AALBORG UNIVERSITY

STUDENT REPORT

Numerical optimization applied in sheet metal forming

Finite Element Modelling

Department of Mechanical and
Manufacturing Engineering
Fibigerstræde 16
DK-9220 Aalborg Ø

Petrut Alexandru Matei

Preprint

This document is written and typeset in LaTeX. For the body of this document the Latin Modern 11pt. font is used, while Helvetica is used for headings. The inner and outer margin is 3 cm and 2 cm respectively, while the top and bottom margin is 2.5 cm.

Department of Mechanical and Manufacturing Engineering
Fibigerstræde 16
DK-9220 Aalborg Ø
http://www.en.m-tech.aau.dk/

AALBORG UNIVERSITY

STUDENT REPORT

Title:
Numerical optimization applied in sheet metal forming: Finite Element Modelling

Semester:
4th

Semester Theme:
Master thesis

Project Period:
2016-02-01 to 2016-06-01

ECTS: 30

Supervisor:
Karl Brian Nielsen

Number of pages:
59
84

Printed copies:
2

Petrut Alexandru Matei
pmatei14@student.aau.dk

Abstract

This project has been carried out in collaboration with the Department of Mechanical and Manufacturing Engineering, with the purpose of developing the knowledge in creating a numerical optimization algorithm that can cope with highly non linear functions in sheet metal forming.

An analysis of sheet metal processing has been performed with a focus on critical aspects in this type of process. Further on, a study has been made in implementing optimization algorithm and using a finite element software. Multiple models have been developed in the finite element software and implementations of optimization algorithm are made. An analysis of the results is performed, based on which has been concluded that the finite element simulations are within the imposed limits. The optimization algorithm is proven to be stable, providing with an optimum point for different simulations. It is proven that numerical optimization and simulations can be an extremely effective tool in the sheet metal forming field. Based on the results of the report, further implementations are recommended in order to improve the performance of the overall process.

Preface

This project has been undertaken by Petrut Alexandru Matei, as part of the 4th semester on the Manufacturing Technology master program at Aalborg University, Aalborg. The project it is viewed as a master thesis, since the forth semester it is the last semester. The project period has ran from 2016-02-01 until 2016-06-01. A reading guide for the project is given at page xiii. The project is been carried out by collaborating with the Department of Mechanical and Manufacturing Technology. The scope of the project is to develop the knowledge in the numerical optimization field, combined with finite element simulations of deep drawing processes.

Petrut Alexandru Matei, Aalborg University, 2016-06-01

Acknowledgements

I would like to acknowledge Karl Brian Nielsen, for the support offered throughout my two years at Aalborg University and especially for the help provided as a supervisor in my last semester. Through the guidance and time throughout the semester, I have learned how to cope with new topics and develop skills based on the knowledge acquired.

Abbreviations

FEM . . . . . Finite Element Model
LDR . . . . . Limited Drawing Ratio
LSPP . . . . LS PrePost
AAU . . . . . Aalborg University

Reading Guide

This report consists of chapters numbered 1 to 16. Figures and tables are consecutively numbered in order of the chapters in which they appear. A list of key abbreviations is available on page xi. All citations are referred to in the text by the author-year method. The full list of references is available in the bibliography on page 61. Several files are enclosed with this report, such as a study performed in parallel with the report and a instruction manual on finite element modelling. The full contents of the enclosed files is listed in Appendix A, page 63. In the PDF version of this report, internal cross-references are maroon and references are blue. The chapters of the report are described briefly below.

Chapters

Chapter 1 Introduction: sets the stage for the project, collaboration with Aalborg University and the Department of Mechanical and Manufacturing Engineering.

Chapter 2 Optimization and finite element simulation: introduces the reader to the topic that will be undertaken in this project, the importance of the topic in the modern industry, illustrated with examples.

Chapter 3 Introduction to sheet metal processing: offers an overview on sheet metal forming process, in respect to critical aspects that have to be taken in consideration, the information gathered is to be used in the FEM simulations.

Chapter 4 Problem statement: a concrete problem statement for the project is given, based on which the future work will be developed.

Chapter 5 Optimizer architecture: gives a description of the architecture used in developing the optimizer, such as communication and information flow between software.

Chapter 6 FEM software and simulation: a description of the FEM software is given in respect to the considerations made when the model is developed.

Chapter 7 FEM model: focuses on the model developed in respect with the knowledge gathered

Chapter 8 Optimization algorithms: illustrates the technique used to develop the optimizer, and what considerations are important.

Chapter 9 Programming the optimizer: describes the code developed in order to develop the optimizer.

Chapter 10 Complete optimizer: where a description of the actual optimizer and its functionality is given.

Chapter 11 Methodology: where the methodology used to perform the analysis of the optimizer results is described.

Chapter 12 Results analysis: focuses on determining the simulation results, matching them with the optimization results, and based on which interpretations are made of the optimizer performance and changes that improve the optimization process.

Chapter 13 Implementation in tube hydro forming: describes the results obtained by implementing the optimizer in a new process, different than the one it was created for.

Chapter 14 Future work: delivers a perspective on how the optimizer can be improved in respect to the optimization, simulations and the complete architecture.

Chapter 15 Discussion: focuses on criticizing the relevance of the results found and the major findings,with respect to their limitations.

Chapter 16 Conclusion , based on the problem statement.

Appendices

Appendix A Enclosed Files: lists and briefly describes the enclosed files.

Appendix B LS Dyna key files: offers the complete files for the simulation software.

Appendix C Programing : contains the programming code used fro the optimizer.

Contents

CHAPTER 1

Introduction

"The most interesting part of using optimization techniques to determine an optimum is the fact that I can reach results that I was not thinking of"

This is a quote from a brief meeting with Karl Brian Nielsen, a professor at Aalborg University (AAU), describing how an optimizer can handle complex problems in a better way that the human mind can. Aalborg University is an educational institution founded in 1974, which uses a problem based and project oriented teaching. One of the main aspects of this teaching method is that it allow students to cope with real engineering problems, that companies are facing in the industry. In the department of Mechanical and Manufacturing Engineering, one emphasis is placed on using optimization software in different areas of the production, such as process improvement, production flow, etc., with the purpose of coping with problems that companies are facing nowadays in their manufacturing plants. From the proposal of the project the following is stated as the initial problem statement:

"The goal of the project is to develop and investigate the performance of using an optimizer combined with a simulation software, when coping with process problems that can be found in the industry"

CHAPTER 2

Optimization and finite element simulation

In this chapter the academic interest in using optimization it is described, with a focus on the reasons why the topic is of interest for AAU, for a future engineer and why is the knowledge in creating optimization systems valuable. Section 2.1 introduces the reasons for which one would want to use an optimization software, while Section 2.2 focuses on using optimization and FEM at a process level.

2.1 Importance of using optimization systems

It is important to understand that an optimization system is not universal applicable, but the knowledge of creating an optimizer can be considered universal, since most of the optimizers have a similar structure. Therefore, if one knows how to create an optimizer, one will be able to apply it in different fields, not only in production engineering. Such fields can be related with optimizing transportation, tasks organizing and many other fields, since reaching an optimum is a must in every business, no matter how one defines the optimum. Optimization is basically a mathematical concept that is applied to the real world.

Aalborg University has a special relationship with Danish companies, since the projects undertaken by students are based on problems that exist in industry. Many companies are facing issues with the fast developing technology, high request for customized products and short delivery therms, thus creating a need to comply with the demand in a smaller time frame. There are multiple factors that can influence the production process, such as salea information, process time, product development time and external factors, such as the relation with the sub suppliers.

In such situations, a company does not have time to have extensive trial and error processes in order to test their products. This conjuctural frame creates a need to implement nmerical optimization and FEM simulation to reduce the time used to develop and launch a product, to reach an optimum of the production process in respect to the demand, or to take decisions based on which the profit is maximized.

The academic interest lies in using optimization software that is tailored made for different applications, thus experience is gained and further on, the University can initiate a research in fields where the optimization is not very easy to implement, where functions are highly non linear and more often than not, multiple adjustments must be made in order to reach an optimum. Gaining this type of knowledge can lead back to problems that are already existing in industry, and companies that cannot afford to invest in a research, can easily benefit, since the problems that they encounter may have already been solved in a different project.

2.2 Using optimization software at a process level

The focus of the project will be on process control, due to the fact that one can extract the process from the whole production process and work with it on a individual level. For example, in the aeronautic industry, the main challenge is related with the usage of new materials, or production processes in order to yield high quality parts. The investigation focuses more on using FEM software and optimization software to create parts that have to fit multiple requirements, such as strength, fatigue resistance product life span and reliability on the results obtained. The reasons for focusing on the process lies in the usage of expensive materials, high quality standards, high production costs and especially low production rates.

In this type of situations, where for example one aircraft is produced each month, as stated by Boeing in their production forecast Boeing 2016, improvements on a process level can yield lower costs for the production as a whole. Expensive parts, produced in small batches are the main source for the high costs of producing an air plane, and since there is so little to optimized in the production flow, improving the process is most often the only way to have a better rate of cost per part. In this type of situation, using and optimization and an FEM software can prove to be an extremely powerful tool, yielding better geometries, spotting mistakes in the design or creating new design features to reach the quality requirements. One example of using optimization software with FEM simulation is illustrated in Figure 2.1, where topology optimization is used to determine the structure of an airplane rib, a component used to support the wings.

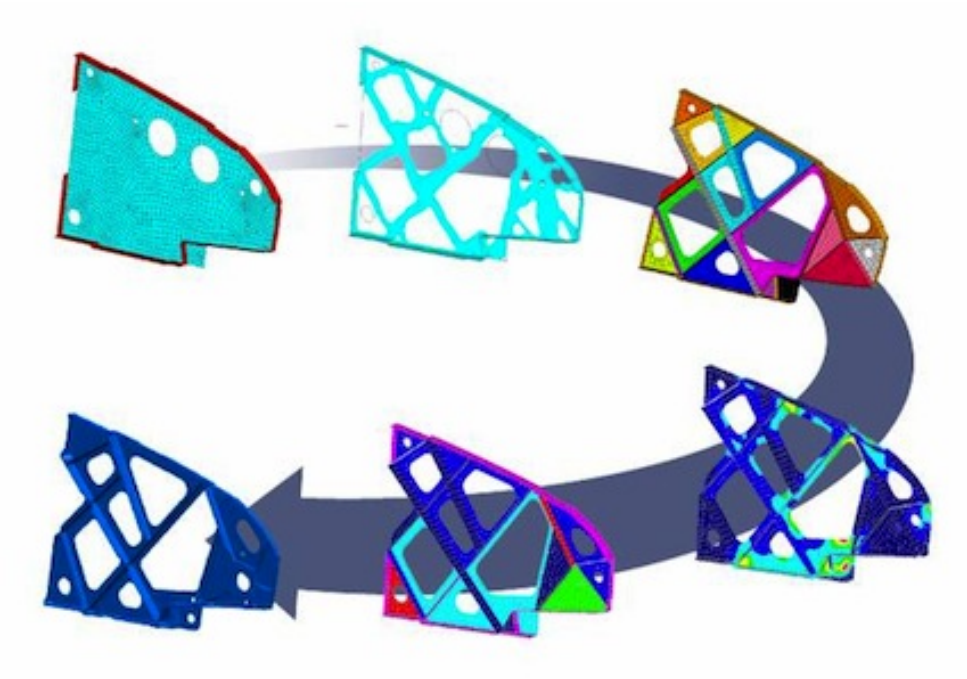

Figure 2.1: Optimization of an airplane rib.

CHAPTER 3

Introduction to sheet metal processing

In this chapter an introduction to sheet metal processes is given, regarding the basic process, the history behind it and the role it plays in the modern times. Afterwards, the importance of improving the process is described, along with the process that is chosen for this project. Lastly an introduction to the software and technique used in this project is given.

3.1 Basic process and history

The process of metal forming it is an ancient one, but the first machine that was built in order to industrialize the process has been invented by Leonardo da Vinci in 1480, with the purpose of reducing the thickness of metal sheets. Later in the 17th century, the machine was used to reduce the thickness of led plates, or to produce gold coins. The first rolling mill has been recorded in 1768 in Newcastle, England and it was able to produce plates of 1500 [mm] by 700 [mm] in various shapes.

Nowadays, the sheet metal processes are varied, the industry playing an important role in large manufacturing companies, such as car manufacturing, aeronautics and other metal products. In Europe only metal working industry has reported a 300 [mil] turnover in 2008, based on The sheet process can vary from stamping, rolling, bulging laser forming to hydro-forming in order to achieve cut patterns or various shapes.

Although the sheet metal process has been improved increasingly in the past decades, there are certain parts of the process which by improving them can yield cost optimized products. One aspect that plays a key role in the price/product lies in the process where molds are used to give the shape for the metal sheet. This could be either a hydro-forming process or trough the use of a punch that forces the sheet to change its shape.

These processes have certain control methods such as pressure, or the punch velocity that can influence the quality of the product. In order to optimize these parameters, a strong and less costly approach it is to simulate the process and optimize it using numerical optimization techniques. One of these processes it is illustrated in Figure 3.1, where a punch it is used to change the shape of a metal sheet, process also known as deep drawing. In these type of processes the control of the forces are crucial, resulting in high quality products or low quality products as illustrated in Figures 3.2 and 3.3, where a faulty process can result with wrinkled products, while only a fault free process can result with a high quality product.

This type of problems will be further analysed in this report, in order to determine how problems that are specific to deep drawing process can be avoided by taking an analytical approach, since more often than not this approach it is the cheapest one.

In order to develop a simulation that can replicate the deep drawing processes, the Finite Element Analysis (FEA) software LS- Dyna it is used, since it can handle highly non linear processes, with increased plastic deformations in a short process time. For developing the optimization algorithm, Java Oracle it is used, mainly due to the existing knowledge. In Section 8.1 an introduction is given to the optimization method used for the purpose of the project.

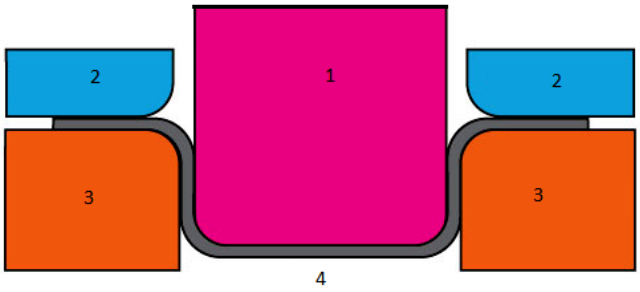

Figure 3.1: Illustration of sheet metal processing using a punch (1), blank hollder (2), die (3) and the metal sheet (4)

Figure 3.2: Illustration of a faulty deep drawing process of a sink.

Figure 3.3: Illustration of a sink obtained trough a fault free deep drawing process.

3.2 Process performance factors

Deep drawing of metal sheets is a process that depends on a multitude of factors, and since the process is already well developed, there are certain rules that one has to respect in order to achieve a successful deep drawing process, based on Plesha 2011. Therefore one has to focus on:

- The ratio of the blank diameter to the punch diameter.
- The clearance between the die and the punch.
- The radius of the punch.
- The die shoulder radius.
- The blank holder and the punch force.
- The friction and lubrication.

Deep drawability

The therm deep drawability refers to the property of the blank of sustaining thickness reduction, which is determined by the ratio of the maximum blank diameter and the punch diameter, also called Limited Drawing Ratio (LDR) Plesha 2011. The ratio is determined using an experimental method, based on the normal anisotropy of the material, noted with R. The way R is computed is related with the anisotropy of cold metal sheets. R is computed trough a tensile test, using a 20% elongation. The tensile test is take at 45 degree, at 90 degree and in the normal direction. Then an average R is computed, and based on this value, there is experimental data data one can use to determine the ratio of the diameters. It is important to state that this is the technique that provides a very accurate result. Table 3.1 gives typical average values for R.

Table 3.1: Typical values of R_{avg} for various sheet metals.

Material	R_{avg}
Zinc	0.4-0.6
Hot-rolled steel	0.8-1.0
Cold-rolled rimmed steel	1.0-1.4
Aluminium alloys	0.6-0.8
Titanium alloys	3.0-5.0
Stainless steel	0.9-1.2
High strength alloy steel	0.9 -1.2

Figure 3.4 illustrates the relationship between LDR and R_{avg}, where it can be seen that for steel, LDR value is ‘between 2.1 and 2.2.

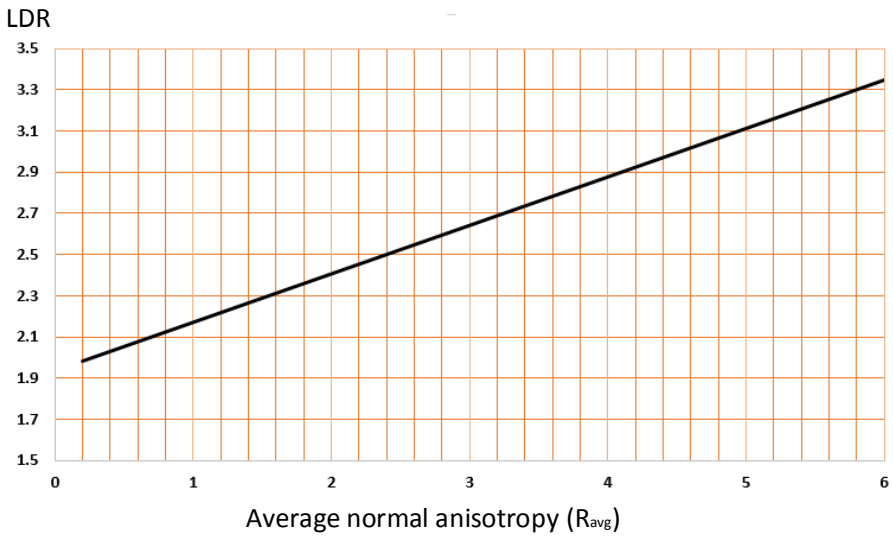

Figure 3.4: Illustration of how LDR is influenced by R_{avg}.

The blank holder pressure and the punch force

If the blank holder force is low, then wrinkles will appear, if it is to high, there will be tearing in the material.

The blank holder pressure is considered to be between 0.7% and 1.0% of the sum of the yield strength and the ultimate tensile strength of the sheet metal, which for a mild steel is 1100 [MPa], making the blank holder pressure to be around 11 [kN].

The punch force is determined based on the distance needed to travel and the time requested for the process.

Other recommendations

One recommendation refers to the clearance between the die and the punch, which is recommended to be with 14% higher than the thickness of the metal sheet.

Lubrication should only be considered at the contact between the die, or the blank holder and the blank, since it helps to reduce forces, thus increasing drawability, while for the punch, it is the either way around. The geometry such as the radius of the die shoulder or the punch has to be considered according to the process, but they should be sufficient to allow a smooth flow of the material.

The thickness of the plate has to be between 0.5% and 1% of the diameter in order to avoid wrinkles by using the blank holder. if lower than 0.5%, the wrinkles may not be avoided.

CHAPTER 4

Problem statement

The purpose of this chapter is to define the problem statement based on which the work of the project will be continued. Chapter 2 introduces the interest of implementing FEM simulation with an optimization process in order to achieve desired results. In Chapter 3 a description of the sheet metal process is given. The article Matei 2016 represents a paper that has been written during this project in order to illustrate how an optimization algorithm works, where an introduction to optimization algorithms is given. The paper can be found enclosed with the remaining files of the report. Based on the knowledge acquired in respect to the optimization algorithm and the sheet metal processing, the following problem has been stated:

Develop an optimizer capable of handling highly nonlinear functions, which can be implemented with a variety of geometry, in order to result in determining the global optimum based on the data obtained from a stable FEM simulation process.

The work that will be carried out will have a focus on the optimization algorithm, the structure of the complete optimization system and lastly on the FEM modeling. Although it is important to match the results with the reality, due to time limitations it is not possible to perform and analyze a deep drawing process in laboratory. The focus will be placed on creating a robust algorithm, that can handle multiple geometries, and based on the results obtained, recommendations for future implementation will be considered.

CHAPTER 5

Optimizer architecture

The purpose of this chapter is to explain the overall architecture used to connect the optimization software with the FEM simulation, how data is extracted and how it is manipulated. Multiple software systems are interconnected in order to enable communication between the different environments they operate in. Section 5.1 describes the architecture itself, what are the purposes of using each software and a brief description of the software is given as well. In Section 5.2 it is described how all the software can communicate with each other, what types of command are used and relevant to know. Lastly, in Section 5.3 a discussion is given on how could this implementation be improved, what other options could one use and why is this implementation chosen.

5.1 The overall architecture

In Figure 5.1 the overall architecture is illustrated. The main software that compose the architecture are:

- MobaXTerm, a software used for communication with the AAU supercomputer.
- UNIX, the operating system found on the AAU supercomputer.
- Java Oracle, used to build the optimization algorithm.
- LS Dyna, used to perform the FEM simulations.
- LS PrePost (LSPP) used as a pre-processor.

Java Oracle is the software of choice for developing the optimization algorithm. The implementation in Java Oracle focuses on launching applications such as LS Dyna or LS PrePost, extracting data, manipulating and rewriting data on files.

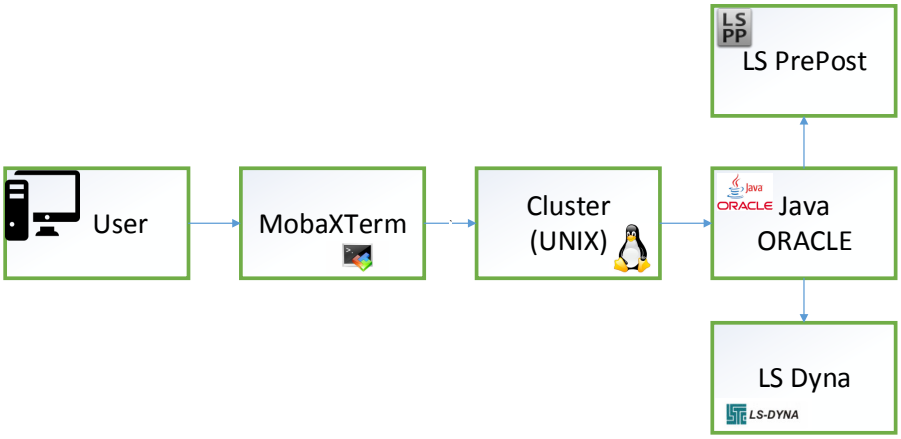

Figure 5.1: The overall architecture used to optimize the FEM simulation.

LS Dyna is chosen for multiple reasons, related with the existing knowledge in the university, the fact that it has a long tradition in the FEM analysis, with many examples that can be found for free, thus making the learning process easier.

LS PrePost it is a free software, and it is used to create the simulation environment and then post it in LS Dyna. In this way one does not need to have a supercomputer to run complicated simulations. LSPP it is used as well for extracting data from simulations and to visualize results.

MobaXTerm is a software used to create terminals, in which a user can connect and access another computer, in this case the supercomputer AAU has provided with.‘ Although it has many more applications, this is its main purpose in the project.

5.2 Extracting and manipulating data

The data needed to run the optimization process is related with FEM parameters, FEM results and launching applications in batch mode. The files used by LS Dyna are a text file, where all the information about the FEM simulation is written. Further on the data outputted by LS Dyna or LSPP it is written in a text file, thus one only has to enter and access it. Example 5.2.1 is given regarding the code implemented in Java to start and run LS Dyna.

example 5.2.1

```
public static void igniteDyna2() {
try {
String[] cmd = { "/bin/sh", "-c", "dyna_s i=model.k NCPU= -8 > dyna.tmp" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("Dyna running, waiting to finish");
p.waitFor();

} catch (InterruptedException t) {
System.out.println("Error: didnt wait for dyna?");
}
} catch (IOException e) {
System.out.println("Error: did not execute the command rigth.");
}
}
```

The most important aspect of this command is the line ”$/bin/sh$”, ”$-c$”, ”$dyna\_s\ i = model.k\ NCPU = -8 > dyna.tmp$” where the command ”$/bin/sh$”, ”$-c$” enables running LS Dyna in batch mode, and the rest of the command specifies the software that is going to be started, in this case LS Dyna, the file that the software should run and the number of the processors should be used while running the file.

One feature that plays a key role is related with merging two files together. This approach it is used for simplifying the data manipulation, since in a small text file the data is simpler to extract and write back. This is illustrated in Example 5.2.2. As in the previous example the command here it is specific for UNIX systems. The command "cat file.k curve.k > model.k" adds the information form the file "‘curve.k"’ to the file "file.k" and the new created file will be called "model.k", where the information from both files will be placed.

example 5.2.2

```
public static void mergeKfiles() {

try {
File tempFile = new File("model.k");
if (tempFile.exists()) {
System.out.println("delete file for design parameters change");
tempFile.delete();
}
String[] cmd = { "/bin/sh", "-c", "cat file.k curve.k > model.k" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("waiting for process, mergeKfiles");
p.waitFor();
} catch (InterruptedException t) {
System.out.println("Error: didnt wait for merged files");
}
} catch (IOException e) {
System.out.println("Error: The files could not be merged.");
}

}
```

5.3 Discussion

In this chapter, only one approach has been described, since is the one used in the project. The commands described in Section 5.2 are not the only ones that can be used, since there is no right way of coping with this specific situation. The implementation of the software into creating an architecture can be done in multiple ways, the one chosen here provides a good computational time, a stable communication between the software and a fairly easy way of understanding how all the problems are being coped with. Further on , in the chapters where the software implementation will be described at an individual level, there will be a link to the commands described in this chapter.

FEM software and simulation

The purpose of this chapter is to introduce the reader to LS Dyna, regarding basic knowledge that it is important to have in order to understand the complete process behind implementing the optimization algorithm. LS Dyna is a software that can handle highly non linear processes, in respect with plastic and elastic problems. For this project, LS Dyna it is used to perform the FEM simulation, using the super computer that AAU is equipped with. Having the capacity to use this computer enables a faster simulation time, due to the 48 cores that it has within 8 processors.

6.1 Simulation development in a pre processor

In order to create the geometry and input the conditions for the process that is simulated, a pre processor is used, called, LS PrePost (LSPP). LSPP saves the data in a file that can be read as a text file. The complete simulation file is composed of multiple keys, where a key represents a specific detail, such as geometry, loads or control options.

These keys are loaded in LS Dyna as a whole, and processed further on. One key example is illustrated in Example 6.1.1 where one material is defined.

example 6.1.1

```
*MAT_RIGID
$#     mid        ro         e        pr         n    couple         m    alias
         2 7.8700E-9 2.0000E+5  0.300000     0.000     0.000     0.000
$#     cmo      con1      con2
     0.000         0         0
$# lco or a1      a2        a3        v1        v2        v3
     0.000     0.000     0.000     0.000     0.000     0.000
```

It can be seen in Example 6.1.1 that the definition of the key starts with ”*”, afterwards the name of the key is given, which is unique for each key. The position of the keys in the key file, as the output file from LS PrePost it is called, has no importance. This enables to combine multiple key files into one, and obtain a new key file containing the information from different files. This can be done trough the software or just adding the information from one file into another by copying it or jut adding them together. This technique can be used in order to make changes to certain properties of the model and add them later on. As for example, one could change the values given for this material, where "ro" stands for density, "e" for Young's modulus, and "pr" for Poissons ratio, and add them to the model. In this way one can manipulate the material characteristics and run simulations without having to look trough all the listed keys, or using different files.

Another important aspect of LS Dyna is that it uses three unit systems to which one has to pay attention when creating the model. In the previous example, a key is defined for a material. In this key, the value for density is given as $7.87E-9$ but no unit is listed. These is where the units consistency is important. Here the density of steel is defined as in [ton/mm] which is a derivative of the metric system. All the other units used, have to be defined in respect to this unit system. Further on, LS Dyna offers the possibility of using the metric system, the imperial or a customized one, and but one has to pay attention to the units he is using since they are

no where specified in LS PrePost. The units consistency systems are illustrated in Appendix B, Table B.1.

6.2 Material selection

LS Dyna enables the user to replicate of variety of materials, by offering a set of 281 material models. A material model has incorporated features that are specific to a type of material but at the same time, some of the features can be predominant or not, although they exist within the model, LSTC 2016 material section. In order to illustrate what a model is, steel is defined using two material models. One material model is material 001 which is a purely elastic model, while the other one chosen for this example is material 018, which is a model that has incorporated features for defining plastic characteristics of a material.

Thus, defining the steel trough material 018 the user has to input data for defining the plastic characteristics, such as hardening exponent, strength coefficient or strain rate parameter, while material 001 requires just the generic properties, such as density, Poisson's ratio and Young's modulus.

When choosing a material, it is important to emphasize more the aspects that are important to the simulation that is going to be performed. In a simulation of a deep drawing process, usually the die, the blank and the punch are made from materials that are similar to a purely elastic material, in comparison with the blank, that has to undergo a plastic deformation. In this specific case it is recommended to focus on replicating the material used for the blank in correspondence with the reality, and keep the other material as rigid material. The reasons behind it are related to the time wasted on defining the other materials, the fact that it will increase the simulation time and last, the results will be only slightly different.

6.3 Shell elements

The geometry in LS Dyna can be created using a solid element or a shell element. Choosing one of the element dictates how the geometry should be created. A solid element is illustrated in Figure 6.1 and a shell element is illustrated in Figure 6.2. If one chooses to work with solid elements the geometry that needs to be created has to have a volume, while if choosing shell elements, the volume it is not required. The shell and solid elements can also be combined in a simulation, as illustrated in Figure 6.3, where the geometry is a mix of planar surfaces and volumes.

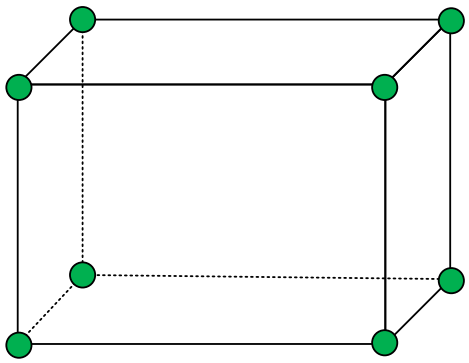

Figure 6.1: 8 nodes solid elements.

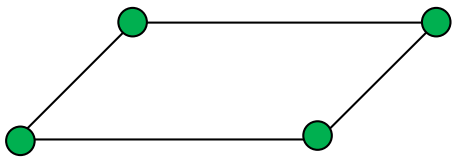

Figure 6.2: 4 nodes shell elements.

The shell element is preferable in a simulation due to the fact that it has less nodes to define one element, 4 compared to 8 as it can be seen in the illustration above.

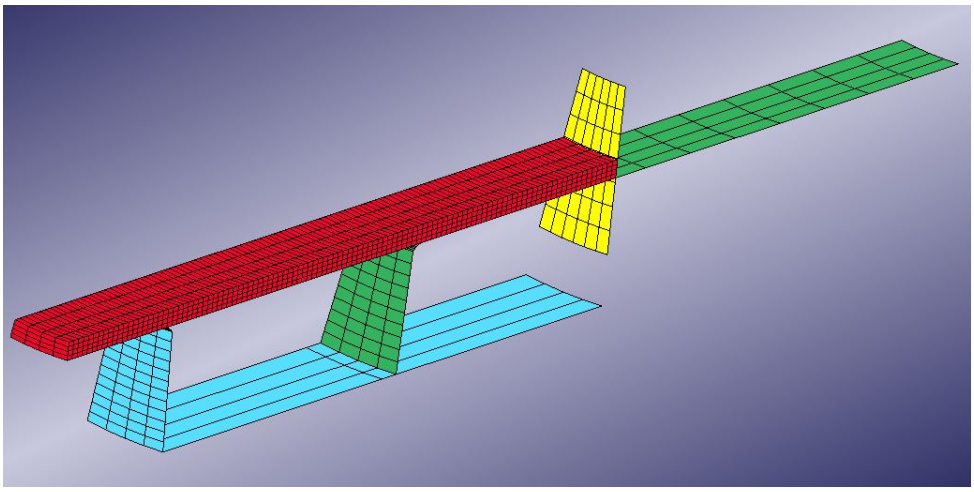

Figure 6.3: Illustration of a shell and solid elements used in hydro=forming process.

This implies a faster simulation time and a simpler way to define the geometry. The shell elements have different formulations in LS Dyna, and their definition impacts directly the performance of a simulation, due to the fact that the elements have different integration method. The default shell element is using the Belytschko-Tsay formulation, based onLSTC 2016 shell elements section, and it is set to default since it provides good results within a large number of fields. When choosing a type of element shell one has to make a trade of between the computational cost of the element and its effectiveness. The Belytschko-Tsay formulation is consider an extremely effective element with a low computational time. The problem with such and element is that the element can degenerate during a simulation into a triangular element, which is not desired in certain applications. Further on, some elements perform better for simulations that involve high deformations, such as the fully integrated formulation, but at the same time it is 3 times more costly in computational time. For applications in the deep drawing field, the default formulation can be used for accurate results, thus the Belytschko-Tsay formulation will be used further on in the report for shell elements.

6.4 Discretization

The discretization, or the mesh, represents a split of the geometry into smaller shapes, that can be either planar shapes, for shell elements or 3D geometrical shapes for solid elements. The discretization of a model, using shell or solid elements can be done in multiple ways. One has to be in control on how the discretization is done due to multiple factors mesh related, that can influence a simulation. These factors are:

- Small elements are resulting in longer computational time than bigger elements.
- Elements with different shapes, in terms of size and shape can yield different results.
- Non symmetrical mesh is resulting in inconsistent results across the model.
- High discrepancy between the mesh used on different components, the outcome of the simulation will not fit the reality.

In Figure 6.4 a thorough controlled mesh is illustrated, while in Figure 6.5 a mesh that is made using the auto mesh option is given. It can be observed that the auto mesher cannot split the geometry into an symmetrical mesh, and different element shapes are used such as triangular, rectangular or quadrilateral elements. In order to be in control of the mesh, a N line mesh is used.

Since the desired shape of the mesh is a quadratic mesh, the mesh is defined using four edges, on which a number of nodes is specified. If on the surface there are not four edges available, then the geometry is split in order to create multiple surfaces that are delimited by four edges, as in Figure 6.4

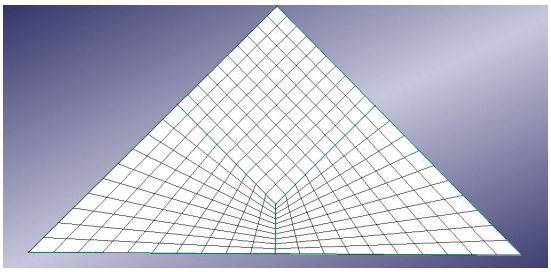

Figure 6.4: Geometry split in order to reach 4 edges delimited surfaces. The mesh is well in control.

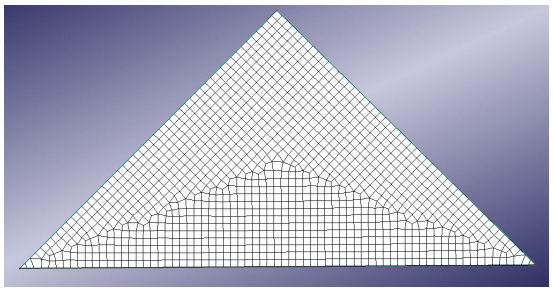

Figure 6.5: Automatic mesh of the surface. Little control on the mesh.

For the models used in the project, the mesh will be created as it follows.

- The focus on the blank will be to have a mesh that contains even distributed elements.
- The parts that are treated as being rigid bodies, the die, the punch or the blank holder, will be meshed with elements of the same size as the blank, and a small focus on the mesh distribution.
- The mesh on the blank has to allow replicating aspects of the deep drawing process, such as wrinkling, spring back etc.
- The mesh should be big enough in order not to result in a increasing simulation time.

6.5 Surface contact

The contact it is described at a very brief level, but it is important to know what type of contact one should choose and why. When defining the contact , two main characteristics remain identical throughout the different contact types, and that is the slave and master definition. The slave defines the part on which the forces will be computed, while the forces on the master usually are computed only if necessary. The contact can be defined as sliding, tied, between two surfaces, nodes to surfaces or edges on surface.

What a contact does, it enables to check for penetration between elements of different parts, and compute the contact forces as well. The way the contacts are defined can increase or reduce the simulation time and stability. LS Dyna offers a variety of contacts, some of them being suited only for specific applications, as it can be seen in LSTC 2016 section contact definition.

For the simulations that will be performed, the contact used will be "AUTOMATIC ONE WAY SURFACE TO SURFACE". This type of contact is suited when the master is a rigid body, very common in deep drawing processes. Further on, the difference between the mesh from different parts is less likely to affect the simulation results, and the nodes checked for penetration are the save nodes, thus resulting in a less computational time. The contact is illustrated in Figure 6.6.

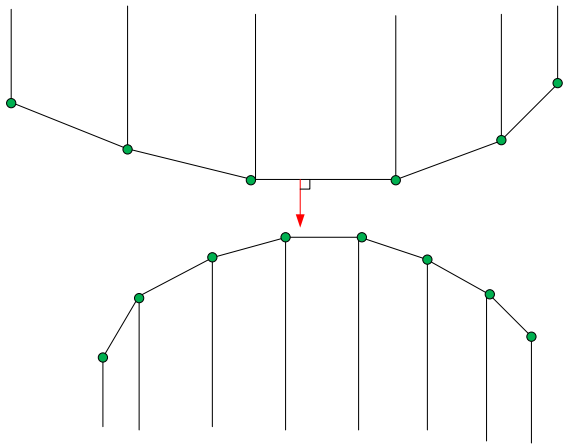

Figure 6.6: Illustration of the contact AUTOMATIC ONE WAY SURFACE TO SURFACE

6.6 Governing equations of the software

The importance of this section lies in the fact that most of the decisions taken in replicating the reality are based on the governing equations. Although, they will only be brushed on the surface, it is a good starting point for getting a hold on the basics. The section is completely based on Cook 2002 The governing equations are going to refer to the computation of the internal forces and time integration, in order to illustrate how they can affect the performance of a simulation. It is important to emphasis that the computation is done for each element, and afterwards, it s summed in order to obtain the value for the model. If the stress situation for the element is known, the internal forces are computed as follows:

$$r_{int} = \int_{V_e} [B]^T \sigma dV \tag{6.1}$$

where $[B]^T$ is the strain displacement relation vector and σ is the stress vector. By summation trough all the elements, the global internal force vector is computed.

The time integration starts from the general dynamic equilibrium point:

$$[M]a + [C]v + [K]x = F_{ext} \tag{6.2}$$

where [M] is the mass matrix, [C] is the dumping factor and k [K] is the element matrice. a, is the acceleration vector, v the velocity vector and x is the time. The equation must hold at different times for linear elastic problems. Assuming that the start point is at time t, a computation is made by incrementing the time with Δt. The displacement trough time incrementation is expanded in a Taylor series as follows:

$$x_{t+\Delta t} = x_t + \Delta t * \dot{x_t} + \frac{\Delta t^2}{2} * \ddot{x_t} + ... \tag{6.3}$$

$$x_{t-\Delta t} = x_t - \Delta t * \dot{x_t} + \frac{\Delta t^2}{2} * \ddot{x_t} - ..... \tag{6.4}$$

Summation of the two equation yields:

$$\ddot{x_t} = \frac{x_{t+1} + x_{t-1} - 2x_n}{\Delta t^2} \tag{6.5}$$

Subtraction yields :

$$\dot{x_t} = \frac{x_{t+1} - x_{t-1}}{2\Delta t} \tag{6.6}$$

Equations 6.5 and 6.6 are inserted in Equation 6.2 and using the values for the external forces, the internal forces are computed. The internal force is computed as following:

$$F_{int} = [M]\ddot{x} - F_{ext} \tag{6.7}$$

One can chose to include the dumping factor as well. One element is illustrated in Figure 6.7, in order to illustrate the displacement computation.

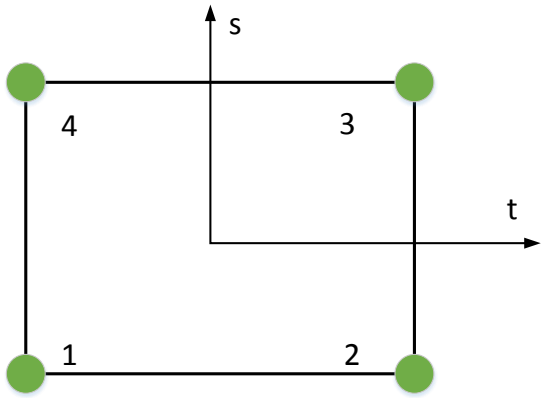

Figure 6.7: Element illustration used for describing the displacement computations.

The shape function matrix [N] for a 4 nodes element can be written as $N_1 = \frac{1}{4}(1-s)(1-t)$, $N_2 = \frac{1}{4}(1+s)(1-t)$, $N_4 = \frac{1}{4}(1-s)(1+t)$ and $N_3 = \frac{1}{4}(1+s)(1+t)$, where s and t are coordinates within the surface of the middle point. The element position is computed in respect to the global coordinate system. The element position in respect with x axis is computed by interpolating trough the shape matrix in respect to x, and the position of the element in respect to y is computed by applying the same technique. From the displacement field the engineering strains are computed.

6.7 Discussion

This chapter introduces the reader to topics that are more complex than illustrated, but for the purposes of the following chapters the description will suffice. The basics are given in order to enable the justification of decisions taken when developing the FEM model, but also in order to interpret the results. In this project the focus will be placed only on the selection made in respect to material, shell elements, etc. The reason for not going deeper in different material types, different shell elements or contact types is due to the complexity of each selection. LS Dyna provides with a manual that covers the topics described in greater detail, and is important to state that a thorough research is needed before using the software, since it does not put any constrains on the user, thus mistakes with a high impact on the simulation are possible.

CHAPTER 7

FEM model

The purpose of this chapter is to introduce the geometry used in the simulations and the FEM considerations that are to be implemented in LS Dyns. In section 7.1 a detailed description of the geometry is given, with a focus on how is the geometry created in LS Dyna to replicate a real model and also the dimensions used. Section 7.2 gives a description of how the parts are modeled in FEM software. In Section 7.3 a discussion is taken in respect with the approach taken, what other methods could have been used, and the reason for using the methods presented.

7.1 Geometry

The geometry consists of a set-up in which a deep drawing process of a metal plate is done. The set-up consists of four parts, the punch, the blank, the blank holder and the die. Figure 7.1 and 7.2 illustrate the die and the blank holder, while Figure 7.3 and 7.4 illustrate the punch and respectively blank.

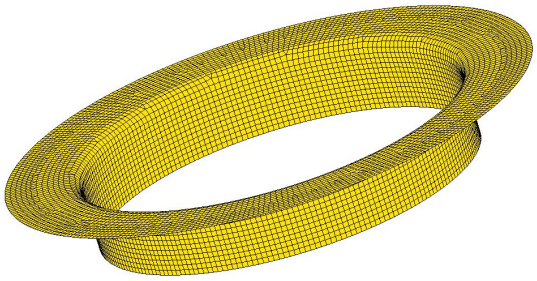

Figure 7.1: Illustration of the die.

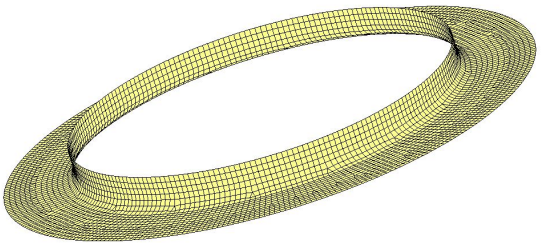

Figure 7.2: Illustration of the blank holder.

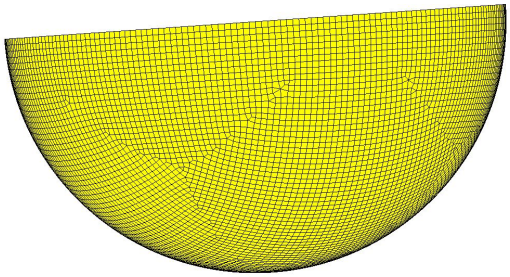

Figure 7.3: Illustration of the punch.

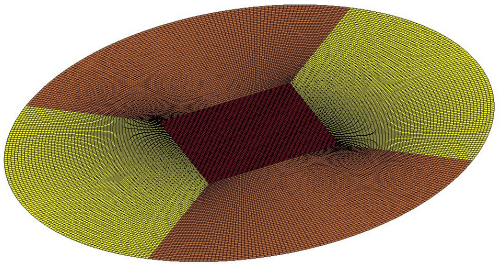

Figure 7.4: Illustration of the blank.

In Figure 7.5 the complete set-up it is illustrated, while in Figure 7.6 geometrical details are given. The geometrical measurements are:

- Rp the punch radius
- Rd the die radius
- Rb the blank radius

The punch is moving in the z direction, thus forcing the blank to deform. Pressure it is applied on the blank holder in order to achieve a process without wrinkling and to avoid the spring back effect. The travel distance of the punch, the pressure applied on the blank, and other aspects related with the physical process are defined in Chapter 12, since there are multiple geometries involved.

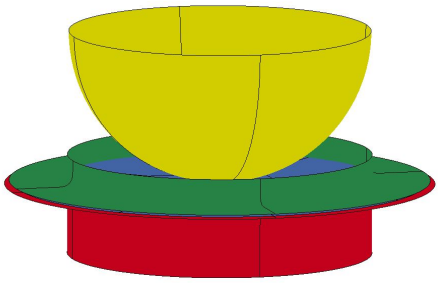

Figure 7.5: Illustration of the complete assembly.

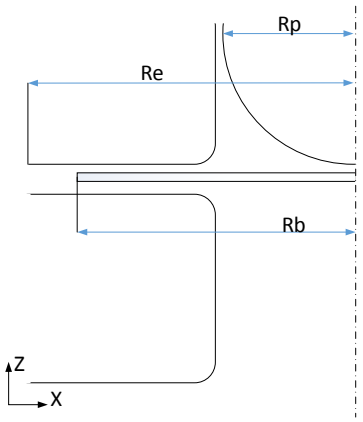

Figure 7.6: Two dimensional drawing of the assembly, with main dimensions.

The geometry of the punch creates a deep drawing process where the blank will be under continuous thinning process. The punch it is a half of sphere, and this is the reason for which a high degree of stress it is expected. In order to cope with this problem a punch with the shape of a round prism and rounded edges is used. Figure 7.7 illustrates the punch that will be used in parallel in order to compare the two simulations.

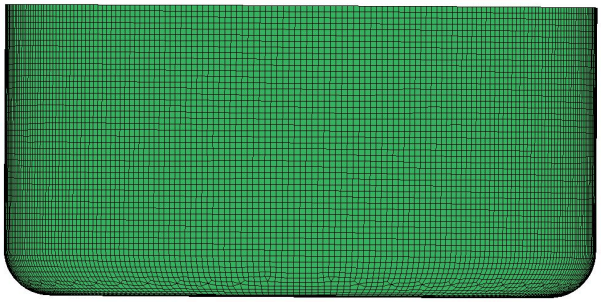

Figure 7.7: Circular prism punch, with round edges.

7.2 FEM modelling

Due to the simulation time that increases with the number of the elements used for the discretization of the model, and since a high number of elements it is required to simulate the process properly, only a quarter of the model will be used for the actual simulation.

The blank it is meshed as illustrated in Figure 7.8, while Figure 7.9 illustrates the pattern used for the meshing process. The blank it is divided in three areas, in order to make the discretization simpler and more reliable. Thus, a high number of elements it is used in order to create the mesh for the two symmetrical surfaces. In this way it is ensured that certain phenomena of the process can be replicated by the model, such as wrinkling, material stretching or other material behaviors. Further on the mesh used for the model provides with a simple way of controlling the elements in the blank, but also to understand the process results outputted by the simulation software.

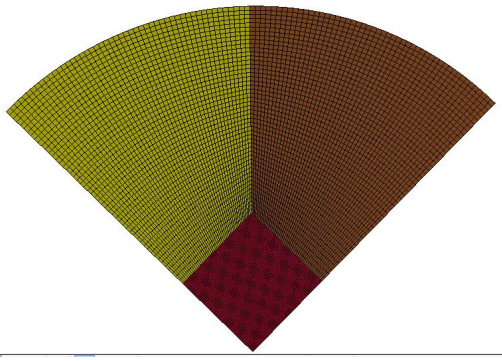

Figure 7.8: Illustration of the mesh on the blank.

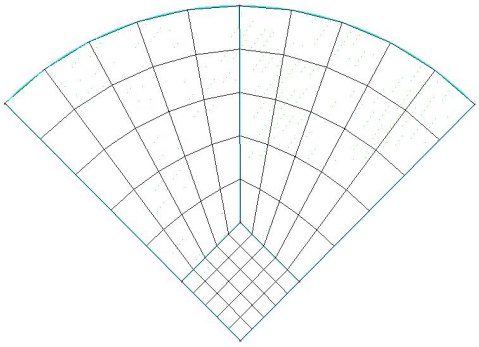

Figure 7.9: Illustration of the pattern used to create the mesh.

The straight edges of the blank have symmetry conditions to match the quarter of the model, but at the same time some components have additional constraints, in order to ensure that the simulation process will be a stable one. The die is completely constrained, while the punch and the blank holder are allowed to move only in the z direction. No rotational constraints are allowed.

For the FEM model, shell elements have been used due to less calculation time, since they need less elements than a solid material, and provide with a much more simpler way to create the geometry.

The blank is the only component that has plastic properties, in order to enable the deformation process, while the rest of the components are treated as a rigid body. The properties of the two materials are illustrated in Table 7.1, where it is important to mention that the material properties defined are the minimum required in order to replicate a material behaviour. Since the purpose it is not to create a very accurate model of the real material, but to implement an optimization algorithm for the model, then the model is created with the minimum requirements. It is as well important to specify that the material properties are defined in this manner to match the units consistency in LS Dyna, as illustrated in Appendix B.

Other aspects that are related to the modelling of the geometry are the shell element chosen, which in this case is the default shell element, with the Belytschko-Tsay formulation. The contact between the parts is AUTOMATIC ONE WAY SURFACE TO SURFACE, important since the mesh in the parts uses elements with similar length, but not identical. The reason for using this type of contact is explained in detail in Chapter 6 Section 6.5. It is important to mention that the contact is always defined as having the deformable part as the slave part.

The travel distance of the punch is specified using a load curve that defines the length of the travel distance within a specific time, in this way the simulation is not ran to fast, and aspects

Table 7.1: Material properties used in the simulation.

	Material	
Properties	Rigid	Plastic
Density [ton/mm]	7.87E-09	7.87E-09
Young's modulus [N/mm^2]	2.00E+05	2.00E+05
Poisson ratio [-]	0.3	0.3
Strength coefficient [Mpa]	-	550
Hardening exponent [0]	-	0.25

such as kinetic energy are kept in the acceptable proportions. The simulation time will generally be 0.01 [s], but if needed it can be increased. This will be specified when it will be done. Other aspects that are relevant are the mass scaling, which is not used in the simulations, since the time for running one simulations is around 5[min], which is considered acceptable mostly because the results will not be altered. It is important to understand that this is an outline for the basic model used in the FEM software. Changes might be made along the line and they will be listed when needed.

7.3 Discussion

In the previous sections a description is given to explain the basic model. The question that arises is if the model could be improved. In respect in how the geometry is created in LS Dyna, creating a full scale model more often than not will not bring an advantage, rather a disadvantage. In terms of discretization, different approaches can be taken, one better than others, but since the purpose of the project is not to make a deep study in FEM modelling, it is enough to have a model that is stable and performs acceptable, when compared with real data. Further on ,the elements used are 50 times smaller than the dimensions that define the blank, thus the model can replicate phenomena such as spring back or wrinkling. The elements used provide a good trade of between the simulation time and the accuracy of the results. Other approaches could have been taken for determining the mesh size, such as the adaptive mesh option, but in this particular case, one can easily approximate the element size, without being wrong.

Modelling only a quarter of the model might seem to much as well, since the model can be reduced to a few degrees of a complete circle, but doing it in this manner makes it easier to create the geometry and to mesh the blank in a more efficient manner. Further on, when creating a smaller model, smaller elements have to be used. Smaller elements, with a high energy within result in large computational time, inaccurate results, and even unstable simulations.

CHAPTER 8

Optimization algorithms

The purpose of this chapter is to describe the optimization algorithm used in the project. In section 8.1 a description of the basic algorithm is given with respect to the mathematics behind it. Further on, in Section 8.1.1 an improvement to the actual method is presented while Section 8.2 ends the chapter with a discussion on the method used, and what other methods could have been used.

8.1 Optimization method used for optimum search

This section is based on Endelt 2016 and Arrora 2004, Chapter 8, and it shows how an optimization algorithm is set up in order to yield an optimum process, where the desired outcome has to be reached. Before starting the optimization implementation it is important to understand what is to be optimized, what optimization technique can be used and how to translate the physical process in a mathematical model. The implementation technique used is a gradient based optimization technique combined with a non linear least square algorithm.

This type of algorithm cannot be found in commercial solvers and enables customization for a specific problem. The specific of non linear equations is that the coefficients of the function cannot be fitted using simple matrix techniques, thus the solution is found trough an iterative search. The objective function, based on Endelt 2016 is evaluated as :

$$f(x) = \sum_{j=1}^{m} r_j^2(x) \tag{8.1}$$

$$r_j(x) = \hat{y}_j - y_j \tag{8.2}$$

The structure of the objective function is used to illustrate the error between the requested, or imposed data set represented by y_j, in Equation 8.2, and the estimated data set obtained after one iteration process, represented by $\hat{y}_j$ in the mentioned equation. In Equation 8.1, x represents the design vector, while in Equation 8.2 r is the residual vector.

In order to determine which is the search direction for the next iteration, a steepest descent optimizer is created in Java Oracle, where the gradient points the opposite direction of the search direction. The purpose is to use the gradient in order to determine in which direction, steps should be taken in order to minimize the objective function. For a multi variable function, the gradient is computed as a vector of the partial derivatives of that function, as illustrated in Equation 8.3.

$$c = \nabla f(x) = \frac{\partial f}{\partial x_i}, \text{ where } i = 1, n \tag{8.3}$$

c is the gradient at point x_i and n is the number of variables or parameters. The search direction for the steepest descent is defined as :

$$d = -c \tag{8.4}$$

The next step is to determine the step size used for updating the vector x and make a new iteration. First, updating the vector x is done by updating each component of the vector as follows:

$$x_{k+1} = x_k + \alpha_k * d_k \tag{8.5}$$

where k is the counter for the current point, d is the search direction and α is the step size that has to be taken in order to minimize the function. The step size, α is computed as follows:

$$\alpha_k = \frac{f_k - (1-p) * f_k}{-||c||^2} \tag{8.6}$$

wheref_k is the current function value, p is the reduction in the objective function, and c is the gradient at the current point. Before implementing the algorithm it is important to set some conditions for the optimizer to stop. One criterion is that the length of the gradient $\|c\|$ is zero, but this is always hard to obtain, thus, ϵ is created as a value that the length of the gradient should be smaller than. A second criterion is to allow only a certain number of iterations, thus k is initialised as a counter of each iteration.

The complete algorithm structure is illustrated in Figure 8.1, and it represents the following steps:
1. Set the initial parameters for the objective function f.
2. Define ϵ and set counter k.
3. Calculate the gradient c at point x_k.
4. Calculate the length of the gradient $\|c\|$.
5. Evaluate the stopping conditions, $\|c\|$ smaller than ϵ and k smaller than the maximum value. If they are not accomplished move forward.
6. Calculate search direction as - c, then calculate the step size α.
7. Update the current point as $x_{k+1} = x_k + \alpha_k * d_k$.
8. Update counter $k = k + 1$.
9. Start again from step 3

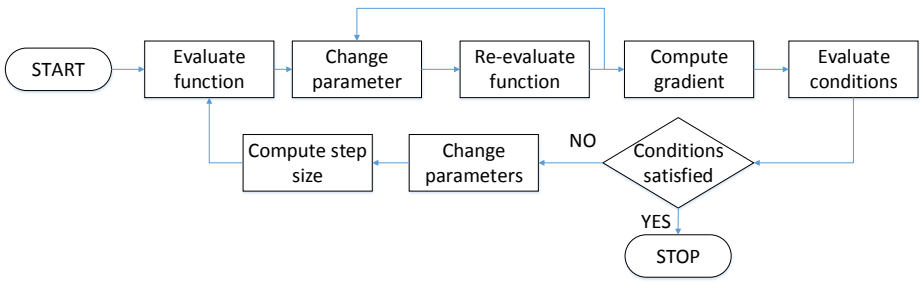

Figure 8.1: Illustration of the optimization algorithm.

One aspect that has to be considered is that the gradient cannot be computed thus an approximation method is used, as follows:

$$c_{k+1} = \frac{f_k - f_{k+1}}{\alpha} \tag{8.7}$$

The next step is to understand how to translate the real world problem in order to fit the optimization algorithm. First it is important to understand what is there to be optimized and what can be controlled from the simulation in order to reach the proposed goal. In the simulation for tube hydro-forming, several aspects have to be considered as important, such as:

- Reaching the proposed geometry.
- Avoid severe thinning.
- Avoid destroying the material by using a very high pressure.
- Avoid high plastic strains.
- Avoid wrinkles.

All this problems are translated in a function that has to be minimized, and their implementation should be made taking in consideration the value scales they have, since one error function could have small values while another can yield high values in the beginning. In order to understand this type of situation, in Figure8.2 two error values are plotted during an iterative process of minimizing the objective function. It can be seen that in the beginning one error plays an

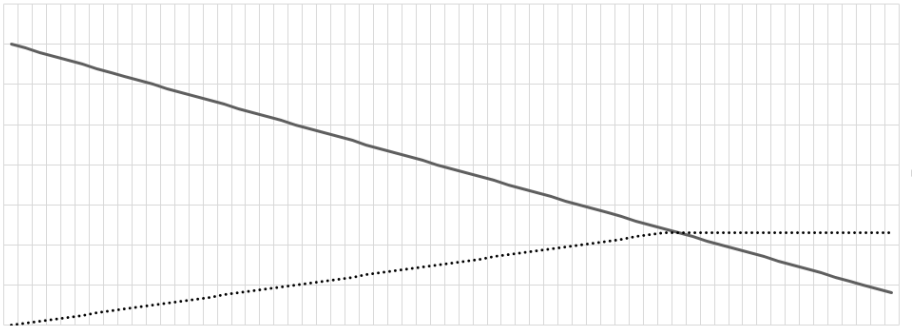

Figure 8.2: Values plotted during an iterative process, for two functions.

important role in the objective function, while the second one has a significant smaller value. Throughout the iterative process, one function decreases in value while the other increases and gets stabilized around a certain value, that could be higher than the first function. From here two types of situation can be extracted, one where the function with small values needs to be artificially increased in order for it to make a difference in the yielded value of the objective function. The second situation occurs when the function with smaller values increases above the other function, this resulting in a possible movement in the opposite direction of the optimum point.

8.1.1 Choosing the conjugated gradient method

The conjugated gradient method is slightly different than the steepest descent since what it does is to incorporate the information from the last iteration into the actual iteration. Based on Endelt 2016 this small difference that consists of a few code lines yields a much higher convergence rate. In the conjugated gradient method the information from the previous iteration is incorporated as following:

$$\beta_k = \left(\frac{\|\nabla f(x_k)\|}{\|\nabla f(x_{k-1})\|} \right)^2 \tag{8.8}$$

Based on Equation 8.8 the search direction is computed as:

$$d_k = -\nabla f(x_k) + \beta_k * d_{k-1} \tag{8.9}$$

where d_{k-1} is the search direction from the previous iteration, thus making the conjugated gradient method applicable after the first iteration.

The Hessian

In order to get a better approximation of the function, it is recommended to use the second order information or the Hessian, which is basically the second derivative of the function. The Hessian its proven to be computational expensive, and more than once has been proven that the information from the Hessian has a small or no impact on the function evaluation, but can have a high impact on the computational time. For this reason, and due to the simplicity of the process simulated, the Hessian has been excluded from the function.

The step size

The step size in the simulation it is approximated after running the optimization software and visualizing the results. Although there are multiple methods to approximate the step size, such as the golden section or other line search methods, there are also computational expensive and for running a simulation with a low number of parameter, one can approximate the step size and by supervising the optimization process, one would now when the step size is either to big or to small.

Algorithm performance

The function used can definitely can use improvements and but due to time limitations, this will be considered as a future work. Implementations such as the Broyden class, the totally structure secant method, the Quasi-Newton method and many others have been considered, but they have also not been implemented due to the fact that they are an addition to the method implemented, and not a complete different method.

In Figure 8.3 illustrates the performance of different implementations.

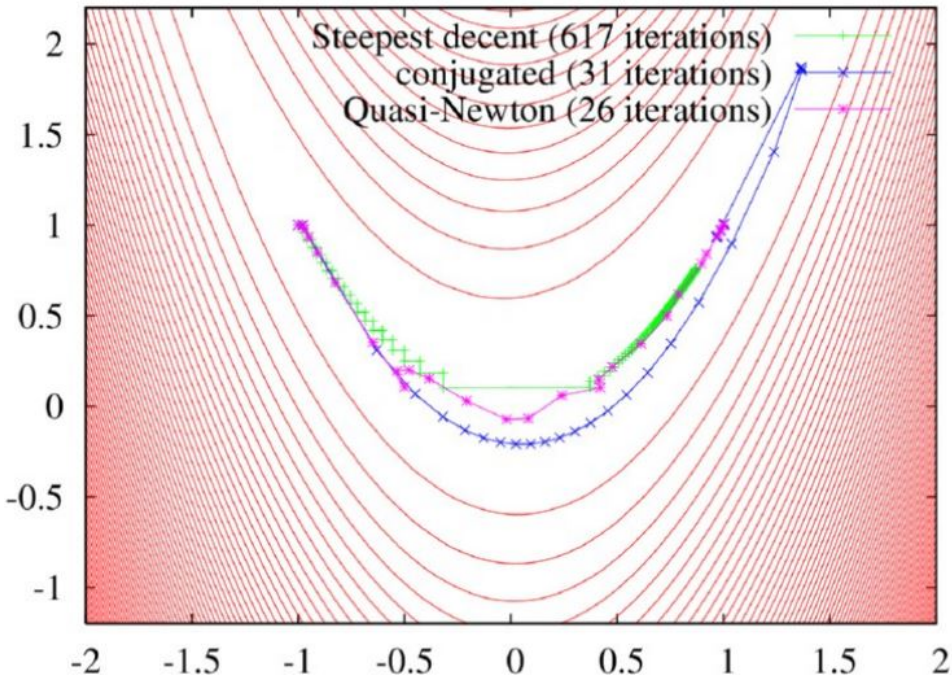

Figure 8.3: Illustration of the performance for different implementation, Endelt 2016 .

The steepest descent method is a very simple and basic method, not very well suited for complex functions, which can easily stop at a local optimum, for this reason the steepest descent it is used with the least square error.

The analysis is taken from Endelt 2016 and it shows that for a relatively simple function, using the steepest descent method, will not yield in a good optimization time. Based on Endelt 2016 the method with the fastest convergence rate is the Quasi Newton implementation, but one has to consider the fact that this optimizations are tailored for specific applications, thus over complicating them might result in a function that performs at a lower level. This is also one of the reasons for using the implementation as it is, since the user it is familiarized with the expected results, understands the behaviour of the optimization algorithm and can take decisions based on the results obtained.

8.2 Discussion

The discussion focusing on the methods that could have been used and the methods used in this report. Endelt 2016 provides with multiple methods that can improve the step size determination, and the information that is included when approximating the objective function. Arrora 2004 provides with multiple methods that can be implemented in order to linearise the function or to improve the overall performance of the optimizer. Even so, the method chosen is very stable and performs well with this types of engineering problems.

CHAPTER 9

Programming the optimizer

The purpose of this chapter is to describe the implementation made in Java eclipse of the optimization algorithm. The focus will be placed on how the code was developed, in therms of strategy and also in therms of knowledge required in the programming language. The chapter is based on the documentation provide by Java Oracle on their official website Oracle 2016. Section 9.1 describes the strategy taken in developing the algorithm while Section 9.2 focuses more on specific commands used in the programming language. Lastly, Section 9.3 ends the chapter with a discussion on the implementation made.

9.1 Programming strategy

It is important to describe the structure of a Java class, in order to understand how a program is built. One class is illustrated in Example 9.1.1 A Java class starts with a package declaration, then the class is started, public variables are defined, main method created and then secondary methods are created. The package declaration refers to a set of instructions that are available, based on which certain operations are performed, such as opening a file, reading it or writing it.

example 9.1.1 `Class structure in Java programming language`

```
public class Function {

public Function() {
//public variables are to be declared here
}

public static void main(String[] args) {
//the main programming code goes here

}
//secondary methods go here

}
```

The class definition gives information about the type of the class used, either public or private, and the name of the class. In the main method the commands are written, and they work with the information within the class. The secondary methods are accessed if necessary, and they can provide with a small computational method or they can run other classes and extract the information from them.

Strategy

The programming strategy takes advantage of the option to call other classes from on main class. A class is a set of commands, and the main advantage of using multiple classes is the possibility of adapting the code for different types of simulations, where one might need to make changes to the code. The programming strategy is illustrated in Figure 9.1 and it is based on the optimization technique illustrated in Chapter 8. The strategy consists in having a main function, where the main optimization strategy is implemented, while functions that are not related with the mathematical optimization process are to be kept in other classes, and used if needed.

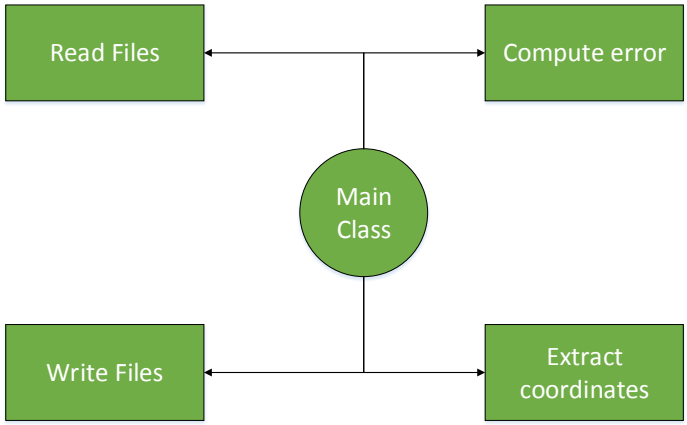

Figure 9.1: Programming strategy used to implement the optimization.

In Figure 9.1 classes are constructed to cope with the reading and writing files, extracting coordinates or compute an error. The reason behind using them as external functions lies in the need to adapt these classes to the different types of files that need to be read or different error computing methods. One implementation based on Appendix 9 needs the following external classes:

- One main class.
- Two classes used for writing files.
- Two classes used for reading files.
- Three classes used to compute the error.

It is worth mentioning that if one wants to use values from another class, one has to use public variables, on which the information is stored, since the information from each class will be deleted at the end of the program. Other classes can be used by implementing secondary methods in a class, and use those methods to call other classes and extract information from them. Example 9.1.2 illustrates a secondary method, used to run a class and extract information out of it. The name of the method is "EFPP", , the information extracted is in the variable "totEFPP" and will be stored in the variable "EffectivePP". The method runs a class for extracting the effective plastic strain from the result files of LS Dyna. This type of methods are used also to create a small computational program or to launch another program, such as LSPP.

example 9.1.2 `Extract the effective plastic strain`

```
public static void EFPP(String[] start){
EFPP.main(start);
 EffectivePP=EFPP.totEFPP;
}
```

9.2 Knowledge needed to develop the computer program

In order to build a similar program one will have to know how to use the following:

- Mathematical operators, such as "+, -, / and *".
- Logical operators, such as "while, if, && or || ".

- Array operations.
- Using different type of data such as string, integer, double etc.

The main challenge is to cope with commands that are different according to the package type it is used. One such package is Common.Math.Appache, which is a free set of instruction based on which certain mathematical operations are performed, such as array operations or having to raise to a power a specific number. Other examples that require more research is how to read files, hot extract data, split it in categories and then handle it in a form of an array or anything that is convenient. There are many packages that can handle this command in different ways, and one has to decide what is the approach that fits better. The packages come with a description of each set of commands. The method used in the project is a method provided by java.io. called "file reader".

9.3 Discussion

The code built for the optimizer it is at beginner level, from the knowledge perspective and from the method used to format certain commands, since there are many ways in which this can be done smarter. Even so, this is only important from a clean aspect of the code and perhaps from the amount of work that has been done to create the code. From the performance perspective of the code, the computational factor does not come into play, since the computers used are very powerful and the simulation time is very high compared with the computational time used by the Java program. The down side of the code is that every time a change needs to be done, there are many changes to do as a consequence, thus requiring an improvement from this perspective.

CHAPTER 10

Complete optimizer

The purpose of this chapter is to introduce the reader to how the optimization is implemented on top of the FEM simulation, and what approaches are taken in order to obtain the desired results. The chapter takes offset in Section 10.1 where results of simulations without running the optimizer are presented in order to understand what are the problems that occur in this type of deep drawing process. Section 10.2 introduces the working principle used for linking the objective function with the simulation, while in Section 10.3 a discussion is made with the focus on the approach taken for coping with the problems presented in Section 10.1

10.1 Combining the optimization software with the FEM simulation

In order to understand how the optimization software is combined with the FEM analysis, a description of the problem has to be given. Thus, the simulation it is being ran for the first time, with zero fous on good results.

In Figure 10.1 and 10.2 the results obtained after running the two simulations are illustrated.

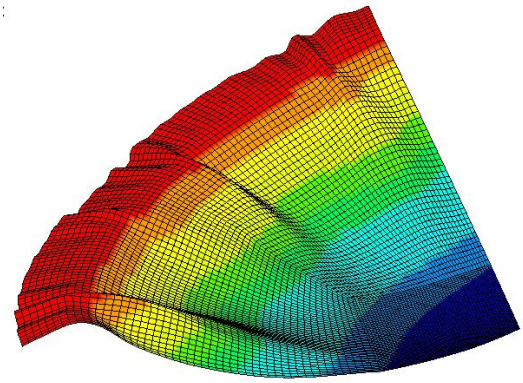

Figure 10.1: Blank deformation for round punch.

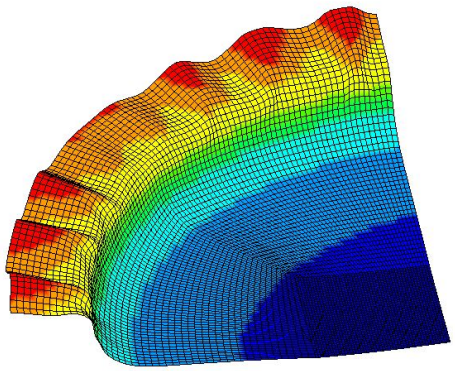

Figure 10.2: Blank deformation for the circular prism punch.

The problems discovered in the beginning are the following:

- Wrinkles on the area between the blank and the die.
- Wrinkles on the deformable area, for the spherical punch.
- High plastic strain.
- Intense thinning.
- Spring back effect.

From the pool of problems that occur, there are certain aspect that are solvable quite easy by reducing the travelling length or increasing the thickness of the punch. These type of problems do not necessary present an particular interest, since the solution is quite obvious, and relatively

simple to solve. Therefore for these problems, the punch travel distance is reduced to an acceptable result from the thinning and the plastic strain perspective.

The simulation will be considered the starting point from which other problems can be solved using the optimisation algorithm, such as the wrinkling, the spring back effect or other problems that might occur. In order to be able to connect the algorithm with the simulation it is important to understand what is there to be optimized, therefore, as explained in Arrora 2004, Chapter 2, the optimization data has to be collected.

Data collection

The shape of the plate is defined in LS Dyna by the coordinates values of the nodes, therefore two sets of nodes are selected and their coordinates are being extracted after the simulation. The nodes are illustrated in Figure 10.3.

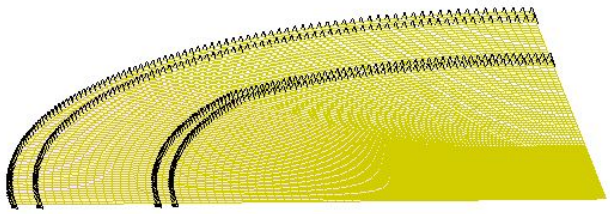

Figure 10.3: The two node sets selected for the optimizer.

As it can be seen the node sets are distributed along a circle with a radius smaller than the radius of the blank. It is important to mention that for better results, one set contains the nodes from two circles. Thus, the first set contains the nodes that are distributed along the two circles with the higher radius value, while set two contains the rest of them. In LS Dyna, the coordinates value for the two nodes are obtained by setting the value for the nodout key in the database section to the simulation time, thus the coordinates are extracted at the end of the simulation, and not during the simulation. Another important step is to include the sets into the history key, since if they are not included, no values will be extracted.

Another data that has to be collected refers to the values for the curve that contains the pressure level on the blank holder during the simulation. In the simulation of the deep drawing process the retraction of the blank holder from the blank it is included, in order to simulate the behaviour of the deformed blank right after the deformation process, where problems such as the spring back effect might occur. The pressure on the blank it is controlled trough a load curve, that defines the value of the pressure at different steps in the simulation. The load curve is illustrated in Figure 10.4. The load curve is defined trough five different points, out of which only four are included in the optimizer. The reason for this is to simulate the last part of the process.

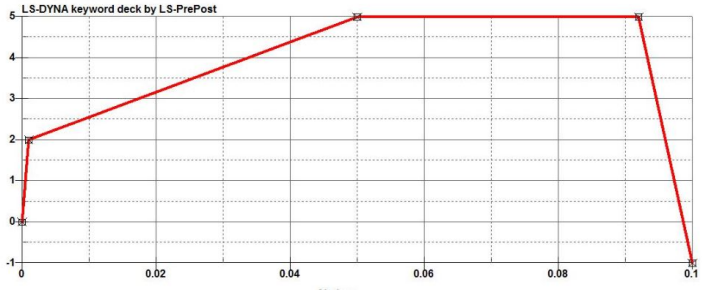

Figure 10.4: The load curve used to define the pressure on the blank holder

Objective function

As stated in Section 8.1 the equation for defining the objective function is specific in this case for the least square problems, where the scope is to minimize the error given trough the objective function. Therefore, the equation is:

$$f(x) = \sum_{j=1}^{m} r_j^2(x) \tag{10.1}$$

$$r_j(x) = \hat{y}_j - y_j \tag{10.2}$$

,where Equation 10.1 defines the sum of the error, while Equation 10.2 defines the error. The error can be computed in different ways, such as the difference between the coordinate of one node in respect to the average value of all the coordinates in the set, or as a distance between the current position of the node and a desired position. The objective function, can be defined as a sum of a sums of errors, as illustrated by Equation 10.3:

$$f(x) = \sum_{j=1}^{m} r_{j1}^2(x) + \sum_{j=1}^{m} r_{j2}^2(x) + ... + \sum_{j=1}^{m} r_{jn}^2(x) \tag{10.3}$$

When summing up multiple errors, one has to pay attention to the need to scale the values, since they can influence the result, based on Matei 2016. For the proposed simulations, in the beginning only the exterior node set is introduced in the objective function, and based on the results obtained, other types of errors will be introduced.

10.2 Working principle

The working principle used is as follows:

- Java is started and the first step is to launch the FEM software.
- The FEM simulation runs once with a set of initial parameters.
- Java launches LSPP to extract other data, such as thickness of the elements.
- The coordinates of the nodes are read and the error is computed.
- Each parameter of the load curve is changed and a new error is computed, thus 4 errors are computed in one iteration.
- The gradient is computed, along with the step size.

- The values for the load curve are changed and the iterative process starts again.

The load curve is defined in a different file, in order to make the reading and changing of its values easier. The model is defined in another file, and they are merged before each simulation of the deep drawing process. This is done using the command : " cat model.k curve.k>sim.k". The command is specific for machines that are running Unix software. The iterative process is stopped when either the function reaches a certain number of iterations or when the value of the gradient length is smaller than an imposed value, usually 0.1 or smaller.

10.3 Discussion

The discussion focuses on the approach taken in implementing the optimization on top of the FEM software. The approach used takes advantage of certain characteristics of the FEM software and LSPP, such as:

- Possibility of adding multiple text files together in order to create a new model, as stated in Chapter 6.
- Using MobaXTerm and Unix allows to run simulations in batch mode.
- Using the command file from LSPP allows repeating a certain action.
- LS Dyna outputs information in a text file, such as nodal coordinates, plastic strain, etc.

In order to reach the desired outcome of the FEM simulations, such as low plastic strain, desired geometry, removing the spring back effect, aspects such as element thickness, nodal coordinates, effective plastic strain and nodal position have been considered. The approach of using a very simple error function as illustrated by Equation 10.1 it is a very simple but effective manner of coping with this type of situations. This approach is rather simple from the perspective of software usage, since there are other techniques in literature that can be used to achieve the same goal, but these techniques require a completely new set-up and due to time limitation there have not been investigated.

CHAPTER 11

Methodology

The purpose of this chapter is to describe the methodology used to perform the analysis of the optimization software and the simulation results. Section 11.1 introduces the method used to analyze the performance of the optimization algorithm, while Section 11.2 illustrates the methods used to analyze the simulation results. Lastly, in Section 11.3 a discussion is given on the methodology used.

11.1 Optimization analysis method

Running the optimization algorithm can prove to be time consuming and one might need to know if the algorithms performs as expected, if the function reached an local optimum and stabilized itself around that point. One example to illustrate this situation is given in Figure 11.1 where a local optimum is illustrated.

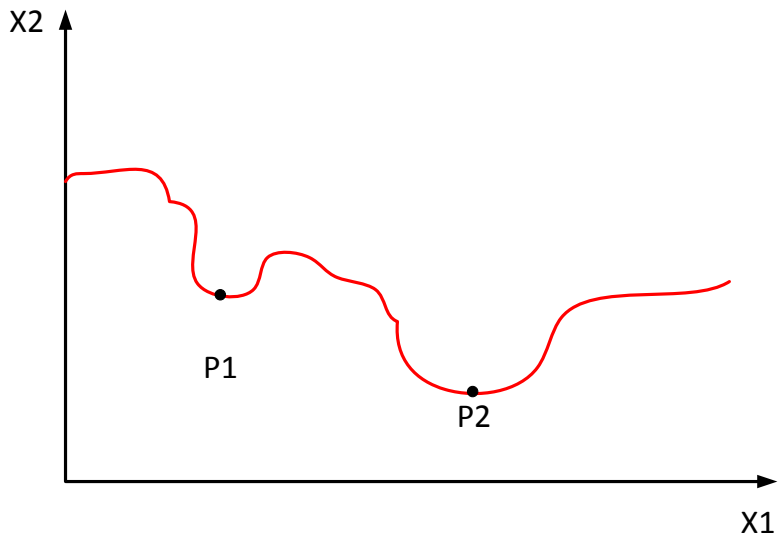

Figure 11.1: Function evaluation, P1 local optimum, P2 global optimum.

One indicator of this type of situation is when the value of the length of the gradient is fluctuating. Therefore one can supervise the value of the gradient, which can be outputted in a log file and once the value fluctuates then it means that the function has either reached an optimum or a local optimum. This is a specific type of problem when the error is computed using weights.

When the optimization has reached a global optimum, an analysis is performed of the performance of the algorithm. The analysis will focus on the convergence rate of the optimizer, i.e. how many iterations needs to converge. Further on, a focus will be placed on how much time it has taken in total to converge and the evolution of key parameters, such as the evolution of the objective function, the parameters that need to be optimized and the individual error value. In Figure 11.2 one can see an example of a chart, where evaluations of the parameters are given, based on Matei 2016.

It is important to state that the evaluation will be done individually for each simulation, in order to get a better understanding of how the algorithm performs for that specific situation, and lastly the results from the different simulations can be compared.

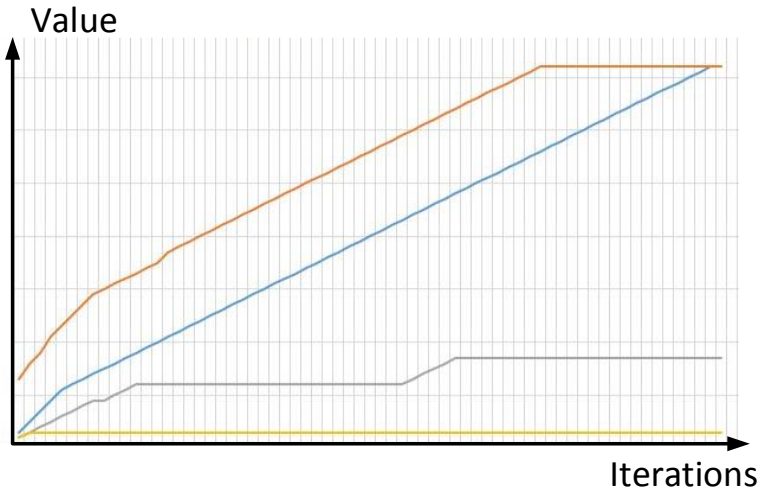

Figure 11.2: Parameters values, extracted after each iteration. The graph is built as values over iteration.

11.2 Simulation analysis method

The simulation analysis will be performed when an optimal it is reached, regardless of a local optimum or a global optimum. For a local optimum, the evaluation will be made at a brief level, taken in consideration only aspects that are relevant to understand if the simulation is feasible or not. These aspects are focusing on geometry matching the desired geometry or effective plastic strain not exceeding the maximum values.

If a simulation matches those criteria, then one can assume that the optimum reached it is also a global optimum. If not, then one has to evaluate how the objective function is defined and change the objective function. Once the optimization reaches the global optimum, then the focus will be placed on more aspects, as following:

- Geometry match.
- Level of effective plastic strain.
- Severe thinning.
- Contact forces between the blank and punch, blank and die and blank holder.
- Element deformation.

The results that will be obtained are compared with the data that is available in the literature for deep drawing processes, since no experimental work is performed.

Figure 11.3 illustrates how the thickness is distributed when a potential local optimum point has been reached. This is usually matched with high values of the objective function. Figure 11.4 illustrates the thickness distribution at a global optimum point. It can be seen that in Figure 11.4 the thickness distribution is even distributed along the blank, while in Figure 11.3 the thickness distribution is uneven, the distribution is mixed and the geometry does not match the imposed parameters.

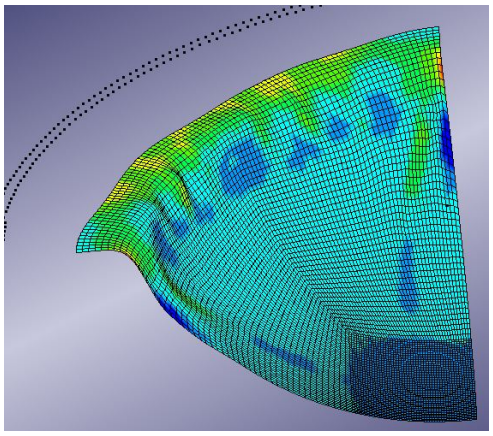

Figure 11.3: Thickness distribution after reaching a local optimum.

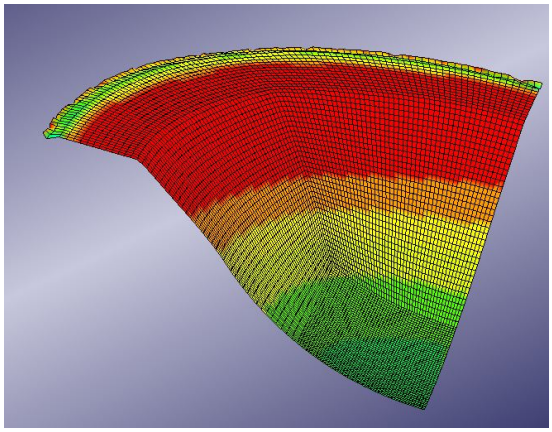

Figure 11.4: Thickness distribution after reaching a global optimum.

In the end, the results from the simulations and optimization process are compared, in order to see if the algorithm performs better on different simulation, and why.

11.3 Discussion

The methodology used in this project illustrates that only a theoretical approach is taken on the subject, while no experiments are used to verify the data. The reasons for this are determined by the reduced time frame, while at the same time the literature provides with a thorough documentation regarding the deep drawing process.

Another aspect is the time frame used for the project, compared with the amount of new tasks that had to be undertaken and the skills that one has to acquire in order to have a solid knowledge in the field. Regarding the results obtained, they will be compared with standard requirements from sheet metal process and as long as they are in between the imposed limits, the focus can be placed more on the optimizer, since it is more important to prove that the optimizer can find a minimum, and that minimum fits reasonable in the expected reality.

The methodology will be respected in a ratio of 90% since, there could be changes that have to be made for one simulation, and not necessary for the other ones.

CHAPTER 12

Results analysis

In this chapter the simulation results will be given, with the purpose of illustrating important aspects of the optimization process, such as convergence rate, how fast it converges. At the same time an emphasis is placed on the FEM results, in respect to the feasibility of the simulated process. Each process that is going to be simulated will be treated individually, due to the fact that a different deep drawing process can output different types of problems, that will need to be coped with.

12.1 Deep drawing process using a half of sphere punch

This simulation has the following specifics:

- The punch diameter, D_p is 360 [mm].
- The blank diameter, D_b is 720 [mm].
- The thickness of the blank is 1 [mm].
- The travel distance of the punch is 180 [mm].
- The error includes a geometrical error and a thickness error.
- The thickness error is computed as the sum of the thickness of the elements.

The pressure values on the blank are set to 1.5 [MPa], one simulation is ran and the error data is extracted in order to evaluate the individual scale of each error. The geometrical error is evaluated at 153, out of which 60 is due to the error on the flange. At the same time, the error from the thickness is evaluated at 643. Investigating the geometry, it has been noticed the level of thickness reduction is low, thus when the pressure is increased, one would expect to have a higher thickness reduction. This implies that the thickness reduction will have a higher weight in the objective function, which will be detrimental to the geometrical error. The geometry contains multiple wrinkling areas, illustrated in Figure 12.1.

In order to cope with this problem, the geometrical error will be placed at the power of 3 and the optimization process is started. Table 12.1 illustrates the values reached when the optimizer reaches a global optimum. In Figure 12.2 the evolution of the design parameters are plotted.

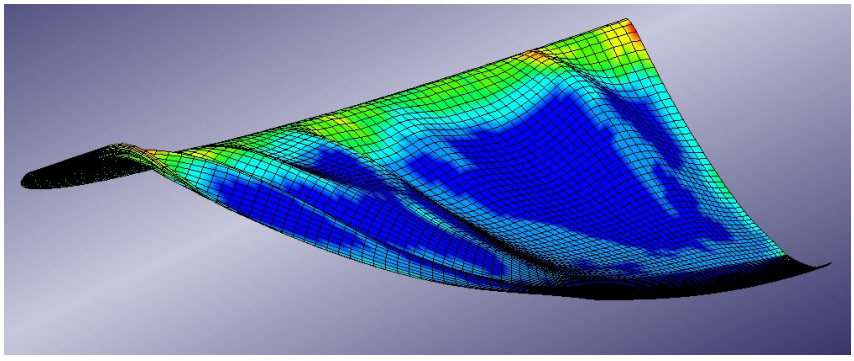

Figure 12.1: Illustration of the geometry and plastic strain distribution along the blank.

Table 12.1: Results from the optimization process

Time [s]	Pressure [Mpa]
0.001	0.2
0.005	10.44
0.0092	21.27
0.01	0.6
Iterations	27
Objective function value	26287
Gradient length	0.01

The iteration process start from the value 1.5 for each design point. The first 20 iterations the values are moving rapidly towards the optimum point. Afterwards, mainly due to the step size, the design points are fluctuating near to the optimum point, but it takes 7 iterations in order to reach the optimum. The values are plotted on two different axes, since they evolute at different scales.

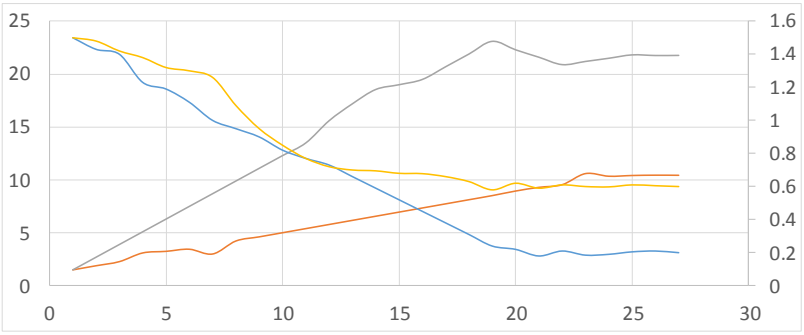

Figure 12.2: Chart illustrating the evolution of the design points. The blue line matches the evolution point 1, red line matches point 2, grey line matches the evolution of point 3 while yellow line the evolution of point 4.

From Table 12.1 it can be seen that the geometrical error, which does not include the error from the flange is 26287. One has to consider that this error is to the power of 3. Thus, when extracting the root, the error is 29.74. This is the cumulated error from 200 nodes. When divided, the error per node is 0.14 which represents 0.14 [mm]. The error points towards the fact that the radius for this points on the blank is $180[mm] \pm 0.14[mm]$. When the optimization process started the error was $180[mm] \pm 0.4[mm]$. Based on the two errors, it can be concluded that the error was reduced at 35% of the initial value. Figure 12.3 illustrates the plastic strain distribution on the blank. The highest level reached is 3.5. Since is a small value, no further investigation is needed. It worth to point out that the areas where the highest levels are reached, are the areas where the boundary conditions are reached, or where the three surfaces that are forming the blank are united trough the nodes. This is a fact that is worth investigated on a future research.

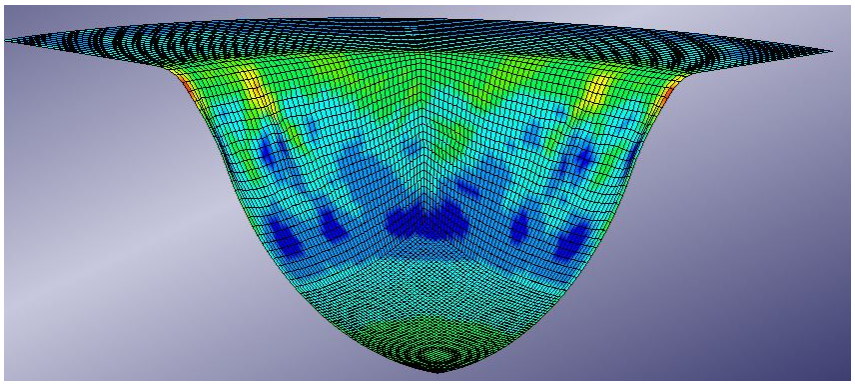

Figure 12.3: Plastic strain distribution along the blank, when reaching a local optimum.

12.2 Deep drawing process using a round prismatic punch

The simulation developed for the round prismatic punch has the following characteristics:

- The punch diameter, D_p, is 520 [mm]
- The blank diameter D_b is 720 [mm]
- The thickness of the blank is 1 [mm]
- The travel distance of the punch is 90 [mm]
- The error includes the geometrical error only.
- The geometrical error is computed only for the flange.

The optimization is ran only once, with a pressure of 1 [MPa] on the blank holder in order to observe the results. The simulation results indicated low values for the plastic strain and a small reduction in the thickness along the surface of the blank. This situation is expected, since the prismatic round punch has the high friction levels around the edges. Further on, a big area from contact surface between the punch and the blank is flat. Based on the first indications, the optimizer is ran only for the flange. The optimization process took 7 iterations to finish, and the results from the optimization process are illustrated in Table 12.2. The next step is to investigate

Table 12.2: Results from the optimization process

Time [s]	Pressure [Mpa]
0.001	-0.02
0.005	1.78
0.0092	2.7
0.01	0
Iterations	7
Objective function value	0.04
Gradient length	0.01

the thickness distribution and the plastic strain. Figure 12.4 and 12.5 illustrate the thickness distribution along the blank and respectively the plastic strain distribution along the blank.

The lowest value for thickness of an element is 0.989 [mm], resulting in a thickness reduction of 10%. The highest value for the plastic strain is 0.44, which is in within the recommended values. The force on the blank holder is extracted and it reaches a total of 12 kN, which is the recommended value for the material.

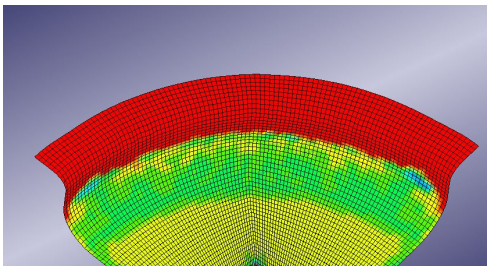

Figure 12.4: Thickness distribution on the blank.

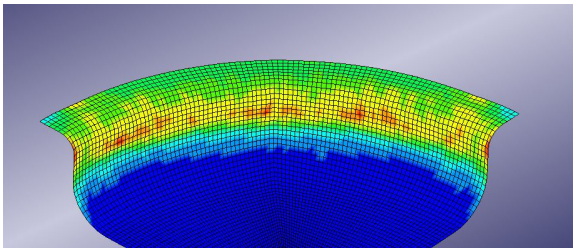

Figure 12.5: Plastic strain distribution along the blank.

One has to notice that the properties based on which the highest force admissible is determined for a mild steel. This property can be changed, trough certain processes a that can be undertaken to improve the steels mechanical properties, or by reducing the friction between the die, blank holder and the blank. The force between the blank holder and the blank reaches a peak at 13 kN, which is a positive result, since here a higher force reduces the strains from the blank.

At this point the optimization process and the simulation results combined, point to the fact that the results are within the specified limits.

12.3 Deep drawing using a larger spherical punch

Since both of the first simulations have been providing with positive results after running the optimizer, another approach that is taken is to increase the size of the spherical. The simulation is ran as it follows:

- The punch diameter, D_p is 360 [mm].
- The blank diameter, D_b is 720 [mm].
- The thickness of the blank is 1 [mm].
- The travel distance of the punch is 180 [mm].
- The error includes the geometrical error only.

The simulation is ran once in order to evaluate the error level that is encountered, as in previous situations. The error level encountered is 247 in regards to the geometrical error, while the thickness error is evaluated at 937. This is mainly to the increased surface that is going to be deformed and due to the lower punch travel, which allows to form wrinkles. The geometrical error is used to the power of 3, and the thickness error is used as previously. The optimizer reaches a an optimum for the values illustrated in Table 12.3

Table 12.3: Results from the optimization process

Time [s]	Pressure [Mpa]
0.001	3
0.005	18
0.0092	38
0.01	10
Iterations	58
Objective function value	26873
Gradient length	0.01

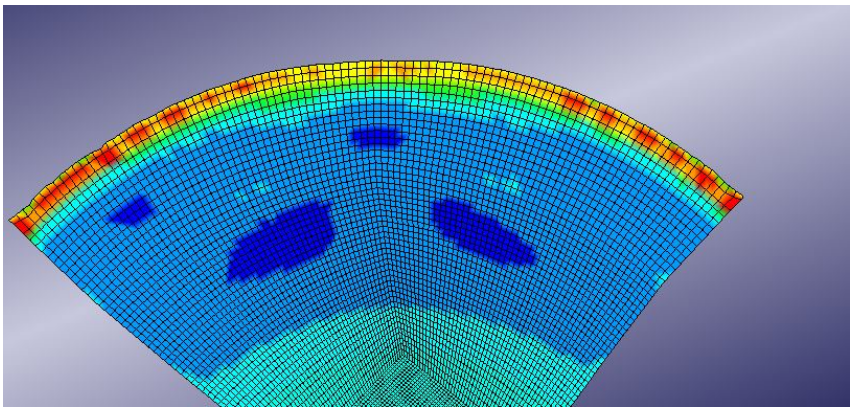

Figure 12.6: Plastic strain on the flange, after the optimum is reached

The main problem in this particular case is that due to the geometry, high values are obtained for the plastic strain. Failure in the material can be seen along the flange, where the plastic strain is 0.7 in the elements close to the edge of the flange, as in Figure 12.6. The plastic strain is then reduced to values from 4.9 on the rest of the flange until 5.2 on the tip of the blank. It can be concluded that the optimizer has performed as expected but the FEM model, in respect to the geometry is not feasible.

Implementation in tube hydro-forming

In this chapter, an example is given on how the optimization algorithm performs, implementation beeing made on a geometry that has been provided with by the Department of Mechanical and Manufacturing engineering. Section 13.1 introduces the geometry used, the objective function and the design parameters, while Section 13.2 describes the results obtained by the optimizer. The chapter ends with a discussion on the optimizer performance and the conclusions drawn based on this performance. It is important to mention that the data presented is based on Matei 2016, a paper written for the purposes of this project.

13.1 Optimizer implementation

The optimization is implemented on the geometry illustrated in Figure 13.1, where the goal is to benefit from the squeezing effects and the bulging effects of the process, as illustrated in Figure 13.2. The process involves a tube that has pressure applied on the inside, in order to deform it, while at the same time, the tube is being squeezed by two moving tools. This process, should improve the final product from multiple quality aspects, such as material tearing, effective plastic strain, geometry fitting the required geometry etc. The design parameters are defined by three load curves, used to control the pressure applied on the tube, and the movement of the two moving tools that are squeezing the tube. The feeder is associated with V_1 in above mentioned figure, and the punch is associated with $V2$ in the same figure. There are several aspects to consider here, before the implementation is made, such as the fact that the movement of the punch is predetermined, thus, the punch can move faster or slower in certain time intervals but at the end it will have to reach the same ending point. The pressure values and the movement values, based on the calculations made have values on different scales. The reason for this is that the unit consistency system demands in this case that the velocity should be in [mm/s] while the pressure in [MPa]. Since the velocity has to be around 6000 [mm/s] due to the short simulation time, which is 0.01[s], the pressure cannot reach such high values without destroying the material.

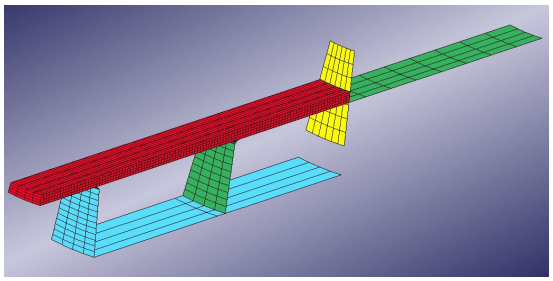

Figure 13.1: Illustration of the geometry used for the tube hydro forming process.

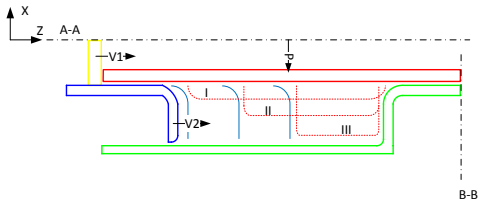

Figure 13.2: Illustration of the tube hydro forming process.

Objective function

The objective function is defined here in multiple ways, in order to test the optimizer but also to overcome certain problems that might occur during the process. The objective function is defined as a difference between the nodal coordinates after each simulation and the nodal coordinates of

the wanted geometry, which are represented by coordinates of key points on the punch or the mold. This is illustrated in Figure 13.3, where the error is represented by the red dotted lines. The tube is illustrated in its initial position by the red rectangle, and in a possible position by the dotted black line.

To this objective function certain approaches are taken to overcome a to high effective plastic strain, severe rupturing or getting the function caught up in local optimum points. The effective plastic strain has been incorporated in the function, as a simple summation. The difference between the initial thickness and the one at the end of the simulation has been incorporated as well. It is important to mention that the objective function, no matter how is defined, focuses on a set of elements or nodes, that are symmetrically placed on the geometry, in respect to the main axis of the tube.

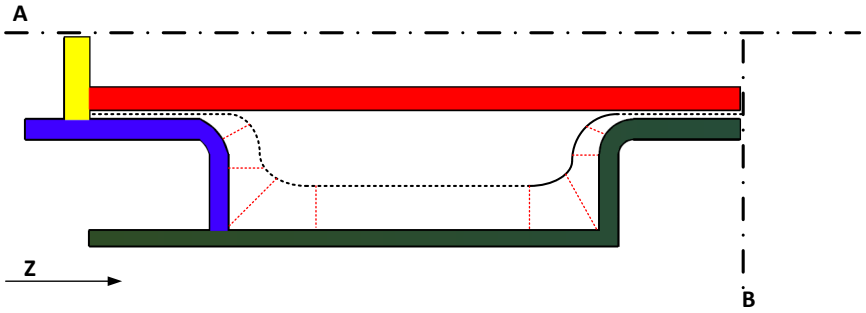

Figure 13.3: Error computation in the hydro forming process

13.2 Results

The results are presented as an illustration and they will not be as detailed as they are in the paper. In the first optimization process ran the design parameters contain all the values versus time points that are defined for the pressure and the two moving parts. The optimizer has been ran with the geometrical error only and with the plastic strain and the thickness error afterwards. In the first case, the optimizer has reached on optimum after 70 iterations, while in the second case the function got stopped in a local minimum. The results of both simulations are illustrated in Figure 13.4 and Figure 13.5.

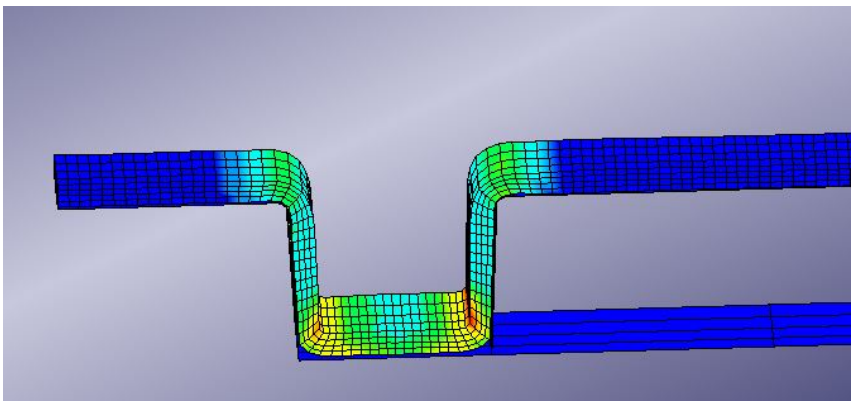

Figure 13.4: Distribution of plastic strain at the end of the optimization process.

In the first simulation the geometrical error is very easy to overcome but there are problems with the quality of the product. Although the geometry condition is being fulfilled, the plastic

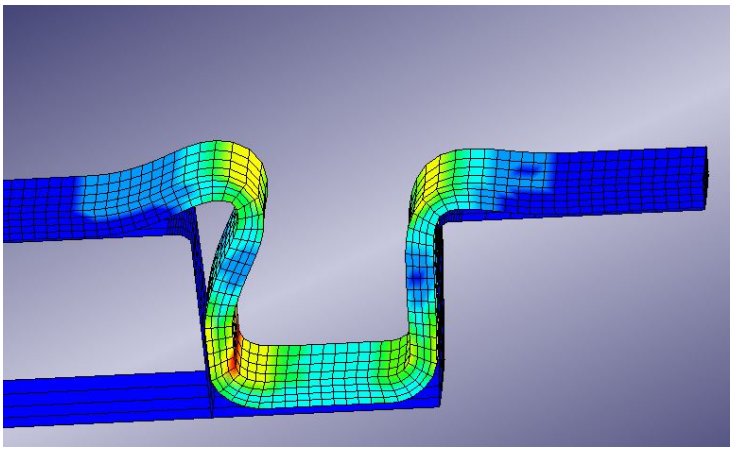

Figure 13.5: Distribution of the effective plastic strain at the end of the second optimization process.

Table 13.1: Values for the design points at different time in the simulation

Time [s]	Pressure[Mpa]	Pressure [Mpa]
0.001	22	25
0.0033	102	34
0.0066	66	24
0.01	131	64
Time [s]	Velocity [mm/s]	Velocity [mm/s]
0.001	11640	9557
0.0033	11651	9567
0.0044	8645	7565
0.0066	1653	1571
0.01	5	1573
Time [s]	Velocity [mm/s]	Velocity [mm/s]
0.001	10078	11002
0.0033	9072	7992
0.0044	8870	7790
0.0066	8872	7791
0.01	4	4541
Iterations	70	123

strain level is very high, the maximum being at 1.2. This implies that the tube will be destroyed in the process. The second simulation is feasible from the strain and stress perspective but the geometry condition is not fulfilled. For the both cases there are different explanations for these specific situations.

Table 13.1 illustrates the results after the optimization process is done. In the simulation where only the geometrical error is considered the optimizer finds that the best approach to reach the desired geometry is by increasing the pressure earlier in the simulation, but this is an action that leads to having a result that is not even feasible, with high plastic strain. The second approach, where the effective plastic strain or the thickness is included, it has been noticed that more often that not the simulation will reach a local minimum. This is mainly because there is a point in the optimization process where the plastic strain error becomes higher than the geometrical error. Combining this with the effect that has been obtained from the geometrical error, results in a local minimum from which the function cannot stabilize or pass it. In order to cope with this problem it is proposed that in the beginning of the simulation, the values for the

pressure are imposed and cannot be changed. Trough trial and error, and due to the experience gained, the pressure is set to 1 [MPa] until the simulation reaches 0.0044 [s]. After making this decision, the optimizer improved its performance and also the results obtained. In Figure 13.6 and 13.7 the distribution of the plastic strain is illustrated. The results are illustrated in Table 13.2. It can be seen that the iteration process reaches the optimum faster. Using the

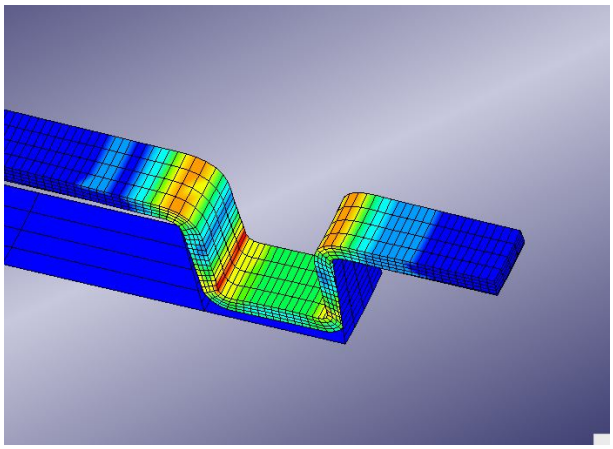

Figure 13.6: Distribution of plastic strain at the end of the first optimization process.

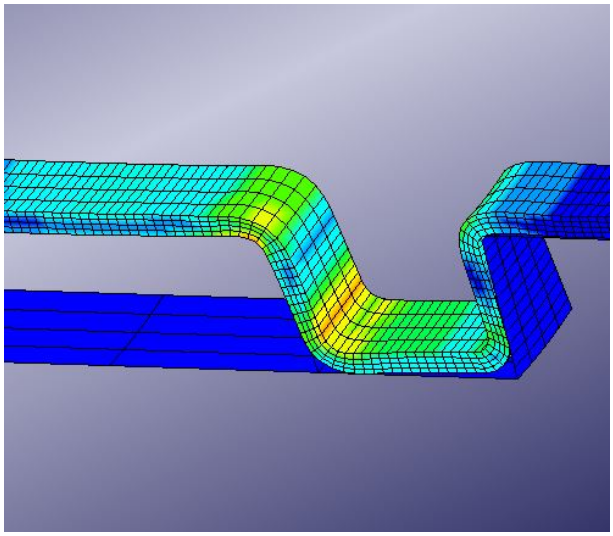

Figure 13.7: Distribution of the plastic strain at the end of the second optimization process.

geometrical error the optimizer reaches an optimum at 67 iterations while the optimizer that uses the effective plastic strain reaches an optimum after 79 iterations. From the effective plastic strain perspective, the maximum reached is 0.61 and 0.57 in both optimizations. The level of plastic strain is higher an only one line of elements, which is placed on the geometry radius, which is expected. It can be assumed that at this point the results from the optimization process are feasible, and one can draw the conclusions.

Table 13.2: Values for the design points at different time in the simulation

Time [s]	Pressure[Mpa]	Pressure [Mpa]
0.001	1	1
0.0033	1	1
0.0044	89	34
0.0066	66	77
0.01	186	148
Time [s]	Velocity [mm/s]	Velocity [mm/s]
0.001	11640	9557
0.0033	11651	9567
0.0044	8645	7565
0.0066	1653	1571
0.01	5	1573
Time [s]	Velocity [mm/s]	Velocity [mm/s]
0.001	10078	11002
0.0033	9072	11992
0.0044	8870	8790
0.0066	8872	6791
0.01	4	0
Iterations	67	79

13.3 Discussion

The reason for illustrating this example is to show how important is to understand the process when using a function that has a high number of design parameters. Further on, the example is used to illustrate the optimizer can prove to be a very powerful tool, but not on its own. Therefore one has to analyse the process and to match the optimizer behaviour with the simulation results. A few conclusions are drawn from Matei 2016, as following:

- Simulation results have to explain the optimizer results.
- When working with a high number of design parameters, it can prove effective to constrain some.
- When using multiple functions incorporated in the error function, the optimization process has to be supervised, in order to avoid local minimum.

CHAPTER 14

Future work

In this chapter to focus will be placed on the future work, that can be carried out with the focus of improving the level of reliability on the optimizer, simulation and the implementation made. The perspective taken is based on the need to have an optimizer with a higher performance, a simulation that can provide with accurate results and a methodology of implementing the two software.

Simulation

From the perspective of improving the simulation one good suggestion is to work with material models that have been proven to match the behaviour of real life materials. This is very important in order to understand how much the simulation matches the real life behaviour. The level of discretization implemented in the simulation process might play a role in the results, such as improving simulation time or improving the results that are obtained at the end of each iteration.

In order to improve the simulation time it is recommended to use mass scaling. This technique implies faster computational time but at the same time the results will be distorted, this being one of the reason for which it has not been implemented in this project, but a thorough investigation can point out if it is worth using mass scaling. One of the main concerns in this area is the fact that the results obtained might be detrimental since it might require to cope with new problems.

One perspective that has been taken is to remove the need to scale the geometry in case that is needed, thus one might use the command files from LS Dyna to change the geometry but keep the same nodal ID. In this way one can manipulate the geometrical values and the software will automatically generate new simulations.

Optimization

The optimization process as stated in Chapter 8 is a basic one, but as proven by the tasks used in the project it can still be effective. There are multiple methods that one can use to improve the performance of the process, and this are definitely aspects that are worth considering.

The first step that should be taken is to improve the step size determination, since it can play such an important role in the number of the iteration process . An ideal situation is when the steps taken are big when the function is far from the optimum and small when it is close to the optimum.

When working with functions that are highly non-linear or with weights, in this report, the method was more close to a trail and error process. It is worth investigating how can this method be included in the optimization process so the user will not need to change the weights itself or to intervene when the function reaches a local optimum.

One other aspect that has not been used in the optimization process is using the second order information in order to improve the accuracy of the optimizer. This is worth investigating, especially for highly no linear functions where the second order information can play an important role, due to the difficulty of reaching an optimum in these type of functions.

Implementation

In the implementation process there are a few things that need to be handled in respect with the desire of making this kind of tool to be accessible to the industry.

First a GUI has to be developed in order for enhancing a user friendly method of using the software. Second, the implementation in Java Oracle is using a basic knowledge that allows to make changes but with a high work volume in certain situations. Therefore one recommendation is to rewrite the code in order to reduce the work volume changes in respect with controlling the design parameters, the functions that are used to compute the error.

One other aspect that is very important is the situation when the simulation process ends with a wrongful simulation or with very distorted results. These type of situations have been encountered quite often and they can prove to be difficult to avoid. The recommendation is to develop the optimizer in such a way that it can interpret the output from LS Dyna and make changes according to it. For example one way of doing it is to verify if the parameters are in a specific range in respect one to another or if the LS Dyna output indicates a normal termination.

14.1 Application of the optimization in industry

There are many applications that can arise in the deep drawing industry. There are many deep drawing techniques used nowadays, in which no blank holder is used. In these type of application, it is important to determine the force of the blank.

Other applications might involve hydro forming process, and it can be either by deforming the blank using water pressure, or by introducing a liquid between the blank and the die in order to reduce the friction as much as possible. In this type of application, an optimizer is essential since the functions used are complex, with a high number of design parameters.

The features that LS Dyna offers are very much in favour of creating an optimizer that can focus on changing material properties, changing geometries or adjusting contact forces. This can all be done by taking advantage of the command file and manipulate the information with the optimizer in respect with the desired topic.

CHAPTER 15

Discussion

The purpose of this chapter is to focus on major findings in this project, to discuss them and point out why they are important. The focus will be placed as well on the explanations for the findings, considering different alternatives and how relevant they are. Lastly, the limitations of the project are acknowledged.

The major findings

The major findings in this project are to be considered from different perspective, since the project takes into account multiple disciplines such as mechanical and mathematical, translated trough the FEM simulation and respectively the optimization process. From the FEM perspective it can be seen as illustrated in Chapter 12 that following the recommendations from dedicated literature one can obtain results that are feasible and match the criteria imposed, such as avoiding severe thinning and reaching desired geometrical requirements.

Although the simulation can be improved from different points of view, and they will be discussed later in the chapter, even with a basic description of the material, contact type and element control one can obtain results that are feasible. This aspect is important because it can be considered a strong base for a future development of the project, giving a thorough understanding of how important is to focus on the behavior of the FEM software.

The optimization technique used points to the fact that such an optimizer can become a very powerful tool, and it can be used in multiple fields. The fact that the optimizer was stable , and managed to reach an optimum on different geometry shows the flexibility that this type of algorithms can provide. Another major fact is the focus on functions that are highly non linear, since they can yield in a local optimum. The optimization process used in the project can be described as slow, thus proving that the basic techniques are feasible but not desirable, thus, moving towards more advanced methods will be required in order to make such an optimizer a reliable one.

Relevancy

The optimization software has provided with results that can be considered relevant, from the mathematics point of view. It is important to understand that the relevancy of the results of the overall process, are highly dependent on the accuracy of the optimizer and the performance of the FEM results. The optimizer has found values for the design parameters that are indicating that the function values is very close to the value of the global minimum point.

The results in the simulation are feasible but they have not been backed up by any experimental work, and based on this the results can be considered relevant only in this frame of work. If one wants to apply the optimizer for a real deep drawing process, it can do so by improving the FEM simulation. As long as the optimizer can find a minimum, then the FEM simulation can be changed at any time.

Limitations

There are several limitation that have to be addressed, based on the optimization performance and the FEM simulation. As specified before the FEM results are feasible, but they are not backed by experimentation work, making hard to prove that the FEM results are going to match a real deep drawing process.

The optimization process can be described as feasible, given the amount of different simulations that has to work with. The most important aspect that limits the optimizer is not the fact that it would yield an optimum or not, but the flexibility and the computational time that is required to reach that global optimum. This mainly due to the basic method used, with no other implementations in respect to the step size determination or the level of information used when approximating the function in order to determine the size of the gradient.

CHAPTER 16

Conclusion

In Chapter 4 the following problem statement has been given:

Develop an optimizer capable of handling highly nonlinear functions, which can be implemented with a variety of geometry, in order to result in determining the global optimum based on the data obtained from a stable FEM simulation process.

Chapter 6 illustrates the development of the FEM simulation, with the results illustrated in Chapter 12. It can be stated that all the simulations used in the project have been stable, providing with consistent results.

The optimizer has been developed in Chapter 9 based on Chapter 8, and implemented in Chapter 10. Chapter 12 illustrates the capability of the optimizer to determine the global optimum.

It can be concluded that trough this project, based on Chapter 12 and 15 that the goals from the problem statement in Chapter 4 have been accomplished.

Bibliography

Arrora, J. S. (2004). *Introduction to Optimum Design* (cited on pages 25, 29, 36).

Boeing (2016). *Boenig production rate in the foruth quarter of 2016*. URL: http://boeing.mediaroom.com/2016-01-21-Boeing-to-Reduce-747-Production-Rate-Recognize-Fourth-Quarter-Charge (cited on page 4).

Cook, M. P̃. (2002). *Concepts and Applications of Finite Element Analysis, 4th Edition* (cited on page 19).

Endelt, B. (2016). 'Gradient based optimization scheme'. In: (cited on pages 25, 27–29, 63).

LSTC (2016). *LS Dyna Manual Support*. URL: http://www.dynasupport.com/manuals (cited on pages 16–18).

Matei, K. B. N. P. A. (2016). *Hydro-forming process for tube production* (cited on pages 9, 37, 39, 49, 53, 63).

Oracle, J. (2016). *Java Oracle Online Essential*. URL: https://docs.oracle.com/javase/tutorial/essential/io/file.html (cited on page 31).

Plesha, M. (2011). *Engineering Mechanics* (cited on pages 6, 7).

APPENDIX A

Enclosed files

A list of the enclosed files is provided in this chapter. The files enclosed are to be found on a CD as well, since the report will be printed with a hard copy.

- LS Dyna key files.
- Complete programming code in Java Oracle, multiple files.
- Matei 2016, a study ran in parallel with the report.
- LS PrePost cook book for creating an FEM model in LSPP.
- FEM result files for simulations.
- Endelt 2016 Optimization article.

Key file used in LS Dyna

This chapter contains the complete key files used in LS Dyna for running the simulation, the command file for the preprocessor and a set of units consistency table. There are three key files, one containing the complete model, and one containing the load curve and the last one contains the command file used for controlling the pre processor.

LS Dyna main key file

```
$# LS-DYNA Keyword file created by LS-PrePost 3.1 - 23Jan2012(09:04)
$# Created on May-24-2016 (11:52:31)
*KEYWORD MEMORY=56000000 NCPU=8
*TITLE
$# title
LS-DYNA keyword deck by LS-PrePost
*CONTROL_ENERGY
$#    hgen      rwen    slnten     rylen
         2         2         2         1
*CONTROL_HOURGLASS
$#     ihq        qh
         1  0.100000
*CONTROL_SHELL
$#  wrpang     esort     irnxx    istupd    theory       bwc     miter     proj
 20.000000         0        -1         1         2         2         1        0
$# rotascl    intgrd    lamsht    cstyp6    tshell    nfail1    nfail4  psnfail
  1.000000         0         0         1         0         0         0        0
$# psstupd    irquad     cntco
         0         0         1
*CONTROL_TERMINATION
$#  endtim    endcyc     dtmin    endeng    endmas
  0.010000         0     0.000     0.000     0.000
*DATABASE_NODOUT
$#      dt    binary      lcur     ioopt      dthf     binhf
  0.010000         0         0         1     0.000         0
*DATABASE_RCFORC
$#      dt    binary      lcur     ioopt
 1.0000E-4         0         0         1
*DATABASE_BINARY_D3PLOT
$#      dt      lcdt      beam     npltc    psetid
  0.001000         0         0         0         0
$#   ioopt
         0
*DATABASE_HISTORY_NODE_SET
$#     id1       id2       id3       id4       id5       id6       id7       id8
        26        27         0         0         0         0         0         0
*BOUNDARY_PRESCRIBED_MOTION_RIGID
```

```
$#     pid       dof       vad      lcid        sf       vid     death     birth
    700001         3         2         2-180.00000         01.0000E+28     0.000
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
        19         0         1         1         0         1         1         1
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
        23         0         1         0         0         1         1         1
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
        24         0         0         1         0         1         1         1
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
        25         0         1         1         0         1         1         1
*LOAD_SHELL_SET_ID
$#      id                                                                heading
         1
$#    esid      lcid        sf        at
         1         1  1.000000     0.000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE
$#     cid                                                                title
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
    400001    500001         3         3         0         0         0         0
$#      fs        fd        dc        vc       vdc    penchk        bt        dt
     0.000     0.000     0.000     0.000     0.000         0     0.0001.000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000  1.000000 1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                                title
         2
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
    400002    500001         3         3         0         0         0         0
$#      fs        fd        dc        vc       vdc    penchk        bt        dt
     0.000     0.000     0.000     0.000     0.000         0     0.0001.000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000  1.000000  1.00000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                                title
         3
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
    400003    500001         3         3         0         0         0         0
$#      fs        fd        dc        vc       vdc    penchk        bt        dt
     0.000     0.000     0.000     0.000     0.000         0     0.001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt       fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000  1.00000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                                title
         4
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid       spr       mpr
    400003    600001         3         3         0         0         0         0
$#      fs        fd        dc        vc       vdc    penchk        bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
```

```
$#     sfs       sfm       sst       mst      sfst      sfmt      fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                              title
         5
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid      spr       mpr
    400002    600001         3         3         0         0        0         0
$#      fs        fd        dc        vc       vdc    penchk       bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt      fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                              title
         6
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid      spr       mpr
    400001    600001         3         3         0         0        0         0
$#      fs        fd        dc        vc       vdc    penchk       bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt     fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                              title
         7
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid      spr       mpr
    400001    700001         3         3         0         0        0         0
$#      fs        fd        dc        vc       vdc    penchk       bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt      fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                              title
         8
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid      spr       mpr
    400002    700001         3         3         0         0        0         0
$#      fs        fd        dc        vc       vdc    penchk       bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt      fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*CONTACT_AUTOMATIC_ONE_WAY_SURFACE_TO_SURFACE_ID
$#     cid                                                              title
         9
$#    ssid      msid     sstyp     mstyp    sboxid    mboxid      spr       mpr
    400003    700001         3         3         0         0        0         0
$#      fs        fd        dc        vc       vdc    penchk       bt        dt
     0.000     0.000     0.000     0.000     0.000         0    0.0001.0000E+20
$#     sfs       sfm       sst       mst      sfst      sfmt      fsf       vsf
  1.000000  1.000000     0.000     0.000  1.000000  1.000000 1.000000  1.000000
*PART
$# title
LSHELL1
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
    400001         4         3         0         0         0        0         0
```

```
*SECTION_SHELL
$#   secid    elform      shrf       nip     propt   qr/irid    icomp     setyp
         4         2  1.000000         2         1         0        0         1
$#      t1        t2        t3        t4      nloc     marea     idof    edgset
  1.000000  1.000000  1.000000  1.000000 -1.000000     0.000    0.000         0
*MAT_POWER_LAW_PLASTICITY
$#     mid        ro         e        pr         k         n      src       srp
         3 7.8900E-9 2.0000E+5  0.300000 550.00000  0.250000    0.000     0.000
$#    sigy        vp
     0.000     0.000
*PART
$# title
LSHELL2
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
    400002         4         3         0         0         0        0         0
*PART
$# title
LSHELL3
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
    400003         3         3         0         0         0        0         0
*SECTION_SHELL
$#   secid    elform      shrf       nip     propt   qr/irid    icomp     setyp
         3         2  1.000000         2         1         0        0         1
$#      t1        t2        t3        t4      nloc     marea     idof    edgset
  1.000000  1.000000  1.000000  1.000000  1.000000     0.000    0.000         0
*PART
$# title
LSHELL1
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
    500001         1         1         0         0         0        0         0
*SECTION_SHELL
$#   secid    elform      shrf       nip     propt   qr/irid    icomp     setyp
         1         2  1.000000         2         1         0        0         1
$#      t1        t2        t3        t4      nloc     marea     idof    edgset
  1.000000  1.000000  1.000000  1.000000     0.000     0.000    0.000         0
*MAT_RIGID_TITLE
Rigid
$#     mid        ro         e        pr         n    couple         m     alias
         1 7.8700E-9 2.0000E+5  0.300000     0.000     0.000    0.000
$#     cmo      con1      con2
     0.000         0         0
$# lco or a1      a2        a3        v1        v2        v3
     0.000     0.000     0.000     0.000     0.000     0.000
*PART
$# title
LSHELL1
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
    600001         1         1         0         0         0        0         0
*PART
$# title
LSHELL1
$#     pid     secid       mid     eosid      hgid      grav   adpopt      tmid
```

```
    700001         1         1         0         0         0         0         0
*DEFINE_CURVE
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp
         2         0  1.000000  1.000000     0.000     0.000         0
$#              a1                  o1
           0.000          -0.0100000
       0.0091000           1.0000000
       0.0100000               0.000
*DEFINE_CURVE
$#    lcid      sidr       sfa       sfo      offa      offo    dattyp
         3         0  1.000000  1.000000     0.000     0.000         0
$#              a1                  o1
           0.000               0.000
       0.0010000           0.5000000
       0.0500000           1.0000000
```

Key file for the load curve

```
*DEFINE_CURVE
$#    lcid      sidr       sfa       sfo      offa      offo     dattyp
         1         0  1.000000  1.000000     0.000     0.000          0
$#              a1                  o1
       0.0010000                5.35
       0.0050000                2.13
       0.0092000                0.56
       0.0100000                 0.6
*END
```

The command file used to control LS PrePost

```
* LS-PrePost Command file, Created on May-24-2016 (14:27:06)
* Created by LS-PREPOST 2.4 - 15April2010(10:30)
**
$# LS-PrePost command file created by LS-PREPOST 2.4 - 15April2010(10:30)
$# Created on May-24-2016 (14:27:06)
openc d3plot "d3plot"
selectpart on S400001/0
selectpart on S400003/0
selectpart on S400001/0 S400002/0 S400003/0
fringe 67
pfringe
state -1;
output append 1
output /home/alex/Test/thick 12 1 0 1 0 0 0 0 1 0 0 0 0 0 0
exit
```

Table B.1: LS DYna unit consistency sistem

MASS	LENGTH	TIME	FORCE	STRESS	ENERGY	DENSITY	YOUNG's
kg	m	s	N	Pa	J	7.83E+03	2.07E+11
kg	cm	s	1.0e-02 N			7.83E-03	2.07E+09
kg	cm	ms	1.0e+04 N			7.83E-03	2.07E+03
kg	cm	us	1.0e+10 N			7.83E-03	2.07E-03
kg	mm	ms	kN	GPa	kN-mm	7.83E-06	2.07E+02
g	cm	s	dyne	$dyne/cm^2$	erg	7.83E+00	2.07E+12
g	cm	us	1.0e+07 N	Mbar	1.0e+07 Ncm	7.83E+00	2.07E+00
g	mm	s	1.0e-06 N	Pa		7.83E-03	2.07E+11
g	mm	ms	N	MPa	N-mm	7.83E-03	2.07E+05
ton	mm	s	N	MPa	N-mm	7.83E-09	2.07E+05
lbf-s^2/in	in	s	lbf	psi	lbf-in	7.33E-04	3.00E+07
slug	ft	s	lbf	psf	lbf-ft	1.52E+01	4.32E+09
$kgf - s^2/mm$	mm	s	kgf	kgf/mm^2	kgf-mm	7.98E-10	2.11E+04
kg	mm	s	mN	1.0e+03 Pa		7.83E-06	2.07E+08
g	cm	ms	1.0e+1 N				
1.0e+05 Pa		7.83E+00	2.07E+06		9.81E-04		

APPENDIX C

Java Code

. This chapter contains a series of Classes that are important in order to run the optimizer. The complete optimizer will be enclosed. The implementation is not explained, but comments regarding certain actions in the optimizer are given within the code, they are marked by //.

C.0.1 Java code for the main class

```
import java.io.File;
import java.io.IOException;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;

public class Main_function {

public static ArrayList designPoints = new ArrayList();
public static ArrayList gradientSS = new ArrayList();
public static double[] gradientM = new double[1];
public static double[] gradient1 = new double[4];
public static double[] gradient2 = new double[4];
public static double[] gradient3 = new double[4];
public static double[] dx = new double[15];
public static double EffectivePP;
public static double ref1z;
public static double ref2z;
public static double ref5z;
public static double ref4z;
public static double ref2x;
public static double ref1x;
public static double ref5x;
public static double ref4x;
public static double ref3;
public static double gradientLength = 1;
public static double highestError;
public static double function = 0;
public static double function3 = 0;
public static double function2;
public static double ThicknessError;
public static double inc;
public static double gradient;
public static float value;
public static double value2;
public static int i;
public static int tri;
public static double grad1;
```

```
public static double alfa = 0;

public static double p = 0.1;
public static int counter = 0;
public static double epsilon = 0.01;
public static double f0 = 0;
public static double f0a = 0;
public static double f1 = 0;
public static int p2;
public static double l = 60;
public static int SimStat = 0;
public static double thickSum;
public static String run = "start";

public static void main(String[] args) throws IOException {
while (!(counter > 100) && !(gradientLength < epsilon)) {
// run dyna and get first value for the function f0, which is the
// sum of the error for each node

String[] k = { "403116"..... "400055" };
// Initialize
String[] k2 = { "400712"....."400085" };

System.out.println("counter is" + counter);
mergeKfiles();
try {
Thread.sleep(3000);
System.out.println("waiting");
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
igniteDyna2();
try {
Thread.sleep(3000);
System.out.println("waiting");
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

String k1;
String k12;
f0 = 0;
for (int p1 = 0; p1 < k.length; p1++) {
k1 = k[p1];
read(k1);

f0 = (function + f0);

}
f0a = 0;
```

```
for (int p1 = 0; p1 < k2.length; p1++) {
k12 = k2[p1];
read2(k12);

f0a = (function2 + f0a);

}
System.out.println("error length is" + f0a);
double average2 = f0a / k2.length;

double average = f0 / k.length;

f0 = 0;
f0a = 0;
for (int p1 = 0; p1 < k.length; p1++) {
k1 = k[p1];
read(k1);

function = function - average2;

function = function * function;
f0 = (function + f0);
}
for (int p1 = 0; p1 < k2.length; p1++) {
k12 = k2[p1];
read2(k12);

function2 = function2 - average;

function2 = function2;
f0a = (function2 + f0a);

}
System.out.println("error length is" + f0a);
thickness();
thikExtract("as");
f0 = f0 + f0a + thickSum;
inc = 0;
curves(inc);
String Status = k[0];
read(Status);
String[] start = null;

System.out.println("f0, function+f0 is" + f0);

// ================================================
// Change individual value for the design points
double gradientLengthSq = 0;

for (i = 0; i < designPoints.size(); i++) {
try {
Float value = Float.valueOf((Float) designPoints.get(i));
```

```
value2 = value;
} catch (NumberFormatException ex) {
System.err.println("err49Main");
}
if (i < designPoints.size() - 1) {
value2 = (value2 + 1.2);

increment(value2, i);
// ================================================================
// Run dyna and get individual function value
mergeKfiles2();
try {
System.out.println("waiting");
Thread.sleep(3000);

} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
igniteDyna2();
try {
System.out.println("waiting");
Thread.sleep(3000);

} catch (InterruptedException e) {

e.printStackTrace();
}

read(Status);
if (SimStat != 0) {
value2 = (value2 - 1.2);

increment(value2, i);
// ================================================================
// Run dyna and get individual function value

mergeKfiles2();
try {
System.out.println("waiting");
Thread.sleep(3000);

} catch (InterruptedException e) {

e.printStackTrace();
}
igniteDyna2();
try {
System.out.println("waiting");
Thread.sleep(3000);

} catch (InterruptedException e) {
```

```
e.printStackTrace();
}

}
}

f1 = 0;
for (int pr = 0; pr < k.length; pr++) {

k1 = k[pr];

read(k1);

f1 = (function + f1);

}

average = f1 / k.length;

f1 = 0;
for (int p1 = 0; p1 < k.length; p1++) {
k1 = k[p1];
read(k1);

function = function - average;
function = function * function;
f1 = (function + f1);
}
f0a = 0;
for (int pr = 0; pr < k2.length; pr++) {

k12 = k2[pr];

read2(k12);

f0a = (function2 + f0a);

}

average2 = f0a / k2.length;
f0a = 0;

for (int p1 = 0; p1 < k.length; p1++) {
k12 = k2[p1];
read2(k12);

function2 = function2 - average2;

function2 = function2;
f0a = (function2 + f0a);
}
```

```
thickness();
thikExtract("as");
System.out.println("the length error is " + f0a);
f1 = f0a + f1 + thickSum;

if (i == 4) {
f1 = f0;
}

System.out.println("Effeective plasticstrain is " + EffectivePP);
gradient = (f1 - f0) / 1.2;

gradient1[i] = gradient;
System.out.println("Thickness error,  is" + function3);
System.out.println("f1, function2+f1 is" + f1);
System.out.println("gradient  is" + gradient);
System.out.println("error length is" + f0a);
// computing the gradient for one change only

double gradientSq = gradient * gradient;

int b = 0;
b = b + 1;

// computing the gradient for the step size calculation
gradientLengthSq = (gradientSq + gradientLengthSq);

System.out.println("counter is  " + counter);

}
gradientLength = Math.sqrt(gradientLengthSq);
System.out.println("GradientLengthis  " + gradientLength);
System.out.println("out of the loop");

// ======================================
// compute step size and increment all the design variables

if (!(gradientLength == 0)) {

alfa = (f0 - (1 - p) * f0) / (-(gradientLength * gradientLength));
if (counter == 0) {
for (int tr = 0; tr < gradient1.length; tr++) {
Float point = Float.valueOf((Float) designPoints.get(tr));
gradient2[tr] = alfa * (gradient1[tr]) + point;

} // gradient is the search direction and is multiplied with

}

System.out.println("stap size calc " + alfa);

if (!(counter == 0)) {
```

```
double beta = gradientLength / gradientM[0];
beta = beta * beta;
for (int tr = 0; tr < gradient1.length; tr++) {
Float point = Float.valueOf((Float) designPoints.get(tr));
gradient2[tr] = point + alfa * (gradient1[tr] + beta * gradient3[tr]);

}
}

}

System.out.println(Arrays.toString(gradient2));
incrementCurves(gradient2);
gradientM[0] = gradientLength;
for (int tr = 0; tr < gradient3.length; tr++) {
gradient3[tr] = gradient1[tr];
}

counter = counter + 1;

}

}
// Compute the length error of the node set

public static void nodeErr(String[] argsInMain) {
Node_error nodeSeg = new Node_error();
nodeSeg.main(argsInMain);
double thickness = nodeSeg.errNode;
double diff = 5 - thickness;
ThicknessError = Math.sqrt((Math.pow(diff, 2)));
}

// compute the effective plastic strain
public static void EFPP(String[] start) {
EFPP.main(start);
EffectivePP = EFPP.totEFPP;
}

public static void thikExtract(String run) throws IOException {
extractThickness thickness = new extractThickness();
thickness.main(run);
thickSum = extractThickness.totSum;
thickSum = thickSum / 2;
}

//// Read value of the design points and compute error; The output is in
//// variable function
public static void read(String k1) {

ReadFile nodeid = new ReadFile();
```

```
// String k2=(Arrays.toString(k));

nodeid.main(k1);
SimStat = nodeid.SimStatR;
double zcomp = nodeid.zcomp;
double xcomp = nodeid.xcomp;
double ycomp = nodeid.ycomp;
double fx = 0;// represents the value of the difference between the
// reference points and the one to be optimised

function = zcomp;

}

// read values for reference points
public static void readRef(String node) {

ReadFile reference = new ReadFile();

// String k2=(Arrays.toString(k));

reference.main(node);
if (p2 == 0) {
ref1z = reference.zcomp;
ref1x = reference.xcomp;
}
if (p2 == 1) {
ref2z = reference.zcomp;
ref2x = reference.xcomp;
}
if (p2 == 2) {
ref3 = reference.xcomp;
}

if (p2 == 3) {
ref4z = reference.zcomp;
ref4x = reference.xcomp;
}
if (p2 == 4) {
ref5z = reference.zcomp;
ref5x = reference.xcomp;
}
reference.main(node);
}

//// Read value of the design points and compute error; The output is in
//// variable function2; this is used to create and compare function values
public static void read2(String k12) {

ReadFile nodeid = new ReadFile();

nodeid.main(k12);
```

```
double zcomp = nodeid.zcomp;
double xcomp = nodeid.xcomp;
double ycomp = nodeid.ycomp;
double length = xcomp * xcomp + ycomp * ycomp;
double length1 = Math.sqrt(length);
double length2 = 5 - zcomp;

double ip = length1 * length1 + length2 * length2;
ip = Math.sqrt(ip);

double fx = ip;

function2 = fx;

}

// Increment one design point
public static void increment(double args, int i) throws IOException {
ChangeParameter increment = new ChangeParameter();
increment.main(value2, i);

}

public static void incrementCurves(double args[]) throws IOException {
changeCurves increment = new changeCurves();
increment.main(gradient2);

}

// increment all the design points
public static void curves(double inc) throws IOException {
{
ReadCurves increment = new ReadCurves();
increment.main(inc);

designPoints = increment.designPoints;
}
}

// Start LS dyna and run the simulation
public static void igniteDyna2() {
try {
String[] cmd = { "/bin/sh", "-c", "dyna_s i=model.k NCPU= -8 > dyna.tmp" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("Dyna running, waiting to finish");
p.waitFor();

} catch (InterruptedException t) {
System.out.println("Error: didnt wait for dyna?");
}
} catch (IOException e) {
```

```
System.out.println("Error: did not execute the command rigth.");
}
}

// merge key files
public static void mergeKfiles() {

try {
File tempFile = new File("model.k");
if (tempFile.exists()) {
System.out.println("delete file for design parameters change");
tempFile.delete();
}
String[] cmd = { "/bin/sh", "-c", "cat file.k curve.k > model.k" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("waiting for process, mergeKfiles");
p.waitFor();
} catch (InterruptedException t) {
System.out.println("Error: didnt wait for merged files");
}
} catch (IOException e) {
System.out.println("Error: The files could not be merged.");
}

}

public static void mergeKfiles2() {

try {
File tempFile = new File("model.k");
if (tempFile.exists()) {
System.out.println("delete file for parameter change");
tempFile.delete();
}
String[] cmd = { "/bin/sh", "-c", "cat file.k parameter.k > model.k" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("waiting for process,mergeKfiles2");
p.waitFor();
} catch (InterruptedException t) {
System.out.println("Error: didnt wait for merged files");
}
} catch (IOException e) {
System.out.println("Error: The files could not be merged.");
}

}

public static void thickness() {

try {
```

```
File tempFile = new File("thick");
if (tempFile.exists()) {
System.out.println("delete file for design parameters change");
tempFile.delete();
}
String[] cmd = { "/bin/sh", "-c", "lspost c=lspost3.cfile" };
Process p = Runtime.getRuntime().exec(cmd);
try {
System.out.println("Extacting the thickness");
p.waitFor();
} catch (InterruptedException t) {
System.out.println("Caanot start  lspost");
}
} catch (IOException e) {
System.out.println("Cannot extract values");
}

}
}
```

C.0.2 Java code for writing in file

Java code for changing values in a file

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Scanner;;

public class ChangeParameter {
public static ArrayList<Float> designPoints = new ArrayList<Float>();
public double count[]=new double[1];


public void main(double args, int i) throws IOException {

File tempFile = new File("parameter");
if(tempFile.exists()){
tempFile.delete();
}
tempFile.createNewFile();

PrintWriter pw = new PrintWriter(new FileWriter(tempFile));
String line = null;

Scanner input = new Scanner(new File("curve.k"));
while (input.hasNextLine()) {
```

```
line = input.nextLine();

if ((!(line.length() == 40))) {
pw.println(line);
pw.flush();
// System.out.println(line);
//
}
String remove="a1";
if ((line.length() == 40)&&line.indexOf(remove) != -1) {
pw.println(line);
pw.flush();
}

if ((line.length() == 40)&&line.indexOf(remove) ==- 1) {
String coordLine1=line.substring(0,30);


if (count[0] == i) {

String coordLine = line.substring(30, line.length());
String coordLine2 = coordLine.replaceAll("[^0-9.]", "");

float coord = Float.valueOf(coordLine2);
coord = (float) (args);

coord=coord*100;
coord=(int)(coord);
coord=(float) (coord);
coord=coord/100;
String tail = String.valueOf(coord);
String st="          ";// has to be 10 in length



tail =st.substring(tail.length())+tail;

line=coordLine1.concat(tail);

System.out.println("replaced line is" + line);
}
count[0]=count[0]+1;
pw.println(line);
pw.flush();

}

}
input.close();

// System.out.println(designPoints);
```

```
pw.close();



}

}
```

C.0.3 Java code for reading values from the file

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Scanner;;

public class ReadCurves {
public ArrayList<Float> designPoints = new ArrayList<Float>();

public void main(double args) throws IOException {

File tempFile = new File("tempfile.k");
File originalFile = new File("curve.k");
PrintWriter pw = new PrintWriter(new FileWriter(tempFile));
String line = null;

Scanner input = new Scanner(new File("curve.k"));
while (input.hasNextLine()) {

line = input.nextLine();

if ((!(line.length() == 40))) {
pw.println(line);
pw.flush();

}

String remove = "a1";
if ((line.length() == 40) && line.indexOf(remove) != -1) {
pw.println(line);
pw.flush();
}

if ((line.length() == 40) && line.indexOf(remove) == -1) {
String coordLine = line.substring(31, line.length());
String coordLine1 = line.substring(0, 30);
String coordLine2 = coordLine.replaceAll("[^0-9.]", "");
```

```
float coord = Float.valueOf(coordLine2);
coord = (float) (coord + args);

designPoints.add(coord);
String tail = String.valueOf(coord);
String st = "          ";// has to be 10 in length

tail = st.substring(tail.length()) + tail;

line = coordLine1.concat(tail);

pw.println(line);
pw.flush();

}

}
input.close();

pw.close();

originalFile.delete();

// Rename the new file to the filename the original file had.
tempFile.renameTo(originalFile);
tempFile.delete();

}
}
```