::Drawing::ContentAlignment::TopLeft;
          this->label5->Location = System::Drawing::Point(12, 328);
          this->label5->Name = L"label5";
          this->label5->Size = System::Drawing::Size(110, 107);
          this->label5->TabIndex = 15;
          this->label5->Text = L"Note: this program works for Turbo PMAC with DPR feature; it is required to be ru"
            L"n in Administrtor Mode";
          //
          // MyForm
          //
          this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
          this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
          this->ClientSize = System::Drawing::Size(458, 491);
          this->Controls->Add(this->label5);
          this->Controls->Add(this->pictureBox1);
          this->Controls->Add(this->label3);
          this->Controls->Add(this->tbSocketConnection);
          this->Controls->Add(this->tbPeriod);
          this->Controls->Add(this->label4);
          this->Controls->Add(this->btTimerUpdate);
          this->Controls->Add(this->label2);
          this->Controls->Add(this->tbMotor1Pos);
          this->Controls->Add(this->btGetResponse);
          this->Controls->Add(this->label1);
          this->Controls->Add(this->tbDisconnect);
          this->Controls->Add(this->tbConnected);
          this->Controls->Add(this->btConnectDevice);
          this->Name = L"MyForm";
          this->Text = L"PMAC_Adapter";
          this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &MyForm::MyForm_FormClosing);
          (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();
          this->ResumeLayout(false);
          this->PerformLayout();

        }
    #pragma endregion
      private: System::Void btConnectDevice_Click(System::Object^  sender, System::EventArgs^  e) {
```

```
                if (bDeviceOpen == false) {
                    pmacdoc.SelectDevice((int)this->Handle, dwDevice, bSuccess);
                    if (bSuccess) {
                        pmacdoc.Open(dwDevice, bDeviceOpen);
                        tbConnected->Text = "Connected";
                    }
                    else {
                        tbConnected->Text = "Connection Failed";

                    }
                }
                else {
                    MessageBox::Show("Device already connected");
                }


            }// this is the end of connection operation


            private: System::Void tbDisconnect_Click(System::Object^  sender, System::EventArgs^  e) {
                if (bDeviceOpen) {
                    pmacdoc.Close(dwDevice);
                    bDeviceOpen = false;
                    tbConnected->Text = "Disconnected";
                }

            }
        private: System::Void btGetResponse_Click(System::Object^  sender, System::EventArgs^  e) {
            double val;
            if (bDeviceOpen == false) {
                MessageBox::Show("Please Connect the PMAC first");

            }
            else
            {
                pmacdoc.GetCommandedPos(dwDevice, 0, 1.0, val);
                char *data_in;

                machine_vector->position.X = -val / 16000;
                tbMotor1Pos->Text = machine_vector->position.X.ToString("0.0");
                //tbMotor1Pos->Text = val.ToString("0.00");
            }

        }//end of the get single response function


        private: System::Void MyForm_FormClosing(System::Object^  sender, System::Windows::Forms::FormClosingEventArgs^  e) {
            if (bDeviceOpen) {
                pmacdoc.Close(dwDevice);
                bDeviceOpen = false;
                tbConnected->Text = "Disconnected";
            }

        }// end of the closing function


        private: System::Void btTimerUpdate_Click(System::Object^  sender, System::EventArgs^  e) {
            timerStreamData->Enabled = !timerStreamData->Enabled;
            if (timerStreamData->Enabled) {
                btTimerUpdate->Text = "Adapter On, Click to Turn Off";
            }
            else
            {
                btTimerUpdate->Text = "Adapter Off, Click to Turn On";
            }

        }//end of stream data button


        private: System::Void timerStreamData_Tick(System::Object^  sender, System::EventArgs^  e) {

            if (tbPeriod->Text == String::Empty) {
                timerStreamData->Interval = 1000;
            }
            else {
                int Interval_temp;
                Interval_temp = Convert::ToInt64(tbPeriod->Text);
                if (Interval_temp < 50) {
                    Interval_temp = 50;
                }
                timerStreamData->Interval = Interval_temp;
            }



            if (mySocket->isConnected())
            {

                char *data_in;
                // read incomming messgages from the node
                if (mySocket->isAvailable() != 0)
                {

                    tbSocketConnection->Text = "Connected";
                    //Sleep(10);
                    data_in = mySocket->read();

                    double val;
                    if (bDeviceOpen == false) {

                    }
                    else
                    {
                        pmacdoc.GetCommandedPos(dwDevice, 0, 1.0, val);
                        tbMotor1Pos->Text = val.ToString("0.00");
                    }
                    //pmacdoc.GetNetActualPosition(dwDevice,0,1.0, val);
                    //lBResp->Items->Add(val);
                    machine_vector->position.X = - val / 16000 + 1.0;
                    tbMotor1Pos->Text = machine_vector->position.X.ToString("0.00");
                    //if (strcmp(data_in, "get machine_vector") == 0)
                    //{
                    // the JSON serialized machine vector
                    char* buffer = serialized_machine_vector();
                    // print to the console
                    std::cout << std::string(buffer) << "\n\n>";
                    // print to the socket
                    mySocket->write(std::string(buffer));
                    // free memory
                    json_free_serialized_string(buffer);
                    //}
                    //else {
                    // // send and ECHO message to the console and send it through the socket
                    //printf("%s\n>", data_in);
                    //mySocket->write(data_in);
                    //}

                }
                else {
                    // send a heartbeat to OSCM.
                    mySocket->write("ping");
                }
            }
            // 2. if the socket is not connected:
            if (!mySocket->isConnected())
            {
                // wait for a connection on the listening socket
                tbSocketConnection->Text = "Waiting to Connection";
                if (mySocket->connect())
                {
                    // print a message
                    printf("Adapter server has succesfully connected to OSCM client ");// play as a wait function
                }

            }






        }//end of timer function


        private: System::Void label3_Click(System::Object^  sender, System::EventArgs^  e) {
        }// do nothing in this function



        private: System::Void btAdapterConnect_Click(System::Object^  sender, System::EventArgs^  e) {




	}// do the adatper autherization in this click


	private: System::Void label5_Click(System::Object^  sender, System::EventArgs^ e) {
	}
	};
	}
```

# Appendix B: Code of Force Control Algorithm

In this Appendix, the detailed implementation of the force control and compliance compensation algorithm described in Chapter 2 will be listed, which includes the force control algorithm simulation (B-1) and force control PLCC program (B-2):

### *B-1 Force Control Simulink Simulation Diagram*

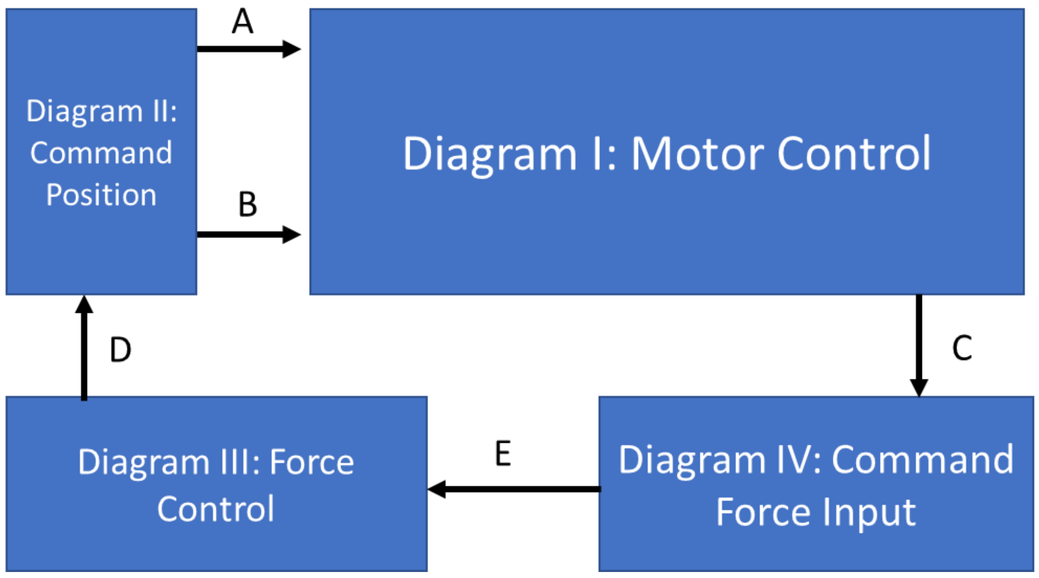
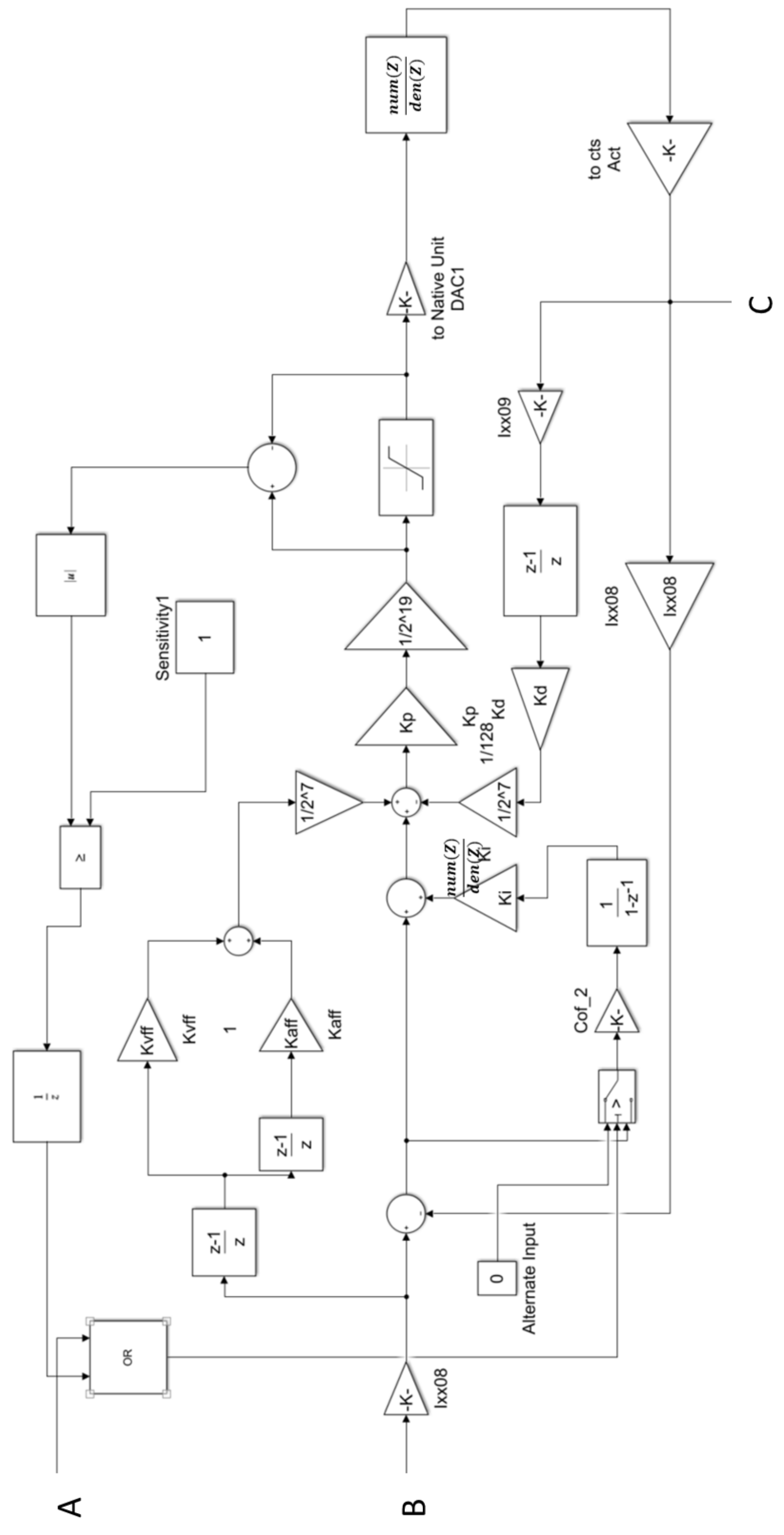
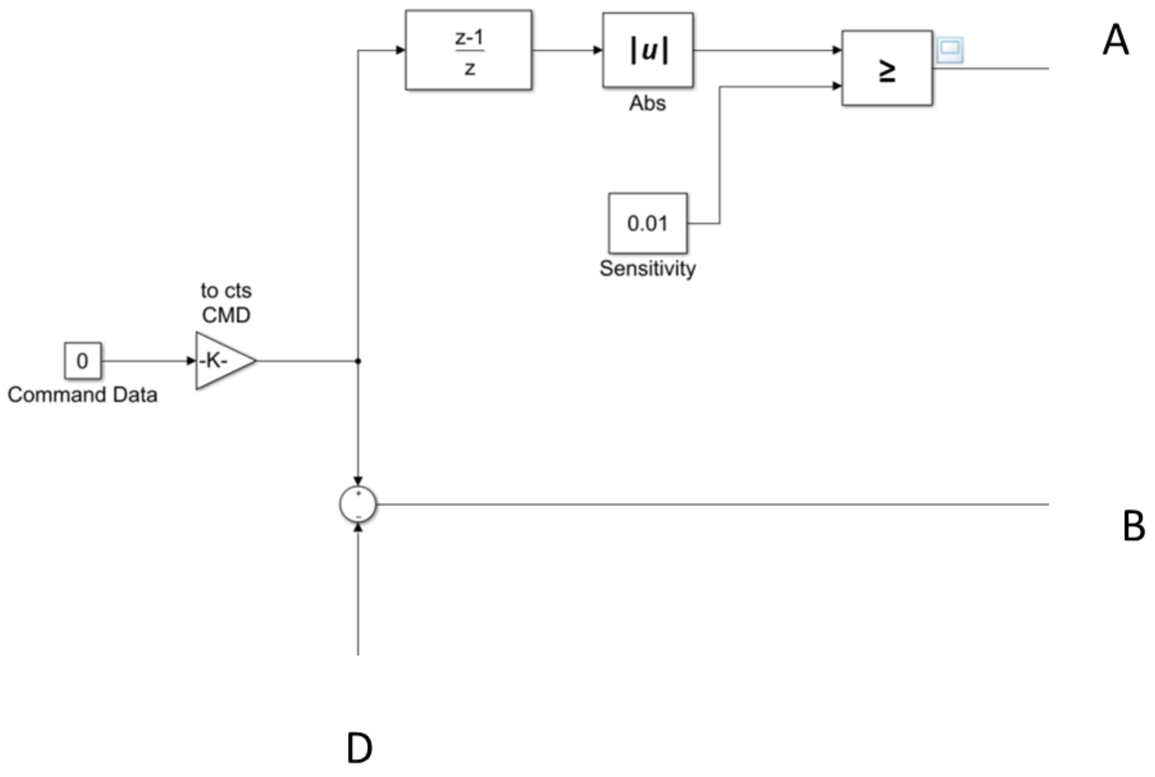


Diagram II: Command Position
A
B
Diagram I: Motor Control
D
C
Diagram III: Force Control
E
Diagram IV: Command Force Input

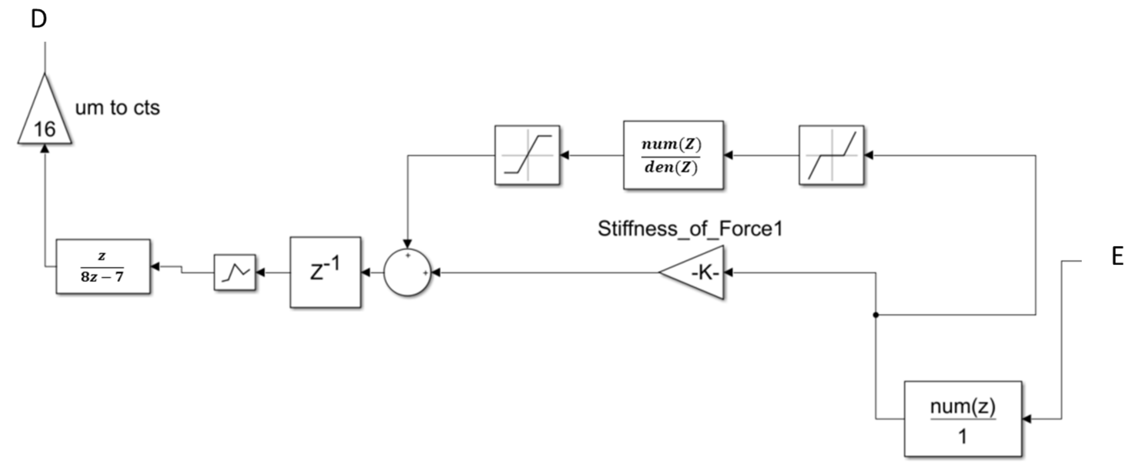
**Appendix Figure B-4 Overall simulation diagram structure (A, B, C, D, E designates the connection between sub-diagrams)**


A
B
C
-K-
Ixx08
OR
1/z
z-1/z
Kvff
Kaff
1
|u|
Sensitivity1
≥
1/2^7
Kp
1/128
Kd
Ki
num(Z)/den(Z)
1/2^19
1/(1-z-1)
Cof_2
0
Alternate Input
to Native Unit DAC1
to cts Act
Ixx09

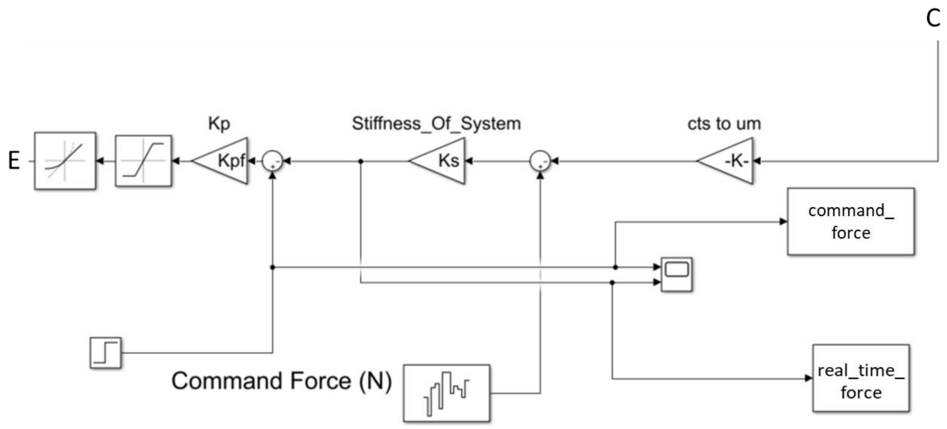
**Appendix Figure B-5 Simulation sub-diagram I**



**Appendix Figure B-6 Simulation sub-diagram II**



**Appendix Figure B-7 Simulation sub-diagram III**



**Appendix Figure B-8 Simulation sub-diagram IV**

## *B-2 Force Control PLCC Program*

In this section, a compiled PLC program for the force control algorithm is provided (PLCC Program 1). The program starts with memory allocation for the utilized variables and then different subprograms with their own functionalities, including force measurement, transformation, filtering and modification for the position loop. Moreover, a PLC program is also provided, which defines the PMAC configurations for force control algorithm and uploads the force/position data to the dual port ram for data collection purposes by the host computer.

***PLCC Program 1:***

```
CLOSE
DELETE GATHER
L1->X:$8001,0,24,s; servo period for force loop
L2->X:$8002,0,24,s; counter in the linear interpolation
L5->X:$8005,0,24,s; P gain value of force loop
L6->X:$8006,0,24,s; The I gain value of force loop
L7->X:$8007,0,24,s; counter for the interpolation
L8->X:$8008,0,24,s; running status variable for force loop
L9->X:$8009,0,24,s; cos value of the angle * 256
L10->X:$8010,0,24,s;sin value of the angle * 256
F11->L:$8011; distance in x direction
F12->L:$8012; distance in y direction
F13->L:$8013; overall distance
F14->L:$8014; cos value of the angle
F15->L:$8015; sin value of the angle
L23->X:$8023,0,24,s;maximum P gain value
L24->X:$8024,0,24,s; maximum P gain increment
L25->X:$8025,0,24,s; maximum I gain value
L27->X:$8027,0,24,s; dead band value for I gain
L101->X:$8101,0,24,s; voltage variable for x
L201->X:$8201,0,24,s; for y
L301->X:$8301,0,24,s; for z
L401->X:$8401,0,24,s; for u
L501->X:$8501,0,24,s; for v
L601->X:$8601,0,24,s; for w
L102->X:$8102,0,24,s; initial voltage for x
L202->X:$8202,0,24,s; for y
L302->X:$8302,0,24,s; for z
L402->X:$8402,0,24,s; for u
L502->X:$8502,0,24,s; for v
L602->X:$8602,0,24,s; for w
L103->X:$8103,0,24,s; force value for x
L203->X:$8203,0,24,s; for y
```

```
L303->X:$8303,0,24,s; for z
L403->X:$8403,0,24,s; for u
L503->X:$8503,0,24,s; for v
L603->X:$8603,0,24,s; for w
L404->X:$8403,4,20,s; 4 bits to 24 bits for u force value
L504->X:$8503,4,20,s; for v force
L604->X:$8603,4,20,s; for w force
L705->X:$8705,0,24,s; final force in normal direction
L706->X:$8706,0,24,s; command force in normal direction
L707->X:$8707,0,24,s; P gain compensation in normal direction
L708->X:$8707,8,16,s; P gain compensation divided by 256
L109->Y:$3401,12,12,s;M varible to read the raw force data directly from x
L209->Y:$3403,12,12,s;from y
L309->Y:$3405,12,12,s;from z
L409->Y:$3409,12,12,s;from u
L509->Y:$340B,12,12,s;from v
L609->Y:$340D,12,12,s;from w
L711->X:$8711,0,24,s; compensation in t step
L712->X:$8712,0,24,s; compensation in t-1 step
L713->X:$8713,0,24,s; compensation in t-2 step
L714->X:$8714,0,24,s; compensation in t-3 step
L721->X:$8721,0,24,s; force value after moving average
L722->X:$8721,8,16,s; force value divided by 256
L723->X:$8723,0,24,s; final I gain term
L724->X:$8723,8,16,s; final I gain term divided by 256
L425->X:$8425,0,24,s; final I gain term in u direction
L525->X:$8525,0,24,s; final I gain term in v direction
L431->X:$8431,0,24,s; PI inpterploation for current step (in u)
L432->X:$8432,0,24,s; PI inpterploation for previous step (in u)
L433->X:$8433,0,24,s; Temp value for incremental interploation (in u)
L434->X:$8431,4,20,s; 4 bits to 24 bits of L431 (in u)
L435->X:$8435,0,24,s; Final force interplotation value (in u)
L436->X:$8435,4,20,s; 4 bits to 24 bits of L435 (in u)
L531->X:$8531,0,24,s; PI inpterploation for current step (in v)
L532->X:$8532,0,24,s; PI inpterploation for previous step (in v)
L533->X:$8533,0,24,s; Temp value for incremental interploation (in v)
L534->X:$8531,4,20,s; 4 bits to 24 bits of L531 (in v)
L535->X:$8535,0,24,s; Final force interplotation value (in v)
L536->X:$8535,4,20,s; 4 bits to 24 bits of L535 (in v)
OPEN PLCC 0
CLEAR
IF (L8 < 1)
        L431 = L432 + L433 * L7; this is for the linear interploation of u axis
        L531 = L532 + L533 * L7; this is for the linear interploation of v axis
```

```
ENDIF
L435 = L434 * 1 + L436 * 15; this is for the exponential decay filter
L535 = L534 * 1 + L536 * 15; this is for the exponential decay filter
M669 = -1 * ITOF(L435) * 16; Here, the sign need to adjusted based on the axes vs motor
M769 = M669;
M869 = ITOF(L535) * 16;
M969 = M869;
L7 = L7 + 1;
CLOSE
CLOSE
DELETE GATHER
OPEN PLCC 6
CLEAR
L102 = L109; get the initial value for the raw force data
L202 = L209
L302 = L309
L402 = L409;
L502 = L509
L602 = L609
L706 = 0; the command force value with scale of 1536
L1 = 10; This is the period of voltage filter, unit of servo cycle
L5 = 0; This is the P gain value
I5211=I5111+5; to switch the phase of those two different timer
DISABLE PLCC 6
CLOSE
CLOSE
OPEN PLCC 7
CLEAR
I5211 = ITOF(L1); Here should be the timer interval you would like to assign
WHILE (I5211>0)
ENDWHILE
F11 = (- M661 + M161)/3072; the sign is changed to agree with axis
F12 = (M861 - M361)/3072
F13 = SQRT(F11 * F11 + F12 * F12)
IF (F13 < 1)
        F13 = 1; avoid 0, minimum 1 count
ENDIF
F14 = F11/F13
F15 = F12/F13
L9 = FTOI(F14*256)
L10 = FTOI(F15*256)
L101 = L109 - L102
L201 = L209 - L202
L301 = L309 - L302
```

```
L401 = L409 - L402
L501 = L509 - L502
L601 = L609 - L602
L103 = L101 *  110 + L201 * 108 ;this is force matrix
L203 = L101 * -91 +  L201 * 115; the force matrix is scaled and truncated by 96 (Ix08)
L303 = L301 * -191;
L403 = L401 *  -116 + L501 * -104;
L503 = L401 * -119 + L501 * 94;
L603 = L601 * 187;
L705 = L404 * L9 + L504 * L10; get the final force
L707 = (L706 - L705) * L5;
CLOSE
CLOSE
DELETE GATHER
OPEN PLCC 8
CLEAR
L23 = 128 * 6; i.e. 128 um maximum
L24 = 16 * 6; i.e. 16 um/ L1 servo cycle, i.e., 4 mm/s
L25 = 1024 * 3 * 1024; 1024 um maximum for I term
L27 = 6 * 3 * 1024; i.e. 6 um for minimal value to enable I gain, preventing zero shifting
I5212 = I5112 + 5; create phase difference between compensation and force control
L6 = 16; The Cof of I term in the PID controller, with a base of 0.88 * Kp
L711 = L708;
L712 = L711;
L713 = L711;
L714 = L711;
L721 = L711 * 512;
L723 = 0;
L425 = 0;
L525 = 0;
L431 = 0;
L432 = L431;
L433 = 0;
L435 = 0;
L531 = 0;
L532 = L531;
L533 = 0;
L535 = 0;
L7 = 1; Initialize for the counter in PLCC 0 & 9
L8 = 0; Initialize for status variable in PLCC 0 & 9
DISABLE PLCC 8
CLOSE
CLOSE
OPEN PLCC 9
```

```
CLEAR
I5212 = ITOF(L1)
WHILE (I5212>0)
ENDWHILE
L8 = 2;
L711 = L708; Take the force value
IF (L711 > L23); Safety margin for P term value
        L711 = L23
ENDIF
IF (L711 < -L23)
        L711 = -L23
ENDIF
IF (L711 > (L24 + L712)); Safety margin for P term velocity
        L711 = L24 + L712
ENDIF
IF (L711 < (-L24 + L712))
        L711 = -L24 + L712
ENDIF
L721 = L711 * 71 + L712 * 185 + L713 * 185 + L714 * 71; Moving Average for L721
L714 = L713; Push back
L713 = L712;
L712 = L711;
IF (L721 > L27); Dead band for P term value
        L721 = L721 - L27
        L723 = L723 + L722 * L6; Calculate the I term
ENDIF
IF (L721 < -L27)
        L721 = L721 + L27
        L723 = L723 + L722 * L6; Calculate the I term
ENDIF
IF (L723 > L25); Safety margin for I term value
        L723 = L25
ENDIF
IF (L723 < -L25)
        L723 = -L25
ENDIF
L425 = L724 * L9; Calculate the compensation in u direction
L432 = L431; Push back for the interplotation in u direction
L433 = (L425 - L432) / L1; Calculate the increment value
L525 = L724 * L10; Calculate the compensation in v direction
L532 = L531; Push back for the interplotation in v direction
L533 = (L525 - L532) / L1; Calculate the increment value
L7 = 1; Initialize the interplotation counter in the operation
L8 = 0;
```

```
CLOSE
OPEN PLCC 12
CLEAR
I5511 = 122; Here should be the timer interval you would like to assign
WHILE (I5511>0)
ENDWHILE
M4801 = -M161/32; Command Position
M4802 = M361/32
M4803 = M561/32;
M4804 = -M661/32;
M4805 = M861/32;
M4806 = -M1061/32;
M4811 = -M169; Compensation Position
M4812 = M369
M4813 = M569;
M4814 = -M669;
M4815 = M869;
M4816 = -M1069;
M4821 = M4103; Force Data
M4822 = M4203
M4823 = M4303;
M4824 = M4403;
M4825 = M4503;
M4826 = M4603;
M4831 = M4706; Command Normal Force Data
M4832 = M4705; Real Normal Force Data
M4833 = M4723; Compensation Value in normal direction
CLOSE

CLOSE
DELETE GATHER
M4001->X:$8001,0,24,s; servo period for force loop
M4002->X:$8002,0,24,s; counter in the linear interplotation
M4005->X:$8005,0,24,s; P gain value of force loop
M4006->X:$8006,0,24,s; The I gain value of force loop
M4007->X:$8007,0,24,s; counter for the interplotation
M4008->X:$8008,0,24,s; running status variable for force loop
M4009->X:$8009,0,24,s; cos value of the angle * 256
M4010->X:$8010,0,24,s;sin value of the angle * 256
M4011->L:$8011; distance in x direction
M4012->L:$8012; distance in y direction
M4013->L:$8013; overall distance
M4014->L:$8014; cos value of the angle
M4015->L:$8015; sin value of the angle
```

```
M4023->X:$8023,0,24,s;maximum P gain value
M4024->X:$8024,0,24,s; maximum P gain increment
M4025->X:$8025,0,24,s; maximum I gain value
M4027->X:$8027,0,24,s; dead band value for I gain
M4101->X:$8101,0,24,s; voltage variable for x
M4201->X:$8201,0,24,s; for y
M4301->X:$8301,0,24,s; for z
M4401->X:$8401,0,24,s; for u
M4501->X:$8501,0,24,s; for v
M4601->X:$8601,0,24,s; for w
M4102->X:$8102,0,24,s; initial voltage for x
M4202->X:$8202,0,24,s; for y
M4302->X:$8302,0,24,s; for z
M4402->X:$8402,0,24,s; for u
M4502->X:$8502,0,24,s; for v
M4602->X:$8602,0,24,s; for w
M4103->X:$8103,0,24,s; force value for x
M4203->X:$8203,0,24,s; for y
M4303->X:$8303,0,24,s; for z
M4403->X:$8403,0,24,s; for u
M4503->X:$8503,0,24,s; for v
M4603->X:$8603,0,24,s; for w
M4404->X:$8403,4,20,s; 4 bits to 24 bits for u force value
M4504->X:$8503,4,20,s; for v force
M4604->X:$8603,4,20,s; for w force
M4705->X:$8705,0,24,s; final force in normal direction
M4706->X:$8706,0,24,s; command force in normal direction
M4707->X:$8707,0,24,s; P gain compensation in normal direction
M4708->X:$8707,8,16,s; P gain compensation divided by 256
M4109->Y:$3401,12,12,s;M varible to read the raw force data directly from x
M4209->Y:$3403,12,12,s;from y
M4309->Y:$3405,12,12,s;from z
M4409->Y:$3409,12,12,s;from u
M4509->Y:$340B,12,12,s;from v
M4609->Y:$340D,12,12,s;from w
M4711->X:$8711,0,24,s; compensation in t step
M4712->X:$8712,0,24,s; compensation in t-1 step
M4713->X:$8713,0,24,s; compensation in t-2 step
M4714->X:$8714,0,24,s; compensation in t-3 step
M4721->X:$8721,0,24,s; force value after moving average
M4722->X:$8721,8,16,s; force value divided by 256
M4723->X:$8723,0,24,s; final I gain term
M4724->X:$8723,8,16,s; 8 bit to 24 bit of L723
M4425->X:$8425,0,24,s; final I gain term in u direction
```

```
M4525->X:$8525,0,24,s; final I gain term in v direction
M4431->X:$8431,0,24,s; PI inpterploation for current step (in u)
M4432->X:$8432,0,24,s; PI inpterploation for previous step (in u)
M4433->X:$8433,0,24,s; Temp value for incremental interploation (in u)
M4434->X:$8431,4,20,s; 4 bits to 24 bits of M4431 (in u)
M4435->X:$8435,0,24,s; Final force interplotation value (in u)
M4436->X:$8435,4,20,s; 4 bits to 24 bits of M4435 (in u)
M4531->X:$8531,0,24,s; PI inpterploation for current step (in v)
M4532->X:$8532,0,24,s; PI inpterploation for previous step (in v)
M4533->X:$8533,0,24,s; Temp value for incremental interploation (in v)
M4534->X:$8531,4,20,s; 4 bits to 24 bits of M4531 (in v)
M4535->X:$8535,0,24,s; Final force interplotation value (in v)
M4536->X:$8535,4,20,s; 4 bits to 24 bits of M4535 (in v)
M4801->DP:$60456; Command Position of x
M4802->DP:$60457; of y
M4803->DP:$60458; of z
M4804->DP:$60459; of u
M4805->DP:$6045A; of v
M4806->DP:$6045B; of w
M4811->DP:$6045C; Compensation Value of x
M4812->DP:$6045D; of y
M4813->DP:$6045E; of z
M4814->DP:$6045F; of u
M4815->DP:$60460; of v
M4816->DP:$60461; of w
M4821->DP:$60462; Force Data in x
M4822->DP:$60463; in y
M4823->DP:$60464; in z
M4824->DP:$60465; in u
M4825->DP:$60466; in v
M4826->DP:$60467; in w
M4831->DP:$60468; Command Normal Force Data
M4832->DP:$60469; Real normal force
M4833->DP:$6046A; Compensation Value in normal Direction
OPEN PLC 2
CLEAR
I8=0; Change the PLCC period to servo time
I51=0; Disable the original compensation table
I132=2250; Redefine the velocity feedforward for Motor 1
I232=2250; Redefine the velocity feedforward for Motor 2
I332=2350; Redefine the velocity feedforward for Motor 3
I432=2350; Redefine the velocity feedforward for Motor 4
I532=2100; Redefine the velocity feedforward for Motor 5
I632=2000; Redefine the velocity feedforward for Motor 6
```

```
I732=2000; Redefine the velocity feedforward for Motor 7
I832=1900; Redefine the velocity feedforward for Motor 8
I932=1900; Redefine the velocity feedforward for Motor 9
I1032=2100; Redefine the velocity feedforward for Motor 10
I135=0;  Disable the feedforward for the Motor 1
I235=0;  Disable the feedforward for the Motor 2
I335=0;  Disable the feedforward for the Motor 3
I435=0;  Disable the feedforward for the Motor 4
I535=0;  Disable the feedforward for the Motor 5
I635=0;  Disable the feedforward for the Motor 6
I735=0;  Disable the feedforward for the Motor 7
I835=0;  Disable the feedforward for the Motor 8
I935=0;  Disable the feedforward for the Motor 9
I1035=0; Disable the feedforward for the Motor 10
DISABLE PLC 2
CLOSE
OPEN PLC 3
DELETE GATHER
CLEAR
I5312 = 500
WHILE (I5312>0)
ENDWHILE
IF (M669 > 76800)
        M669 = M669 - 76800
        M769 = M769 - 76800
ENDIF; Here is for decreasing M669 and M769
IF (M669 < -76800)
        M669 = M669 + 76800
        M769 = M769 + 76800
ENDIF; Here is for increasing M669 and M769
IF (ABS(M669) < 76800)
        M669 = 0
        M769 = 0
        DISABLE PLC 3
ENDIF; Here is for decreasing M669 and M769 to zero and close them
CLOSE
```

# Appendix C: Manual for Force Control Algorithm

In this Appendix, the detailed manual for the force control and compliance compensation algorithm described in Chapter 2 is given. It includes the reference manual for the PLCC program (C-1) and operation manual for the force control program (C-2):

***C-1 Reference Manual for the PLCC program***

General Version Notes:
-----------------------------------------------------------------------------------------------
In this version (v2.0), the PLC program is transformed into PLCC
The PLCC program utilize the L variable with short fix operation
To accelerate any possible operation
The all possible parameter is avoid, i.e. they are meant to be utilized directly
L variable cannot be checked directly, they are checked through M variable with same location finding
-----------------------------------------------------------------------------------------------
This version (v3.0) is for the force control of the bottom tool,
The force is filtered by the PLCC of the previous compensation
The force is first taken as voltage, and then being filtered
The filtered result will be used to calculate the input signal combined with the command force
The input signal will be limited in both the displacement and velocity
Essentially, it limits the proportional gain term
Then, the input command is accumulated to bring the integer gain term
The integer gain term will also be limited in both displacement and velocity
The two terms are added, interpolated, and then, filtered
-----------------------------------------------------------------------------------------------
In this version (v3.7), the data reporting is packaged in PLCC 11 and 12 with previous version
The M variable is defined, and the corresponding M variable is defined accordingly in PLC 2
-----------------------------------------------------------------------------------------------
In this version (v3.8), the P gain term is omitted with a much larger I gain term
The I gain term is equal to the velocity control, the control need to be carefully adjusted
-----------------------------------------------------------------------------------------------
In this version (v4.0), the controlled axis is transformed into the bottom tool, so some of the transformation is needed and the calibration/verification need to be done;
The force is only controlled in the normal direction without considering the others;
Additional PLCC added to calculate the normal direction at the real-time.
-----------------------------------------------------------------------------------------------
Note that, the compiled PLCC utilize directly the memory location for P variable to calculate,

In Turbo PMAC, we could see P1 -> L$6001 and also P2 -> L$6002

Since for consistency, we will utilize P20xx, so that the start address will be L$8000
And then, to keep consistency of the program terminology, we going to use just the X memory
i.e. P2s21 -> Ls21 -> X:$8s21,0,24,s
Note that all the plcc are supposed to be equipped with s indicating the sign.

For the checking purpose, all the L variable will also be represent by a M variable separated if possible
The M variable will be assigned through a separate PLC program
Then, the variable itself can be checked through direct acquiring the M variable
To simplified the variable, the terminology will be determined as such
P2s21 -> Ls21 -> X:$8s21,0,24,s->M4s21

s denotes motor, t denotes glbbal axes, r denotes the normal axes

Also note since the U,V,W denote different algorithm with X,Y,Z,
The system has different terminology as t instead of s to diffiate the value

Theoretically, we can directly check the corresponding P variables, to know whether a L variable is updated correctly.

Also, in this version, more axes are added since we have the normal direction, the tagential direction, (probaly another tangential axis)
The normal direction will be called r, r value will be 7,8,9 depending on normal, tangential or other direction

```
----------------------------------------------------------------
-----                                      -------
-----                                      -------
-----        P2s21 -> Ls21 -> X:$8s21,0,24,s->M4s21       -------
-----   or    P2t21 -> Lt21 -> X:$8t21,0,24,s->M4t21      -------
-----   or    P2r21 -> Lr21 -> X:$8r21,0,24,s->M4r21      -------
-----                                      -------
-----                                      -------
-----                                      -------
----------------------------------------------------------------
```

&&PLC Definition:

PLCC 0 - Final Linear Interpolation and Exponential Decay Filter,
- It is defined together with compensation
PLCC 6 - Force Initial Value Restore
PLCC 7 - Matrix Evaluation, Transfer the force to input signal of proportional part
- The force will be transformed into the normal direction
PLCC 8 - Force Moving Average Cof Value Restore and Matrix Value Initialization

- It will also initialize the value used in the linear interpolation

PLCC 9 - Moving Average Filter for the proportional gain,
The integration portion also need to be added,
The integration portion will also be limited
PLCC12 - Status monitoring PLCC to copy the value and assign them to DPR

PLC 2 - M variables (for monitoring)& I variables (for setting up the process, accleration feedforward term plus the I8 term is initialized, need to remember to turn off the composition)
PLC 3 - Move the Motor 6 and 7 gradually to return to non-compensated position
PLC 4 - Move the Motor 8 and 9 gradually to return to non-compensated position
---------------------------------------------------------------
&&Operation Procedure

when start the application

1.Check Fu channel First!

This is Channel 20 for LABVIEW
and M5073 in Delta

2. Then set I5 = 0, so that that no PLCC can be run

3. Upload the PLCC documents, and then DISABLE all the PLCC 0..9

4. Set I5 back to 2, enable PLCC 6 and 7,

5. Set I8 = 0, and disable all the acceleration gain and setup the desired M variable definition
- Or, just run the downloaded the PLC 2 now you can check the raw force/compensation data
- You need to run PLC 2 to disable all the accel feedforward!!!!!!!!!!!!!!!!!!!

6. Enable PLCC 8 and 9, we can check the after-filtered data

7. Enable the designated motors, so allow the motor to respond correctly

8. Set I5 = 3, enable PLCC 0, we can finally check the Compensation value

9. The default P gain is set to be 0, to enable control, set the desired force value and gradually enable the M4005
--------------------------------------------------------------------

To stop the application,

disable PLCC 0 first,

Turn I5 = 0

IF M169 - M569 not equal to 0, set up back to zero

then, disable PLC 6..9,

--------------------------------------------------------------
&& Timer Definition

I5211 is the timer used in PLCC 7
to control the voltage-to-force conversion
its period is controlled by P2001 in PLCC 2
to accelerate the process, P2001 has been removed, and substitute with the direct number

I5212 is the timer used in PLCC 9
to control the moving average filter
its period is controlled by P2001 in PLCC 2
to accelerate the process, P2001 has been removed, and substitute with the direct number

----------------------------------------------------------------
&& Global Variable Definition
P2001,L1: the servo period for the voltage-to-force/ moving average operation in PLCC 7/PLCC 9
P2002,L2: the counter for interpolation of the compensation value, initialized in PLCC 4, used in PLCC 0 & 5
P2003: empty, not reserved
P2004 L4: empty, not reserved

P2005 L5: P gain of the axes in U,V,W, with scale 1 um/N, could be further adjusted, default value 2,defined in PLCC 6
P2006 L6: The initial value of I gain vs P gain, with scale of 1 * 226/256 (0.883), default value of 8, (7.0 * Kp), defined in PLCC 8.
The I gain will implemented dynamically as L6 or 0 depending on the P gain value with minimal or maximal band
// To differentiate U,V and W, in the final convert scheme, it will be convert into M reading with different scale
P2007,L7: the counter for interpolation of the compensation value, initialized in PLCC 8, used in PLCC 0 & 9
P2008,L8: the status variable of the PLCC, initialized in PLCC 8, used in PLCC 0 & 9

P2009, L9: the cos value of the current contact angle, with a base of 256
P2010, L10: the sin value of the current contact angle, with a base of 256

P2011, F11: the distance from top to bottom tool in x direction (i.e. (M661 - M161)/3072 )
P2012, F12: the distance from top to bottom tool in y direction (i.e. (M861 - M361)/3072 )
P2013, F13: the true distance from top to bottom tool SQRT(F11 * F11 + F12 * F12)
P2014, F14: the cos value of the current contact angle
P2015, F15: the sin value of the current contact angle

P2021, L21: Maximum Compensation Value in 12 * um; (Defined in PLCC 4)
P2022, L22: Maximum Compensation Speed in 12 * um/ L7 servo cycle; (Defined in PLCC 4)

P2023, L23: Maximum Compensation Value in P gain term, with scale 6 * um, default value, 512 um;
: Note that L23 also help to limit the maximum increment of the I gain, limit the speed value from I gain
: The speed is L23 * L6*226/256, default value is 3.5 mm/s;
P2024, L24: Maximum Compensation Speed in P gain term, with scale 6 * um/ L7 * servo cycle, default value, 4 mm/s;
P2025, L25: Maximum Compensation Value in I gain term, with scale 3 * 1024 * um, default value 1024 um
P2026, L26: Empty, unused
P2027, L27: Minimum Compensation Value to enable I gain, with scale 3 * 1024 * um, default value, 3 um
// note: here, three axes are assumed to be equal to each other
// so only one value is defined here

-------------------------------------------------------------
&& Channel Specific Variable Definition
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
& The operation is defined with integer operation,
& The range of L variable is 2^23, which is 8e6
& The raw voltage reading is 2^11, which is 2e3
& So, the raw force reading is 2e5
& Later, we need to confine all the parameters in the 2^23 range
& The other safety is that P gain is confined into 128 um
& while that the I gain is confined into 1024 um
& For I gain definition, note that it is integrated at 226 Hz
& So I gain COF should have a 226 Hz integrated
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

P2t01, Lt01 - ADC channel specific raw data after force zero (Defined in PLCC 7)
P2t02, Lt02 - ADC channel specific initial force zero (Defined in PLCC 6)
P2t03, Lt03: raw force value for the volatage channel, with a scale 3 * 32 * N, maximum 2e5 (Defined in PLCC 7)

P2t04, Lt04: 4 bits to 24 bits of the P2t03, with a scale of 3 * 2 * N,
P2r05, Lr05: final force value in the normal direction, with a scale of 3 * 32 * 16 * N, maximum 3e6
P2r06, Lr06: command foce value in the normal direction, with a scale of 3 * 32 * 16 * N
P2r07, Lr07: net compensation value after P gain, can be expressed as (P2t06 - P2t05) * L5, L5 = 0,1 or 2, with a unit 3 * 32 * 16 um
P2r08, Lr08: the 8 bits to 24 bits of Lt07, with a unit 6 * um
P2r09, Lr09, temp value point to the M variable, will be the same with M5069, M5070, M5071 for x/y/z and M5073,M5074,M5075 for u/v/w

p2r11, Lr11 - Compensation value in t step, with scale 6 * um , maximum 3e4 um
(Initialize in PLCC4)
P2r12, Lr12 - Compensation value in t-1 step, with scale 6 * um , maximum 3e4 um
(Initialize in PLCC4)
P2r13, Lr13 - Compensation value in t-2 step, with scale 6 * um , maximum 3e4 um
(Initialize in PLCC4)
P2r14, Lr14 - Compensation value in t-3 step, with scale 6 * um , maximum 3e4 um
(Initialize in PLCC4)

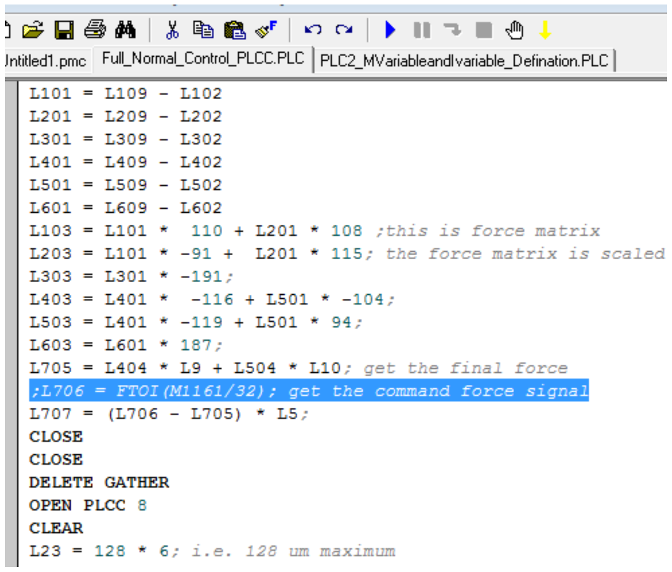
P2r21, Lr21 - Force value after moving average filter, with scale 3 * 1024 * um, maximum 1.6e6 um
P2r22, Lr22 - 8 bits to 24 bits of Lt21, with scale 3 * 4 * um, maximum, 6e3 um
P2r23, Lr23 - Accumulated Error, with scale of 3 * 1024 * um, maximum 3e6 um, Lt23 = Lt23 + Lt22 * L6
P2r24, Lr24 - 8 bits to 24 bit of Lt24, with scale 3 * 4 * um, maximum 636 um
P2t25, Lt25 - Accumulated Error in axis direction, with scale of 3 * 1024 * um, maximum 3e6 um

P2t31, Lt31 - Force interpolation value for current step, with scale 3 * 1024 * um, maximum 7e6
P2t32, Lt32 - Force interpolation value for previous step, with scale 3 * 1024 * um, maximum 7e6
P2t33, Lt33 - Temp value for the increment interpolation, with scale 3 * 1024 * um, maximum 7e6, Lt33 = (Lt23 - Lt32)/L1
P2t34, Lt34 - 4 bits to 24 bits of Ls31 (divided by 16), with scale 3 * 64 * um, maximum 1e6
P2t35, Lt35 - Final Force interpolation value for check, with scale 3 * 1024 * um, maximum 7e6
P2t36, Lt36 - 4 bits to 24 bits of Ls35 (divided by 16), with scale 3 * 64 * um, maximum 1e6
Ms69 - Final control compensation value for the motor, with scale of 96 * 32 * 16 um

### *C-2 Operation Manual for the PLCC program*

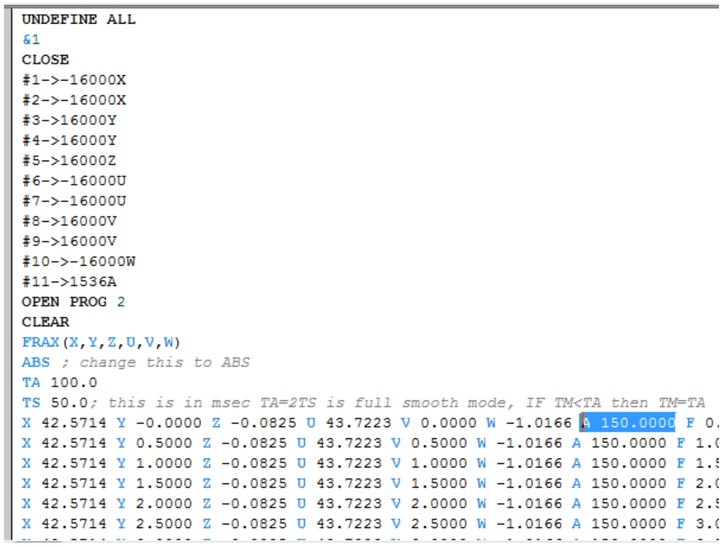
1) make sure I5 = 0;

2) download the force control PLC with different version, (C:\Users\AMPL_DSIF\Desktop\DSIF_Configuration\Force_Control\PLC_Version4 **\PLC_v4.5** for regular **Force Control**) or
3) (C:\Users\AMPL_DSIF\Desktop\DSIF_Configuration\Force_Control\PLC_Version4 **\PLC_v4.7** for **Force Maintainence**)
4) Including **Full_Normal_Control_PLCC.PLC, PLC2_XXX, PLC3_XXX, PLC4_XXX**
5) Please check in Full_Normal_Control_PLCC.PLC **Line 134 (** L706 = FTOI(M1161/32); get the command force signal **)** to see whether it is commented out**.** If it is commented out by the semicolon as shown in the Fig below, then the Desired Force need to be manually input;


Jntitled1.pmc
Full_Normal_Control_PLCC.PLC
PLC2_MVariableandIvariable_Defination.PLC

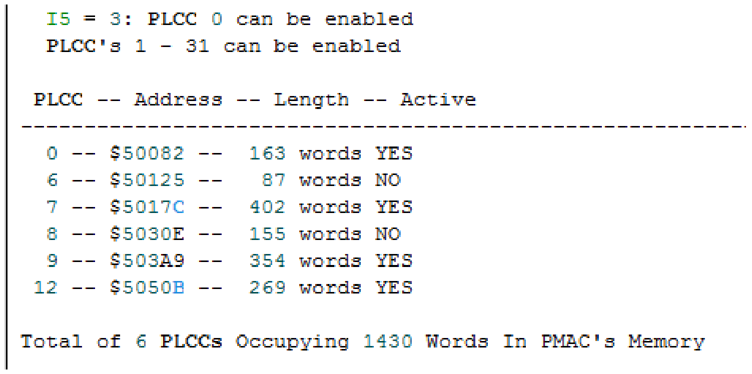
```
L101 = L109 - L102
L201 = L209 - L202
L301 = L309 - L302
L401 = L409 - L402
L501 = L509 - L502
L601 = L609 - L602
L103 = L101 *  110 + L201 * 108 ;this is force matrix
L203 = L101 * -91 +  L201 * 115; the force matrix is scaled
L303 = L301 * -191;
L403 = L401 *  -116 + L501 * -104;
L503 = L401 * -119 + L501 * 94;
L603 = L601 * 187;
L705 = L404 * L9 + L504 * L10; get the final force
;L706 = FTOI(M1161/32); get the command force signal
L707 = (L706 - L705) * L5;
CLOSE
CLOSE
DELETE GATHER
OPEN PLCC 8
CLEAR
L23 = 128 * 6; i.e. 128 um maximum
```

6) If it is not, the command force will depends on Axis A in the motion program, as shown in the Figure 2.



```
UNDEFINE ALL
&1
CLOSE
#1->-16000X
#2->-16000X
#3->16000Y
#4->16000Y
#5->16000Z
#6->-16000U
#7->-16000U
#8->16000V
#9->16000V
#10->-16000W
#11->1536A
OPEN PROG 2
CLEAR
FRAX(X,Y,Z,U,V,W)
ABS ; change this to ABS
TA 100.0
TS 50.0; this is in msec TA=2TS is full smooth mode, IF TM<TA then TM=TA
X 42.5714 Y -0.0000 Z -0.0825 U 43.7223 V 0.0000 W -1.0166 A 150.0000 F 0.
X 42.5714 Y 0.5000 Z -0.0825 U 43.7223 V 0.5000 W -1.0166 A 150.0000 F 1.
X 42.5714 Y 1.0000 Z -0.0825 U 43.7223 V 1.0000 W -1.0166 A 150.0000 F 1.
X 42.5714 Y 1.5000 Z -0.0825 U 43.7223 V 1.5000 W -1.0166 A 150.0000 F 2.
X 42.5714 Y 2.0000 Z -0.0825 U 43.7223 V 2.0000 W -1.0166 A 150.0000 F 2.
X 42.5714 Y 2.5000 Z -0.0825 U 43.7223 V 2.5000 W -1.0166 A 150.0000 F 3.
```

7) Put I5 = 2, enable plc 2,  then put I5 = 3, then, make sure you see these two screen:



```
 I5 = 3: PLCC 0 can be enabled
 PLCC's 1 - 31 can be enabled

PLCC -- Address -- Length -- Active
-----------------------------------------------------------
 0 -- $50082 --  163 words YES
 6 -- $50125 --   87 words NO
 7 -- $5017C --  402 words YES
 8 -- $5030E --  155 words NO
 9 -- $503A9 --  354 words YES
12 -- $5050B --  269 words YES

Total of 6 PLCCs Occupying 1430 Words In PMAC's Memory
```

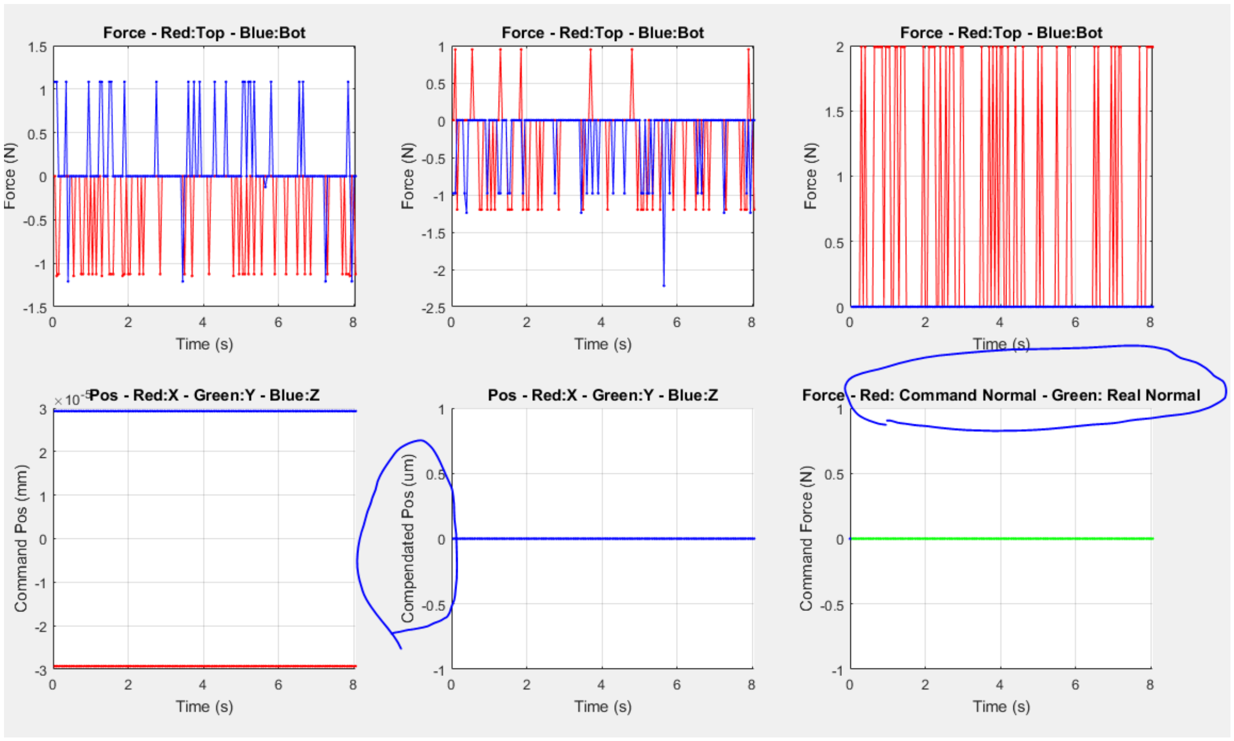
8) under the view, PLCC status

9) After check them, remember to close these view windows just to save the windows memory

10) Now, you can start the MATLAB script (Force_Position_Gathering.m), under the folder C:\Users\AMPL_DSIF\Desktop\DSIF_Configuration\Force_Control\Monitoring_v1.2_force_control

11) Then, check the load cell reading correctly if you shake the tools.

12) Also, the last two windows are important, one is how much the controller compensate for the the force control in the unit of um, the other is what is the current horizontal normal force and what is the command one.



13) Then, you can go ahead with the typical toolpath algorithm, enable the motors, home them and run the toolpath,

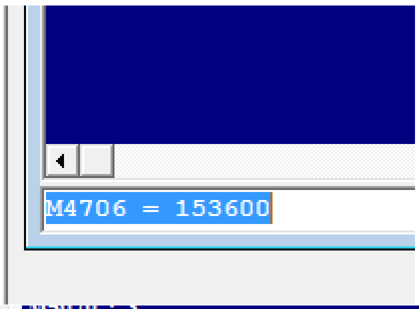
14) Right before you turn on the force control, if you already see some non-zero command force value form the window 6, it usually means the motion program already set the command force from the Axis A, which means the Line 134 is not commented in the Full_Force_Control.PLC file.

15) If it did get commented, and you still see non-zero force, make sure you follow all the previous instructions correctly. Stop the experiment, and turn I5 = 2, and lift up the tool.

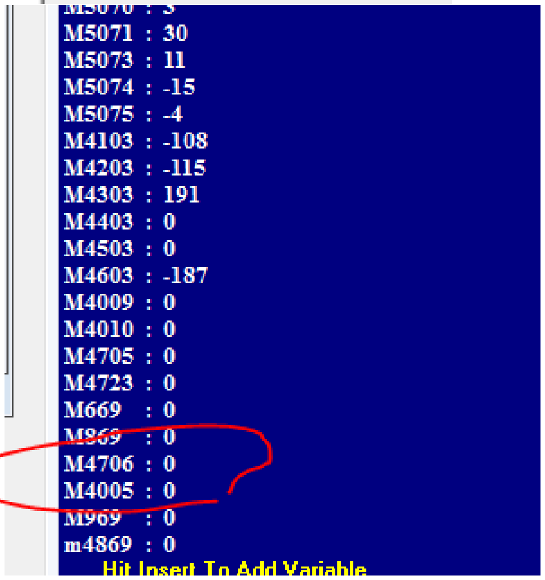
16) If you do input the A value, but still see a zero command force, check whether the line is uncommented correctly.

17) But keep in mind, once you start the physical experiment, you should not change the Full_Normal_Control.PLCC. Than means, if you make error, you should stop the experiment, and redo it.

18) If you want to put the command manually, make sure Line 134 is commented, and then put M4706 = Force_value * 96 * 16, for example, if you want force = 100, you will put, M4706 = 153600. Before you do that, make sure M4005 = 0 in the watch window



```
M4706 = 153600
```



```
M5070 : 3
M5071 : 30
M5073 : 11
M5074 : -15
M5075 : -4
M4103 : -108
M4203 : -115
M4303 : 191
M4403 : 0
M4503 : 0
M4603 : -187
M4009 : 0
M4010 : 0
M4705 : 0
M4723 : 0
M669  : 0
M869  : 0
M4706 : 0
M4005 : 0
M969  : 0
m4869 : 0
   Hit Insert To Add Variable
```

19) Turn on M4005 = 1, wait the system to response (see the real force change in window 6 in the MATLAB interface), and then if everything is fine, change M4005 = 2

20) To stop the force control in a soft and safe way (but might be slow), change I5 = 2. Remember to do this before lifting up the tool.

21) After the tool is back to 0,0,100 and 0,0,-100, remember to enable PLC 3 and enable PLC 4 in the command window. You should see three zero position value in the Windows 5th in MATLAB interface before you turn off the machine. If not, you need to realign the two tools once you restart the machine.

22) In case, emergency happens, or you are doing some fracture test, and you don’t have the time to put I5 = 2, press Ctrl+D will stop the force control, (Remember it will not stop the motion of the tool, you still need to press Ctrl+A) . Ctrl+D will stop all the PLC, including the force reading (interface one to the MATLAB). To reenable the force reading, turn I5 = 2 first, and then enable PLCC 7, and Enable PLCC 12.
23) *If turn of the Turbo PMAC, everything will be reset as the default. So if you mess up something, you can turn off the machine and start over.

# Appendix D: Code of On-line Springback Compensation Algorithm

In this appendix, the detail manual of the springback compensation algorithm described in Chapter 4 will be listed, which includes the reference manual for the simulation-based springback prediction (D-1), the control point selection algorithm (D-2) and on-line force gathering/springback modification algorithm (D-3).

## *D-1 Python Script for the Simulation-based Springback Prediction Method*

```python
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqusConstants import *
import time

import csv
import json
import os

Working_Directory = 'D:/HuaqingRen/CAE_Automated_Pyramid/Step3_Abaqus'

# for change the directory
os.chdir(Working_Directory)
# for debuging purpose
file = open('error_execption.txt','w')
# read the configuration file
configure_path = 'Configure_OSCA.json'

data = json.load(open(configure_path))

Inc_Depth = float(data['Inc_Depth'])

Prefix = data['Part_Prefix']

Sheet_Thickness = float(data['Sheet_Thickness'])
# read the CSV file
```

```


field_data = (())

csv_file_path = Prefix + '.csv'

with open(csv_file_path) as csvfile:
    readCSV = csv.reader(csvfile,delimiter = ',')
    for row in readCSV:
        row = [float(i) for i in row]
        field_data = field_data+(tuple(row),)

total_loop = field_data[-1][1]  # this is the total loop number
total_point_number = field_data[-1][0]  # this is the total point number

# change the directory so that the ODB is output into a single folder
os.chdir((Working_Directory + '/Output_ODB'))
previous_loop = 0;

for i in range(1,int(total_point_number+1)):
    for j in range (2,5):
        scale = 4 + 2 * j
        current_loop = field_data[i-1][1]
        # define the IGS file path
        igs_folder_path = (Working_Directory +'/IGS_Part/')
        part_number = "_N"+str(int(i))+"_S"
        part_number_L = "_L" + str(int(current_loop)) + "_S"
        part_number2 = str(int(scale))
        saved_suffix = ".igs"
        read_file_path = igs_folder_path + Prefix + part_number_L + part_number2 + save
d_suffix
        read_file_path = str(read_file_path)
        # visualization checking option
        bool_check = False
        # open igs file
        mdb.openIges(
            read_file_path, msbo=False, scaleFromFile=OFF, trimCurve=DEFAULT)
        My_Part = mdb.models['Model-
1'].PartFromGeometryFile(combine=False, convertToAnalytical=1
            , dimensionality=THREE_D, geometryFile=mdb.acis, name='Cone_2_H11_S9',
            stitchEdges=1, stitchTolerance=1.0, type=DEFORMABLE_BODY)
        My_Model = mdb.models['Model-1']
        # define the thickness field path
        TF_file_folder=(Working_Directory +'/Thickness_Field/')
        TF_file_middle =Prefix + '_TF' + str(int(current_loop))+'.csv'
        TF_file_path = TF_file_folder + TF_file_middle
        thickness_field_data = (())
        with open(TF_file_path) as csvfile:
            readCSV = csv.reader(csvfile,delimiter = ',')
            for row in readCSV:
                row = [float(m) for m in row]
                thickness_field_data = thickness_field_data+(tuple(row),)
        My_Model.MappedField(name='AnalyticalField-1', description='',
            regionType=POINT, partLevelData=False, localCsys=None, pointDataFormat=XYZ,

            fieldDataType=SCALAR,
            xyzPointData=thickness_field_data)
        # set mesh control and generate mesh
        My_Part.setMeshControls(elemShape=TRI,
            regions=mdb.models['Model-1'].parts['Cone_2_H11_S9'].faces)
        My_Part.seedPart(deviationFactor=0.1,
            minSizeFactor=0.1, size=2.0)
        My_Part.generateMesh()
        # set the material property and shell section via the defined analytical field
               My_Model.Material(name='Aluminum')
               My_Model.materials['Aluminum'].Elastic(table=((68000.0, 0.3), ))
               My_Model.HomogeneousShellSection(idealization=NO_IDEALIZATION,
                   integrationRule=SIMPSON, material='Aluminum', name='Section-1', numIntPts=7
                   , poissonDefinition=DEFAULT, preIntegrate=OFF, temperature=GRADIENT,
                   thickness=Sheet_Thickness, thicknessField='AnalyticalField-1', thicknessModulus=None, thicknessType=
                   ANALYTICAL_FIELD, useDensity=OFF)
               faceset = My_Part.Set(faces= My_Part.faces,name='Set_Surface')
               My_Part.SectionAssignment(region = faceset, sectionName  = 'Section-1' )
               # generate the instance
               My_Model.rootAssembly.DatumCsysByDefault(CARTESIAN)
               My_Instance = My_Model.rootAssembly.Instance(dependent=ON, name=
                   'Cone_2_H11_S9-1', part=My_Part)
               # create the step
               mdb.models['Model-1'].StaticStep(initialInc=0.1, maxInc=0.1, name='Step-1',
                   nlgeom=ON, previous='Initial')
               My_Assembly = My_Model.rootAssembly
               # define the BC in the boundaries
               # clamp but movable edge in the right side, U3 = 0, UR2 = 0
               rightedge = My_Assembly.Set(edges=My_Instance.edges.getByBoundingBox(124.0, -126.0, -1.0, 126.0, 126.0, 1.0)
                   , name='Right_Edge')
               My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
                   distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
                   'Right_Edge_BC', region=rightedge, u1=UNSET,
                   u2=UNSET, u3=0.0, ur1=UNSET, ur2=0.0, ur3=UNSET)
               # clamp but movable edge in the left side, U3 = 0, UR2 = 0
               leftedge = My_Assembly.Set(edges=My_Instance.edges.getByBoundingBox(-126.0, -126.0, -1.0, -124.0, 126.0, 1.0)
                   , name='Left_Edge')
               My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
                   distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
                   'Left_Edge_BC', region=leftedge, u1=UNSET,
                   u2=UNSET, u3=0.0, ur1=UNSET, ur2=0.0, ur3=UNSET)
               # clamp but movable edge in the top side, U3 = 0, UR1 = 0
               topedge = My_Assembly.Set(edges=My_Instance.edges.getByBoundingBox(-126.0, 124.0, -1.0, 126.0, 126.0, 1.0)
                   , name='Top_Edge')
               My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
                   distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
                   'Top_Edge_BC', region=topedge, u1=UNSET,
          u2=UNSET, u3=0.0, ur1=0.0, ur2=UNSET, ur3=UNSET)
        # clamp but movable edge in the bottom side, U3 = 0, UR1 = 0
        botedge = My_Assembly.Set(edges=My_Instance.edges.getByBoundingBox(-126.0, -126.0, -1.0, 126.0, -124.0, 1.0)
          , name='Bottom_Edge')
        My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
          distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
          'Bottom_Edge_BC', region=botedge, u1=UNSET,
          u2=UNSET, u3=0.0, ur1=0.0, ur2=UNSET, ur3=UNSET)
        # find the top right vertice to fix U1 and U2
        toprightpoint = My_Assembly.Set(name='Top_Right', vertices=
          My_Instance.vertices.getByBoundingBox(124.0, 124.0, -1.0, 126.0, 126.0, 1.0))
        My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
          distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
          'TopRight_Point_BC', region=toprightpoint, u1=0.0,
          u2=0.0, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
        # find the top left vertice to fix U1
        topleftpoint = My_Assembly.Set(name='Top_Left', vertices=
          My_Instance.vertices.getByBoundingBox(-126.0, 124.0, -1.0, -124.0, 126.0, 1.0))
        My_Model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
          distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
          'TopLeft_Point_BC', region=topleftpoint, u1=0.0,
          u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
        # create the global nodes set to apply the load condition
        mesh_nodes=My_Instance.nodes
        if len(mesh_nodes) == 0:
          file.write('No load nodes generated, please check the mesh or the igs part!!!     ')
        # create the load to the specific region
        # the above line is to throw out the error for the error mesh
        deltax, deltay, deltaz = 1.5, 1.5, 1000.0
        x, y, z = field_data[i-1][2], field_data[i-1][3], 0
        xmin, ymin, zmin = x-deltax, y-deltay, z-deltaz
        xmax, ymax, zmax = x+deltax, y+deltay, z+deltaz
        LoadNodes = mesh_nodes.getByBoundingBox(xmin, ymin, zmin, xmax, ymax, zmax)
        LodeNodes_Set = mdb.models['Model-1'].rootAssembly.Set(name='load points', nodes=LoadNodes)
        loadnodes_number = len(LoadNodes)
        if len(LoadNodes) == 0:
          file.write('No Load Region is selected')
        # apply the load
        mdb.models['Model-1'].ConcentratedForce(cf3=(-600.0/loadnodes_number), createStepName='Step-1',
          distributionType=UNIFORM, field='', localCsys=None, name='Point_Load', region= LodeNodes_Set)
        # output request
        My_Model.fieldOutputRequests['F-Output-1'].setValues(frequency=
          LAST_INCREMENT, variables=('S', 'PE', 'PEEQ', 'PEMAG', 'LE', 'U', 'RF',
          'CF', 'CSTRESS', 'CDISP', 'STH'))
        My_Model.FieldOutputRequest(createStepName='Step-1', name=
```

```python
                    'F-Output-2', rebar=EXCLUDE, region=
                    mdb.models['Model-1'].rootAssembly.sets['load points'], sectionPoints=
                    DEFAULT, variables=('U', 'RF', 'CF'))
                # visualization check
                if bool_check == True:
                    session.viewports['Viewport: 1'].setValues(displayedObject=My_Assembly)
                    session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')
                    session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=ON,
                        bcs=ON, predefinedFields=ON, connectors=ON)
                    session.viewports['Viewport: 1'].forceRefresh()
                    time.sleep(2)
                file.close()
                # create the job
                job_name =  Prefix + part_number + part_number2
                job_name = str(job_name)
                mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
                    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
                    memory=90, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
                    multiprocessingMode=DEFAULT, name=job_name, nodalOutputPrecision=FULL,
                    numCpus=4, numDomains=4, numGPUs=0, queue=None, resultsFormat=ODB, scratch=
                    '', type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
                mdb.jobs[job_name].writeInput(consistencyChecking=OFF)
                # for easy of use, this is the end of the preprocessing with the input written
                # the job then can be submitted with batch script or with python loop
                mdb.jobs[job_name].submit(consistencyChecking=ON, datacheckJob=False)
                mdb.jobs[job_name].waitForCompletion()
                mdb.close()
```

```python
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from abaqus import *
from abaqusConstants import *
from odbAccess import *
from textRepr import *
```

```

import csv
import json
import os

Working_Directory = 'D:/HuaqingRen/CAE_Automated_Pyramid/Step3_Abaqus'

# for change the directory
os.chdir(Working_Directory)
# for debuging purpose
file = open('error_execption.txt','w')
# read the configuration file
configure_path = 'Configure_OSCA.json'

data = json.load(open(configure_path))

Inc_Depth = data['Inc_Depth']

Prefix = data['Part_Prefix']
# read the CSV file

field_data = (())

csv_file_path = Prefix + '.csv'

with open(csv_file_path) as csvfile:
    readCSV = csv.reader(csvfile,delimiter = ',')
    for row in readCSV:
        row = [float(i) for i in row]
        field_data = field_data+(tuple(row),)

total_loop = field_data[-1][1]  # this is the total loop number
total_point_number = field_data[-1][0]  # this is the total point number

# change the directory so that the ODB is output into a single folder
os.chdir((Working_Directory + '/Output_ODB'))


for i in range(1,int(total_point_number+1)):
    for j in range (2,5):
        scale = 4 + 2*j
        current_loop = field_data[i-1][1]
        part_number = "_N"+str(int(i))+"_S"
        part_number2 = str(int(scale))
        saved_suffix = ".txt"
        opended_suffix = ".odb"
        open_path = Prefix + part_number + part_number2 + opended_suffix
        open_path = str(open_path)
        save_path = Prefix + part_number + part_number2 + saved_suffix
        save_path = str(save_path)

        odb = openOdb(open_path)
        file = open(save_path,'w')

        FrameNumber = len(odb.steps['Step-1'].frames)
        for k in range(0,FrameNumber):
        # the above code get the specific frame of the step 1
        # and then, the displacement/load field of that step is extracted
            specificFrame = odb.steps['Step-1'].frames[k]
            displacement = specificFrame.fieldOutputs['U']
            force = specificFrame.fieldOutputs['CF']
            load_points = odb.rootAssembly.nodeSets['load points']
            centerDisplacement = displacement.getSubset(region = load_points)
            centerForce = force.getSubset(region = load_points)
            DisplacementValues = centerDisplacement.values
            ForceValues = centerForce.values
            U1, U2, U3 = [], [], []
            F1, F2, F3 = [], [], []
            for v in DisplacementValues:
                U1.append(v.dataDouble[0])
                U2.append(v.dataDouble[1])
                U3.append(v.dataDouble[2])
            for vk in ForceValues:
                F1.append(vk.dataDouble[0])
                F2.append(vk.dataDouble[1])
                F3.append(vk.dataDouble[2])
            U1_average = sum(U1)/float(len(U1))
            U2_average = sum(U2)/float(len(U2))
            U3_average = sum(U3)/float(len(U3))
            F1_sum = sum(F1)
            F2_sum = sum(F2)
            F3_sum = sum(F3)
                #file.write(str(U1_average)+ '\n')
                #file.write(str(U2_average)+ '\n')
                file.write(str(F3_sum)+','+str(U3_average)+'\n')
            file.close()
            odb.close()
            del odb

```

### *D-2 MATLAB Script for the Control Points Determination*

```
%% clear the workspace and load the toolpath file
clear;
close all;
addpath('Circle_Fit_Function\');
addpath('Clustering\');
%% read in the config file
FileName = 'Configure_OSCA.json';
config = loadjson(FileName);
tool_radius = config.Tool_Radius; % the tool radius is 5 mm
filename_tmp = config.Toolpath_File;
M_top = csvread(filename_tmp,1);
part_prefix = config.Part_Prefix;
```

```
critical_length = ceil(config.Critical_Length/config.Inc_Curve);
Maximum_cluster = round(config.Maximum_Cluster);
Position = M_top(:,1:3);
Normal = M_top(:,4:6);
sheet_thickness = config.Sheet_Thickness;
visual_option = true;
%% plot of the toolpath
figure (701);
plot(Position(:,1),Position(:,2),'.-','MarkerSize',10);
axis equal;
axis([-60,60,-60,60]);
set(gca,'FontSize',15,'Fontname','Arial');
figure (702);
plot(Position(:,1),Position(:,3),'.-','MarkerSize',10);
axis equal;
% axis([-60,60,-60,60]);
set(gca,'FontSize',15,'Fontname','Arial');

%% characterize each point for loop
Loop_index = M_top(:,17);
Loop_index = circshift(Loop_index,1);
Loop = Loop_index;
for i = 2:length(Loop_index)
    Loop(i) = sum(Loop_index(1:i));
end
%% characterize each point for wall angle
Wallangle = acos(Normal(:,3))/pi*180;
Norm_Wallangle = 45; % this is the non dimensional base for wall angle,
                    % which is 90 degree
Wallangle_Norm = Wallangle/Norm_Wallangle;
%% characterize each point for inplane curvature
critical_points = 5; % which means that the proces will gather all the
information at the local point
                      % plus the critical points prior to the point
                      % plus the another group of critial points after the
point
                      % total will be 2 * critical_points + 1
                      % this is suggested to be propotional to the radius
                      % of the tool, here I choose two times of the radius
                      % considering the inc_curve = 1 and then the tool
                      % radius is 5mm
 upper_limit_radius = 10000; % this is the upper limit of
in_plane_curve = []; % reserve the space for in_plane_curve array
in_plane_curve_converg = []; % reserve the space for checking of the converge
in_plane_curve_sign = []; % initialize the space for checking the curve sign
```

```
for j = 1: max(Loop)
    Position_singleloop = Position(Loop == j,:);
    Loop_singleloop = Loop(Loop == j);

    x_mean = mean(Position_singleloop(:,1));
    y_mean = mean(Position_singleloop(:,2));

    in_plane_curve_singleloop = [];
    in_plane_converg_singleloop = [];
    in_plane_curve_sign_singleloop = [];

    for i = 1:length(Loop_singleloop)

        if i < (critical_points + 1)
            i_tmp = i + critical_points;
            Position_singleshifted =
circshift(Position_singleloop,critical_points,1);
            index_i = (i_tmp - critical_points):(i_tmp + critical_points);
            x_pos = Position_singleshifted(index_i,1);
            y_pos = Position_singleshifted(index_i,2);
            xy = [x_pos,y_pos];
            [K,Converg] = CircleFitByPratt(xy,upper_limit_radius);
            in_plane_curve_singleloop(i) = K(3);
            in_plane_converg_singleloop(i) = Converg;
        elseif i > (length(Loop_singleloop) - critical_points - 1)
            i_tmp = i - critical_points;
            Position_singleshifted = circshift(Position_singleloop,-
critical_points,1);
            index_i = (i_tmp - critical_points):(i_tmp + critical_points);
            x_pos = Position_singleshifted(index_i,1);
            y_pos = Position_singleshifted(index_i,2);
            xy = [x_pos,y_pos];
            [K,Converg] = CircleFitByPratt(xy,upper_limit_radius);
            in_plane_curve_singleloop(i) = K(3);
            in_plane_converg_singleloop(i) = Converg;
        else
            index_i = (i - critical_points):(i + critical_points);
            x_pos = Position_singleloop(index_i,1);
            y_pos = Position_singleloop(index_i,2);
            xy = [x_pos,y_pos];
            [K,Converg] = CircleFitByPratt(xy,upper_limit_radius);
            in_plane_curve_singleloop(i) = K(3);
            in_plane_converg_singleloop(i) = Converg;

                if i == 120 & visual_option == true % for the visualization
purpose
                               % for the different location, change the
                    x_pos_display = x_pos;
                    y_pos_display = y_pos;
                    z_pos_disaply = Position_singleloop(index_i,3);

                    xc = K(1);
                    yc = K(2);
                    Re = K(3);
                    kth = linspace(0,2*pi,360);
                    xe = Re*cos(kth)+xc; ye = Re*sin(kth)+yc;
                    ze = Position_singleloop(i,3) * ones(360,1) ;
                    figure (101)

plot3(Position_singleloop(:,1),Position_singleloop(:,2),Position_singleloop(:
,3),'-b.','LineWidth',0.4);
                    hold on;
                    plot3(x_pos_display,y_pos_display,z_pos_disaply,'-
yo','MarkerSize',8,'LineWidth',2.0);
                    plot3(xe,ye,ze,'-r','LineWidth',2);
                    axis equal;
            %        axis([-50,50,-50,50,-40,10]);
                   set(gca,'FontSize',15,'Fontname','Arial');

                end
        end

        x_current = Position_singleloop(i,1);
        y_current = Position_singleloop(i,2);

        center_vector = [x_current - x_mean, y_current - y_mean];
        local_vector  = [x_current - K(1), y_current - K(2)];

        in_plane_curve_sign_singleloop(i) =
sign(dot(center_vector,local_vector));
    end

    in_plane_curve_singleloop(in_plane_curve_singleloop > upper_limit_radius)
= upper_limit_radius;
    in_plane_curve = [in_plane_curve,in_plane_curve_singleloop];
    in_plane_curve_converg =
[in_plane_curve_converg,in_plane_converg_singleloop];
    in_plane_curve_sign =
[in_plane_curve_sign,in_plane_curve_sign_singleloop];
end

    in_plane_curve = 1./in_plane_curve;

%% normalize the inplane curve

   maximum_curvature = 1/tool_radius;

   In_plane_curve_Norm =
in_plane_curve_sign.*in_plane_curve./maximum_curvature;

%% check effect change based on the wall angle

   norm_factor = Wallangle_Norm;

   norm_factor_change = []; % this is for the initialization

 for j = 1: max(Loop)

    norm_factor_singleloop = norm_factor(Loop == j,:);
    Loop_singleloop = Loop(Loop == j);

    norm_factor_change_singleloop = [];

    for i = 2 : length(norm_factor_singleloop)

    factor_increase_singleloop = norm_factor_singleloop(i) -
norm_factor_singleloop(i-1);
    norm_factor_change_singleloop(i) =
abs(factor_increase_singleloop/mean(norm_factor_singleloop));
```

```
    end
    norm_factor_change_singleloop(1) = abs((norm_factor_singleloop(1) -
norm_factor_singleloop(length(norm_factor_singleloop)))...
        /mean(norm_factor_singleloop));
    norm_factor_change = [norm_factor_change,norm_factor_change_singleloop];
 end
 norm_factor_change_wallangle = norm_factor_change';
 %% check effect change based on the in plane curve
   norm_factor = In_plane_curve_Norm';
   norm_factor_change = []; % this is for the initialization
 for j = 1: max(Loop)
    norm_factor_singleloop = norm_factor(Loop == j,:);
    Loop_singleloop = Loop(Loop == j);
    norm_factor_change_singleloop = [];
    for i = 2 : length(norm_factor_singleloop)
    factor_increase_singleloop = norm_factor_singleloop(i) -
norm_factor_singleloop(i-1);
    norm_factor_change_singleloop(i) =
abs(factor_increase_singleloop/mean(norm_factor_singleloop));
    end
    norm_factor_change_singleloop(1) = abs((norm_factor_singleloop(1) -
norm_factor_singleloop(length(norm_factor_singleloop)))...
        /mean(norm_factor_singleloop));
    norm_factor_change = [norm_factor_change,norm_factor_change_singleloop];
 end
 norm_factor_change_inplane_curve = norm_factor_change';

 norm_factor_change = norm_factor_change_wallangle +
norm_factor_change_inplane_curve;
 %% visualization faction
 if visual_option == true
     figure (1);
     plot3( M_top(:,1), M_top(:,2), M_top(:,3),'.');
     hold on;
     plot3( M_top(norm_factor_change>0.1,1), M_top(norm_factor_change>0.1,2),
M_top(norm_factor_change>0.1,3),'ro');
     axis equal;
     figure (2)
     plot3(M_top(Loop==14,1),
M_top(Loop==14,2),norm_factor_change_wallangle(Loop==14),'-bo');
      figure (3)
     plot3(M_top(Loop==14,1),
M_top(Loop==14,2),In_plane_curve_Norm(Loop==14),'-bo');
```

```
     figure (4)
      plot(M_top(:,1),In_plane_curve_Norm);
 end
%% clustering for representive regions
norm_factor_change_matrix = [Wallangle_Norm,In_plane_curve_Norm'];
figure (301)
Idx = [];
Cost = [];
cluster_group = [];
for j = 1:max(Loop)
    X = norm_factor_change_matrix(Loop==j,:);
    [Idx_tmp, Cost_tmp,cluster_group_tmp] =
Automatic_Constrained_1D_Clustering(X,critical_length,Maximum_cluster);
    % note that the cost function in each loop denote the cost increase
    % when do the n cluster combination
    if j == 16
        for i = 1:cluster_group_tmp
            Y = M_top(Loop == j,1:2);
            if i == 8
               plot(Y(Idx_tmp==i,1),Y(Idx_tmp==i,2),'go','MarkerSize',12);
            else
            plot(Y(Idx_tmp==i,1),Y(Idx_tmp==i,2),'o','MarkerSize',12);
            hold on;
            axis equal;
            end
        end
    end
    Idx = [Idx, Idx_tmp];
    Cost = [Cost, Cost_tmp]; % need to be careful when access the cost
function
    cluster_group(j) = cluster_group_tmp;
end
%% for noting the critical point
Boundary = [];
Middle = [];
for j = 1:max(Loop)
    X = Idx(Loop==j);
    Y = circshift(X,1);
    Boundary_tmp = ~(X == Y);
    cluster_group_tmp = cluster_group(j);
    if cluster_group_tmp == 1
        Boundary_tmp(1) = true;
    end
    Bourndary_Idx = find(Boundary_tmp); % should be the length of
cluster_group
    Center_Idx = 0 * Bourndary_Idx;
    if cluster_group_tmp >1
        Center_Idx(1) = round(0.5 * (Bourndary_Idx(1)+ length(X)) + 0.5 *
Bourndary_Idx(cluster_group_tmp));
        if Center_Idx(1) > length(X)
            Center_Idx(1) = Center_Idx(1) - length(X);
        end
        for i = 2: length(Bourndary_Idx)
            Center_Idx(i) = round(0.5 * Bourndary_Idx(i-1) + 0.5 *
Bourndary_Idx(i));
        end
    end
    Middle_tmp = 0 * Boundary_tmp;
    if cluster_group_tmp >1

        for k = 1: cluster_group_tmp

            Middle_tmp(Center_Idx(k)) = true;

        end

    end

%     Boundary = [Boundary, Boundary_tmp];
%     Middle = [Middle, Middle_tmp];
    B_index_tmp = 0;
    M_index_tmp = 0;
    B_index_max = 1; %1 % maximum number of the boundary point, or simply a
large number can be assigned
    M_index_max = 2; %2  % maximum number of the middle point,  or simply a
large number can be assigned
    for k_tmp = 1:length(Boundary_tmp)
        if Boundary_tmp(k_tmp) == 1
            B_index_tmp = B_index_tmp + 1;
            if B_index_tmp > B_index_max
                Boundary_tmp(k_tmp) = 0;
            end
        end
         if Middle_tmp(k_tmp) == 1
            M_index_tmp = M_index_tmp + 1;
            if M_index_tmp > M_index_max
                Middle_tmp(k_tmp) = 0;
            end
        end


    end
    Boundary = [Boundary, Boundary_tmp]; % this is a trick way
                                               % only pick several point
                                               % since the part symetricity
    Middle = [Middle, Middle_tmp];
end
```

```
%% global visualization

figure (400);
    hold on;
for j = 1:max(Loop)
    Y = M_top(Loop==j,1:2);
    Idx_tmp = Idx(Loop==j);
    Boundary_tmp = Boundary(Loop==j);
    Middle_tmp = Middle(Loop ==j);

for i = 1:cluster_group(j)

            if i == 8
               plot(Y(Idx_tmp==i,1),Y(Idx_tmp==i,2),'g.','MarkerSize',12);
            else
            plot(Y(Idx_tmp==i,1),Y(Idx_tmp==i,2),'.','MarkerSize',12);
            hold on;
            axis equal;
            end

            plot(Y(Boundary_tmp==1,1), Y(Boundary_tmp==1,2),'ro',
'MarkerSize',12);
            plot(Y(Middle_tmp==1,1), Y(Middle_tmp==1,2),'bo',
'MarkerSize',12);

end

end


%% output the point and index


    index_local = ((Boundary == 1) | (Middle==1));
    M_local = M_top(index_local,1:3);
%     M_local = M_local';
    Loop_local = Loop(index_local);
    Number = 1:length(Loop_local);
    Number = Number';
   M_output = [Number Loop_local M_local];
   file_name = [part_prefix,'.csv'];


   csvwrite(file_name,M_output);


%% output for the thickness field distribution
xfit = linspace(-125,125, 200);
yfit = linspace(-125,125, 200);
[xq,yq] =   meshgrid(xfit, yfit);

x = Position(:,1);
y = Position(:,2);
z = Position(:,3);
```

```
normalq = griddata(x,y,abs(Normal(:,3))* sheet_thickness,xq, yq, 'nearest');
zq = griddata(x,y,z,xq, yq, 'nearest');
% close all;
for i = 1:max(Loop)
    x_output = reshape(xq,1,numel(xq));
y_output = reshape(yq,1,numel(yq));
z_output = reshape(zq,1,numel(zq));
t_output = reshape(normalq,1,numel(normalq));
    z_height_tmp = z(Loop == i);
    truncated_height = z_height_tmp(1);
    index = (z_output < truncated_height);
    z_output(index) = truncated_height;
    t_output (index) = sheet_thickness;
%     figure (303);
%     plot3(x_output,y_output,t_output);

    M_output = [x_output', y_output', z_output',t_output'];
    file_name = [part_prefix,'_TF',int2str(i),'.csv'];
    csvwrite(file_name,M_output);
end


%% k mean position based clustering
for j = 8
X = M_top(Loop == j,1:2)/80 + 0.5;
Y = [Wallangle_Norm,In_plane_curve_Norm'];
Y = norm_factor_change_matrix(Loop==j,:);
X = [X,Y];

rng(1); % For reproducibility
[idx,C] = kmedoids(X,8,'Replicates',5);
figure (102);
hold on;
    for i = 1:8
    if i == 8
       plot(X(idx==i,1),X(idx==i,2),'go','MarkerSize',12);
    else
    plot(X(idx==i,1),X(idx==i,2),'o','MarkerSize',12);
    hold on;
    axis equal;
    end

    end
end
%
%% linkage clustering
for j = 2
    X = M_top(Loop < 7,1:2);
    [idx,Dist]=knnsearch(X,X,'k',80);
    D = zeros(size(X,1));
for ii = 1:length(X)
    D(ii,idx(ii,2:end)) = 1;
end
cLinks = linkage(D, 'ward');
c = cluster(cLinks, 'maxclust', 9);
    figure (103);
    hold on;
    for i = 1:9
    plot(X(c==i,1),X(c==i,2),'o','MarkerSize',12)
    end
end
```

### *D-3 MATLAB Script for the On-line Force Gather and Springback Modification*

```
clear all;
close all;
addpath('Including/');
addpath('generated_toolpath/');
addpath('toolpath_generation/');
addpath('Delta_Tau_Func/');
load('Config.mat');
load([config.Part_Prefix,'_OSCA.mat']);
%% toolpath & system configuration
control_no = ceil(config.Control_Loop_Inc/config.Loop_Inc);
last_point_compensation = 0;
last_loop_compensation = 0;
max_inc_loop = 2 * config.Loop_Inc;
config.Inc_Curve = 0.5;
max_inc_point = 0.01 * config.Inc_Curve; % only 1 um is allowed per change
point (0.5 in this case)
average_base = round(2 * config.Tool_Radius / config.Inc_Curve);
config.Tool_Speed = 6.0;
config.Max_Line = 4000;
%% Initialize the related global variables
global gathering_status Update_Freq;
global hPmacDevice deviceNum;
%% ask to chose the device, can judge whether the machine is on-line or not
hPcommServer = actxserver('PcommServer.PMacDevice');
hPmacDevice = hPcommServer.invoke('IPmacDevice');
%% start the device connection
deviceNum = 0;
open_success = hPmacDevice.Open(deviceNum);
if open_success ~= 1
    error('Delta Tau Connection Failed. Please Check the Device and Restart
Connection');
```

```
end
gathering_status = true;
%% save the files in the local dictionary
file_name = [config.Part_Prefix,'_',int2str(i),'.csv'];
%% initialize/start the stop GUI
Panel_Handle = GUI_Stop_Panel();
%% initialize/ start the figure
[Axes_Handle, Plot_Handle] = Figure_Plot_Innitialize() ;
%% initialize the data storage class
Gathered_Position_Force = Force_Position_Data();
%% initialize the data plotting/saving timer
Update_Freq = 20;
Gather_Timer = Position_Timer(1/Update_Freq, Axes_Handle, Plot_Handle,
Gathered_Position_Force);
%% wait and check the status variable to stop the timer
   maximum_wait_time = 500 * 60;  % this is the maximum time for a loop to
run
   save_interval = 30;
for i = 1 : 3
            file_name = [config.Part_Prefix,'_',int2str(i),'.mat'];
            load(file_name);
            index_selected = M_selected(:,1);
            loop_range_min = (i - 1) * control_no + 1;
            loop_range_max = i * control_no;
            selected_index = (M_full(:,2) >= loop_range_min) & (M_full(:,2)
<= loop_range_max);
            M_full_selected = M_full(selected_index, :);
            % springback predict phase
            if i >  1
                [force_value,springback_value] =
sprigback_reference(index_tmp,force_tmp, M_full_selected(:,7:12));
            else
                force_value = 0 * M_full_selected(:,1);
                springback_value = 0 * M_full_selected(:,1);
            end
               % compensation value caculate for the control point
            compensation_selected = springback_value * 1.0;
                % linear interploatation to get the full loop interplotation
            compensation_inter =
interp1(M_full_selected(:,1),compensation_selected, M_selected(:,1)...
                ,'linear','extrap');
            % average filter based on average compensation value
              compensation_ave_fil =
average_filter(M_selected(:,2),compensation_inter,...
                loop_range_min, loop_range_max,
last_loop_compensation,max_inc_loop );
            % impose the first point of the interplotation to be the last
point of the last loop
```

```
            [index_imposed, compensation_imposed] =
initial_imposing(M_full_selected(:,1), compensation_ave_fil, ...
                loop_range_min, config.Index_Level,
last_point_compensation );
            % gradient filtering limit the gradient
               compensation_grad_fil = gradient_filter(compensation_imposed,
max_inc_point);
            % moving average filter
                compensation_mov_fil =
mov_aveg_filter(compensation_grad_fil,average_base );
            % final process - take out the first imposed point, and return
the new
            % last_point_value and last_loop_value
            [compensation_final, last_point_compensation,
last_loop_compensation] = final_filter(...
                compensation_mov_fil, M_selected(:,2), loop_range_max);
            L = get(Panel_Handle.ls,'string');  % Get the users choice.
            L{length(L)+1} = ['Compensation
value:',num2str(last_point_compensation) ];
            set(Panel_Handle.ls,'string',L); % Set the new string.
            % change the target toolpath
            M_changed = M_selected;
            M_changed(:,6) = M_changed(:,6) - compensation_final;
            M_changed(:,9) = M_changed(:,9) - compensation_final;
            M_changed(:,12) = M_changed(:,12) - compensation_final;
            toolpath_file_name =
['PMAC_toolpath\',config.Part_Prefix,'_PMAC_',num2str(i),'.txt'];
            writeMachineCurves_withIndex(M_changed(:,7:9),
M_changed(:,10:12),...
                M_changed(:,1), config.Tool_Speed,...
                toolpath_file_name, start_M, end_M);
            % visulization portion

Figure_Visulization(M_selected,compensation_final,loop_range_max);
            % ask for the confirmation for download
            decision  = Confirm_Uploading_Run(last_loop_compensation);
            if decision == false
                break;
            end
            % function to download
            toolpath_to_upload =
['PMAC_Trial\',config.Part_Prefix,'_PMAC_',num2str(i),'.txt'];
            path_name = pwd;
            filename = [path_name,'\', toolpath_to_upload];
            a_tmp =  hPmacDevice.Download(deviceNum,filename,1,1,1,1);
            % start the timer and run the program
            gathering_status = true;
            start(Gather_Timer);
            pause(1);
            % ask for the confirmation for running program
            decision  = Confirm_RunmProgram(i);
            if decision == false
                break;
            end
            SendCommand('&1 B2 R');   % run the toolpath
            L = get(Panel_Handle.ls,'string');  % Get the users choice.
            L{length(L)+1} = ['Program run at:',datestr(datetime('now'))];
            set(Panel_Handle.ls,'string',L); % Set the new string.
            % while-loop to run the program
            while true
            pause(1);
               if gathering_status == false
                   stop(Gather_Timer);
                   break;
               end
            end
            % pull the data and save it into the csv file

[Time_Data,Command_Position_Data,Compensated_Position_Data,Force_Data,
Virtual_Index, Normal_Force_Data,Motor_Status] =
Gathered_Position_Force.pull_data();
            index_tmp = Virtual_Index; % save the index data to the index
variable
            force_tmp = Force_Data; % save the force data to the temp
variable
            Data_To_Save = [Time_Data, Command_Position_Data, ...
                            Compensated_Position_Data,Force_Data,...
                            Virtual_Index, Normal_Force_Data,Motor_Status];
            filename_csv =
['gathered_signal\',config.Part_Prefix,'_Singal_',num2str(i),'.csv'];
            dlmwrite(filename_csv,Data_To_Save,'-append');
            %toc;
            % save the data
            %tic;
            L = get(Panel_Handle.ls,'string');  % Get the users choice.
            L{length(L)+1} = ['Saved at:',datestr(datetime('now')) ];
            set(Panel_Handle.ls,'string',L); % Set the new string.
            pause(1);
 end
%% close the machine connection
hPmacDevice.Close(deviceNum);
```