DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Optimization Software for Automation in Die Design for Polymer Processing

Monografia submetida à Universidade Federal de Santa Catarina como requisito para a aprovação da disciplina:

DAS 5511: Projeto de Fim de Curso

Tarik Medeiros Siqueira

Florianópolis, Julho de 2012.

Optimization Software for Automation in Die Design for Polymer Processing

Tarik Medeiros Siqueira

Stephan Eilbracht

Orientador na empresa

Prof. Marcelo R. Stemmer

Orientador do curso

Esta monografia foi julgada no contexto da disciplina
DAS 5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Acknowledgements

First I'd like to thank my family, who always supported me and shared all the good and bad moments in my life. My mother, Joelma Medeiros Siqueira, and father, Luís Benício Tavares Siqueira, for giving me the best education possible, always believing in me and forming the best family I could ever ask for. My brother who made my life much more fun and taught me how to care for others.

I deeply thank my girlfriend, Karen Félix da Silva, who made me company through all these years I was in the university. I could not have done without your help and love.

I also thank Stephan Eilbracht for making this internship possible and for all the support given during this period. Professor Marcelo R. Stemmer for creating the cooperation between the Federal University of Santa Catarina and RWTH – Aachen and being my mentor during this work. Alberto Xavier Pavim for conducting the selection process and giving all the assistance needed to the students.

All my appreciation to my friends in Brazil who participated in important moments in my life and influenced me to become who I am today. To my friends who were also here in Germany and lived together this experience.

To my work colleagues: Guilherme Bencke, for his contribution to the project and my learning; and Daniela Pennartz for all the support and for helping me experience more of the german culture.

Resumo

O plástico está presente em várias aplicações domésticas e industriais, colocando sua indústria como uma das maiores e mais importantes do mundo. A versatilidade de produtos possibilita sua aplicação nas mais diversas áreas. Para os próximos anos, espera-se um crescimento em torno de 5% ao ano neste ramo.

Dentre os vários processos de fabricação, a extrusão é o mais utilisado. Este processo permite uma flexibilidade quanto ao material a ser utilizado e à forma do produto final desejado. Esta vantagem é possível devido ao molde, uma peça que faz parte da extrusora. Ela é responsável por transformar o fluxo de material polimérico em um produto final com a geometria desejada.

O design do molde é bastante complexo visto que existem várias variáveis discretas com grande quantidade de valores possíveis envolvidas no processo. O comportamento não linear do fluxo de material polimérico dentro do molde também torna a obtenção dos resultados de forma analítica uma tarefa quase impossível. Por isso, a simulação por software é imprescindível.

A complexidade da geometria resulta em milhares de possíveis combinações de parâmetros. O processo para encontrar a melhor é geralmente feito simulando cada ferramenta individualmente e aplicando alterações manualmente por tentativa e erro, o que demanda bastante tempo. Além disso, os moldes são geralmente utilizados com vários materiais, que possuem características distintas, e a ferramenta precisa satisfazer as especificações para todos os casos.

Este projeto, denominado Wendigo e em desenvolvimento no Institut für Künststoffverarbeitung (IKV) – Universidade RWTH Aachen, busca desenvolver um programa capaz de captar todas as especificações do usuário, simular e fornecer os resultados obtidos assim como realizar a otimização do molde para encontrar a melhor geometria sem a necessidade de um operador durante o processo. É importante ressaltar que o escopo do projeto limita-se apenas a moldes utilizados para a fabricação de produtos com formas tubulares.

A simulação utiliza-se de um modelo do processo construído utilizando um circuito elétrico como analogia. Esse método é conhecido como controle de volume e

também chamado de “teoria de rede”. Para calcular os resultados desejados, o sistema de matrizes esparsas é gerado e resolvido através de um algoritmo iterativo utilizando um *solver* optimizado para esse tipo de sistema desenvolvido por terceiros e fornecido sob licença livre.

A optimização do processo de design do molde analisa três parâmetros de saída: distribuição de volume, histórico da taxa de cisalhamento e diferença de pressão total. Essa tarefa é realizada pelas estratégias distintas implementadas que se utilizam dos parâmetros de qualidade calculados pela simulação como as funções a serem minimizadas para a obtenção das melhores soluções de geometria que atendam às especificações do usuário.

Abstract

The plastics are present in many domestic and industrial applications, placing its industry as one of the biggest and most important in the world. The versatility of products enables its applications in vast distinguishable areas. For the next years, a growth of 5% is expected to this segment.

Among the fabrication processes, the extrusion is the most used. This process allows a flexibility regarding the material to be used and the profile to be shaped in the final product. This advantage is possible because of the die, which is a part of the extruder machine. The die is responsible for transforming the flow of the melted polymeric material into the final product with a specific geometry.

The design of the die is reasonably complex since there are many variables with a large range of values involved in the process. The non-linear behavior of the polymeric melt flow inside the die also transforms the analytical calculation of the results into an almost impossible task. Therefore, the simulation through software is highly necessary.

The complexity in the geometry results in thousands of possible combinations of parameters. The process of finding the best one is usually done by simulating each tool individually and applying changes manually by trial and error, which takes a large amount of time. Furthermore, the tools are usually used with multiple materials, which have different characteristics, so the design has to satisfy the constrictions in all cases.

This project, named Wendigo and under development at the Institut für Kunststoffverarbeitung (IKV) – RWTH Aachen University, seeks to develop a program capable of collecting all the user specifications, simulate and return the results calculated as well as perform the optimization of the tool to find the best geometry without the need of one operator during the process. It is important to note that the scope of the project is limited only to dies used to manufacture products with tubular shapes.

The simulation uses a model of the process built using an electrical circuit analogy. This approach is known as control volume and also referred to as “network

theory”. In order to calculate the desired results, a system of sparse matrices that represents the network is obtained and solved through an iterative algorithm using a third party open source solver optimized for this task.

The optimization of the die design process analyses three output parameters: volume distribution, shear rate history and total pressure drop. This task is performed by the different strategies implemented that use the quality parameters calculated by the simulation as the function to be minimized in order to find the best geometry solutions that meets the user’s specifications.

Index

List of images

Simbology

Abreviations:

IKV – Institut für Künststoffverarbeitung

RWTH – Rheinisch-Westfälische Technische Hochschule

FVM – Finite Volume Method

mA – Milliamperes

C# – C Sharp

GUI – Graphic User Interface

UML – Unified Modeling Language

RAM – Random Access Memory

CSR – Compressed Sparse Row

Symbols:

$\dot{\gamma}$ – Shear Rate

v – Viscosity

r – Radius

w – Width

h – Height

n – Viscosity

a_T – Temperature shift factor

A – Carreau parameter: zero shear rate viscosity

B – Carreau parameter: reciprocal transition rate

C – Carreau parameter: slope of the viscosity curve

C_1 – Constant of value 8.86

C_2 – Constant of value 101.6

T – Desired temperature

T_s – Standard temperature

T_0 – Temperature at which the viscosity is known

ºC – Degrees Celsius

L – Land length

º – Degrees

Q – Quality index

Qv – Volume flow quality

Qs – Shear rate quality

Qp – Pressure drop quality

Chapter 1. Introduction

1.1: Motivation

The industry of plastics is one of the largest all over the world. The variety of products conceived are used in many applications such as health, nutrition, shelter, transportation, security, communication, sports, leisure activities and industry. The perspectives are that the demand for plastics will grow around 5% per year [13].

The market of plastic processing machinery, with higher intrinsic value, is expected to grow around 4.7% yearly. The extrusion equipment will be leading this expansion, pushed by the growth of construction [14].

The die is one of the main parts of an extruder machine. It's responsible for transforming the flow of material into a certain profile of thickness and shape necessary for the final product. Because the melted plastic is a non-Newtonian fluid its viscosity is not constant and depends on the shear rate [1]. The tools are usually used with multiple materials, which have different characteristics that lead to different optimum solution, so the design has to satisfy the constrictions in all cases. These are only a couple of the reasons why it's very complex to design a die that meets the specification required.

Besides the complicated material behavior, the number of parameters and its possible values (parameters are discrete) that can be adjusted to create a die results in an enormous amount of possible tool geometries. For example, if 20 parameters are considered and each has 100 possible values there are 1e+40 different combinations. This would also have to be multiplied by the number of different materials and processing points (temperature and throughput) that would be used in the tool since each of them would have to be analyzed independently as they generate different results. It would easily take thousands of attempts to find a good solution. And still, the designer has no way of knowing if that is the best solution possible.

The use of software to simulate the results of a possible tool is nowadays largely used. The main problem is that the entire die is usually a combination of two

or more parts, and most of the simulation programs available only evaluate each part separately, not taking into account the integration of them. They were also developed many years ago, and because the computers were much slower, a lot of simplifications had to be made in order to conceive a simulation within an acceptable time. Recent simulations programs based on the Finite Volume Method have achieved very good results regarding the accuracy of the data, but the process requires powerful computers, skilled engineers, a 3D-model and very long processing times [2]. Since the goal is to create an optimization program that will run hundreds of simulations, the calculation times of the FVM make its use unfeasible in this project.

The process of specifying a good die through software is still very manual, as the programs are usually capable of running the simulation based on the user input, but they are not able to change the parameters in order to search for a better solution. This means that the designer will have to spend a lot of time executing these operations and analyzing the output.

This project will create a program able to optimize the design process of dies. For this purpose, it will simulate the rheology of the tool under operating conditions, evaluate the results and change the geometry specification in order to achieve better results. The process is illustrated in Figure 1.

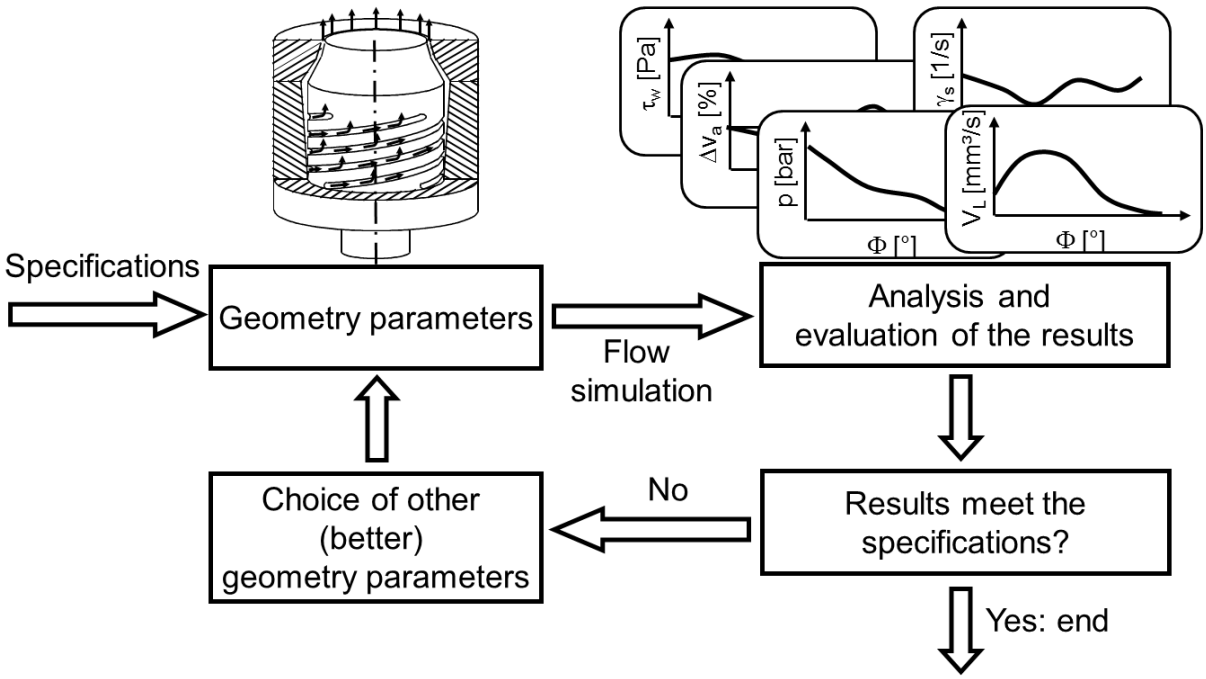

Figure 1 - Optimization process scheme

The simulation uses a model of the process built using an electrical circuit analogy. This approach is known as control volume and also referred to as “network theory” and have already been successfully used in this context [3]. In order to calculate the desired results, a system of sparse matrices that represents the network is obtained and solved through an iterative algorithm using a third party open source solver optimized for this task. While some simplifications will have to be made in order to use this strategy, the results should still be close to the real behavior. This analysis will provide fast results. The speed is necessary because the software will be able to automatically search for the optimum design, which means that thousands of simulations might be necessary.

The optimization of the die design process analyses three output parameters: volume distribution, shear rate history and total pressure drop. This task is performed by the different strategies implemented that use the quality parameters calculated by the simulation as the function to be minimized in order to find the best geometry solutions that meets the user’s specifications. The methods have different characteristics that will be studied and compared in order to apply them the most efficient way.

1.2: Context in graduation course

During the realization of this project, knowledge from several disciplines studied in the Control and Automation Engineering graduation at Federal University of Santa Catarina were applied.

The disciplines related to software programming were the most useful. Among them, Methodology for Software Development was used since the beginning as the UML language was used. Industrial Informatics 2 was also applied through the multithreaded concept.

Disciplines like Calculus, Algebra, and Transport Phenomenon were also very important to understand the physical problem, create and evaluate the abstract model that was used to represent the real problem. Electrical Circuits was essential to build and solve the network generated by our model.

1.3: Report structure

This report is divided in five main parts. The chapter “Extrusion machinery” explains the parts, operation, design specifications and goals of an extruder machine highlighting the die, which will be the only part considered in the software. In the next chapter, called “Mathematic model”, it’s explained how the real behavior of the process is being represented in the software using the circuit analysis approach. The following chapter “Software development” contains the information about the design and implementation of the software. “Optimization” describes all the theory and implementation made on this topic. In the chapter “Results” the achievements of the program are exposed and analyzed. The last chapter “Conclusion and perspectives” discuss the main parts of the work and the future of the project.

Chapter 2. Extrusion machinery

The extrusion is the most efficient and widely used process of melting plastics. It transforms plastic material, generally in granular or powder form, into one continuous melt. The last can be shaped into a large variety of forms and then cooled down to achieve the final desired properties for the material. Each of these steps occurs in different parts of the tool and will be briefly explained. Because this project focuses only on the extrusion die, this part will be detailed later in its own section.

2.1: Tool overview

The Figure 2 is a scheme of a typical extruder and its components. The extrusion process starts with the raw plastic material, identified in the picture as the Resin, being fed into the Hopper, which function as a buffer that stores the content and releases it continuously inside the barrel. The screw is constantly rotated by the motor, in the left in the picture, attached to the barrel. This movement pushes the material forward. The heat caused by the friction of the material being pushed, combined with external heaters, melts the mass into a viscous fluid. The main goal of this part is to create a flow of material as homogeneous as possible. At the end of the screw, the melt is pumped inside the die, responsible for shaping a profile and form the semi-finished product. The capability of using different dies gives the flexibility to conceive many different products.

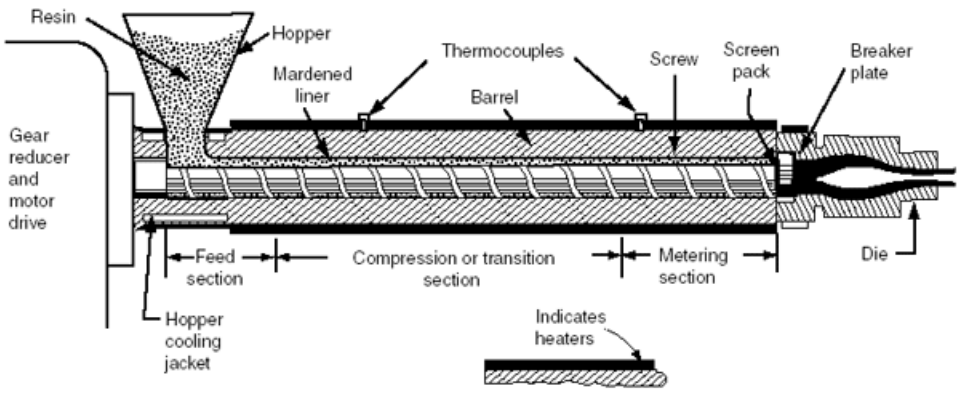

Figure 2 - Scheme of a typical extruder machine [16].

The last step of the process is the cooling. The material leaves the die and enters the calibrator, illustrated in Figure 3, where it will be cooled by a specific

amount and speed to preserve the geometry and achieve the desired mechanical properties.

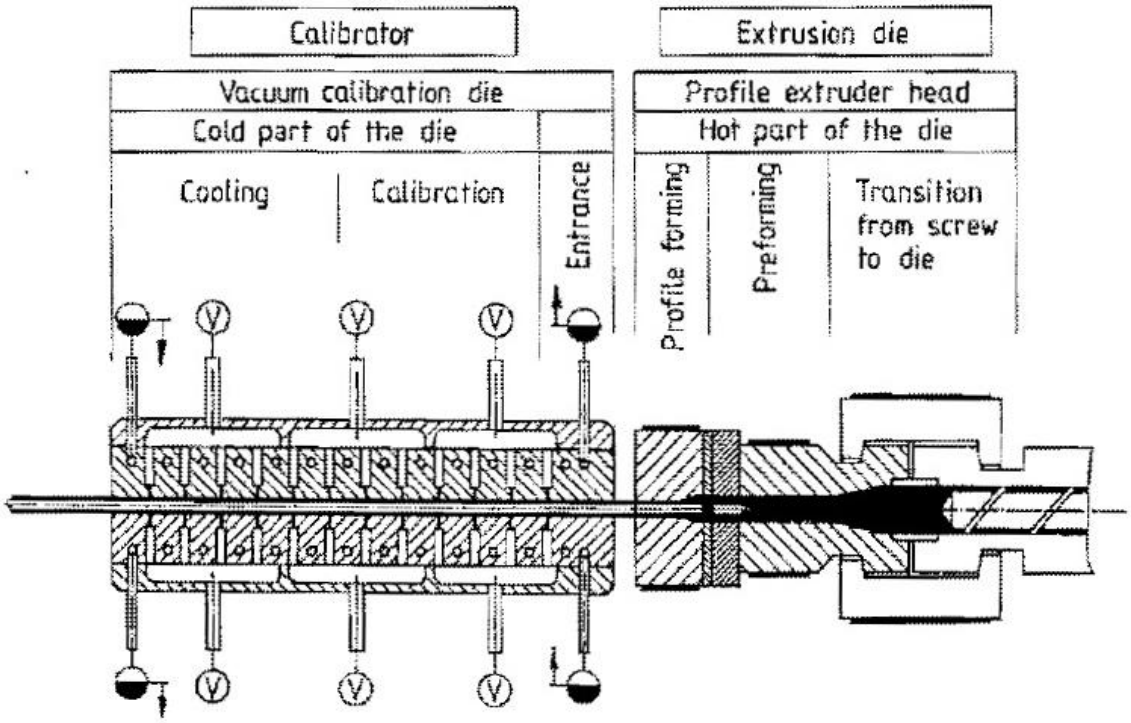

Figure 3 – Scheme of an extrusion die and a calibrator

2.2: Properties of polymeric melts

The polymeric melts that flow through the dies are viscous and behave as non-Newtonian fluids as they also present elasticity. Therefore, to perfectly describe them mathematically very complex and equations would be required. Since this project seeks only an approximate result for its simulations, some simplifications are made to simplify the formulas applied [18]. The main equations will be exposed in this section. It's also very important to consider the boundary conditions of the model which may limit the user input data that will describe the material being simulated.

During the flow, the melt adhere to the die walls causing a shear deformation and changing the velocity through the tool. This effect is represented through the shear rate equation according to the geometry of the flow channel.

$$\dot{\gamma} = \frac{4 \cdot v \cdot 0.815}{\pi \cdot r^3}$$ Shear rate for a pipe

$$\dot{\gamma} = \frac{6 \cdot v \cdot 0.772}{w \cdot h^2}$$ Shear rate for a slit

In the equations above it's possible to see that the shear rate $\dot{\gamma}$ is directly proportional to the flow velocity v and inverse to the geometry, represented by the radius r in the pipe and width w and height h in the slit. The Figure 4 shows the difference between a Newtonian and this pseudo plastic fluid.

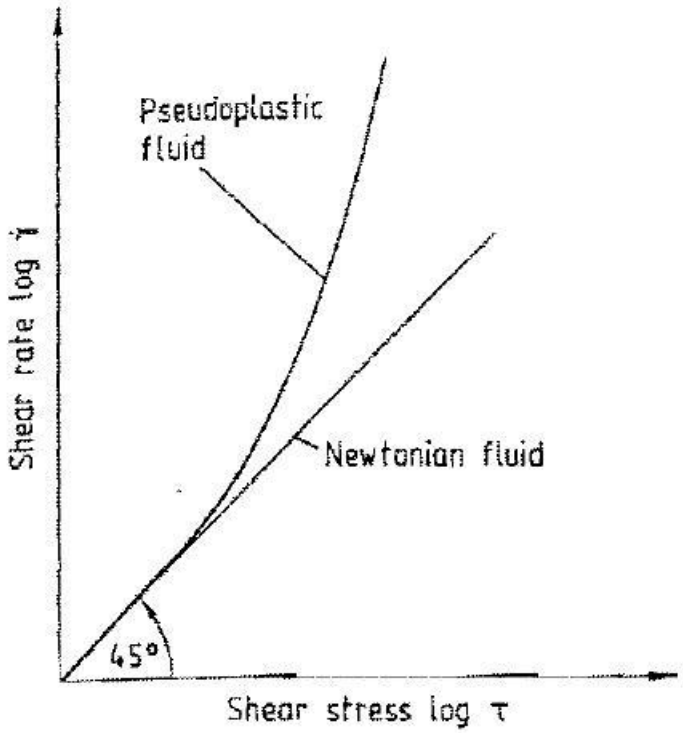

Figure 4 - Shear rate behavior for Newtonian and Pseudo plastic fluid [18].

As mentioned earlier, the viscosity n is a characteristic present in the melts and can be represented by many different models. The Carreau Constitutive Equation was chosen because it can represent the actual behavior of the material over a broad range of shear rates. The equation, considering a temperature shift factor a_T, is:

$$n = \frac{a_T \cdot A}{(1 + a_T \cdot B \cdot \dot{\gamma})^C}$$

The parameter A is the zero shear rate viscosity, i.e., the value of the viscosity when the material has a shear rate of zero. B is the reciprocal transition rate and C the slope of the viscosity curve in the pseudo plastic region at $\dot{\gamma} \rightarrow \infty$. Their graphical meaning can be seen at Figure 5.

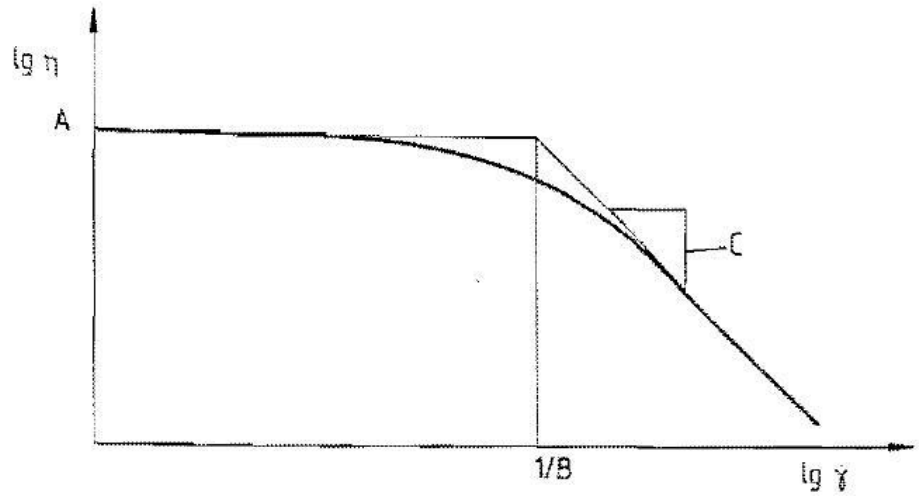

Figure 5 - Graphic representation of the Carreau model and its parameters[18].

The viscosity of a certain material along a range of shear rate values forms a specific curve that can be translated and rotated in a graphic depending on the temperature. Therefore, this curve is usually determined for one standard temperature, and a shift factor a_T can be calculated to adjust it to a different one.

$$a_T = 10^{\left(\frac{C_1 \cdot (T_0 - T_s)}{C_2 + (T_0 - T_s)} - \frac{C_1 \cdot (T - T_s)}{C_2 + (T - T_s)}\right)}$$

C_1 and C_2 are constants with values of 8.86 and 101.6, respectively. T is the desired temperature, T_s the standard (approximately 50ºC above the glass transition temperature) and T_0 an arbitrary one at which the viscosity is known. Figure 6 shows various curves from the same material at different temperatures. The shifting in the curves can be seen following a specific direction and can therefore be determined.

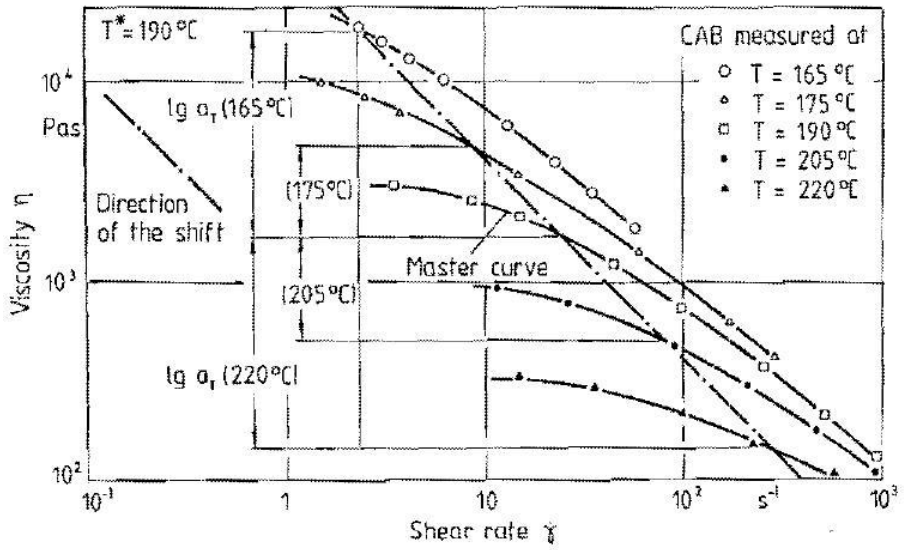

Figure 6 - Temperature influence on viscosity curve [18].

2.3: Die

After the material behavior has been modeled one can focus on the die geometry. The polymer melt flows through the die, where its final profile is shaped. This tool can be used to achieve many forms, such as flat films, pipes, filaments, hollow and open profiles [19]. The software proposed in this project will be able to design only tubular dies, which itself is composed by a few primary dies as shown on Figure 7. This focused effort benefits the project by allowing much simpler construction of the model due to the constant features. This translates into speed and lower memory consumption, two critical aspects that will determine the success of the project.

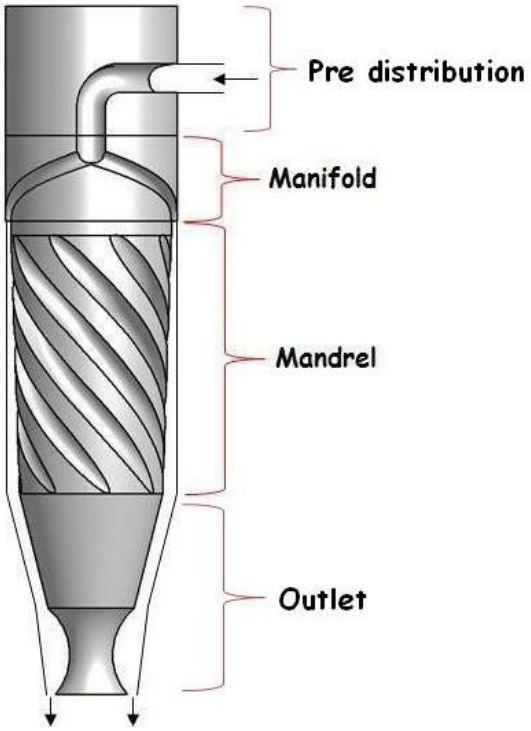

Figure 7 - Die scheme.

The two main parts of the tool are identified in the middle of the picture above: The manifold is responsible for transforming the flow coming from a tubular channel into a film distribution wrapped around the cylindrical tool, while the mandrel improves the homogeneous distribution. The first part of the tool is called pre-distribution and divides the flow of material equally so that each coat hanger is fed by the same amount. The last part is the outlet, where geometry changes can still be made to the final shape.

Some general restrictions for the whole tool design are the size of the tool and the pressure drop necessary to force the flow of the material. The quality depends usually on the mass distribution profile and shear rate history along the tool.

2.3.1: Pre distribution

This first part of the die receives the melt flow from the extruder and divides it to match the number of manifolds used in the next part of the tool (see Figure 8). It is very important that all the divided flows are as equal as possible so that a uniform distribution along the circumference is more easily achieved. For that, each flow is divided only into two channels. This approach gives better result as it is easier to make a division symmetrical to just one axis.

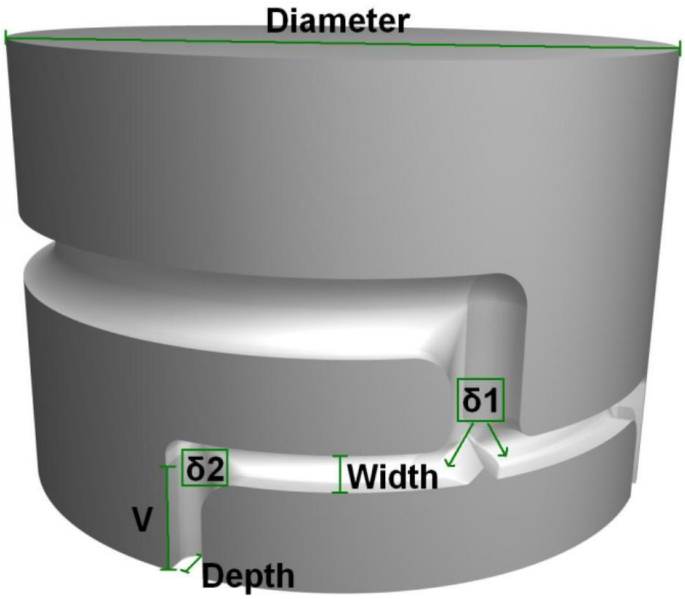

Figure 8 - Pre distribution tool with parameters.

After splitting a channel in two, they spread into opposite directions until they are as far apart from any channel as they can be. In the first division for example, each will run 90º to each side, being separated by 180º in both directions. Then they will go down by a certain length of the tool and be divided again if needed. The next division will increase the amount of channels to 4 because all the existing channels have to be divided to keep the symmetry of the tool. Now each will be separated by 90º to each side.

The channels being separated as described results in a restriction of how many manifolds are possible to have. The number follows the series 2^n where n can be 0 or any positive integer. This way, a few possible quantities of manifolds are: 1, 2, 4, 8, 16 and 32. Theoretically this series could be infinite. However, this is a physical tool and, as described earlier, the more manifolds there are the closer the channels will be and the manifolds will get smaller as well. Therefore, we chose to limit our software to work with a maximum of 32 manifolds, which is considered a large amount as the most common tools vary between 1 and 8.

After all the divisions necessary are done the number of channels, and therefore material flows, match the number of manifolds in the next part. Each channel is then connected to the center of a manifold, feeding the material to the next step.

2.3.2: Manifold

The function of this part is to evenly distribute the melt that comes through a channel to a flat region. It was originally designed and used to produce flat film and sheet shapes. Later it was adapted and wrapped around a cylinder to produce pipe and tubular shapes. Different geometries can be used to achieve this goal. The most important can be seen in Figure 9 and are called T, fishtail and coat hanger manifold.

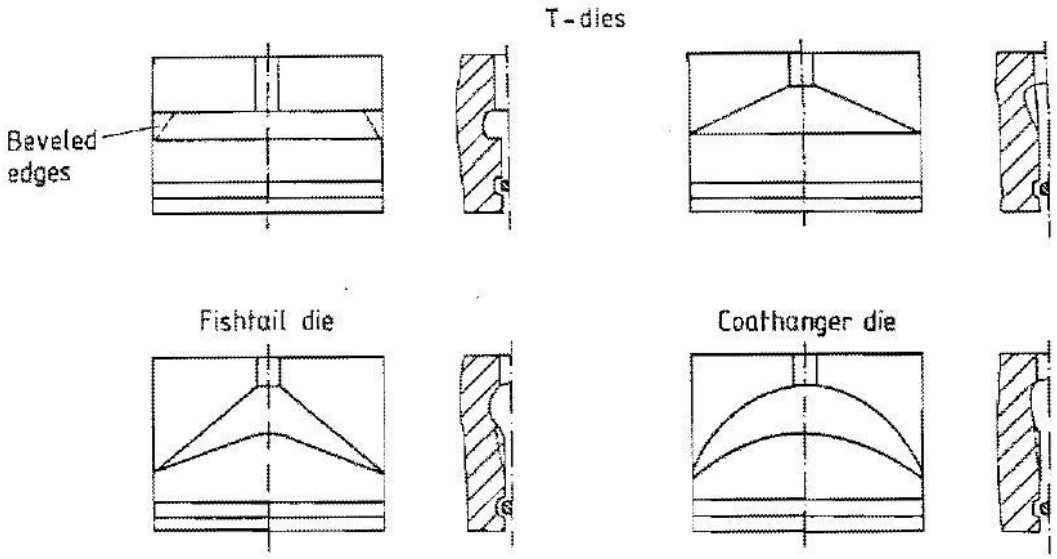

Figure 9 - Manifold shapes [4].

The last is the most frequently used because, if properly designed, provides a good distribution and higher independence of the operating conditions. Therefore it was the first choice to be implemented in the software. The other geometries will be added later once the simulation process is validated.

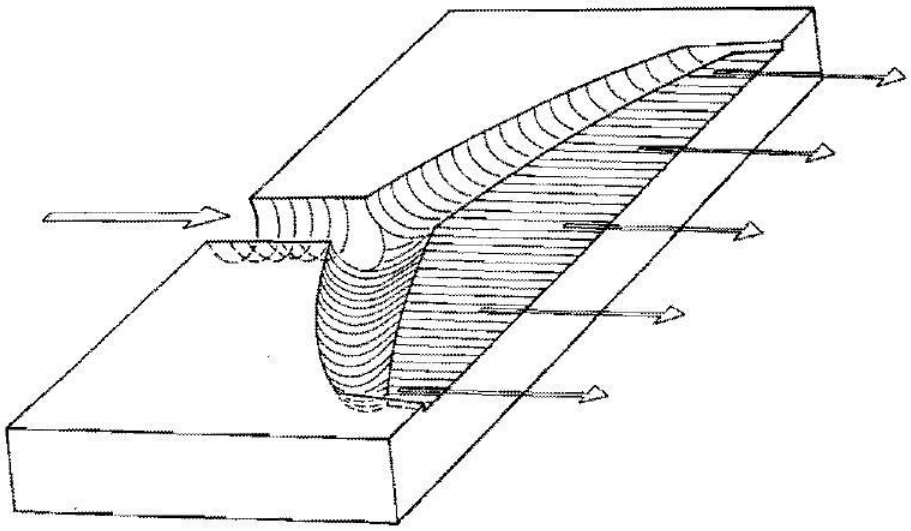

Figure 10 – Scheme of melt flow inside a coat hanger [5].

The melt coming from the pre distribution flows inside from the channel above the tool as seen on Figure 10. The following two channels in opposite directions help the melt distribute along the whole width. Through the whole channels the material also flows straight to the land area, which has a certain height.

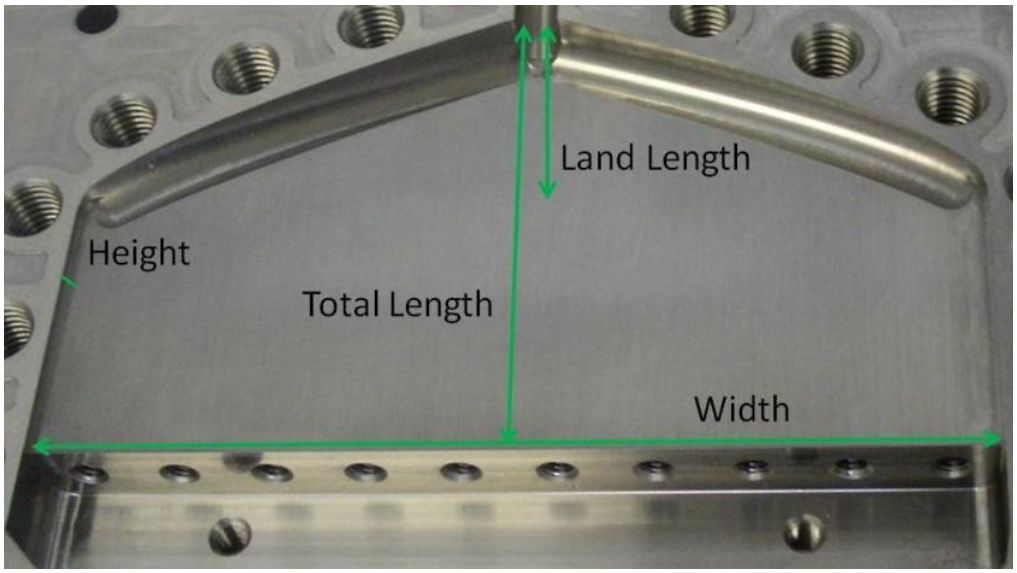

Figure 11 - Real coat hanger manifold with parameters indicated.

All the dimensions shown in Figure 11 plus the radius at the center of channel are input parameters that define the design of this tool. Changes here can influence the total pressure drop and the distribution of the material. The curvature and the radius of the channels are specified by the equations:

$$y(x) = y_0 \cdot \left(\frac{x}{L}\right)^{\frac{2}{3}}$$

$$R(x) = R_0 \cdot \left(\frac{x}{L}\right)^{\frac{1}{3}}$$

Usually, more than one coat hangers are used in this part to achieve a better distribution. Even then, there are critical points such as the junctions between two manifolds where the material distribution is not so homogeneous. Therefore, another different tool is needed.

2.3.3: Mandrel

The mandrel is fed directly by the manifold output, thus the melt already comes spread all around the cylinder. But because the distribution is still not perfectly homogeneous, the spirals (as seen in Figure 12) are created to allow the material to flow more easily in the horizontal direction, while it still keeps the vertical flow. A part of this flow enters the spirals, flow through them and leaves the spiral channels anywhere along the die, promoting a more profound mixture of material and even dispersion [20].

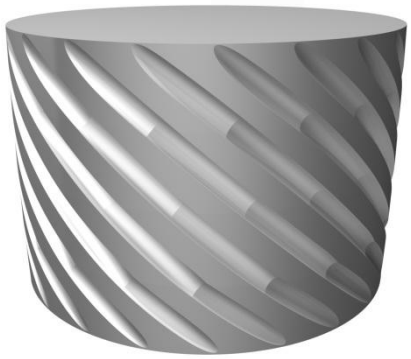

Figure 12 - Mandrel tool scheme.

The specification of this tool can be very complex depending on the designer. The size is determined by the diameter and height. Since this tool is located inside an outer cylinder, there's a gap between the two of them, which allows the melt to flow through. This gap can be customized as the user can specify a function or points for interpolation that describe small variations of the diameter. The channels also have a similar approach regarding its depth. Figure 13 shows an example of both defined using points and interpolation.

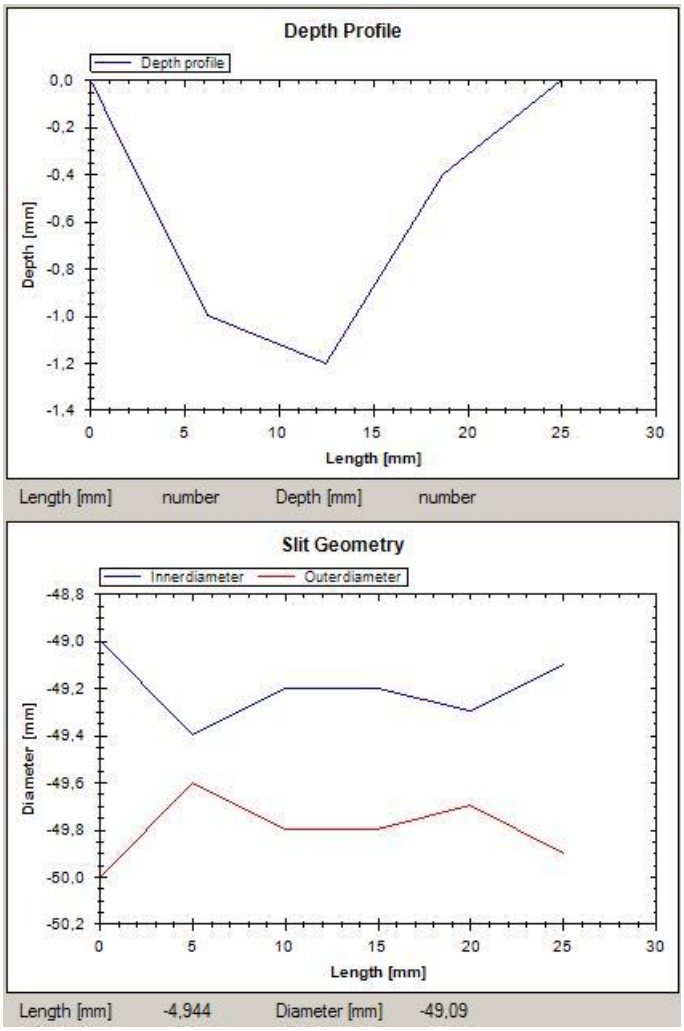

Figure 13 - Spiral depth and mandrel slit geometry visualization.

It's also important to specify the quantity of spirals, maximum width and the enlacement, i.e., how many degrees it will run across the cylinder. This is the last part of the tool designed to change the melt flow and achieve homogeneous volume distribution and shear rates along the circumference. Therefore, it is expected the output of this part to be very similar to the specified in the final product. The last section of the tool may still introduce only slight changes to these characteristics as its design has no feature to promote significant influences.

2.3.4: Outlet

This last part of the tool (seen in Figure 14) is designed to apply geometrical changes to the shape being extruded. In the cylindrical tools specified in this project it is often desired to change the diameter, hence this will be the only modification allowed in this tool.

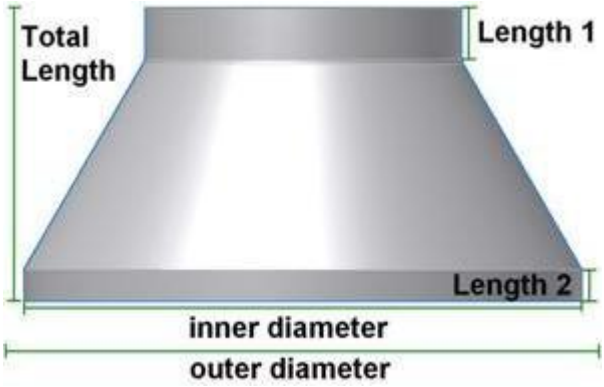

Figure 14 - Outlet with parameters

The beginning of the tool has the same diameter as the mandrel part and will receive the material flow coming from the last. Depending on the relation between the mandrel and the final diameter there will be an expansion, if the latter is larger, or a contraction in the opposite situation. The parameters are much simpler: inner and outer diameter, total length, length 1 (beginning section with the same diameter as the mandrel) and length 2 (final area with the desired output diameter).

Chapter 3. Mathematic model

In order to simulate the behavior of the process a mathematical model is needed. Solving the problem through analytical equations is not practical since they are very complex and would not take advantage of the computer processing. A possible solution would be to use Finite Volume Method (FVM). This method is proven to be very accurate, flexible and is already implemented with success in other programs. The disadvantage of FVM is that the description and analysis are very complex and the setup and simulation takes too long. Since the goal of this project is to conceive a program that can optimize a tool simulating hundreds or thousands of possibilities in order to find some of the best ones, a slow simulation wouldn't fit properly.

A more suitable solution found was the Circuit Analysis Method where an electrical circuit is designed to represent the physical process. This model have already been applied, studied in the die context and proven to produce good approximation results [6][7]. The simplifications made are the discretization of the flow, approximation of geometries and two dimensional behavior. While each of them contributes to a more inaccurate result, they are necessary and the total error is acceptable. After running the optimization the user will be able to view the results from the best solutions found and select one or more to simulate them in independent FVM software for more detailed and accurate results before choosing which is the best solution.

3.1: Circuit Analysis Theory

Electronic circuits can contain a lot of different components, such as resistors, capacitors, inductors, diodes, transistors and sources. It can also be very big regarding the amount of devices. To solve complex circuits with the classic approach of using Kirchhoff's circuit laws is unpractical, since it could generate very complex equations and the lack of pattern on how to determine meshes creates another difficulty when trying to implement this method in a program.

David J. Sager [21] proposed a simple method to resolve these circuits and take advantage of the computer processing capability. This approach generates lots of simple equations that will form a sparse matrix system. Nowadays there are fast solvers optimized for this type of systems. After conducting tests with this method it was decided to be used since the speed and accuracy met the desired range.

In this method the network is made of devices, each one with its own characteristic and pair of terminals. The terminals are connected in pairs by links, which will have currents associated. A group of terminals connected by links forms a node that has a voltage. The description of network is broken into two parts. In the first one the topology of the network is built by the connection of terminals. The second part describes the devices, using the equations that represent its characteristics.

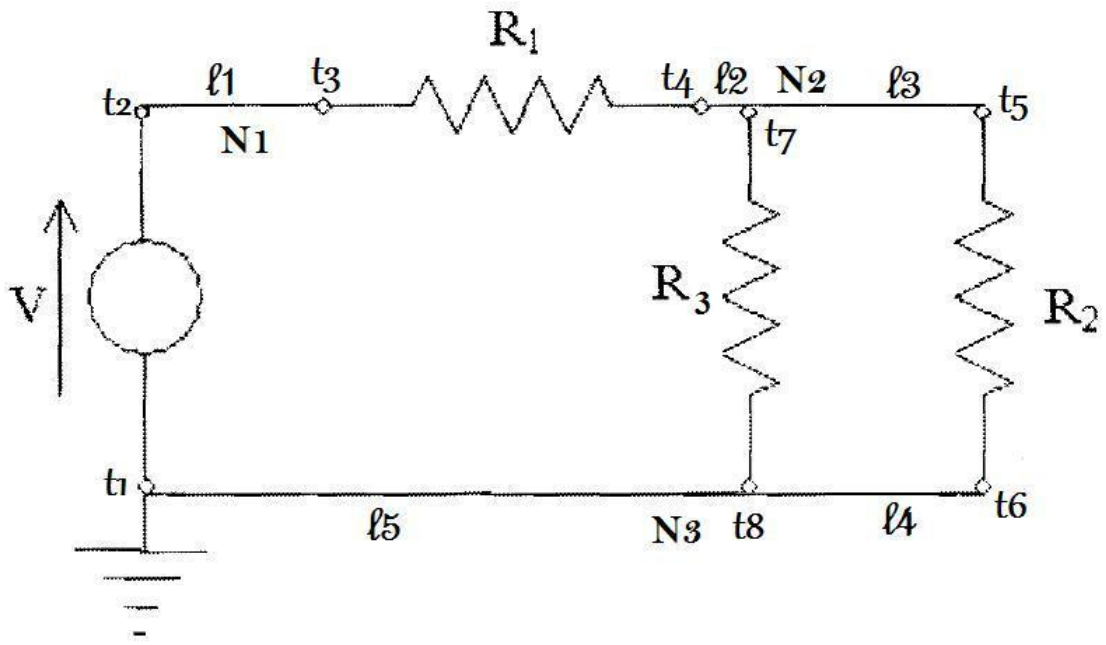

Figure 15 - Simple circuit example.

In the example above, Figure 15, the circuit is made of four devices, three resistors and one source. Each one of them has two terminals so the total is eight. There are five links connecting pairs of terminals, and only three nodes. These elements describe the topology. The resistance and the source power describe the characteristics of the resistors and the source, respectively. According to Ohm's Law:

$$V = R * i$$

Where V is the potential difference measured across the resistor, R the resistance and i the current that flows through. Expressing V as the difference between the potential on the inflow and outflow ($V_{in} - V_{out}$) and subtracting $R * i$ from both sides we arrive at:

$$(V_{in} - V_{out}) - R * i = 0$$

This is the equation used to describe the resistors in the example given and results in “negative” resistance values inside the matrix, which indicates only the direction of the flow.

	Link 1	Link 2	Link 3	Link 4	Link 5	Node 1	Node 2	Node 3
GRD	0	0	0	0	0	0	0	1
Device 2	0	0	1	-1	0	0	0	0
Device 3	0	1	0	0	-1	0	0	0
Device 4	-1	0	0	1	1	0	0	0
Device 1	-R1	0	0	0	0	1	-1	0
Device 2	0	0	-R2	0	0	0	1	-1
Device 3	0	-R3	0	0	0	0	1	-1
Device 4	0	0	0	0	1	-1	0	1

Figure 16 - Network matrix for simple example circuit

The Figure 16 shows the matrix that describes the circuit illustrated previously. In the upper left corner (green x green) is the topology section where the inflow links are represented by 1 and the outflow by -1. Only the ground receives a value in the upper right part (green x yellow) to indicate its position. The lower left section (yellow x green) is where all the resistances values are. For the device 4, which in this case is the source, the value 1 is set to be multiplied by the other matrix that carries its real value. The last area (yellow x yellow) describes the nodes connections to the devices, 1 for inflow and -1 for outflow.

For the model constructed in this project only two types of devices are needed: resistors and current source. In this analogy the electric current represents the volume flow in the real die and the voltage is the pressure. The source acts as the extruder before the die, which feeds the last with the material melt while the resistors reproduce the resistance to the flow imposed by the physical tool. This way, the equation used for electric circuits can be adapted as:

$$(P_{in} - P_{out}) - R * v = 0$$

Where P is the pressure, R is the flow resistance and v the volume flow.

3.2: Design of the Circuit Model

Using the concepts explained in the previous section, a network was created to simulate the real flow of the material inside a die. The first idea was to create a symmetric network so that no distortion from the real geometry would be needed and the numbering patterns would be easier to develop. Because in the manifolds section the symmetry is dictated by the number of manifolds and in the mandrel part by the number of spirals they would, in most cases, not match. That would require two different amounts of columns creating a problem to connect them.

To solve this issue a transition line was created with function of only connecting the different networks without changing the data. After considering a few designs to this part, one was chosen and implemented. The implementation took several months and this was the most complex task of the project. In the end, when the model was evaluated through several analysis methods, we discovered that it was not behaving like expected. Instead, it was changing the profile of the volumetric flow. It was decided then to create another model without the transition but introducing a few small distortions to the end and beginning of the spirals. This model is expected to introduce less error, although its implementation it's still incomplete.

3.2.1: Model with transition

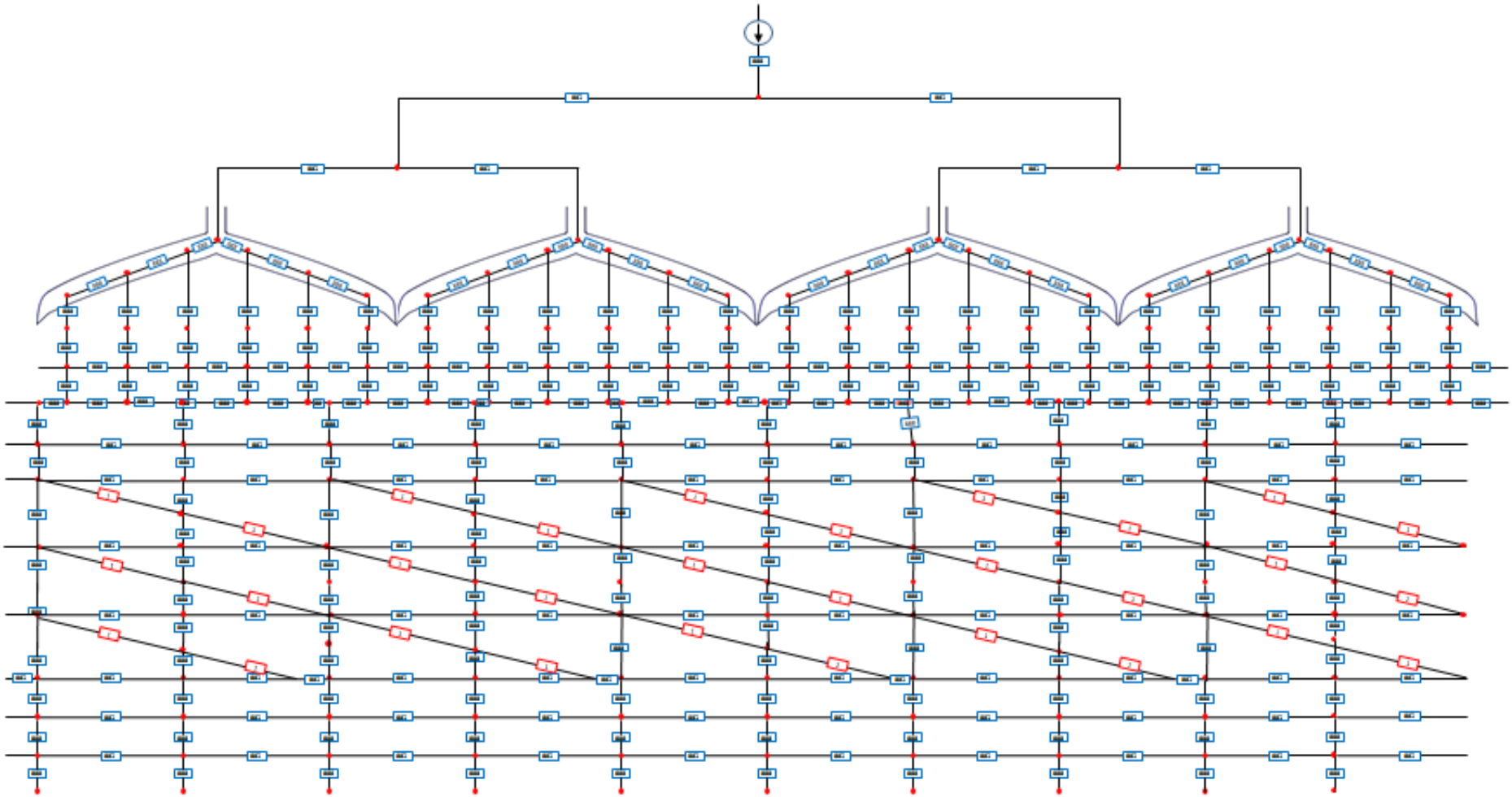

Figure 17 - Network model scheme.

At the top of the Figure 17 the pre distribution level is represented. Since they are basically channels that divide the flow of the material, one element placed between each division is enough to represent the behavior properly. In reality these channels do have some more complexity since there are chaotic flow due to the division of channels, changes of shear rate and curves. Therefore, when processing each of these elements, they are expanded to more devices where factors can be applied to simulate all those effects. The result is then applied to the original element represented in the network above.

The coat hanger section is more complex, and the resolution, which is an input parameter, is used to decide how many columns will be placed in one coat hanger. As the resolution increases, more columns are placed. The result is that each element simulates a smaller part of the tool. Because approximations in geometry and flow behavior are made, this affects the results of the simulation. Higher resolutions will conceive more accurate results, but it will take more time to process since the number of devices increases. There is a minimum of six columns for each coat hanger in order to properly represent their geometry.

The channels in the coat hanger are represented by devices along its length while vertical devices simulate the flow of material out of the channels and into the land region. In this last section, the flow is mostly in the vertical direction, thus three vertical elements were placed and just one line of horizontals.

The number of columns can, and most times will, be different in the coat hanger and the mandrel section. That’s due to the minimum of columns or the necessity of a multiple value depending on the number of coat hangers. In order to change from one to the other, a transition line is needed. The number of devices in this line is the sum of the columns from both sections. Because the amount of columns in both areas are not necessarily multiple, this transition creates a distortion in the distribution profile, even if it is already homogeneous. To measure this effect, simulations with a simple circuit of a possible transition were made.

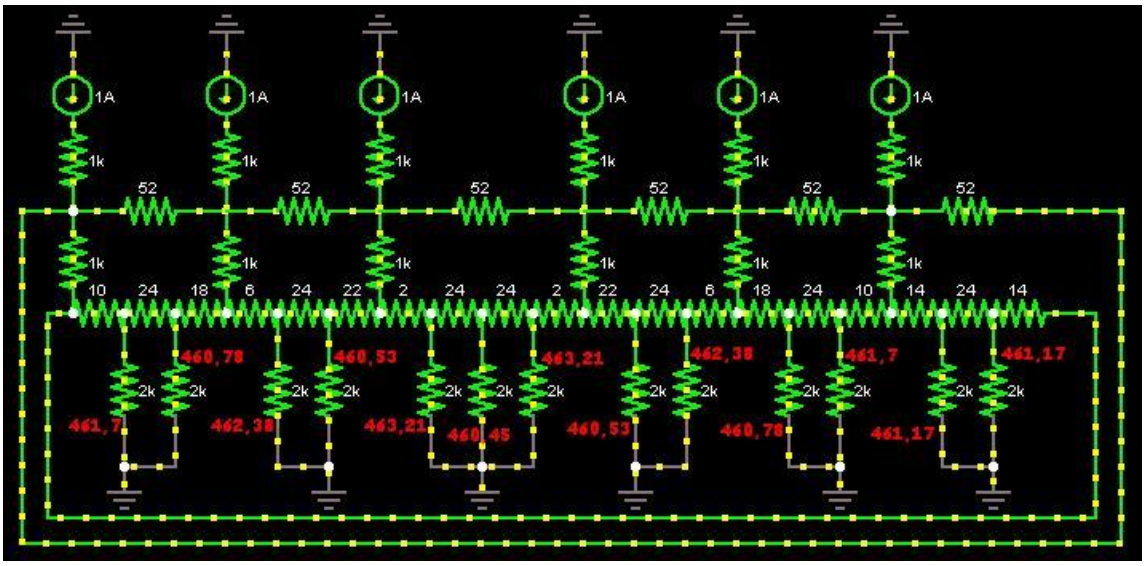

Figure 18 - Simple transition simulation.

In this case a uniform profile was simulated on the input. The resistance values in the transition line are proportional to the distance between the two terminals that they are connected to. The output expected in a perfect transition with no distortion would be 461.54 mA in each device, which is the total input of 6 A divided by 13 output links. What the simulation demonstrated were multiple values in the range between 460.45 and 463.21 mA. Although the distortion was under 1%, a simple change can greatly narrow it down. Adding another horizontal line after the transition allows the current to seek a more homogeneous profile. As a result, the output range changed to 461.52 and 461.56 mA.

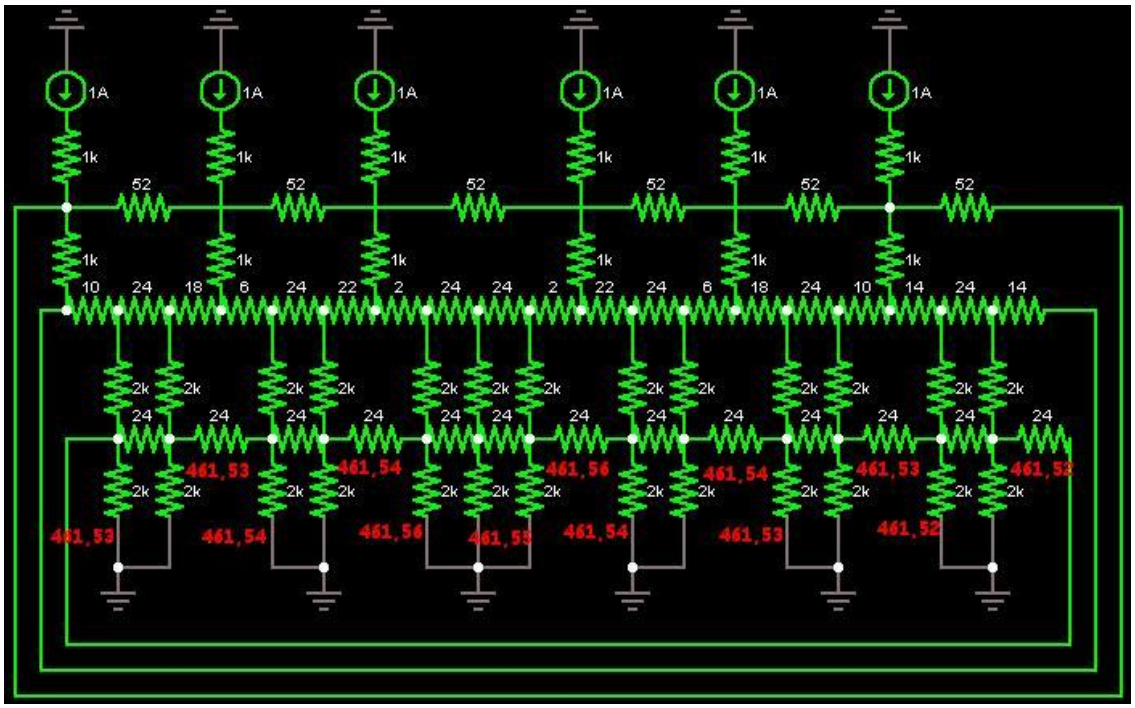

Figure 19 - Simulation of transition with horizontal line following.

The mandrel section continues with vertical and horizontal devices but also adds diagonals required for the spiral channels. Although in the sketch the columns and spirals don't have constant spacing and angles, they do have in the software. All the columns, lines and spirals are equally spaced. The angles of the diagonals are also constant. The pattern repeats horizontally, each time a new spiral starts, and vertically. When a spiral crosses a vertical line between two nodes, one additional node and device are created in order to properly place the spiral's terminal. At the end of the spirals, depending on their enlacement, they do not finish exactly on a column. Again there's the need to create another node and device, this time on the horizontal line, to accommodate the spiral properly.

The outlet area is the last one, and it consists of a fixed design with three lines of both vertical and horizontal elements. This gives the possibility for the material to accommodate and distribute evenly along the whole tool creating a homogeneous profile.

At the end of the implementation, a graphic showing the volume flow distribution immediately before (red) and after (green) the transition was plotted (Figure 20). In a lot of cases the difference between the two lines was very significant, and the line after the transition was closer to the one at the end of the tool

(blue). This occurred because instead of just sampling from the line as we had projected, all the melt is actually forced to pass through the transition elements influencing deeply the previous profile.

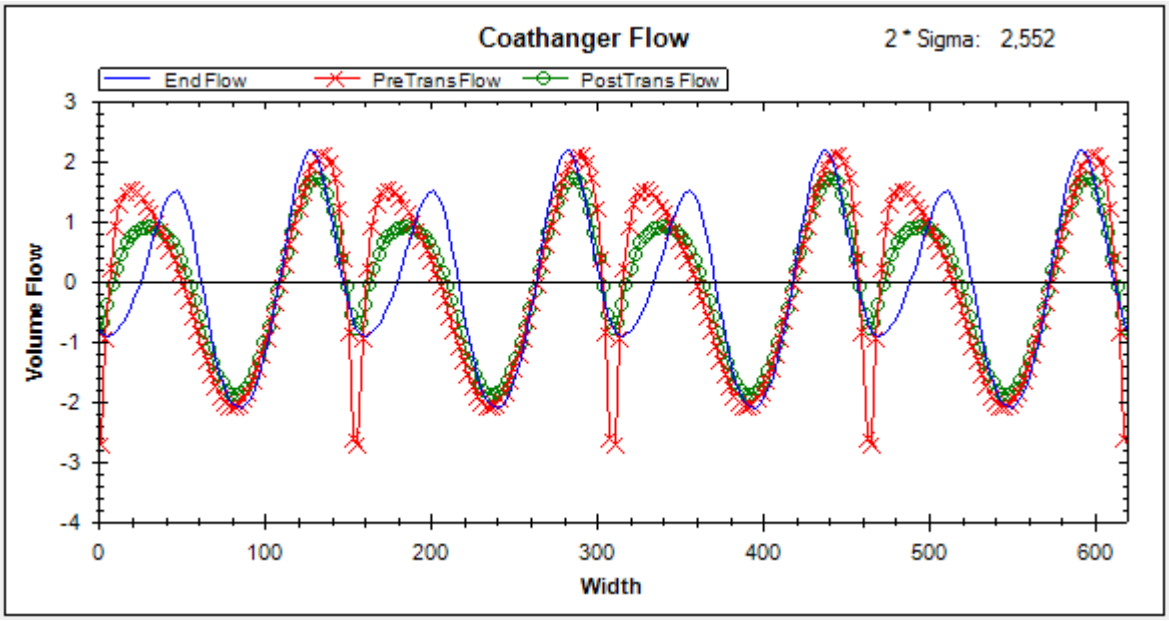

Figure 20 - Volume flow before and after the transition.

3.2.2: Model without transition

This model (Figure 21) is similar to the previous one, but the network of the manifold section was reduced to increase performance, the transition was removed and the entire following network was adapted to the new number of columns. One other key change was at the beginning and end of the spirals. Since now their positions do not match with the columns of the network they had to be extended or contracted to the nearest column. Although this introduces some error, it is expected to be smaller than the one created by the transition line in the previous model.

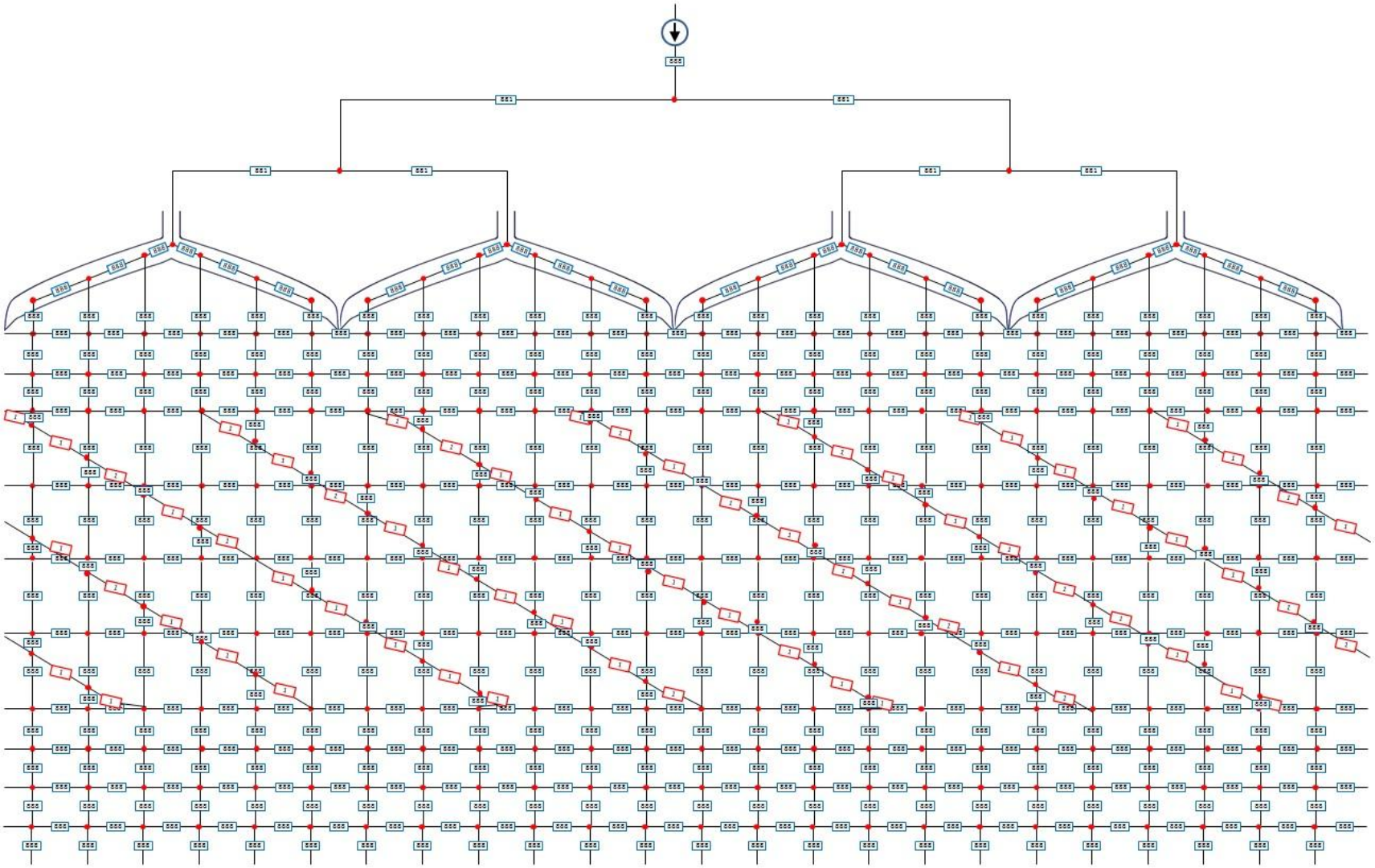

Figure 21 - Network model without transition.

Although visually the changes may seem small, it completely changed all the numbering patterns. This forced the implementation to be made all over again almost from zero. As this is the most complex task and takes months to be completed, it was not possible for me finish it without abandoning the optimization part. Since the optimization is a crucial point to the project and was proposed since the beginning, we decided to interrupt this implementation and proceed with the planned activity. It's important to note that the optimization algorithms to be implemented would work the same for any model, but the results would be different. Therefore, the old model was enough to evaluate most of the behavior of the optimization methods.

Chapter 4. Software development

This project will create a program to be used in the industry of plastic. Although the Institute of Plastics Processing has already developed a program for simulating and designing a tool over the past two decades, it is currently very limited comparing to what can be achieved after all the progress made regarding discretization and simulation in these years. Also the processing speed of the computers increased greatly, allowing them to perform tasks that were impossible in the past.

The software company who supported this project suggested the solution to be implemented using C# and the .NET Framework, despite the old software be coded in Delphi and the adoption of the same language would make it easier to reuse the code. Moreover, the project seeks to create a completely new program, and the Microsoft tools have advantages over Delphi that makes it more suitable for the implementation.

4.1: C# and .NET Framework

The Microsoft Visual Studio together with the .NET Framework and C# language have very good solutions and implementations of structures to help the development of both processing and Graphic User Interface (GUI). The spread use of the Windows Operating System also contribute to create a large community of software developers which makes the amount of available libraries and other shared applications considerably high and very easy to find.

The .NET framework was design to provide [22]:

- Consistent object-oriented programming environment.
- Code-execution environment that minimizes software deployment and versioning conflicts.
- Safe execution of code, including one created by unknown or semi-trusted party.
- Elimination of the performance problems of scripted or interpreted environments.

- The developer experience consistent across widely varying types of applications.
- All communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The common language runtime and the .NET Framework class library are the main resources of the .NET Framework. The first one manages code at execution time. This provides services such as memory, remote and threads management, while also applying strict type safety and other forms of code accuracy that promote security and robustness. The Figure 22 shows the relationship of the common language runtime and the class library to applications such as the one developed in this project, mainly managed, and to the overall system. The solver used in our project is the only part of the application that runs unmanaged code, since it is a third party solver implemented in C and only imported and used by our program.

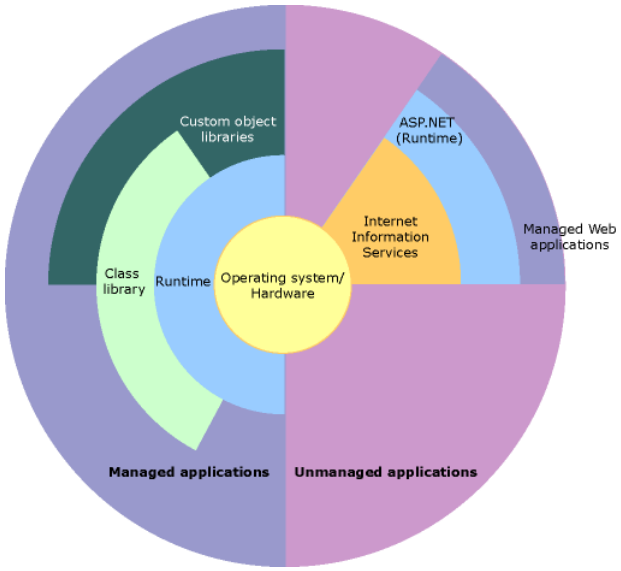

Figure 22 - .NET Framework in context.

4.2: Software modeling

Before starting to implement the program, it is essential that a model is built. This helps to visualize the system at aim, specify structures and behavior, gives a template to guide the construction and documents the decisions that were made.

4.2.1: UML

The most common language for performing this activity is the Unified Modeling Language (UML). Its current form is called UML 2 and specifies thirteen diagrams types shown in the following Table 1.

Diagram	Description
Activity Diagram	Depicts high-level business processes, including data flow, or to model the logic of complex logic within a system.
Class Diagram	Shows a collection of static model elements such as classes and types, their contents, and their relationships.
Communication Diagram	Shows instances of classes, their interrelationships, and the message flow between them. Communication diagrams typically focus on the structural organization of objects that send and receive messages.
Component Diagram	Depicts the components that compose an application, system, or enterprise. The components, their interrelationships, interactions, and their public interfaces are depicted.
Composite Structure Diagram	Depicts the internal structure of a classifier (such as a class, component, or use case), including the interaction points of the classifier to other parts of the system.
Deployment Diagram	Shows the execution architecture of systems. This includes nodes, either hardware or software execution environments, as well as the middleware connecting them.
Interaction Overview Diagram	A variant of an activity diagram which overviews the control flow within a system or business process. Each node/activity within the diagram can represent another interaction diagram.
Object Diagram	Depicts objects and their relationships at a point in time, typically a special case of either a class diagram or a communication diagram.
Package Diagram	Shows how model elements are organized into packages as well as the dependencies between packages.
Sequence Diagram	Models the sequential logic, in effect the time ordering of messages between classifiers.
State Machine Diagram	Describes the states an object or interaction may be in, as well as the transitions between states. Formerly referred to as a state diagram, state chart diagram, or a state-transition diagram.
Timing Diagram	Depicts the change in state or condition of a classifier instance or role over time. Typically used to show the change in state of an object over time in response to external events.
Use Case Diagram	Shows use cases, actors, and their interrelationships.

Table 1 - UML diagrams types [23].

Not all of the diagrams above are necessary when modeling a system or program. The designer should decide which ones are significant. For this project the Use Case, Sequence and the Class Diagrams were selected.

4.2.2: Use Case Diagram

The user should be able to interact with the software by inputting parameters, changing the settings, starting a simulation, analyzing the solution and starting an optimization. These cases are specified in the diagram in Figure 23.

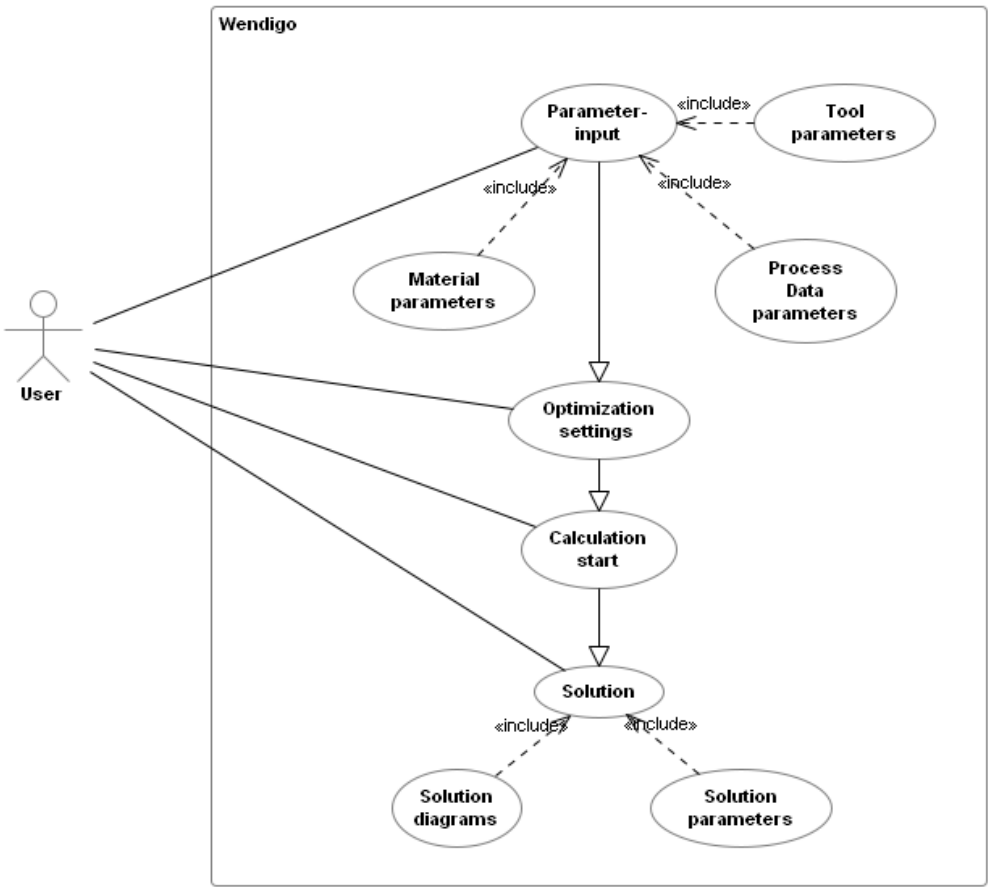

Figure 23 - Wendigo Use Case Diagram.

4.2.3: Sequence Diagram

The sequence diagram in Figure 24 shows the flow of the program and its classes when stimulated by the user. Actions represented are opening program, setting parameters, starting calculation and optimization, viewing solution and closing the program.

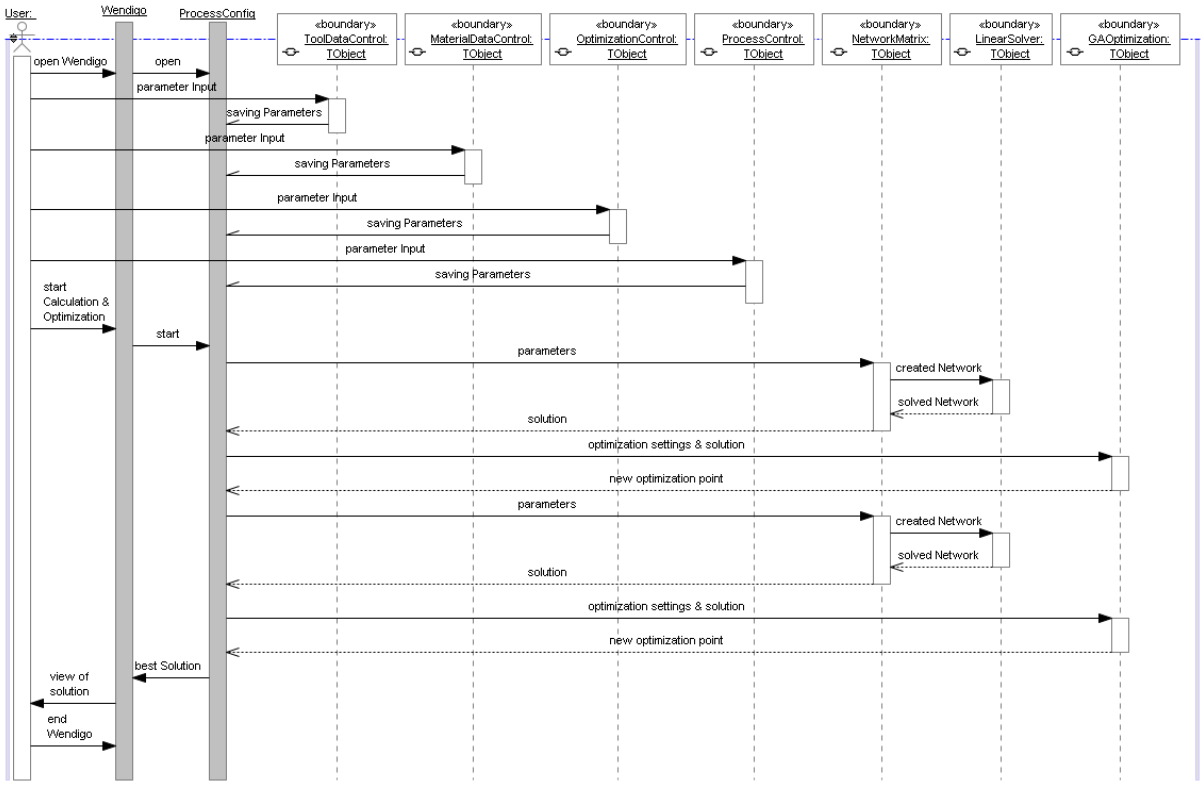

Figure 24 - Sequence diagram

4.2.4: Class Diagram

The great number of input parameters requires a complex GUI, composed of forms and controls, and configuration files to store the data. These three types of classes can be identified by the names already mentioned attached to the end of the names of the classes. Other classes like NetworkMatrix and the LinearSolver are processing classes used during the simulation case. GAOptimization and HillCimbing implement different optimization strategies. The diagram with the main classes is shown in Figure 25.

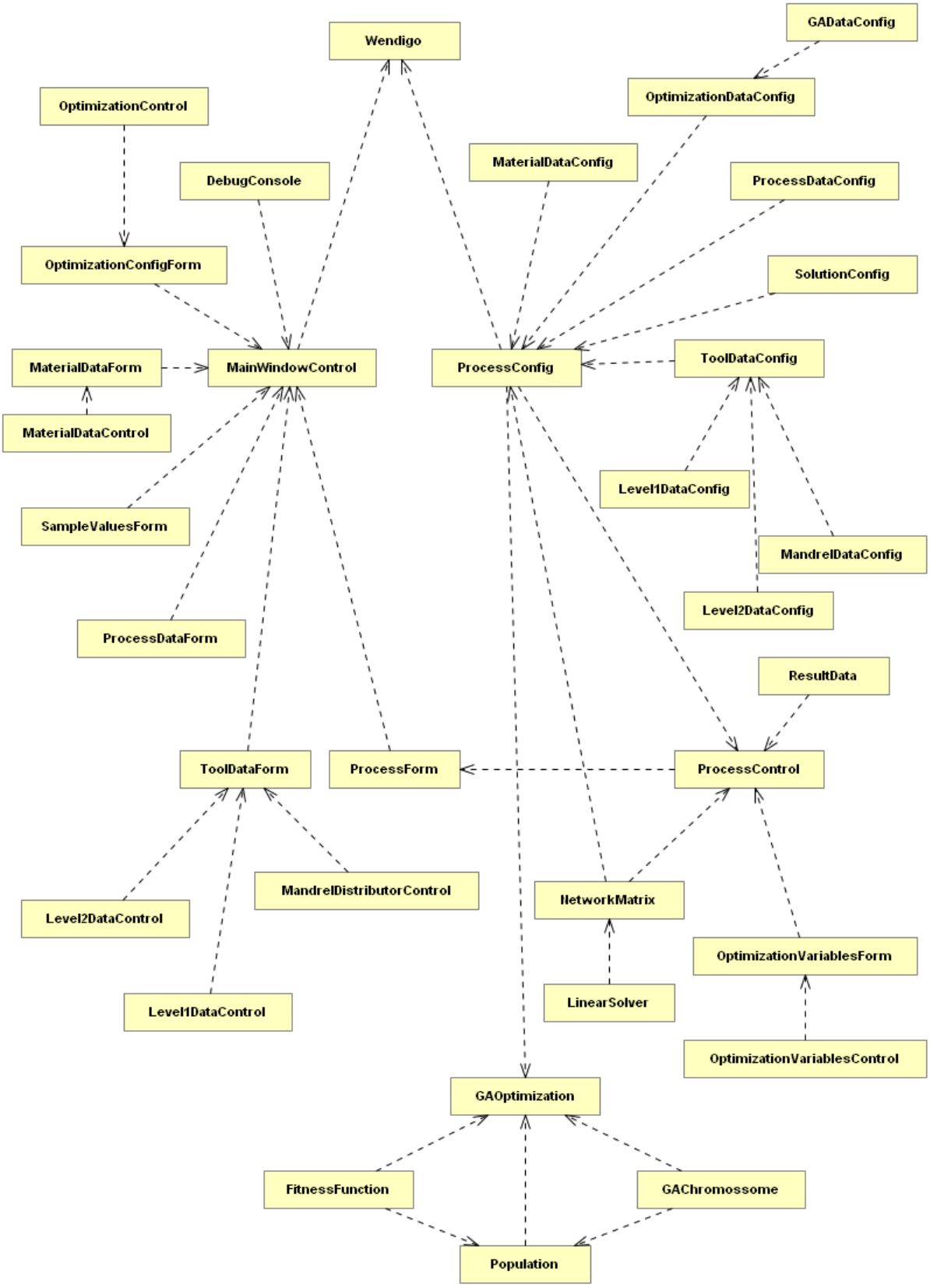

Figure 25 - Wendigo Class Diagram.

4.3: Software architecture

The simulations that will be executed by the user, directly or through the optimizations methods, require many parameters and can take considerable time, especially in the last case which will run several simulations without stopping. To avoid the behavior where the user needs to wait the end of the processing to interact with the program again or even one that there's no way of knowing if the program 'froze' or is still processing, the use of multiple threads is necessary.

The GUI will run on a separate thread, allowing the user to interact with the program by changing parameters, settings, canceling the processing and viewing partial results while the simulation is still running. The changes mentioned would take effect only at the next simulation. It is not possible to change the parameters of an ongoing calculation.

The processing thread reads all the information from the configuration files before starting, and just after it is done it updates the results. In the case of optimization, the results are updated for each solution calculated, not at the end of the whole process. In some optimizing strategies it would be possible to run more than one simulation at the same time with different parameters. This would require creating one processing thread for each simulation. That way, computers with multiple cores could use the hardware available more efficiently.

Four groups of functional classes are present: controls and forms for the GUI, configuration for data managing and processing.

4.4: Graphic User Interface

The GUI was designed to be intuitive to use and provide fast access to all the features needed. The main window, shown in Figure 26, contains:

1. General control menus;
2. Simulation parameters and some numerical results to the right;
3. Tabs that allows the input of parameters necessary to the simulation;
4. Graphic tools that shows characteristics of the simulation;
5. Application log to keep track of the processing;

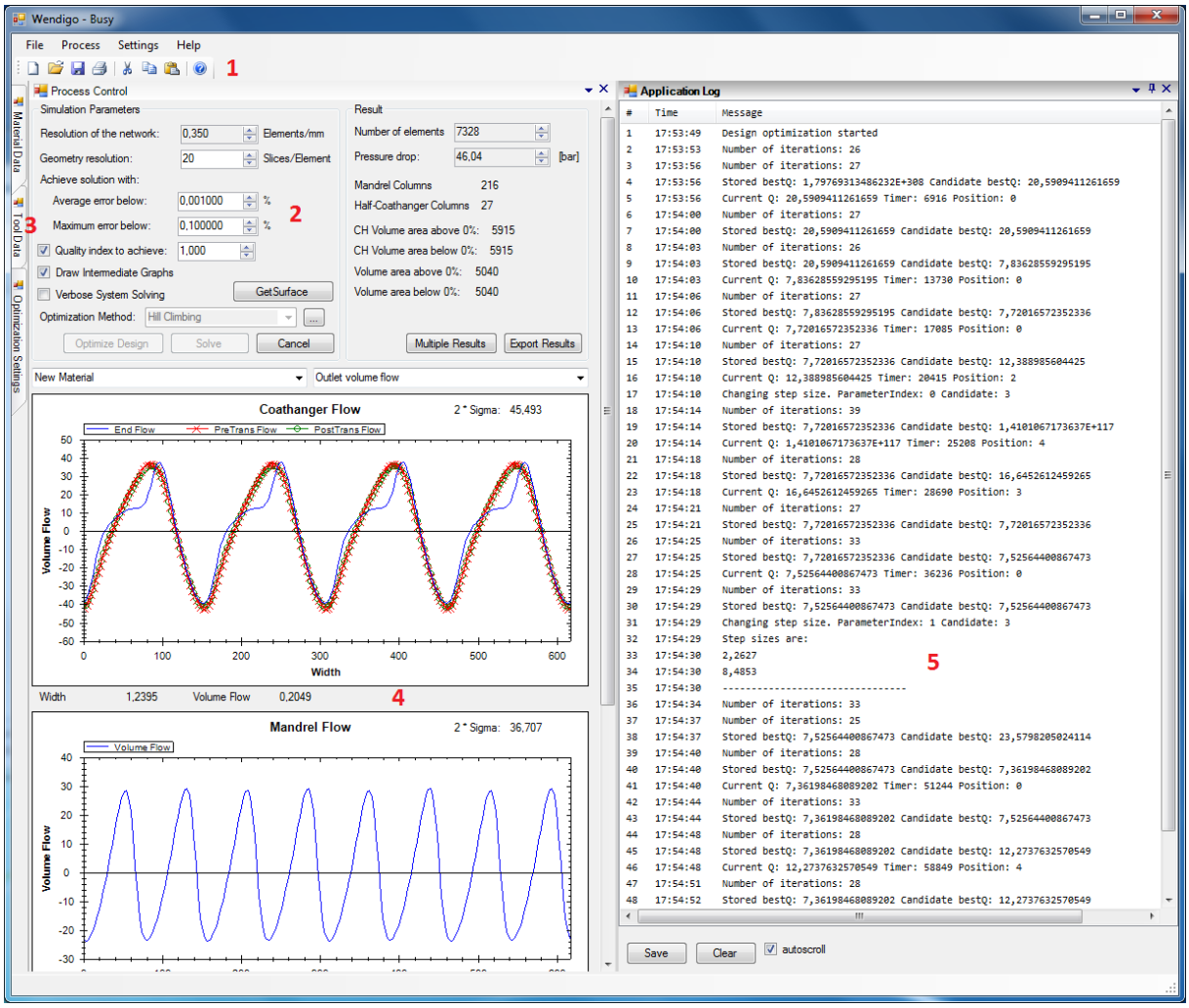

Figure 26 - Wendigo main window.

It was also made in a flexible way, that the tabs can be opened, closed and even pinned to the main window of the program if desired (Figure 27).

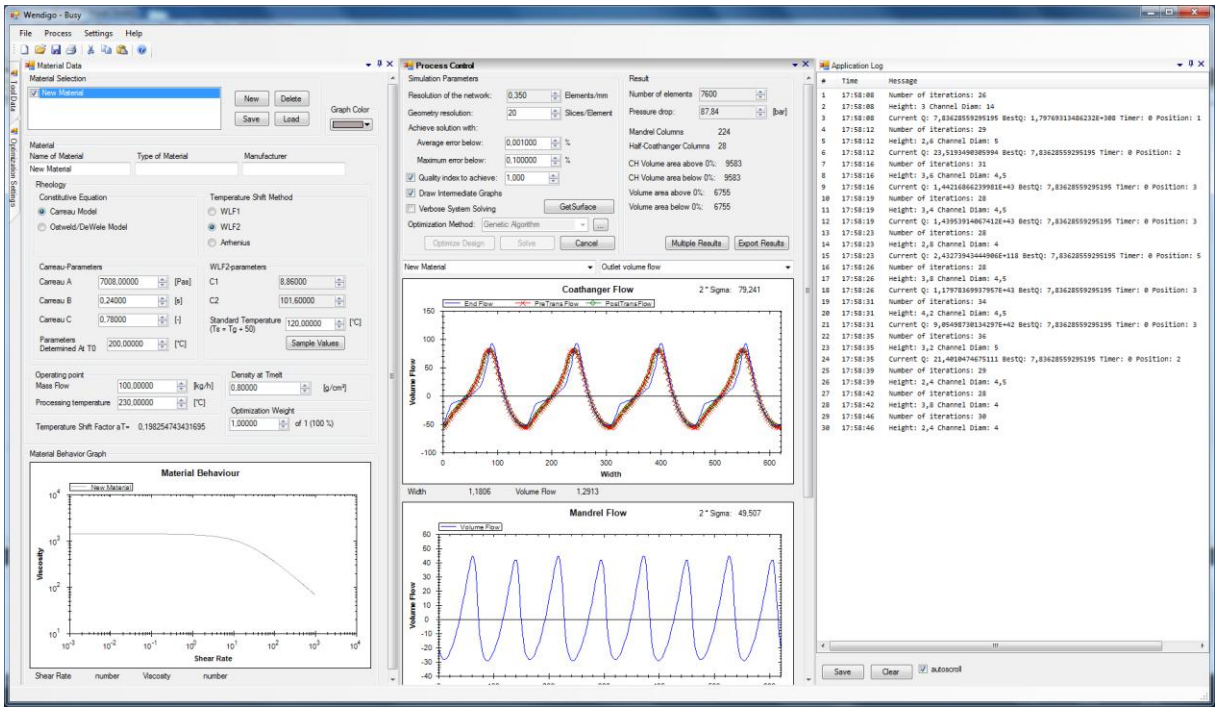

Figure 27 - Wendigo main window with MaterialData tab pinned.

4.5: Data managing

All the input given by the user through the GUI is automatically stored in the configuration objects, which are stored in the RAM memory at first. In order to save the configurations in the hard drive, it is necessary to use the “Save” button present in each tab, or the general one in the upper menu. The files saved in the hard drive have different extensions, so that a material data cannot be loaded later in the tool form, for example. The general file saves all the configurations in the program. The files contain the serialization of the configuration objects. The .NET Framework implements the Serialize and Deserialize methods, which save or load respectively all the class attributes into a file.

During the runtime of the program, each class has access to the configuration object that contains all the information necessary. Most of the configuration classes instantiate only one object and more than one class is able to read and/or write in it. This way it is possible to share data without the use of arguments, although in some cases this is also used. When necessary, it is also used events to trigger the update of a data that is used among more than one class or is conditional to one.

4.6: Simulation processing

When a simulation starts, the program reads the parameters needed from the configuration files, and calculates many variables that will be used later for the construction of the network model. It determines for example, how many columns the mandrel and coat hanger sections will have, how many elements sections and the whole network will be made of and the resistances values that characterize each device.

After that, a routine containing all the patterns necessary to properly connect the devices and create the network starts. Different patterns exist depending on the nature of the object it represents: nodes, links and devices, for example. Besides that, the patterns also changes depending on the region of the tool, such as coat hanger, transition or mandrel. This routine creates the matrices that compose the system and insert the correct numbers at the correct positions regarding both the topology and the characteristic parts of the model. Figure 28 shows a hand drawing of one of the simplest cases possible for a network, with the numberings for devices, nodes and links until the mandrel section. As the resolution and geometry parameters increase, the network can get a lot bigger and more complicated than it already is shown below.

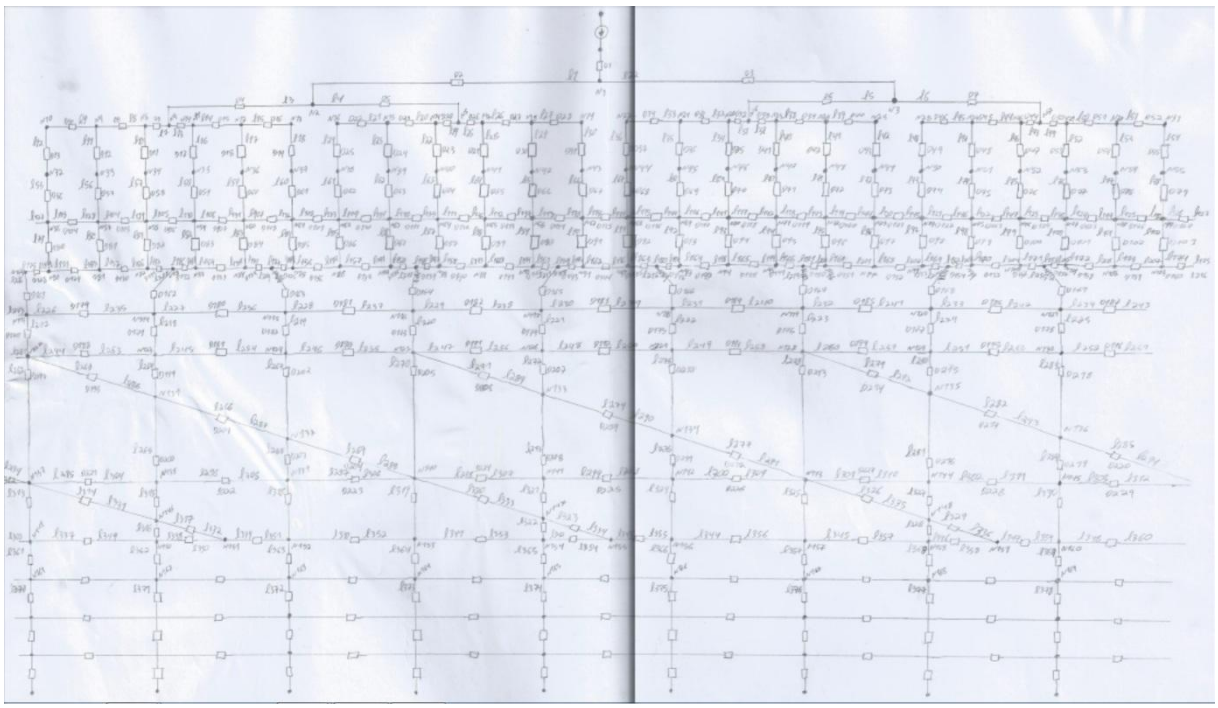

Figure 28 - Hand draw of a network design.

The full square matrix has the dimension of two times the number of devices, which could lead to a very big matrix that would easily exceed the memory available. Since this matrix is very sparse, having in most cases less than 1% of non-zero elements, another representation was used.

The Compressed Sparse Row (CSR) format transforms a matrix of two dimensions into three one dimensional vectors as seen in Figure 29.One array (a) stores all the non-zero values of the original matrix. The second (col) saves the column where the element belongs. Both of them will have a dimension equal to the number of non-zero elements in the matrix. The last (rb) contains the position of the elements in the first array that a new line begins. This vector has a size equal to the number of lines in the original matrix plus one. Thus, two to three values are necessary to represent each non-zero element in the matrix. For this reason the matrix needs to have more than 50% of null values to benefit from this representation. That is not the case for the example below, but it is just for illustration purpose. As mentioned earlier, the matrices used in our model have more than 99% of null values.

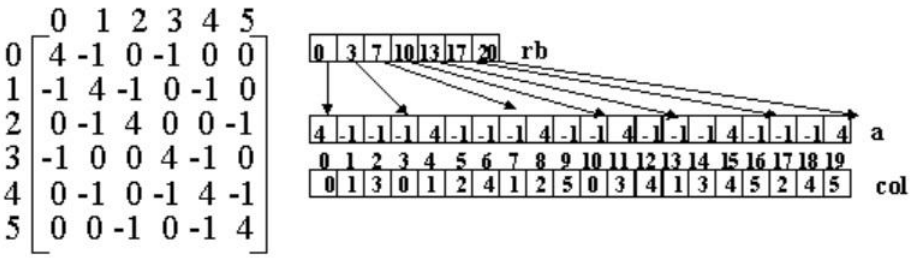

Figure 29 - CSR format example [24].

After the matrices are all built, the processing starts an iterative loop to find a solution to the system. These iterations are necessary because, although the system sent to the solver has a unique solution, the values used to describe the system are not known and have to be guessed at first. The resistances values depends on the volume flow inside that element, which is also what we are trying to calculate. So for the first iteration the volume flow is guessed as an average value calculated by the division of the total input amount by the number of existing elements. The result of this iteration is then used to calculate new resistances for the next iteration and so on. When the difference between the result and the input value is smaller than a specified acceptable error the solving has converged to the solution. The results are

then saved, used to calculate other evaluative information, plotted in graphics and displayed to the user.

All the matrices are passed to a solver that is an open source solution implemented in C and highly optimized for sparse systems. Comparisons were made with the MATLAB and this last one took considerably more time.

Chapter 5. Optimization

The die design process optimization is the main feature of this project. The combination of great number of geometry details, and therefore large quantity and range of input parameters (Figure 30), creates thousands of possible tools. For example, if 10 parameters are selected to be optimized and each has 100 possible values there are 1e+20 different combinations. Together with the complexity of the material flow behavior they make the prediction or analytical calculation of the best geometry nearly impossible. To achieve the desired result the engineers often have to simulate hundreds of geometry combinations manually introducing changes in the input parameters of simulation software. This is a time-consuming task that will occupy an employee with the basic task of changing values and starting the optimization. While the experience of the professional will influence the time spent and the quality of the result achieved, it will still take quite some time.

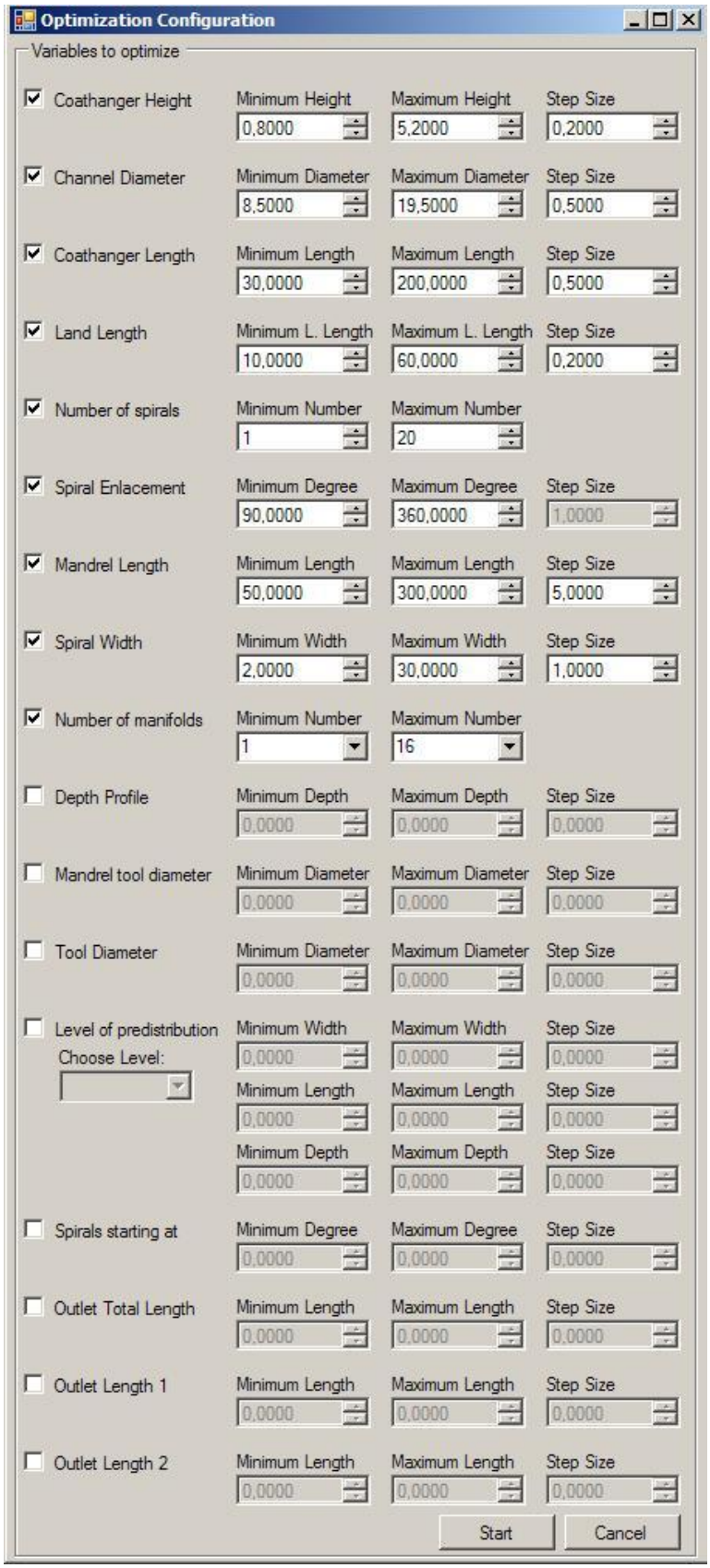

Figure 30 - Some of the possible parameters to be optimized

The processing power of the computers increased greatly in the last couple of decades allowing repetitive and high complexity tasks to be performed in a very short time. Having both characteristics, the optimization methods were also largely applied in many different industrial fields. Their goal is to use the computer to search and find the best solution for a certain problem. This will free the employee who was assigned for this task to use its intellect and skills in a more challenging task.

5.1: Optimization Parameters

The function that will be optimized is the quality index, which is simply called Q. Its value is calculated after each simulation is completed and depends on three characteristics of the solution: the volume flow distribution at the outlet, the shear rates along the various paths of the tool and the total pressure drop. For each one a specific quality value is calculated and later all three values are combined to achieve the final Q. This calculation is simply a weighted average with the user defined values for the weights. For all these functions the lowest values (positive close to zero) are the best, while high values are worse [8].

There is also one important restriction to the parameters that will be used to modify the geometry. Since the goal of this procedure is to find an optimum geometry that can be manufactured, the parameters cannot assume all the real values. First, they will have to be positive as they all measure or count a physical dimension or feature. Then a discrete vector of possible values is calculated depending on the user input for minimum and maximum value and step size when available. This will allow the user to guarantee that the final geometry will be able to be manufactured. For example, the diameter precision of the drilling tool used to create the spirals could be 1mm. In this case, it would be useless to make any change in the parameter below that value. Other parameters such as number of spirals have a step size of 1 since a fractional spiral would only create asymmetry in the tool, which is not wanted.

5.1.1: Volume flow quality index (Qv)

The volume flow distribution at the outlet is one of the most obvious quality parameters since the main task of an extrusion die is to deliver a plastic melt with an uniform melt distribution.

At the end of the simulation three graphics are plotted showing the difference in % values of the simulated volume flow at the end of each section of the tool and the theoretical average. An example is shown in Figure 31.

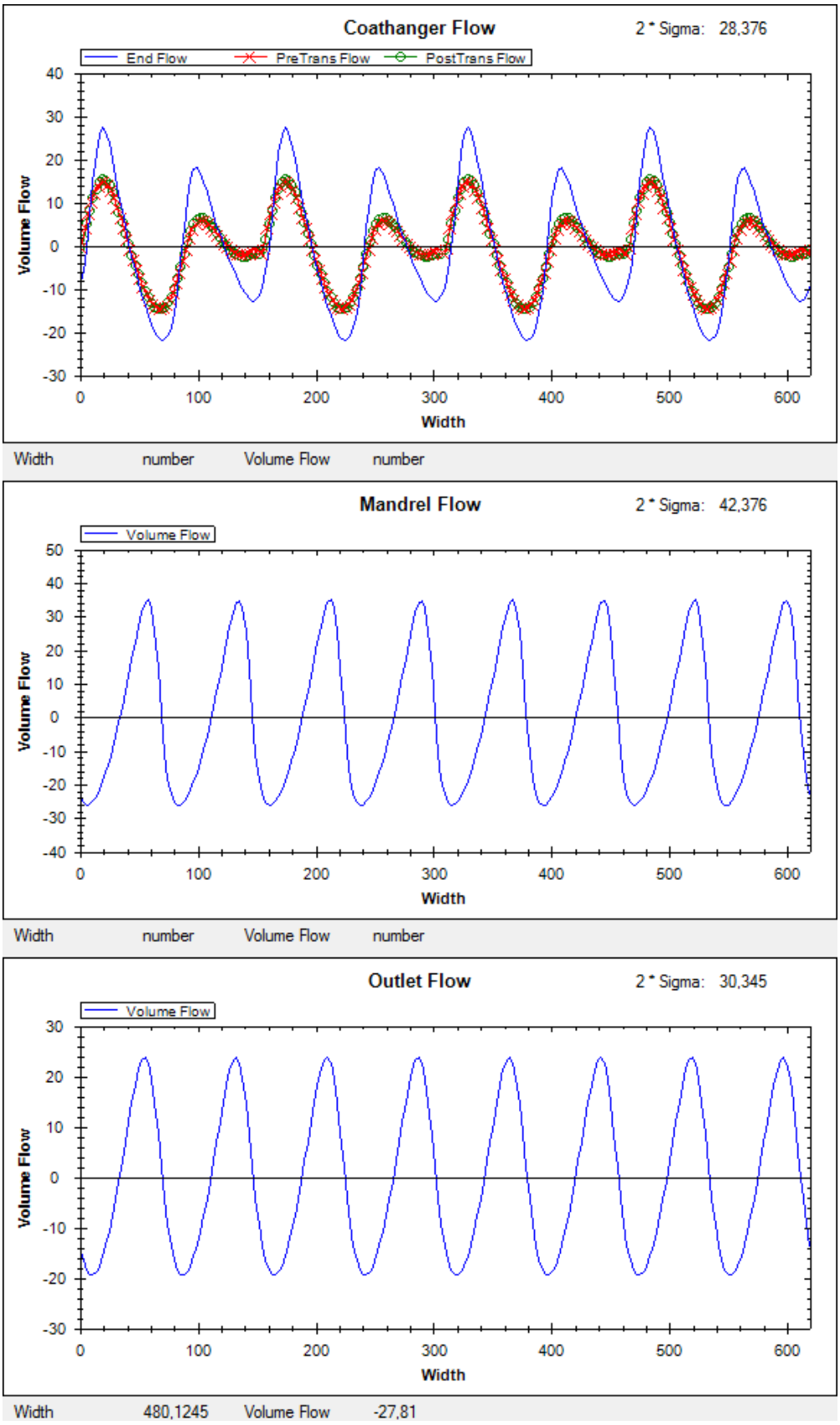

Figure 31 - Volume flow graphics.

From this graphic it's calculated the divergence parameter, which is the absolute difference between maximum and minimum divergences of the outlet graphic.

$$divergence\ parameter = |maximum\ divergence - minimum\ divergence|$$

This value is then normalized by a linear function to achieve the final Qv value. In this case, a 2% difference was set to a Qv of 1 resulting in the following equation:

$$Qv = \frac{divergence\ parameter}{2}$$

5.1.2: Shear rate quality index (Qs)

Taking the local values of the shear rates into account is critical since the shear rate strongly influences the flow behavior of the thermoplastic melt. Regions in the extrusion die that exhibit reasonably low shear rates tends to have material sticking to the walls, resulting in change of the flow behavior and long purging times while changing materials. On the other hand, high shear rates could lead to degradation of material.

The Qs is calculated based on the difference between the maximum and minimum value found in all regions of the tool and the ones specified by the user. Figure 32 illustrates one example.

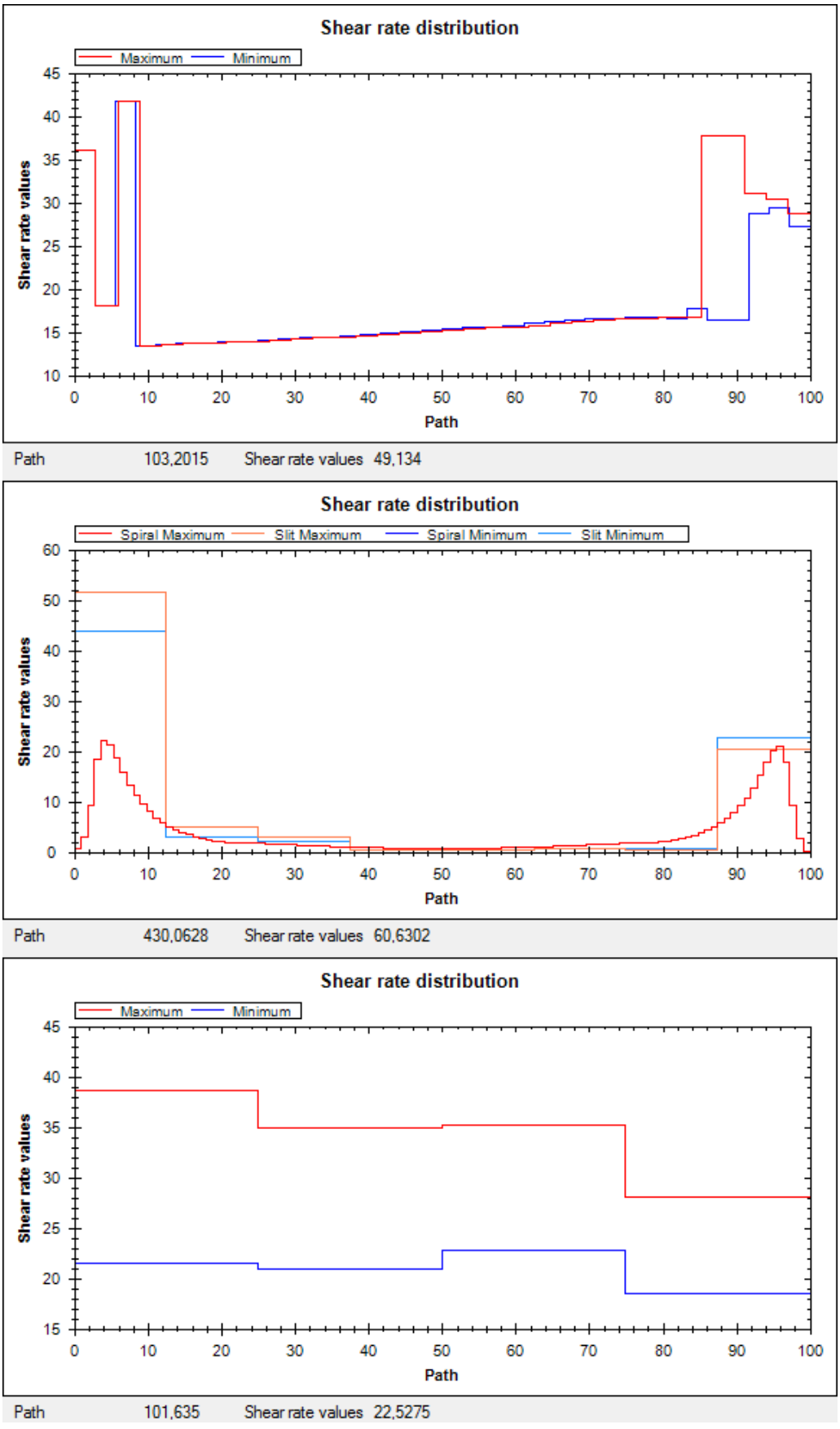

Figure 32 - Shear rate graphics.

There's also a penalty function Bs that is multiplied to the Qs. If the values are inside the accepted range, the Bs will be 1 and a linear function will normalize the final Qs value. An inhomogeneity of 25% is set to a value of 1.1 as 0.1 is added to avoid loss of information.

$$Qs = 0.1 + 4 * \frac{\dot{y}_{actual,max} - \dot{y}_{actual,min}}{\dot{y}_{boundary,max} - \dot{y}_{boundary,min}}$$

In the other case, the penalty function returns a very high value degrading the Qs to express the result out of the acceptable conditions. The final value is:

$$Q_{\dot{y}} = Qs * Bs$$

Where:

$$Bs = \begin{cases} 1, & \dot{y}_{actual,max} \; and \; \dot{y}_{actual,min} \; \; inside \; boundary \\ 1 + e^{\dot{y}_{actual,max} - \dot{y}_{boundary,max} - 5}, & \dot{y}_{actual,max} \; \; outside \; boundary \\ 1 + e^{\dot{y}_{boundary,min} - \dot{y}_{actual,min} - 5}, & \dot{y}_{actual,min} \; \; outside \; boundary \end{cases}$$

5.1.3: Pressure drop quality index (Qp)

The overall pressure drop is necessary to overcome the flow resistance of the extrusion die so that the material can be processed at the chosen processing temperature with the desired throughput. The upper boundary is set because there's a limit to the Δp_{max} that a melt pump can deal with.

The total pressure drop is calculated by the difference of pressure at the beginning and at the end of the tool. Similarly to the previous one, if this value is inside the acceptable range specified by the user a linear function is applied. Within a certain range above the maximum value a more inclined (with higher values) linear function is applied. This is due to the fact that this parameter is usually chosen with a safety margin. After this margin is exceeded, the function is exponentially increased.

$$Qp = \begin{cases} \frac{\Delta p}{\Delta p_{max}}, & \Delta p < \Delta p_{max} \\ \frac{\Delta p - \Delta p_{max} + \Delta p_{max,exceeding}}{\Delta p_{max,exceeding}}, & \Delta p_{max} \leq \Delta p \leq \Delta p_{max} + \Delta p_{max,exceeding} \\ 1 + e^{\Delta p - \Delta p_{max} - \Delta p_{max,exceeding}}, & \Delta p \geq \Delta p_{max} + \Delta p_{max,exceeding} \end{cases}$$

5.2: Quality Surface

To exemplify the result of the combination of the three quality parameters described above, a 3D surface (Figure 33) was calculated by varying only two parameters: the height of the slit in the coat hanger and the diameter of the distribution channels. For better visualization the plot was made with the Z coordinate expressing 1/Q so that the top of the hill expresses a good quality value.

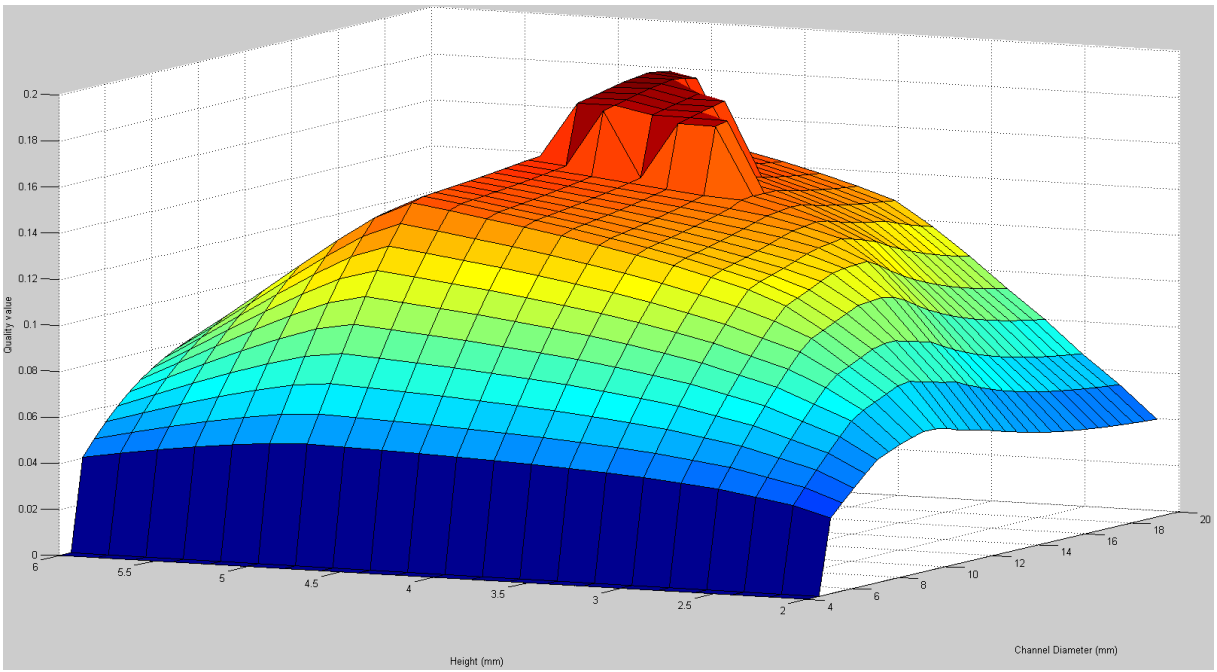

Figure 33 - Quality Surface

5.3: Optimization Algorithms

There are many optimization methods available that can be used to achieve the goal set for this project. Each one will have different characteristics and procedures that may lead to equal or different results depending on the situation. There is no best method for all the cases, and because in this project the number of parameters is large and their influence very different the implementation of a few methods will be made so that a proper comparison can be made.

Independent on the method and the parameters that guide them, there is always a trade-off between time consumed and significance of the solution found. It's important to note that many methods could converge to a solution that is a local best but not global. As more time is spent, i.e. more different solutions are evaluated, the

chance of finding the best global value increases as more points within the same possible candidates are contemplated. Thinking to one extreme, if all the possible combinations are considered then the best global result will be found with 100% confidence. On the other hand, that's what the optimization methods avoid as in most cases this approach would take too much time. Usually each method has its own parameters that can be adjusted to change the behavior of the algorithm to be faster or more significant.

The algorithms can be classified into two different groups according to the nature of their progress: Deterministic and Stochastic. Figure 34 shows the classification of a few methods into these two groups as well as some more specific. In the first group, given a certain function to be optimized and a starting value, their progress will always be identical. This is good because the results are constant for a certain starting condition and can always be reproduced if desired. On the other hand it is susceptible to the starting point, so for a different one the results could be completely different. The second group behaves the opposite way. There is always randomness in the process that makes each optimization unique and may lead to different results even for the same starting condition.

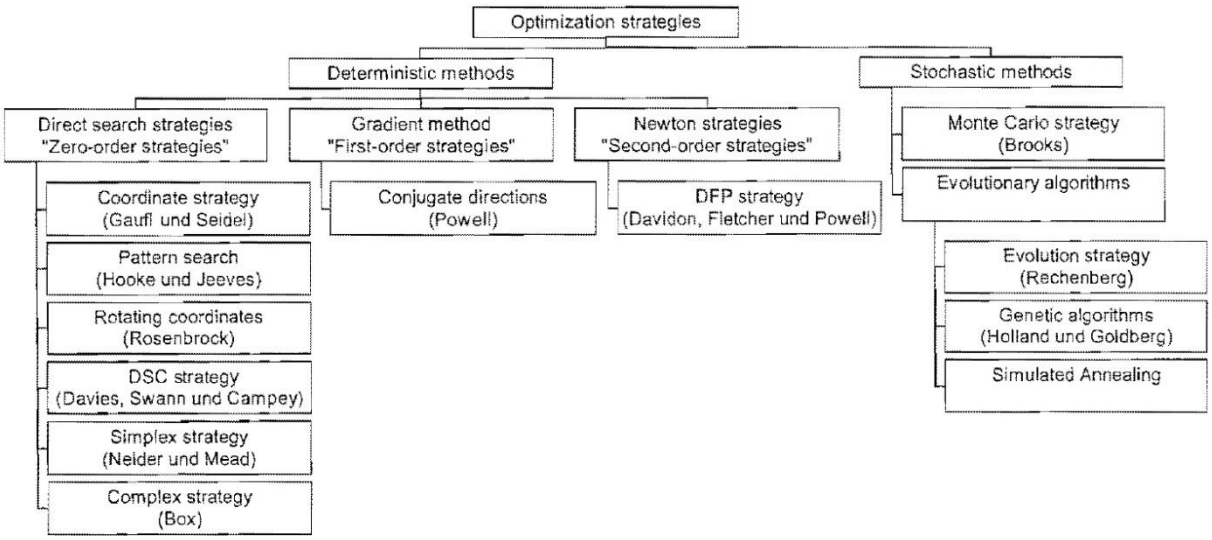

Figure 34 - Optimization strategies [9].

The optimization algorithms can only be implemented in the context of the project after the simulation module is working because they need the quality indexes calculated to use as the parameter to be optimized. Due to that restriction, their development was scheduled to the end of my participation in the project. The problems encountered in the first network model and the necessity of creating and

implementing a second model had an impact on the schedule and the time available for the optimization was reduced. Therefore only two strategies will be developed and implemented at this point. To explore the difference between the two groups one of each was selected. The Hill Climbing method, which is a variation of the Coordinate strategy [10] listed in Figure 34, was chosen from the deterministic group while the Genetic algorithm from the stochastic.

5.3.1: Hill Climbing

In this method there are basically two loops: the outer loop will run all the calculations until one of the stopping conditions are reached. They can be convergence of a solution, maximum time exceeded or maximum number of iterations achieved. So the main function of this loop is check these conditions at the end of each iteration and run the method again or exit and display the results. The inner loop is where all the logic of the method is.

First the algorithm creates a number of candidates, in this project set to 5. These candidates are responsible for evaluating different combination of parameters. In each loop only one parameter will be modified in the candidates. The changes depend on two parameters that can be set by the user: acceleration and spread rate. The acceleration is a value larger than one and controls how fast the algorithm will converge. As explained before the speed also causes low significance, so it's important to find a balanced value. The spread rate indicates the distance, between the two extreme candidates, as the percentage of the range of possible values for each parameter. The five candidates are calculated as:

Candidate 1: initial value – acceleration * step size

Candidate 2: initial value – step size / acceleration

Candidate 3: initial value

Candidate 4: initial value + step size / acceleration

Candidate 5: initial value + acceleration * step size

This means that the initial value, which is always the best value of the last iteration or the user input value in the first iteration, will be the center point. Two values will be symmetrically created to each side. To keep the points 2 or 4 at the

same distance from the points 1 or 5 and the center, the acceleration (Figure 35) was set to$\sqrt{2}$. This was done to minimize the distance uncovered by any candidates.

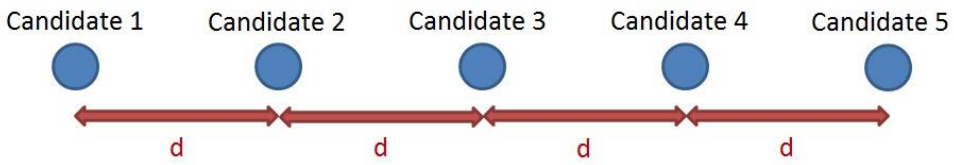

Figure 35 - Candidates illustration

The step size is calculated at the first iteration as:

$$step\ size = \frac{spread\ rate * range}{2 * acceleration}$$

This will guarantee that the candidates 1 and 5 are distant by the spread rate * range. The simulation is then run for all the candidates and their Q values compared. The candidate that present the best Q value will be used for all the other parameters optimization and it will be the next initial value if another iteration of the optimization is required. The algorithm then proceeds to the next parameter to be evaluated calculating all 5 candidates and repeating the procedure described to this new parameter. This is done until all the parameters selected by the user to be optimized are tested.

After each iteration the step size is modified depending on which candidate had the best quality index. If the central point was the best the step size is multiplied by a factor of 0.75. This will place the new candidates 1 or 5 in the middle of the old ones and 2 or 4. In case the best candidate was 2 or 4 the step size is divided by the acceleration, and for 1 or 5 it is multiplied by the same constant. At some point all the step sizes will be smaller than the minimum change allowed to each parameter. In this case the algorithm has converged to the optimum solution.

An optimization with the mentioned parameters was performed for the surface already shown previously. The Figure 36 shows the points evaluated during the process. Each color represents a different iteration of the algorithm, although there are some iterations with the same colors and points that were present in more than one iteration but are painted only with the color of the last one. It's easy to see that the algorithm converged to the top of the hill and explored most of its points.

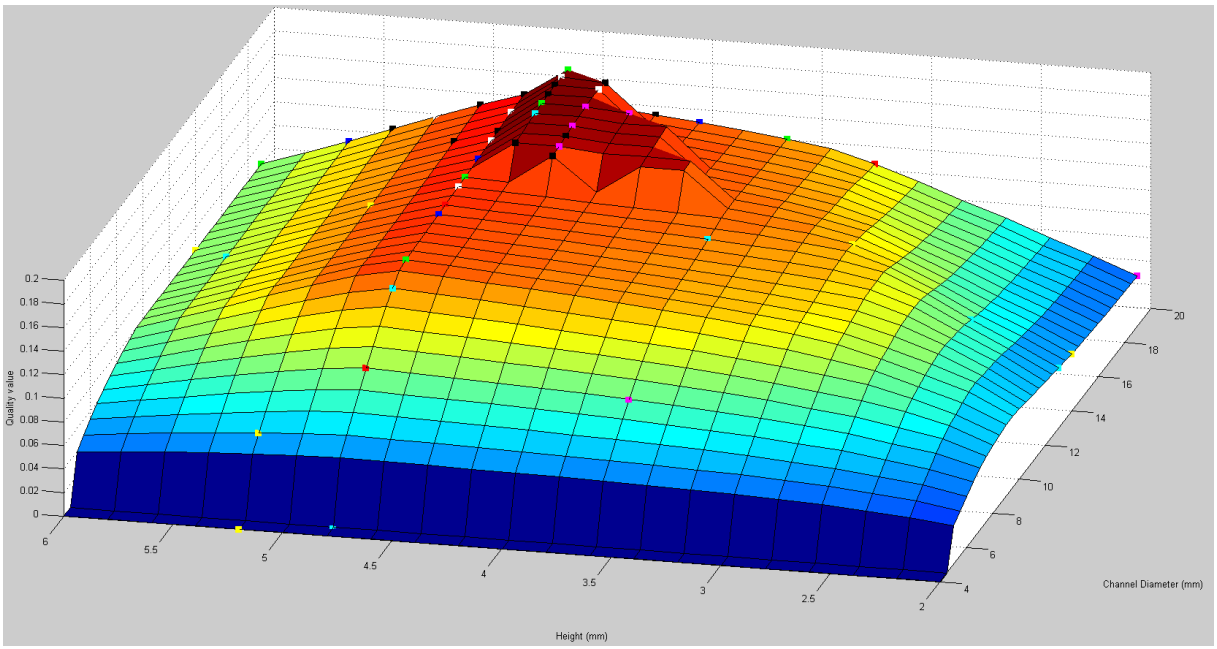

Figure 36 - Hill Climbing Optimization

5.3.2: Genetic

The evolutionary methods are based in the biological evolution concept. Certain principles present in the original process were transformed to fit the engineering context (Figure 37).

Principle	Biology	Engineering
Replication	Identical duplication of the genetic material	Copying an existing parameter set
Mutation	Random, undirected modification of the genotype	Random, undirected modification of the parameter set
Birth	Formation of a phenotype	Selecting the new parameter set
Life	Survival of the life form in the environment	Evaluating the quality of the selection
Selection	Selection of the most well-adapted organism and death of the less well-adapted individuals	Comparing the qualities of the parameter sets and discarding the parameter sets with the lower qualities

Figure 37 - Biological principles in engineering optimization context [11].

The basic idea of the algorithm is to apply random changes through mutation and crossover and then select the best. To start, a population with a size specified by the user is created randomly. Each individual has a different set of parameters.

The parameters are coded all together in a binary representation (see Figure 38). This way, a long vector of binary values is created for each member of the population simulating a DNA code. Mutations will happen to this code randomly by changing one or more values in it. This way the population has already changed. Next it's applied the crossover, when the codes from two distinct parameter sets are exchanged by a random fraction. From this increased new population, the best codes are selected to serve as 'parents' to the next generation while the rest are discarded. If the goals of the optimization process are met the method is finished.

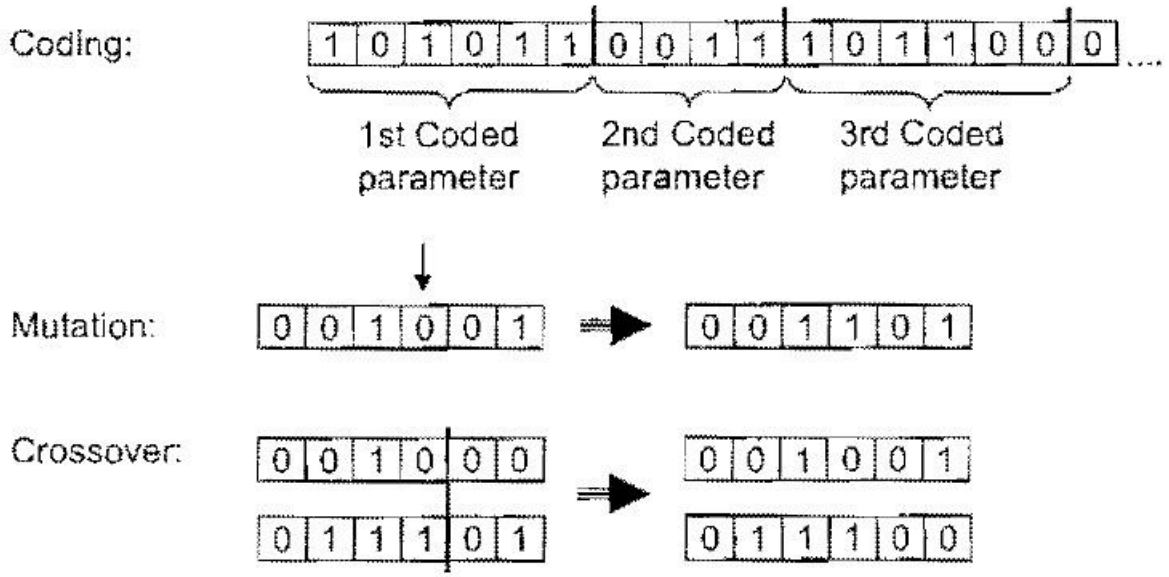

Figure 38 - Genetic coding example [12].

The parameters that specify the progress of this method are: the size of the population, the method for selecting the 'surviving' points, the mutation and the crossover balance rates. A higher population will lose speed to benefit a better significance. The selecting method controls the randomness of the progress, as an Elite will select only the best, Random will select regardless the fitness value and Rank will perform a mix of the previous. The mutation and crossover balances will control how much each of them will occur. A high mutation is more likely to promote bigger changes in the population, while the higher crossover will generate more children close to the population that already exists.

An optimization with a population of 20 individuals, selection method of Elite, mutation and crossover balance rates both at 0.5 was performed. The graphic in Figure 39 shows all the points evaluated during the process. It's evident that it spreads along the entire surface much more than the hill climbing. The top of the hill was not as explored because this is a slower method, and to compare this aspect the same time limit of 6 minutes was imposed to both.

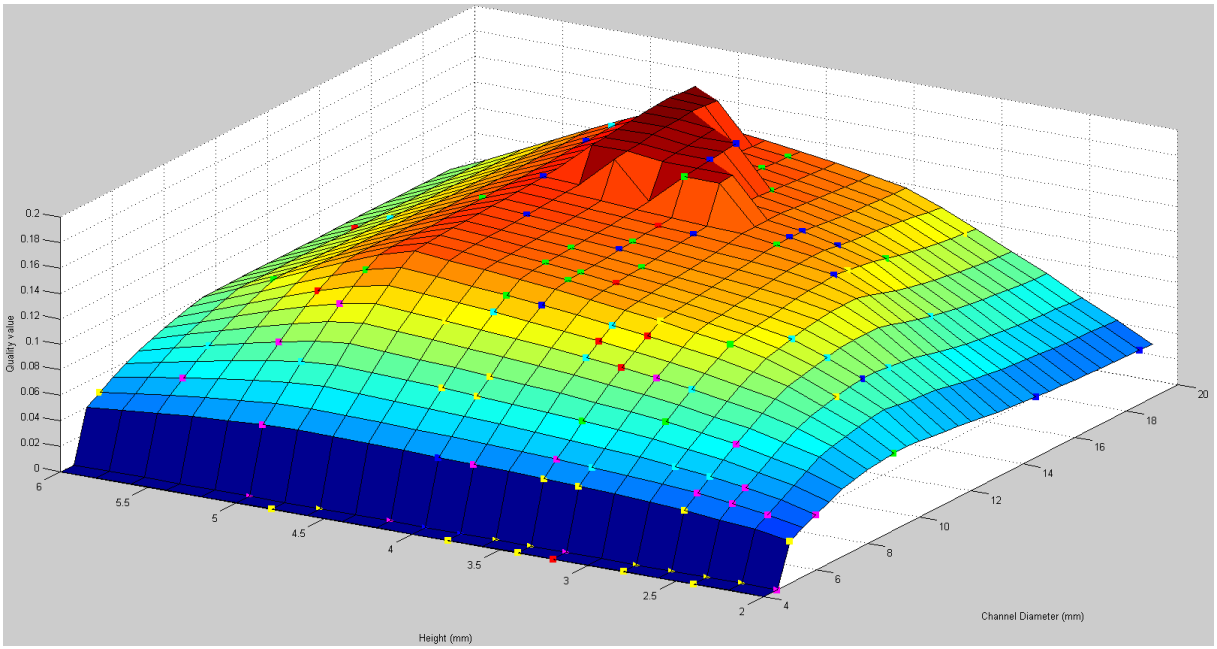

Figure 39 - Genetic Algorithm Optimization.

Chapter 6. Results

The network design implementation, required for the software processing, was divided into five parts and planned to be implemented gradually. Following the sequence of implementation planned, the parts are: coat hanger, pre distribution, transition, mandrel and outlet. This way, it was possible to perform the simulation completely and get results, although using an incomplete model. This choice was made in order to validate the algorithm implementation, circuit analysis method and the third party solver in early stages of the development.

The optimization methods implemented, Hill Climbing and Genetic, were studied and compared regarding their differences, strengths, weaknesses and their parameters' influence on its behaviors. There are still more studies to be made in these aspects as I had little time left to conduct extensive studies. Also new modifications and way to apply them can be implemented, besides new optimization methods.

6.1: Analysis of the simulation results

After implemented the first model test were made to validate the result. Methods and graphics to show the results of the simulation were made to support this process. As mentioned, the first model proved to have higher distortion than expected mainly because of the transition line that didn't behave as projected. A second model, without the transition, was proposed and started to be implemented but it was not finished due to lack of time. This will be the first activity to be continued by the next person involved in the project, as it is crucial to all the rest of the development.

The behavior of the modeled process was firstly analyzed by the influence that the parameters made in the final result when a change was applied. For example, if the height of the coat hanger is increased and the channel diameter decrease by a good amount, the material is expected to flow mostly through the middle. In the opposite situation it would be expected to flow through the borders. This behavior was not observed in the first model as significantly.

The accuracy of the results can only be obtained after comparing with a measurement of a physical process, which will be done in the future when the network modeling is complete. For the physical measure the profile of extrude will be divided into several parts, their weight measured in a high precision balance and then compared to the simulation values.

The matrix generated by the algorithm implemented was checked through another auxiliary algorithm, which was easier to debug, through worksheets exported to Excel files and through handmade network ones, with higher reliability. This method is working properly, although in the optimization phase with the introduction of more geometry tests a few bugs were still found and corrected.

Comparisons of the results were also made between this and the old software, which has been used for years in industry. They were different from each other, supporting the perception that the model is not behaving as desired. Since they have processing differences regarding the models, a different result was already expected, but they should still be close to each other as the old software is a reliable tested one.

6.2: Performance

The performance of the program exceeded the expectations. The number of iterations required to achieve a result within the specified condition were between 35 and 45. Considering the network contains thousands of devices, this is considered to be a small number. Simulations completed in less than five seconds for most of the typical combination of parameters. This will allow the optimization to run hundreds or thousands of simulations within an acceptable time. Even if it takes a few hours, compared to the manual process of adjusting parameters and running each simulation, the duration is still saving a lot of time. More sophisticated thread management was also applied to the optimization in order to increase efficiency, but still faced instability. This approach can be resumed again later.

One important problem found was regarding the memory usage of the program. C# and the .NET Framework uses a Garbage Collector to manage all the memory allocation, so the user does not need to manually implement allocation and desallocation methods or calls. Even then, with the multiple simulations performed

either in the surface generation method or the optimization a continuous increase in the use of virtual memory was noted. At a certain point, the program reaches a limit specified by the operating system and crashes.

Right now this limit is above 1 thousand simulations, which is enough for all optimization tests done. But as the number of parameters selected to be optimized increase, the number of simulations grow and this limit might be passed. The new .NET Framework 4.5 expected to be release the next year already support extended memory usage. A better solution of course would be to fix the memory leak, but that is not such an easy task in this environment as the .NET Framework does not allow the user to manually control this aspect.

6.3: Optimization Strategies

As described in previous sections, the two methods implemented behaved very differently, as expected. Their application should also be done in order to take advantage of their characteristics. The hill climbing converges very fast to a hill, but does not guarantee that it is the global maximum. One way to take advantage of it is to perform a series of optimization for the same parameters, saving the best results of each and changing only the starting value in order to find different maximums.

The genetic algorithm is great to optimize the entire surface with just one execution. That will give a higher significance that the best result it finds in the end is indeed the global one. To explore more these characteristics a few changes in the parameters of both methods were done to evaluate their influence.

6.3.1: Hill Climbing

In order to increase the speed of the method, the starting range was lowered from 80% to 20%. This will create initial points more close together and make them reach the conversion faster. Also, the shrinking rate when the middle candidate is identified as the best one was reduced from 0.75 to 0.25. This will make the spread from one iteration to the other decrease much faster. The new optimization behavior can be seen in Figure 40.

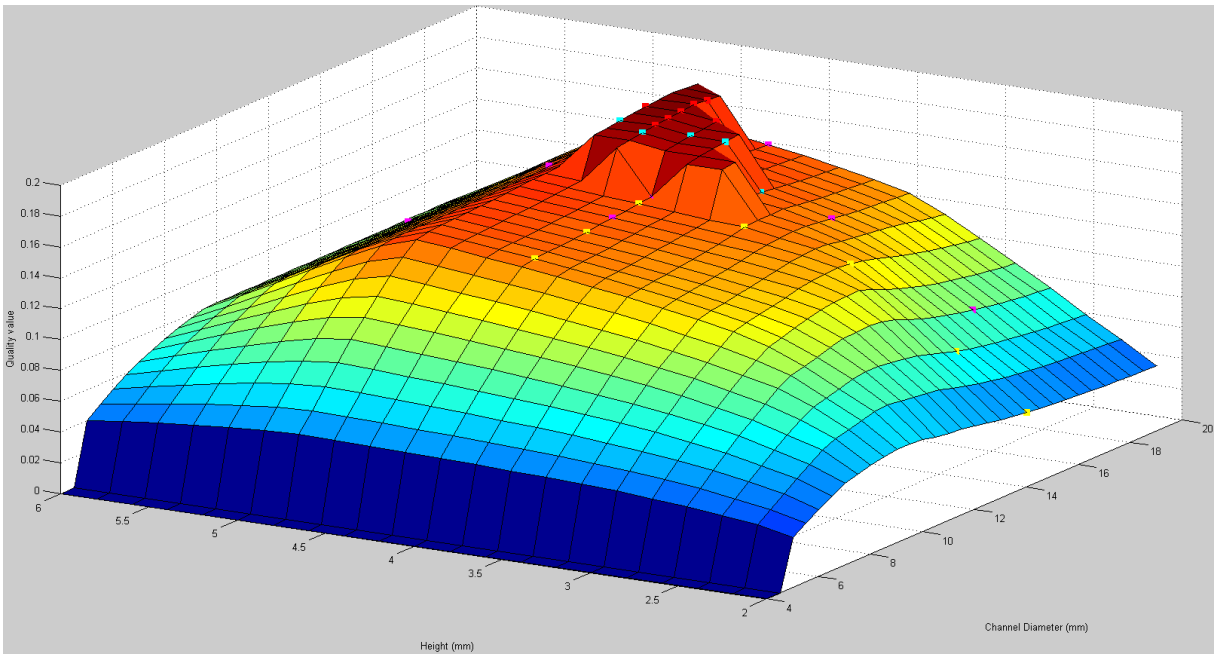

Figure 40 - Faster Hill Climbing Optimization.

It's noticeable that there are much less points than the first HC optimization shown. For the same surface and starting point this one converged in only 4 iterations, against 10 of the first one. This is of course a solution with a lower chance of finding the global maximum. But its use multiple times with different starting points may find the global maximum faster.

6.3.2: Genetic

Two additional GA optimizations were performed to examine the influence of the mutation and the crossover balance rates. The results can be seen in Figure 41.

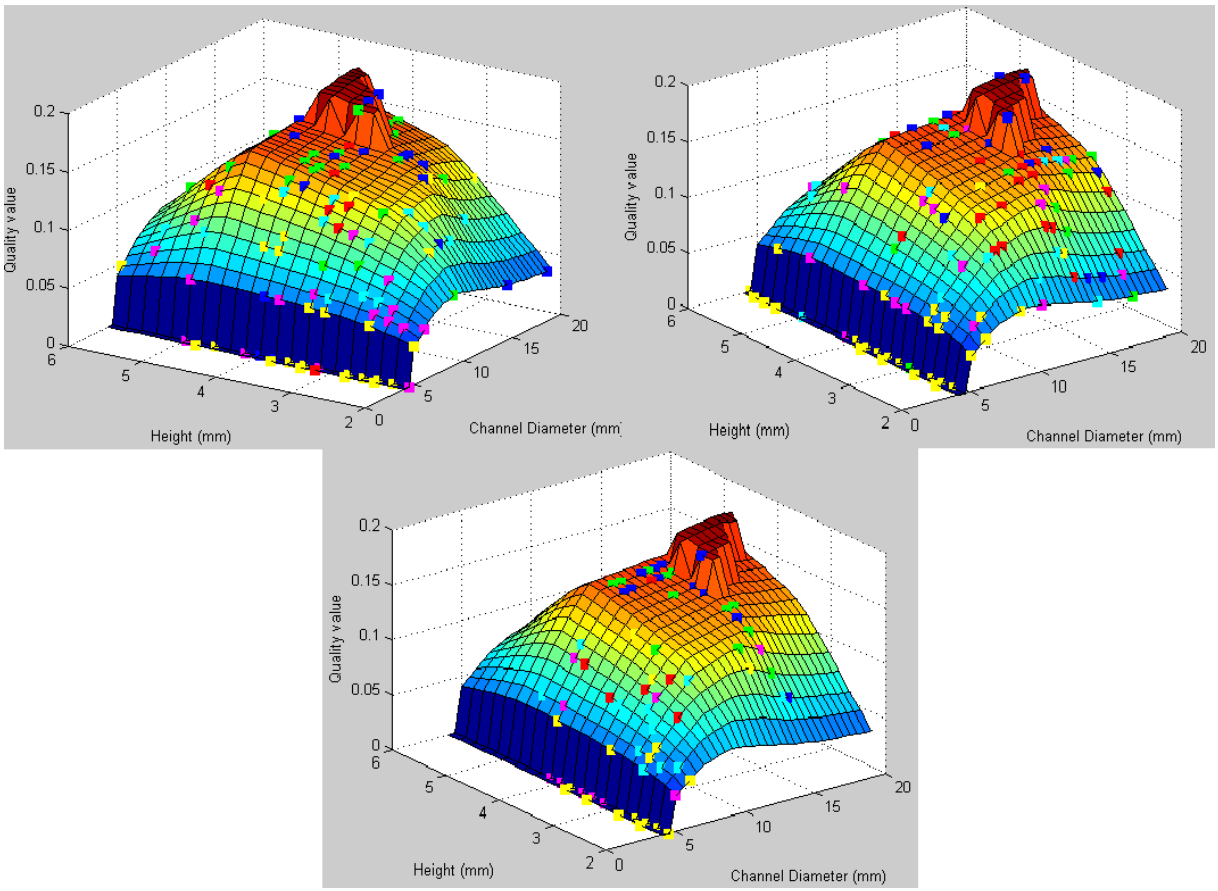

Figure 41 - GA Optimization comparison.

The first one on the upper left corner is the one already shown earlier, with rates of 0.5 for mutation and cross over balance. The one to the right has 0.75 of mutation and only 0.1 of crossover. It's visible that the points from each iteration are more spread throughout the surface. The lower one has opposite settings, 0.1 for mutation and 0.75 for crossover. The difference in this case is very big. The populations of each iteration (points with the same color) are much more close to each other and therefore there are larger areas not covered by any point. Since the goal with this method is to cover more parts of the surface as possible, the second combination of parameters seems more appropriate.

Chapter 7. Conclusions and perspectives

The results achieved so far shows that the project is working towards a successful accomplishment of the objectives set. The decisions regarding core aspects such as the mathematical model, solving method, GUI, optimization strategies and software architecture proved to match the requirements.

The electric circuit model was implemented and tested, although it's still not complete because a second model had to be made to replace the one with the transition line which presented higher than expected errors. The algorithm was able to correctly generate the matrices system of equations. The numbering patterns for all the devices, nodes and links were successfully identified and implemented.

The solver proved to be fast and reliable as desired. It was implemented in C, compiled into a .dll file and imported to the program. The integration required the use of the CSR format for representing the matrices, which also contributed to keep the memory usage as low as possible. The memory leakage is a problem that is very important to be solved since it can limit the software capabilities in the future. Although multiple optimizations could be performed to accomplish the task, it increases the need of an operator and may not produce same results as there are methods with random factors.

The two optimization strategies implemented, Hill Climbing and Genetic, showed good results. Changes to their behavior and the way they're used can still be further explored. Combination of the two strategies is also a possible approach that could take advantage of the good characteristics of both at the same time. The implementation of other parameters and more strategies are also desired.

The software architecture and GUI design are working as intended and contributing to the good usability the program achieved.

At the time of the writing of this report, the implementation of the second model of the network was interrupted due to lack of time in the schedule. It is vital to the project that this model is finished later, as it will bring more accurate results.

Another desired increment to the software is the support for other geometries of tools. There are different shapes of manifolds that were shown, different channel

geometries, approximation methods that are used in the industry and would be welcomed features to make the program more broadly used.

Bibliography:

[13] Germany Trade and Invest, The Plastics Industry in Germany, Issue 2010/2011, available at http://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Industry-overviews/industry-overview-plastics-industry-in-germany.pdf, accessed in July 2012

[14] The Freedonia Group, World Plastics Processing Machinery, available at http://www.freedoniagroup.com/brochure/23xx/2381smwe.pdf, accessed in July 2012

[15] CHHABRA, R.P.; Non-Newtonian Fluids: An Introduction, available at http://www.physics.iitm.ac.in/~compflu/Lect-notes/chhabra.pdf, accessed in July 2012

[16] The All India Plastics Manufacturers' Association, available at http://www.aipma.net/images/pp17.gif, accessed in July 2012

[17] KÖPPLMAYR, T. & AIGNER, M & MIETHLINGER J; A comparative study of viscous flow in slit-exit cross section dies using network analysis, Journal of Plastics Technology, 2012, 8 (1), 53-74

[18] MICHAELI, Walter; Extrusion Dies for Plastics and Rubber, Hanser, 3rd revised Edition

[19] OSSWALD, Tim A. & MENGES, Georg; Materials Science of Polymers for Engineers, Hanser, 2nd Edition

[20] OSSWALD & BAUR & BRINKMANN & OBERBACH & SCHMACHTENBERG; International Plastics Handbook, Hanser, 4th Edition

[21] SAGER, David J. Electronic Circuit Analysis, available at http://home.comcast.net/~stager21/Circuits.html, accessed in October 2011

[22] .NET Framework Conceptual Overview, available at http://msdn.microsoft.com/en-us/library/zw4w595w.aspx, accessed in July 2012

[23] Agile Modeling, Introduction to the Diagrams of UML 2.0, available at http://www.agilemodeling.com/essays/umlDiagrams.html, accessed in July 2012

[24] SADAYAPPAN, P. Solution of Sparse Linear Systems, available at http://www.cse.ohio-state.edu/~saday/894/lec5/img007.gif, accessed in July 2012

[25] HOPMANN, C. & EILBRACHT, S.; Towards automatic optimization of flow channel geometries in complex multi-level dies for film extrusion, EngOpt 2012 – 3rd International Conference on Engineering Optimization